立体映像配信に関する画像品質評価法及びシステム構成技術に関する研究

Research on the Technologies for Picture Quality Evaluation and System Architecture for Three-Dimensional Image Distribution

指導教官　安田　浩　教授

妹尾　孝憲

東京大学　先端科学技術研究センター

特任教員

2006 年 12 月 1 日

■ 目 次 ■

■　図表目次　■

■ 第 1 章 ■

序論

1.1 映像技術の進化と課題
1.2 立体映像技術の現状と課題
1.3 本研究の目標と目的
1.4 立体映像評価技術の目標と目的
1.5 立体映像生成技術の目標と目的
1.6 立体映像符号化技術の目標と目的
1.7 本論文の構成

第1章　序論

「百聞は一見に如かず」の諺がすべてを語る様に、情報伝達手段としての映像の利用は最も有効な手段と思われ、太古の壁画や地上絵に始まり、1897 年ドイツで発明されたブラウン管に端を発する電子的な映像情報伝達技術は、時空の壁を越えて全ての人々に情報共有と相互コミュニケーションへの恩恵をもたらして来た。

これまで、情報伝達手段としての理想映像を求めて、人々が共有したいと思った光景をそっくりそのまま相手に伝えられる様に、電子映像の高精細化や伝送効率の改善、平面映像の限界を打破する立体映像化の試みなどが精力的に行われて来たが、未だ実物と同レベルの映像品質までには至っていない様に思われる。

そこで本論文では、提示された映像世界を現実の世界と感じられる様な、理想的な映像世界を効率良く配信する為に必要な技術として、３次元立体映像の入出力系、符号化伝送系、表示された映像品質の評価方法に関する提案と検討を行う。

1.1 映像技術の進化と課題

映像情報伝達手段としてのマルチメディアコンテンツの符号化・伝送技術は日増しに進化している。最初の民生用の画像圧縮技術は、ISO（国際標準化機構）傘下の MPEG（Moving Picture Expert Group）で 1990 年に草案が作成された MPEG-1 規格[1]と思われるが、その後、1995 年に MPEG-2[2]、1999 年に MPEG-4[3]と新規格が作成される度に、その符号化効率は3～6dBづつ改善されており、2003年に標準化されたMPEG-4 AVC 規格（Advanced Video Coding、ITU-T が制定した H.264 規格と同じもの)[4]では、MPEG-1 の約 4 倍の高画質を同じビットレートで実現することが出来るなど、高精細度映像の高能率伝送／蓄積による高臨場感再生が可能になりつつある。また、映像の解像度も、MPEG-1 の 360 画素×240 ラインから、MPEG-2、MPEG-4 での 720 画素×480 ラインの標準テレビジョンサイズや、1920 画素×1080 ラインの HDTV サイズでの撮影・表示システムが普及し、米国を中心とした 4096 画素×2160 ラインのデジタルシネマ規格の制定や、我国では更に大画面の 7680 画素×4320 ラインのスーパーハイビジョンの開発も始まっている等、高画質化・大画面化・高解像度化の追求が休みなく続けられている。しかしながら、平面映像の解像度や符号化効率の改善のみでは、現実世界を直接見る様な臨場感は得難いと言う課題がある。

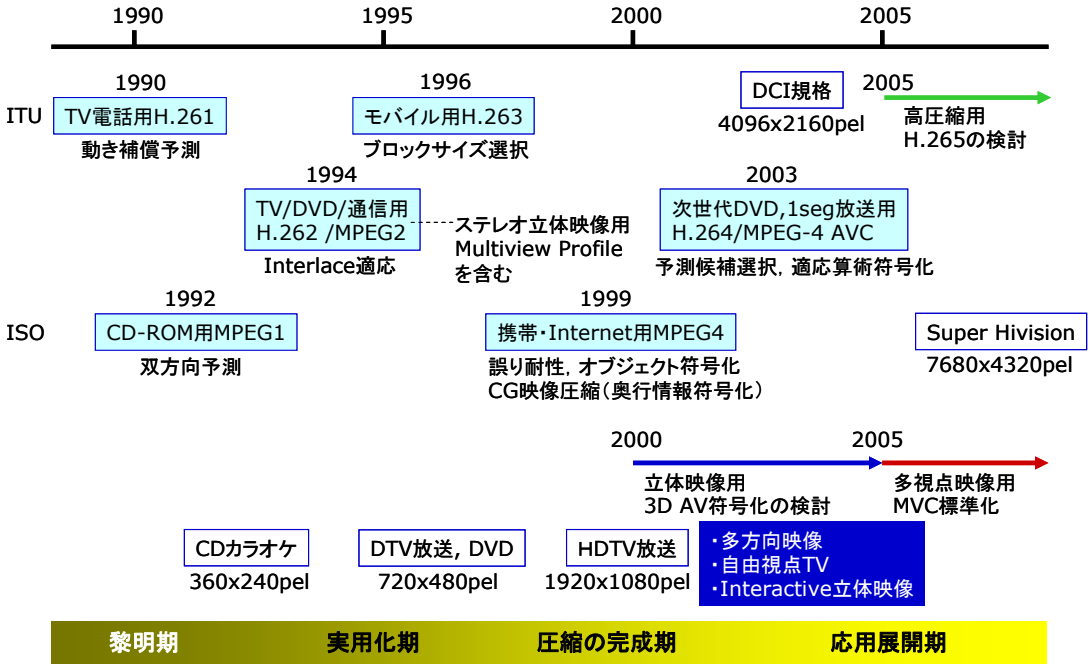


図 1- 1　近年の映像技術の進化

この問題を解決する為には、人が 3 次元空間を見た時に知覚する全ての視覚情報を伝達する必要があり、これを実現する手段として、映像の立体化の取り組みが精力的に進められ、各種の立体映像方式が提案されている。２眼式立体映像は、立体視要因の一部しか満たさない為、眼精疲労の問題などが指摘されており、ホログラフィによる光線再生法は膨大なデータ量の為、動画再生が困難である。奥行標本化法や多視点映像による立体映像は立体視要因をほぼ満たすが、映像の入出力・処理に用いるハードウエアの制約で未だ実用化の域に達していない。

1.2　立体映像技術の現状と課題

(a)　表示技術

立体視の主要因の一つは、左右の目の位置が異なる事により、両網膜に写る被写体映像の位置の僅かなずれから生じる両眼立体視である事が知られている[5]。立体映像の実用化取り組みは、1922 年に米国で、左右レンズの色が赤青と異なるアナグリフ眼鏡を用いて、左右眼で異なる２つの映像を看視するステレオ立体映画の試みが開始されているが、カラーの再現性が悪く、赤青という補色による左右視覚の視野闘争の問題があり、

長続きはしなかった。

その後、1932 年に偏光フィルターが実用化されると、これを用いたカラーのステレオ立体映画ブームが 1937 年から始まった。立体を強調する為に大きな輻輳角を付ける事が行われたが、焦点距離が変化しない為、看視者に眼精疲労が生じ、これも長続きしなかった。その他、左右レンズに濃度差の付いた眼鏡を通して平面運動をする映像を見ると、暗い映像が脳で認識されるまでの時間遅れにより左右眼で運動視差が生じ、奥行が知覚される現象が Pulfrich 効果[5]として知られており、この原理を利用して平面映像の動きに応じて片方の目に時間的に遅れた映像を提示する立体テレビが試作されたが、やはり立体視要因の一部を使うものであり、普及には至らなかった。

その後、レンチキュラーレンズを用いた立体映像や、ホログラムを用いた立体映像、視差バリヤーを用いた裸眼立体ディスプレイ等の出現により、5 年から 10 年の周期で立体映像ブームが繰り返して現在に至っている[6]。

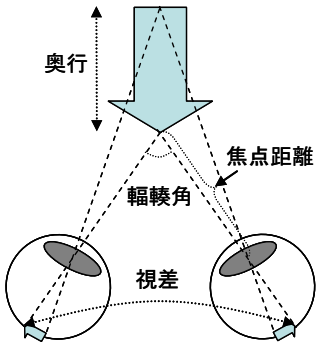


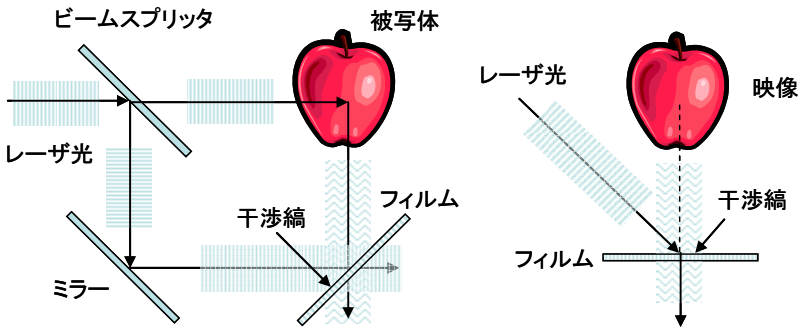
図 1- 2　両眼視差による奥行知覚

図 1- 3　アナグリフ立体映像の例 [6]

上述した MPEG 標準化組織においても、1994 年草案の MPEG-2 規格において、両

眼ステレオ映像による立体映像符号化方式の標準が、Multiview Profile として制定されたが、撮像系や表示系などの周辺技術を含めた総合システムの実現コストが高い割に、輻輳角と焦点調節の乖離の問題が解決されておらず、実用化までには至らなかった。MPEG-4 規格制定後の 2000 年以来、再度 3D AV 符号化方式の標準化検討が開始され、産業界の要請の有無や、実用的な要素技術の有無の調査が慎重に行われた結果、従来のステレオ立体映像の持つ課題を解決すると共に、視聴者が自由に視点を変えられる自由視点テレビへの展開を可能にするマルチビュー映像符号化方式は、カメラ間の相関を利用する等の従来の標準ではカバーされない技術を必要とすることから、新たな標準化が 2005 年から開始された。しかしながら MPEG では、多数の視点から得られた映像の相関を利用した圧縮方式の標準化のみをターゲットとしており、これらの映像からどの様にして立体映像を作り出すかと言う課題は、検討対象にしていない[7]。

従来のステレオ立体映像が持つ輻輳と調節の乖離問題を解決する為には、被写体から出た全ての方向の光線を再現する必要がある。その手段の一つとして、対象物から出た光線の強さと方向を、参照光との干渉縞として記録再生するホログラフィ方式[8]がある。この方式は、撮影されたホログラムに参照光を当てる事により干渉光が再現され、この干渉光が元の被写体から出た光の方向も含めて再生するので輻輳と焦点調節の不整合はなく、自然な立体映像を表示可能であるが、干渉縞のサイズは光の波長（0.5 ミクロン程度）のオーダーである為、そのデータ量が膨大であり、現在のハードウエア処理能力では実写動画の実現が困難な事や、遠景の撮影や対象物の色の再現等に課題がある。



(a) 記録系　　(b) 再生系

図 1- 4　ホログラフィーによる立体映像

微小レンズをアレー状に配列したレンズ板を通して撮影した多数の微小映像を同じレンズ板を通して看視するインテグラルフォトグラフィー方式[9]は、被写体から出る光の方向に応じて焦点を結ぶ位置が異なり、これを再投影する事により像位置に応じた光線方向が再現され、この光線を見る方向を変えると像の見える面が変化する光線再生方式の立体映像方式であるが、レンズ数が再生画像の画素数に相当する為、多数のレンズ

を必要とし、解像度を上げるのが困難と言う課題と、映像面がレンズの焦点距離の位置にある為、被写体が近づくとボケる課題があり、これを光学系と信号処理の工夫で改善する検討が続けられている[10]。この微小レンズをシリンドリカルレンズにする事により、微小映像から垂直視差を無くして、微小映像の各画素並びをレンズの円筒軸から少し傾ける事により、垂直方向にも水平視差の異なる画素を配置して非常に多数の水平視差を持った微小映像を構成して、これをレンズアレーで看視する高密度指向性画像による立体映像方式は[11]、光線再生法の一つであるが、一本一本の十分細く絞った多数の指向性光線を空中に投影する事により、映像位置に焦点を合わせる事が可能となり、自然な立体映像を表示出来るが、指向性画像数は数 10 から 100 のオーダーが必要な為、大画面化と高解像度化に限界がある。

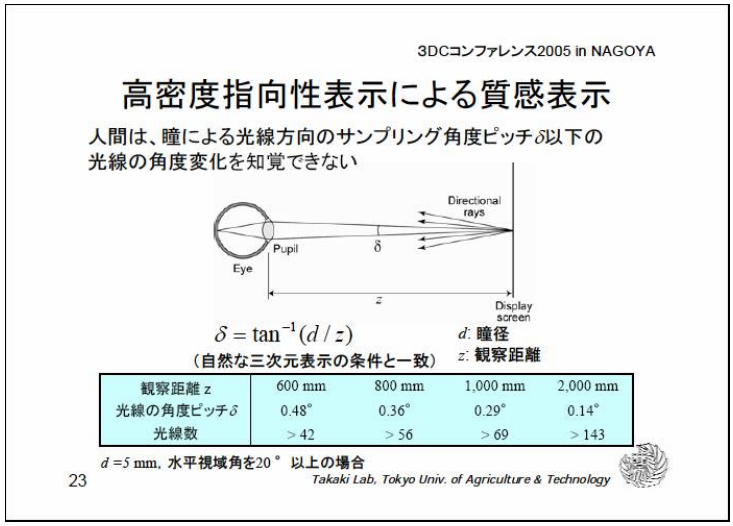


図 1- 5　高密度指向性表示 [39]

かつて提案されたバリフォーカルミラーによる奥行標本化方式[12]は、動画の為には高速に機械振動する反射鏡がボトルネックとなる点や、深い奥行の再生が困難である等の課題があるが、真に奥行のある立体映像を表示可能で、輻輳と調節の問題を解決する。この方式において、機械振動をなくす手段の 1 つとして、可変焦点レンズがある[17][18][19]]。文献[17]は、液晶レンズに２つの異なる周波数の電界を交互に掛けて、液晶の配向を変える事によりレンズの屈折率を変えて多層表示を行うものであるが、１つの表示パネルで多数の奥行の異なる映像を表示する必要がある為、高速な表示パネルを必要とする事と、屈折率は常に変化し固定出来ないので、１枚の映像の中でもその描画時間の違いによって奥行位置が異なる課題がある。文献[18][19]は、水と油の界面形状を電圧で制御して可変焦点レンズを形成するが、重力の影響の為、大口径にする事が困難で、立体シネマ等の大画面用途には不向きと思われる。

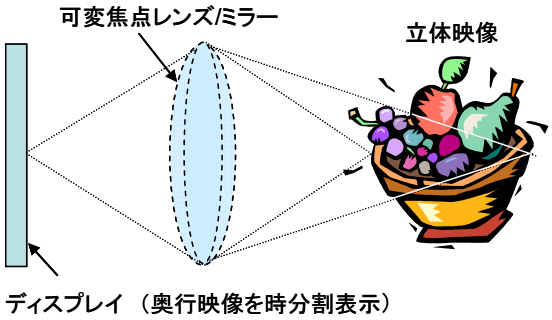


図 1- 6　可変焦点レンズ／ミラーによる立体映像表示

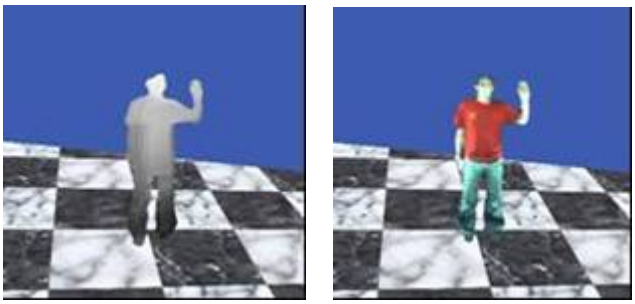
いずれの方式においても、立体映像のデータ量は膨大になるので、立体視要因を満たしつつよりデータ量の少ない立体映像方式の開発と、映像データの高能率圧縮技術の開発が、実用化には不可欠である。

(b)　符号化技術

前節で述べた通り、理想に近い立体映像は、多数の焦点位置や視点位置から得られた異なる映像を合成表示する事で得られるので、この多数の映像を効率良く圧縮符号化する事が、立体映像の記録や伝送に必要不可欠であり、この技術の進化なくしては立体映像の普及はおぼつかない。両眼ステレオ映像を階層符号化する方式は、MPEG-2 規格[2]にも定められている事は先に述べたが、一般的に検討されている階層化は、左右映像をブロックに分割し、片方の映像をベースレーヤとし、左右映像で対応するブロックの相対位置のズレを動きベクトルで現して残りの映像を予測符号化するものであるが、左右映像の視点位置が異なる事や、左右カメラの特性ばらつきの為、単なるブロックの平行移動では予測誤差が多く、効率の良い予測が出来ない。現在進行中の MPEG-4 AVC 拡張規格における多視差映像符号化（MVC: Multiview Video Coding）の標準化検討に於いても、各種の検討が続いているが、カメラ間の符号化効率の悪さを時間軸方向の階層予測符号化でカバーする試み等が報告されているが[20]、多視点映像間の巧妙な相関利用技術開発はこれからの課題である。

図 1- 7　ブロックの平行移動による左右カメラ映像間予測［7］

マシンビジョンやコンピュータグラフィックスの分野でも、実写映像からコンピュータ上に三次元オブジェクトを構成し、これを任意視点からの平面映像に射影して自由視点映像を得る試みが検討されている[21]。ここでも課題は、正確な３次元モデルの構成とその３次元データを如何に圧縮するかであり、複数カメラのシルエット映像間のマッチングを行って、カメラ映像と被写体位置の間に成立する射影方程式を解き、被写体の三次元形状データを得て、これに特定の視点からの映像を貼り付けたもの (Image-based Visual Hulls)や、３次元形状をポリゴン等の多面体で近似して、各面の傾きを考慮して映像をテキスチャとしてマッピングする方法（Polyhedral Visual Hulls）や、三次元データを複雑さに応じて断片化して、各断片に色と面の法線方向情報を持たせる事により、立体映像を不均一に近似する方法（3D Video Fragments）や、規則的に並べられた３次元画素 (voxel) を 2 値化して形状のみを表し、別途色情報を持つ方法 (Volumetric Visual Hulls）等が、検討されている。いずれの方法も、多視点映像から三次元形状を得る為の処理が重く、実時間で動かす場合には解像度が上げられないと言う課題がある。



(a) ３次元形状情報　　(b)色情報を追加

図 1- 8　3D CG 技術を用いた三次元映像の例 [21]

(c)　品質評価技術

従来の平面映像の客観的な品質評価法は、信号対雑音比（SNR）を用いて、再生映像を原画と比較して品質を評価する事が一般的であった。

$$SNR = 10\log_{10}(S/N) \tag{1.1}$$

ここで、S は平均信号電力、N は平均雑音電力である。映像の世界では信号のピークレベルに注視点が集まる事が多く、そのピーク電力と平均雑音電力を比較する PSNR（Peak Signal-to-Noise ratio）で映像の品質を示す事が一般的に行われている。

$$PSNR = 10\log_{10}(255^2/N) \tag{1.2}$$

PSNR は、映像の符号化・再生システム中での信号劣化を評価するには適しており、立体映像を構成する一手段であるマルチビュー映像符号化方式の評価[7]にも用いられているが、立体映像の観点からのパラメータは入っていない。

立体映像の品質評価に関する研究では、視野の大きさ、解像度、立体感、双方向性、異種感覚協調などが、臨場感に影響するとの報告[22]はあるが、画質や総合品質のレベルとの関連性には触れていない。又、平面映像と立体映像の主観値を因子分析して、８つの心理要因を抽出した報告[23]では、立体映像の場合、客観値に基づかない主観的因子得点では総合評価値を推定不可能な事が示されており、定量的な総合評価方法は確立されていない。

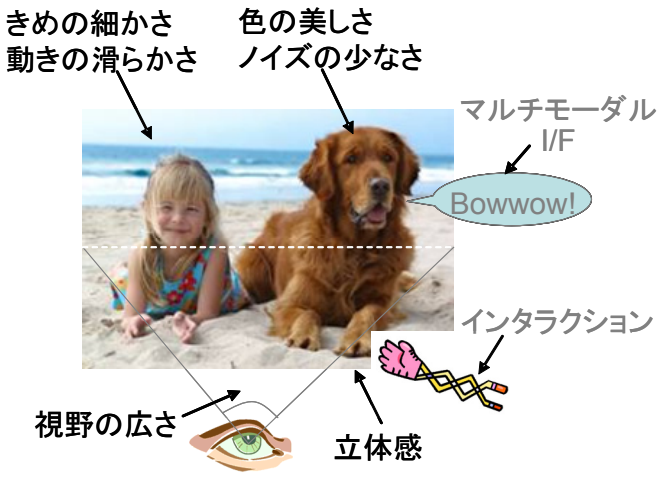


図 1- 9　立体映像の臨場感度要因

1.3 本研究の目標と目的

立体映像配信の実用化取り組みは様々な角度から進められているが、上述した様に未だ課題が多い。そこで本研究の目標は、

「実物の様に見える高品位な立体映像サービスをコストパフォーマンス良く配信する為の要素技術の確立」を目指すものとする。

この目標は、先ず現状の立体映像の品質を客観的・総合的に評価して、理想に近づける為には何が必要かを明確にして、この要件をコストパフォーマンス良く満たす立体映像方式を考案し、更に膨大な立体映像データを効率良く伝送・蓄積する方法の確立により、その達成が加速されると思われる。従って、本研究の目的は、

１．立体映像の総合品質を、あらゆるパラメータを考慮しながら、主観値と整合して定量的かつ客観的に計測する技術を確立して現状の立体映像の品質評価と解析を行い、

２．この結果から明確になる立体視要因を満たす、理想的な立体映像を、コストパフォーマンス良く生成する技術を確立し、
２．それによって発生する大量の映像データを、視覚特性や映像間の相関を利用して、効率良く高画質に圧縮符号化する技術の確立、

とする。これらの目的とする技術の確立を通じて、理想的な立体映像配信システムの実現に貢献する。

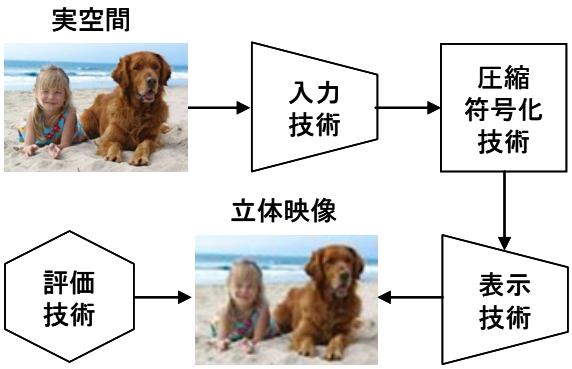


図 1- 10　理想的な立体映像を配信する技術群

1.4 立体映像評価技術の目標と目的

前節で述べた様に、立体映像の総合品質を客観的・定量的に評価する方法や、主観値項目から総合品質の評価値を得る方法は未だ確立されていない。そこで、本論文で目指す立体映像評価技術の目標は、

「現状の立体映像の品質を総合的に評価して、理想立体映像の要件を明確にする為に、コンテンツの違いや看視者のばらつきを排除して、立体映像の総合品質を客観的・総合的に評価出来る評価方法の確立」

とし、その目標達成の為に、

１．客観的評価尺度としての臨場感度を定義し、その客観値を映像の物理パラメータから求める手法を確立すると共に、
２．提示された映像から受ける主観評価項目の得点から、主観的総合品質評価値を得る手法の確立、及び
３．この様にして得られた客観値と主観値の整合化を図る方法の確立を行い、
４．この評価法を用いて各種の立体映像の主観的・客観的総合品質を計測して、理想立体映像に近づける為には何が有効かを明確にする

事を目的とする。

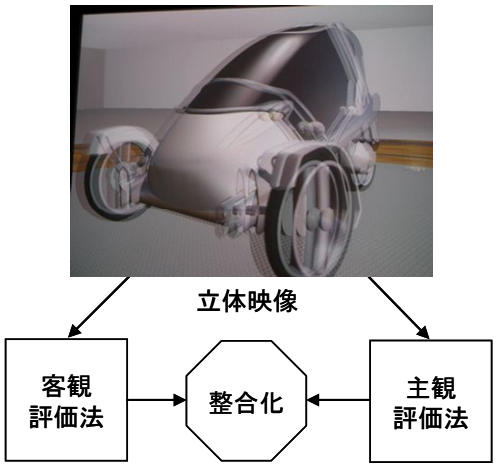


図 1- 11　立体映像の総合品質評価

1.5 立体映像生成技術の目標と目的

先述の評価結果より明らかになる、理想立体映像の配信に必要な要件をふまえて、本論文で目指す立体映像生成技術の目標は、

「実空間から得られる視覚情報の全てを効率良く取得・再生出来る方式の実現」

に置き、この目標を達成する手段として、

１．立体視の５大要因である、両眼視差、輻輳角、焦点調節、運動視差、画像要因の全てを効率良く取得・再生出来る方式の考案、

２．この方式を実現する為の映像入出力デバイスと信号処理方式の検討、及び

３．この検討に基づいた画像処理実験による実現可能性の検証と課題の明確化

を目的とする。５大要因の内、運動視差は、看視位置を変える事により映像の見え方が自然に変わる事であるが、これを完全に実現する為には、実空間の３次元方向の光を全て取得しなければならず、効率が悪い。通常の映像看視では看視者があまり大きく動き回る事は少ないので、ここではカメラが動く事により画像要因として等価的に映像に組み込まれる運動視差の補完効果を考慮に入れて、コストパフォーマンスの良い立体映像の実現を目指す。又、理想的な映像である為には、立体感要因の確保のみでは不十分で、大画面化・高画質化が容易に実現出来る方式を目指す。

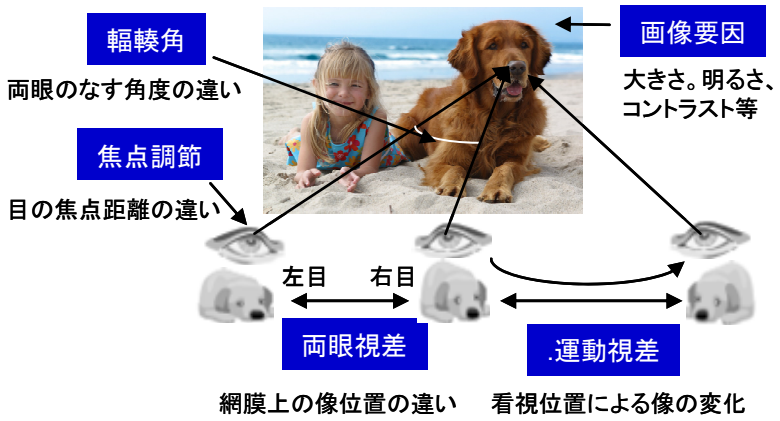


図 1- 12　立体視の５大要因

1.6 立体映像符号化技術の目標と目的

上記の検討結果から明らかになる、大きな運動視差を持った多視点立体映像配信を実用化する際に課題となる膨大な映像データ量の削減の為に、本論文で目指す立体映像符号化技術の目標は、

「膨大なデータ量になる多視点立体映像データを、映像の持つ時空間相関を効率良く利用して、高能率に圧縮符号化する技術の実現」

とし、その具現化手段として、

１．多視点映像の相関を、被写体面の向きに着目して射影幾何学的に解析し、映像間に存在する規則性を抽出して、

２．その規則性を利用した多視点映像の視差補償予測符号化方式を考案し、

３．その方式を実際の多視点映像に適用して有効性を確認する

事を目的とする。多視点映像の符号化は、現在その技術が広く研究されており、自由視点映像の実現にも容易に適用出来るものであり、今後の展開が期待されている。

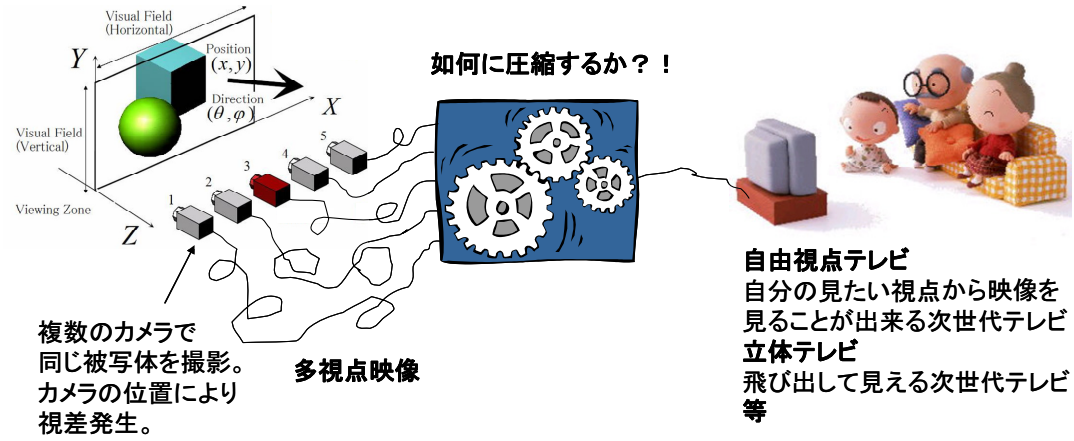


図 1- 13　多視点映像の圧縮［55］

1.7 本論文の構成

以上より、本論文の構成は、上記課題の整理と解決案の提案および、その実験検討結果をまとめたものであり、第 1 章は、「序論」とし、立体映像の背景と現状及び課題に言及し、本研究の目標および目的、理想的な立体映像を効率良く配信する為に必要な要素技術の提案と研究課題について述べ、本論文の構成を示す。

第 2 章は、「研究の背景と関連研究」と題し、まず立体映像生成技術に関して、両眼ステレオ映像・多眼立体映像の研究、ホログラフィによる立体映像の研究、奥行標本化による立体映像の研究等をレビューし、現状と課題を述べる。次に、立体映像符号化技術に関して、MPEG-2 マルチビュープロファイル、光線空間法、3 次元モデルベース符号化方式等に関して、現状と課題を述べる。更に、立体映像の評価技術に関して、ステレオ映像の画質評価の研究、立体映像の感性・心理的側面評価の研究、大画面映像の臨場感・没入感評価の研究をサーベイし、その現状と課題を明らかにする。

第 3 章は、「立体映像配信に関する画像品質評価法及びシステム構成技術の概要」と題し、本論文で検討するテーマの概要を説明する。先ず第 1 のテーマは、理想的な立体映像の要件を整理し、現状の立体映像がどの程度その要件を満たしているかを主観的・客観的に計測する手法を確立すると共に、計測結果から理想立体映像の実現に必要な要件を明確に事である。第 2 のテーマは、本論文で提案する計測手法による実験結果から示される、理想立体映像の要件を満たす立体映像を効率良く生成する方法を考案する事である。第 3 のテーマは、本論文で提案する立体映像生成方式としての空間標本化法の検証実験を通じて明らかになる、真に理想的な立体映像実現に不可欠な、多視点立体映像に伴う課題である、膨大なデータ量の映像を高精度かつ効率良く圧縮・符号化する方

式の確立である。本章では、これらのテーマを解決する方法の提案とその有効性を、検証実験を通じて明らかにしてゆく事を述べる。

第４章は、「３次元立体映像の品質評技術」と題し、理想的な立体映像を目指す為の品質評価方法として、立体映像から得られる物理パラメータの積からなる評価尺度を新たに定義し、立体映像の総合品質評価に用いる事を提案する。この評価尺度の特徴は、看視者のばらつきや、映像コンテンツに依存する偏りを避け、定量的・普遍的な評価尺度を提供出来る事であり、その為の評価パラメータの組み合わせ方と、評価式の導き方を提案し、マルチ画面立体 CG 画像、大画面立体 CG 動画、大画面立体アニメ、大画面立体実写映像等をそれぞれ計測した結果を主観評価結果と照合してその整合性を検証し、映像が非実写の合成映像で、映像コンテンツの影響を除去しきれない場合は、コンテンツ依存パラメータとして映像の本物らしさ係数の導入が有効である事を示すと共に、立体映像の評価結果から、理想立体映像に近づける為には、画素数の増加と立体感要因を全て満たす立体方式が必要不可欠である事を示す。

第 5 章は、「３次元立体映像の生成技術」と題し、前章の立体映像評価結果を踏まえて、立体感要因を全て満たす理想的な立体映像を効率良く生成する方式として、空間標本化法を提案する。この方式の特徴は、三次元の実空間をそのまま三次元の小映像空間に射影してサンプリングするもので、再生される映像は実際に奥行を持った実像として看視出来るので、輻輳と調節の乖離と言った従来の問題が解決され、理想的な立体映像を提供する事が出来る。ここでは、この方式の原理と、実現に必要な多層型立体映像入力デバイス、多層型立体映像表示デバイスの構造について論じると共に、投射型の立体映像実現に必要な屈折スクリーンの構造についての検討を行う。更に、上述した空間映像の標本化によって発生する不要なボケ映像の除去方法を考案し、検証実験を通じてその有効性を確認する。又、多層映像の輝度分配によるアンチエイリアス効果を用いた奥行分解能の増強について実験・考察すると共に、高品質化検討として、オクルージョン問題の考察と、コンピュータグラフィックス映像と実写映像を融合したハイブリッド立体映像の合成方法を論じると共に、十分な運動視差の確保には、マルチカメラによる多視点からの映像取得が必要である事を述べる。

第 6 章は、「３次元立体映像の符号化技術」と題し、5 章の検討結果から必要となる、マルチカメラによる多視点映像の高能率・高画質な圧縮符号化方式として、オブジェクトの面傾斜適応型予測符号化方式を提案する。この方式の特徴は、各カメラへのオブジェクトの射影映像を小ブロックに分割し、カメラの相対位置による透視射影の規則性を利用した映像間のブロックマッチングを行うことにより、高精度の視差補償射影予測を実現する事である。ここでは、本提案方式が、多視差映像間の相関を直接利用して予測符号化する方式の為、オブジェクトの形状を求める必要がなく、多視点映像の圧縮に適する事や、視差情報間の規則性を利用して、高速・高能率に予測が行える事について論じる。更に、上述の多視点立体映像の高画質符号化を実現する、オブジェクトの面傾斜

に適応した視差補償予測符号化方式の確認実験を、片方向予測、双方向予測、ブロックサイズ適応、ブロック形状適応の順に行い、その効果を確認・検討すると共に、予測に用いた視差情報の高能率符号化方法の提案とその検証を行う。

第 7 章は、「結論と今後の課題」と題し、本論文で提案した 3 次元立体映像配信の為の要素技術のまとめを行い、これらの技術が理想立体映像配信に有効である事を述べる。具体的には、提案の立体映像総合品質評価技術は、映像品質を物理パラメータで表すと共に、映像コンテンツの影響を加味する事により、主観評価に整合した計測が出来る事より、今後普及する大画面立体映像の品質改良の指標として有益である事を述べる。又、この様な理想立体映像を実現する方式として提案した、空間標本化法による立体映像生成技術は、自然な実像型立体映像をコストパフォーマンス良く提供する事が可能で、従来の立体映像の課題を解決し、立体 TV などの立体映像の普及に有望と思われる事を述べる。次に、この方式をマルチカメラに適用して、大きな運動視差を持つ立体映像や自由視点映像の配信等のサービスを実現する場合に必要となる、多視点映像の符号化技術として、オブジェクトの面傾きに適応した視差補償射影予測符号化方式を提案し、本方式が高画質符号化に有効である事を述べる。

又、今後の課題としては、提案した立体映像の品質評価法の実用化の為の検討や、理想立体映像を実現する為の、多層型映像入出力デバイスや屈折スクリーンの開発を通じた、立体シネマへの応用展開の為の検討や、視差補償射影予測方式を end-to-end の符号化・復号化システムに適用する為の検討と共に、立体映像に適合するセキュリティや著作権保護技術との組み合わせによる、立体映像のコンテンツ配信への応用等の、将来研究の方向性に付いて述べる。

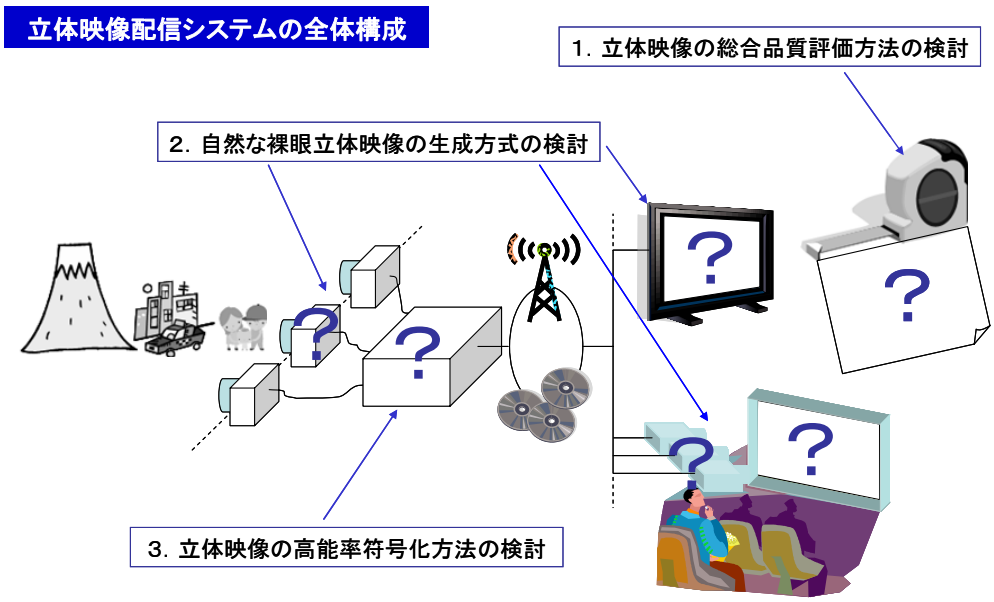


図 1- 14　本研究で取り組んだテーマ

■ 第 2 章 ■

研究の背景と関連研究

2.1　立体映像の生成技術
2.2　立体映像の符号化技術
2.3　立体映像の品質評価技術

第2章　研究の背景と関連研究

視覚による情報伝達・共有は最も効率の良いコミュニケーション手段と思われるが、その為には、実物を直視した時に得られる全ての視覚情報を伝達する必要がある。この目標達成の為には、映像は奥行情報を持つ必要があるが、この様な立体映像の情報伝達に関して、立体映像を生成する各種方式と、立体映像を圧縮符号化する技術と、品質評価技術の現状を以下に詳しく述べ、現状の課題を明確にする。

2.1 立体映像の生成技術

まず立体映像生成技術に関して、両眼ステレオ映像・多眼立体映像の研究、ホログラフィによる立体映像の研究、奥行標本化による立体映像の研究等をレビューし、現状と課題を述べる。図 2-1 は、立体映像表示方式を分類したものであるが[24]、大きく分けて、2 眼式立体映像と 3 次元立体映像がある。

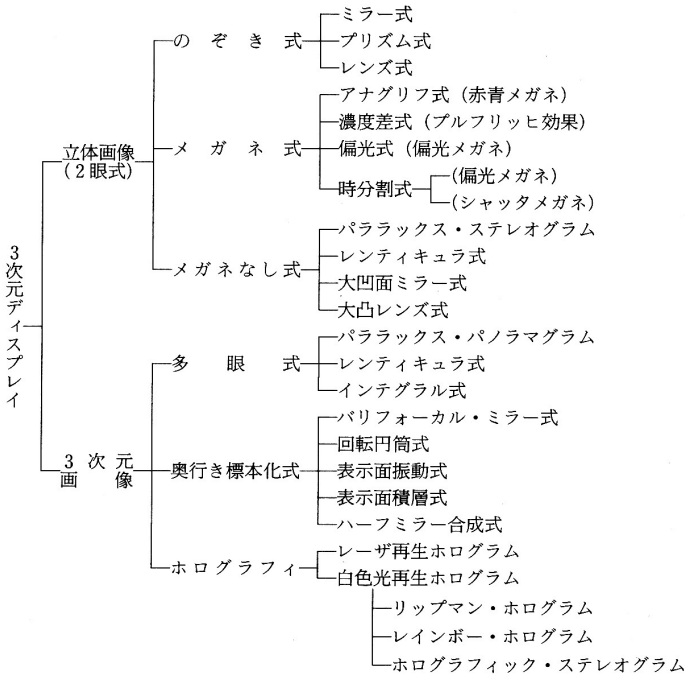


図 2-1　立体映像表示技術の分類 [24]

(a)　2 眼式立体映像

2 眼式立体映像は、1.2 節で述べた様に、左右眼に別々の映像を見せる事により、両眼視差と輻輳角を再現する事で立体視を行う。従来は、左右眼に異なる色フィルタを使用したアナグリフ方式や、異なる濃度のフィルタによる脳の知覚時間の差を利用した Pulfrich 効果型や、偏光眼鏡、シャッターグラス等で左右映像を分離していた。図 2-2 は、液晶ディスプレイの走査線 1 本毎に偏光方向が直交するマイクロポールと呼ばれる偏光パネルを用いた製品である。眼鏡とディスプレイの偏光方向を合わせる為に、直線偏光では看視者の姿勢が制限されるが、円偏光を使う事により姿勢の制約が減らせる。左右映像の表示をラインインターリーブで行うのは、人の視覚の垂直方向の感度が低い事を利用したもので、立体映像を表示しても水平方向の画素数は変わらず、高精細な立体映像が効率良く表示出来る。又、偏光眼鏡を外せば、通常の 2D ディスプレイとして使用出来、用途は広い。課題は、立体視をする時に眼鏡を掛ける必要がある事である。

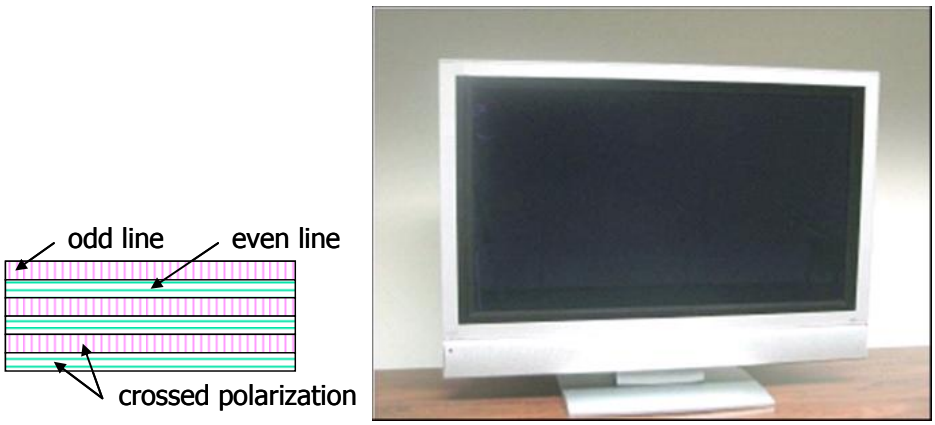


図 2- 2　偏光眼鏡方式の立体ディスプレイ

(http://www.arisawa.co.jp/jp/products/3d.html)

この眼鏡を掛ける負担をなくした、裸眼式の立体ディスプレイが各種考案されている。その原理は、画素毎に水平若しくは垂直に左右画像をインターリーブして配置し、その前面にレンズや視差バリヤーを設ける事により、左右眼が夫々別々に左右の対になる画素を見る様にしたものである。視差バリヤーは、図 2-3 に示す様に液晶パネルの後ろに配置して、バックライトの方向を制御するものや、液晶パネルの前面に配置して、前方から見える画素位置を制限するもの等がある。立体表示をしない時は、液晶パネルからなる視差バリヤーを透明にして、通常の 2D ディスプレイとして使えるが、立体映像の表示時は、左右眼が正しく左右の画素を見られる位置に制約があり、適切な位置で見ないと左右映像が反転したり同時に見えたりして、正常な立体視の出来ない看視領域が存在する。又、ディスプレイに近付きすぎても、遠く離れすぎても、視差バリヤーの間隔

と両眼の間隔が合わず、立体視が出来ない。又、視差バリヤーを ON にするとバックライトの光量が半減し、映像が暗くなると言う課題もある。

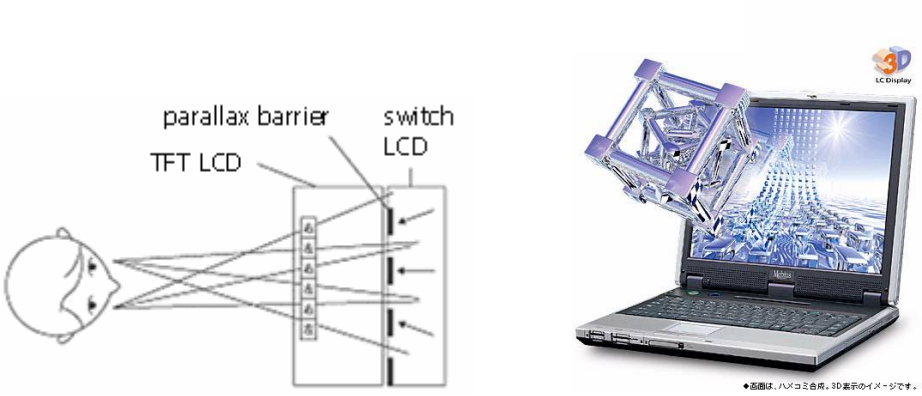


図 2- 3　視差バリヤー方式裸眼立体ディスプレイ

(http://www.sharp.co.jp/products/pcrd13d/index.html)

立体視域を拡大する為に、ディスプレイ上面にステレオカメラを付け、看視者の両眼位置を検出して常に両眼に正しい映像が届く様に視差バリヤ位置を調整するディスプレイも検討されている[25]。このシステムでは更に、看視者の位置に合わせて映像の視点を変える事により運動視差も実現している。又、このカメラを利用して表示映像を看視者のゼスチャで操作する試みもある[26]が、システムが大掛かりになる事と、個人用途に限定される。

立体映像表示時の解像度の半減と輝度の半減の課題を解決する為に、視差バリヤーを用いず、液晶パネルのバックライトを左右眼用に別々の位置から交互に点灯しながら入射させ、レンチキュラレンズとプリズムを組み合わせた両面プリズムシートでバックライトの方向を左右眼用に夫々整形し、それに同期して、左右映像を交互に表示する事で、左右映像の光が夫々左右眼に分離して届く様にしたものの例を図 2-4 に示す[28]。この方式は、時分割表示方式であるが、シャッターグラスや偏光眼鏡を必要とせず、裸眼で立体視が可能であり、映像の解像度低下もなく明るいが、表示周期が半減するので、フリッカーを抑える為には、2D 映像に比べて 2 倍のフレームレートで左右映像を切り替える必要がある。現在の液晶の応答速度は十分早速くこの要求に答えられる。しかし 2 眼式である為、画素の間隔とレンズや視差バリヤーまでの距離に応じて、左右眼がそれぞれ正しい画素を見られる位置の制約がある。

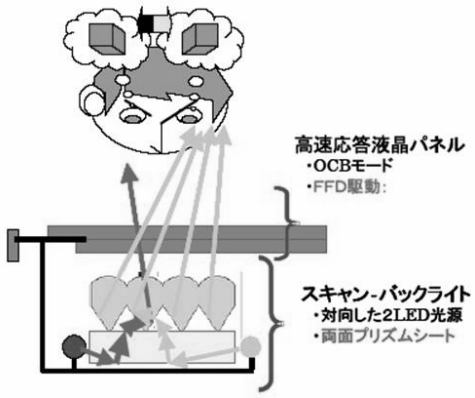


図 2- 4　スキャンバックライト方式 2 眼式立体映像表示方法 [28]

これらの 2 眼式は、焦点調節と運動視差が欠如している為、自然な立体とは異なり、ディスプレイ面に焦点を合わせたまま、両眼の輻輳角のみを変化する立体視に慣れていない看視者には、負担や苦痛を強いる事になり、長時間の使用に伴う眼精疲労や頭痛等が報告されており[29]、望ましい立体映像表示の為のガイドラインが作成されている[30]。余談であるが、一時期、頻繁に明滅する映像を見て幼児が気絶する光感受性発作が起こり、立体視でも同様の障害が懸念されたが、この症状は約 0.5 秒程度の繰り返し周期の強い光の明滅時に起こる事が解明され、立体映像との関連性は否定された。又、輻輳と調節の乖離は、慣れの問題であり、これまで立体視の経験がなくても、自転車に乗る事が覚えられる様に少しの練習を積めば、自然に立体視の能力が身に付く。その為の練習用として、ランダムドットや固定パターンの繰り返し配置を視差に合わせて微妙にずらせた 3D ステレオグラムの絵本が沢山出版されているものが利用出来る。練習を積むと、全く無意味なランダムパターンの中から、突然透明感のある立体映像が浮き出した時の感動は忘れられず、立体映像の普及に大きく貢献していると思われる。

表示映像を多眼式にする事により、眼鏡を不要にする試みも行われている[31]。２眼式の立体映像の課題である運動視差の欠如や立体視の破綻は、多視点映像を使う事で軽減する。図 2-5 は、視差バリヤー方式の 4 視点と 7 視点映像を用いた多視点ディスプレイの試作例である。多視点映像にすることにより、看視位置の移動が可能になり運動視差が獲得される。、視差バリヤーの代わりにレンチキュラーレンズを用いた多眼方式の立体ディスプレイでは、2D 映像の表示が困難であるので、レンズを液晶で構成し、これにかける電圧を変える事で屈折率を変えて、レンズ機能の ON/OFF を切り替える事により、2D/3D 共用のディスプレイの可能性も報告されている[32]。

又、ディスプレイを水平に置いて立体視する事も検討されている[33]。平置きにして映像が正しい形状に見える為には、画像をあらかじめ変形しておかなければならないが、

タッチパネルと重ねたインターラクティブな操作を行い易くなり、ゲーム等の応用が考えられている。

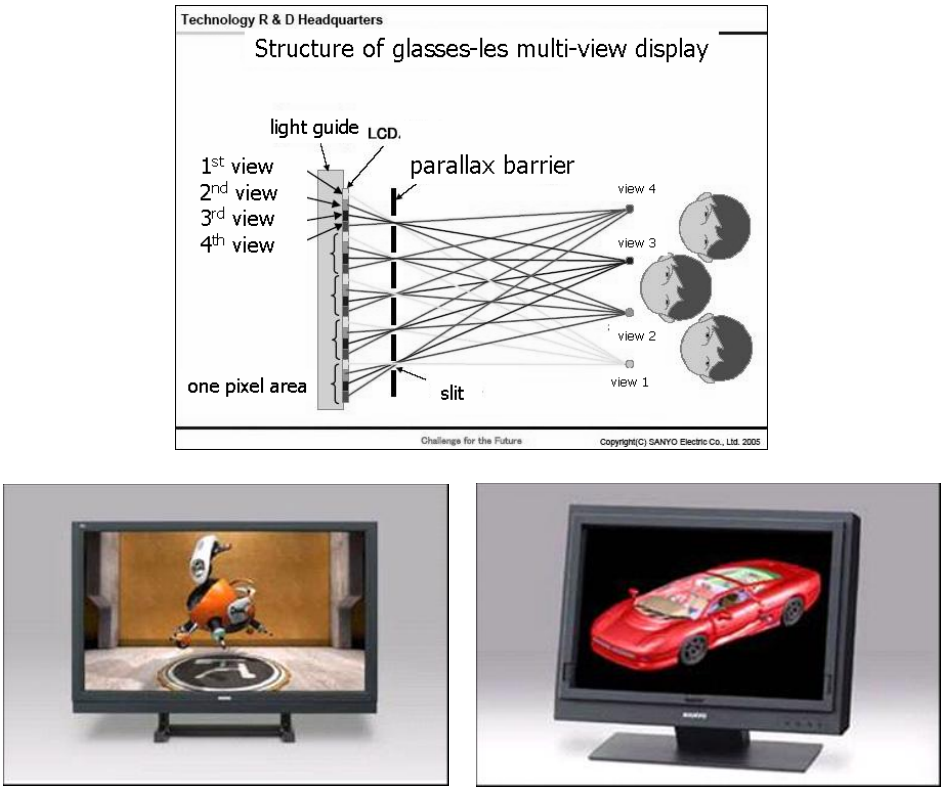


図 2- 5　多視点立体映像ディスプレイ [31]

課題は、視点数を増す程、大きな運動視差を獲得出来るが、データ量が増し、通常のディスプレイでは解像度が下がる事と、焦点調節効果は依然として持たず、看視位置に左右像が反転する領域が残る事である。

(b)　光線再生法

1.2 節で示したホログラフィ方式は、対象物から出た光線の強さと方向を干渉縞として記録し、これに光を当てて元の光の方向と強さを再生するので、空間に実像を見る事が出来る[34][35]。看視位置を変える事により見える光の方向が変化するので、運動視差があり、自然な立体映像を表示可能であるが、干渉縞のデータ量が膨大である為、現在のハードウエア処理能力では、再生を実時間で行う試みはあるが[36]、実写動画の撮影は困難である。撮影を簡略化する為、ステレオカメラ映像から被写体の奥行量をステレオマッチングで求めてホログラムパターンを計算し、これを元にレーザー光の位相を空間光変調器で変調してホログラフィーを再生する試み等があるが[37]、入力が 2 視点である為、運動視差は少ない。ホログラフィーの原理を応用し、もっと少ないデータ量で

光線再生を行うものに、インテグラルフォトグラフィーや超多眼映像がある。

多数の微小映像を多眼レンズで再生するインテグラルフォトグラフィー方式[9]は、多視点立体映像方式の延長であるが、水平視差の他に垂直視差も持つのでより自然に近い立体映像が得られる。最近では、多数の視差映像を確保しながら解像度を上げる為に複数のプロジェクタを用いた報告や[95]、実写映像を作成する為にカメラアレーを球面状に配置する提案[96]や、小型 LCD を用いたディスプレイで生じるモアレを軽減するカラーフィルタ配置の提案[97]などが報告されている。垂直視差を捨て、その分、水平視点数を多くして映像間隔を狭め、複数の視差画素が同時に同じ網膜に写る様にする超多眼映像では[38]、それらの画素を重ね合わせて 1 つの像が網膜上に形成される様に、焦点調節が働く。この合焦点位置は、画像の視差によって構成される両眼の輻輳角のクロス点と一致するので、自然な焦点調節のまま立体視が可能になる。又、看視位置を変えると、常に複数の視差映像が同時に網膜に写りながら変化して行くので、滑らかな運動視差が得られる。各視差画像は、複数の画像が瞳径を介して網膜に写る様、非常に接近して配置（約 0.6 度間隔）する必要があり、又両眼から同じ被写体映像が見える様にする為に、数 10 から 100 枚のオーダーの視差画像が必要である。

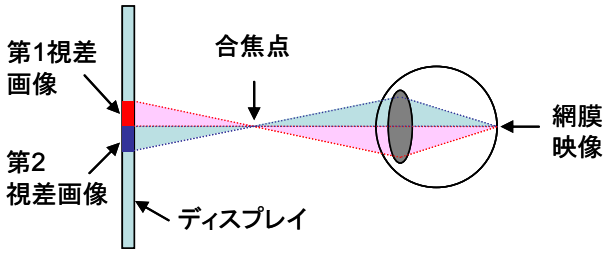


図 2- 6　超多眼映像の焦点調節効果

これを実現する為に各視差画像を光学系で細く絞られた光線で投影し、全映像が空間の同一点に収束投影される様に配置した収束化光源列を用いることが提案されているが[38]、視差画像毎の多重投影の代わりに画素毎に異なる方向の光線をインターリーブしても同じ効果が得られ、表示系が小型化出来る[39][40][41][42]。図 2-7 にこの方式の構造を示すが、マイクロレンズアレーの 1 つ 1 つが空間中の仮想平面上の画素に対応しており、その個数により、映像の解像度が決まる。各レンズの背面には、そのレンズを通る全方向の光を表示する画素を集めた微小ディスプレイがレンズの個数分設けられている。各微小ディスプレイの画素数は、垂直視差をなくしても、超多眼条件を満たす為には数十の水平視差が必要であり、一直線上に並べられない為、図の左端に示す様な斜め方向の変形 2 次元に配列されており、全ての画素の水平位置は異なる。これを焦点位置に配置したマイクロレンズを通して見ると、看視点の水平位置に依存した 1 つの画素しか見えず、左右眼は異なる画素を見る為、多視点映像による立体視が出来る。画素が垂直方向にも配置されている為、上下の画素は看視位置を上下に移動しないと見えないの

で、これを解消する為に垂直拡散板を設けて全画素の光が十分広く垂直方向に拡散する様にしてあるが、マイクロレンズの代わりにシリンドリカルレンズ（かまぼこ型レンズ）を使う事により、拡散板を省略出来る事が報告されている[43]。このシステムで再生された映像例を図 2-7 に示すが、ガラス製品等の光の反射が特定の看視位置のみで生じ、被写体の質感までが再生されている。

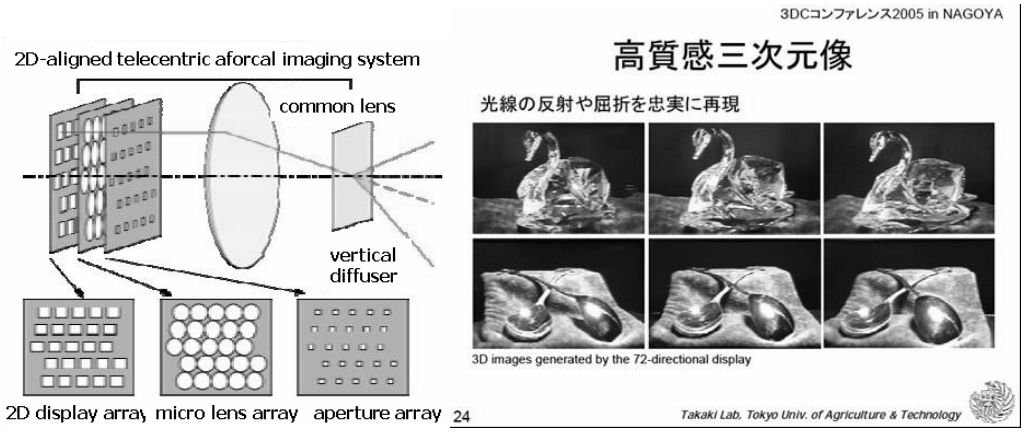


図 2- 7　高密度指向性表示システムの構成［39］

この方式では、映像の奥行量がディスプレイ面から離れる程、より広い範囲の光源からの光が網膜の同じ点に映り、映像がにごるのを防ぐ為、各画素の光をレンズアレーで平行光に整形し、隣の視差画像からの光が網膜の同一点に映らない様にする必要がある。この方式は、次節で述べる光線空間法に基づく立体映像の表示システムとして使え、見る位置によって見える光線の方向が変わるので運動視差が得られると共に、光線は十分細く絞られているので、映像位置に焦点を合わせる事が可能となり、自然な立体映像を表示出来るが、視差数は数 10～100 のオーダーが必要な為、高解像度化に課題がある[43]。

(c)　奥行標本化

3 次元映像を表示する方式として、バリフォーカルミラー[12]や回転ミラー[13][14]、回転スクリーン[15][16]を用いて映像表示面を空間中で移動させて立体映像を構成する奥行標本化方式は、可動部がネックとなり、小型化・低消費電力化・耐久性の観点から課題がある。可動部を持たない奥行標本化方式のディスプレイとして、標本化映像の多層表示がある[44]。この方式は、DMD（Digital Micromirror Device）と呼ばれる微小な鏡を画素として並べた表示パネルに映像を映し、信号の強弱によって鏡の角度を変えて光源の反射強度を変える事により映像を投影する DLP（Digital Light Processing）プロジェクタを内臓し、液晶パネルを 20 枚重ねたスクリーンにこの映像を投影し、表示される映像面の奥行位置に同期して 20 枚の液晶パネルの内 1 枚を半透明にして乱反射を

起こさせる事により、表示される映像の奥行を変化させる。このディスプレイで動画を再生する為には、20 倍高いフレームレートで映像を投影しなければばらず、高速な信号処理が必要となるが、真に奥行のある映像を表示可能で、立体視の 5 要因を全て満し、CG で作成した立体映像の表示等に適する。他の課題は、1 つの光源で最背面から投射する為、20 枚の液晶を光が通過する間に減衰が大きくなるので、明るい光源を必要とする事と、陰面処理をしないと立体映像が透けて見えることである。

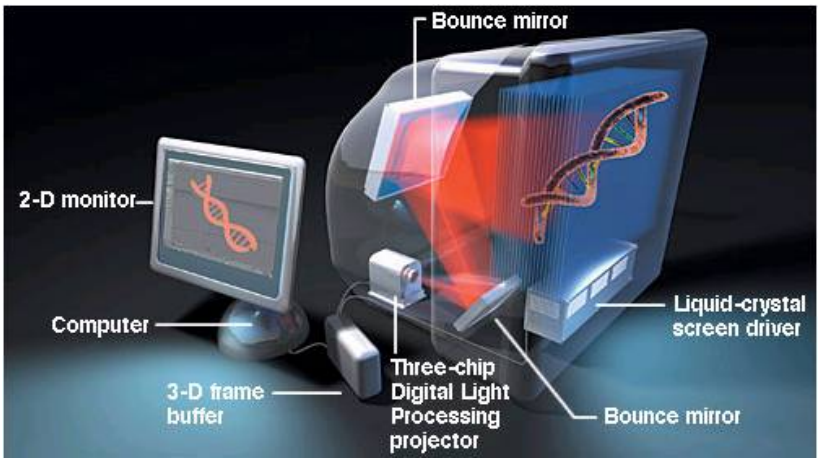


図 2- 8　奥行標本化による立体ディスプレイ［44］

2.2 立体映像の符号化技術

次に、立体映像符号化技術に関して、MPEG-2 マルチビュープロファイル（ステレオ映像符号化方法）、光線空間法、3 次元モデルベース符号化方式等に関して、現状と課題を述べる。

(a)　MPEG-2 ベースの符号化

MPEG-2 マルチビュープロファイルでは、2 画面方式による立体映像の圧縮規格が決められている[2]。ここでは、MPEG-2 階層符号化規格を利用し、ベースレーヤで左画像を符号化し、エンハンスレーヤに右画像を入れる事のみが決められている。ベースレーヤは、MPEG-2 規格で圧縮されるが、エンハンスレーヤの圧縮方法は規定されていない。通常は、左画像を参照画像として右画像を予測し、その予測誤差をエンハンスメントレーヤとして MPEG-2 規格で圧縮するが、視差量が大きいと単純差分では予測誤差が大きくなり、符号化効率は悪い。左画像に動き補償を適用して右画像を予測する方法は、被写体までの距離が十分大きく、動き補償を行うブロック内の映像までの距離が一定とみ

なせる場合には符号化効率が高くなる[45]。しかし、オクルージョンの発生や、ブロック内の各画素までの距離が一定とみなせない場合には、予測精度が悪くなり符号化効率が下がるので、予測残差の符号化で変換符号化を工夫する試みなどがなされている[46]。

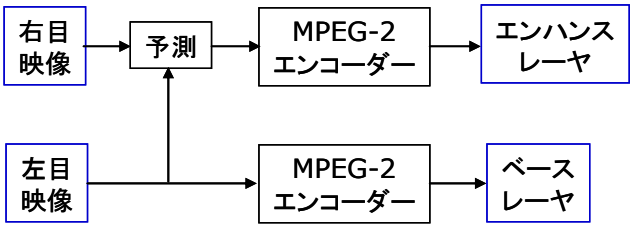


図 2- 9　MPEG-2 マルチビュープロファイルの応用例

(b)　光線空間法

2 画面以上の多視点映像の符号化例としては、一直線上に等間隔で平行に並ぶ複数のカメラ映像を重ね合わせたものを 3 次元の多視点映像空間とすると、被写体の 1 点から出た光は、重ね合わされた画像を貫く 1 本の直線上に並ぶ。この直線の傾きは、視差量すなわち奥行を表すので、この傾きから被写体までの距離情報を得て、被写体の 3 次元形状を 3 角メッシュで表すと同時に、多視点映像からメッシュの各面に貼り付けるテキスチャデータを得る方法が提案された[47]。この方法では、被写体までの距離探索を、画像に含まれる直線の傾き抽出処理で実現出来る特徴があり、Hough 変換等が使える。課題は、多方向からの映像を全て持つ為、データ量が非常に多くなる事と、カメラ数が少ない場合は直線が離散的になる為、直線の傾き検出が困難になる事である。直線の補間方法としては、直線映像のフラクタル性を利用した方法[48]などが報告されている。

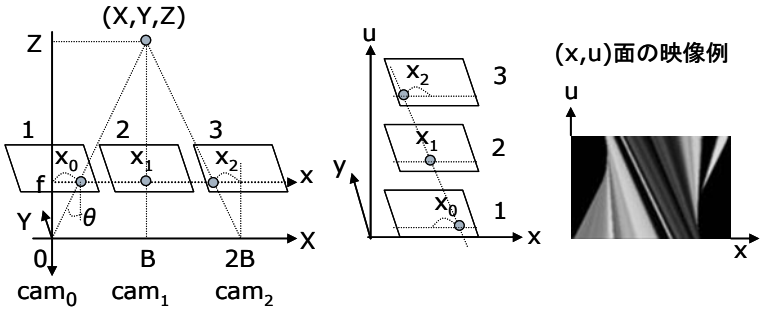


図 2- 10　多視点映像で構成される映像空間［47］

この方式はその後、自由な視点からの映像を容易に取り出せる様、多視点映像の画素の並び替えを行い、u 軸を被写体上の点から出た光がカメラに入射する角度 $u=\tan\theta$ とする事により、被写体とカメラ間に仮想のスクリーン（X,Y）を想定して、このスクリ

ーン上の仮想画素(x, y)を通る任意方向（θ, φ）の光線の内、同じ方向の光を集めたものを x-y 面とする光線空間法となった[49]。この様にすると、任意の u 面画像を抜き出す事により、任意の角度から被写体を見た時の正射影映像が得られ、斜めに切り出す事により、切り出した画面の中で光線の角度が連続変化して自然な中心射影映像が得られるので、自由視点 TV を容易に実現出来る。又、図に示す様に u 面を斜めに切る角度を変えれば画角を変えられ、広角や望遠レンズで撮った画像や、負の傾きにすると被写体を回り込みながら見た様な不思議な画像が得られる。画素の並べ替えは、同じ(x, y)座標の画素を集めてカメラ番号順に x 軸方向に並べれば u 面画像が得られるので、機械的に行える。u 軸上には、x-y 仮想面上の仮想画素毎にその画素を通る全ての方向の光が並んでいるので、これを超多眼映像の 1 画素とすれば、容易に超多眼ディスプレイに立体表示出来るメリットもある[54]。課題は、このままではデータ量が多い事と、任意角度で映像を切り出す為には、u 軸方向に十分高密度に画像を配置する必要があり、非常に多数（約 100 台）のカメラを使うか[51]、カメラ数が少ない場合は、距離の異なる被写体毎に仮想スクリーンを複数枚に分割して、各スクリーンを被写体付近に配置する方法などが提案されている[52]。圧縮には 4 次元 DCT などが検討されている。又最近では、光線空間の内、再生視点位置から見える範囲を重点的に符号化する事によって全体のデータ量を減らす試みもある[53]。

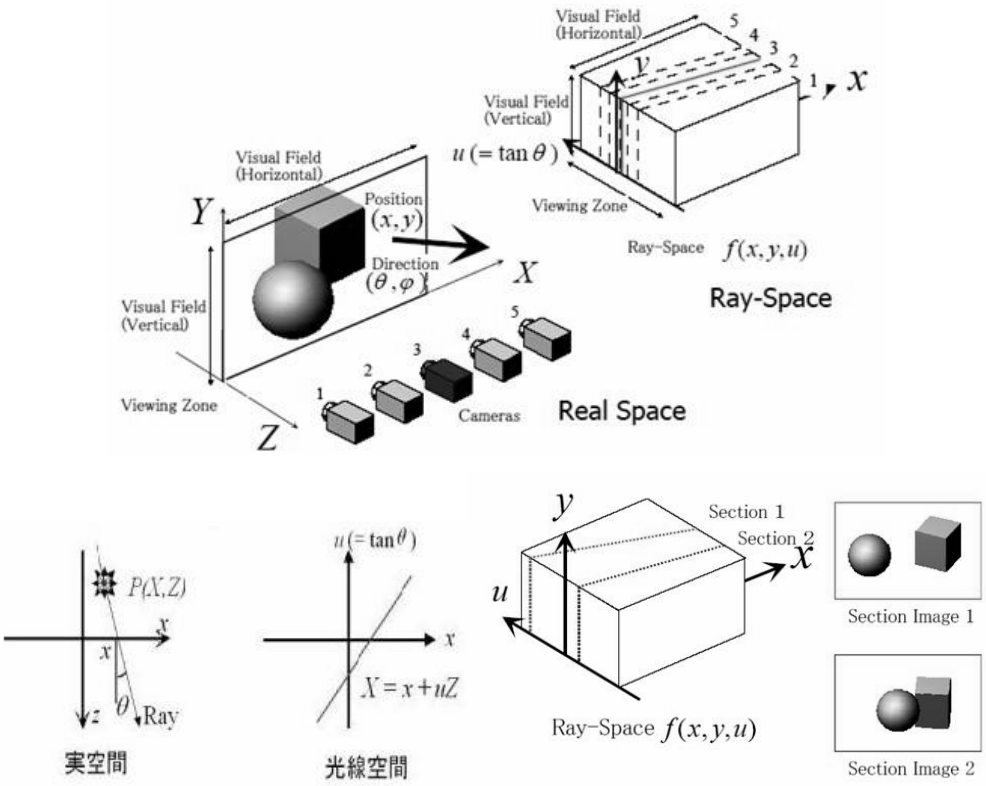


図 2- 11　光線空間法

(http://www.tanimoto.nuee.nagoya-u.ac.jp/)

現在、この光線空間法の入力映像である多視点映像の高能率圧縮符号化方式（MVC: Multi-view Video Coding）の標準化が、国際標準化機構(ISO: International Standardization Organization）傘下のワーキンググループ(MPEG: Motion Picture Expert Group）で始まっている[7]ことは先に述べた。ここで想定されている多視点映像は、直線上、円弧上、又は格子上に配置した複数のカメラで共通の被写体を撮影して得られる映像であり、これらの映像を処理することにより、ユーザが希望する任意視点からの映像を再生する自由視点テレビの実現や[55][56]、異なる視点からの映像を多眼立体ディスプレイに多重表示する事により、両眼視差や輻輳角の違いによる奥行知覚を生成する立体テレビの実現などが想定されている。自由視点テレビや立体テレビを実現する為の視点数は、多い程精度の高い映像生成が可能になるが、それだけ伝送すべきデータ量が増える為、高能率な圧縮符号化方式が望まれており、従来MPEGで標準化されて来た、１視点映像の時間軸方向の相関を利用した動き補償フレーム予測の考えを、カメラ間映像の予測にも拡張する事が検討されている[57]。

従来の動き補償方法では、フレーム映像を小ブロックに分割し、各々のブロックが時間的に前後のフレームのどの位置から動いて来たかを、ブロックの上下左右への平行移動によるブロックマッチングで探索し、最小のマッチング誤差を与える平行移動量を動きベクトルとして符号化すると共に、映像の予測誤差であるマッチング誤差を変換符号化・量子化・可変長符号化するものであり、フレームの中にある動物体の動きを補償して高精度な予測符号化を実現する。この考えは、カメラ間映像の予測にも適用可能であり、その場合には、被写体の動きの代わりに、カメラ位置の違いに起因する静的な視差による同一ブロックの位置の違いが映像間で存在するので、これをカメラ間の動きベクトルと見なせば、従来のブロックマッチングによる動き補償手段や、得られた動きベクトルの圧縮符号化手段がそのまま適用出来る。

しかし、従来の動き補償はブロックの平行移動による探索しか行わない為、多視点映像の様なカメラ位置によって、同じ被写体の面の向きが異なって見える場合には、同じ被写体の写っているブロック位置を探索しても、被写体の形状が異なって射影される為、ブロックマッチング誤差が小さくならないと言う問題がある。この問題に対処する為には、被写体までの距離を求めると共に、被写体の面がカメラの光軸に対してどの様に傾いているかを抽出して、射影幾何学的にカメラ映像の形状を求める必要がある。コンピュータビジョン技術によるエッジやコーナー等の画像の特徴点抽出と[58]、それらの映像間のマッチングにより、エッジやコーナーで囲まれる面形状を推定するアプローチは、エッジやコーナーの画像内での分布の非均一性や、滑らかに変化する面など被写体の輪郭抽出が困難な場合があると共に、抽出されたエッジやコーナーをつないで得られる面は非定型であり、定型ブロックの変換符号化を軸にする従来の画像圧縮技術とは整合し難いと言う課題がある。

(c)　コンピュータビジョンによるアプローチ

コンピュータビジョンをベースにした立体映像符号化では[59][60]、被写体の周囲を取り巻く多視点カメラで撮影した映像から3次元モデルを作る。3次元モデルの形状は、視差画像間のマッチングを行って対応点の視差量を得、この値から、カメラの位置と姿勢を現すカメラパラメータを使って、エピポーラ幾何による射影方程式を解き、被写体の各点の空間座標を求める。マッチングは、被写体のコーナーやエッジ等の特徴点を、画像処理により輝度や色成分の変化部として抽出し、他のカメラ映像でも同様にして対応点を抽出する方法が一般的である。対応点の決め方には、2 台のカメラの位置と姿勢で決まるエピポーラ幾何を利用して探索を行う。その為には、正確なカメラ座標が知られている必要があり、テストパターン等を用いて対応点の誤差が最小になる様にカメラパラメータを求める。これをカメラの校正（rectification）と呼ぶ。

この様にして得られた3次元空間の座標値を、3角形や四角形の小さなパッチで覆ったものをメッシュと呼び、コンピュータグラフィックスで立体形状を表すのに用いられている。各カメラ映像と 3 次元空間の被写体との間の射影関係を用いて、形状にテキスチャを貼り付けて 3 次元映像データが出来上がる。3 次元映像データの持ち方は、メッシュによる形状データと、その上に貼り付けるテキスチャデータとを別々に持つ方法と、voxel と呼ばれる 3 次元画素に変換して画素毎に空間座標と色情報を持つ方法がある。

一旦 3 次元映像データが求まれば、任意視点から見た映像は、この 3 次元データをその視点に射影して得られる。

3D モデリングによる立体映像の符号化方法に関しては、MPEG-4 SNHC 規格[3]で定められている様に、メッシュデータを適応予測符号化し、テキスチャデータは wavelet 変換等で圧縮符号化する方法が一般的である。課題は、3 次元形状データ取得の為の処理が重く、高精細度動画の実時間符号化が困難な事である。

(a) 形状(距離)情報とテキスチャ　　(b) 多面体表現　　(c) voxel 表現

図 2- 12　コンピュータビジョンによる立体映像例 [59]

2.3 立体映像の品質評価技術

本節では更に、立体映像の品質評価技術について、ステレオ映像の画質評価の研究、立体映像の感性・心理的側面評価の研究、大画面映像の臨場感・没入感評価の研究をサーベイし、その現状と課題を明らかにする。

(a) 立体映像の総合品質評価

立体映像の総合品質に関しては、臨場感の用語が用いられ、高臨場感を実現するディスプレイが開発のターゲットになって来ている[22][61]。そこでは、高臨場感とは、直接目で見る世界を忠実に再現し、あたかも自分がそこにいる様な感じを与える実写映像を対象にしたものと、仮想的空間で新しい視覚感性インパクトを与える合成映像を対象とするものに分けられ、両者の混在したものを Mixed Reality と定義している。この高臨場感を実現するディスプレイに要求される機能は、広視野角、立体感、高精細とされており、立体感により臨場感が高まったとの主観評価報告もあるが[62]、主観評価結果のみであり、客観的、定量的評価は不十分である。又、不用意に映像の立体化を行うと、撮影時と表示時の映像ジオメトリ比が変化する事に起因する「箱庭効果」と呼ばれる、被写体がミニチュアの様に縮小して見えたり、望遠レンズを用いた時に、被写体までの距離とカメラ間隔が不適切な場合に生じる「書き割り効果」と呼ばれる、被写体が薄い平板の様に見えたり、「画枠歪み」と呼ばれる、突出した立体がディスプレイの縁に掛かると突然立体感が喪失する現象や、同じ視差量でも画面の上に行く程遠くに感じる「鉛直奥行傾斜現象」や、2 眼式立体映像による不自然な運動視差の発生等の問題が指摘されており、注意が必要である。

広視野角
高解像度
立体化

図 2- 13 高臨場感サービスの視聴イメージ
（http://www.nhk.or.jp/strl/open98/4-7/realidx-j.htm）

(b)　立体映像の臨場感評価

高臨場感放送サービスを目指した立体ハイビジョン画像から受ける心理的印象を分析した報告[63]では、映像の印象を、反対の意味を持つ言葉対の間を 7 段階に分けた評価尺度で測る Semantic Differential Technique を用いて、385 個の形容詞の中からハイビジョン又は立体ハイビジョンを表すと思われる 37 語を抽出し、その主観値を得、これらの相関行列から 8 個の固有値を得、これを因子の次数として、因子分析を行った結果、美しさ・繊細感、生命感、安定感、濃淡感、大小・遠近感、現実感、連続感が因子として抽出された。この因子の得点を説明変量にして、主観評価の際に得た良い／悪いの間を 7 段階に評価させた値を映像の総合品質評価値である目的関数として重回帰分析を行った結果、平面ハイビジョンでは、8 因子から総合評価値が有意に推定出来るが、立体ハイビジョンでは有意な推定は出来ないとの結果が報告された。又、最近の報告[71]でも、スーパーハイビジョンの平面映像を用いた主観評価実験結果から、水平視野角が増すほど臨場感が増す事と、力量感と臨場感の間に相関（0.795）が見られたが、重心動揺値から没入感を求めたものは、主観的な臨場感とはほとんど相関が見られないとの報告があった。これらの原因は、抽出された因子に、「生命感、安定感、現実感、連続感」などの映像コンテンツに依存する様な感性的形容詞が多く、又、「良い／悪い」の主観品質評価語が、好き／嫌い等の看視者の好みに左右されやすい用語であった為ではないかと思われる。又重心動揺による没入感の計測では、コンテンツの動きに重心動揺が大きく影響されるのに対し、静止画コンテンツで評価した臨場感との対比では、コンテンツの影響が大きく現れ、主観値と客観値が整合しなかったものと思われる。

平面映像の評価に関する別の報告でも、主観値と客観値を整合させる為には、両者に関係のある評価項目を使うことや[72]、人の視覚特性を盛り込んだ物理パラメータを使う事の重要性が指摘されている[73]。従って、立体映像の総合品質評価においても、映像の大きさやきめ細かさ、滑らかさと言った物理パラメータに裏打ちされた主観評価項目を用いると共に、客観パラメータには、視覚特性から導かれる拘束条件をあてはめた評価を行う事が重要であると思われる。

(c)　立体映像の画質評価

立体映像の画質評価に関する報告[64]では、映像のエッジとそれ以外の部分での画質劣化の見え方の相違や視差情報を考慮した画質評価モデルとして PQS_{stereo}（Picture Quality Scale for coded stereoscopic image）を用いた評価結果が報告されている。ここでは、左右の原画像と符号化画像の RGB 値を CIE の L*a*b 色空間成分に変換し、この符号化誤差から歪要因の主成分分析を行い、更にテキスチャ特徴量に関する主成分を加

えた後、主観評価実験により得られた MOS（Mean Opinion Score）値との関係を非線形重回帰モデルに当てはめて PQS_{stereo} の値を得ている。

$$PQS_{stereo} = \frac{4}{1+\exp\{-\beta(PQS_{mr}-3)\}}+1$$
$$PQS_{mr} = b_0^* + \sum_{i=1}^{N} b_i^* Z_i + \sum_{j=1}^{M} b_{N+j}^* Y_j \qquad (2.1)$$

ここで、b_i^*は回帰係数、Z_iは基礎歪要因の主成分、Y_jはテキスチャ特徴量の主成分である。この方法は、主観評価実験から得られる主観画質の MOS 値を目標値として画質評価モデルの係数を決める為、主観評価実験値との整合性が良いが、JPEG 符号化誤差のみの評価であり、立体画像の総合品質に影響すると思われる、画像の大きさから生じる没入感や、立体映像で感じられる立体感などの評価・関連性は調べられていない。又、画質にどの様な物理パラメータがどれ位利くのかと言う事が明示的に示されないので、システムの改善を行う場合の見通しが立ちにくいと言う課題がある。

(d) 大画面映像の没入感評価

大画面映像の没入感を計測した報告[65]では、ディスプレイに表示する回転映像の視野角や提示映像の上下位置に起因する重心動揺を計測し、視野角が大きくなる程重心動揺が大きくなり、没入感が高まる事や、回転運動する映像の位置がディスプレイの下方（足元付近）にある時に重心動揺が大きくなる事が報告されている。

映像コンテンツと映像酔いとの関係を調べた報告[66]では、観察者の周囲の壁に一方向回転運動や往復運動をする CG 映像を投影し、映像酔いに関する主観値と、誘導自己運動感覚（ベクション）の強度とを自己申告で 11 段階に評価させた結果、視線を軸とした回転運動で大きな値が得られた事が示されている。

これらは、全視野が映像で覆われるような場合には、視覚のよりどころが回転運動する映像しかなく、特に不断動かないはずの地面位置が動く様に見える場合には、錯覚によって平衡感覚を失うものと思われる。錯覚を利用したものでは、小さな図形でも、どうしても平行線に見えない図形や、静止画なのに動いて見える図形などが多数報告されている[67]。これらも没入感と呼ぶべきかどうかは、今後の議論を待つ必要があると思われるる。これらの報告は、映像コンテンツに起因する主観値の測定で、映像の大きさや精細さ、立体感などの、映像の客観品質との相関は計測されていない。又、高品質映像を目指す為の没入感評価では、自然映像や立体映像での評価も必要と思われる。

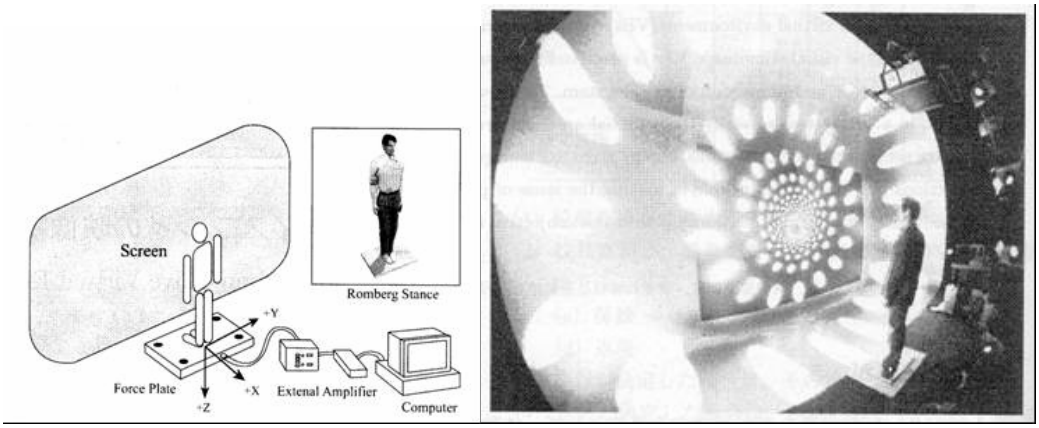


図 2- 14　没入型ディスプレイでの重心動揺測定[65]

■ 第 3 章 ■

立体映像配信に関する画像品質評価法及びシステム構成技術の概要

第3章　立体映像配信に関する画像品質評価法及びシステム構成技術の概要

本章では、本論文で提案・検証する、理想的な立体映像である 3 次元立体映像を配信する為に必要な画像品質評価法及びシステム構成技術の概要を述べる。最初に、従来技術の問題点を要約し、これを解決する為に、本論文で提案する、立体映像配信に必要な技術の一つである 3 次元立体映像の品質評価技術の概要を述べ、次に、この評価結果から指摘される課題を解決する為の、理想的な立体映像を効率良く生成する3次元立体映像生成技術の概要を述べ、更に、その検討結果から指摘される課題を解決する為の、大きな運動視差確保に必要な多視点化により発生する課題である、大量のデータを効率良く圧縮符号化する為の、立体映像符号化技術の概要について述べる。

3.1 従来技術の問題点

第 1 章で述べた様に、理想的な立体映像配信を実現する為には、

1. 立体映像の総合品質評価評価方法の確立
2. コストパフォーマンスの良い理想的な立体映像生成方法の確立
3. 高能率・高画質な立体映像符号化方法の確立

が不可欠である。この要求に対して、第 2 章で述べた様に、従来技術では、これらの目標を目指す技術はあるが、いずれも課題がある。

第 1 の品質評価方法に関しては、符号化ノイズと画質の関係や、水平視野と臨場感の関係、回転映像と没入感の関係などの個別の検討はあるが、これらを総合して立体映像の総合品質を表すパラメータとして定式化されたものはない。

又、第 2 の立体映像生成方法は、2 眼式立体映像の課題を解決する為の、裸眼式立体映像方式や、ホログラフィ的な考えを導入した多視点立体映像方式等、一歩理想に近付いたものはあるが、実現に要するコストが高く、高精細度化が困難である等の課題があり、コストパフォーマンス良く理想に近い立体映像を生成する技術は未完成である。

第3の膨大な量になる立体映像データの高能率符号化方法に関しては、今まさに検討が始まった段階であり、完成された技術はない。

従って、本論文では、これらの技術課題の解決に貢献する為に、視覚要因を全て満たす 3 次元立体映像を理想的な立体映像として、これを実現する手段として

1. 3 次元立体映像の総合品質評価方法

2. コストパフォーマンスの良い 3 次元立体映像の生成方法

3. 3 次元立体映像の符号化方法

の提案とその検証を行い、将来の立体映像配信の実用化に貢献する事を目指す。以下にその概要を記す。

3.2 3 次元立体映像評価技術の概要

3 次元立体映像の総合品質を評価する尺度を求める為には、3 次元立体映像の理想モデルが必要である。ここでは、3 次元立体映像が持つ物理パラメータを全て列挙し、その全ての項目を計測する事により、その総合品質を評価する事を試みる。ここで計測する物理パラメータは、映像が視野を覆う大きさ、時空間的きめ細かさ、映像信号のダイナミックレンジと雑音、立体視要因等の全てを考慮し、これを没入感に関する項目 I と、画質に関する項目 Q と、立体感に関する項目 T にまとめ、その積で総合客観品質 R を表す事を試みる。

$$R = f(I, Q, T) \tag{3.1}$$

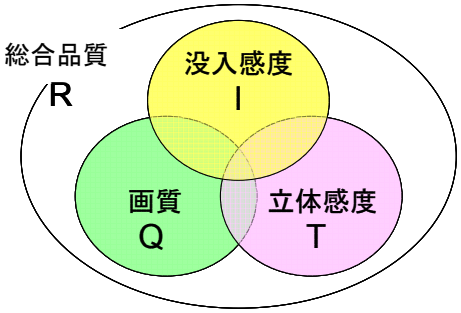


図 3- 1　立体映像の総合品質

次に、この客観品質 R は、主観値と整合しなければ意味がないので、視差バリヤー型裸眼立体実写映像を用いて主観品質評価実験を行い、得られた主観値 MOS_R と客観値 R の間を次のロジスティック関数で結び、

$$MOS_R = \frac{4}{1+\exp(\alpha - \beta R)} + 1 \tag{3.2}$$

両者の非線形回帰分析を行って、主観値と客観値の整合化を図ると共に、客観値を得る為の評価式の改良を重ねて、客観値の信頼度を高めると共に、多大な労力を必要とする主観評価実験を行わなくても、映像の物理パラメータのみから容易に総合品質の指標が得られる様にした。このシンプルな評価式を用いる事により、立体映像機器のどのパラメータがどれ位総合品質に影響するかが容易に分かる様になった。これにより、立体映像機器の生産や新方式開発

などで必要になる評価実験を容易にし、産業的に貢献する事を目指す。主観評価実験に当たっては、評価者の個人的バラツキを排除する為に、評価者に事前の説明を十分に行うと共に、提示映像は、理想的な立体映像とは全く実物の様に見える映像と仮定して、評価をし易くすると共に、実写映像を用いる事により、主観評価と客観評価の整合化を図った。

次に、この様にして得られた評価式が広く一般に適用出来るか否かを確認する為に、評価式の検定を行うと共に、更に多数の評価者に多種類の立体映像を評価してもらい、評価値の検証を行った。ここでは、多数の評価者に評価内容の理解が十分に行き渡らない事を想定し、評価項目数を増やして、その平均値で総合評価を行う事により、評価結果が偏らない様に心掛けた。又、評価項目の主成分分析を行い、主観評価項目の平均値で総合品質が表せる事を確認した。提示する映像コンテンツや立体方式での偏りを排除する為に、出来るだけ広範囲のコンテンツと立体方式で評価を行う様努めたが、実際に評価データが得られた立体映像は、

1. シャッターグラスによるマルチ画面立体バーチャルリアリティ(VR)映像
2. 分光眼鏡による大画面立体コンピュータグラフィックス(CG)映像
3. 偏光眼鏡による大画面立体アニメ映像
4. 偏光眼鏡による大画面立体実写映像
5. 偏光眼鏡による大画面立体実写合成混合映像

である。これらの評価結果より、実写映像では主観値と客観値の整合が見られたが、CG 映像等の合成映像では客観値が主観値よりも高めに出る事が分かった。この原因は、主観評価目標が実物の様に見える映像としたのに対し、提示される映像が合成映像である為、本物に見え難い事に起因すると思われたので、合成映像の場合は本物らしさ係数 K_R を導入して客観値と主観値の整合化を図った結果、K_R=0.445 の時に両者はほぼ整合する事が分かった。この値は、映像合成技術の進化に伴って変化して行くと思われるので、今後の継続調査が必要である。本提案の評価式は立体視要因を全て含み、全ての方式の立体映像の評価に適用可能であるが、主観値との整合性検証は、裸眼式及び各種眼鏡方式の立体映像と実写・合成映像の組み合わせで確認されたので、これらの立体映像の評価に特に有効と思われる。

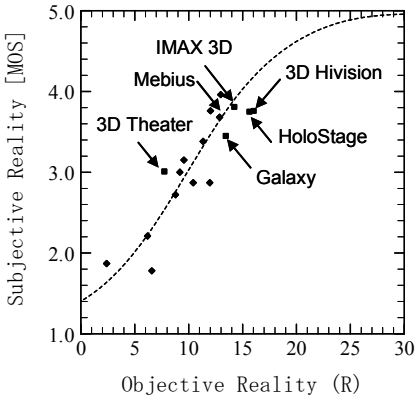


図 3-2　立体映像の総合品質計測結果例

又、これらの評価結果の分析から、理想立体映像に近づく為には、単なる大画面化のみでは総合品質は上がらず、表示画素数の増加が視野角の拡大と画素密度の向上の両方に有効な事が明確になった。更に、立体感の総合品質への影響は、小さな奥行量で飽和する事が分かったが、この原因は、立体感要因の内、僅かな奥行量で飽和する両眼視差の感度が最も高い事の他に、評価した映像がいずれも 2 眼式の立体映像であった為に、単なる輻輳角の増加ではその他の立体視要因が伴わず、結果として主観値も客観値も共に総合品質が上がらなかったものと思われ、理想的な立体映像の実現には、映像の高画素数化と全ての立体視要因を満たす立体化が重要である事が判明した。

3.3 3 次元立体映像生成技術の概要

上記の検討結果から示唆される課題の内、第 1 の理想立体映像を実現する為の画素数の増大は、現在の半導体技術の延長線上で時間と共に解決されて行くと思われるので、ここでは、第 2 の課題である立体視要因を全て満たす理想的な立体映像をコストパフォーマンス良く生成する技術の検討を行う。本論文で提案する立体映像生成技術では、高解像度化が容易な様に画素を 3 次元に配列する事を考える。この様にする事により、立体映像の表示は従来の 2 次元ディスプレイの多層化で実現出来、光線再生法の様に平面上に全ての画素を並べる必要が無くなり、小型化と実装の容易さの両方が実現される。画素を線形 3 次元空間配列する場合、有限な被写体は表示可能であるが、無限遠を含む被写体の場合は、無限大の奥行が必要になり、実現不可能になる。しかし、人の視覚特性は、遠方になるほど解像度が落ちると共に、奥行分解能も低下する。従って、無限遠を含む線形な3次元空間を座標変換して、視覚特性に合った非線形空間に写像する。具体的には、人の視覚器官と同じ特性の焦点距離 f の凸レンズを用いて線形空間(X, Y, Z)を射影し、出来た非線形映像空間(x, y, z)を 3 次元標本化する。この非線形射影方程式は次の様に表されるが、

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \frac{f}{Z-f} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3.3}$$

無限遠点 Z=∞は、射影空間では有限点 z=f に射影され、近点 Z=D は、Z=f の近傍である z=Df/(D-f)に射影されるので、非常にコンパクトな 3 次元映像空間が得られる。しかもこの映像空間の構造は視覚特性にマッチしているので、自然界を見たのとほぼ同じ感覚が得られる。

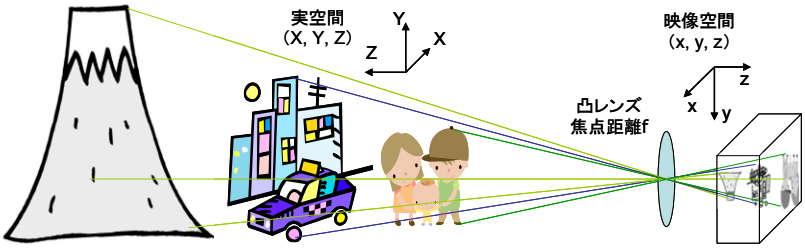


図 3- 3　空間標本化法による 3 次元立体映像の標本化

この映像空間の 3 次元標本化は、動画への応用を考慮して、3 次元イメージセンサを用いて、一度のサンプリングで全データを取得する事を検討する。3 次元イメージセンサは平面に配列されたセンサアレーを積み上げて構成出来るが、全てのセンサーに光を届かせる為には、センサー素子を透明な半導体で作る必要がある。透明な半導体素子は、多数知られており、透明なトランジスタの試作も行われており、液晶ディスプレイの様に実用化されているものもあるので、実現可能と思われる。

3 次元ディスプレイは、透明な液晶パネルを多層配置したものが既に商品化されているので、現在でも入手可能であるが、有機 EL 素子などの発光ダイオードアレーを透明にして積層するのが、装置の小型化に有効と思われる。更に大画面化するには、ディスプレイパネルの多数貼り合わせで大画面を構成する方法と、3 次元映像をプロジェクターで投影する方法が考えられる。投影する場合には、無限遠の被写体は無限遠に焦点を結ぶので、スクリーンを無限遠に設置する必要があり、現実的でないが、反射光を入射光の方向に返す屈折スクリーンを用いる事で、全ての距離の被写体映像を有限の距離から看視出来る様になる。この屈折スクリーンは、コーナーキューブミラーや、屈折率 2 のガラス球等を敷き詰めて作る事が出来る。

3 次元標本化された映像は、各レーヤに被写体の距離に応じた合焦オブジェクトと共に、他のレーヤで焦点を結ぶべき被写体のボケオブジェクトを含み、映像全体が暗くなったり、コントラストの低下をもたらすので、よりクリヤーな立体映像の生成の為にはこれを除去し、合焦オブジェクトのみを抽出する事を検討した。

d=50mm	d=50.425mm	d=50.85mm	d=52.125mm
(D=∞)	(D=6m)	(D=3m)	(D=1.2m)

図 3- 4　空間標本化法でサンプリングされた映像

ここで提案する合焦オブジェクト抽出方法は、先ず、映像の高周波成分を Laplacian フィルタで抽出することにより、合焦判定を行うが、被写体テキスチャの高周波成分が元々少ない場

合、判定閾値が変動するので、これを防ぐ為に、隣接するレーヤ間のピーク検出を行うことで、判定閾値の変動を抑える。次に、Laplacian フィルタは高感度であるが、誤判定も起こり易く均一な領域抽出が困難であるので、抽出された領域の融合処理を行う。ここでは、後で述べる輝度分配による奥行分解能の拡大を考慮し、線形なローパスフィルタを用いて合焦領域の平滑化を行った。その結果、合焦領域のエッジが鈍るので、エッジ整形関数を適用した結果を、原画像に加える輝度オフセット値とした。

この結果下図に示す様に、各レーヤで焦点の合ったオブジェクトのみが抽出された。これをz軸方向に重ねて配置する事で、奥行を持った3次元立体映像が形成される。この3次元立体実写映像空間に 3D のコンピュータグラフィックス映像を射影し、実写映像と合成映像の混在したハイブリッド映像の検討や、映像品質を高める為のオクルージョンの検討を行った。

d=50.00mm　　d=50.425mm　　d=50.85mm　　d=52.125mm

図 3- 5　合焦オブジェクトのみが抽出された立体映像

この立体映像方式では、撮影と表示に多層デバイスを必要とするが、年々進化する半導体技術により低コスト化は容易と思われる。又、各レーヤのオブジェクトは夫々、1 つのレーヤにしか存在しないので、全てのオブジェクトを合わせても、1 枚の平面映像とほぼ同じデータ量しか持たず、蓄積・伝送は容易である。従来の平面映像に奥行データを追加して擬似立体映像にする方式との違いは、オクルージョンデータを持つので僅かな運動視差が得られ、自然な立体映像が生成出来ることである。課題は、オクルージョン量が少ないので、看視位置が映像のほぼ正面に限られる事である。この問題の解決方法としては、複数のカメラを並べて異なる視点位置からの映像を取得し、不足するオクルージョン部分を埋める方法や、光学的又は磁気的なヘッドトラック装置で看視者の顔位置を検出し、その位置に合わせて映像を切り替える方式等が考えられるが、映画や TV を含めて通常の映像鑑賞では、看視者があまり動き回る事はないので、過度の運動視差確保の為に膨大なデータ量を持つのはコストパフォーマンスが悪いと思われる。

しかしながら、映像世界の中を自由に動き回る仮想現実(VR)用途等では、十分な運動視差確保が必要であり、真に理想的な立体映像の実現には、多視点化が必要と思われる。この場合の課題は、映像データ量が膨大になる為、効率の良い圧縮符号化技術の開発が不可欠になる。

3.4 3 次元立体映像符号化技術の概要

前述の検討から指摘される課題を解決する為に、ここでは、立体映像を多視点化した場合の高能率符号化技術の検討を行う。先にも述べた様に、空間標本化による 3 次元立体映像では、奥行方向の映像の冗長性は殆どなく、圧縮の観点からは 1 枚の平面映像と見なせるので、ここでは多視点映像を得る為のマルチカメラ映像間の相関について検討する。この検討結果は、映像オブジェクトを奥行ごとに分離した場合にも適用出来る。

多視点映像では、下図に示す様に等間隔に配置された複数のカメラで共通の被写体を撮影するのが一般的である。その場合、各カメラ映像は写真に示す様に互いに似ているが少しずつ異なる（写真では、被写体の撮影アングルを変えて変化を誇張してある）。従って、単なる映像間の差分予測やブロックの平行移動予測では予測誤差が減らず、高画質化に課題がある。

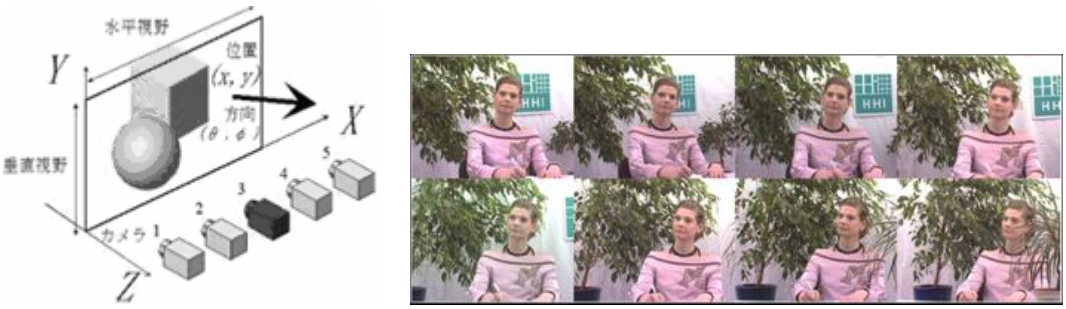


図 3- 6　多視点立体映像の例

そこで、本研究では、被写体（X, Y, Z）とカメラ映像(x, y）間の射影方程式を解析する事により、映像間の相関の規則性を調べた。

ここでは、一般的な平行カメラ配置の場合を想定し、被写体面の傾きに着目して、同じ被写体像がカメラ位置によってどの様に歪むかを調べた。被写体が平面の場合は、そのカメラ映像は Affine 変換で得られる事が知られているが、被写体上で規則的な方形などの形状は、カメラ映像上では不等辺四辺形に歪み、映像間の規則性を見出しにくい事と、方形ブロックを基本とする従来の MPEG などの圧縮方式との整合性が悪い問題がある。そこで、映像面上の方形なブロックに写る被写体像が他のカメラでどの様に写るかを解析した結果、幅と傾きの変わる平行四辺形になる事が解明された。特に、被写体の中に遠方まで伸びる通路等で構成される面がある場合にブロックの幅変化が大きく、床や天井がある場合に傾き変化が大きい事が判明した。

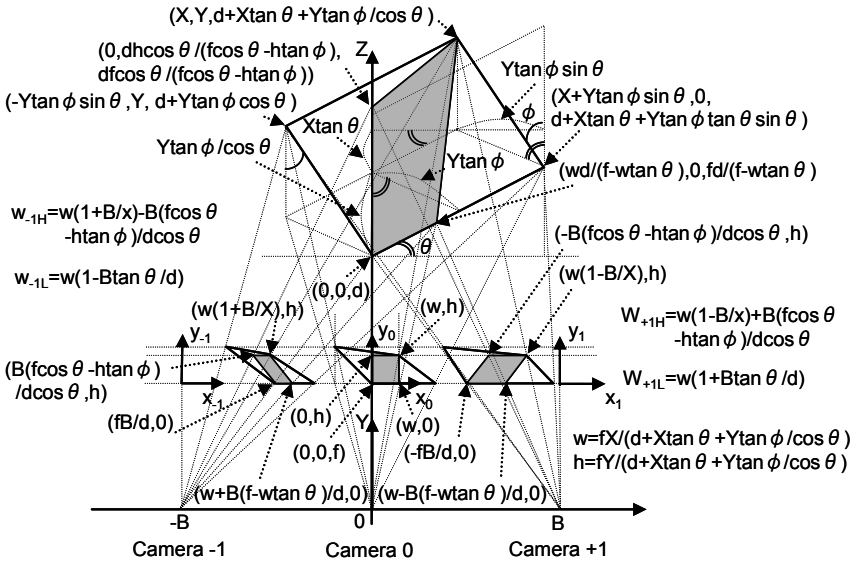
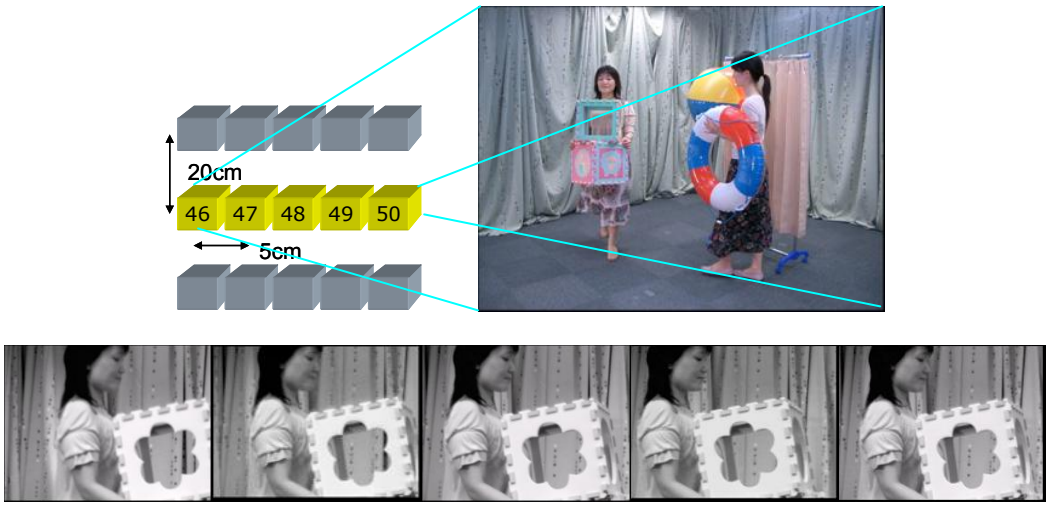


図 3- 7　任意方向を向いた被写体面とカメラ映像の関係

この結果より、カメラ映像間の予測符号化を行う場合に、単なるブロックの平行移動マッチングのみでなく、ブロックの幅探索と傾き探索を行えば、より予測精度が高まることが期待される。更に、ブロック位置と幅と傾きは、カメラ位置で決まる一意的な関係があるので、1つのカメラ映像で対応ブロックが見つかれば、他のカメラ上での対応ブロックは探索する事無く、一意的に求められる事及び、この関係を利用して、ブロック情報を圧縮出来る可能性がある事も判明した。

そこで、国際標準化機構（ISO）傘下の MPEG（Moving Picture Expert Group） MVC（Multiview Video Coding）テストシーケンスを使用し、符号化効率を調べた。

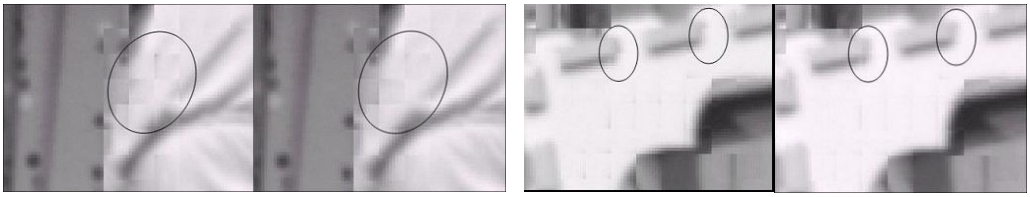


camera 46　　camera 47　　camera 48　　camera 49　　camera 50

図 3- 8　実験に用いた多視点映像の例

先ず、片方向予測と双方向予測を比較した結果、片方向予測では参照映像でカバーされないエリアが出来る為、予測効率が上がらないが、両側から予測する双方向予測を行う事により、このエリアがカバーされると共に、前景に隠れて予測出来なかったオクルージョン部も予測出来る様になり、予測精度は大きく改善される事が分かった。しかし、ブロックサイズが大きいと、ブロックの中に距離の異なる被写体が写る様なオブジェクトの境界部分でオクルージョンの発生により予測が合わず、画質が改善されない事が判明したので、ブロックサイズを 16×16 画素から 8×8 画素に変更した結果、更に最大 3.29dB の画質改善が見られた。

次に、8×8 画素のブロックの傾きと幅の調整を行い、予測精度を確認した結果、幅調整を行う事により約 0.5dB の改善が見られ、傾き調整を行う事により約 0.3dB の改善が見られた。映像の中に遠方まで伸びる通路や床がある場合には、更に改善効果が上がると思われる。



補償なし　　幅補償有(34.33dB)　　補償なし　　傾き補償有(34.06dB)

図 3-9　ブロックの幅補償と傾き補償結果の例

次に、参照ブロックの形状補正に必要な幅情報や傾き情報の符号化を検討し、夫々の情報を画像データと見なして画像データの符号化と同じ処理を施す事により効率良く符号化される事が分かり、総合のレート歪み特性での改善効果を確認した。

更に、予測に使われたブロック情報(視差情報)の圧縮方法を検討した。カメラ間隔を B とすると、多視点映像間で対応するブロックの位置座標の差(視差ベクトル Δ)は、次式を満たす事が射影方程式の解析から分かっているので、これを利用する事により、視差ベクトル全てをデコーダに送る必要はなく、視差情報の圧縮が行える。

$$\Delta_n = n \times \Delta \qquad (3.4)$$

ここで、Δ は参照カメラに隣接するカメラ映像での視差ベクトル、Δ_n は、参照カメラから n 倍離れたカメラ映像での視差ベクトルである。従って、視差ベクトルが1つ分かれば、これを参照視差ベクトルとして、他のカメラの対応するブロックの視差ベクトルを上式より予測する事が出来る。課題は、対応するブロック位置がカメラによって変わる事であるが、そのブロック位置も上式から求める事が出来る。

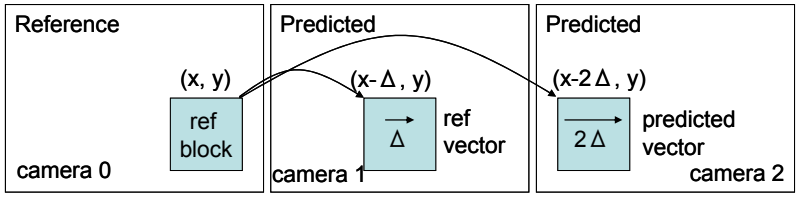


図 3- 10　視差ベクトル間の規則性

以上の検討に基づく視差ベクトルの予測符号化実験を行った結果、単に隣接カメラの視差ベクトルを予測した場合は、視差ベクトルの情報量削減に対して、画像の予測誤差が増大するため、総合のレート歪み特性では改善が見られなかったが、複数のカメラ映像の視差ベクトルから1つの代表視差ベクトルを予測し、この視差ベクトルを共通的に用いる事で、視差ベクトル情報の大幅な削減が可能になり、特に低ビットレートでレート歪み特性が改善される事が分かった。

■ 第 4 章 ■

3 次元立体映像の評価技術

第4章　　3 次元立体映像の評価技術

本章では、立体映像配信に関する技術の評価手段である、立体映像の総合品質評価方法を検討する。立体映像の総合品質は、単に信号対雑音比で表される画質のみでは評価し切れず、映像の奥行が見える事から生じる強い臨場感や本物感、又、大画面・高精細映像から感じられる没入感等を総合して評価する必要がある。この総合評価尺度として臨場感度を定義する。

4.1 臨場感度の定義

臨場感度は、立体映像の総合品質を、提示される映像世界を看視者がどれ位実物と感じるかの尺度で表す。臨場感度は、高品位立体映像の総合品質評価尺度として定義するが、工業的にも利用できる様にする為、カメラや記録・伝送装置、ディスプレイやプロジェクタ等の映像機器の性能を客観的・定量的に表せる様、提示される映像の物理パラメータのみを用いて定義した。立体映像の物理パラメータとしては、画像サイズ、画素の時空間密度、輝度・色空間のダイナミックレンジの広さ、符号化歪や雑音の量、奥行・立体感度（両眼視差、運動視差、等）などが上げられるが、これらのパラメータを、画像の大きさやきめ細かさから得られる没入感（I）項と、映像の色や輝度の広がりや純粋さを表す画質（Q）項及び、現実世界では必ず存在する立体感（T）項に大別して、次式で定義した。

$$R = 100 \times I \times Q \times T \tag{4.1}$$

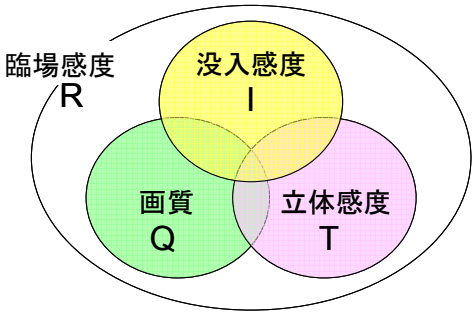


図 4- 1　臨場感度のパラメータ

ここで、臨場感度の基準値は 100 とし、各パラメータは独立に変更可能で、例えば大

画面に起因する没入感は、画質や立体感からは得られず、いずれが欠けても互いに補え合えない独立パラメータであるので、その積として定義した。これにより、一部の機能のみが優れていても、総合評価値は高くならず、総合評価値を高める為には、全ての機能を高くする必要があり、総合的な技術進化が期待出来る。尚、立体感に関しては、後述する様に通常の映像であれば、平面映像であってもゼロにならない様に定義したので、この評価式は、立体映像のみならず平面映像の評価にも適用出来る。

(a)　没入感度

没入感（Immersion）は、看視者の視野を覆う様な大画面映像から得られるが、提示される映像の時空間的な滑らかさも重要であるので、これらのパラメータを映像の視野角（VA)、画素密度（PD)、フレーム密度（FD）で表し、次式で没入感度（I）を定義した。

$$I = VA \times PD \times FD \tag{4.2}$$

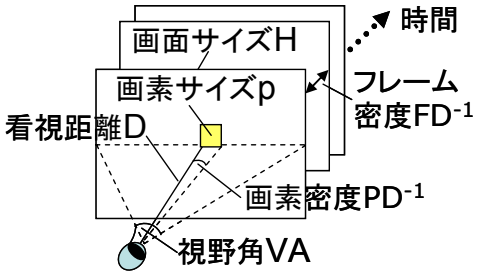


図 4- 2　没入感度のパラメータ

＜視野角 VA＞

映像の大きさは、映像を見る距離により見え方が変わるので、映像のサイズと看視距離とで構成される視野角(VA)で評価する。

$$VA = 2 \times \tan^{-1}(H / 2D)/\mathrm{a} \tag{4.3}$$

ここで、H：画面サイズ(m)、D：映像面までの看視距離(m)である。a=1.0：正規化計数は、没入感が高くなると言われている水平視野角 60 度[71]で、視野角項がほぼ 1.0 となるように決めた。図 4-3 に示す様に、人の両眼の視野角は水平約 220 度、垂直約 130 度の楕円であり、アスペクト比は約 3 : 5 であるが、通常のディスプレイの画面アスペクト比では、水平方向の視野カバー率が低く、又、視差バリヤー方式等で立体映像化する場合には、水平方向の画素密度が低くなる傾向があるので、視野角項は、水平視野角で殆どの場合代表出来る。通常の映像鑑賞では、画面が大きいほど看視距離が大きく取られるので、視野角はほぼ同じになる傾向がある。

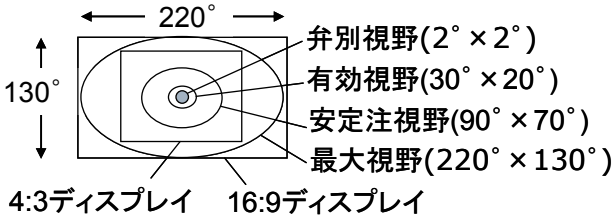


図 4- 3　人の視野角とディスプレイのアスペクト比

＜画素密度 PD＞

映像のきめの細かさは、画素の大きさで決まるが、これも見る距離で変わるので、画素のサイズで構成される見込み角の逆数で表し、視力 1.0 で弁別出来る見込み角（$\tan^{-1}$(1.5mm/5m)）で正規化した。ここで、p：画素サイズ(m)、D：看視距離である。通常のディスプレイでは、画素アスペクト比はほぼ 1 であるので、画素サイズは横方向でも縦方向でも差は少ない。

$$PD = \frac{\tan^{-1}(1.5/5000)}{\tan^{-1}(p/D)} \cong 3\times10^{-4}\times D/p \qquad (4.4)$$

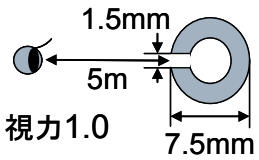


図 4- 4　視力 1.0 の目の分解能

＜フレーム密度＞

映像の動きの滑らかさは、フレームレートで表し、図 4-5 に示す様に、ほぼちらつきを感じなくなるフレーム周波数 60Hz で 1.0 となる様に正規化した。

$$FD = F/60 \qquad (4.5)$$

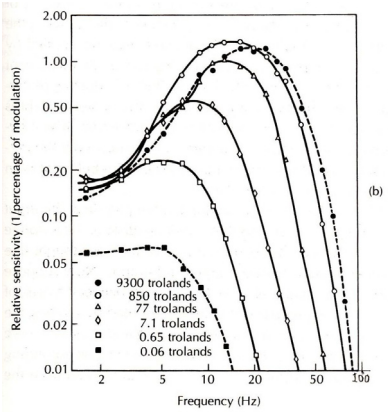


図 4- 5　フリッカ検知限

ここで、F：フレームレート(Hz)である。図 4-5 の縦軸は、正弦波状に明るさが変化する光を見た時に、フリッカーを感じなくなる光の変調度（振幅/平均の明るさ）の逆数を 100 倍してプロットしたものであり、0.01 が変調度 100%に相当する。各グラフは、光の平均の明るさを変えた場合であり、平均の明るさは、網膜に達する光の量（単位：trolands）で表されている。図より 60Hz 以上では殆どの場合フリッカーを感じない。

この様に、映像の総合品質パラメータを視野角と画素密度とフレーム密度の積で表すと、立体映像方式の違いが相殺され、視差バリヤ方式は画素密度に、時分割方式はフレーム密度に反映されるので、没入感度としては差がなくなる。この傾向は、後述の大画面立体映像の主観品質評価でも確認された。

(b)　画質

画質（Quality）は、映像のダイナミックレンジの広さを輝度・色信号の量子化時の雑音量で評価し、純粋さを信号処理系で加わる雑音量で評価すると、両者の和として表現できるが、視覚特性の非線形性を考慮して、次式の様に画質項（Q）を定義した。

$$
\begin{aligned}
Q &= \exp(-NP/b) \\
NP &= 10^{-SNQ/10} + 10^{-SNP/10} \\
SNQ &= 6 \times n + 2 \quad (dB) \\
SNP &= 10 \times \log_{10}\left(\frac{2^n - 1}{N}\right) \quad (dB)
\end{aligned}
\tag{4.6}
$$

ここで、NP：信号電力対雑音電力比、b=0.01：正規化係数（SN20dB 相当、この時 Q≒0.37）SNQ：AD 変換時の信号対雑音比、SNP：信号処理系のピーク信号対雑音比、n：量子化ビット数、N：信号処理系の平均雑音電力である。

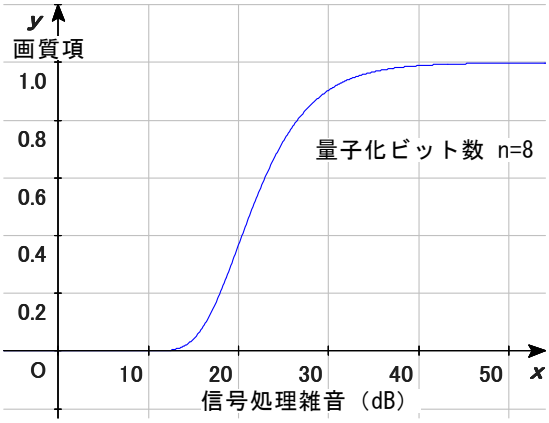


図 4-6　信号処理雑音と画質項の関係

(c)　立体感度

人が映像を見て奥行（立体感）を感じる要因は、両眼視差(B)、輻輳角(C)、焦点調節(A)、運動視差(M)、及び画像要因(P：映像中の物体の大きさ、明るさ、コントラスト等)の５つが主であり、それぞれ看視距離をパラメータとして互いに補間し合っているので[74]、どれか一つが欠けても立体感が失われる事は無いが、全ての要因が揃わないと不自然さが残る。従って、これらの重み付け和として立体感度(T)を定義した。

$$T = \frac{B + 0.04(C + A) + 0.25(M + P)}{1.58} \tag{4.7}$$

ここで、各項の重みは、文献[74][85]に報告されている図 4-7 に示す様な近距離（1m 程度）での感度比を用いた。

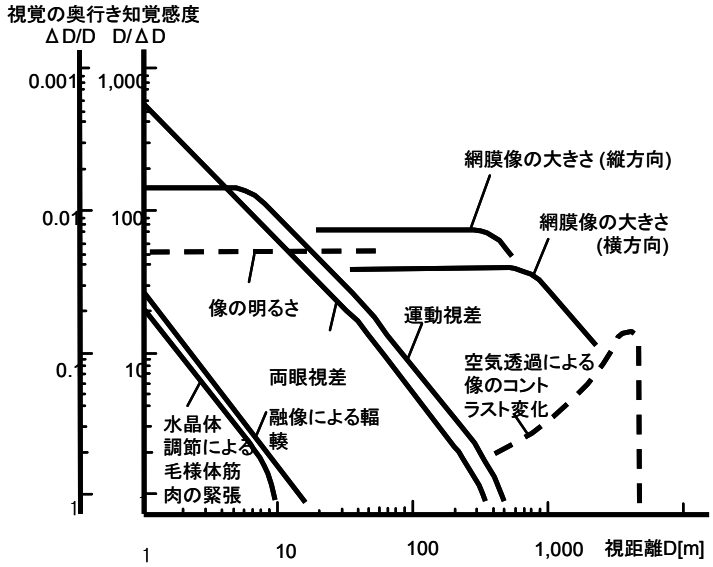


図 4- 7　立体視要因と感度

＜両眼視差 B＞

ある距離の物体を注視した場合、注視点は左右網膜の同一点に写るが、物体に奥行があるとそれに応じて両眼の対応点は反対方向にずれる。このずれ量が小さいと両眼融合が生じ、奥行のある単一の立体像として知覚される。このズレ量が奥行の深さの感覚を与え、立体感を生むので、両眼視差項（B）は、映像面を注視した時に、映像の奥行で生じる網膜上の視差の変化量として次式で定義した。

$$\begin{aligned} B &= r \times \frac{\partial \theta}{\partial D} \times \frac{\Delta}{c} \cong \frac{Lr\Delta}{cD^2} \\ \theta &= 2 \times \tan^{-1}(\frac{L}{2D}) \cong \frac{L}{D} \quad (rad) \end{aligned} \tag{4.8}$$

ここで、r=19mm：眼球のレンズ～網膜間距離、θ=輻輳角、D：映像面までの看視距離(mm)、L=65mm：両眼間距離、 Δ=立体映像の奥行(mm)、 c=0.10(mm)：正規化係数（注視点融合領域[約 15～20 分]での両眼視差変化量）である。両眼融合は、網膜細胞が最も密に分布する中心窩でのみで生じる。この大きさは 0.1mm 程度であるので、両眼視差変化量が 0.1mm 以下であれば、注視点を動かさずに単一物と認識出来る。奥行量がこれ以上になると、融合領域からはみ出し、像が 2 重に見える。これを補正する為に、注視点移動が生じて、立体感要因は輻輳角項や焦点調節項に移行するので、両眼視差項の最大値は 1 とする。この値は、注視点までの距離によって変化するが、次に述べる輻輳角変化を伴う様な通常の立体映像であれば、どの注視点でも飽和値に達している。

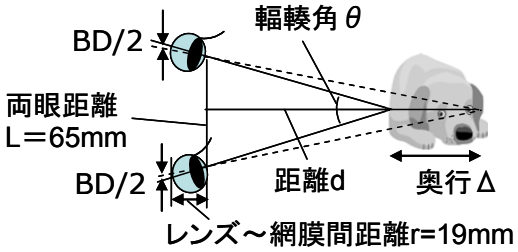


図 4- 8　両眼視差

＜輻輳角 C＞

両眼が注視点を見込む角度を輻輳角と言うが、注視点までの距離によってこの角度が変化すると、毛様体筋に加わる力が変化し、この変化から奥行感が得られるので、輻輳角項 (C) は次式の様に立体映像の最近距離 D_{min} と最遠距離 D_{max} で変化する輻輳角の差分として定義した。

$$C = \frac{(\theta_{max} - \theta_{min})}{d} \cong (\frac{1}{D_{min}} - \frac{1}{D_{max}}) \times \frac{L}{d} \qquad (4.9)$$

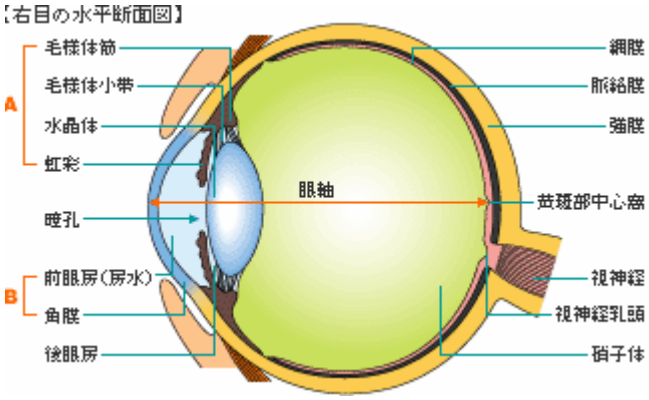


図 4- 9　眼球の断面図

ここで、θ_{max}：最大輻輳角。両眼間距離 L を立体映像までの最短距離 D_{min} で割った値で近似した。θ_{min}：最小輻輳角。両眼間距離 L を映像の最遠距離 D_{max} で割った値で近似。d=0.26：正規化係数（D_{min}=250mm：明視の距離、D_{max}=∞の対象物による輻輳角変化量）である。

＜焦点調節 A＞

注視する物体までの距離が変わると、それに応じて目のレンズの焦点距離が変化する。この変化で奥行を感じるので、焦点調節項（A）は、実際に奥行を持つ立体映像の距離の関数として次式で定義した。

$$A = \frac{(f_{max} - f_{min})}{e} = \frac{r^2(D_{max} - D_{min})}{(D_{max} + r)(D_{min} + r)e} \qquad (4.10)$$

$$f = \frac{D \times r}{D + r}$$

ここで、f_{max}=立体映像の最遠点 D_{max} を見た時の目のレンズの焦点距離、f_{min}=立体映像の最近点 D_{min} を見た時の目のレンズの焦点距離、r＝眼球のレンズ～網膜間距離、e=1.34：正規化係数（D_{min}=250mm、D_{max}=∞の対象物を見た場合の目の焦点距離変化量）である。焦点調節項は、輻輳角項と異なり、単眼でも生じる立体感を表す。輻輳角の違いのみで立体視を行う映像では、この項は 0 となり、回転スクリーンによる立体映像など、真に奥行を持つ映像のみがこの項を持つ。

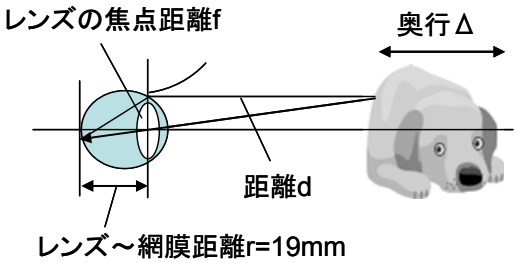


図 4- 10　焦点調節

＜運動視差 M＞

立体は見る角度によって見え方が変わり、この変化で感じる奥行感を運動視差と言う。この効果は片目でも生じるが、これは両眼視差を時間差で感じている事に相当する。運動視差項（M）は対象物の周りを回り込める角度として以下の様に定義した。

$$M = \frac{\phi}{2\pi} \qquad (4.11)$$

ここで、ϕ：視角変化範囲(対象物の周りを回り込める角度(rad))、2π：正規化係数。被

写体の周囲を一周出来る場合に 1.0 とした。運動視差が水平・垂直など多方向にある場合は、最も大きい運動視差の得られる方向で計測する。

図 4- 11　運動視差

＜画像要因 P＞

平面映像であっても、透視図法的な構図であったり、像の大きさ、明るさ、コントラストの変化等で立体感が生じる。これを画像要因と言うが、動画の場合は構図が変化する事により、運動視差と同じ効果が現れる。これらは全て映像の中身に依存し、時事刻々変化するので、これらを総合したものを画像要因項（P）とし、これらの要因の内一つでもあれば 1、特殊な画像で何も立体感要因が無ければ 0 と、2 値化した。

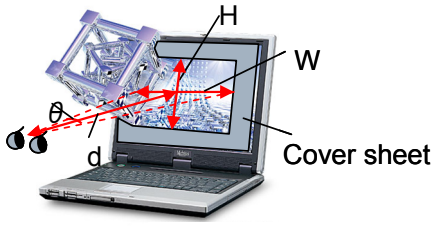
$$P=\begin{cases}1 & (if\ any\ pictorial\ cue\ exists)\\ 0 & (otherwize)\end{cases} \tag{4.12}$$

これより、通常の平面映像は、何らかの画像要因があるので P=1 であり、他の立体感要因がなくても、立体感度は(4.7)式より 0.16 となる。

図 4- 12　画像要因の例

4.2 臨場感度の計測

前節で定義した臨場感度の値が主観値と整合するか否かを確認する為に、以下の基礎実験を行った。実際には、定義式の形を大まかに決めておき、主観評価結果と整合する様にパラメータの詳細を決めて行ったが、そのプロセスを全て記述するのは煩雑であるので、前節では最終の定義式のみを示した。実験には、図 4-13 に示す視差バリヤー方式の 15 インチ両眼ステレオ型立体ディスプレイを用いた。提示映像は、図 4-14 に示す 8 種類の 8 ビット量子化映像であり、35mm の奥行量を、ディスプレイ面を基準としてその前後に均等に付けた。看視距離は、0.5m とし、被験者は、20 代から 60 代の自己申告視力 0.5 から 1.5 の男女 6 名である。テストでは、提示映像の画面サイズ(H)、看視距離(D)、印加雑音量(SNP)、立体の奥行度(Δ)をそれぞれ変えて、通しで 2 回提示し、2 回目に提示された時に、映像から受ける画質や立体感、没入感などを総合して、どれ位実物らしく見えたかの度合を、総合品質評価値(臨場感度)として、VAS(Visual Analog Scale)法[75][76]に基づき、1 から 5 までの目盛りのある枠内に自由にプロットしてもらった。VAS 法は、医学分野で苦痛の評価など参照刺激を与えられない場合に用いられる、単一刺激評価尺度で、全く苦痛を感じない場合を最低値、これ以上無い苦痛と感じる場合を最高値として、その間を連続値で評価するものであるが、本実験の様に、理想映像である実物を提示出来ない場合の評価に適する。看視者には事前に、実物と同じに見えれば 5 の位置に、全く実物に見えなければ 1 の位置に、評価値をプロットする様に依頼した。



諸元	PC-RD1-3D Mebius(Sharp)
立体方式	2視差バリヤー
画面サイズ	15インチ (305x232mm)
解像度	1024x768pel /2plane
フレームレート	60Hz
看視距離	0.5m

図 4- 13　主観評価実験用立体ディスプレイ（外観と仕様）

図 4- 14　テスト画像

以上の様にして求めた主観値を、各テスト条件での臨場感度の計算と比較した。臨場感度の客観値の計算例を以下に示す。

没入感度項 I:

図 4-13 のディスプレイ仕様と(7.2)～(7.5)式より

$$
\begin{aligned}
&H = 305mm,\ D = 500mm,\ F = 60\,fps \\
&p = H/(1024\,pel/2) = 0.596mm/pel \\
&VA = 2\times\tan^{-1}(H/2D) = 0.592 \\
&PD = 3\times10^{-4}\times\tan^{-1}(p/D) = 0.252 \\
&FD = F/60 = 1.0 \\
&I = VA\times PD\times FD = 0.150
\end{aligned}
\tag{4.13}
$$

フレームレートは、静止画の場合フリッカーを感じないので、フレーム密度がデフォルト値 1 となる様にした。

画質項 Q:

映像は 8bit 量子化で印加雑音なしの場合、(4.6)式より、

$$
\begin{aligned}
&n = 8bit,\ N = 0 \\
&SNQ = 6 \times n + 2 = 50 \quad (dB) \\
&SNP = 10 \times \log_{10}\{(2^n - 1)/N\} = \infty \quad (dB) \\
&NP = 10^{-SNQ/10} + 10^{-SNP/10} = 10^{-5} \\
&Q = \exp(-NP/0.01) = 1.0
\end{aligned}
\tag{4.14}
$$

立体感度項 T:

両眼ステレオ立体方式なので、焦点調節項 C と運動視差項 M は 0 であり、(4.7)～(4.12)式より

$$
\begin{aligned}
&L = 65mm,\ r = 19mm,\ \Delta = 35mm,\ D = 500mm,\ c = 0.1mm \\
&B = Lr\Delta/(cD^2) = 1.73 > 1.0 \Rightarrow B = 1.0 \\
&D_{\min} = D - \Delta/2 = 517.5mm,\ D_{\max} = D + \Delta/2 = 482.5mm,\ d = 0.26 \\
&C = (1/D_{\min} - 1/D_{\max})L/d = 0.035 \\
&A = 0,\quad M = 1.0,\quad P = 1.0 \\
&T = \frac{B + 0.04(C + A) + 0.25(M + P)}{1.58} = 0.792
\end{aligned}
\tag{4.15}
$$

看視距離0.5m、映像の奥行量35mmの場合、両眼視差項は1.73となるので飽和値1とした。以上より、臨場感度の計算値 R は以下となる。

$$
R = 100 \times I \times Q \times T = 11.9 \tag{4.16}
$$

(a) 視野角効果

視野角効果は、看視距離を固定して画素密度を不変にし、対角画面サイズ 15 インチ、アスペクト比 4：3 のディスプレイ周辺をカバーして、実質的に 15/11/7.5 インチ相当の画面サイズとした時の、各水平視野角での臨場感度の主観評価値を 8 種の画像毎に 6 名の被験者から得、その平均値と標準偏差を求めた。同時に、各々の条件での臨場感度の計算値を求めた結果を、図 4-15 に示す。図中の黒点と実線は、VAS 法で求めた MOS 値の平均と標準偏差を表し、点線は、後述する回帰分析により求めたロジスティック関数を用いて、臨場感度の計算値を主観値に変換したものである。

主観評価結果より、視野角効果に関しては、30 度程度以上では臨場感度が高まり、それ以下では臨場感が失われる傾向がある事が分かる。この要因は、人が高密度情報を

同時処理出来る有効視野である水平約 30 度、垂直約 20 度[5]の中に、提示映像以外のものが入るか否かであると思われる。

項目	値		
画面サイズ(inch)	7.5	11	15
水平視野角(度)	17.3	25.2	33.9
没入感度	0.076	0.111	0.149
臨場感度	6.58	9.56	11.9
MOS値(平均)	1.78	3.15	3.68
（標準偏差）	0.37	0.77	0.52

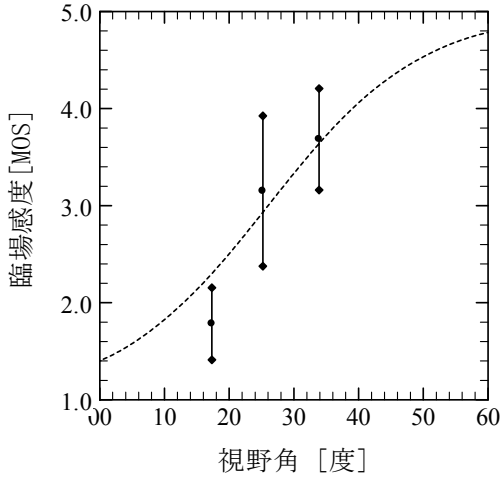


図 4- 15　視野角と臨場感度

(b)　画素密度効果

画素密度効果の測定は、ディスプレイの画素サイズを変える事は困難であるので、看視距離を 0.5m、0.68m と変える事により、画素密度を変化させた。この時の臨場感度の主観評価値を計測した。図 4-16 にその結果と計算値とを示す。立体映像表示の為に水平画素数が半分になっているので、1 画素サイズは 0.6mm 幅である。又、視野角を一定に保つ為、看視距離 0.5m では画面サイズを 11 インチに制限した。

項目	値	
看視距離(m)	0.5	0.68
水平画素密度(rad^{-1})	840	1143
正規化画素密度	0.25	0.34
没入感度	0.111	0.151
臨場感度	8.78	11.9
MOS値(平均)	2.72	3.38
（標準偏差）	0.53	0.9

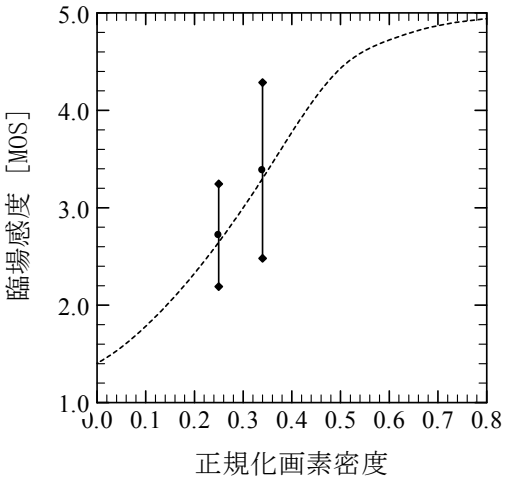


図 4- 16　画素密度と臨場感度

画素密度効果に関しては、視力 1.0 の分解能の 1/2 に近づくにつれ、臨場感が高くなる事が分かる。看視距離 0.5m では画面全体で良好な立体視が得られたが、0.68m では、提示映像周辺部での視差バリヤー位置の不整合が生じ、左右映像のクロストークにまでは至らなかったが一部平面映像化し、立体感が低下したにも拘わらず臨場感が高まった。この要因としては、画素密度の差が弁別出来る高視力視野領域が、視野角 2 度程度の網膜上の中心窩による弁別視野である為、周辺視野での立体感減少が影響しなかったものと思われる。

(c)　画質効果

画質効果は、信号処理系雑音として白色 Gauss 雑音を提示映像に加えて、ピーク信号対雑音比を 50dB(8bit 量子化相当)から、31.2dB、24.6dB、21.3dB と変えた場合の臨場感度の主観値を計測した。、図 4-17 にその主観値計測結果と計算値を示す。

項目	値			
SNQ(dB)	50	50	50	50
SNP(dB)	21.3	24.6	31.2	∞
画質項	0.476	0.706	0.926	1
臨場感度	6.18	9.18	11.1	11.9
臨場感度(平均)	2.21	3	3.76	3.96
(標準偏差)	0.56	0.46	0.48	0.5

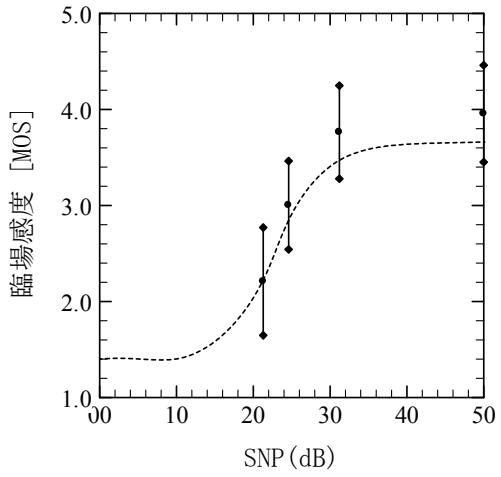


図 4- 17　信号対雑音比と臨場感度

画質（雑音）効果に関しては、重畳雑音比(SNP)30dB 以上で、臨場感度がほぼ飽和した原因は、重畳ノイズが映像にマスキングされて見えなくなった為と思われる。印加雑音比 SNP が低くなると臨場感度が急に下がるのは、雑音の影響が立体映像の奥行方向に非線形に現れ、立体映像の歪みの為にノイズ検知限が上がった為と思われる。

(d)　立体感効果

立体感効果の測定は、立体映像の奥行量を 0mm、17.5mm、35mm と変えた場合の

臨場感度の主観値を計測した。その結果を図 4-18 に示す。

項目	値		
奥行量(mm)	0	17.5	35
視差変化量(mm)	0	0.1	0.2
両眼視差項	0	1	1
輻輳角変化量(度)	0	0.25	0.52
輻輳角項	0	0.017	0.035
立体感度	0.158	0.792	0.792
臨場感度	2.39	10.4	11.9
MOS値(平均)	1.87	2.87	2.87
(標準偏差)	0.57	0.57	0.88

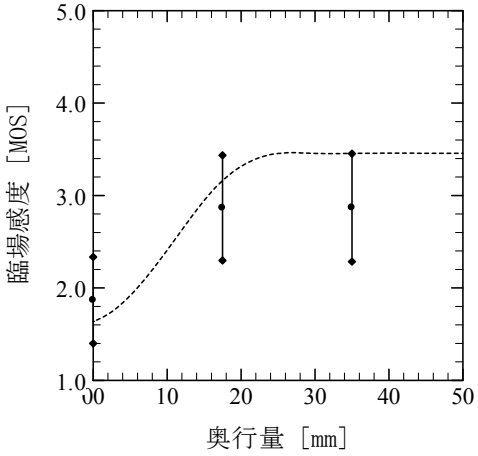


図 4- 18　奥行量と臨場感度

立体感度項に関しては、奥行量 0 での臨場感度の主観値の平均値は 1.87 であったが、画像要因項を 1 とした計算値の臨場感度は図 4-18 の点線が示す様に 1.65 であり、両者はほぼ整合していると思われる。

奥行量が 17.5mm で臨場感度が高まったのは、両眼視差量が両眼の融合限界（注視点付近で約 15～20 分：視差量で約 0.083～0.11mm）に近づいた為と思われ、奥行量 35mm の映像で差が出なかったのは、立体視要因の内、両眼視差項が飽和し、輻輳角項への寄与となったが、その感度が低い為、これらを総合した立体感度値では大きな差が出無かった為と思われる。

又、立体映像の奥行をディスプレイ面より手前に取る場合は、飛び出し量を増すにつれ、映像サイズが変わらないので矮小化して見える箱庭効果があり、逆にディスプレイより奥に表示する場合は、見掛けの距離が遠くなるのに映像サイズが変わらないので拡大化して見える事が知られている。本実験条件の様に表示面の前後に奥行を設ける場合には、突出立体と引っ込み立体による影響は両方とも、臨場感度に反映されているが、映像の看視距離に比して奥行量が十分小さい為、両者の影響はあまり大きくなかったと思われる。

4.3 主観値と客観値の整合化

臨場感度の客観値は、映像の物理パラメータのみで定義されており、個人差や映像コ

ンテンツの違いによる主観の影響を排除して、提示映像の総合品質を客観的に測定するのに適した品質測度である。一方、主観評価値は、絶対値評価が困難な点や、個人差や映像コンテンツの偏りを排除仕切れない側面があるが、客観値計測のパラメータに乗らない品質の差を検出するのに有効である。両者の整合性を確認する事により、客観値計測と主観値計測の両方の妥当性検証が可能になる。

(a)　回帰分析

臨場感度の主観値と客観値の整合性を見る為に、縦軸を主観値とし、横軸を客観値として、前節の主観評価実験で測定した臨場感度の客観値 R と主観値 MOS の散布図をプロットしたものを図 4-19 に黒点で示す。

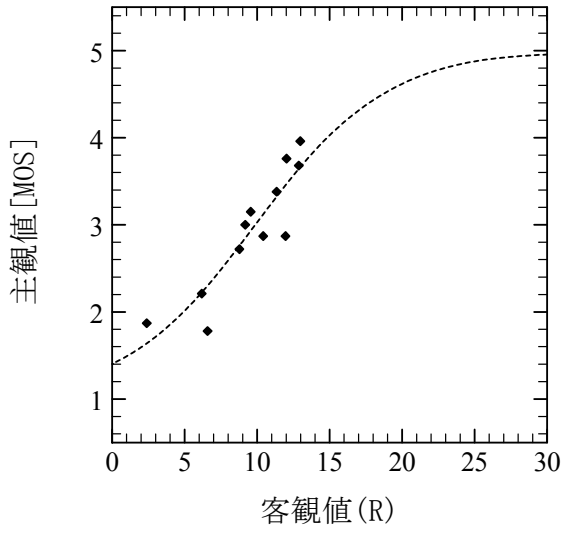


図 4- 19　臨場感度の主観値と客観値の相関

図からもかなり高い相関が見られるが、回帰分析を行って両者の整合度を確認する。一般に線形回帰分析は、(4.17)式に示す様な直線の関数を、散布図の全ての点から最も近くなる様に、最小 2 乗法で求めるものである。

$$MOS_R = \alpha + \beta \times R \tag{4.17}$$

しかし、臨場感度の様な主観値を扱う場合は、分布の両端において飽和現象が起こり分布の形が線形にならない事が知られている。図からもその傾向が伺われるので、ここでは、次式の様なロジスティック関数を用いて分布の形を代表させた。この関数は、R→0 で MOS_R→1 となり、R→大で MOS_R→5 となるので、主観値の傾向を良く表せる。

$$MOS_R = \frac{4}{1+\exp(\alpha - \beta \times R)} + 1 \tag{4.18}$$

この非線形関数は、変形により次の様な線形関数として表せるので、これを用いて線形回帰分析を行った。

$$\ln(\frac{4}{MOS_R - 1} - 1) = \alpha - \beta \times R \tag{4.19}$$

この線形回帰方程式は、(4.19)式の左辺を次式の様に S と置き、

$$S = \ln(\frac{4}{MOS_R - 1} - 1) \tag{4.20}$$

主観値 MOS_R から求めた値 S と、客観値 R を 4.19 式の右辺に入れて計算した値との差を、全てのサンプルに付いて 2 乗和した値が最小になる様にした最小 2 乗法を用いて、和の偏微分がゼロになる条件を解いて下式の様に求められる事が知られている[94]。ここで、C_{RR}, C_{SS} は偏差平方和（分散 σ^2 にサンプル数 N を掛けたもの）と呼ばれる。

$$\begin{aligned}
&S = (\overline{S} - \frac{C_{RS}}{C_{RR}}\overline{R}) + \frac{C_{RS}}{C_{RR}}R \\
&\overline{R} = \frac{1}{N}\sum_{i=1}^{N} R_i, \quad \overline{S} = \frac{1}{N}\sum_{i=1}^{N} S_i \\
&C_{RR} = \sum_{i=1}^{N}(R_i - \overline{R})^2, \quad C_{SS} = \sum_{i=1}^{N}(S_i - \overline{S})^2 \\
&C_{RS} = \sum_{i=1}^{N}(R_i - \overline{R})(S_i - \overline{S})
\end{aligned} \tag{4.21}$$

この式を(4.19)式右辺と比較して、

$$\alpha = \overline{S} - \frac{C_{RS}}{C_{RR}}\overline{R}, \quad \beta = -\frac{C_{RS}}{C_{RR}} \tag{4.23}$$

と、α 及びβが求まる。

以上の回帰分析の計算を、Windows Office Excel の関数機能を用いて求めた結果、α=2.2、β=0.22 となった。この結果を、図 4-19 の点線で示すが、分布の形を良く現している。この回帰分析の精度は、回帰式からの予測値 $\overline{S}$ と実測値 S の相関係数 R を 2 乗した R^2 で評価され、決定係数と呼ばれるる。一般に、決定係数の値は、サンプル数を増すと精度が上がらなくても、増加するので、サンプル数が増しても決定係数の値が増大しない様に、次式で示す自由度調整済み決定係数 $\overline{R^2}$ を用いて評価した。

$$\overline{R^2} = 1 - \frac{N-1}{N-p-1}(1-R^2) \tag{4.24}$$

ここで、N：サンプル数 12、p：説明変量数であり、今は臨場感度の客観値一つなので p=1 である。この値は 0.78 であったが、この値が 0.8 以上の場合、良い近似といわれて

いるので、ほぼ良い精度で客観値と主観値の関係を表している事が確認できた。

(b)　回帰関数の検定

更に、この客観値と主観値の整合関数が、広く一般的に適用出来るか否かの有効性を示す為に、F 検定を行った。F 検定は、主観値 MOS_R の実測値ばらつき(分散) V_T を、(4.18)式で与えられる回帰関数の予測値ばらつき V_R と、回帰関数では予測出来なかった残差ばらつき V_E の和として表せる事を利用し、予測値ばらつき V_R と残差ばらつき V_E の比 $F=V_R/V_E$ の値から、有効性の検定を行うものである。この比は、回帰式の自由度(p=1)と残差の自由度(N-p-1=10)の F 分布に従うので、以下の帰無仮設を立て、これが棄却域に入るか否かを確認する。

帰無仮説 H_0:　回帰式は役に立たない($V_R \fallingdotseq V_E$)。

対立仮説 H_1:　回帰式は役に立つ($V_R > V_E$)。

この回帰分析の結果、F 値は 40.6 であった。一方、自由度 p=1(変量数：臨場感度の客観値)、N-p-1=(基礎実験のサンプル数：12)-(変量数：1)-1=10 の F 分布曲線から得られる、有意水準 5%の棄却域の域値は F 分布の表[77]より 4.96 であるので、回帰式の予測ばらつき V_R は残差ばらつき V_E より十分大きく、帰無仮説は棄却され対立仮説が成立し、臨場感度の回帰関数は有効であり、広く一般に適用出来る事が確認された。

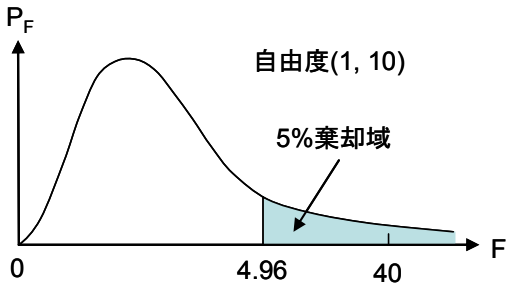


図 4- 20　自由度 1, 10 の F 分布

4.4 マルチ画面 CG 立体映像

前節までは、立体映像の総合品質評価尺度として臨場感度を定義し、その客観値と主観値の整合性を確認したが、以下では、この臨場感度式を用いて各種の大画面立体映像の評価を行い、評価式の有効性を検証する。評価したのは、3D CG 映像をマルチスクリーンに投影する仮想現実空間と、3D CG アニメを分光眼鏡で看視するシステムと、ステレオ立体アニメ映画と、ステレオハイビジョンカメラで撮影した実写風景及び、シネマ用デジタルカメラによる実写立体映像と 3D CG 映像を混合した SF 映画である。これらの評価実験では、システムの展示会を利用して、視聴者にアンケート調査を行い、画質に関する評価項目 6 種に関して、1 から 5 まで目盛りの入った枠内に主観値を自由にプロットしてもらった。同時に参考データとして、立体映像を見る事に起因する疲労感等の心理項目 4 種の評価も行った。評価者は、立体映像フォーラムの呼び掛けで集まった、立体映像に理解と関心のある人達であるが、システムのデモ日時や場所が異なるので、毎回その参加者や人数はばらついた。

本システムは、仮想現実（VR）空間構築用のシステム（Christy HoloStage）で、5 台のコンピュータとプロジェクタを同期運転する事により、看視者の前面・右横面・床面に CG 立体映像を投影し、位置センサの付いたシャッターグラスを掛けた看視者が動く事により、立体映像の見え方が変わると同時に、手に持ったコントローラーで、仮想空間内を自由に移動出来るものである。他の看視者は、通常のシャッタグラスを掛けて代表者が制御する仮想空間を見る。

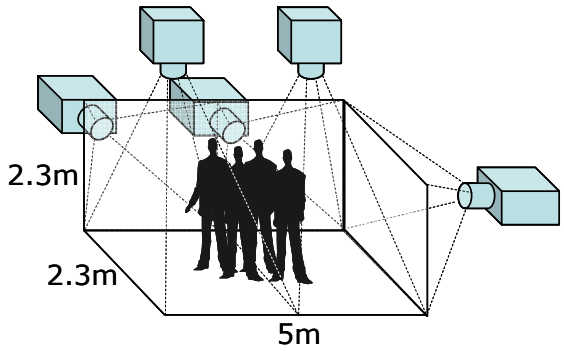


図 4- 21　マルチ画面 3D CG 映像表示システム（HoloStage）

(a)　評価条件

本システムで使用されたプロジェクタの仕様を表 4-1 に示す。投影面は、正面と床面

を夫々２面に等分割し、横面と合わせて５台のプロジェクタでつなぎ投影する。プロジェクタの画面のアスペクト比と分割スクリーンのアスペクト比が合わないので、プロジェクタの縦解像度一杯にスクリーン投影し、横方向の余った画像は切り取られている。従って、正面と床スクリーンの解像度は、1050×2283pel、横スクリーンの解像度は、1050×1050pel となる。ポリゴン数は、提示映像の平均ポリゴン数である。

表 4- 1　マルチ画面 CG 立体映像表示システム(HoloStage)の仕様

諸元	Christy HoloStage
スクリーン構成	プロジェクタ5台で3面につなぎ投射
立体方式	時分割ステレオ
画面サイズ(縦x横x奥行m)	2.3x5.0x2.3
プロジェクタ1台の解像度(縦x横pel)	1050x1400
総ポリゴン数	250,000
フレーム周波数f(Hz)	96
量子化ビット数(bit)	8
信号SN(dB)	50
看視距離(m)	1.15

評価者は、図 4-22 に示す様な 20 代から 60 代を中心とする、視力 0.5 以上が 75%を占める男性 17 名である。

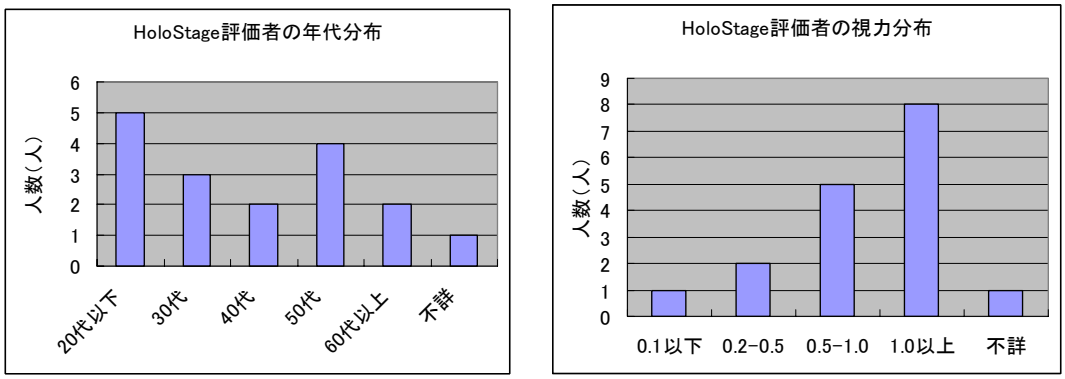


図 4- 22　HoloStage 映像評価者の分布

評価映像は図 4-23 に示す様なビル内外の風景及びアジア地域の気象図の CG 映像である。評価者は、シャッタ眼鏡を掛け床面スクリーンの中央付近に立ち、前方・右横・床面の立体映像を看視する。映像の注視点は、位置センサの付いた眼鏡を掛けた看視者の位置と向きで変化し、視差量はほぼ明視の距離から無限遠まで変化した。

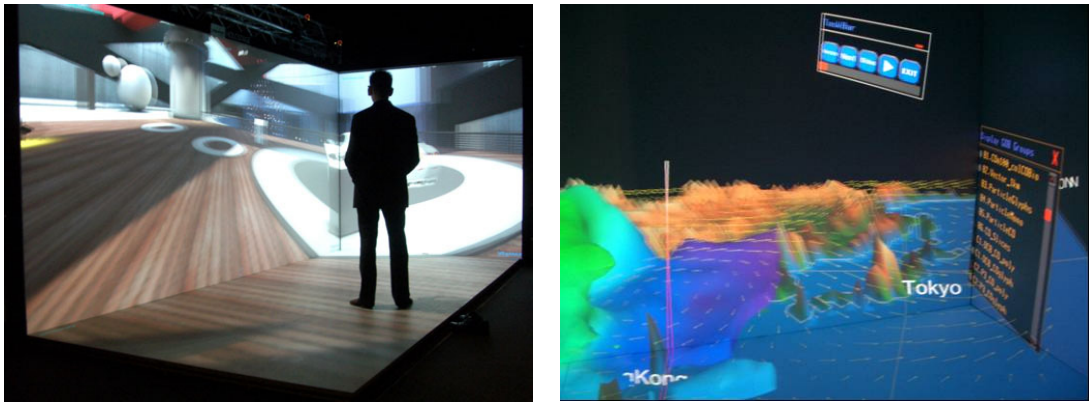


図 4- 23　HoloStage 映像の例

アンケート用紙は、図 4-24 に示す様に、映像品質に関する 6 項目（本物らしら、没入感、きれいさ、細かさ、滑らかさ、立体感）と、心理項目に関する 4 項目（疲労感、眼鏡意識、ノイズ感、違和感）の主観値を 1 から 5 までの枠内に自由にプロット出来る。

見学された立体映像システム毎に，映像がどれくらい実物の様に見えたか，違和感，疲労感などを，1～5までの間で該当すると思われるレベルにチェック（〇印，又は，レ印）を入れて下さい．（中間レベル

システム/映像名（　　）　←非常に強く感じる　全く感じない→

	5	4	3	2	1
本物らしさ（実物の様に見える）					
映像世界の中に居ると感じる					
映像のきれいさ（色，輝度の深さ）					
映像のきめ細かさ					
動きの滑らかさ					
立体感(奥行き感)					
見た後の疲労感					
眼鏡が気になる・煩わしい					
映像のノイズが気になる					
形・見え方の不自然さ・違和感					
	5	4	3	2	1

その他ご感想

図 4- 24　主観品質調査アンケート用紙の例

(b)　評価結果

図 4-25 に評価項目毎の主観値の平均を示す。又表 4-2 に全データを示す。スクリーン映像が看視者の周りを取り巻く立体映像であるので、没入感と立体感に高い評価値が得られたが、提示映像がポリゴン数の余り多くない CG 映像であった事と、気象図は高さ方向が約 100 倍にデフォルメされていた為、本物感はあまり高くならなかった。又、疲労感や眼鏡意識などの心理要因は低い値であった。

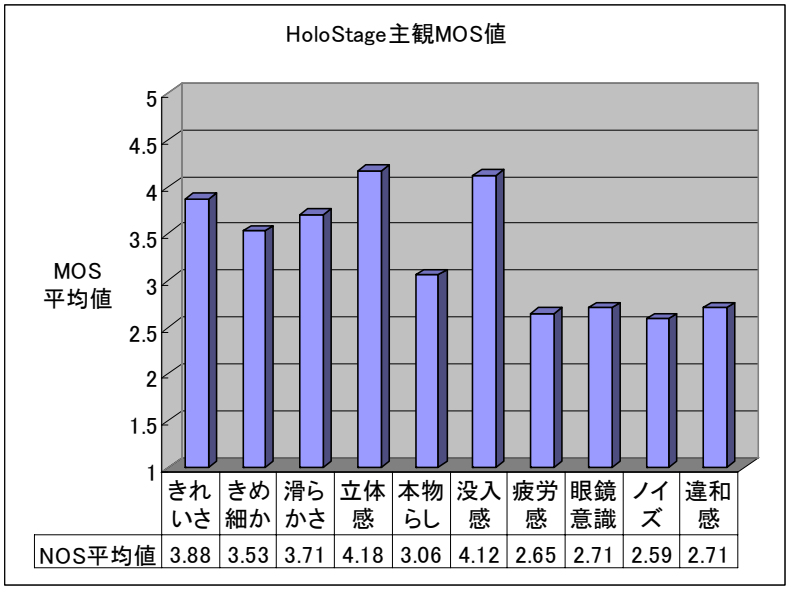


図 4- 25　HoloStage 映像の主観評価結果

表 4- 2　HoloStage 主観評価データ

年代	視力	きれいさ	きめ細かさ	滑らかさ	立体感	本物らしさ	没入感	疲労感	眼鏡意識	ノイズ	違和感	コメント
40	0.2-0.5	4	2	2	4	3	4	1	2	5	3	
60以上	1.0以上	3	2	3	4	2	3	2	3	3	3	CGのビルの中の車
		3	3	4	4	2	4	2	3	3	3	CGのサンデッキ移動
20以下	1.0以上	4	3	5	5	2	4	4	3	2	3	
30	1.0以上	5	5	3	5	3	4	4	3	2	2	没入感が非常に高い
20以下	0.5-1.0	3	4	4	4	5	4	3	2	2	2	
40	1.0以上	4	4	3	4	3	4	2	1	2	3	
20以下	0.5-1.0		4	4	5	4	5	5	3	3	3	スクリーンのエッジが気にな
20以下	0.1以下	5	4	4	4	2	5	4	5	4	4	スクリーンの境界線が気にな
30	1.0以上	5	5	5	4	4	4	2	1	2	1	基本性能がよく分かった
50	0.2-0.5	4	4	5	5	4	5	3	3	1	2	処理速度が速い、気象データソフトが素晴しい
50	1.0以上	4	3	4	4	4	5	2	2	4	3	インタラクティブの為高臨場感が得られた
20以下	1.0以上	4	4	5	4	2	4	4	5	2	2	
60以上	0.5-1.0	4	3	4	4	4	4	1	3	2	3	
50	0.5-1.0	3	3	3	4	3	4	2	3	1.5	3	
30	1.0以上	4	4	3	4	3	5	2	2	2	3	
50	0.5-1.0	3	3	2	3	2	2	2	2	3.5	3	
17名	平均	3.875	3.5294118	3.70588	4.1765	3.058824	4.1176	2.6471	2.705882	2.588	2.7059	
	標準偏差	0.7188	0.8744746	0.98518	0.5286	0.966345	0.7812	1.1695	1.104802	1.049	0.686	

(c)　主成分分析

評価者が、どのような観点からテスト映像を評価したかを知る為に、6 項目の品質評価項目に対する、17 名の評価者の主観値の主成分分析を行った。主成分分析は、複数の説明変数（品質評価項目）がある場合に、その説明変数で構成される多次元空間に評価点をプロットし、評価値の分布の主軸を求めるものであり、説明変量を圧縮合成して、総合評価や分布の特性が見られるものである。主成分分析で得られる第 1 主成分は、分布の分散が最大になる方向（分布の主軸：最も大きな広がりの方向）を示し、第 2 主成分以降は順次、これらと直交して次に大きな広がりの方向を示す。

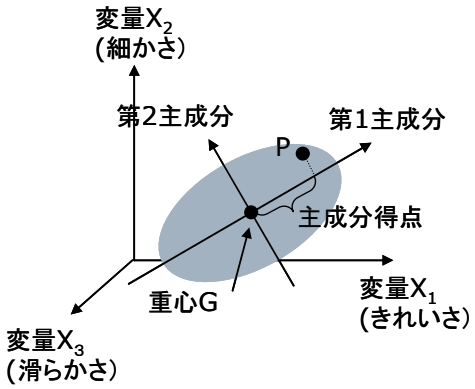


図 4- 26　多変量の分布と主成分

主成分の求め方は、標本空間の重心を通る次式の様な直線を引き、この直線が分布の主軸と重なる様にその傾き（$a_1, a_2, \ldots, a_n$）を求める問題である。

$$Z = a_1 x_1 + a_2 x_2 + \cdots a_n x_n \tag{4.25}$$

この直線は、図 4-26 の点 P で示す各標本値からの距離が最小になる様に求める事により、主成分でこの分布の特徴を現した時に、失われる情報量が最小となる。その為には、図で、各標本値 P から直線に下した垂線の足と分布の重心との距離を主成分得点と言うが、この値が最大になる様に直線の傾きを決めれば良い事になる。この主成分得点の分散は、

$$Q = \frac{1}{N}\sum_{k=1}^{N}\{\sum_{i=1}^{n} a_i x_i(k) - \sum_{i=1}^{n} a_i \overline{x_i}\}^2 = \sum_{i=1}^{n} a_i{}^2 S_{ii} + 2\sum_{i\neq j}^{n} a_i a_j S_{ij}$$

$$S_{ii} = \frac{1}{N}\sum_{k=1}^{N}\{x_i(k) - \overline{x_i}\}^2 \tag{4.26}$$

$$S_{ij} = \frac{1}{N}\sum_{k=1}^{N}\{x_i(k) - \overline{x_i}\}\{x_j(k) - \overline{x_j}\}$$

と表せるので、この Q を最大にする問題となる。ここで、n=6 は説明変量の数で、N=17 はサンプル数であり、S_{ii} は各説明変量の分散であり、S_{ij} はその共分散である。この問題は、Lagrange の未定係数問題として解けるので、次の拘束条件を設定し、

$$\sum_{i=1}^{n} a_i{}^2 = a_1{}^2 + a_2{}^2 + \cdots + a_n{}^2 = 1 \tag{4.27}$$

この条件下で、次式の評価関数 G の極大点を求めれば良い。

$$G = Q - \lambda(\sum_{i=1}^{n} a_i{}^2 - 1) \tag{4.28}$$

この式の極大点では、各係数による傾きが 0 になるので、これを編微分した次式を満たす。

$$\frac{\partial G}{\partial a_i} = 2a_i S_{ii} + 2\sum_{j\neq i}^{n} a_j S_{ij} - 2\lambda \sum_{i=1}^{n} a_i = 0$$
$$\frac{\partial G}{\partial \lambda} = -(\sum_{i=1}^{n} a_i{}^2 - 1) = 0 \qquad (4.29)$$

この第 1 式を行列式で表すと、

$$\begin{bmatrix} S_{11}-\lambda & S_{12} & \cdots & S_{1n} \\ S_{21} & S_{22}-\lambda & \cdots & S_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ S_{n1} & S_{n2} & \cdots & S_{nn} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \cdots \\ a_n \end{bmatrix} = 0 \qquad (4.30)$$

となる。この式がある a_i の組み（直線の傾き）で成立する為には、[a_i]の係数が 0 になれば良いので、この条件を次式の様に表し、これを満たす λ を求める。

$$\begin{bmatrix} S_{11} & S_{12} & \cdots & S_{1n} \\ S_{21} & S_{22} & \cdots & S_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ S_{n1} & S_{n2} & \cdots & S_{nn} \end{bmatrix} = \lambda \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 1 \end{bmatrix} \qquad (4.31)$$

この式は固有方程式と呼ばれ、(4.29)の第 2 式の条件の下に、この式を解いて得られる λ を固有値、その時に得られる ai の組[a_1, a_2, ...a_n]^T を固有ベクトルと言う。この式は、べき乗法により解ける。べき乗法は、上式の両辺に任意の初期値（通常は all 1）を仮定した固有ベクトル $\mathbf{x}$=[a_1, a_2, ...a_n]^T を掛けた次式において、

$$\begin{bmatrix} S_{11} & S_{12} & \cdots & S_{1n} \\ S_{21} & S_{22} & \cdots & S_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ S_{n1} & S_{n2} & \cdots & S_{nn} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \cdots \\ a_n \end{bmatrix} = \lambda \begin{bmatrix} a_1 \\ a_2 \\ \cdots \\ a_n \end{bmatrix} \qquad (4.32)$$

$\boldsymbol{x}$=[1, 1,⋯1]^T と置いて、左辺の値[S]$\boldsymbol{x}$=[a_1, a_2,⋯a_n]^T を求める。ここで、[S]は(4.32)式の左辺第 1 項である。得られた a_i のうち、その最大値で、各 a_i を割る。

この様にして得られた[a_1', a_2',...1,...a_n']^T を新しい $\boldsymbol{x}$ と見なして、その値[S]$\boldsymbol{x}$ =[a_1, a_2,...a_n]^T を求め、得られた新しい a_i の最大値で各 a_i を割る。これを繰り返して、a_i の最大値が、一定の値に収束すれば（変化分が 10^{-5} 程度）、その時の最大値 a_i が第 1 固有値 λ_1 になる。これより、固有ベクトル $\boldsymbol{x}_1$ は、その時の $\boldsymbol{x}$=[a_1, a_2,...a_n]^T を標準化したものになる。標準化は、

$$x_1 = \frac{x-\bar{x}}{\sigma} = \frac{x-0}{\sqrt{\sum_{i=1}^{n} a_i{}^2}} \qquad (4.33)$$

で行える。

第 2 固有値は、$[S_1]=[S]-\lambda \boldsymbol{x}_1 \cdot \boldsymbol{x}_1^T$ として、[S]から第 1 主成分を引き去ったものを、新しい[S]と見なし、同様にして、$[S]\boldsymbol{x}=[a_1, a_2,\ldots a_n]^T$ を満たす $\boldsymbol{x}$ を求める。

得られた a_i の最大値が、第 2 固有値 λ_2 になる。この時の $\boldsymbol{x}$ を標準化したものが、第 2 固有ベクトル $\boldsymbol{x}_2$ である。以下、同様に第 n 主成分まで求める事が出来るが、第 1 主成分が最も良く分布の性格を表し、第 2 主成分以下、順に寄与率は少なくなる。

得られた主成分が、元の説明変数とどれ位関係があるかを見る為に、主成分負荷量を次式の様に求めると、

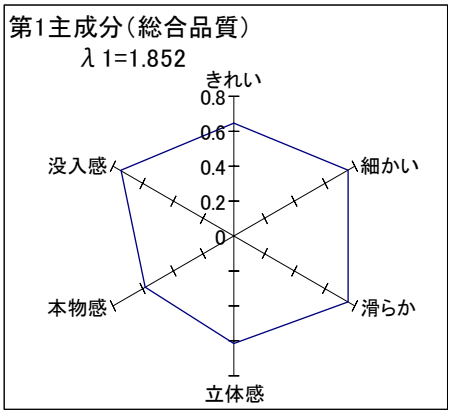
主成分負荷量＝√固有値×固有ベクトル/√説明変量の分散　　(4.34)

この値は、主成分と各説明変量の相関係数に等しく、値が 1 に近い程、関係が深い事を示す。

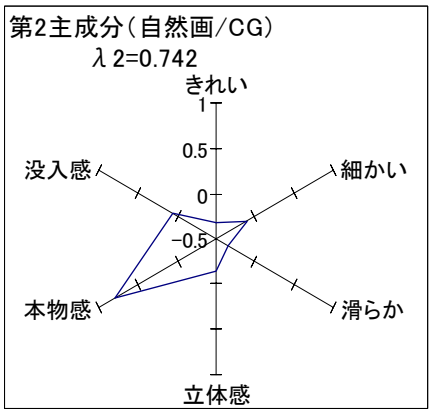
以上の議論より、表 4-2 に示す映像品質に関する 6 項目の説明変量の主成分分析を Excel で行った結果を表 4-3 及び図 4-27 に示す。

表 4- 3　HoloStage 主成分分析結果

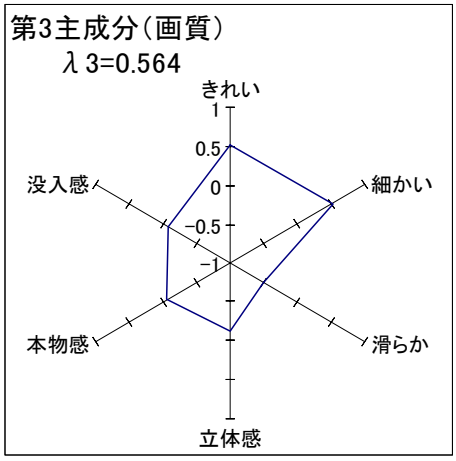
主成分n	1		2		3		4		5	
固有値λ	λ 1= 1.852		λ 2= 0.742		λ 3= 0.564		λ 4= 0.39		λ 5= 0.146	
固有ベクト	固有ベクト	負荷量	固有ベクト	負荷量	固有ベクト	負荷量	固有ベクト	負荷量	固有ベクト	負荷量
きれい	0.332	0.649	−0.265	−0.327	0.483	0.521	0.239	0.214	0.061	0.033
細かい	0.472	0.757	−0.106	−0.108	0.572	0.507	−0.456	−0.335	−0.067	−0.03
滑らか	0.527	0.75	−0.397	−0.358	−0.646	−0.507	−0.33	−0.216	−0.09	−0.036
立体感	0.231	0.613	−0.086	−0.144	−0.086	−0.125	0.275	0.334	0.892	0.664
本物感	0.402	0.584	0.867	0.797	−0.092	−0.074	−0.143	−0.095	0.078	0.032
没入感	0.417	0.749	0.042	0.0481	−0.08	−0.079	0.728	0.6	−0.427	−0.215
意味	総合品質		本物らしさ		画質		没入感		立体感	



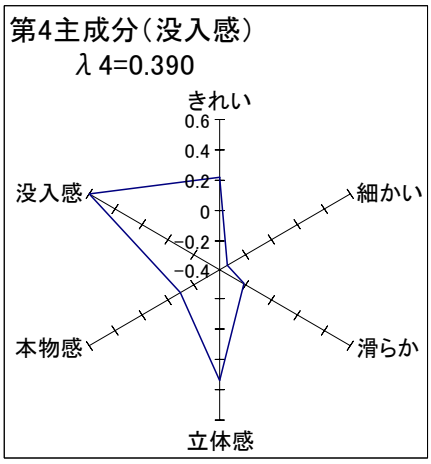
(a)第 1 主成分



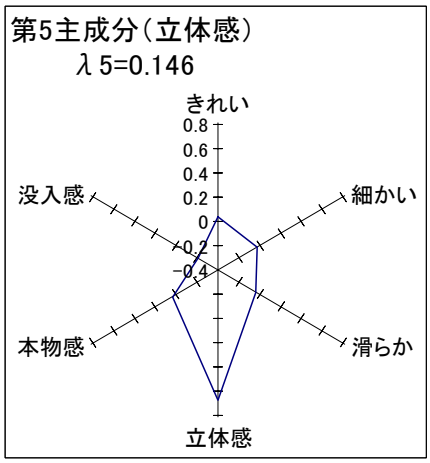
(b) 第 2 主成分



(c) 第 3 主成分



(d) 第 4 主成分



(e) 第 5 主成分

図 4- 27　HoloStage の主成分負荷量

これより、第 1 主成分が全ての評価項目をほぼ満遍なく含む事から、総合品質と見なせる事が分かる。本物感が第 2 主成分として求まったが、これは提示映像が CG 合成映像であった為、あくまで写実にこだわった評価者と、あり得る仮想現実として受け入れた評価者の間で、本物らしさの評価が大きく分かれた事を現していると思われる。又、第 3 主成分以降は、それぞれレベルは低いが、画質、没入感、立体感に対応し、これらがほぼ独立している事を示している。これらの主成分分析の結果から、映像の総合品質の主観評価値は、没入感・画質・立体感などに関する全ての評価値の平均値で表すのが良いと思われる。

(d)　臨場感度評価

以上の検討結果より、6 項目のアンケート形式による評価値の平均値 3.74 を臨場感度

の主観値とみなした。一方、臨場感度の定義式より HoloStage の臨場感度の客観値を求めると、表 4-4 の様になる。

表 4- 4　HoloStage 臨場感度の客観値

項目	値
水平視野角(rad)	3.14
水平画素密度(rad^{-1})	512
フレーム密度(f/s)	48
没入感度項 I	0.386
量子化雑音(dB)	50
信号処理雑音(dB)	∞
画質項 Q	0.999

項目	値
両眼視差	1
輻輳角	1
焦点調節	0
運動視差	0.7
画像要因	1
立体感度項 T	0.93
臨場感度 R	35.73

ここで、水平視野角は、床面の中央に立った評価者が正面と横の画面を見込む角度として求め、水平画素密度は、評価者は主に正面画面を見ているので、この画面上での 1 画素のサイズから求めた。横面や床面では看視距離は正面より遠くなるが、画素密度項を掛けると距離項は相殺され同じ没入感値になる。フレーム密度は、シャッターグラスによる時分割多重型ステレオ立体映像であるので、プロジェクタのフレーム周波数の半分とした。画質項は、映像が RGB 各 8bit の原画であるので、量子化雑音は 8bit 相当とし、信号処理雑音は 0 とした。立体感項は、提示映像の奥行が明視の距離（25cm）程度から無限遠まで変化するので、両眼視差項を飽和値 1.0 とし、輻輳角項は、正規化値と同じになった。運動視差は、画面のコーナー付近の立体映像の周りを回りこめる角度として求めた。焦点調節項は、ステレオ立体であるので、0 とし、画像要因は 1.0 とした結果、臨場感度の客観値は、35.73 となった。この値の主観値との整合性については、4.8 節で検討する。

(e)　心理要因分析

主観評価項目の内、立体映像を見る事による疲労感、眼鏡意識（眼鏡が気になる）、ノイズ感（映像のノイズが気になる）、違和感（過度の立体化等による映像の不自然さ）の 4 項目(x_i)が、きれいさ、立体感等の画質項目や、年齢、視力等(y_i)と関係があるか否かを調べる為に、各評価項目の相関係数を求めた。相関係数($R_{x,y}$)は、次式により求められる。

$$R_{xy} = \frac{Cov(x,y)}{\sqrt{Var(x) \times Var(y)}}$$

$$Cov(x,y) = \frac{1}{N}\sum_{i=1}^{N}\{(x_i - \bar{x}) \times (y_i - \bar{y})\} \tag{4.35}$$

$$Var(x) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2, \quad \bar{x} = \frac{1}{N}\sum_{i=1}^{N}x_i$$

評価結果を表 4-5 に示し、相関の高いものの散布図を図 4-28 に示す。年代と疲労感の間にかなり強い負の相関が見られるが、これは、看視者が立体映像に関心の高い人々であった為、年配者は立体映像を長年見慣れており、輻輳と焦点調節を容易に調和させることが出来たが、若年者はあまり見慣れていない為それが困難であった事が原因の一つと思われる。疲労感と眼鏡意識の間には正の相関が見られるが、これは、眼鏡を掛ける事の煩わしさが疲労感を助長しているものと思われる。又、立体感と疲労感の間に正の相関が見られるが、違和感との間には有意な相関が見られなかった。この理由としては、立体感を強く感じる人は、両眼視差方式の輻輳と焦点調節の不整合の調整努力を無意識下に行っている為に、違和感としては認識されず、疲労感として認識されたものと思われる。又、映像の細かさと疲労感の間にも正の相関が見られたが、これも映像の細部を見分けようとする努力が疲労感につながったものと思われる。

きめ細かさときれいさの間に相関が見られるが、これは、きめの細かさが見分けられる人は、色や輝度の広がり滑らかさも敏感に感じられる傾向があるものと思われる。又、没入感と立体感の間にも相関が見られるが、大画面になると映像の奥行を強く感じる傾向が現れているものと思われる。これらの傾向は、図 4-28 に示した主成分分析の第 3、第 4 主成分にも表れている。

表 4- 5　品質と心理要因の相関

	きれいさ	きめ細か	滑らかさ	立体感	本物らし	没入感	疲労感	眼鏡意識	ノイズ感	違和感	視力
きめ細かさ	0.622										
滑らかさ	0.309	0.41									
立体感	0.417	0.326	0.466								
本物らしさ	0.096	0.331	0.216	0.223							
没入感	0.496	0.361	0.454	0.552	0.404						
疲労感	0.359	0.5	0.447	0.613	−0.091	0.322					
眼鏡意識	0.112	−0.023	0.318	0.201	−0.451	0.187	0.543				
ノイズ	0.011	−0.497	−0.457	−0.368	−0.252	−0.09	−0.177	0.024			
違和感	−0.214	−0.558	−0.413	−0.193	−0.444	0.069	−0.06	0.291	0.516		
視力	−0.202	0.103	0.034	−0.021	−0.024	−0.256	−0.067	−0.333	−0.41	−0.326	
年代	−0.427	−0.56	−0.377	−0.334	0.029	−0.353	−0.731	−0.257	0.034	0.155	0.062

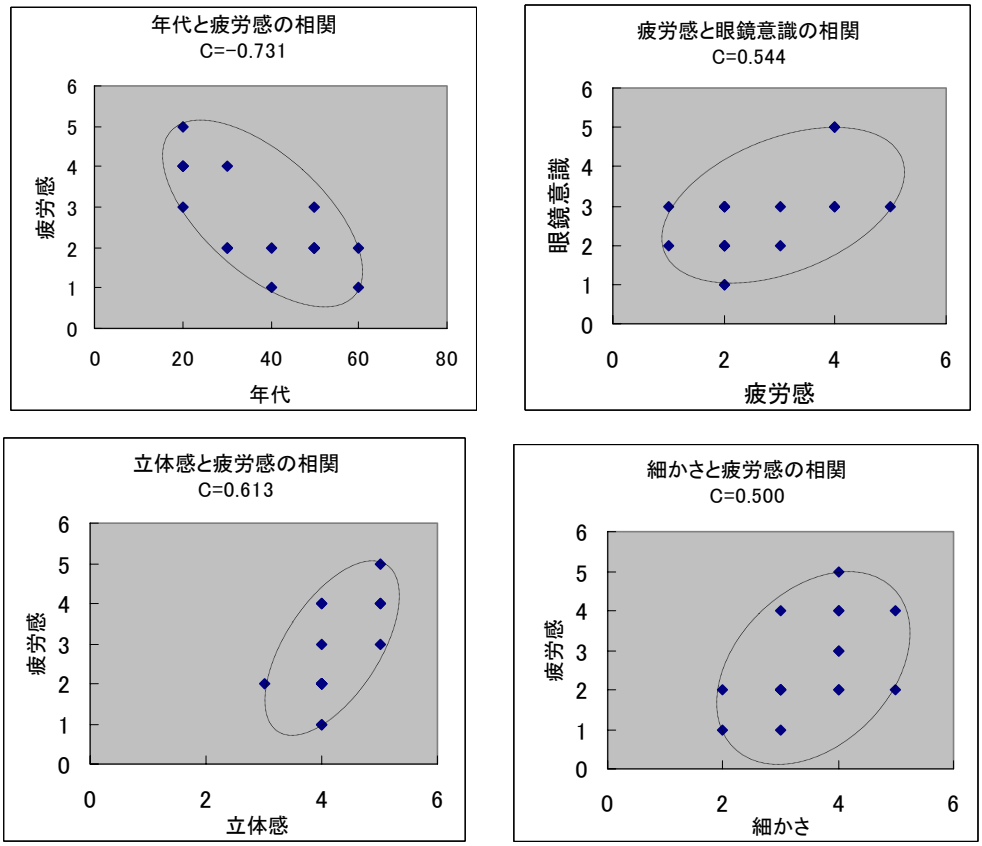
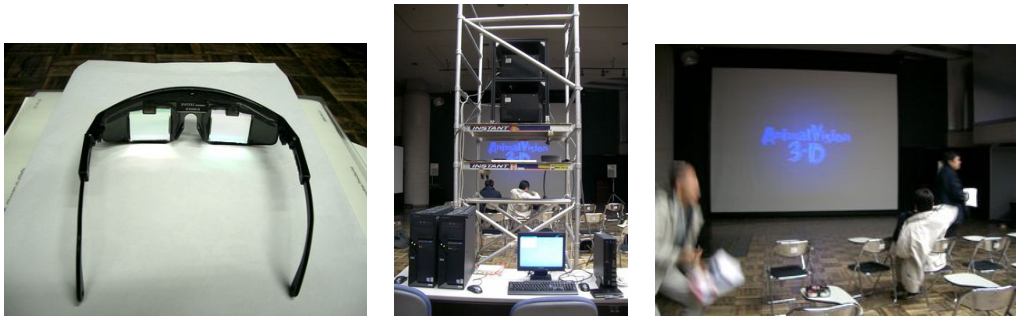


図 4- 28　立体映像から受ける心理要因の相関

以上の結果より、望ましい立体映像としては、裸眼方式が理想であるが、眼鏡方式の場合は、軽く負担の少ない眼鏡が望まれる。例えば、日本人の眼鏡人口は 60%と言われているので、立体映像方式を標準化出来れば、それに合わせて偏光コーティングをあらかじめ各個人の眼鏡に施しておけば、自分の眼鏡の上に別の眼鏡を掛けると言う不自然さが解消される。健常視力者は、立体視の為に新たに眼鏡を掛ける必要があるが、戸外でステレオ音楽を聴く為にヘッドホンを掛ける事がファッションになり、誰も違和感を訴えない様に、立体映像を見る為に眼鏡を掛ける事が、ファッションになるかも知れない。又、立体映像の作り方も、過度の立体化や微細化を抑えて、立体視の 5 要因を満たした目に優しい映像作りが望まれる。

4.5 分光フィルタ式大画面立体 CG 動画

次に評価したのは、2 台のプロジェクタに分光フィルタを付け、左右眼映像を光の波長で多重化したステレオ立体映像表示システムである。



(a) 分光眼鏡　　(b) プロジェクタと PC　　(c) スクリーン

図 4- 29　分光フィルタ式大画面立体 CG 動画表示システム（Galaxy）

(a)　評価条件

本システムで使用されたプロジェクタの仕様を表 4-6 に示す。夫々のプロジェクタの映像は、図 4-30 に示す分光フィルタで互いに重ならない 3 帯域の波長に分割され、通常のホワイトスクリーンに投影される。左右眼に入る光の色は異なっているが、等色条件を満たす様に、左右眼映像の色調があらかじめ調整されているので、自然なカラー映像が看視される。看視者は、プロジェクタ側のフィルタ分光特性と同じ分光フィルタ眼鏡を掛け、ステレオ立体映像を看視する。プロジェクタの画面アスペクト比は 3 : 4 であるが、映像のアスペクト比は 9 : 16 であったので、プロジェクタの縦解像度 1050pel の内、実際に映像に使用されたのは 787pel であった。

表 4- 6　分光フィルタ方式大画面立体 CG 動画システム（Galaxy）の仕様

諸元	Barco Galaxy
スクリーン構成	プロジェクタ2台で 1面に重ね投射
立体方式	分光眼鏡ステレオ
画面サイズ(縦x横x奥行m)	3.375x6
プロジェクタ1台の解像度 (縦x横pel)	1050(実使用787)x1400
総ポリゴン数	不明
フレーム周波数f(Hz)	60
量子化ビット数(bit)	8
看視距離(m)	6
信号SN(dB)	40

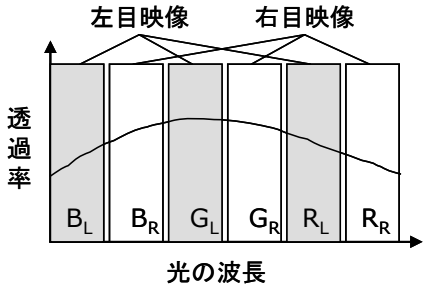


図 4- 30　分光フィルタの仕組み

評価者は、図 4-31 に示す様な女性 6 名を含む、20 代から 60 代にほぼ正規分布する、視力 0.5 以上が 80%を占める評価者グループ 53 名である。

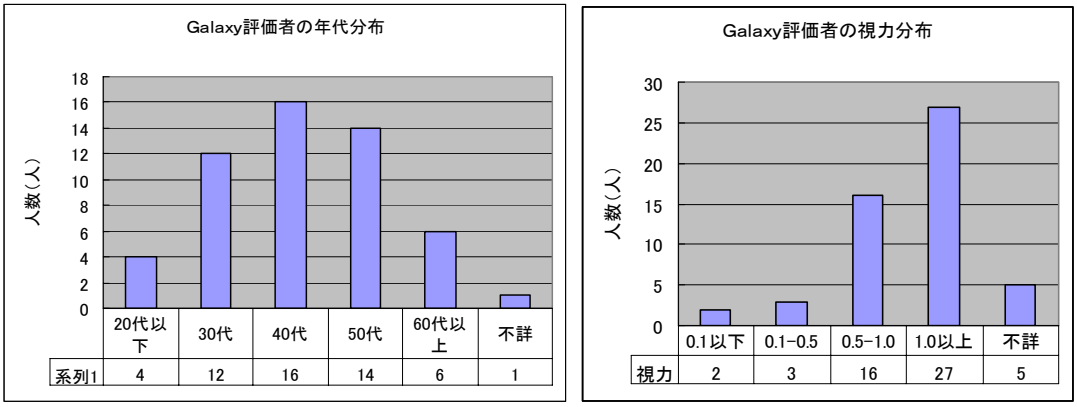


図 4- 31　Galaxy 映像評価者の分布

評価映像は、図 4-32 に示す様な、海や森の生態を現した 3D CG アニメである。評価者は、分光眼鏡を掛け、スクリーンから約 6m 離れた椅子に座って、約 12 分の立体映像を看視した。映像の奥行はスクリーンまでの距離の約半分から無限遠まで変化した。

図 4- 32　Galaxy 映像の例（実際に投影されたものはステレオ化されている）

評価項目と評価方法は、4.4 節と同じ画像品質 6 項目と、心理要因 4 項目の絶対評価である。

(b)　評価結果

主観評価結果の項目毎の平均を、図 4-33 に示す。又、全データを表 4-7 に示す。立体感は高い評価が得られたが、スクリーンまでの距離が比較的大きかったので、没入感はやや高い程度である。映像は、細部まで描かれた CG 映像であったが、動物が擬人化されていたので、本物らしさの評価はあまり高くならなかった。又、小さなオブジェクトが高速に動くシーンでは、フレームのずれがあるのか両眼融合を行う事が困難で立体視がくずれた為、滑らかさも余り高い評価にならなかった。眼鏡があまり普及していない分光フィルタ方式であった為、眼鏡意識がやや高くなった。

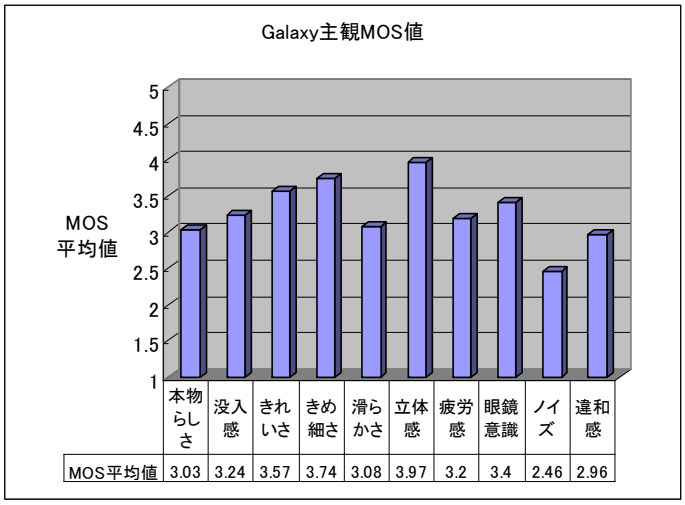


図 4- 33　Galaxy 映像の主観評価結果

表 4- 7　Galaxy 映像の主観評価データ

年代	視力	本物感	没入感	きれいさきめ細さ	滑らかさ	立体感	疲労感	眼鏡意識	ノイズ感	違和感	コメント
		2.5	3	4.5	5	3.5	5	4	5	3	3	
40	1.0～	3	3	3	4	2	4	4.5	5	2	3	
60～	1.0～	3	3	5	5	3	3	4				
50	0.5-1.0	3	4	5	5	2	4		1	3	2	
40	0.5-1.0	3	2	3	3	1	3	5	5	2	4	動きの速いものがぶれる
40	1.0～	3	3	4	4	4	4	5	5	3	3	画面が暗い
30	0.1-0.5	3	3	2	3	3	4	4	4	2	3	
～20	1.0～	4	3	4	4	3	4	5	4	2	4	
60～	1.0～	4	4	4	5	5	5	4	2	2	2	
30	～0.1	3	2	3	3	4	4	2	5	1	2	
40	1.0～	3	3	2	3	3	4	2	5	3	2	
30	0.1-0.5	3.5	3.5	4.5	5	3	3.5	2.5	5	1.5	3.5	
30	～0.1	4	3	4	4	5	5	2	4	3	2	
30	1.0～	3	4	3	3	3	4	3	3	4	3	
50	0.5-1.0	2	3	3	2	3	3.5	2	3	3	3	
40	0.5-1.0	4	4	4	4	3	4	1	2	3	3	
50	0.5-1.0	3	3	3	3	3	5	2	5	4	3	
40	1.0～	2	3	4	5	3	4	3	4	2	3	慣れるまで時間が掛かる
～20	1.0～	3	3	3	3	3	4	2	3	2	2	近すぎると見にくかった
40	0.5-1.0	1	2	2	2	2	3	4	3	4	5	
40	0.5-1.0	1	2	2	2	2	3	4	4	4	4	
60～	1.0～	4	4	2	3	2	5	4	4	2		
30	1.0～	2	2	3	3	2	4	4	3	4	3	
～20	1.0～	3	4	4	4	4	4	5	2	2	4	立体度強く目に負担あり
50	1.0～	3	3	4	4	3	3	4	5	1	3	
30	0.5-1.0	3	4	4	3	2	5	4	3	4	3	
30	1.0～	3	4	3	4	3	4	3	4	3	2	
40	1.0～	3	5	5	5	5	5	2	3	2	1	素晴らしいコンテンツ
30	1.0～	3	3	3	3	3	4	2	3	3	3	
40		4	3	4	4	3	4	1	2	2	3	シルバースクリーンより違和感少
50	0.5-1.0	3	3	5	5	4	3	3	3	2	3	
30	1.0～	4	4	5	5	5	5	3	3	2	2	
40	0.5-1.0	3	4	4	4	5	4	4	1	2	3	
50	1.0～	3	4	4	4	2	4	2	3	3	3	
50	1.0～	3	3	4	3	3	3	3	3	1	3	
30	1.0～	3	3	4	4	4	4	4	4	3	3	
30	1.0～	2				2	4				3	本物感はコンテンツに依存
40	0.5-1.0	2	2	2	4	4	4	3	5	3	3	
60～	1.0～	4	4	4	5	3	3	3	3	1	5	左右画像で1-2コマずれた感じ
50	1.0～	4	4	4	4	4	4	4	4	4	4	
40	0.5-1.0	4	4	2	3	2	5	4	4	2	3	
40	1.0～	3	4	5	5	3	5	4	2	1	2	
50	1.0～	3	3	4	4	3	4	3	3	3	3	
40		3	4	4	3	3	3	4	5	3	3	
50	1.0～	3.5	4	4.5	4.5	4.5	4.5	2.5	2	1.5	3	突出量多いと焦点合せ難い
40	0.5-1.0	3	3	4	4	3	4	2	3	1	1	
60～		2	3	3	3	3	4	4	4	3	4	
60～		2	2	3	3	3	3	4	4	3	4	
50	0.5-1.0	3	4	3	3	2	4	3	4	3		
～20	1.0～	4	3	4	4	2	4	4	2	2	2	
50	0.5-1.0	3	3	3	3	3	4	1	1	2	3	
50	0.5-1.0	3	1	2	3	2	3	1	1	1		
50	0.1-0.5	4	4	4	4	3.5	4	3.5			3.5	通常眼鏡使用
41.923	0.83229	3.0283	3.2404	3.5673	3.74038	3.08491	3.9717	3.1961	3.4	2.46	2.9592	平均値
		0.73	0.7827	0.9184	0.87161	0.94429	0.64611	1.1139	1.21218	0.92494	0.8344	標準偏差

(c)　臨場感度評価

4-4 節と同様に、映像品質 6 項目の主観値の平均値を求めると、3.44 となるが、これを Galaxy 映像の臨場感度の主観値とみなす。他方、臨場感度の定義式より、臨場感度の客観値を求めると、表 4-8 の様になる。

表 4- 8　Galaxy 臨場感度の客観値

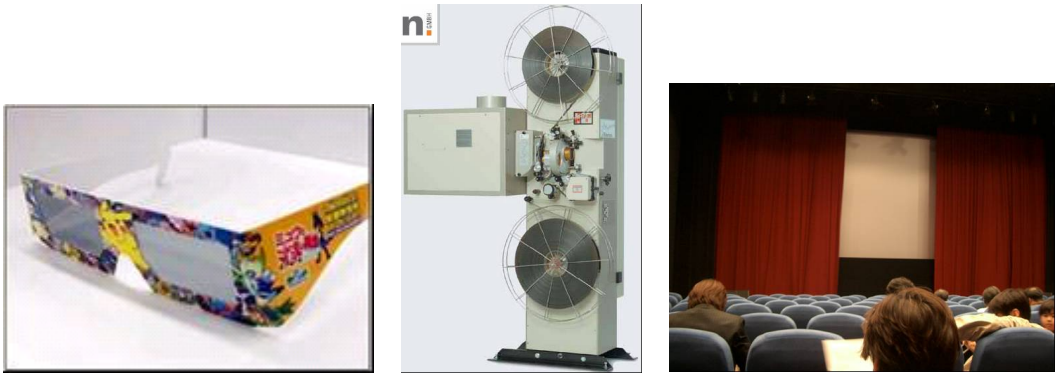
項目	値
水平視野角(rad)	0.927
水平画素密度(rad^{-1})	1400
フレーム密度(f/s)	60
没入感度項 I	0.389
量子化雑音(dB)	50
信号処理雑音(dB)	40
画質項 Q	0.99

項目	値
両眼視差	1
輻輳角	0.083
焦点調節	0
運動視差	0
画像要因	1
立体感度項 T	0.793
臨場感度 R	30.59

これより、画面は大きいが、看視距離が画面幅と同じであるので、水平視野角はほぼ正規化レベルの値となった。逆に画素密度は高い値となったが、没入感項としては、両者はキャンセルしあう。フレーム密度は、左右眼に夫々1 台のプロジェクターを使うので、フレームレートのままである。画質項は、提示画像が 15Mbps の MPEG-2 に圧縮された映像であった事から、信号処理雑音を 40dB とした。立体感項は、提示映像の奥行がスクリーンまでの距離の半分（3m）程度から無限遠まで変化したので、この値から求めた。左右眼別映像による立体であるので、焦点調節と運動視差は 0 とした。画像要因は 1.0 とした結果、臨場感度の客観値は、30.59 となった。この値の主観値との整合性についても、4.8 節で検討する。

4.6 大画面立体アニメ

ここで評価したのは、35mm フィルムを上下 2 分割して撮影した左右眼映像を、フィルム映写機に掛け、上下の映像を夫々直交した偏光フィルタに通した後に、光学的に多重化してスクリーンに投影する立体映画である。看視者は、簡易な紙製の偏光眼鏡で映像を看視する。



(a) 偏光眼鏡　　(b)フィルムプロジェクタ　　(b) シアター

図 4- 34　大画面立体アニメ映画館（3D Theater）

(a)　評価条件

本システムで使用されたプロジェクタの仕様を表 4-9 に示す。プロジェクタは、Kinoton 社の通常のフィルム映写機 FP30D であるが、レンズ部分が立体映画用に取り替えられており、フィルムの各コマ上下の左右映像に偏光を加えて多重化し投影する。スクリーンは、横 8m、縦 4.4m のビスタサイズのシルバースクリーンである。視聴位置は、最前席がスクリーンから約 2.5m、最後部席が約 10m、平均約 6m である。プロジェクタ光源は 3KW 以上 10KW までのランプが推奨されているが、プロジェクタ側と眼鏡の偏光版で光量が約 1/4 になるので、通常の映画館と同様に室内の照明を落して視聴した。システムは、35mm フィルムのアナログプロジェクタであるので、解像度は、35mm フィルムの解像力 60 本/mm[78]にフィルム幅 35mm を掛け、横 2100pel とした。

表 4- 9　大画面立体アニメ映画館（3D Theater）の仕様

諸元	Akiba 3D Theater
スクリーン構成	35mm Film上下2分割
立体方式	偏光眼鏡ステレオ
画面サイズ(縦x横x奥行m)	4.4x8.14
プロジェクタ1台の解像度(縦x横pel)	1440x2100
総ポリゴン数	（手書きアニメ）
フレーム周波数f(Hz)	24
量子化ビット数(bit)	（フィルム）
看視距離(m)	（平均6.24m）
信号SN(dB)	（フィルム）

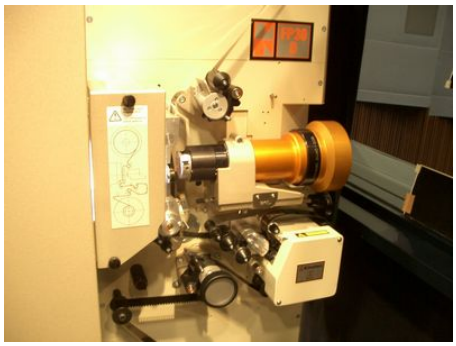


図 4- 35　立体映画用レンズ

評価者は、女性 2 名を含む 33 名であり、年代は、40 代から 50 代を中心に、20 代から 60 代の間でほぼ正規分布をしている。視力は、0.1 以下が 5 名いるが、0.5 以上が約 67%を占めていた。

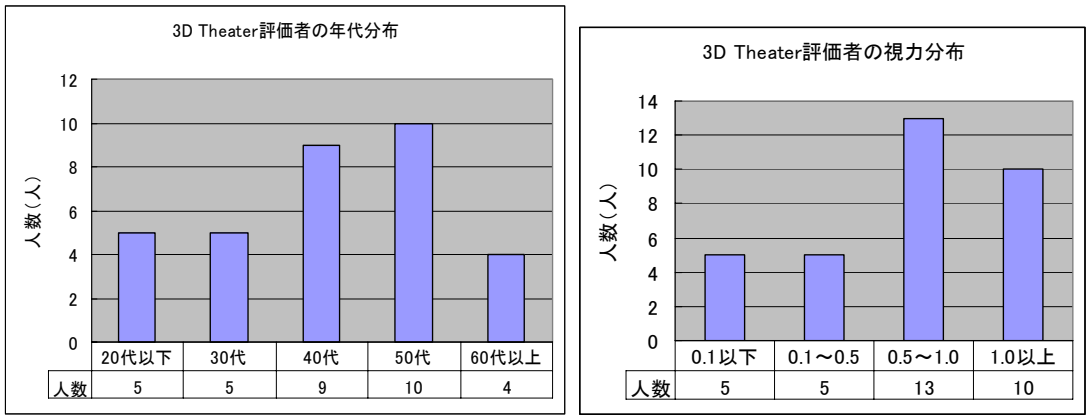


図 4- 36　3D Theater 評価者の分布

評価映像は、手書きのキャラクタや背景映像を、セル画毎に奥行位置を変えて配置した、書き割り的な立体映像と、3D CG を使った道や岩、林のボリューム感のある風景に、2D キャラクタを貼り込んで合成された立体映画（約 15 分）である。動きの激しい場面で、立体視が困難になる部分が一部あった。又、手書きアニメの味わいを残す為か、動きや形状の陰影などに、人工的な感じのするシーンが見られた。

図 4- 37　3D Theater 映像の例

(b)　評価結果

評価結果を図 4-38 と表 4-10 に示す。映像品質に関しては、2D の手書きアニメの立体化の為か、全体的に低目の評価値となったが、その中では、立体感の評価が最も高く、きれいさ、きめ細かさ、没入感はやや高いレベルである。アニメ映像の為、本物感の評価を行わない看視者が 4 名いたが、全体の平均は中間の値となった。動きの滑らかさは、24fps の映画の為と思われるが、やや高い程度である。非圧縮のフィルムの為、ノイズ感はかなり低い。心理要因に関しては、疲労感は普通、眼鏡意識（眼鏡が気になる）は、

やや高いレベルであり、違和感は、やや低い結果であった。これらと性別、年代、視力との間に目立った相関は見られなかった。眼鏡意識の問題は、眼鏡を軽く、個人の顔にフィットする形に改良する事や、自分の眼鏡に貼り付けるフィルムにする等で改善されると思われる。

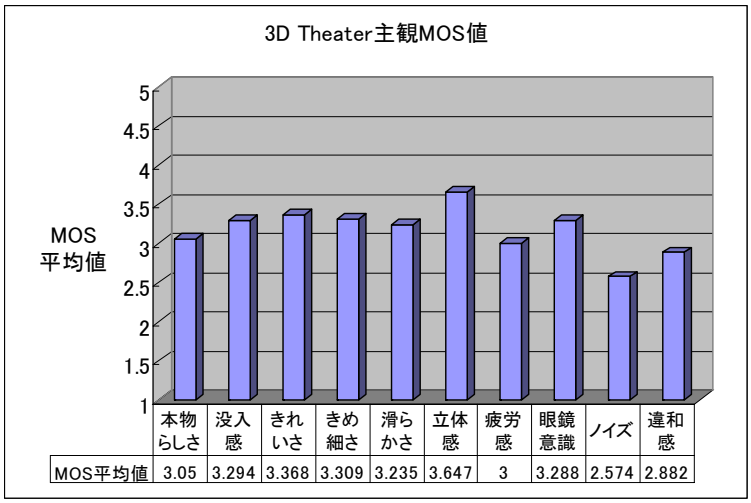


図 4- 38　3D Theater 映像の主観評価結果

表 4- 10　3D Theater 映像の主観評価データ

年代	視力	本物感	没入感	きれいさきめ細さ	滑らかさ	立体感	疲労感	眼鏡意識	ノイズ感	違和感	コメント
		3	3	3	3	3	4	4	5	3	4	
40	1～		3	2	2	2	4	3	2	2	4	
60～	1～	4	3	2	3	3	4	3	4	3	3	
50	0.5～1.0	2	2	5	5	5	4	4	5	1.5	3.5	
40	0.5～1.0	5	4	5	4	4	4	3	3.5	3	2	後半は立体でなくて良い，眼鏡に脂が付いた
40	1～	3	2	3	3	3	3	3	3	3	3	
30	0.1～0.5	4	4	4	3	1	3	5	3	1	3	大人向けの映像が欲しい
～20	1～	5	4	4	4	4	3	3	3	2	3	
60～	1～	3	2	4	3	2	3	4	3	3	3	
30	～0.1	2	3	1	1	1	2	5	5	4	3	
40	1～	4	5	5	5	5	5	1	2	4	3	
30	0.1～0.5	1	4	4	2	3	3	4	4	5	3	大人向けの作品が欲しい
30	～0.1	2	3	3	3	2	4	4	4	4	3	首を傾けられない，激しい動きの3D化は無理，疲れ大
30	1～	4	4	4	3	4	4	4	4	5	5	
50	0.5～1.0	3	3	3	4	4	3		4	3	3	
40	0.5～1.0	2.5	3	3.5	3.5	3.5	4	3	3	2	2.5	書き割り的
50	0.5～1.0	4	4	4	4	4	5	2	4	2	2	
40	1～	3	3	3	3	2	4	1	3	1	3	書き割りを立体と言って良いか？　900円は高い
～20	1～	2	2	3	4	4	3	2	5	3	2	
40	0.5～1.0	2	2	2	3	3	3	3	4	3	3	
40	0.5～1.0	3	2	4	3	3	4	5	4	3	3	
60～	1～		3	2	2	2.5	3	1	1	1	1	ラストシーンが追加されていた
30	1～		4	4	3	3	4	1	2	2	2	
～20	1～	4	4	5	5	5	4	3	5	1	1	背景は立体だがキャラクターは平面的．意図的か？
50	1～	2	3	1	1	1	2	4	3	3	4	
30	0.5～1.0	4	4	5	3	5	5	3		2	2	
30	1～	4	4	2	3	4	4	4	2	3	3	
40	1～	4	4	4	5	5	3	3	2	2	2	
30	1～	3	3	2	4	4	3	3	4	3	2	少し暗く感じる
40		1	2	3	3	3	3	2	3	2	4	
50	0.5～1.0	3	3	4	3	2	4	4	3	2	3	キャラクターも3Dにすれば尚面白い
30	1～	3	5	4	5	4	4	2	2	2	3	
40	0.5～1.0											
50	1～	2	3	4	3	2	4	2	3	3	3	速い動きでストロボ的に見える，3DCGとアニメの視差不整合あり
50	1～		5	3	4	4	5	1	1	1	4	キャラが平面的で違和感あり
39.12	0.82424	3.05	3.2941	3.3676	3.30882	3.23529	3.64706	3	3.28788	2.57353	2.8824	平均
平均		1.0533	0.9055	1.1235	1.02997	1.18855	0.77391	1.199	1.12521	1.05982	0.8533	標準偏差

(c)　臨場感度評価

前節と同様に、映像品質 6 項目の主観値の平均値を求めると、3.0 となる。これを 3DTheater 映像の臨場感度の主観値とみなす。他方、臨場感度の定義式より、臨場感度の客観値を求めると、表 4-11 となる。

表 4- 11　3D Theater 臨場感度の客観値

項目	値	項目	値
水平視野角(rad)	1.156	両眼視差	1
水平画素密度(rad^{-1})	1610	輻輳角	0.202
フレーム密度(f/s)	24	焦点調節	0
没入感度項 I	0.223	運動視差	0
量子化雑音(dB)	∞	画像要因	1
信号処理雑音(dB)	∞	立体感度項 T	0.796
画質項 Q	1	臨場感度 R	17.78

ここで、視野角と等価画素密度は、平均の看視距離を 6.15m として求めた。又、映像の量子化ビット数と信号 SN は、アナログフィルムの為、値が得られないが、ノイズが検知出来ないレベルであったので、画質項は 1 とした。立体感度項は、平均看視距離で映像が看視者の手前 1m 程度まで飛び出したので、最近距離を 1m、最遠距離を無限大として求めた。ステレオ立体映像であるので、焦点調節と運動視差は 0 である。

4.7 大画面立体自然映像

次に評価したのは、2 台の Hivision カメラで撮影された自然風景を、2 台のプロジェクタに偏光フィルタを通して背面投射する 3D Hivision システムである。看視者は、偏光眼鏡を掛けて映像を看視する。

(a) 3D Hivision カメラ

(b) 3D Hivision ディスプレイ

図 4- 39　大画面立体自然映像表示システム（3D Hivision）

(a)　評価条件

3D Hivison システムの仕様を表 4-12 に示す。このディスプレイは Kripton が提供し、スクリーンサイズは、横幅 1.55m の 16:9 ハイビジョンサイズである。映像投射には、S-XGA プラスの 1050x1400pel の解像度を持つプロジェクタを 2 台使うが、画面アスペクト比が合わないので、縦方向の解像度を捨てている。映像は、放送品質のハイビジョン信号であるので、フレームレートは 30Hz であり、信号 SN 比は約 40dB とした。監視距離は最前部で約 2m、最後部で約 4m、平均 3m とやや長めであった。プロジェクタの光源は 250W であり、スクリーンの輝度は 600cd/m² と比較的明るいが、部屋の照明が写り込まない様に、照明を落して看視した。

表 4- 12　大画面立体自然映像表示システム（3D Hivision）の仕様

諸元	3D Hivision
スクリーン構成	プロジェクタ2台で1面に重ね投射
立体方式	偏光眼鏡ステレオ
画面サイズ(縦x横m)	1.55x0.87
プロジェクタ1台の解像度(縦x横pel)	1050x1400
総ポリゴン数	（実写映像）
フレーム周波数f(Hz)	30
量子化ビット数(bit)	8
信号SN(dB)	40
看視距離(m)	3

評価者は、女性 2 名を含む 24 名であり、40 代を中心に 20 代から 60 代の間でほぼ正規分布をしている。視力は、0.1 以下が 3 名いたが、0.5 以上が約 90%を占めている。

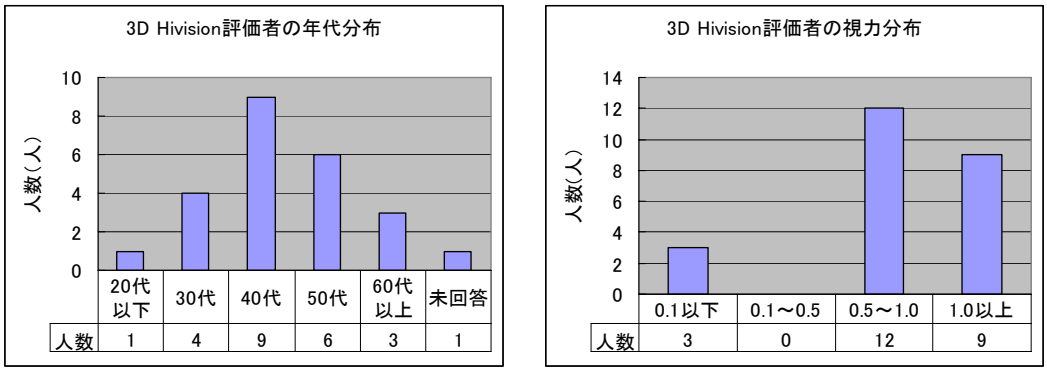


図 4- 40　3D Hivision 映像評価者の分布

評価映像は、以下に示す様な国内外でロケを行った実写映像である。

図 4- 41　3D Hivision 映像の例

(b)　評価結果

評価結果を図 4-42 及び表 4-13 に示す。10 年以上に渡る立体映像撮影ノウハウの蓄積と、コンテンツが自然映像である為、映像品質評価値は、いずれも高い値が出ており、疲労感は低い。没入感がやや低いのは、画面幅に比べて看視距離が約 2 倍と大きかった為と思われる。放送品質のハイビジョン映像の為、ノイズ感はかなり低い。違和感は普通であるが、両眼ステレオ立体映像の為と思われる。画質 6 項目の評価値の平均は、3.75 であった。

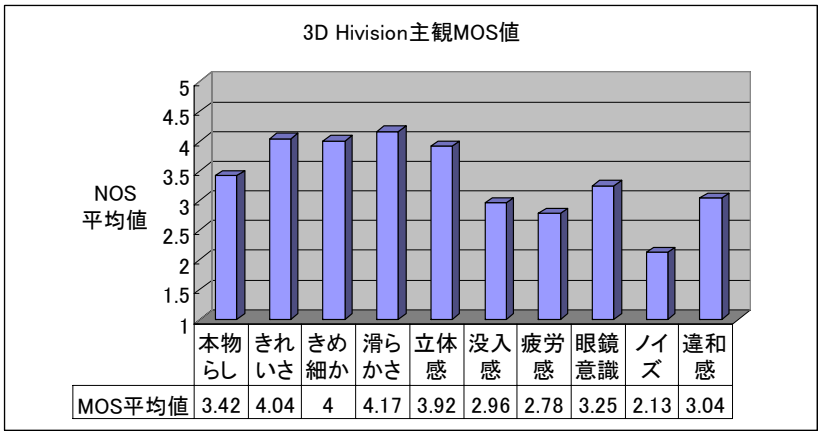


図 4- 42　3D Hivision の主観評価結果

表 4- 13　3D Hivision 映像の主観評価データ

年代	視力	本物らしさ	きれいさ	きめ細かさ	滑らかさ	立体感	没入感	疲労感	眼鏡意識	ノイズ	違和感	コメント
40	0.5～1.0	5	5	5	5	5	4	2	4	2	2	
50	0.5～1.0	2	5	5	5	5	3	2	3	1	4	Camui Mosiri
50	0.5～1.0	5	5	5	5	4	3	1	2	1	1	Cheer Leader
50	0.5～1.0	4	5	5	5	5	5	3	2	1	3	3D Hivision World
40	1～	1	2	3	3	3	1	2	4	1	4	Cheer Leader 面白いが立体でなくて良い
40	1～	2	3	3	3	4	1	3	4	4	5	Camui 水場、水中が見難い
30	～0.1	5	5	5	5	5	5	4	4	2	2	Camui
～20	0.5～1.0	4	5	4	4	5	4	3	4	2	4	
60～	1～	3	4	4	4	4	3	2	2	2	2	Anime
60～	1～	4	4	4	4	3	3	3	2	2	2	Sport
30	～0.1	2	5	5	5	5	1	4	5	1	2	
50	0.5～1.0	4	3	3	5	5	5	3	3	3	4	前後の動きが良く分かる
40	0.5～1.0	4	5	5	4	4	4	2	2	1	2	実写（心の森）素晴らしい、CGはNG
40	0.5～1.0	2	4	4	3	2	2	1	4	2	3	
40	～0.1	4	4	2	3	3	3	3	3	3	3	縦・斜線のギザギザが気になる
30	1～	3	3	2	4	3	2	3	2	3	4	
50	0.5～1.0	4	4	4	4	5	4	2	2	4	2	非常に良い
40	1～	2	2	4	4	2	1	5	5	4	4	近付くと荒さが目立つ
50	0.5～1.0	4	4	4	4	4	4	3	2	2	2	
60～	1～	4	5	5	5	5	4	3	2	1	5	
40	1～	3	4	4	4	3	2	4	4	2	4	
	1～	3	3	3	3	3	2		5	3	4	3D化のメリットがない作品がある
30	0.5～1.0	4	4	4	4	4	4	3	4	2	3	
40	0.5～1.0	4	4	4	5	3	1	3	4	2	2	
42.6	0.7625	3.416667	4.041667	4	4.166667	3.916667	2.958333	2.782609	3.25	2.125	3.041667	平均
		1.076904	0.934486	0.912871	0.745356	0.996522	1.337883	0.930475	1.089725	0.970932	1.098452	標準偏差

(c)　臨場感度評価

このシステムの臨場感度を求めた結果を表 4-14 に示す。視野角は、看視距離の平均値が画面幅の約 2 倍であるので、余り大きくならなかったが、逆に画素密度は高くなった。これらは、没入感項としてはキャンセルし合う。フレーム密度は、ハイビジョン信号を 2 画面デコーダに通して左右眼映像を取り出す方式なので、フレームレート 30Hz とした。又、画質項は、ハイビジョン信号の信号対雑音比を 40dB とした。立体感項は、映像の奥行を画面の前面 1m から無限遠までとして求めた。両眼ステレオ映像であるの

で、焦点調節項と運動視差項は 0 とした結果、臨場感度の客観値は、16.16 となり、主観値 3.75 とほぼよく整合した。

表 4- 14　3D Hivision 臨場感度の客観値

項目	値
水平視野角(rad)	0.51
水平画素密度(rad^{-1})	2711
フレーム密度(f/s)	30
没入感度項	0.206
量子化雑音(dB)	50
信号処理雑音(dB)	40
画質項	0.99

項目	値
両眼視差	1
輻輳角	0.125
焦点調節	0
運動視差	0
画像要因	1
立体感度	0.79
臨場感度	16.16

4.8 大画面立体映像の主観値と客観値の整合化

以上、4 種類の大画面立体映像の臨場感度の主観値と客観値を測定したが、実写映像の 3D Hivision 映像を除いて、他のシステムの客観値は、主観値よりかなり高い値であった。この原因の一つとして、これらの大画面映像が実写映像ではなく、CG や手書きによる非実写映像であった事が上げられる。これらの評価映像例に示した様に、非実写映像は、その映像のきめの細かさや、形、陰影など本物らしさの観点から、実写映像のレベルに達していない様に見え、映像表示システムの性能を使いきっていないと思われる。以下では、これら合成映像の品質について考察する。

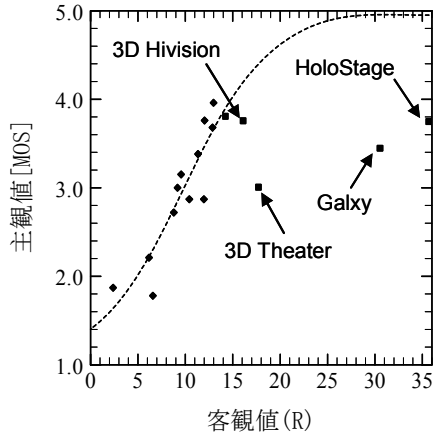


図 4- 43　大画面立体映像の総合品質主観値と客観値

(a)　ポリゴン密度

3D CG 映像では、立体形状の表面を、小さな三角形や四角形の平面をつなぎ合わせて近似する事が行われている。この小片をポリゴンと呼び、これに色を付ける事で立体形状をあらわしている。ポリゴンのサイズを小さくし全体を構成する数を多くすると、複雑な形状を表す事が出来る。逆に少ないと単純な形状しか表せず、本物の緻密さがなくなり、映像が粗く見える。従って、3D CG 映像の場合には、画素密度で映像の緻密さを表すのは適当でなく、使用されたポリゴン数で表す事を考える。

マルチ画面立体 CG 映像（HoloStage）の場合、表 4-1 に示した様に、総ポリゴン数の平均値が既知であるので、図 4-43 に示す様に、立体映像を構成する総ポリゴン数の平均値(N_p)で全画面面積(S)を割って、1 ポリゴン当たりの面積を出し、この平方根を等価画素サイズであるポリゴンサイズ(p_{eq})と見なして、等価画素密度(PD_i)をスクリーン毎に求め、その値にスクリーン面積重み(S_i)を掛けて平均化したポリゴン密度（PGD)を、画素密度に代わる CG 映像の等価画素密度とした。

$$PGD = \frac{1}{S}\sum_{i=1}^{3} S_i \times PD_i, \quad S = \sum_{i=1}^{3} S_i$$
$$PD_i = \frac{D_i}{p_{eq}}, \quad p_{eq} = \sqrt{\frac{S}{N_p}} \tag{4.36}$$

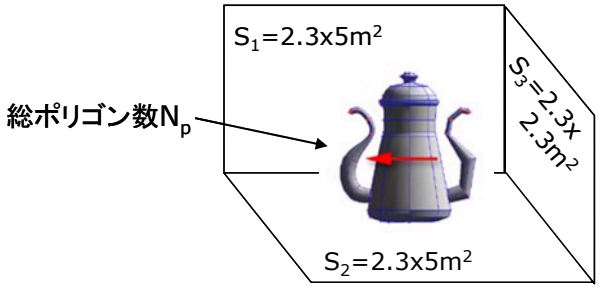


図 4- 44　ポリゴン密度の考え方

このポリゴン密度 PGD を用いて HoloStage 映像の臨場感度を求めると、表 4-15 に示す様に 10.14 となる。HoloStage 映像の品質に関する 6 項目の主観値の平均は、3.74 であるので、これらの値の組を、4.3 節で求めた臨場感度の回帰曲線グラフ（図 4-19）にプロットすると、図 4-45 の様になる。これより、CG 映像のポリゴン数が既知の場合は、画素密度よりポリゴン密度を用いた方が、主観値と客観値の整合は良くなる事が分かった。

表 4- 15　ポリゴン密度から求めた HoloStage 映像の臨場感度

項目	値
水平視野角(rad)	3.14
ポリゴン密度(rad^{-1})	145.2
フレーム密度(f/s)	48
没入感度項 I	0.109
量子化雑音(dB)	50
信号処理雑音(dB)	∞
画質項 Q	0.999

項目	値
両眼視差	1
輻輳角	1
焦点調節	0
運動視差	0.7
画像要因	1
立体感度項 T	0.93
臨場感度 R	10.14

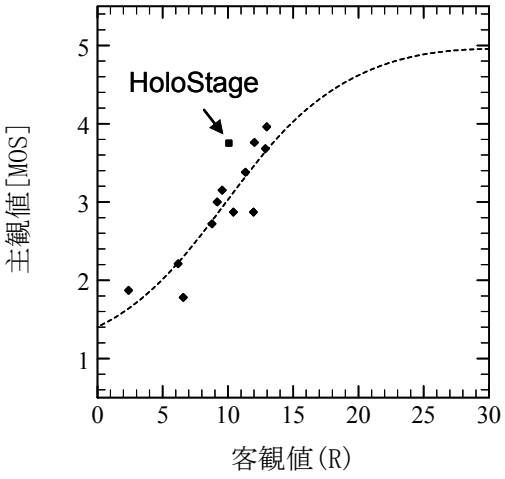


図 4- 45　ポリゴン密度から求めた臨場感度の客観値と主観値の関係

(b)　合成映像のほんものらしさ係数

ポリゴン密度を求めるには、CG 映像の描画に用いられたポリゴン数が既知で無ければならないが、シーンごとにポリゴン数が変わる事や、広く一般の立体映像の評価に適用した場合に、全ての評価映像のポリゴン数を求める事は、手書きアニメの様にその情報が得られないコンテンツも多く現実的でない。又、CG 映像の描画技術は年々進化しており、ポリゴン面を細分割して細かさを増す事が行われている。更にポリゴンによる描画以外にも、粒子モデルや確率モデルを用いて、柔らかいものや煙などの不定形なものを表す技術がある。又、メタボールと呼ばれる、複数の 3 次元空間中の点群を中心として濃度分布を設定し、濃度の閾値を形状の表面として融合や反発する有機体を現す技術等があり、ポリゴンを使わない描画方法が増えている。従ってポリゴン数のみで、CG 画像の本物らしさを表すのには限界がある。他にも、CG 画像を本物らしく見せる技術

には、光源の設定として、放射状に拡散する点光源、一様な平行光線、一部のみを平行光で照らすスポットライトや、全ての面を一様に照らす環境光、天球から一様に照らす天空光などがあり、又陰のつけ方も、ポリゴンの法線と光線方向のなす角度でポリゴン面の明るさを決めるフラットシェーディングや、オブジェクトの頂点の法線方向から、周りの面の明るさを一次補間して決めるグローシェーディングや、オブジェクトの頂点の法線方向の一次補間から各画素の法線方向を求めて、画素の色を決めるフォンシェーディングなど、滑らかな陰を作る技術が開発されている。又、ポリゴンへの着色も、テクスチャマッピングにより、複雑なパターンをポリゴンに貼り付けて、解像度を高める事や、光の反射量を調節出来る反射マッピングや、小さな凹凸を擬似的に表現するバンプマッピングやディスプレイスマッピング、面の透明感を設定出来る透明度マッピングなどがあり、材質のリアル感を増している。この様にして作られる 3D CG 画像の表示方法も、視点から遠い画素から順にフレームメモリに描画して、被写体の前後関係を作り出す Z バッファ法や、フレームのライン毎に画素間の奥行量を計算して描画するスキャンライン法や、3 次元空間上で視点から光源までの光を追跡してオブジェクト表面の反射や屈折の表現を行うレイトレーシング法や、各ポリゴンに光のエネルギー量を割り当てて、オブジェクトの相互反射を計算して柔らかい光を作るラジオシティ法等の技術がある。又、動きのモデル化も、質量を持った頂点をバネで繋いでメッシュを作る事で、柔らかい布の動きを再現したり、人体の関節に光や磁気マーカを付けて各関節の 3 次元的な動きを取り込んで骨格モデルの動きを作り、その表面にかぶせた伸縮するメッシュモデルで皮膚を現す事等が行われる。

これらの技術が総合され、CG 映像の本物らしさが決まるが、その一つ一つの要素技術まで遡って臨場感度のパラメータとして組み込む事は、実用上の負担が大き過ぎると思われる。そこで、ここでは、手書きアニメも含めて合成コンテンツの実写映像に対する本物らしさ係数 k_R を導入し、7 章で定義した臨場感度の客観値 R に、この本物らしさ係数 k_R を掛けたものを、合成映像の臨場感度 R_A とする。この本物らしさ係数は、合成映像の品質を上げる為の各種の技術が盛り込まれた合成映像を主観的に評価して得られた主観値 MOS_R から、回帰関数を解いて主観臨場感度 R_S を求め、この値と、システムのパラメータから計算される臨場感度の客観値 R との比として求める事が出来る。(4-18)式より、合成映像の臨場感度 R_A は次式となり、これより、合成映像の本物らしさ係数 k_R は、次式で得られる。

$$R_A = \frac{1}{\beta}\{\ln(\frac{4}{MOS_R - 1} - 1) - \alpha\}, \quad \alpha = 2.2, \quad \beta = 0.22$$
$$k_R = \frac{R_A}{R} \tag{4.36}$$

この式に、マルチ画面立体 CG 映像と、分光方式大画面立体 CG 動画と、大画面立体アニメ映画の臨場感度の主観値 MOS_R と客観値 R を入れ、本物らしさ係数 k_R の平均を

取った所、k_R=0.445 となった。表 4-16 に示す様に、それぞれの本物らしさ係数は互いに近い値である。これらの値を、臨場感度の回帰曲線グラフにプロットすると、図 4-45 の様になり、全ての立体映像システムの主観値と客観値がほぼ整合する。

合成映像の本物らしさ係数の考え方は、個々の合成映像の品質を個別に記述する事は出来ないが、多量の映像データから抽出された平均値としての係数を用いる事により、第 1 次近似としては、主観値と客観値の整合に有効であると思われる。

表 4- 16　合成映像の品質係数

	HoloStage	Galaxy	3D Theater
客観値R	35.73	30.59	17.78
主観値MOS_R	3.74	3.44	3
主観臨場感度R_A	13.53	12.03	10
品質係数k_R	0.379	0.393	0.562
k_R平均値	0.445		
合成映像の臨場感度R_A	15.9	13.61	7.92

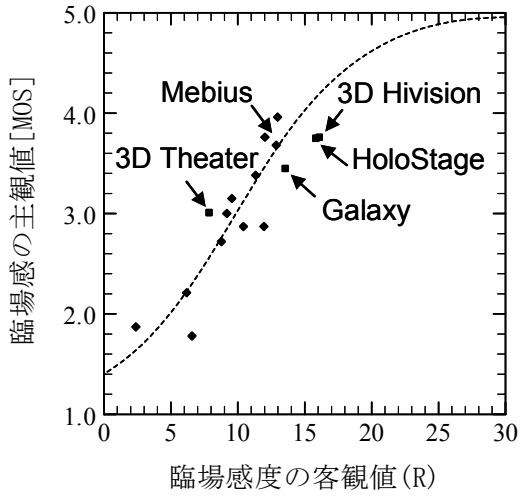


図 4- 46　大画面立体映像の臨場感度

図より、小画面から大画面まで、又実写映像から合成映像までを含めて、臨場感度の客観値と主観値はほぼ整合していると思われる。

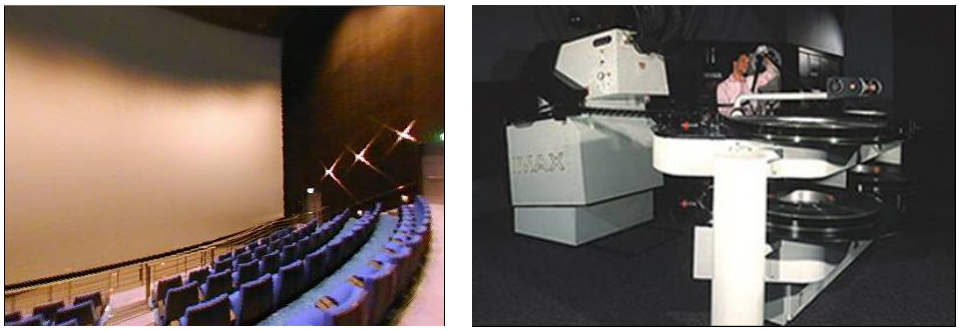
合成映像の品質は年々、ポリゴン数の増加のみでなく、モデルの現し方や補間技術の進化、テキスチャマッピングの高品質化や、照明や陰影のつけ方などの進化が続いているので、この合成映像の本物らしさ係数の値は、映像技術の進化と共に増加し、1.0 に近

付いて行くと思われる。

(c)　実写合成混合映像の評価

以下では参考までに、実写映像と 3D CG による合成映像とが混合された立体映画映像の主観評価結果と客観評価結果を示す。

評価したシステムは、IMAX シアターとして一般公開されている映画館である。



(a) シアター内部　　(b) フィルム映写機

図 4- 47　大画面実写合成混合立体映像システム（IMAX シアター）

表 4- 17　大画面実写合成混合映像システム（IMAX シアター）の仕様

プロジェクタ	IMAX
看視距離D(m)	11
画面横幅H(m)	22
画面縦幅V(m)	16
横画素数hp(pel)	6720
縦画素数vp(pel)	4200
信号横画素数shp(pel)	1920
信号縦画素数svp(pel)	1080
フレーム周波数f(Hz)	24
量子化ビット数n	10
信号SN(dB)	56

映写機は、フィルムサイズ 112x70mm の IMAX であるが、実写部分の撮影は全編、Genesis デジタルシネマカメラ（1920x1080）で行われたので、コンテンツの解像度は映写機の解像度より低い。3D 化は、LR2 本のフィルムを多重投影し、偏光眼鏡で見るステレオ立体映像である。看視距離は、観客席ほぼ中央の位置からの値である。

評価した映像は、実写シーンや CG 合成シーン及び、両者の混合シーンが混在する SF 映画約 2.5 時間であり、その中で約 20 分間立体シーンがある。合成シーンは、実写シーンとの違和感がない様に非常に緻密に描かれており、実写映像と見まがうレベルに仕上がっていた。又、実写シーンが多数を占めたので、本物らしさ係数は 1.0 とした。

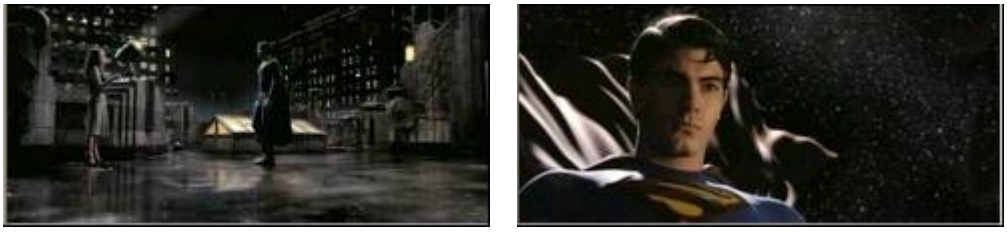


図 4- 48　IMAX シアターの映像例

表 4- 18　IMAX シアターの臨場感度の客観値

水平視野角(rad)	1.571
垂直視野角(rad)	1.258
水平画素密度(pel/rad)	960
没入感度	0.181
画質	0.9997
両眼視差	1
輻輳角	0.050
焦点調節	0
運動視差	0
画像要因P	1
立体感度	0.792
臨場感度	14.334

このシステムの物理パラメータから求めた総合品質の客観値を表 4-18 に示すが、14.33 となった。視野角は 90 度とかなり広いが、画素密度が低いので、両者の積である没入感度はあまり高くない。画質は良く、ノイズは検知されなかったが、デジタルシネマカメラの解像度がスクリーンサイズに比べて低い為か、シーンによっては画質がややソフトに感じられた。映像の飛び出し量は控えめで、時々スクリーンまでの距離の半分程度まで映像が近付くが、大半はスクリーンより奥側に奥行を持った立体映像で見やすい。

主観評価結果を、図 4-49 と表 4-19 に示す。画質評価 6 項目の平均値は、3.80 であった。評価者は 5 名であったが、立体映像関係の業務に携わる人々で立体映像への理解は深く、これまでの主観評価実験の参加者でもあり、年齢は 50 代が 2 名、60 代が 3 名、視力は 0.1～0.5 が 1 名、0.5～1.0 が 2 名、1.0 以上が 2 名であった。

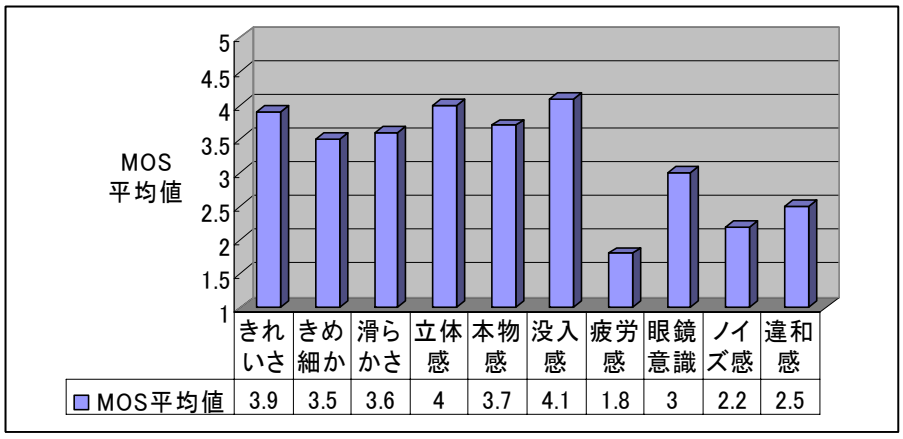


図 4- 49 IMAX シアターの主観評価結果

表 4- 19　IMAX シアターの主観評価値

年代	視力	眼鏡	きれいさ	きめ細かさ	滑らかさ	立体感	本物感	没入感	疲労感	眼鏡意識	ノイズ感	違和感	コメント
50	1	1	3.5	3.5	4	4	4.5	4.5	2	2	3	3.5	広い景色で矮小感
60	1	0	4	4	4	5	5	5	1	2	1	2	
60	0.75	0	4	4	4	4	3	3	1	3	2	2	
60	0.25	1	4	3	3	4	3	5	2	4	3	2	音がうるさい
50	0.75	0	4	3	3	3	3	3	3	4	2	3	
合計			3.9	3.5	3.6	4	3.7	4.1	1.8	3	2.2	2.5	
標準偏差			0.22	0.50	0.55	0.71	0.97	1.02	0.84	1.00	0.84	0.71	

このシステムの客観値と主観値を、図 4-45 の相関グラフに追加プロットしたものを図 4-49 に示すが、両者は良く整合している。この様に、合成映像と実写映像の両方を含む映像では、両者の間に違和感が出ない様、コストを度外視した合成映像の作りこみによって、実写映像の品質レベルに合成映像を合わせ込む事が行われており、この様な混合映像の場合には、本物らしさ係数は 1.0 として、実写映像と同等に扱って良いと思われるが、今後更に多数の混合映像での評価を行って確認する必要がある。又、今後コンピュータ技術が更に進化して行くにつれて、この様な高品質な混合映像が増えて来るものと思われるが、それにつれて合成映像の本物らしさ係数も徐々に改善して行くと思われるので、今後の継続的な評価が重要になる。

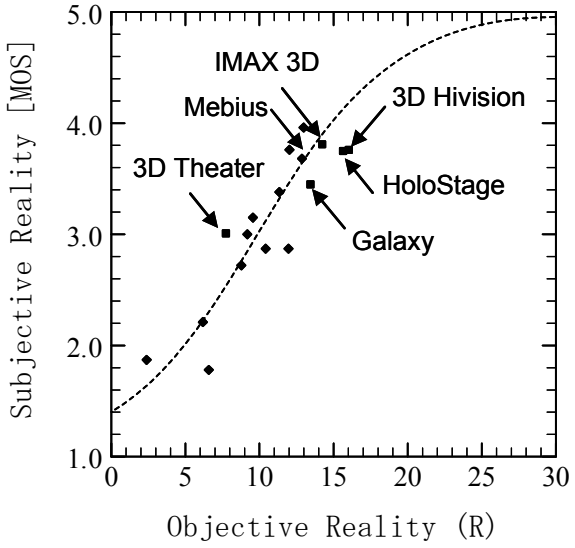


図 4- 50　実写合成混合映像の臨場感度

4.9 立体映像評価結果の考察

前節までに各種方式による立体映像の品質計測を行い、主観的・客観的に高得点の得られる理想に近い立体映像の姿を探って来たが、その計測結果から観測された事項を以下に纏める。

(a) 画素密度

立体映像の主観・客観評価値を上げる為には、視野角の増大が有効である事が基礎実験からも示されたが、プロジェクタ等による単なる大画面化では、画素密度の低下をもたらし、結果として高品質にはならない。この事は、客観値評価式において、視野角項と画素密度項の積を取る事により、画面サイズがキャンセルされる事からも導かれるし、大画面立体映像の主観評価結果からも、大画面映像が必ずしも高得点にならない事からも確認される。この原因として、大画面ほど看視距離が大きくとられる傾向があり、これにより視野角が広まらず評価値が上がらない事も想定されるが、視野角と画素密度の両方を考慮すると看視距離はキャンセルされ、総画素数のみが有効パラメータとして残る。従って、映像品質評価値を上げる為には、総画素数の増大が重要である。

(b) 立体感要因

立体映像の奥行量と総合品質との関係の計測からは、監視距離に対し、僅かな奥行量（7%程度）で主観品質の飽和が見られたが、この原因として、立体感要因の内、感度の最も高い両眼視差量が大きく主観品質に影響を与えていることが推測される。両眼視差の融合による立体視は、網膜上で最も分解能の高い中心窩で生じる事が解明されており、両眼視差量がこの中心窩のサイズを越える場合は、両眼融合が起こらず像が 2 重に見える。その場合には両眼の輻輳角を変化させ、立体映像の奥行量を認識するが、この輻輳角変化による立体視の感度は低く、総合品質にあまり大きな影響を及ぼさなかった。又、評価した立体映像が主に 2 眼式立体映像であった為、単に輻輳角を大きくするだけの立体映像ではその他の立体視要因が伴わない為、総合品質が上がらなかったものと思われる。この傾向は、各種の大画面立体映像で大きな奥行量を持った映像の評価結果にあまり差が出なかった事からも確認出来る。両眼視差以外の立体視要因は総合品質に大きな影響を与えないが、それらの要因がないと立体感が上がらない事は、大画面立体映像のうち運動視差を持つ HoloStage の立体感に関する主観値が最も高かった事からも示されている。従って、総合品質を高める為には、立体視要因を全て満たす立体映像の実現が重要である。

本提案の評価式と主観評価結果との整合性を確認した立体映像システムは、裸眼式立体ディスプレイ及び、シャッターグラス・分光眼鏡・偏光眼鏡の各種２眼式立体ディスプレイ及びプロジェクタであり、看視者の位置情報をフィードバックして運動視差を実現する HoloStage を除いて、焦点調節や運動視差を持たない。従って、本提案の評価式が特に有効な立体映像は、これらの各種 2 眼式立体ディスプレイ及びプロジェクタに、実写映像・合成映像を表示した場合であるが、評価式自体は全ての立体視要因を含むので、運動視差を持つ HoloStage の評価値が他のシステムの評価値と良く整合した事が示す様に、その他の方式の立体映像の評価にも適用できると思われる。今後、更に評価システムを増してその検証を行う事が今後の課題である。

(c)　合成映像

立体映像の物理パラメータが十分高くても、表示する映像のコンテンツの影響により、総合品質が高まらない事は、実写映像とコンピュータグラフィックスなどによる合成映像の主観評価結果から確認された。この影響を除くためには、出来るだけ実写映像での測定が主観値と客観値の整合上望ましいが、合成立体映像もアニメや SF 映画として普及しつつあり、その品質は年々向上し、実写映像に迫るリアリティを持つ映像も今後多数出てくると思われる。従って、これらを含めて評価出来る総合品質評価方法が重要になると思われる。本研究の品質評価技術で提案した本物らしさ係数は、合成映像の客観品質と主観品質のギャップを埋めて両者の整合を取る効果があり、数値で表しにくいコンテンツに依存する要因を含む映像の定量的評価に有効と思われる。

(d)　望ましい立体映像

以上、立体映像の総合品質評価技術の検討から導かれる課題の内、映像の多画素化は、日々進化を続けているコンピュータやディスプレイを支える半導体技術の進化の延長線上で解決されると思われる。又、合成映像のリアリティも、これらのハードウエアの進化に伴うソフトウエアの進化に伴い不断に向上して行くと思われる。立体視要因を全て満たす理想立体映像の実現技術は、2 眼方式ではなく、3 次元立体映像方式によるアプローチが好ましいと思われるが、第 2 章で概観した様に、現在の 3 次元立体映像方式は、膨大なデータ量を必要とし、高精細度化や、撮影から表示までのリアルタイム化に課題がある。従って、理想的な立体映像実現の為には、実時間処理が可能で多画素化が容易な 3 次元立体映像方式の開発が急務であると思われる。

■ 第 5 章 ■

3 次元立体映像の生成技術

第5章　　3 次元立体映像の生成技術

本章では、前章の検討結果から抽出された課題である、理想的な立体映像の生成方式として、全ての立体視要因を効率良く満たす空間標本化法を提案し、その原理と本方式を実現する為に必要なデバイス及び信号処理技術について検討する。

5.1 空間標本化方式の原理

一般に写真は、三次元の立体空間をレンズにより 2 次元平面に写像したものであるが、全ての奥行位置にある対象物を 1 平面にボケなく写像する為に、レンズの絞りを絞って焦点深度を深くして奥行情報を捨てている。逆に焦点深度の浅い明るいレンズを使えば、図 5-1 に示す様に、無限遠から最近距離 D までの立体空間は、レンズの後方 f から d までの小映像空間に写像され、夫々の被写体映像は、距離に応じて異なる位置に結像する。ここで、映像の位置 d は次式を満たす[79][80]。

$$[\infty, D] \Rightarrow [f, d]$$
$$d = \frac{Df}{D - f} \tag{5.1}$$

ここで、f はレンズの焦点距離であり、例えば f=50mm の場合、レンズの前方 D=50cm から無限遠までの被写体は、レンズの後方 f=50mm から d=55.5...mm の間の小空間に写像される。

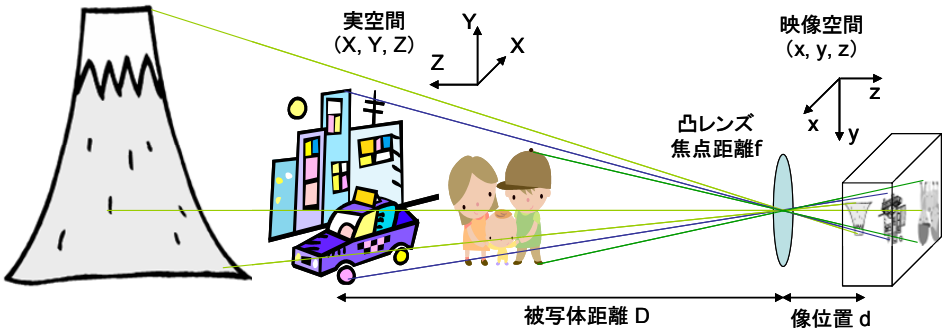


図 5- 1　実空間から映像空間への写像

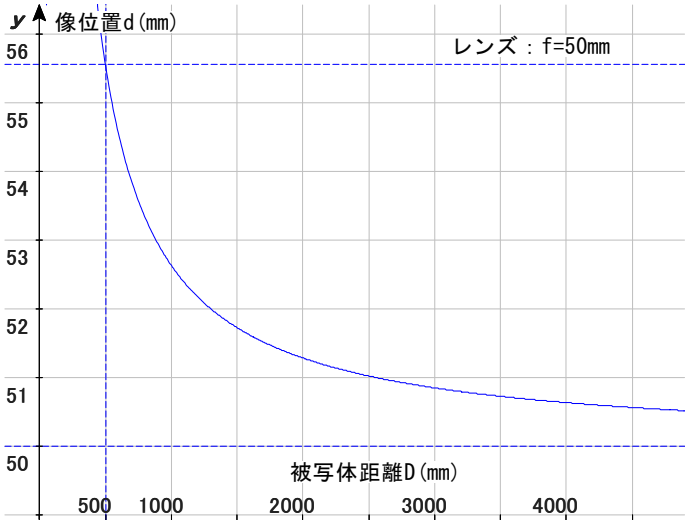


図 5- 2　f=50mm のレンズによる実空間距離と映像空間距離の対応

この時、映像の倍率は、f/(D-f)で与えられ、距離 D にほぼ反比例し、視覚特性に合った自然なサイズ比の映像となる。

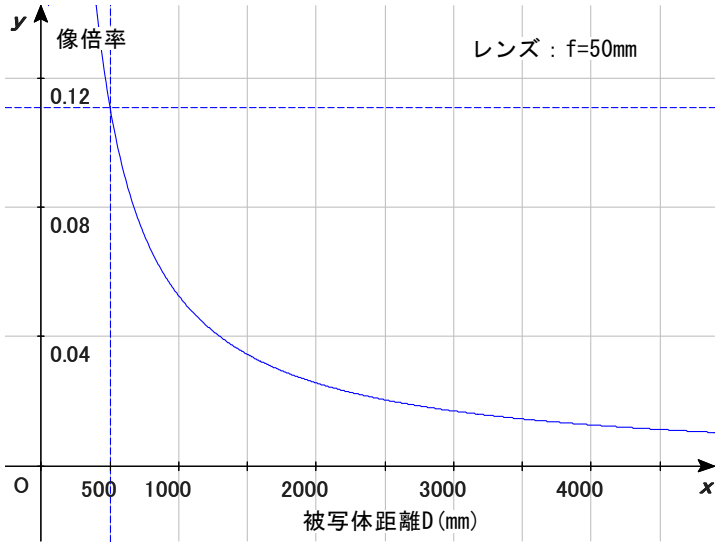


図 5- 3　f=50mm のレンズによる実空間距離と像倍率の関係

この映像は奥行を持った映像であるので、これを 3 次元標本化する事により立体映像を得る事が出来る。

空間標本化法のメリットは、巨大な実空間を視覚特性に合った小空間に射影し、この小空間を標本化する事で、コストパフォーマンス良くコンパクトな立体映像空間を構築出来ることである。

5.2 空間映像の標本化

空間映像の標本化は、従来の縦横の 2 次元平面の標本化ではなく、奥行方向も含めた 3 次元標本化を行う必要がある。その為には、幾つかの方法がある。例えば、レンズと撮像板間の距離を少しずつ変更しながら従来の 2 次元標本化を繰り返す方法が、Depth from Focus 法[81][82]や、Shape from Focus 法[83][84]として知られている。この方法は、1 つの立体映像データを取得する為に、数十回の 2 次元標本化を繰り返す必要があり、これで動画を実現する為には、高速な撮像素子を必要とすると共に、レンズや撮像素子を高速に機械的に動かす必要があり、耐久性や信頼性の点から課題がある。

又、レンズに電界を掛けてその屈折率やレンズ形状を変えながら、異なる焦点距離で 2 次元標本化を繰り返す方法もある[17]。この方法は、レンズや撮像素子の機械的な運動は不要であるが、撮像素子は高速である必要があり、動画に適用するには同様の課題がある。

本論文では、以下に述べる様に、透明な撮像板をレンズの後方に多層に配置する事により、可動部分を持たず高速な撮像素子も必要としない、動画に適する標本化方法を提案する。

(a) 多層撮像板による三次元標本化

標本化は、位置 f から d の間に複数の透明な撮像板を配置する事により実現出来る。夫々の撮像板上には、その位置に応じた距離にある被写体の映像のみが焦点を結び、その他の距離にある被写体はボケるので、ボケた映像を除去する事で、被写体の距離に応じた映像を分離取得する事が出来る。個々の撮像板から得られる映像を重ね合わせると、通常の 2 次元平面映像とほぼ同じになるので、立体映像のデータ量はほとんど増えず、従来と同じ手段で伝送・蓄積が可能である。

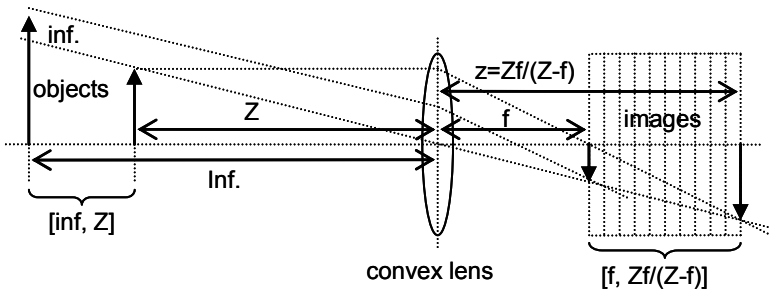


図 5- 4　多層撮像板による標本化

(b)　標本化に必要な撮像板枚数

空間標本化法による映像の奥行分解能は、レンズの直径 L と焦点距離 f で与えられる F 値=f/L で決まり、必要な撮像パネル枚数は、被写体までの最近距離と、各パネルの最大許容ボケ量で決まる。図 5-5 の様にレンズの前方 D の距離にある被写体 a は、レンズの後方 d の位置に像 b を結ぶが、撮像パネルが更に Δ ずれた位置 y にあると、像点はサイズ δ にぼける。

$$\delta = \frac{L\Delta}{d} = \frac{\left|(y-f)D - yf\right|}{FD} \tag{5.2}$$

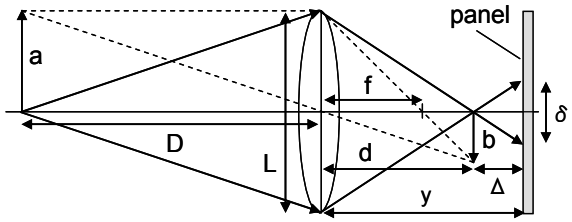


図 5- 5　撮像板位置と映像のボケ量

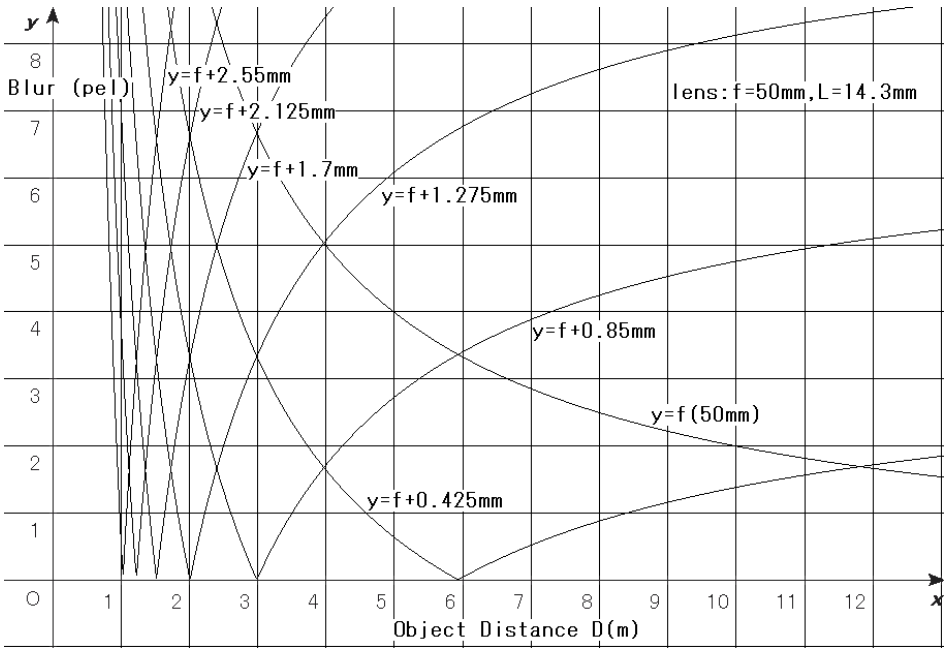


図 5- 6　撮像板位置 y における被写体距離 D とボケ量 B の関係

今、f=50mm、L=14.3mm のレンズを用いた場合のボケ量を画素サイズで正規化し、パネルの位置 y をパラメータにプロットすると、図 5-6 の様になる。ここで、撮像パネル

サイズは 36×24mm、画素数は 1000×667pel とした。このグラフより、撮像パネルを等間隔に配置すれば、各パネルの最大ボケ量を一定に抑えられる事と、許容ボケ量と最近撮影距離を決めれば、必要なパネル枚数が決まる事が分かる。例えば、最大ボケ量を約 1.5pel、最近撮影距離を 1m とすれば、パネル枚数は 7 枚となる。文献[74]によれば、視距離 2m での焦点調節の奥行感度は 0.33m ほどであり、輻輳や両眼視差ではさらに高感度（高分解能）となるため、パネル枚数 7 枚では、奥行分解能が不足するが、5.3 節に述べる様に、撮像パネルの間に出来る像は、その像の前後のパネルにボケ量が内分された映像として撮影され、このボケ量の比を輝度比に変えて、2 枚の表示パネルによる融像効果[68][69]で補間を行う事で、奥行感度の視覚条件は改善される。又、遠距離になる程低下する奥行分解能は、人の視覚特性に適合している。明るいレンズを使えば、奥行分解能を更に上げる事が可能である。

図 5-7 に、同じレンズを使って近距離撮影を行う場合の被写体距離 D とボケ量 B の関係を示す。被写体距離が 1m 以内になると、焦点深度が浅くなり、多数の撮像パネルが必要となるので、近距離撮影の場合は、レンズの口径を絞る事により焦点深度を深くするのが良いとおもわれる。逆に、望遠撮影を行う場合には、焦点距離の長いレンズを使う事により、遠距離での焦点深度を下げて、奥行分解能を上げる事が出来る。

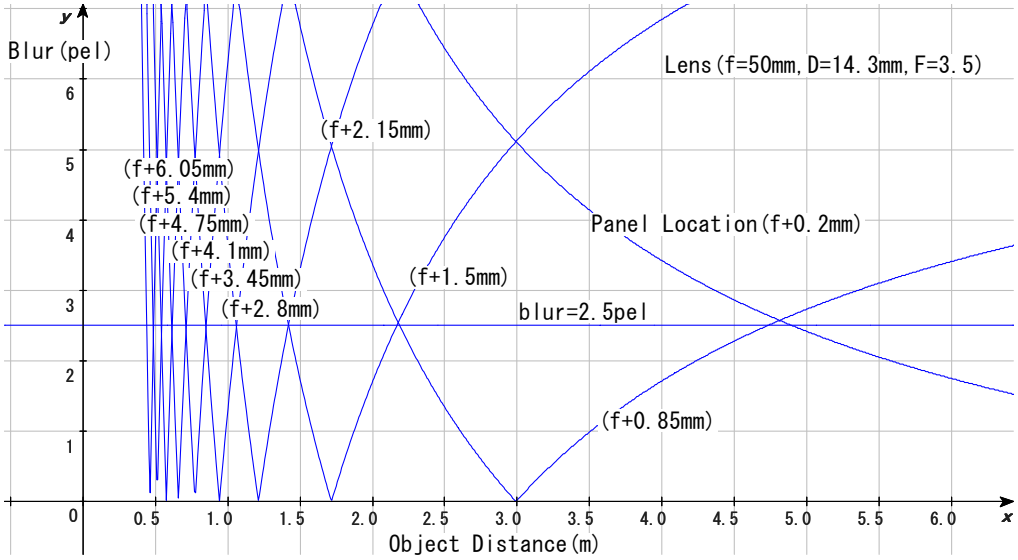


図 5-7　近距離での被写体位置 D とボケ量 B の関係

5.3 空間映像の再生

空間映像の再生は、撮像と同様に幾つかの方法が考えられる。先ず、バリフォーカルミラー[12]や可変焦点レンズ[17][18][19]により、奥行位置を変えながら 2 次元映像を順に表示して行く方法が考えられるが、この方法は、1 つの立体映像を再生する為に多数の平面映像を切り替えて表示する必要があり、高速な表示素子が要求されると同時に、ミラーやレンズの焦点距離を高速に切り変える必要があり、耐久性の観点から問題があると思われる。

同様に、固定焦点レンズと表示板の距離を変えながら複数の奥行位置に映像を投影する方法[81][82]も、機械的な動作を必要とする為、好ましくないと思われる。

ハーフミラーやプリズムを使って、光学的に映像の奥行位置の異なる映像を多重投影する方法[85]は、奥行位置の調整が機械的精度に影響される事と、奥行分解能を上げるには多数の光学素子を重ねる必要があるが、各映像の奥行の差は画面面積に比べてわずかであり、配置の空間的制約が大きいと言う課題がある。

本論文では、以下に述べる様に、透明な映像表示板を重ねて夫々に奥行の異なる映像を表示する事で、機械的な動きが不用で、かつ高速な表示素子も必要としない、映像空間再生方法について検討する。

(a) 映像空間の直接看視

標本化された空間映像は、レンズにより上下左右前後が反転しているので、レンズ軸周りに 180 度回転し、背面から直視すれば、奥行が元の距離 D にほぼ反比例した立体映像になるが、映像サイズも同じ比率で縮小しているので、4 章で述べた様に、被写体までの距離に反比例して映像サイズと奥行分解能が減少する、透視射影に基づく人の視覚特性によく合い、近距離で看視する縮小映像空間と、遠距離の実寸大の空間は同じジオメトリで知覚され、自然な立体映像が見られる。

今、撮影映像の像倍率 M_s は、図 5-4 より、

$$M_s = \frac{f}{D-f} \tag{5.3}$$

となり、この像を見る距離 y を撮影レンズのあった位置からの距離 X で表すと、

$$y = X - d = X - \frac{Df}{D-f} \tag{5.4}$$

であるので、この時の距離倍率 M_D は、

$$M_D = \frac{y}{D} = \frac{X}{D} - \frac{f}{D-f} \tag{5.5}$$

となる。この距離倍率 M_D が、像倍率 M_S と等しければ、映像空間のジオメトリ比は正しく認識されるので、M_D=M_S と置くと、

$$\frac{X}{D}-\frac{f}{D-f}=\frac{f}{D-f} \tag{5.6}$$

より、

$$X=\frac{2Df}{D-f} \tag{5.7}$$

となる。これをプロットすると、図 5-8 の映像サイズ 1 倍のカーブとなる。正しいサイズ比で見る為には、看視距離 X は、レンズの焦点距離 f の 2 倍の位置に取れば、すなわち、像位置からレンズの焦点距離だけ離れた位置から見れば、約 1m 以遠の被写体映像をほぼ正しい比率で見る事が出来る。

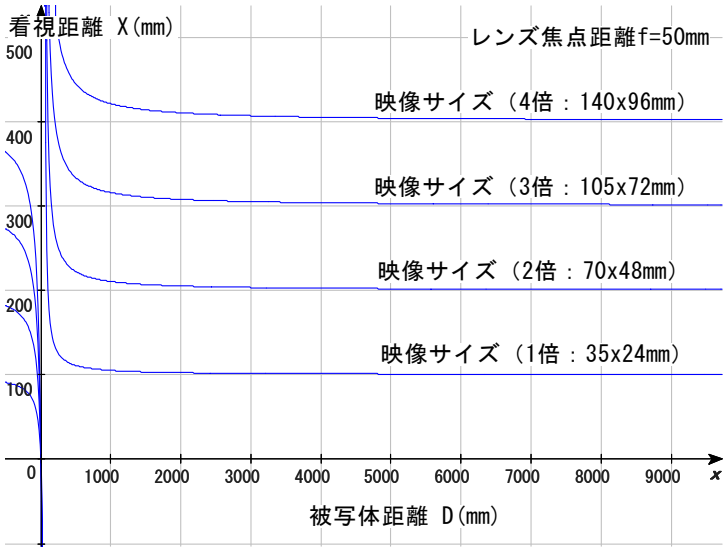


図 5- 8　距離倍率と映像倍率が等しくなる看視距離

しかし、撮影された像サイズのままでは、像サイズも看視距離も小さすぎるので、A 倍に拡大した映像を見る事にする。この時、映像空間の奥行も A 倍にしなければならないので、レンズから映像までの距離 d を A 倍する。

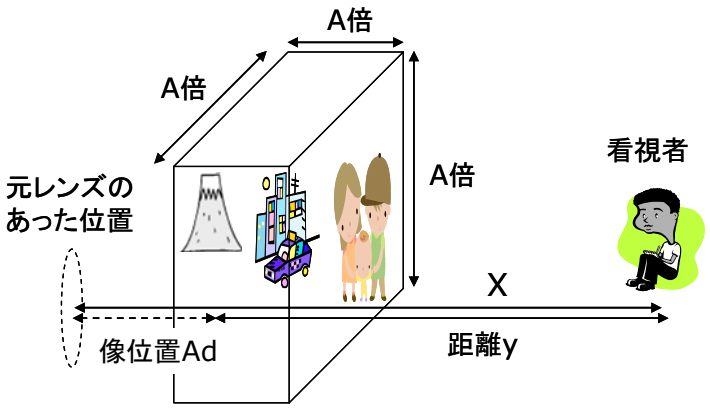


図 5- 9　空間映像の直接看視

その結果、像倍率 M_S と距離倍率 M_D は、下式で与えられ、これを等しいと置くと、看視距離 X を A 倍すれば、正しいサイズ比で看視出来ることが分かる。

$$
\begin{aligned}
&M_s = \frac{Af}{D-f}, \quad y = X - Ad = X - \frac{ADf}{D-f} \\
&M_D = \frac{y}{D} = \frac{X}{D} - \frac{Af}{D-f} \\
&M_D = M_S \quad \Rightarrow \quad \frac{X}{D} - \frac{Af}{D-f} = \frac{Af}{D-f} \\
&\therefore X = A \times \frac{2Df}{D-f}
\end{aligned}
\tag{5.8}
$$

例えば、映像を 10 倍すれば、映像サイズは 35cm×24cm、看視距離 y は 50cm となり、デスクトップでの看視に適する。30 倍にすれば、映像サイズは 105cm×72cm、看視距離 y は 1.5m となり、室内での看視サイズに適する。200 倍にすれば、映像サイズは 7×4.8m、看視距離 y は 10m となり、映画館サイズになる。

図 5-8 より、被写体距離が 1m 以下になると、看視距離 X をカメラの焦点距離 f の 2 倍以上に取らないと正しいサイズ比にならなくなる。例えば、0.5m の距離の被写体映像を正しいサイズ比で見る為には、焦点距離 f の約 2.2 倍の距離から見れば良い。この時、他の距離の被写体のサイズ倍率と距離倍率をプロットすると、図 5-10 の様になる。無限遠を除いて、サイズ比が最大約 10%小さくなり奥行感が強調されるが、許容範囲内と思われる。

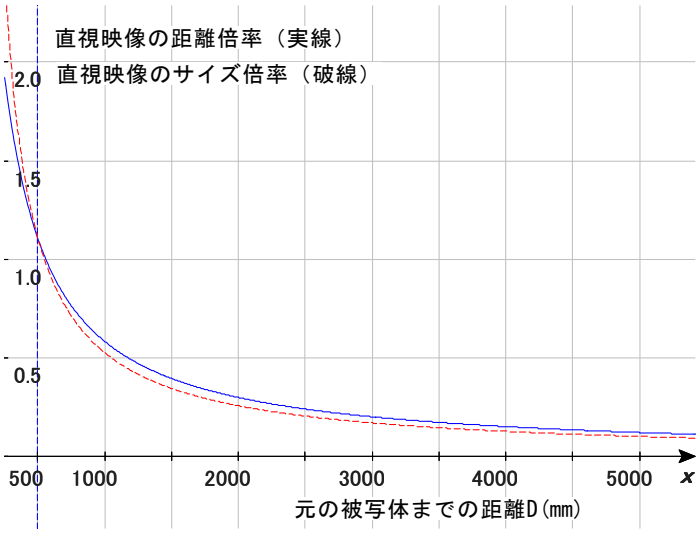


図 5- 10　直接看視映像の奥行距離倍率と像倍率の関係

以上の議論を射影方程式から確認すると、人の視覚は、焦点距離 f の眼球レンズにより、網膜に中心射影される映像より得られるので、実空間中の被写体（X,Y,Z）と網膜上の映像（x,y）との間には、斉次方程式を変形して次式が成立する[58]。

$$\begin{bmatrix} x \\ y \\ f \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{5.9}$$

但し、世界座標系（X,Y,Z）は眼球レンズ中心を原点とし、Z 軸は視線方向とし、網膜上の映像は十分小さく平面で近似出来ると仮定した。

一方、同じ被写体（X,Y,Z）を焦点距離 f_i のレンズで射影して得られる立体映像（X_i,Y_i,Z_i）は、次式を満たす。

$$\begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} = \frac{f_i}{Z - f_i} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{5.10}$$

但し、カメラ座標系（X_i,Y_i,Z_i）は、世界座標系（X,Y,Z）と一致させた。
この映像を原点(Z=0)から Z 方向に D 離れた位置から原点方向に振り返って見た場合、網膜に映る映像（x_i,y_i）は次式で表される。

$$\begin{bmatrix} x_i \\ y_i \\ f \end{bmatrix} = \frac{f}{D - Z_i} \begin{bmatrix} X_i \\ Y_i \\ D - Z_i \end{bmatrix} \tag{5.11}$$

これより、

$$\begin{bmatrix} x_i \\ y_i \\ f \end{bmatrix} = \frac{f}{D - f_i Z/(Z - f_i)} \begin{bmatrix} f_i X/(Z - f_i) \\ f_i Y/(Z - f_i) \\ D - f_i Z/(Z - f_i) \end{bmatrix} \tag{5.12}$$

今、看視距離 D をカメラの焦点距離の 2 倍（D=2f_i）と置き、被写体までの距離 Z はカメラの焦点距離 f_iより十分大きい（Z≫f_i）と見なすと、

$$\begin{bmatrix} x_i \\ y_i \\ f \end{bmatrix} \cong \frac{f}{2f_i - f_i} \begin{bmatrix} f_i X/Z \\ f_i Y/Z \\ 2f_i - f_i \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} x \\ y \\ f \end{bmatrix} \tag{5.13}$$

となり、被写体（X,Y,Z）を直接見たのと同じ映像（x,y）が網膜上に見られる。

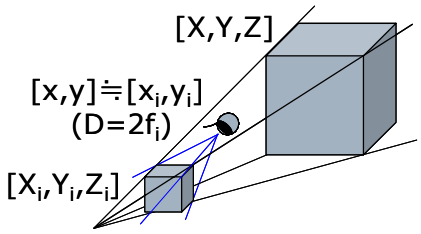


図 5- 11　実空間と射影立体映像の単眼視

次に、この映像を両眼で見た場合の奥行知覚について考察する。4 章で述べた様に、立体映像の奥行知覚は両眼視差変化によるものが最も高感度であるので、射影された立体映像を見た時の両眼視差変化量を求める。

距離 Z にある被写体の奥行がΔである場合の両眼視差変化量は、(4.8)式より、

$$B = \frac{Lr\Delta}{Z^2} \tag{5.14}$$

で与えられる。但し、L=65mm：両眼間距離、r=19mm：眼球のレンズ～網膜間距離である。この被写体を焦点距離 f_iのレンズで射影した立体映像の奥行量Δ_iは、距離 Z の点が $Zf_i/(Z-f_i)$に射影され、距離(Z+Δ)の点が$(Z+\Delta)f_i/(Z+\Delta-f_i)$に射影されるので、

$$\Delta_i = \frac{Zf_i}{Z - f_i} - \frac{(Z + \Delta)f_i}{Z + \Delta - f_i} = \frac{\Delta f_i^2}{(Z - f_i)(Z + \Delta - f_i)} \tag{5.15}$$

となる。この立体映像を原点 Z=0 から距離 $2f_i$離れた所から振り返って見た時の両眼視差変化量 B_iは、映像の近点までの距離が $2f_i-Zf_i/(Z-f_i)$であるので、

$$B_i = \frac{Lr\Delta_i}{(2f_i - \frac{Zf_i}{Z - f_i})^2} = \frac{Lr\Delta(Z - f_i)}{(Z - 2f_i)^2(Z + \Delta - f_i)} \tag{5.16}$$

であるが、今、被写体までの距離 Z は、レンズの焦点距離 f_iや、被写体の奥行Δより十

分大きいと仮定すると、

$$B_i \cong \frac{Lr\Delta}{Z^2} \tag{5.17}$$

となり、被写体を直接見た場合の両眼視差変化量と同じになり、同じ奥行量が知覚される。

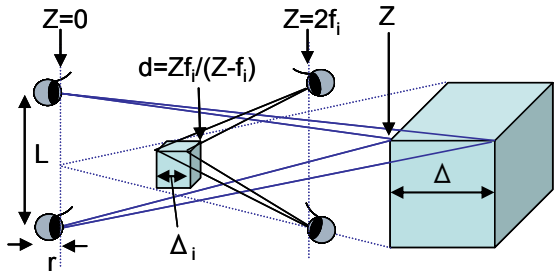


図 5- 12　射影立体映像の両眼視差

同様にして、輻輳角変化量 C は、被写体を直接見た場合、(4.9)式より、

$$C = (\frac{1}{Z} - \frac{1}{Z+\Delta})L = \frac{\Delta L}{Z(Z+\Delta)} \tag{5.18}$$

射影された立体映像を距離 2fi から見た場合、

$$C_i = (\frac{1}{2f_i - \frac{Zf_i}{Z-f_i}} - \frac{1}{2f_i - \frac{(Z+\Delta)f_i}{Z+\Delta-f_i}})L = \frac{\Delta L}{(Z-2f_i)(Z+\Delta-2f_i)} \tag{5.19}$$

今、被写体までの距離 Z がレンズの焦点距離 f_i の 2 倍より十分大きいと仮定すると、

$$C_i \cong \frac{\Delta L}{Z(Z+\Delta)} = C \tag{5.20}$$

となり、直接被写体を見た場合と同じ輻輳角変化量が得られる。輻輳角そのものは直視の場合と異なるが、それは被写体までの絶対距離感であり、映像の奥行感はその変化量で与えられる事は 4 章でも述べた。この事は、我々が通常経験する、遠くの景色には非常に大きな奥行があっても扁平に感じられ、近くにある厚みの薄い被写体と同じ奥行に知覚される事からも説明される。

同様に、焦点調節変化量 A も、被写体を直接見た場合、(4.10)式より、

$$A = \frac{r^2(Z+\Delta-Z)}{(Z+\Delta+r)(Z+r)} = \frac{r^2\Delta}{(Z+\Delta+r)(Z+r)} \tag{5.21}$$

であるが、被写体までの距離 Z やその奥行 Δ に対して眼球サイズ r は十分小さいとみなせるので、

$$A \cong \frac{r^2\Delta}{(Z+\Delta)Z} \tag{5.22}$$

と近似できる。一方、射影立体映像を見た場合の焦点調節変化量 A_i は、

$$\begin{aligned} A_i &= \frac{r^2(2f_i - \frac{(Z+\Delta)f_i}{Z+\Delta-f_i} - (2f_i - \frac{Zf_i}{Z-f_i}))}{(2f_i - \frac{(Z+\Delta)f_i}{Z+\Delta-f_i} + r)(2f_i - \frac{Zf_i}{Z-f_i} + r)} \\ &= \frac{r^2\Delta f_i^2}{(f_i+r)^2(Z+\Delta - \frac{f_i(2f_i+r)}{f_i+r})(Z - \frac{f_i(2f_i+r)}{f_i+r})} \end{aligned} \tag{5.23}$$

となるが、被写体までの距離Zやその奥行Δがレンズの焦点距離 f_i より十分大きい場合、

$$A_i \cong \frac{r^2\Delta f_i^2}{(f_i+r)^2(Z+\Delta)Z} \tag{5.24}$$

と近似できる。通常は原画像を拡大して見るので、看視距離 f_i も拡大され、眼球サイズ r より十分大きいと見なせ、

$$A_i \cong \frac{r^2\Delta f_i^2}{f_i^2(Z+\Delta)Z} = \frac{r^2\Delta}{(Z+\Delta)Z} = A \tag{5.25}$$

と近似でき、被写体を直接見た場合の焦点距離変化量に等しくなる。

運動視差に関しては、実物であれば 2π の運動視差を持つが、遠方の被写体の場合は運動視差を感じる程移動するのは困難である。射影された映像は近距離にあるので、大きく看視位置を変えられるが、大きく移動すると被写体の前景に隠れたオクルージョン部の映像の無い部分が見える為、やはり大きな移動は出来ない。近景の被写体の場合のみ、運動視差の知覚の違いが大きい。

以上の考察より、被写体が比較的近くにある場合を除いて、空間標本化法で撮影した立体映像を直接看視する事で、実物を見たのと同じ立体感が得られる事が分かる。

(b) レンズによる空間拡大

3 次元標本化映像を、図 5-13 に示す様に、焦点距離 f の凸レンズの後方 d'から f までの距離に配置する事により、レンズを通して虚像が距離 D から無限遠に形成される。この時、映像の位置 d'と虚像の位置 D は、次式の関係を満たす。

$$\begin{aligned} &[d', f] \Rightarrow [D, \infty] \\ &d' = \frac{Df}{D+f} \end{aligned} \tag{5.26}$$

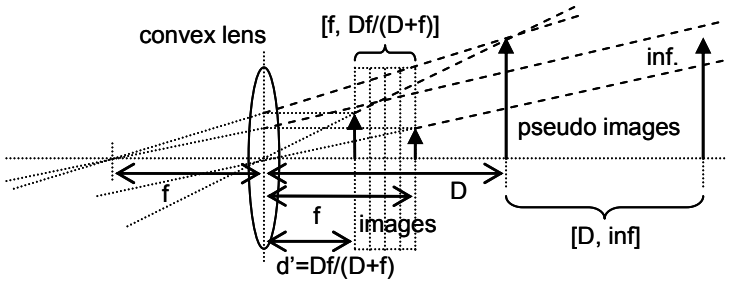


図 5- 13　レンズによる拡大看視

今、簡単の為にレンズの焦点距離 f を標本化時と同じとすると、レンズからの距離 d で撮影された映像を配置する位置 d'を 5.26 式の様に選べば、元の距離 D の位置に虚像が出来る。

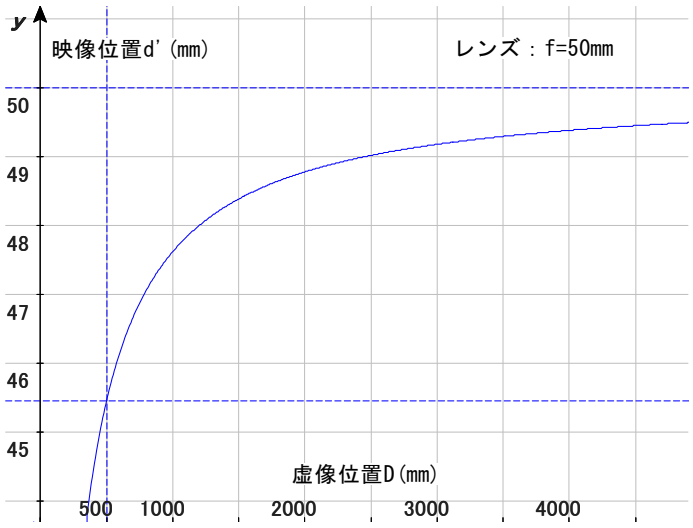


図 5- 14　f=50mm のレンズによる映像位置と虚像位置の関係

$$d' = \frac{df}{2d - f} \tag{5.27}$$

この時、再生映像の倍率は、(D+f)/f 倍になるので、表示映像のサイズを(D-f)/(D+f) = f/(2d-f)倍にする事により、元の位置に元の大きさの映像が形成される。

例えば、レンズの後方 d'=45.45...mm から 50mm の間に標本化映像を 0.818...倍～1.0 倍にして置けば、撮像時と同じ距離 50cm から無限遠の位置に元のサイズの再生像が形成される。 しかし、補正倍率は 20%以下であるので、補正なしでも近景の映像が若干大きく見える程度で、遠近感が少し強調されるが許容範囲と思われる。しかし、この方法は、再生像までの距離 D を大きく取ると、像の拡大率(D+f)/f が無限大まで増大し、レンズ収差による像の歪が無視出来なくなる課題がある。

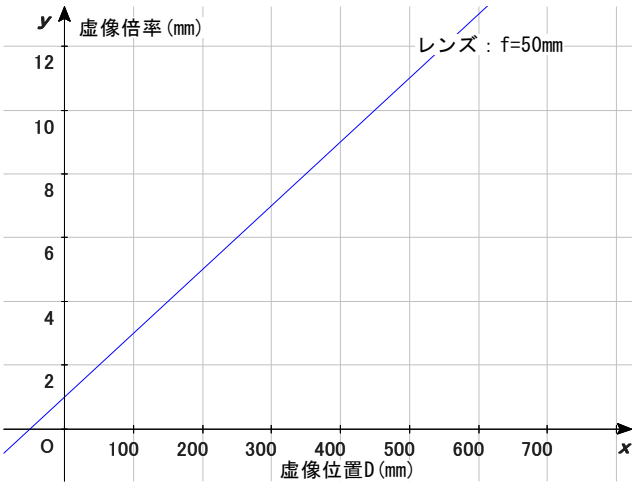


図 5- 15　f=50mm レンズでの虚像位置と倍率の関係

(c)　投影による三次元空間再生

図 5-16 に示す様に、図 5-1 と同じ構成で、凸レンズの後方 f から d の間に再生映像を配置して投影すれば、被写体が元あった位置に元のサイズの実像が形成される。このままでは、光はレンズの前方に進むので、この映像は無限遠から看視する必要があり実用的でないが、光路中に屈折スクリーンを入れて光を折り返せば、スクリーンの前後に立体映像が看視される。

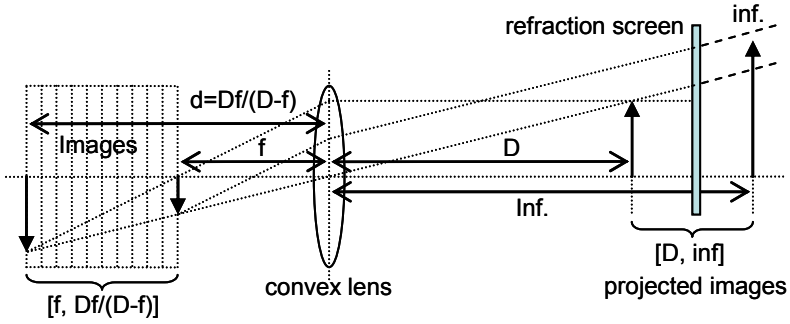


図 5- 16　レンズによる投影再生

屈折スクリーンは、図 5-17 に示す様な、ガラス球を敷き詰めたミラーで構成出来る[85]。ガラス球の屈折率が n=2.0 の場合、曲率半径 r /2 のレンズと見なせば、その焦点距離は、

$$f = \frac{n}{n-1} \times \frac{r}{2} \tag{5.28}$$

より、f=r となり、球の端面で結像する。この面を鏡面とする事により、入射光は球の端面で反射して入射方向に出射するので、スクリーンの手前で結像した映像は、再度同じ点で結像し、スクリーンの前方から看視出来る。スクリーン後方で結像する光も同じ方向に反射されるので、スクリーン後方の本来結像する位置に虚像が看視される。球の直径を 1 画素以下にする事により、映像の解像度を保つ事が出来る。

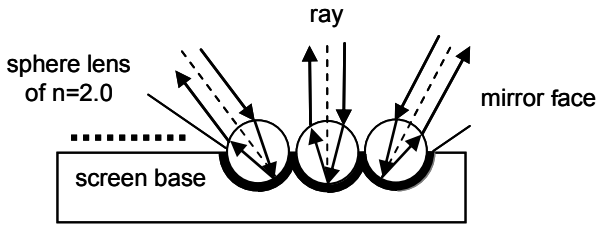
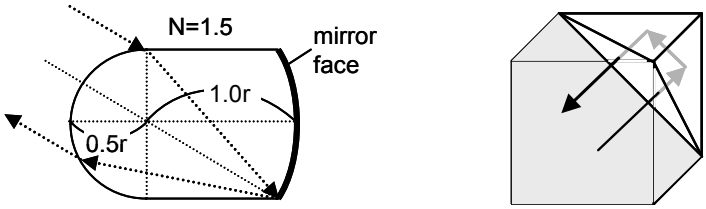


図 5- 17　n=2.0 の球レンズで構成した屈折スクリーン

屈折率 n=2 のレンズは既に実用化されているが[86]、屈折率 n=1.5 の通常ガラスを用いても、図 5-18(a)の様な砲弾型構造にする事により、反射範囲はやや狭くなるが、同じ効果を持たせる事が出来る。又、図 5-18(b)の様な立方体ガラスのコーナーを斜めに切り出したコーナーキューブを敷き詰めたスクリーンでも同じ機能を実現出来る。



(a)　砲弾型レンズ　　(b)　コーナーキューブ

図 5- 18　屈折スクリーン用素材

投影再生の場合、屈折スクリーンで反射された光は元の映像の方向に収束するので、視域が狭くなる課題があるが、投射する原映像サイズを十分大きくしたり、複数の投影機を用いる事で解決出来ると思われる。

5.4 空間標本化方式のデバイス

(a)　撮像デバイス

空間標本化法では、３次元撮像素子が必要になるが、通常の電荷結合素子(CCD）やCMOS センサの電極を透明にして重ね合わせる事により、実現可能であると思われる。図 5-19 に従来の CMOS センサの構造を示す[87]。光を検出するフォトダイオードは、n 型基板に不純物をイオン注入して低濃度の n 型領域を形成し、その上に高濃度の p 型領域を同様に形成して作成される。P 型基板をベースにする事も出来る。ダイオードで発生した電荷は、隣接する MOS トランジスタのゲート電圧を ON にする事により、ドレイン端子から取り出せる。この構造で、フォトダイオードや、基板下の電極、MOS トランジスタを透明にする事により、透明な CMOS センサが出来る。

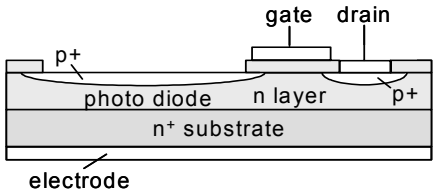


図 5- 19　CMOS イメージセンサの構造

透明な半導体材料としては､酸化亜鉛 ZnO､酸化インジュウム錫 ITO､酸化錫 SnO2、窒化ガリウム GaN などが知られており、透明トランジスタの試作も行われている[88][89][90]。又、色素系光電材料を使った透明なイメージセンサの試作も報告されているので、透明なイメージセンサは実現可能である。

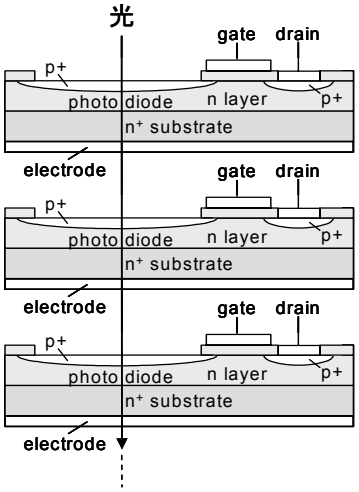


図 5- 20　透明 CMOS イメージセンサの多層配置

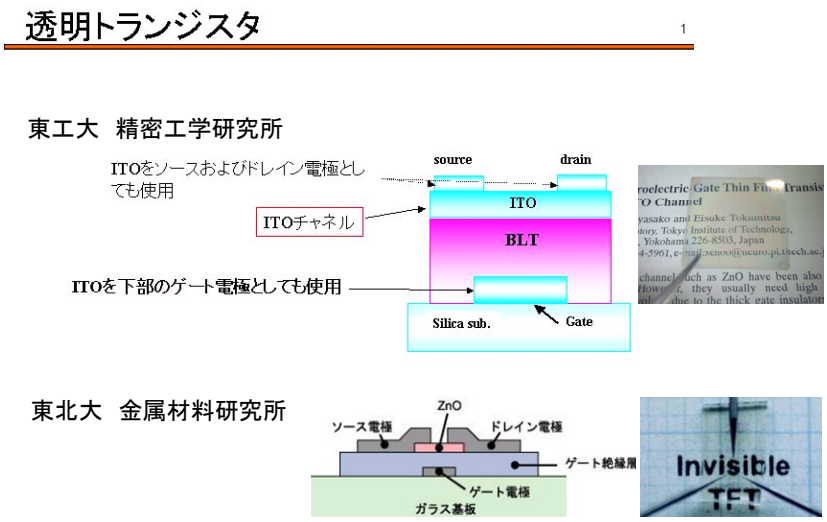


図 5- 21　透明トランジスタの例

これを多層化する為の課題は、透明度の向上、デバイスの等方性確保、フォトセンサの高感度化等であろう。映像のカラー化は、光の波長によって感度の異なる有機光電変換膜の使用[90]や、入射光線をハーフミラーやプリズムで 3 本に分け、夫々RGB3 色のカラーフィルタを通して単色光にしてから、3 次元イメージセンサで撮像すれば良い。

(b)　表示デバイス

表示素子は、液晶ディスプレイ(LCD）や、透明有機 EL パネルの多層配置で実現出来る。図 5-22 に LCD の構造を示す。LCD を構成する電極や駆動回路のトランジスタに、CMOS センサで述べたのと同じ透明デバイスを使い、これを重ね合わせて背後から一括照明する事で、立体映像を表示出来る。

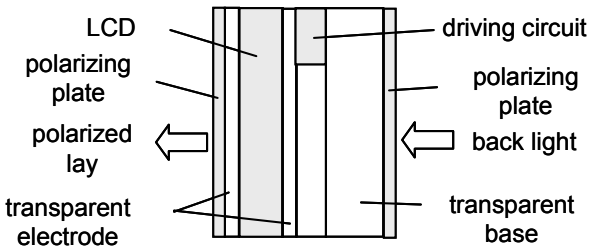


図 5- 22　液晶ディスプレイの構造

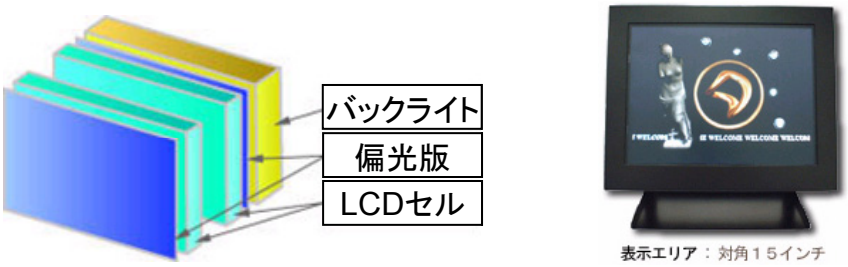


図 5- 23　２層 LCD ディスプレイの例（構造と外観）

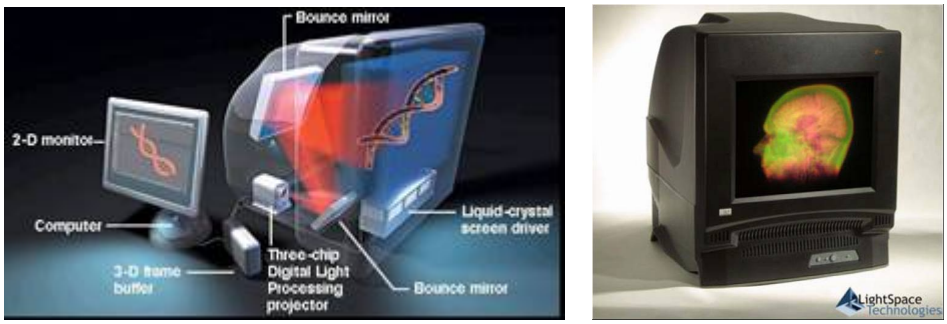


図 5- 24　20 層 LCD ディスプレイの例（構造と外観）

カラー化は、夫々の LCD の画素を 3 つに分け、RGB3 色のフィルタを付けるのが簡単であるが、背景色で前景が影響されるのを防ぐには、映像を RGB3 色に分け、夫々別の表示デバイスで単色立体映像を作り、ハーフミラーやプリズムで合成すれば良い。LCD は単に光の透過率を変えるシャッタであり、自身は発光しないので、透過率が 100%であれば、その LCD 上に映像は看視されず、映像の存在により透過率が下がると光の散乱が生じ、LCD 上に映像が看視され、その LCD の位置に応じた奥行が知覚される。多層 LCD による立体映像表示は既に実現されている[44][68]が、課題は背後から一括照明をする為に、オクルージョン部の映像面積を大きくすると背後の映像が透けて見えることである。

表示素子に、有機 LE パネル[92]や無機 EL パネル等の自己発光素子を用い、発光時には液晶シャッタ等で背後からの光を遮断する構造にすれば、バックライトが不要でかつ、オクルージョン部を大きく確保しても透けて見える事はなく、カラー化も容易である。有機 EL パネルは、CMOS センサとほぼ同じ構造であり、フォトダイオードを発光ダイオードで置き換えれば、透明な表示パネルを実現出来、無機 EL パネルでは既に透明ディスプレイが製作されている[93]。カラー化は各画素を RGB3 色に分けてやれば良く、背景色に影響される事もない。

5.5 空間標本化された 3 次元映像

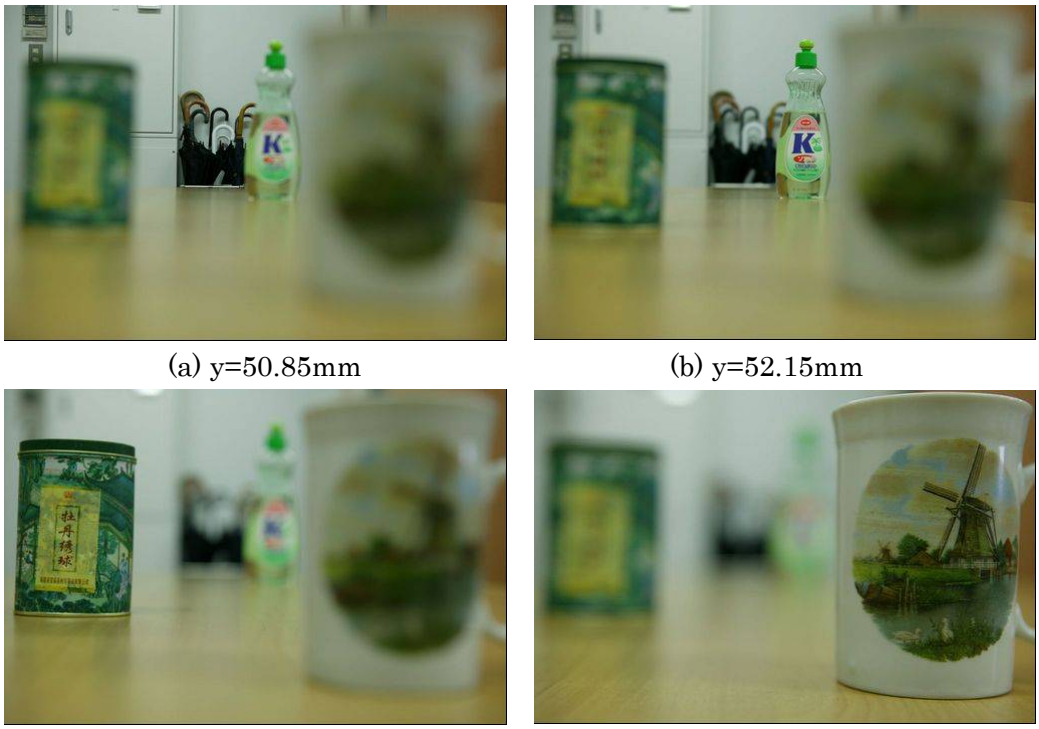
前節までに空間標本化法による立体映像生成の原理と撮像方法及び、表示方法に付いて述べたが、本方式に基づいて撮影した立体映像の各層の映像例を、図 5-25 及び図 5-26 に示す。ここでは、透明撮像素子は未開発であるので、従来の Depth from Focus 法[80]に基づき、レンズの位置を変えながら撮影を繰り返して空間映像の標本化を行った。使用したレンズの焦点距離は、f=50mm であり、レンズから撮像パネル位置までの距離は、図 5-25 の場合、レンズの後方 50mm、50.425mm、50.85mm、52.125mm であり、図 5-6 より、各映像で焦点の合っている距離は、∞、6m、3m、1.2m となる。

(a) y=50.0mm　(b) y=50.425mm

(c) y=50.85mm　(d) y=52.125mm

図 5- 25　空間標本化映像の例

同じレンズを用いて近距離を撮影した例を図 5-26 に示す。この例では、撮像板の位置はレンズの後方 50.85mm、52.15mm、54.75mm、57.35mm であり、図 5-7 より、各映像で焦点の合っている距離は、3m、1.2m、60cm、40cm となる。近距離での被写体までの距離とボケ量の関係を図 5-7 に示したが、近距離になる程、被写界深度は浅くなり、レンズと撮像板距離を大きく取る必要がある。



(a) y=50.85mm　　(b) y=52.15mm

(c) y=54.75mm　　(d) y=57.35mm

図 5- 26　近距離での空間標本化映像の例

これらの映像を重ねて表示すれば、人の目の特性として、フォーカスの合った映像のみを見る様に目の調節機構が働くので、奥行のある立体映像として看視出来るが、このままでは、各レーヤにボケた映像があり、多層表示した場合に暗くなったり 2 重像になるので、ボケ部分の透明化を試みる。

5.6 合焦点の抽出

ボケ部の透明化の為には先ず、映像の中でボケている部分を検出する必要がある。レンズによるボケは、ガウス分布型のローパスフィルタでシミュレート出来る様に、映像の高域成分が無くなる事であるので、低域成分のみの領域を検出すれば、ボケ領域と判定出来る。この目的の為に、低域成分に着目してその抽出を行うと、合焦領域ではテキスチャに依存して高域成分と低域成分が混在するので、誤検出の可能性が高くなる。逆に、映像の高域成分を抽出すると、ボケ部分には高域成分はないので、高域成分のない領域をボケ領域と判定すれば、正しくボケ領域を検出出来る。

(a)　高周波成分検出

高周波成分の抽出には、FFT や DCT などで映像の空間座標毎の輝度値を周波数座標の成分値に変換してから、高域成分を抽出する方法もあるが、同時に低域成分も計算され、処理量が増えるので、ここでは目的の高周波成分のみを、元の空間座標成分から直接抽出するフィルタリング操作を検討した。高周波成分を抽出するフィルタは、映像の一次微分を行うハイパスフィルタが一般的であるが、方向性を持つ為に、フィルタの方向を変えながら多重にフィルタリング処理を行わなければならないので、一度のフィルタリング処理で 2 次元映像の高周波成分が抽出出来る 2 次元 Laplacian フィルタを用いた。このフィルタは、2 次微分フィルタであり、輝度の傾きの大きい所を検出するのみでなく、輝度の傾きが変化する部分も感度良く検出するので、細かなテキスチャも検出出来る。タップ数が多い程出力値は高くなるが、帯域フィルタの特性になり高周波成分の検出感度が下がるので、感度は低くても細かなエッジを検出する少タップフィルタが良いが、少な過ぎると滑らかなテキスチャが検出され難くなる。図 5-27 に、5 タップ、3x3 タップ、5x5 タップ、及び 7x7 タップの Laplacian フィルタを示す。これらのフィルタを用いて、高周波成分を検出した結果を図 5-28 に示す。少タップフィルタの出力レベルは低く見難いので、全て 8 ビットに正規化して表示している。5 タップフィルタでは、オブジェクトのエッジは不明確であるが、合焦領域全体が検出される。タップ数が増える程、オブジェクトのエッジが明確に検出されるが、網目の様な細かなテキスチャが検出され難くなる。

ここでは、比較的滑らかなテキスチャも検出する為に、7x7 タップフィルタを用いて高周波成分を抽出した結果を 8 ビットに正規化したものを図 5-29 に示す。又、図 5-30 に、3x3 タップフィルタで高周波成分を抽出した例を示す。いづれの図からも分かる様に、各映像で合焦領域が明るく抽出されているが、テキスチャ依存性があり、滑らかなテキスチャでは検出レベルが下がると同時に、焦点深度が深いと隣合う撮像板間で、合焦点がどちらに属するのかが区別し難い。

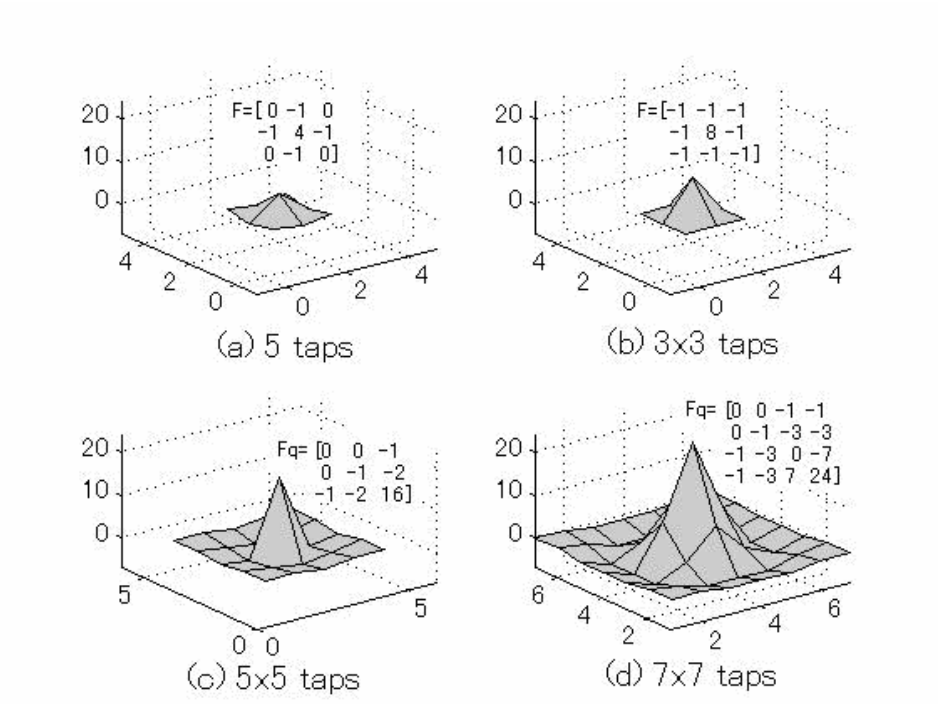
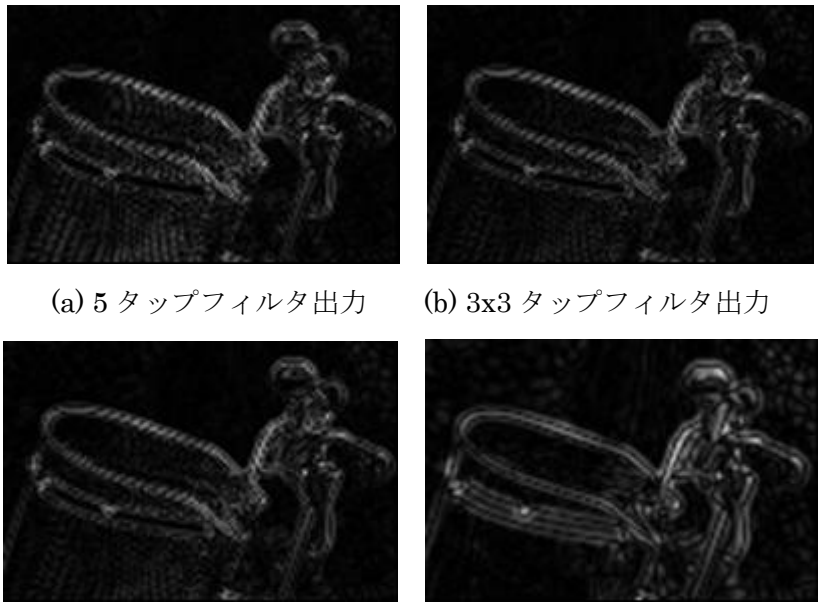


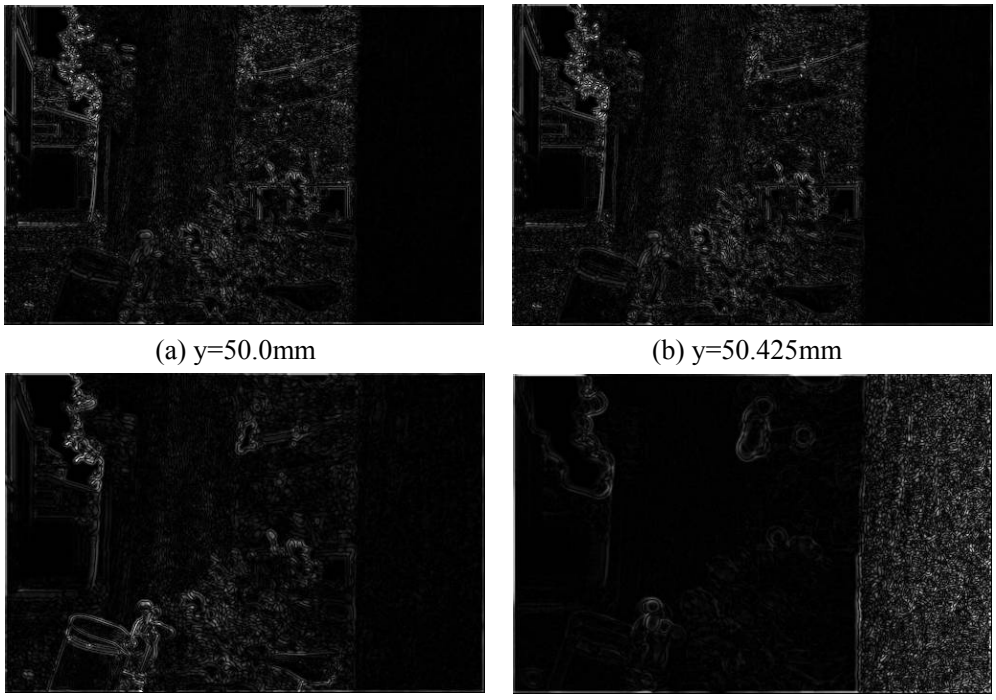
図 5- 27　各種の Laplacian フィルタ



(a) 5 タップフィルタ出力　　(b) 3x3 タップフィルタ出力

(c) 5x5 タップフィルタ出力　　(d) 7x7 タップフィルタ出力

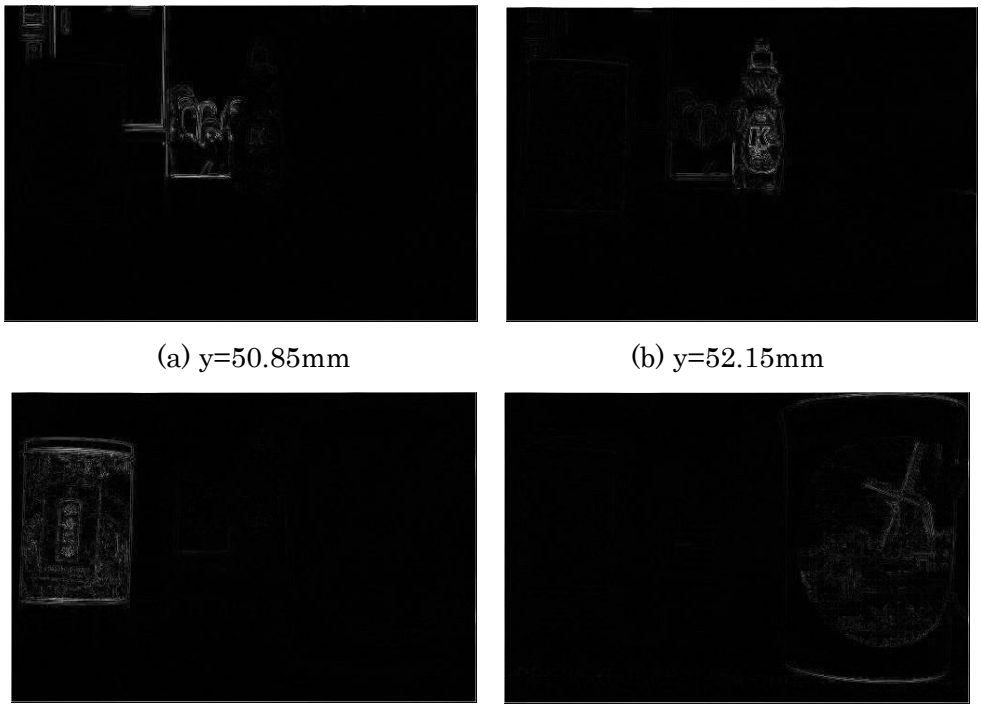
図 5- 28　Laplacian フィルタのタップ数と高周波成分検出結果



(a) y=50.0mm　(b) y=50.425mm

(c) y=50.85mm　(d) y=52.125mm

図 5- 29　7x7 タップ Laplacian フィルタ出力



(a) y=50.85mm　(b) y=52.15mm

(c) y=54.75mm　(d) y=57.35mm

図 5- 30　3x3 タップ Laplacian フィルタ出力

(b)　ピーク検出

合焦領域検出のテキスチャ依存性をなくすには、高周波成分量の絶対値検出ではなく、周りの値との相対値検出が良い。同一面内の相対比較ではなく、レーヤ間で高周波成分のピーク値を検出すれば、合焦レーヤが一意的に決められる。ピーク値は、テキスチャに依存して変動するので、これをなくす為には、2 値検出が適する[83]。図 5-32 及び図 5-33 に 2 値のピーク検出結果を示す。

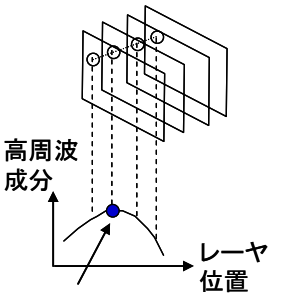
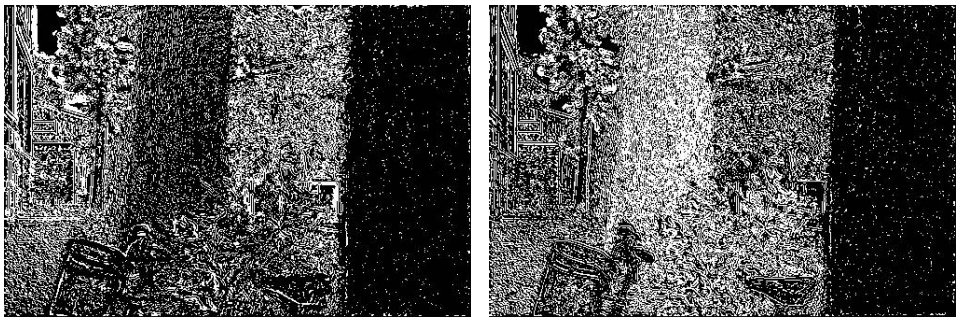
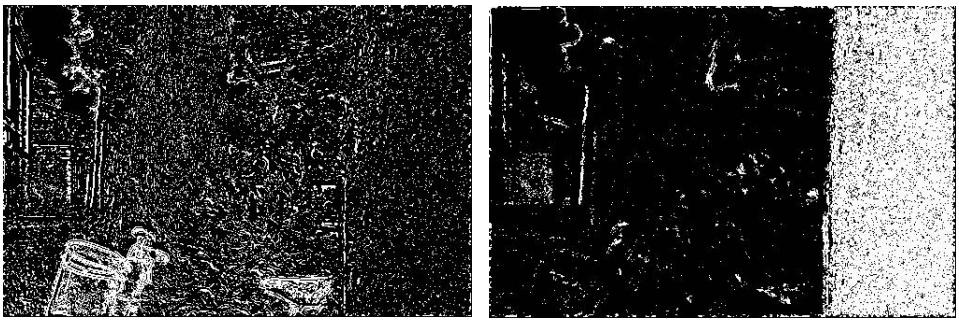


図 5- 31　ピーク検出

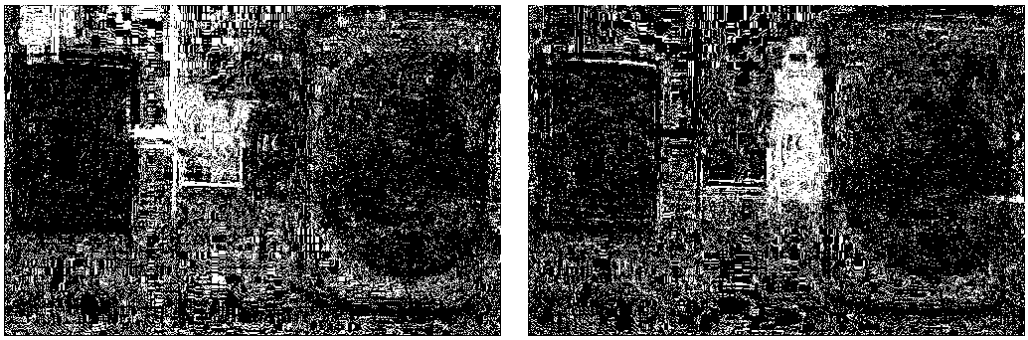


(a) y=50mm　　(b) y=50.425mm



(c) y=50.85mm　　(d) y=52.125mm

図 5- 32　7x7 Laplacian 出力のピーク検出結果



(a) y=50.85mm　　(b) y=52.15mm

(c) y=54.75mm　　(d) y=57.35mm

図 5- 33　3x3 タップ Laplacian 出力のピーク検出結果

夫々のレーヤで、合焦点がほぼ精度良く検出されているが、Laplacian フィルタが高感度である為にノイズの影響を受け易く、又、夫々のレーヤーのテキスチャのボケ量の違いにより高周波成分の出る位置がずれる為、合焦領域にボケ点（黒点）が見られ、ボケ領域にピーク点（白点）が見られる。又、Laplacian フィルタのタップ数が多い方がノイズが少なく、オブジェクトの形状やテキスチャがシャープに出る傾向にあり、タップ数が少ない方がノイズの影響が大きく、オブジェクトの形状やテキスチャの境界が不明瞭になる傾向がある。

5.7 合焦領域の決定

Laplacian フィルタ出力のピーク検出のみでは、合焦領域の中にボケと判定された点が存在し、ボケ領域の中に合焦と判定された点が混在するので、このままでボケ領域除去を行うと、ボケ領域の中に消し残りが出ると共に合焦領域に穴が空くので、ボケ領域と合焦領域の塗りつぶし操作が必要である。

(a)　孤立点除去

不要な孤立点除去には、Median フィルタや Morphological フィルタが使われる事が多いが、いずれも大きな孤立領域除去は、演算量が大になる事や、合焦領域とボケ領域の境界がシャープに出るので、後で述べる映像レーヤ間の輝度分配で奥行解像度を上げる事が困難になる。そこで、ここでは、ローパスフィルタにより合焦領域のエッジのボケ量の線形性を保存したまま、孤立点除去を行う方法を検討した。図 5-34 に検討したローパスフィルタを示し、図 5-35 にフィルタ出力例を示す。少タップのフィルタでは、孤立点が除去されず残り、タップ数を増やす程孤立点は除去されるが、合焦領域の境界も

大きくボケる。図 5-36 に、21x21 タップのフィルタを 2 回通して孤立点除去を行った結果を示し、図 5-37 に、43x43 タップフィルタを 2 回通した結果を示す。これらのフィルタの形は、元のフィルタと同じで、サイズのみが 2 倍になったものであり、セルフコンボリューションで作ったフィルタを 1 回通すのと同じである。

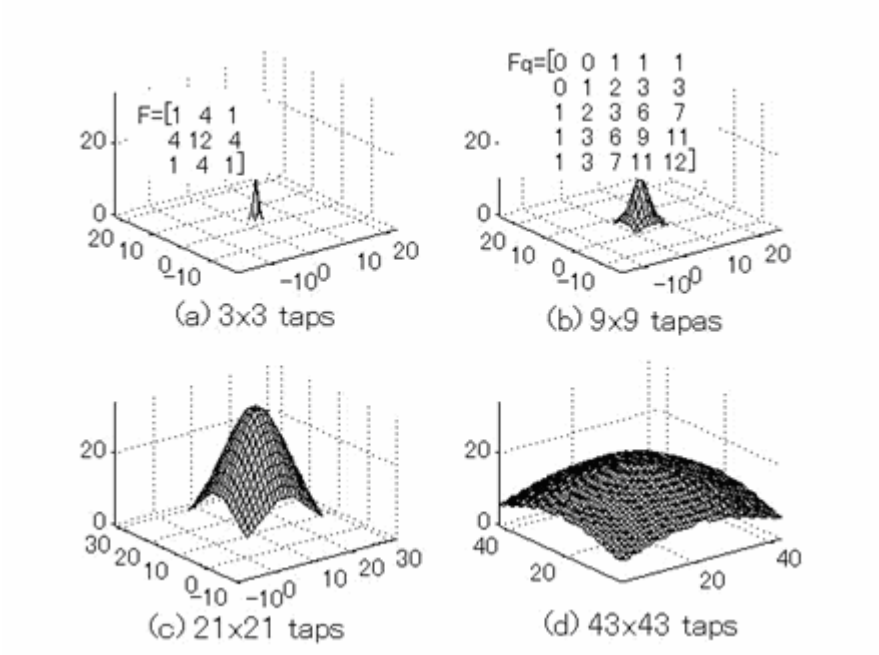
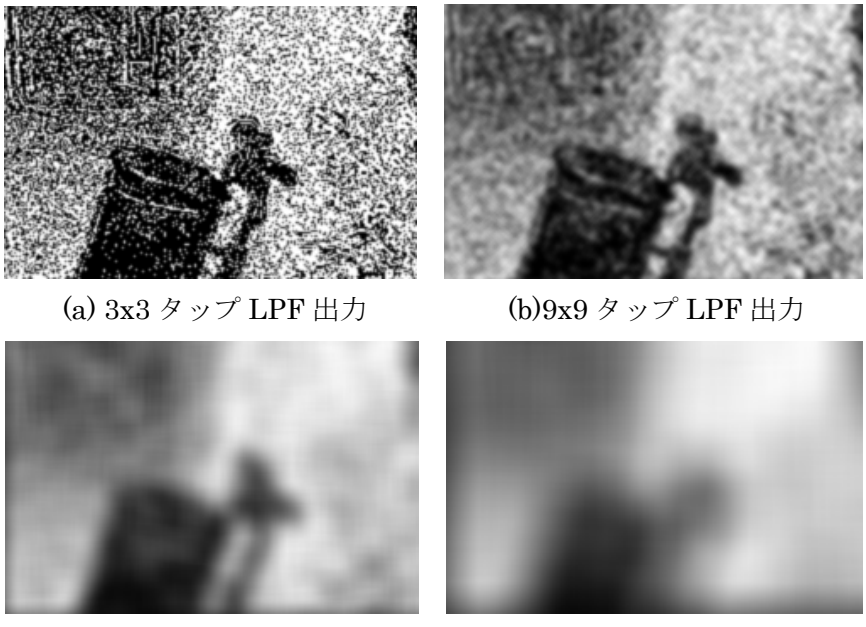


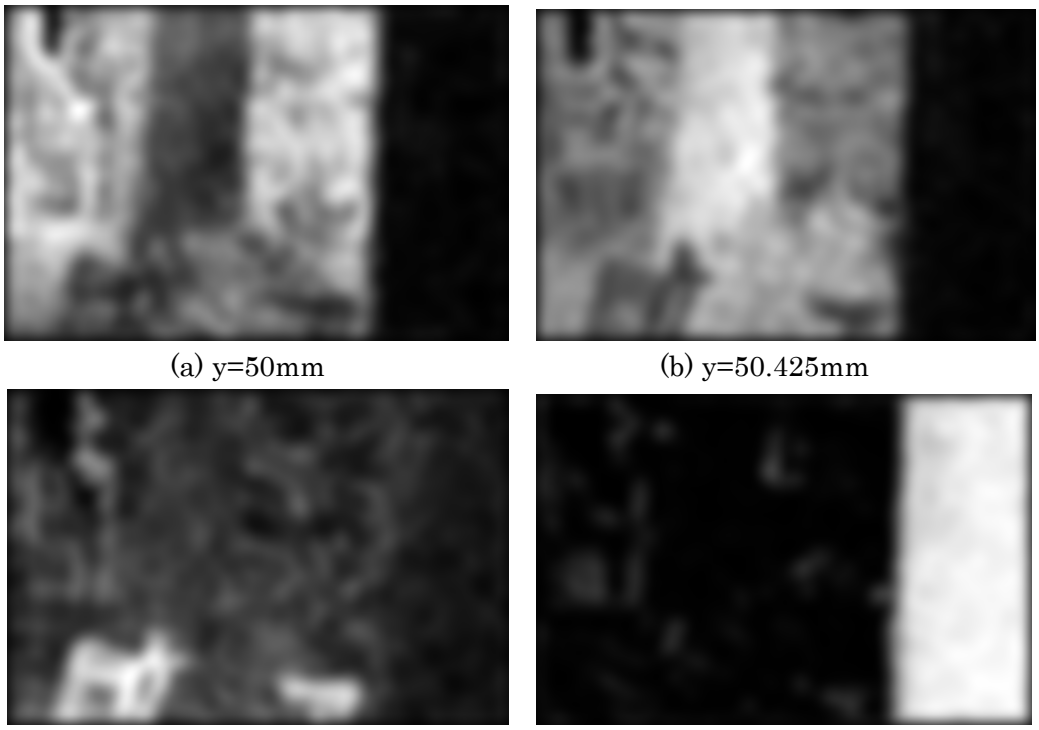
図 5- 34　各種の 2 次元ローパスフィルタ



(a) 3x3 タップ LPF 出力　　(b)9x9 タップ LPF 出力

(c) 21x21 タップ LPF 出力　　(d) 43x43 タップ LPF 出力

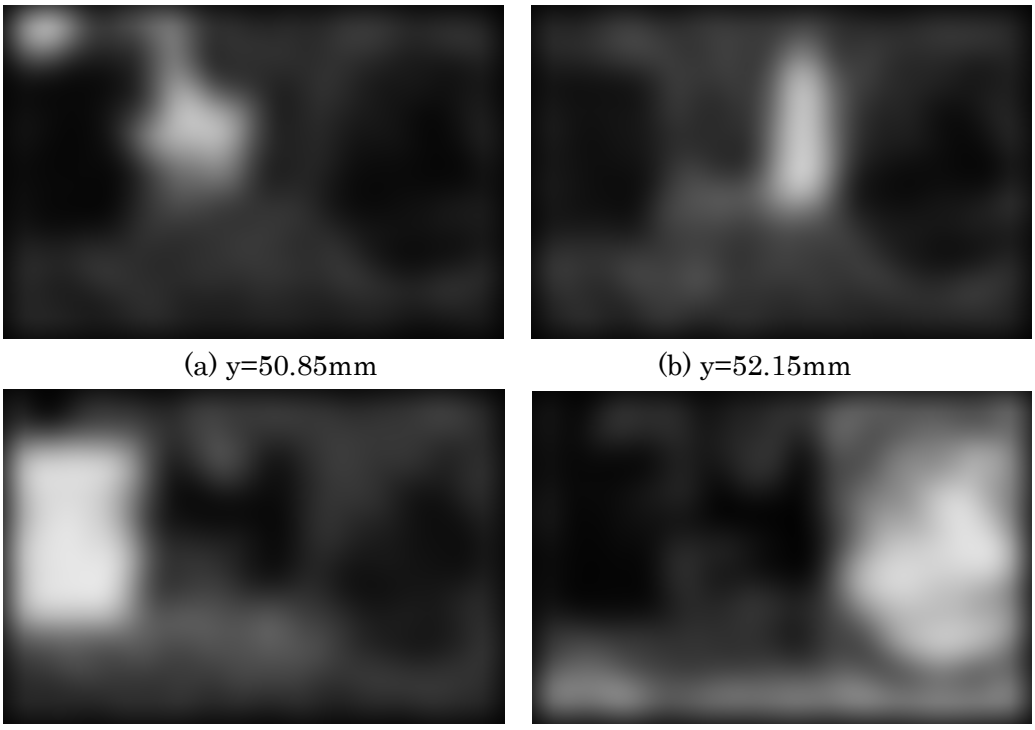
図 5- 35　各種ローパスフィルタの出力例



(a) y=50mm　(b) y=50.425mm

(c) y=50.85mm　(d) y=52.125mm

図 5- 36　21x21 タップ LPF を 2 回通した結果



(a) y=50.85mm　(b) y=52.15mm

(c) y=54.75mm　(d) y=57.35mm

図 5- 37　43x43 タップ LPF を 2 回通した結果

(b)　領域境界の決定

ローパスフィルタで孤立点は消去出来るが、合焦領域とボケ領域の境界が不明瞭になるので、次に、この領域境界の再生を検討する。ローパスフィルタは線形であるので、フィルタ出力では境界が不明瞭であるが、境界位置は保存されている。この位置を明確化するには、合焦領域からボケ領域に遷移する部分の傾きを線形に調整すれば良い。その為に、次式に示す整形関数をフィルタ出力に適用して領域境界の再生を行う。

$$z = \frac{255}{1+\exp(\beta - x/\alpha)} \tag{5.29}$$

ここで、x はローパスフィルタ出力を 8 ビットに正規化した値であり、α は合焦部からボケ部に遷移する部分の傾きを与え、β は遷移中心の閾値を与える。図 5-38 に、α=16、β=8 の場合の整形関数を示すが、α を大きく取る程、ボケ部や合焦部のむらを除去出来るが、合焦領域からボケ領域への遷移が急峻になる為、次に述べる奥行分解能が上がらなくなる。又、β の値は、高周波成分のピーク検出やその後のローパスフィルタ処理でロスがなければ、閾値が 128 となる様に選べば良いが、実際にはロスが存在し閾値が下がるので、その分を補償した低めの値に設定する必要がある。

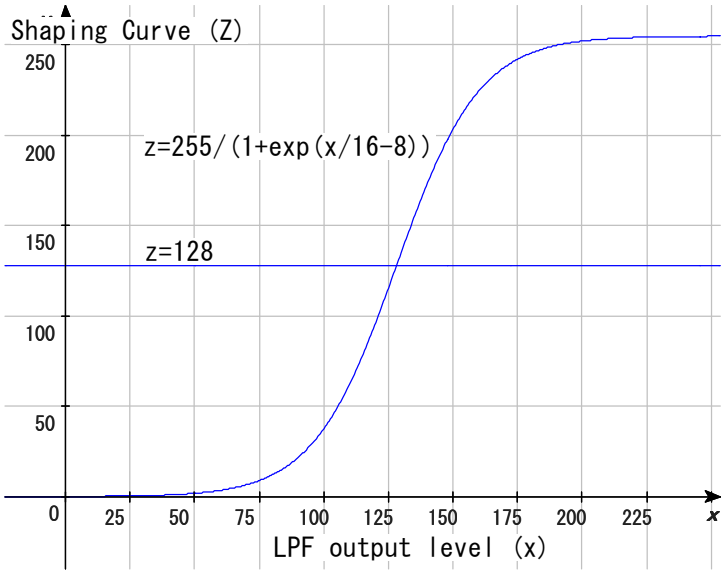


図 5- 38　領域境界の整形関数

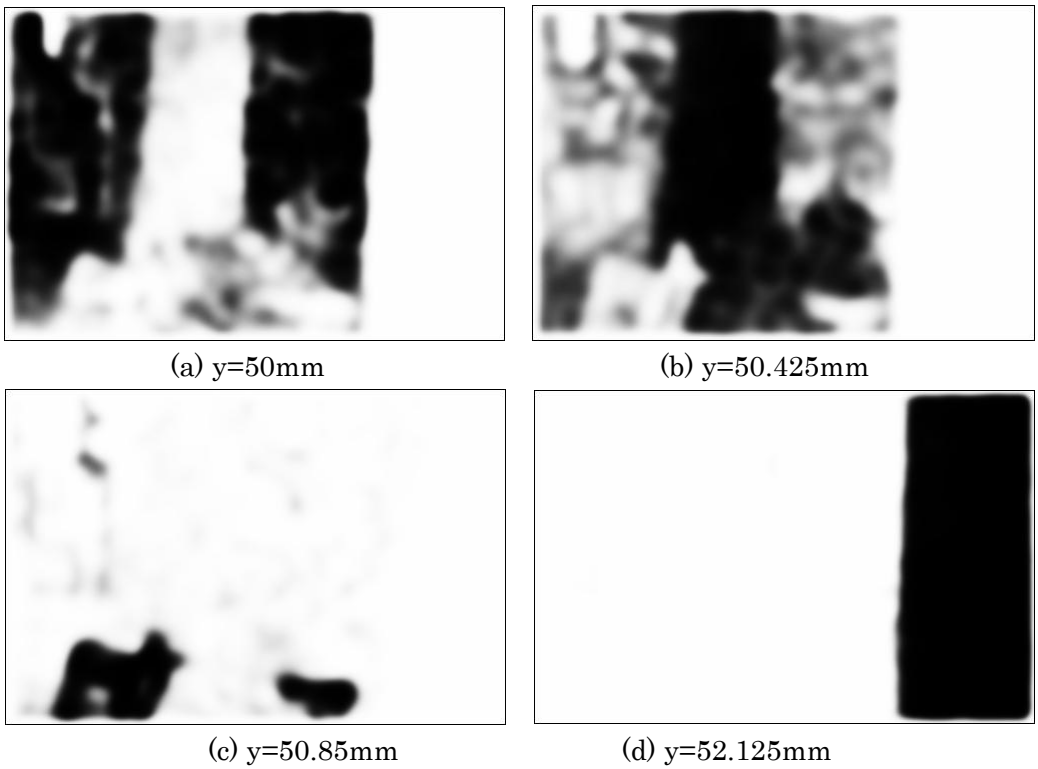
5.8 ボケ領域の削除

(a)　輝度オフセット

ボケ領域が決定されれば、その領域に輝度オフセットを加える事により、ボケ領域を透明化出来る。映像の高域成分のピーク値から抽出したローパスフィルタ出力は、合焦領域程レベルが高くボケ領域程レベルが低いので、このレベルを反転したものが輝度オフセットとなる。その為に、(5.29)式の exp 項に-1 を掛けた次式を用いる。

$$z = \frac{255}{1+\exp\left(x/\alpha - \beta\right)} \tag{5.30}$$

この式を用いて α=16、β=8 として、図 5-37 に示す高域成分のローパスフィルタ出力を輝度オフセットに変換したものを図 5-39 に示す。



(a) y=50mm　　(b) y=50.425mm

(c) y=50.85mm　　(d) y=52.125mm

図 5- 39　輝度オフセット

図より、焦点の合っている部分はオフセット値がゼロで暗く、ボケている領域程大きなオフセット値になり明るくなっているのが分かる。グレーに見えている部分は、被写体の位置が撮像板の各レーヤの中間に位置していたもので、後で述べるレーヤ間の融像効果で、中間の奥行位置に像が再生される。

(b) ボケの透明化

このオフセット値を原画に加えたものを図 5-40 に示す。図より、各レーヤで、焦点の合っている部分のみが残っている事が確認出来る。(b)の映像の中で、中央の太い幹の両側に薄く残っている樹木の映像は、(a)のレーヤと(b)のレーヤの中間に位置するもので、各レーヤを重ねて看視する事により、両者のレーヤの中間の奥行位置に定位して見える。

(a) y=50mm　(b) y=50.425mm

(c) y=50.85mm　(d) y=52.125mm

図 5- 40　ボケ領域を透明化した映像

5.9 中間領域の再生

立体映像の奥行分解能を上げるには、撮像板の枚数を増す方法もあるが、以下に述べる融像効果により、撮像板の枚数を増やさなくても奥行解像度を上げる事が可能である。

(a)　輝度分配による奥行再生

文献[68][70]によれば、同じ映像を 2 枚、距離を変えて重ねて提示し、両者の輝度比を変えると、前後像の間で融像が生じ、見掛けの像位置が、2 枚の映像間の距離を輝度

比で内分する位置に定位する事が報告されている。この様な融像が生じるメカニズムは、同じ像をその距離を変えて見た場合、両眼間に距離がある事により、像のエッジが２段階に色の濃さが変わる階段状に見えるが、そのズレ量は僅かであるので、デジタル文字を滑らかに見せる為に良く知られている、色の濃さ分布によるアンチエイリアス効果により、像のエッジ位置が見かけ上、２つの色の濃さで内分される位置までずれる。前後像の色の濃さが同じであれば、左右眼で同じアンチエイリアス効果が生じ、像の見掛けのエッジ位置は、前後像のズレの中間位置になる。これを両眼で見ると、視差のある映像を見る事になり、両眼融合により、両者の見掛けのエッジが重なる距離に像のエッジがあると感じる。前後像の色の濃さが異なると、見掛けのエッジ位置が色の濃い方に引っ張られるので、両眼視差により、見掛けのエッジの奥行位置が、色の濃い像側に引っ張られ、像の奥行位置が変わる。この変化はほぼ線形である事が報告されている[68]。

従って、多層撮影された被写体の奥行量が分かれば、その奥行量を前後の表示パネルの像の輝度比に換えて表示する事により、表示パネルの間にも映像を定位させることが出来、奥行分解能を上げる事が出来る。融像効果による立体映像に関しては、当初は、小型ディスプレイの試作が行われたが最近は、投射型の大型ディスプレイも試作されている[69]。又、先に紹介した液晶パネル 20 枚を重ねた立体ディスプレイでも融像効果が利用され、奥行分解能を 30 倍高めている[44]。

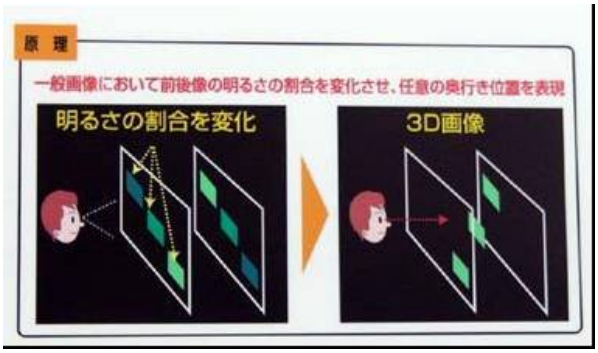
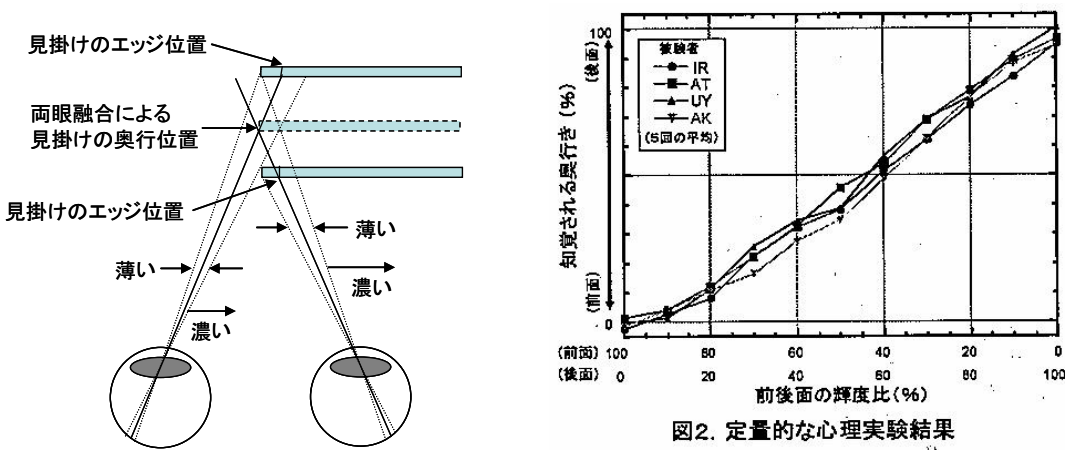


図 5- 41　立体錯視現象に基づく奥行表示



(a) アンチエイリアス効果による融像　　(b) 輝度比と奥行知覚量[68]

図 5- 42　立体錯視の生じるメカニズム

(b)　ボケ量と奥行

被写体距離と像のボケ量の関係を表す 5.2 式を変形し、撮像パネルの位置 y をパラメータに、ボケ量 δ を像の位置 d で表すと次式となり、これをプロットすると、図 5-43 に示す様に、撮像板から像までの距離 |y-d| とボケ量 δ はほぼ比例関係にある。

$$\delta = \frac{L\Delta}{d} = \frac{L|y-d|}{d} \tag{5-31}$$

従って、隣り合う撮像板間の像のボケ量の比は、その像が本来焦点を結ぶべき位置を、2 枚の撮像板間距離の内分比で表した値に等しい。従って、(a)で述べた議論に基づいて、ボケ量の比を映像の輝度比に変換すれば、融像効果で得られる像位置は、本来その映像が焦点を結ぶ位置であり、その像の奥行位置が正しく再生される。

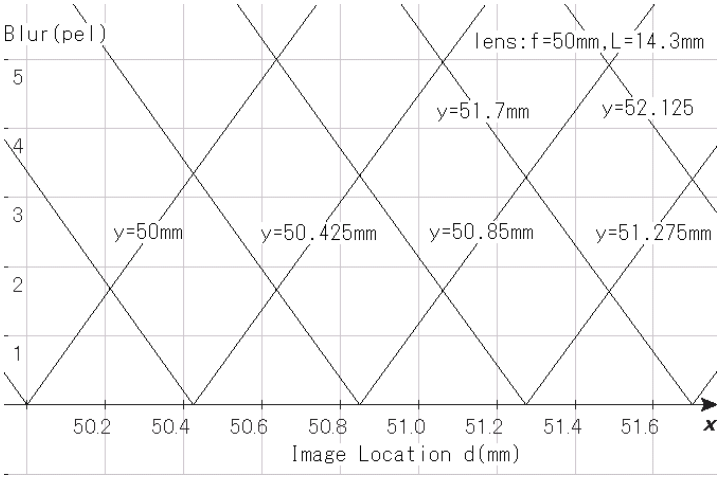


図 5- 43　映像位置 d による撮像板 y 上のボケ量

(c)　ボケ量の輝度分配による奥行再生

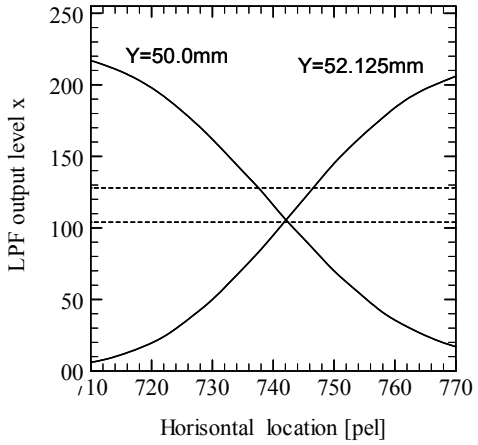
ボケ量の輝度値変換は、5.30 式で行われるが、連続する 2 枚の撮像板の間で焦点を結ぶ映像のボケ量は、図 5-38 の合焦領域の閾値（z=128）のまわりの値になる。この部分で整形関数はほぼ直線とみなせるので、2 枚の撮像板上でのボケ量の比は、この整形関数により輝度比に変換される。α を変える事により直線部分の傾きが変わるが、この傾きは 2 枚のパネルで共通であるので、2 枚のパネル間の輝度比を取れば、直線の傾きはキャンセルされ輝度比に影響せず、いずれの場合も中間位置の映像が再生される。

5.10 空間標本化映像の高品質化検討

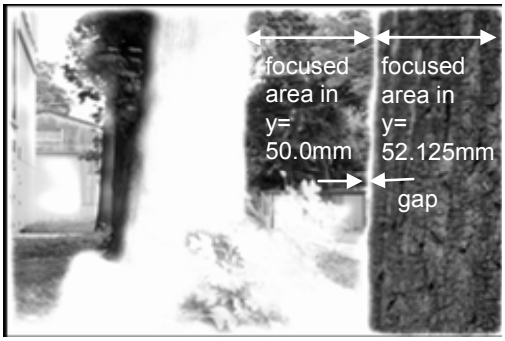
撮像レーヤの中間に合焦する被写体は、上述の融像効果により、レーヤ間を滑らかにつないで表示する事が出来るが、大きく距離の異なる被写体の境界で、映像間にギャップが出来易い現象がある。以下では、この現象の原因解明と、解決方法を検討する。

(a)　前景のボケによる背景領域の変化

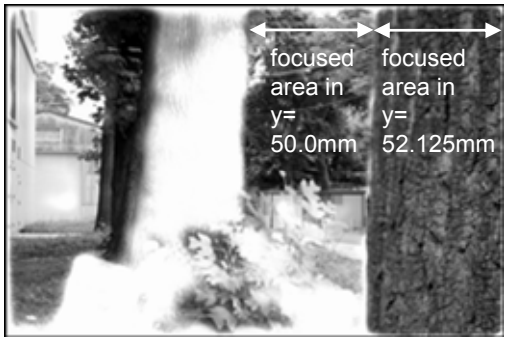
合焦領域の閾値は、5.30 式の β の値で決まるが、β を大きくし過ぎると閾値が上がり、合焦部の一部まで透明化され、多重表示した映像に穴が空く。図 5-44(a)に y=50.0mm と y=52.125mm のレーヤの高周波成分の LPF 出力を、上端から 42 画素（図(b)、(c)の矢印）の位置でスライスした値をプロットしたものを示す。図より、両レーヤの高周波成分は、閾値=約 104 で交差する事が分かる。今、5.30 式による透明化処理を、α=16、β=8（整形の閾値 128）で行った場合、両方のレーヤーを重ねたものを図(b)に示すが、手前の幹と背景の樹木の葉の間に僅かに、ギャップが観測される。このギャップが発生する理由としては、背景映像撮影の際に、前景のボケた幹の映像が背景映像にかぶり、背景の合焦映像の一部がボケとして検出され、合焦検出結果のピーク値が低下した為と思われる。そこで、透明化処理の閾値が図(a)のグラフの交点になる様に、α=16、β=6.5（閾値 104）として透明化処理を行った結果を図(c)に示す。図より、整形の閾値を下げる事により、ギャップがほぼ解消されることが分かる。全レーヤを重ねたものを、図 5-45 に示す。この前景のボケの影響は、背景映像に対してのみ生じ、背景映像のボケは、前景によって遮られる為、前景の映像には影響しない。



(a) 隣合うレーヤ映像の水平位置と映像の高周波成分抽出値 x の関係



(b) β=8 で整形した場合



(c) β=6.5 で整形した場合

図 5- 44　ボケ境界の決定

図 5- 45　α=16, β=6.5 で透明化処理をしたレーヤを重ねた映像

(b)　テキスチャ依存による合焦領域の閾値低下

合焦領域の閾値が低下する他の理由として、被写体のテキスチャが滑らかで殆ど高周波成分を含まない場合は、映像のレーヤ間でのピーク検出を行っても、検出値の S/N が

悪くなり、多くのノイズを含むので、抽出値のローパスフィルタ処理でロスが発生し、高周波成分が減衰する。この様な例として、図 5-46 にα=16、β=6 の整形関数でボケの透明化を行った近距離映像を示す。図では、カップの一部が透明化されている。

図 5- 46　テキスチャの少ない映像をβ=6 の整形関数で透明化処理を行った例

図 5-47 にβ=3.5 として透明化処理を行った場合の各レーヤ映像を示す。図から、各レーヤで合焦領域がその周辺も含めてほぼ完全に残っている事が分かる。この結果から、テキスチャによる閾値の低下も整形関数の閾値を調整する事で解消出来る事が分かる。閾値の決定は、整形後の透明化係数を全レーヤで掛け合わせる事により、合焦領域の和集合を作り、その和集合が透明値（255）を含まない様に、整形関数のβ値にフィードバックを掛ける事により、自動化出来ると思われる。

(a) y=50.85mm

(b) y=52.15mm

(c) y=54.75mm

(d) y=57.35mm

図 5- 47　α=16, β=3.5 の整形関数でボケ部の透明化処理をした映像

これらのレーヤを全て重ねたものを図 5-48 に示すが、ほぼ全画面が合焦映像で埋められている。一部、背景や床部分に輝度のむらがあるが、この部分はテキスチャが殆どない領域で、高周波成分抽出過程の低 S/N の影響により、合焦判定にノイズが乗った結果である。これの解決手段としては、感度と解像度の高いカメラを用いる事により、撮像素子のノイズ量を下げると共に、被写体にある僅かなテキスチャの変化の読み取りを可能にして、高周波成分の検出レベルを上げる事により解決されると思われる。

図 5- 48　α=16, β=3.5 で透明化処理し全レーヤを重ねた映像

(c)　オクルージョン

撮像パネルの位置が前後に異なる事により、図 5-49 に示す様に同じ見込み角の被写体 A、B の像サイズ a、b が、その距離 D1、D2 によって異なる。このサイズ比 a/b は、レンズから像までの距離 d1、d2 の比に等しく、次式で表される。

$$\frac{a}{b}=\frac{d_1}{d_2}=\frac{D_1(D_2-f)}{D_2(D_1-f)} \tag{5.32}$$

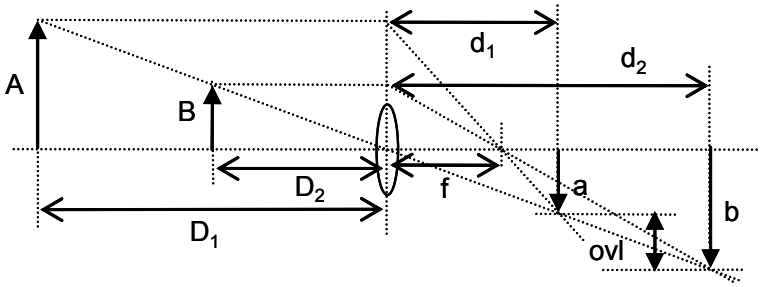


図 5- 49　レンズの光軸中心付近に近景がある場合のオクルージョン

通常この比はほぼ 1 に近く、例えば f=50mm のレンズで、D1=6m、D2=3m の場合、a/b=0.992 であり、背景 a が前景 b より約 1%小さい。直視型ディスプレイに表示する映像のサイズ比としては、補正なしでも違和感はない。画像中央に前景があり、周辺が背

景の場合、図 5-49 から、前景が背景に約 1%オーバーラップするので、オクルージョンが形成され、看視位置をずらす事が可能となり運動視差が獲得される。

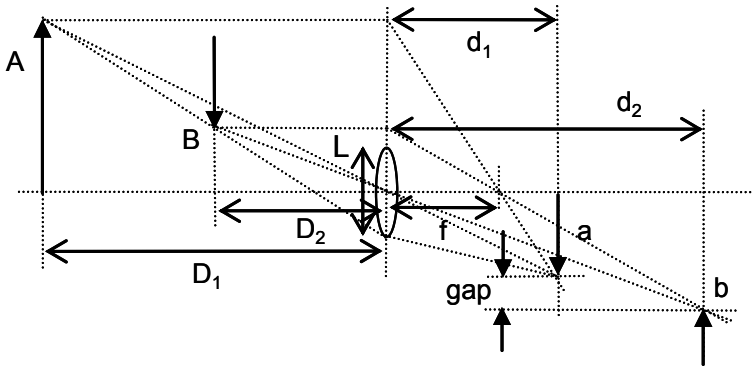


図 5- 50　画像の周辺に近景がある場合のオクルージョン

逆に図 5-50 に示す様に、画像周辺に前景があり、中心部に背景が見える構図では、前景に遮られて背景の写らない領域の内、前景の像 b でカバーされない領域(gap)が出来る。その大きさは、前景と背景の距離差(D1-D2)が大きい程大きくなるが、レンズ径 L の周辺から回り込んで写る背景により軽減され、次式となる。

$$gap = \frac{f(D_1 - D_2)}{D_2(D_1 - f)}\left(\frac{Bf}{D_2 - f} - \frac{L}{2}\right) \tag{5.33}$$

例えば f=50m、L=14.3mm、D1=∞、D2=1m、B=350mm（画面サイズの約 1/2）の場合、gap=0.56mm（画面サイズの約 1.6%）となる。gap を埋める手段としては、映像の透明化閾値を下げてボケ画像で埋める事も可能であるが、口径 L の大きなレンズを使う事や、前景と背景のサイズ比 a/b が 1.0 となる様に映像サイズを補正する事で解決される。又、動画では前後のフレームからオクルージョン部を補間する事も有効と思われる。

レンズ口径を大きくする事により、両眼視差分程度のオクルージョンはカバー可能と思われるが、看視位置を移動する事により生じる、前景の陰に隠れて見えなかった部分が見えてくる、運動視差分をカバーする為には、これをカバー出来る程度に主カメラの位置から離れた位置に補助カメラを置いたマルチカメラ構成とし、補助カメラ映像からオクルージョン部を取り出して、主カメラで取得出来なかった背景部分を埋める事により、運動視差分のオクルージョンの確保が可能と思われる。この処理は、主カメラと補助カメラの透明化係数の映像を、対応点が一致するように視差補償して、主カメラの透明化係数と補助カメラの透明化係数の差分として、補助カメラから見えるオクルージョン領域を得、この部分の映像を補助カメラ映像から切り出して、主カメラ映像に加えれば良い。補助カメラと主カメラの相対位置及び、各撮像板の位置が既知であるので、映

像の視差補償量は一意的に与えられる。又、主カメラと補助カメラを平行カメラ配置とすれば、おのおのの撮像板で撮影される映像の被写体までの距離はカメラによらず同じであるので、映像を平行移動するだけで、主カメラ映像と補助カメラ映像を一致させる事が出来る。

5.11　3D CG 映像との融合

この様にして得られた実写立体映像は、容易にコンピュータグラフィックスで作成した 3D CG 映像と融合させて、ハイブリッド立体映像空間を構成する事が出来る。その手順を図 5-51 に示す。

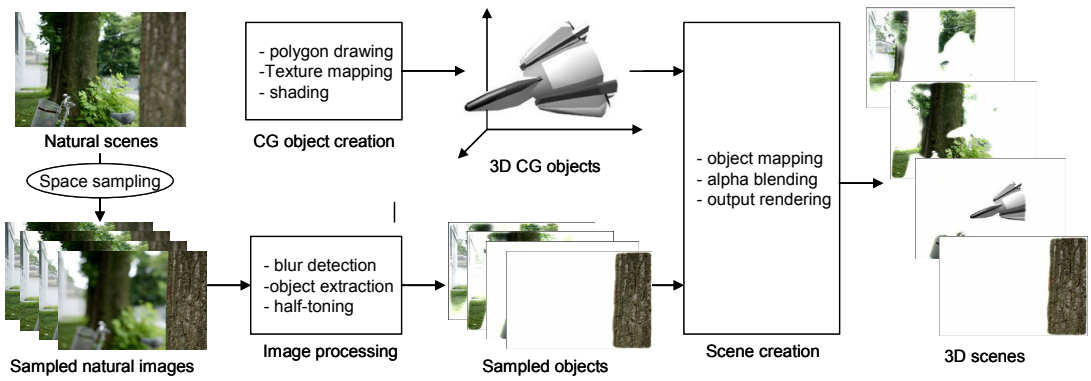


図 5- 51　実写立体映像と 3D CG 映像との融合

実写立体映像は、上述した手順で各レーヤ映像を作成して置き、CG 映像は、3D CG 映像描画ソフトウエアを用いて線形の立体空間で作成した後、図 5-1 に示す実空間と映像空間の射影関係を、斉次座標で表した次式を用いて、映像空間に変換する。ここで、[X, Y, Z]は、線形な 3D CG 映像空間の座標であり、[x, y, z]は、空間標本化法によって射影された映像空間座標であり、f はレンズの焦点距離である。z の値をカメラの撮像板の夫々の位置に取れば、3D CG オブジェクトを各レーヤ映像に分解出来る。

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \frac{f}{Z-f} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{5.34}$$

この時、CG オブジェクトは、実空間とマッチする様にそのサイズと位置を調整して置く。この様にして作成した実写映像と CG 映像のハイブリッド立体映像例を図 5-52 に示す。この例は、CG 映像は奥行があまり大きくなく、y=50.85mm の撮像板のみに射影さ

れた場合である。背景レーヤで CG 映像の陰になる部分は、射影された CG 映像から合焦領域を抽出し、その領域を引き去る事で重なり部分を削除し、前景映像の陰に入る CG 映像部分は、前景映像の合焦領域を CG 映像から引き去る事により、重なり部分を削除した。

図 5- 52　実写映像と 3D CG オブジェクトのハイブリッド映像

以上の映像演算処理は、映像データを行列演算関数で容易に処理出来る MATLAB のソフトウエアを用いて行った。

5.12　3 次元立体映像生成技術の考察

本研究の 3 次元立体映像生成技術で提案した空間標本化法による立体映像のメリットは、

（a）無限遠を含む 3 次元空間を、人の視覚特性に整合したコンパクトな映像空間として標本化するので、従来コンピュータビジョンやコンピュータグラフィックスの分野で行われていた様な線形空間座標でデータを持つ必要が無く、CG 映像データのみならず実写映像も含んだ 3 次元立体映像空間を構成する為に必要な映像データの大幅な削減が可能である。

（b）提案する立体映像空間は、真に奥行を持つ 3 次元映像であるので、両眼視差や輻輳角のみならず、焦点調節やオクルージョンによる運動視差を持ち、立体視要因をほぼ全て満たすので、立体視のために特殊な眼鏡や器具を必要とせず、自然な立体映像が看視出来、疲労や違和感が少ない。

（c）立体映像を生成する為のハードウエアとしては、映像の取得・表示に透明なデバイスを必要とするが、その動作速度は通常速度で良く高速性を要求しない。又、デバイスを奥行方向に積み重ねて立体化を行う為に、1 枚毎のデバイスの高解像度化が容易で、大画面高解像度の立体映像を容易に実現出来る。

（d）焦点の合ったオブジェクトの抽出は、複雑な画像処理や、レーザ光などによる距離計測を必要とせず、単純なフィルタリング処理の組み合わせで実現出来るため、撮影から表示までの処理の実時間化が容易である。更に、抽出された映像オブジェクトは、互いに排他的であり、立体映像でありながら、全体の映像データ量は平面映像のデータ量とほぼ同じであり、伝送・蓄積が容易である。

と言う特徴を持つ。

唯一の課題としては、1 視点映像である為に運動視差が少ない事である。2.2 節(b)で光線空間の考え方を述べたが、我々が自然界で見る映像は、3 次元空間にある被写体からあらゆる方向に出ている光線のうち、看視者の目のレンズを通して網膜上に焦点を結んだものである。従って、看視者の位置を固定しない場合、この光線は空間的な位置の自由度 3 と方向の自由度 2 を持ち、合計自由度 5 の巨大なデータとなる。

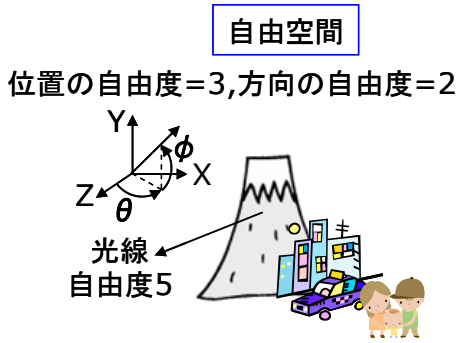


図 5- 53　3 次元空間の光線

多視点映像による立体映像（光線空間法）では、これを空間中に仮想スクリーンを固定して光線の位置の自由度を 2 に減らし、合計自由度 4 の光線を再生して立体映像を得ている。

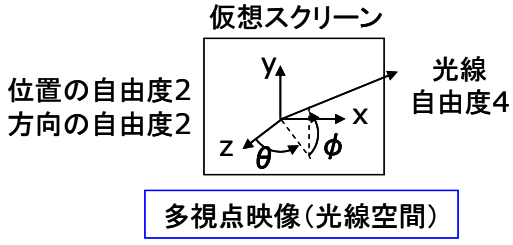


図 5- 54　多視点映像（光線空間法）による光線再生

光源は仮想スクリーンに固定される為、本来焦点調節効果はないが、光線の太さを十分細く絞って、同時に 2 箇所以上から出た光線が瞳の中に入る様にすると、目のレンズ

は２つの光線が網膜上で重なる様に焦点調節を行う結果、輻輳角と一致する点に焦点を合わせて像を看視する事が出来、運動視差の他に焦点調節効果が実現される。その為には、100 枚程度の多視点映像を多重化する必要があり、データ量が膨大となる。焦点調節効果を捨てれば、データ量はまだ多いが 10 枚程度の多眼式立体映像となり、ある程度の運動視差は確保される。

通常の映像鑑賞ではあまり動きまわる事は少なく、映画やテレビ観賞など視点位置が固定される場合が多いと思われるので、大きな運動視差確保の為に膨大な量の映像データを持つ事は無駄が多いと思われる。これに対し、本提案の空間標本化法による立体映像は、光線を凸レンズを通して視覚特性に合ったコンパクトな 3 次元実像に変換し、これを標本化する事で立体映像を再生するもので、再生される光線の自由度は 3 であるが、焦点調節効果を持ち、僅かであるが運動視差も持つ。

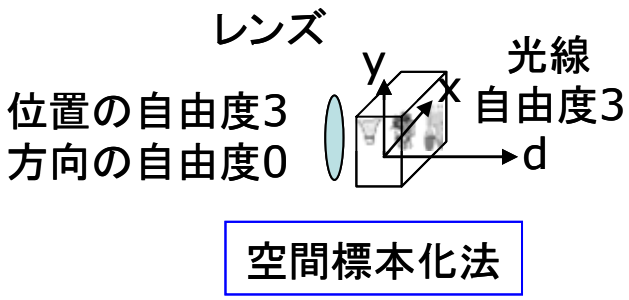


図 5- 55　空間標本化法による光線再生

１視点映像の為看視位置は固定されるが、全ての立体視要因を満たし、理想立体映像をコストパフォーマンス良く実現する手段として有効であると思われる。鑑賞用途であれば、運動視差の実現は撮影カメラの視点を動かす事で、画像要因として映像中に組み込み可能であるので、ハードウエアやデータ量の追加無く実現可能である。

表 5-1 に、多視点映像による立体映像と、空間標本化法による立体映像の原理と特徴の比較を示すが、多視点映像（光線空間法）による立体映像は、複数の看視者が同時に立体映像を見たり、看視者が自由に映像の周りを動きまわれる自由視点 TV としての用途に適すると思われるが、データ量が膨大となる課題がある。一方、空間標本化法は、看視位置は固定されるが、データ量は平面映像とほぼ同じで、コストパフォーマンス良く立体映像を実現し、個人鑑賞用の立体 TV としての用途に適すると思われる。

表 5- 1　多視点映像と空間標本化法による立体映像の比較

	原理	映像空間	視点数	データ量	両眼視差	輻輳	焦点調節	運動視差	主な用途
光線空間法	f(X,Y,θ,φ)	4D自由空間	多視点	×	○	○	○	○	自由視点TV
空間標本化法	g(x,y,d)	3D拘束空間	1視点	◎	○	○	○	△	立体TV

しかしながら、空間標本化法による立体映像であっても、立体映像空間を歩き回る様な仮想現実空間の実現や、それを応用したシミュレーション訓練などへの応用も期待され、この様な用途に拡張適用出来る事が望ましい。この為の大きな運動視差確保には、カメラの多視点化が考えられる。図 5-57 に示す様に、複数のカメラで異なった方向から被写体を撮影し、看視者の位置や動きに応じて表示映像を切り替えたり、屈折スクリーンに多重投影する事により、どの位置からも自然に見える立体映像を生成する事が出来る。

図 5- 56　空間標本化法による立体映像の課題

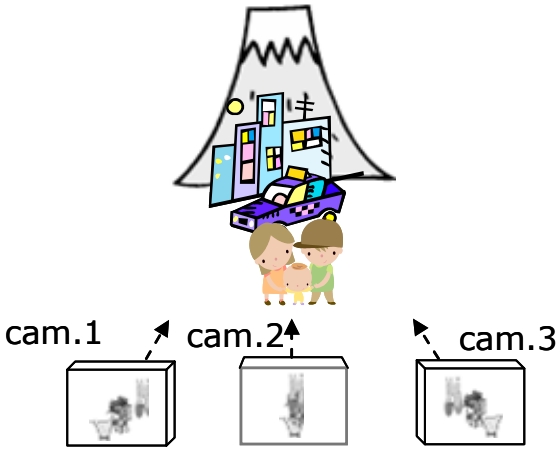


図 5- 57　マルチカメラ化による多視点立体映像

空間標本化法を多視点化するメリットは、視点数が少なくても焦点調節効果が実現されている事である。多視点化した場合の課題は、カメラの台数分、映像データ量が増す事であるが、各カメラ映像の間には相関があるので、これを利用してデータ量を減らせる可能性があり、この検討を次章で行う。

■ 第 6 章 ■

3 次元立体映像の符号化技術

第6章　3 次元立体映像の符号化技術

本章では、前章の検討結果から示俊される課題である、多視点化された立体映像データの圧縮符号化方法について検討する。

先ず、1 視点の空間標本化法で得られた映像データは、ほぼ等価的な 1 枚の映像を被写体までの距離に応じて多層の映像に分解したものであるので、各レーヤには別々の被写体が写っており、レーヤ間の相関は少ない。従って、これらの映像は従来の 2 次元平面映像の圧縮技術[1][2][3][4]を用いる事で、高能率な圧縮が可能と思われる。特に、MPEG-4 のオブジェクト符号化技術を用いれば、形状適応離散コサイン変換（SADCT）等を使って、任意形状オブジェクトを効率良く圧縮出来る。一部、レーヤ間に定位する被写体は、隣合うレーヤにまたがって存在するので、レーヤ間予測が効果的であるが、両者の輝度が異なるので、直接差分は予測効率が上がらず、輝度補正予測が効果的であると思われる。又、レーヤ間にまたがる映像は、基本的に同じ映像であり、輝度比のみが奥行位置によって変化する映像であるので、共通の映像情報は他の部分と同じ符号化方式を用い、輝度比のみ別途パラメータとして持てば、効率良く圧縮可能であると思われる。又、レーヤー間に存在する被写体は画面全体の一部であり、オクルージョン部の前景及び背景の映像は、別オブジェクトである為相関がなくレーヤ間予測の効果は少ないと思われる。従って、空間標本化法で得られた立体映像データのレーヤ間の冗長は殆ど無く、レーヤ内の 2 次元平面映像の相関を利用した圧縮がメインになるが、この技術は既に深く検討されており、既存の MPEG 等の圧縮技術で十分と思われる。

一方、大きな運動視差を確保する為の多視点立体映像は、同じ被写体を位置の異なる多数のカメラで撮影するので、各カメラ映像の相関が高く、カメラ間の相関を利用する事により、効率の良い圧縮符号化が行える可能性がある。この様な多視点映像は、データ効率は悪いが、任意の視点からの映像を容易に取り出せるメリットがある為、立体映像の能動的な鑑賞にも堪えうると言う特徴かあり、立体映像の静的鑑賞のみならず、自由視点 TV 等への応用も期待される。

多視点映像符号化方式の検討は、国際標準化機構（ISO: International Organization for Standardization）傘下の MPEG（Motion Picture Expert Group）グループで、MPEG-4 Part 10 (AVC)の MVC（Multiview Video Coding）拡張（Amd4）としてその標準化作業が始まっている事は、1.2 節及び 2.2 節でも述べたが、ここでは、MPEG から提供されている多視点テスト映像を用いて、被写体の面傾きに適応した高画質・高能率な符号化方式について検討する。

6.1 多視点映像と射影

先ず本節では、多視点映像と被写体の間に成立する射影方程式を解析し、各カメラ映像間に成立する関係を導く事により、効率の良い符号化方法を検討する。

(a) 多視点映像

多視点映像は一般的に、図 6-1 に示す様なカメラ配置で被写体を撮影して得られる。図の例ではカメラ n 台が一直線上に等間隔で並べられているが、MPEG では、被写体を囲む円弧上への配置や、水平垂直の格子点上への配置も想定されている。又カメラ姿勢は、全てのカメラ光軸が平行になる平行カメラ配置の他に、1 点に収束するような配置も想定されている。実際のカメラ配置では、想定されたカメラ間隔や光軸方向に若干のずれが生じるので、既知のテストパターンの撮影により、カメラの位置と方向を示す外部パラメータと、イメージセンサの位置と傾きとレンズの焦点距離よりなる内部パラメータを求め、このカメラパラメータを映像と共に送ったり、送信側で撮影映像を補正し、理想カメラ配置での映像に変換してから送信する事により、理想カメラ配置での映像が容易に得られる様になっている。これは、カメラパラメータの Rectification と呼ばれている。又、カメラ毎に輝度や色のばらつきがあり、これも符号化効率を低下させるので、これを補正した映像を符号化する事が高画質化と低ビットレート化に有効である事が MPEG でも報告されている。MPEG MVC のテスト映像の被写体としては、無限遠点を含む屋外映像や、有限の屋内人物映像等が用いられている。

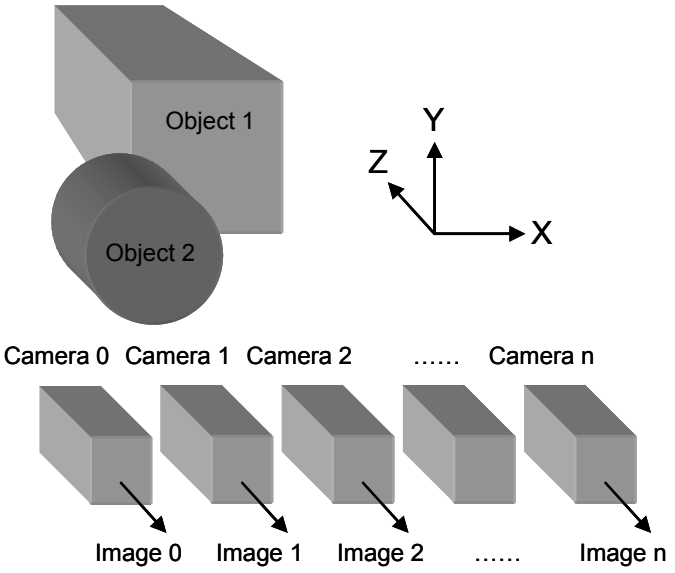


図 6- 1　多視点映像の撮影

(b)　射影方程式

一般に、世界座標系(X, Y, Z)で表した被写体面上の点(X, Y, Z)のカメラ映像座標系(x, y)への射影は、カメラのレンズ中心を原点とするカメラ座標系から見た世界座標の原点の位置 T=(C_X, C_Y, C_Z)と、カメラ座標系から見た世界座標系の回転を、X 軸周りの角度 α、Y 軸周りの角度 β、Z 軸周りの角度 γ で表し、カメラの焦点距離を f、映像座標系(x, y)でのイメージセンサ中心の位置を(u_0, v_0)、イメージセンサの x 方向と y 方向のスケールファクタ(1/画素サイズ)を(k_x, k_y)、イメージセンサの画素並びが平行四辺形的に歪んでいる程度を表すせん断係数を k_s とすると、次の斉次行列で表されることが知られている[58]。

$$\lambda\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} fk_x & fk_s & u_0 \\ 0 & fk_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}\begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}\begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (6.1)$$

$$T = \begin{bmatrix} C_X & C_Y & C_Z \end{bmatrix}^T$$

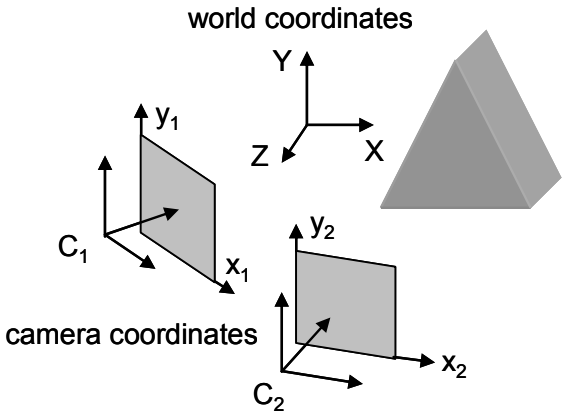


図 6- 2　世界座標とカメラ座標

ここで、λ は 3 次元空間を 2 次元空間に射影する為に生じる不定性を表す媒介変数であるが、その値はこの行列式の 3 行目から決める事が出来る。右辺第 1 項は、カメラの仕様で決まりカメラの内部パラメータと呼ばれる。第 2 項は、3 次元空間を 2 次元映像に射影する為の射影行列であり、第 3 項は、カメラから見た世界座標系の位置(T)と姿勢(R)で決まり、カメラの外部パラメータと呼ばれる。

この式は、任意方向を向いた精度の悪いカメラへの射影を正しく表現出来るが、精度

の良いカメラでは、k_s=0、u_0=v_0=0 と見なせる。以降では簡単の為、k_x=k_y=1 とし、図 5-1 に示す様に、各カメラのレンズ中心（カメラ座標の原点）は世界座標系の原点(0, 0, 0)から順に X 軸上に等間隔 B で並んでおり、カメラの光軸は全て Z 軸方向を向いているものとする。この場合、$T=[-nB, 0, 0]^T$（n=0, 1, 2,...：カメラ番号）、R=単位行列となる。実際のカメラ配置では若干の誤差を伴うが、Rectification 済みの映像では、上記関係を成立させる事が出来る。

以降の検討では、被写体面を小さな平面の集合で近似し、この平面を各カメラ映像へ射影する事により、各射影映像間に成立する関係を解析する。被写体が平面である場合の射影は、Affine 変換で記述出来る事が知られているが[50]、Affine 変換を行う為には、特徴点抽出とその対応点問題を解いて変換係数を求める必要があると共に、変換された平面は不等辺四角形になり、方形ブロックを符号化単位とする従来の圧縮方式[4]とは相性が悪い。そこで、以下では、カメラ映像上の方形ブロックに射影される被写体面を平面とみなし、その平面が他のカメラ映像上でどの様な形に射影されるかを解析して、ブロックマッチングに適する規則性を導く。

6.2 正対する被写体面の射影

(a) 射影方程式

今、図 6-3 に示す様に被写体が Z=d にある Z 軸に垂直な平面と仮定すると、この面上の点の座標は、(X, Y, d)で表され、この点の各カメラへの射影は 6.1 式より次式の様になる。この様な面は、被写体までの距離が十分大きく、被写体自身の奥行が小さい場合に相当する。なお以降の図では、簡単化の為に、視点を YZ 平面上に置き、斜め上から見下ろしたものを平行投影で描写した。

$$\lambda \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix} = \begin{bmatrix} f(X - nB) \\ fY \\ d \end{bmatrix} \tag{6.2}$$

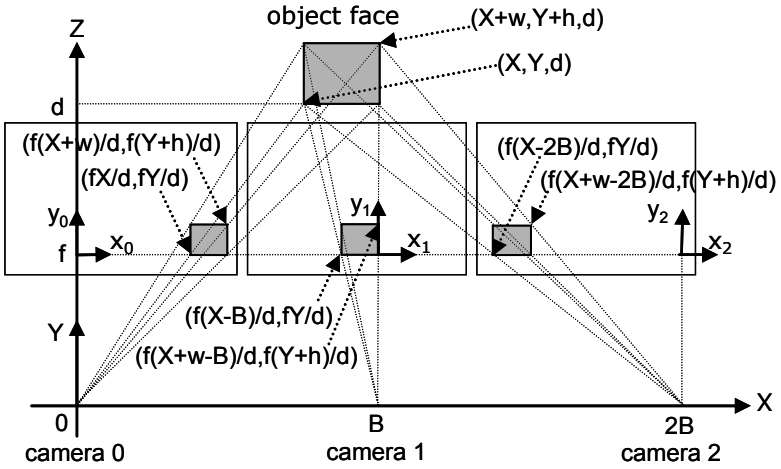


図 6- 3　カメラ光軸に正対する面の射影

(b)　ブロックの平行移動予測

6.2 式より、各カメラでの映像位置(x_n, y_n)は、

$$x_n = \frac{f(X - nB)}{d}, \quad y_n = \frac{fY}{d} \tag{6.3}$$

となり、x_n は Y 成分を含まないので垂直線は垂直に保たれ、y_n は X 成分を含まないので水平線は水平に保たれる。又、y_n はカメラによらず一定であるので、同一被写体を写す各カメラ画像上のブロックの高さは共通である。又、x_n はカメラ番号 n によって変化するので、同一被写体の映像の位置が、カメラによって変化するが、被写体の幅を w とすると、各カメラへの射影は fw/d と共通であるので、カメラ間では映像の形状は不変で、位置が x 軸方向に $\delta_n = x_n - x_0 = -fnB/d$ ずれるのみである事が分かる。従って、カメラ 0(n=0)の映像点(x_0, y_0)を参照ブロックのコーナー点とするカメラ映像間のブロックマッチングは、ブロックの形状を変える事無く、従来の平行移動探索を x 軸方向に行うのみで良い。しかも、参照カメラが左側か右側かで対応するブロックのずれの方向が一意的に決まる為、片側探索のみで良く、この様にして求まった δ_n を視差ベクトルと呼ぶ。今、カメラ 1 の映像の視差ベクトル δ_1 が求まったとすると、他のカメラ映像での対応点は、探索する事無く、カメラ番号 n の値を掛けるだけで求まり、高速に視差ベクトルが求まる。実際には前景に別の被写体があり、オクルージョンが発生している場合があるので、計算で求めた対応点の画素値が参照画素値と同じであるかを確認し、一致しない場合のみ、探索をやり直す必要がある。但し、被写体の照明に方向性はなく、画素値はカメラ間で変化しないと仮定する。この確認は以降の場合でも必要である。

6.3 縦回転した被写体面の射影

次に、被写体面がカメラの光軸に対して縦方向に回転した場合の射影について検討する。

(a) 射影方程式

図 6-4 に示す様に距離 Z=d にある被写体の面が X 軸周りにφ回転している場合、この面上の点の座標は(X, Y, d+Ytanφ)で表される。この様な面は、床を斜め上から見下ろした時や、階段を正面から見た場合等であるが、この点のカメラ映像(x_n, y_n)への射影は、

$$\lambda\begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix} = \begin{bmatrix} f(X-nB) \\ fY \\ (d+Y\tan\phi) \end{bmatrix} \tag{6.4}$$

となる。

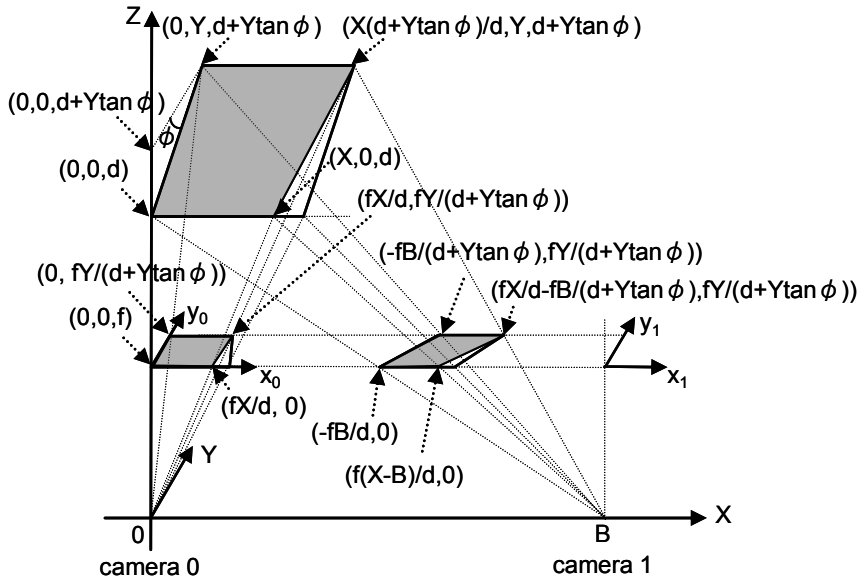


図 6- 4　水平軸周りに回転した面の射影

(b) ブロックの傾き補償予測

(6.4)式を解いて、カメラ映像上の対応点座標は次式となる。

$$x_n = \frac{f(X-nB)}{d+Y\tan\phi}, \quad y_n = \frac{fY}{d+Y\tan\phi} \tag{6.5}$$

先の場合と同様に、カメラ 0 の映像点(x_0, y_0)を参照画像点とすると、他のカメラの対応点に関しては、y_n がカメラ番号を含まないので、各カメラ映像間で対応点の高さは保たれ、y 軸方向の探索は不用である。 x 軸方向には、$\delta_n=x_n-x_0=-fnB/(d+Y\tan\phi)$だけ位置がずれるので、例えば、カメラ 1 の映像で対応点(x_1, y_1)が求まれば、その視差ベクトルの値が$\delta_1=-fB/(d+Y\tan\phi)$と得られるので、この値を n 倍した値だけ x_0 の位置からずらせた位置にカメラ n の対応点があり、その対応点の視差ベクトルは$\delta_n=n\delta_1$で与えられ、高速探索が可能になる。

次に、探索に使うブロックの形状については、6.5 式の x_n は、分母に Y 成分を含むので、被写体面上で X 座標が不変な直線でも射影映像の x 座標値は高さ Y に応じて変化するので斜めに傾き、その傾きをブロックの左上と左下の x 座標の差 s_n として表すと、

$$s_n = \frac{nfBY\tan\phi}{d(d+Y\tan\phi)} \tag{6.6}$$

と表せ、カメラ番号 n により変化する。ブロックの幅に関しては、カメラ 0 の画面上の小さな方形ブロック（横幅 $w=fX/d$、高さ $h=fY/(d+Y\tan\phi)$)から被写体の面を見ていると仮定すると、図 6-4 を参考にして、同じ被写体が射影されるカメラ n の映像上でブロックの横幅は、fX/d と高さやカメラによらず一定となるが事が分かる。すなはち、ブロックの形状は、平行四辺形となる。従って、方形のブロックマッチングでは、マッチング誤差は小さくならないが、ブロックの傾きを変えながら探索を行えば、一致するブロックが存在する。このブロックの傾き s_n は、カメラ番号により変化するが、隣り合うカメラ間での傾きの差は

$$\varepsilon = s_n - s_{n-1} = \frac{fBY\tan\phi}{d(d+Y\tan\phi)} \tag{6.7}$$

となり、一定であるので、例えば、カメラ 1 の映像でカメラ 0 の映像に対応するブロックの傾きの差 ε が探索で求まれば、カメラ n で対応するブロックの傾きは、探索を行う事無く、この値を n 倍して $s_n=s_0+n\varepsilon$ と高速に求める事が出来る。

6.4 横回転した被写体面の射影

次に、カメラの光軸に対して横回転した面の射影を検討する。

(a) 射影方程式

図 6-5 に示す様に被写体の面が Y 軸の周りに θ 回転している場合、この面が Y=0 の面上で Z=d の面と交わる交点の座標を(X, 0, d)とし、その位置から面に沿って水平に L

進んで次に垂直に H 進んだ面上の点は、$(X+L\cos\theta\ ,\ H,\ d+L\sin\theta\ )$ で表されるので、この点の各カメラへの射影は、次式で表される。この様な面は、ビルや部屋の壁を斜めから見た場合に相当する。

$$\lambda\begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix} = \begin{bmatrix} f(X - nB + L\cos\theta) \\ fH \\ d + L\sin\theta \end{bmatrix} \tag{6.8}$$

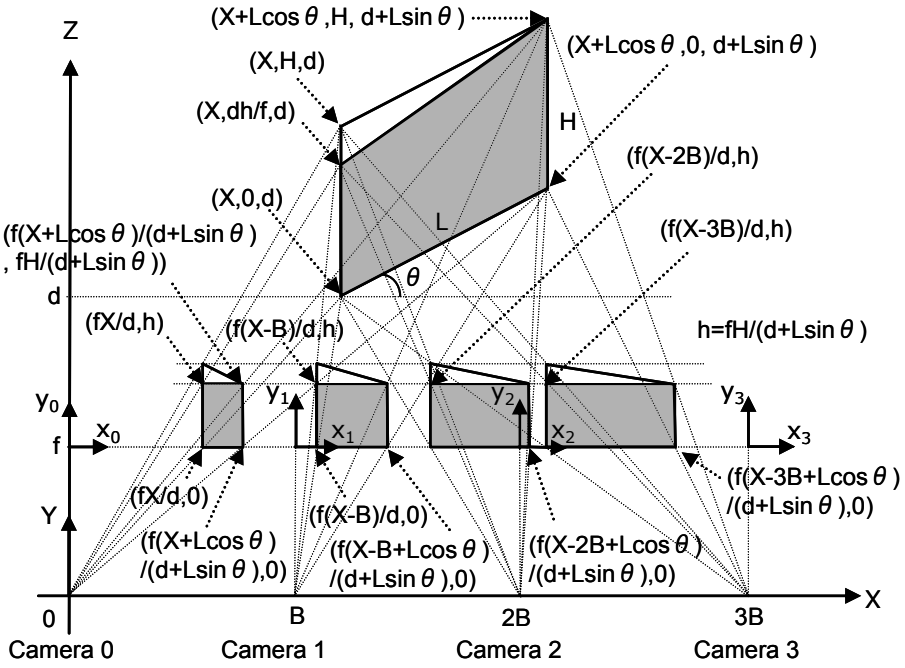


図 6- 5　垂直軸周りに回転した面の射影

(b)　ブロックの幅補償予測

(6.8)式より、各カメラ映像での対応点(x_n, y_n)の座標は、

$$x_n = \frac{f(X - nB + L\cos\theta)}{d + L\sin\theta}$$
$$y_n = \frac{fH}{d + L\sin\theta} \tag{6.9}$$

となり、y 座標値はカメラによらず一定であるので、探索ブロックの高さは保たれる事が分かる。対応点の x 座標値は、カメラ番号に応じて $-fnB/(d+L\sin\theta)$ だけずれる。従って、カメラ 0 の映像上の点 (x_0, y_0) を参照点にしてカメラ 1 の映像上での対応点が求まれば、対応点の視差は $\delta = x_1 - x_0 = -fB/(d+L\sin\theta)$ で与えられる。これを n 倍した値だけ参照

点位置からずらせた所に、カメラ n の画像上の対応点(x_n, y_n)があるので、それらは探索する事無く求められる。

次に探索ブロックの形状に関しては、被写体の面上に描かれた方形ブロックの射影は、図 6-5 に示した様に横に倒れた台形的に歪むが、逆にカメラ 0 の画面上の方形ブロック（幅： $w_0=f(X+L\cos\theta)/(d+L\sin\theta)-fX/d$、高さ： $h_0=fH/(d+L\sin\theta)$ とする）の他のカメラへの射影映像は、幅 : $w_n=f(X-nB+L\cos\theta)/(d+L\sin\theta)-f(X-nB)/d$、高さ : $h_n=h_0$ となる。w_n は Y 成分を含まないので、垂直線は垂直に射影され、ブロックの方形性は保たれる。高さ h_n は一定であるが、ブロック幅 w_n は、カメラ番号 n により変化する。しかし、隣り合うカメラ映像間での幅の差は、

$$\delta_w = w_n - w_{n-1} = \frac{fBL\sin\theta}{d(d+L\sin\theta)} \tag{6.10}$$

となり、一定であるので、たとえばカメラ 0 の映像上の参照ブロックに対応する方形ブロックがカメラ 1 の映像上で求まれば、そのブロックの幅は、参照ブロックの幅に(6.10)式の値を加えたものであり、他のカメラ映像上での対応ブロックの形状は、(6.10)式を n 倍して参照ブロックの幅に加えたものとなり、高速探索が可能になる。以上より、ブロックマッチングは、ブロックの位置と幅を変えながら探索を行えば、一致するブロックが存在することが分かる。幅の異なるブロック同士のマッチングは、ブロックを構成する画素を参照ブロックと同数になる様に再サンプリングして比較すれば良い。

6.5 縦横回転した被写体面の射影

本節では、被写体面がカメラの光軸に対して縦方向と横方向の両方向に回転した場合の射影を検討する。

(a) 射影方程式

一般に、図 6-7 に示す様に被写体の面が垂直軸の周りに θ 回転し、次に新たな水平軸の周りに ϕ 回転しており、この面が Z=d で Z 軸と交わっているとすると、この面上の点の座標は(X, Y, d+Xtan θ +Ytan ϕ /cos θ)と表される。よって、この点のカメラ n への射影は

$$\lambda\begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix} = \begin{bmatrix} f(X-nB) \\ fY \\ d+X\tan\theta+Y\tan\phi/\cos\theta \end{bmatrix} \tag{6.11}$$

で表される。

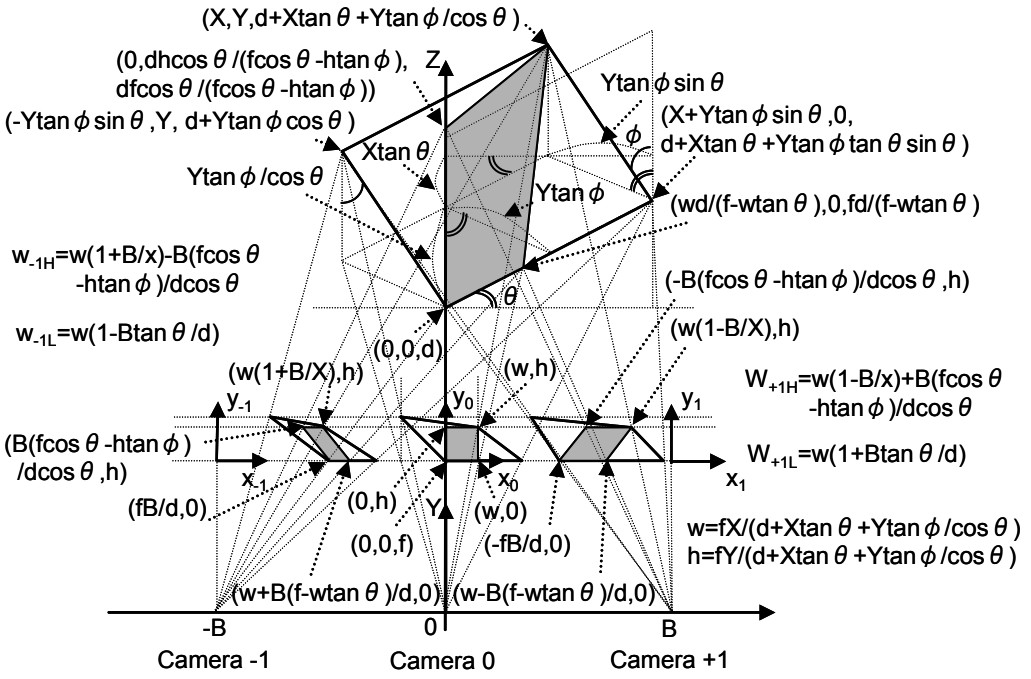


図 6- 6　水平・垂直軸周りに回転した面の射影

(b)　ブロックの幅傾き補償予測

(6.11)式より、カメラ n の映像点の座標は、

$$x_n = \frac{f(X - nB)}{d + X\tan\theta + Y\tan\phi/\cos\theta}$$
$$y_n = \frac{fY}{d + X\tan\theta + Y\tan\phi/\cos\theta} \qquad (6.12)$$

となる。x_n、y_n共に X、Y 成分を分母に含むので、方形な被写体面のカメラ映像は図 6-7 に示す様に不等辺四角形になる。今、カメラ 0 の映像面上の原点付近の方形ブロックの頂点を夫々a_0=(*0, 0*)、 b_0=(*w, 0*)、 c_0=(*w, h*)、 d_0=(*0, h*)とし、これを通して被写体を見た場合、他のカメラで同じ被写体が写るブロックの頂点の座標は次式となる。

$$a_n = (-fnB/d, 0)$$
$$b_n = (w - nB(f - w\tan\theta)/d, 0)$$
$$c_n = (w - nB(f - w\tan\theta - h\tan\phi/\cos\theta)/d, h) \qquad (6.13)$$
$$d_n = (-nB(f - h\tan\phi/\cos\theta)/d, h)$$

ここで、

$$w = fX/(d + X\tan\theta + Y\tan\phi/\cos\theta)$$
$$h = FY/(D + X\tan\theta + Y\tan\phi/\cos\theta) \quad (6.14)$$

である。これより、ブロックの高さ h はカメラによらず一定に保たれる事が分かる。又、隣り合うカメラ映像間で、各頂点の x 座標の差は、

$$\delta_a = -fB/d$$
$$\delta_b = -B(f - w\tan\theta)/d$$
$$\delta_c = -B(f - w\tan\theta - h\tan\phi/\cos\theta)/d \quad (6.15)$$
$$\delta_d = -B(f - h\tan\phi/\cos\theta)/d$$

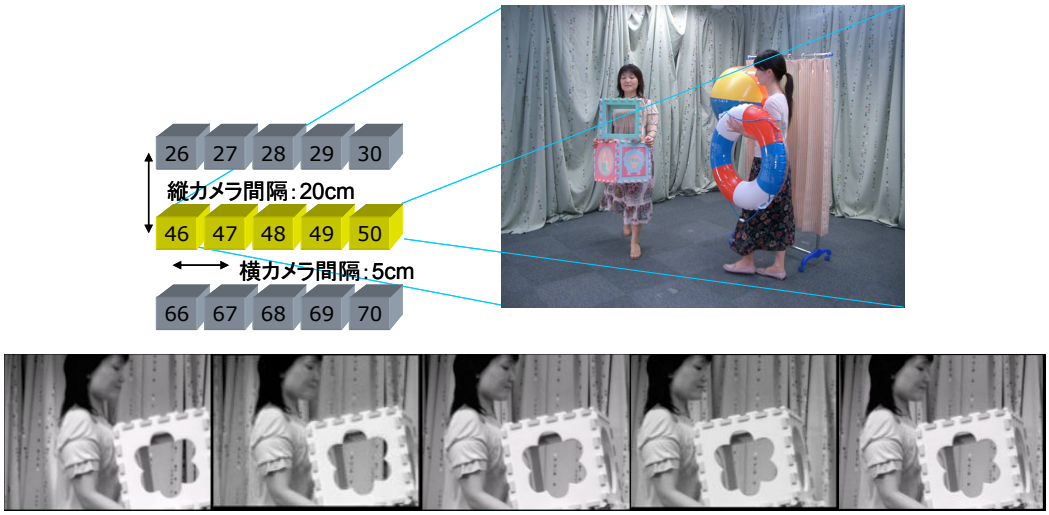
となる。撮影に使うカメラの例としては、焦点距離 f=12.5mm、 画像サイズ 6.47mm×4.84mm 程度であり、得られた映像を画素数 640x480pel にサブサンプリングし、ブロックサイズを $w\times h$=8×8pel とすれば、$h=w\fallingdotseq 81\,\mu$m$\fallingdotseq f/154$ である。θ<86° であれば、$w\tan\theta$ の項は<f/10 となりほぼ無視出来、$\delta_a=\delta_b=-fB/d$ すなはち、底辺は、カメラ間で平行移動の関係にあり、移動量は$-fB/d$ である。又、$\delta_c=\delta_d=-fB/d+Bh\tan\phi/\cos\theta$ であるので、上辺も平行移動の関係にある。移動量は、底辺に対し $\varepsilon=Bh\tan\phi/d\cos\theta$ ずれるので、カメラ n での対応ブロックは同じ幅の平行四辺形となる。傾きのズレ量は、カメラ 0 のブロックを参照ブロックとし、それに対するカメラ 1 のブロックの傾きの差 ε が得られれば、カメラ n での傾きの差は、 ε を n 倍すれば得られ、高速に対応ブロックの形状が得られる。 θ>86° であれば、幅の変化が無視出来なくなり、幅と傾きの両方が変わる。幅の変化量は $\delta_w=Bw\tan\theta/d$ となり、隣り合うカメラ間では一定であるので、先と同様に、カメラ 0 のブロックに対するカメラ 1 のブロックの幅の変化量 δ_w が求まれば、他のカメラでの幅の変化量は $n\delta_w$ と高速に求められる。又、θ<86° かつ ϕ<45° であれば、$h\tan\phi/\cos\theta$<f/10 であり、この項も無視出来るので、ブロック形状はカメラ間で不変な方形と見なせ、ブロックの平行移動で対応ブロックが見つかる。平行移動量は、隣り合うブロック間では $\delta=-Bf/d$ であるので、同様にカメラ 0 に対するカメラ 1 でのブロックの移動量 δ が得られれば、カメラ n での移動量は、$n\times\delta$ と高速に求められる。床を斜めに見下ろしたり、天井を斜めに見上げたりした場合など ϕ>45° では、上辺と底辺のズレ量の差すなはちブロックの傾きが無視出来なくなり、平行四辺形となるが、その傾き量の差は上記と同じ関係が成り立つので、同様にして高速にブロックの傾きが得られる。

以上より、ブロックマッチングにおいて幅探索と傾き探索を行う事により、マッチング精度が改善される事が予測される。尚、被写体面の回転の自由度としては更に、Z 軸周りの回転があるが、この回転は射影に影響を与えないので、考慮不用である。又、カメラ配置としては、平行カメラ以外の配置も考えられるが、任意カメラ配置にすると、カメラ映像間の相関と規則性が低下するが、その見返りとしてのメリットは殆どないと思われる。唯一、円弧状のカメラ配置は、スポーツやゲームに適する場合がある。この場合には、被写体までの Z 軸上での距離がカメラによって変化するので、対応ブロック

は幅と傾きのみならず、高さも変化すると共に、カメラ映像間に簡単な規則性は成立せず、全探索によるブロックの形状補償が必要と思われる。この様な場合の効率の良い符号化方法の検討は、将来課題とする。

6.6 視差補償予測の符号化実験

前節までに、被写体面の向きを考慮する事により、多視点映像間の予測精度が上がる事を示したが、以降の節では、実験により予測精度を確認する。実験に用いたのは、MPEG MVC 標準化検討用のテストシーケンスで、その中から、Akko & Kayo シーケンスのカメラ番号 46、 47、 48、 49、 50 の 5 カメラのフレーム番号 161 の画像から左上 1/4 を切り出したサブ画像(320x240pel)とした。各カメラの間隔は 5cm であり、カメラ映像は、テストパターンを用いて求めたカメラパラメータを用いて rectification がほどこされている。



camera 46　　camera 47　　camera 48　　camera 49　　camera 50

図 6- 7　視差補償予測実験に用いた多視点映像

この撮影システムのパラメータを表 6-1 に示す。これより、隣接カメラ映像間の視差量は、図 6-8 を参考にして、

$$disparity = B \times \frac{f}{D} = 50 \times \frac{12.5}{3000 \approx 5000} = 208 \approx 125\ \mu m \qquad (6.16)$$

となり、画素サイズでカウントすると、

$$\Delta = \frac{disparity}{pixel\ size} = \frac{208 \approx 125}{10} = 20.8 \approx 12.5\ pel \qquad (6.17)$$

となる。遠く離れたカメラ間の視差量は、この値を間に入るカメラ台数倍すれば得られる。ここでの符号化実験には、画像データを行列演算により効率良く処理出来るMATLAB のソフトウエアを用いた。

表 6- 1　撮影システム仕様

項目	値
被写体距離	3−5m
カメラ間隔	5cm
カメラ焦点距離	12.5mm
CCDサイズ	6.47x4.84mm
画素数	640x480pel
画素サイズ	10x10μm

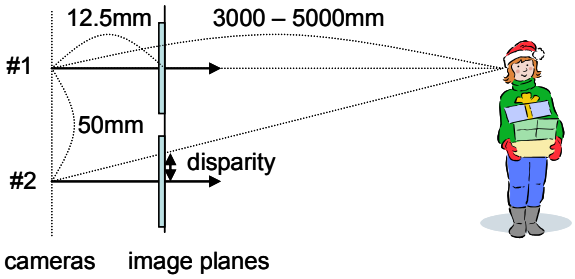


図 6- 8　隣接カメラ間の視差量

6.7 片方向予測と双方向予測

平行カメラの場合、夫々のカメラに写る被写体の範囲は、図 6-7 に示す様にカメラ間距離ずつずれて行くので、1 台のカメラ映像を予測の参照画像に使うと、対応点のない部分が生じ、この部分の予測が出来ない。この部分をむりやり参照画像内にあるオブジェクトで予測しても、完全一致はせず予測効率は上がらない。又、距離の異なる被写体どうしの境界ではオクルージョンが発生し、1 台の参照カメラでは補償しきれない。従って、参照カメラを 2 台用いた双方向予測や、多数の参照カメラからの予測が予測効率を上げるが、予測候補を増やすと予測の処理量が増える課題があるので、適切な参照画像数を設定する必要がある。以下では、ブロックサイズを 16x16pel として、片方向予測と双方向予測の結果を比較する。

(a) カメラ 46（左参照画像）　(b) カメラ 48（被予測画像）(c) カメラ 50（右参照画像）

図 6- 9　参照画像と予測される画像

(a)　片方向予測

図 6-9(a)の画像を参照画像として、図 6-9(b)の画像を予測した結果を図 6-10 に示す。探索は、予測される画像の各ブロックと、参照画像のブロック位置を 1 画素精度で変えながら、予測誤差が最小になるブロック位置を求め、その参照ブロックを予測された画像にはめ込んだ。探索範囲は、画像が rectification されているので、高さ位置は合っており、横方向探索のみを行った。又、カメラの相対位置が既知であるので、横方向探索は、片側探索のみとして予測時間の短縮を図った。探索範囲は、カメラ間隔が 10cm であるので、(6.17)式の値を 2 倍して前後に余裕を付け、Δx=16～56pel とした。ここで、予測された画像は、予測の効果を見る為に参照画像から切り出したブロックのみで構成している。

図 6- 10　ブロックサイズ 16x16pel での片方向予測結果（PSNR=10.66dB）

予測された画像の右側の黒い部分は、参照画像に対応したブロックが存在しなかった部分である。又、人物の右側にある顔画像の繰り返しは、オクルージョンの為、対応画素が無かった部分である。この時の予測画像のピーク SN 比は、10.66dB であり、この予測画像を作るために必要なデータ量は、各ブロックの並行移動量を表す視差ベクトルのみであり、1 ブロック当たりそのエントロピーは約 3.64bit であった。

ピーク SN は、予測された画素値 P_i と原画素値 O_i より、次式で計算した。ここで、n は全画素数である。

$$N^2 = \sum_{i=1}^{n} (P_i - O_i)^2$$
$$PSNR = 10 \times \log_{10}(\frac{n \times 255^2}{N^2}) \quad (6.18)$$

エントロピー計算は、画面内の全てのブロックの視差ベクトル値のヒストグラム H(v)を作り、次式から計算した。但し、N はブロックの総数である。

$$p_v = \frac{H(v)}{N}$$
$$entropy = -\sum_{v} p_v \log_2 P_v \quad (6.19)$$

(b)　双方向予測

次に、図 6-9(a)及び(c)を参照画像として、どちらか予測精度の高い方の参照ブロックのみで作った画像を図 6-11 に示す。双方向予測を行う事により、片方向予測では予測不能で黒く残った部分が全て予測可能になった。又、距離の異なる被写体の境界部分で、オクルージョンとなって予測出来なかった部分もほぼ正しく予測が行えた。その結果、双方向予測画像のピーク SN 比は 30.48dB となり、19.82dB の画質改善が行えた。この時に必要な予測データ量は、どちらの参照画像から予測したかを表す 1bit/block の情報が追加された他には、視差ベクトルの 1 ブロック当たりのエントロピーは、3.83bit とあまり増加はみられなかった。

図 6- 11　ブロックサイズ 16x16pel での双方向予測結果（PSNR=30.48dB）

被写体の境界部分では、ブロック内に距離の異なる被写体が入る為、オブジェクトの

境界部分でブロック歪が大きくなる傾向が観測されるが、双方向予測による画質改善効果は大きい。時間軸方向の予測では、被写体が動かない時は画面内に予測不可能な部分は生じないが、多視点映像の場合は、カメラの位置関係により、1つの参照画像では予測不能なエリアが必ず生じるので、双方向予測の効果は大きい。又、オブジェクトがランダムに動く場合も、時間軸予測では不規則なオクルージョンが発生し、予測効率が悪いが、カメラ間予測の場合には、画像間に時間のずれがなく、カメラ位置は固定であるので、オブジェクトは静止しているとみなせ、カメラ位置のみに依存する規則的なオクルージョンが生じるのみであるが、双方向予測によって反対方向の参照画像から予測すれば、このオクルージョン部は予測可能になるので、多視点映像の符号化では、双方向予測は非常に有効な予測手段であり、符号化に必須のツールと思われる。更に、オブジェクトの位置は、2 つのカメラ映像から求めた視差量から、他のカメラ映像でのオブジェクト位置を事前に推定可能であり、複数のカメラ映像を予測する場合は、効率の良い探索が可能になる。

6.8 ブロックサイズ適応予測

前節で述べたブロックマッチングで正確な予測が行える為には、ブロック内の被写体が単一の面で構成されている必要がある。ブロック内に距離の異なる被写体が複数写っていると、カメラが変わる事により、それらの被写体は別々のブロックに射影されるので、ブロックマッチングが成立しなくなる。又、自然界の被写体面は、床や壁など本来平面のものもあるが、人物や動植物など曲面も多い。曲面は、ブロックサイズを十分小さくすれば平面近似出来る。又、これにより、異なる距離にある被写体同士の境界がブロック内に入る可能性を下げる効果もあるので、ブロックサイズを被写体に応じて可変にする事により予測精度が上がると思われる。以下では、この効果を確認する。

(a) 適応ブロックサイズ変更

16x16pel のブロックサイズで双方向予測を行った後、予測誤差が 1 ブロック当たり 1500 以上のブロックを、更にブロックサイズ 8x8pel の 4 ブロックに細分割して、再予測を行った結果を図 6-12 に示す。図 6-11 に比べ、オブジェクトの境界部分のブロック歪が改善されている事が確認される。この予測画像のピーク SN 比は、33.05dB であり、16x16pel のブロックサイズ固定の場合に比べて 2.57dB の画質改善が得られた。この予測に必要なデータ量は、ブロックの細分割を行うか否かの情報 1bit と、各ブロックの視差ベクトル分であるが、計算を容易にする為、全ブロックを 8x8pel とみなしてエントロ

ピーを求めると約 4.63bit/block であった。

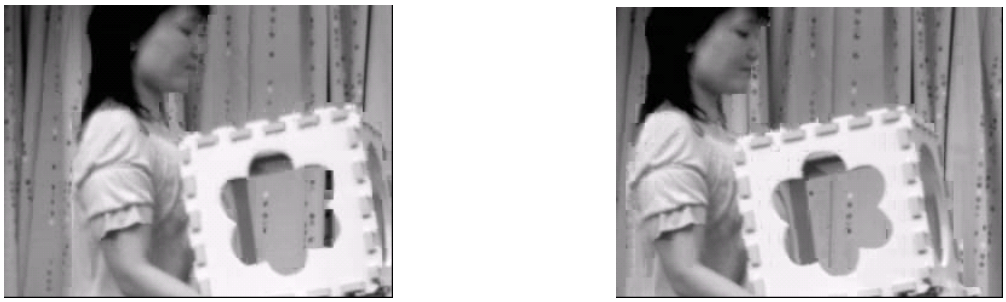
図 6- 12　16x16pel/8x8pel 可変ブロックサイズ予測結果（PSNR=33.05dB）

(b)　全ブロック細分化

更に、全ブロックサイズを 8x8pel として双方向予測を行った場合の予測結果を、図 6-13 に示す。図 6-12 で見られた細かなブロック歪が、更に改善されている。この予測画像のピーク SN 比は、33.77dB であり、先の可変ブロックサイズ予測の場合よりも更に 0.71dB 改善された。この場合に必要な予測データ量は、8x8pel のブロック当たりエントロピー換算で 4.60bit であり、先の可変ブロックサイズの場合よりも若干エントロピーが下がった。この理由は、可変ブロックサイズでのエントロピー計算が近似値である事の他に、各ブロックの予測精度が上がる事により、視差量の分布がより均一化した為と思われる。

図 6- 13　全ブロック 8x8pel での双方向予測結果 #48（PSNR=33.77dB）

この結果より、以降では、全ブロックサイズを 8x8pel として実験を行った。図 6-14 に同様にして、カメラ番号 47 及び 49 の映像を、カメラ番号 46 及び 50 の映像から予測した結果を示す。夫々画質は、32.83dB（4.59bit/block)、33.25dB（4.33bit/block）であった。若干予測精度が下がった原因は、片方の参照画像は近くなるが、他方の参照画像が遠くなるので、被写体面の傾きの影響が大きくなり、その補償をしないブロックの平行移動のみの予測では、予測精度が上がらなかった為と思われる。



(a) #47(32.83dB, 4.59bit/block)　　(b) #49(33.25dB, 4.33bit/block)

図 6- 14　8x8pel 双方向予測結果

6.9 ブロック形状適応

次に、被写体面の傾きに対応したブロック形状補償を行った場合の画質改善効果を確認する。

(a)　ブロック幅適応

ブロックの幅適応は、8x8pel で双方向予測を行った後、得られた参照ブロックの中心位置をピボットとして、参照ブロックの幅を変えて予測を行った。探索したブロック幅の範囲は、(4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 9, 10, 11, 12, 13, 14, 15pel)の 15 候補とした。探索範囲を広げる程、予測の可能性は高くなるが、参照ブロックの幅が小さくなり過ぎると、予測ブロックのテキスチャの横方向解像度が下がり、予測精度が上がらなくなるので、最小ブロック幅は 4.5pel（0.55 倍）とした。探索範囲をほぼ対称形にする為、最大ブロック幅は、15pel（1.9 倍)とし、探索時間の高速化の為に以下の 2 項木探索を行った。

(i)　先ず、初期ブロック幅を w_1=8 として、予測誤差 δ_1 を求める。

(ii)　次に、初期探索範囲を Δ_1=2 として、ブロック幅(w_1- Δ_1)pel 及び、(w_1+2 Δ_1)pel

での予測誤差$\delta_{1-\Delta}$及び、$\delta_{1+\Delta}$を求める。

(iii) 次に、($\delta_{1-\Delta}$, δ_1, $\delta_{1+\Delta}$)の中の最小誤差をδ_2=min($\delta_{1-\Delta}$, δ_1, $\delta_{1+\Delta}$)とし、その時のブロック幅を新しいブロック幅 w_2 とし、探索範囲を半分（$\Delta_2=\Delta_1/2=1$）にして、(ii)と同様の探索を行う。

(iv) (ii)～(iii)の探索を再度繰り返し、得られた、($\delta_{3-\Delta}$, δ_3, $\delta_{3+\Delta}$)の中で最小誤差を与えるブロック幅 w_3 を最適ブロック幅とする。

この探索は、3 回の繰り返しで終了するので、高速なブロック幅探索が行える。

予測される側のブロック幅は 8pel であるので、予測候補のブロック幅を 8pel に変換する必要があるが、図 6-15 に示す様に近傍の 2 画素の線形補間で 8pel を作成した。各係数の重みは、新たに作成される画素位置と元の画素位置の距離の比を簡単な整数で近似する事により、計算負荷を軽減した。

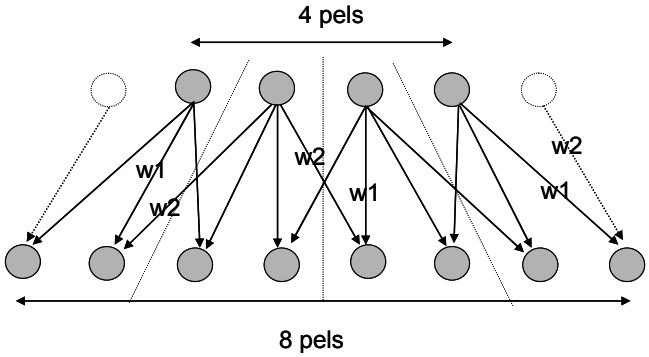
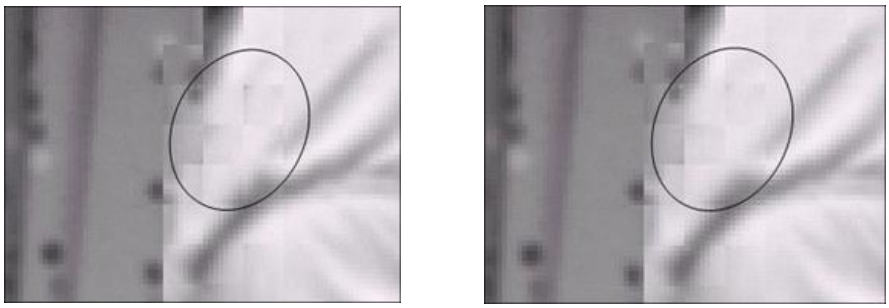


図 6- 15　4pel の参照ブロックから 8pel の予測ブロックを作る例

以上により、ブロックの幅補償を行った予測結果を図 6-16 に示す。図より、人物等の前景のエッジでの歪が軽減している事が確認出来る。この予測画像のピーク SN 比は、幅探索を行わない場合に比べて夫々、カメラ#47 の画像では 0.91dB、カメラ#48 の画像では 0.55dB、カメラ#49 の画像では 0.75dB の改善が見られた。#47 と#49 の画像で改善効果が大きいのは、これらの予測画像では参照カメラまでの距離が大きくなり、被写体面の傾きの影響が大きく出た為、幅補償の効果も大きくなったものと思われる。この予測に必要なデータの追加分は、ブロックの幅情報がブロック当たり、#47 で 2.91bit、#48 で 2.75bit、#49 で 2.85bit であった。被写体が、ビル街や長い通路を正面から見た様な構図の場合には、画質の改善効果はもっと高くなると思われる。

(a) #47(33.74dB,+2.91b/B)　(b) #48(34.33dB,+2.75b/B)　(c) #49(34.00dB, 2.85b/B)



(d) #48 幅補償無し拡大図　　(e) #48 幅補償有り拡大図

図 6- 16　8x8ple ブロックの幅補償を行った予測結果

(b)　ブロック傾き適応

ブロックの傾き補償は、8x8pel のブロックサイズで双方向予測を行った後、得られた視差位置でのブロックの中心をピボットにして、ブロックの傾きを変えて探索を行った。探索した傾きの範囲は、(-2, -1.5, -1, -0.5, 0, 0.5, 1, 1.5, 2pel/block height)の 9 候補とした。ブロックの傾きが大きく変わるのは、床や天井の写った映像であるが、この実験映像にはそれらが無い為、探索範囲は少し狭く設定した。幅探索と同様に、2 項木探索を行ったので、2 回の繰返し探索で収束する。傾いた画素は、図 6-17 に示す様に、両隣の画素に距離の比を簡単な整数で表した重みを加えて加算平均して求めた。

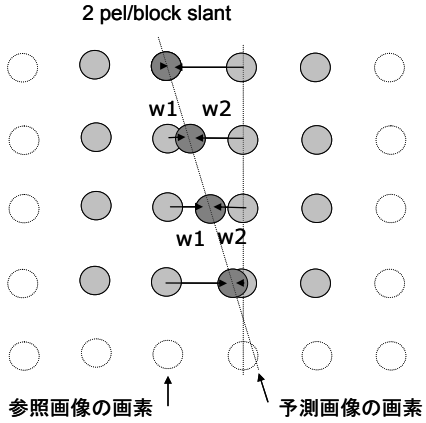
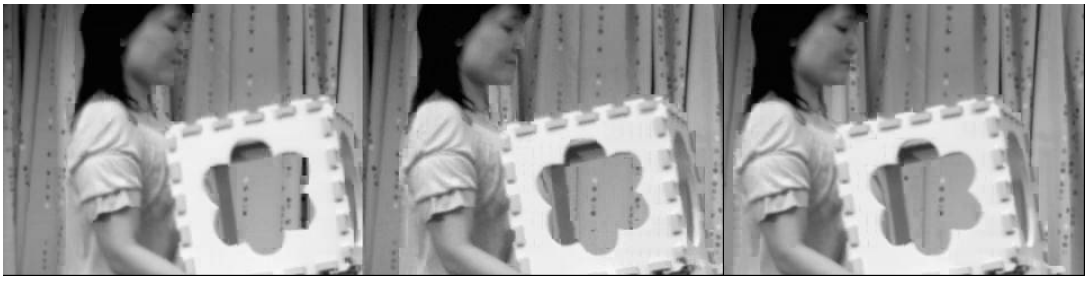
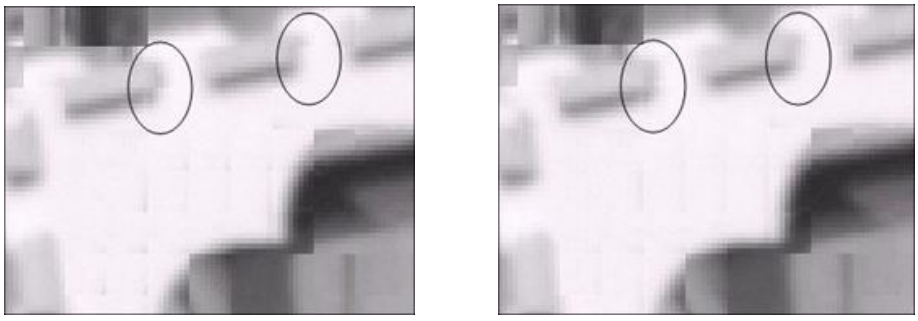


図 6- 17　2pel/block の傾きの画素を作る例

以上の様にして、ブロックの傾き補償を行った予測結果を図 6-18 に示す。図 6-16 に比べて、箱などの底面が見えている部分のブロック歪が軽減している。予測画像のピーク SN 比は、夫々、カメラ#47 の画像で 33.18dB、、カメラ#48 の画像で 34.06dB、カメラ#49 の画像で 33.65dB であり、傾き補償を行わない場合に比べて夫々、#47 で 0.35dB、#48 で 0.29dB、#49 で 0.40dB の改善が見られた。先の場合と同様に、参照カメラまでの距離が大きい場合に、予測精度の改善が大きい。この時に必要な予測データの追加分は、ブロックの傾き情報がブロック当たり、#47 で 2.89bit、#48 で 2.67bit、#49 で 2.85bit であった。映像の中に床や天井等の広い平面が写っている場合は、更に高い画質改善効果が期待される。



(a) #47(33.18dB,+2.89b/B)　(b) #48(34.06dB,+2.67b/B)　(c) #49(33.65dB,+2.85b/B)



(d) #48 傾き補償無し拡大図　　(e) #48 傾き補償有り拡大図

図 6- 18　8x8pel ブロックの傾き補償を行った予測結果

(c)　　ブロック形状補償の他の例とまとめ

図 6-19 に、他のテスト画像でのブロック形状補償の効果を示す。使用した画像は、MPEG MVC のテスト画像 Exit のカメラ番号 5, 6, 7 の第 1 フレームの中央から切り出した 240×320 画素の画像である。図 6-20 に、16×16 画素のブロックで片方向予測した画像と、双方向予測した画像、及び 8×8 画素で双方向予測した結果を、又図 6-21 に、更にブロックの幅補償と傾き補償を行った結果を示すが、夫々、Akko&Kayo のテスト

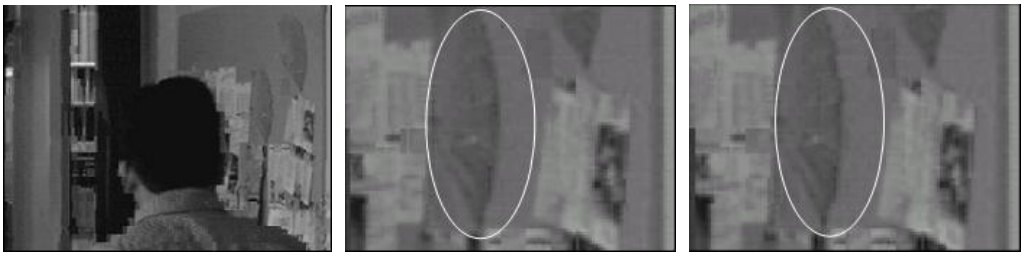
画像と同程度の画質改善効果が確認された。両者の画質改善量がほぼ同じになったのは、画面の構成がどちらも縦の平面とみなせる部分が多く、水平面や斜めに傾いた平面が少ない傾向が一致した為と思われる。

(a) 左参照画像(#5)　(b) 予測される画像(#6)　(c) 右参照画像(#7)

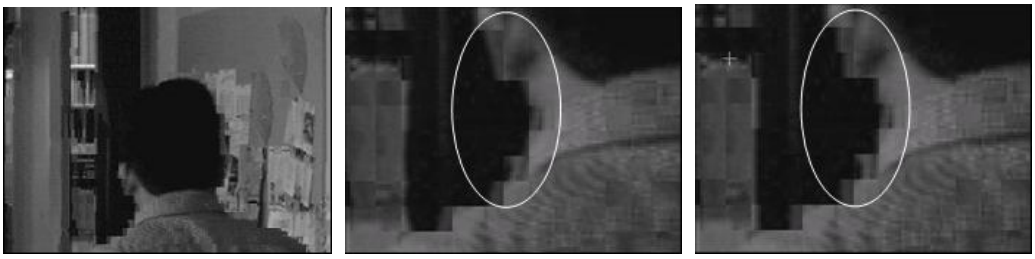
図 6- 19　MPEG MVC テスト画像（Exit）の部分画像

(a) 16x16pel 片方向予測 12.23dB(4.06bit/block)　(b) 16x16pel 双方向予測 26.62dB(5.71bit/block)　(c) 8x8pel 双方向予測 29.03dB(6.45bit/block)

図 6- 20　Exit 画像での片方向予測と双方予測結果



(a) 8x8pel 幅補償付き　(b) 補償有り拡大図 29.56dB(+3.24bit/block)　(c) 補償無し拡大図



(a) 8x8pel 傾き補償付き　　(b) 補償有り拡大図　　(c) 補償無し拡大図

29.32dB(+3.07bit/block)

図 6- 21　Exit 画像の幅補償と傾き補償付き双方向予測結果

以上の結果を表 6-2 にまとめて示すが、被写体の面の傾きに適応させてブロックの幅補償や傾き補償を行うと、夫々、約 0.5dB、0.3dB の画質改善効果が得られた。このブロックの形状補償を行う為には、幅情報や傾き情報のデータをデコーダに送る必要があるが、これらのデータの持つ時空間相関を利用する事により、そのデータ量を効率良く削減する事が可能であると思われる。

表 6- 2　被写体面の傾き適応視差補償予測の画質改善効果

prd. ref.		block	adjust		PSNR(dB)		Gain(dB)		extra bits/block	
L	R	size	W	S	Akko	Exit	Akko	Exit	Akko	Exit
√		16x16			10.66	12.23			3.64	4.06
√	√	16x16			30.48	26.62	19.82	14.39	3.83	5.71
√	√	8x8			33.77	29.03	3.29	2.41	4.6	6.45
√	√	8x8	√		34.33	29.56	0.56	0.53	2.75	3.24
√	√	8x8		√	34.06	29.32	0.29	0.29	2.67	3.07

6.10　視差情報の符号化

前節では、視差補償予測にブロック形状の適応化処理を行うと予測精度が向上する事を確認したが、本節では、視差補償予測に必要な視差ベクトルやブロックの幅・傾き情報の効率の良い符号化方法を検討する。ここでの検討に当たっては、現在 MPEG MVC 標準化で検討に使用されている符号化モデル JMVM1.0 を、視差ベクトルや予測誤差の圧縮に利用した。JMVM1.0 は、MPEG-4 AVC 規格準拠の圧縮コアに階層符号化機能を持たせたもので、C++のソースコードを Windows XP マシン（CPU 1GHz、メインメモリ 256MB） 上の Microsoft Visual C++.NET でビルドし、実行モジュールを DOS コマ

ンドで動作させた。評価に用いたテスト画像は、図 6-7 に示した Akko&Kayo シーケンスの内、最下段カメラ 66, 67, 68, 69, 70 の第 28 フレーム目の映像であり、画像サイズは、640×480 画素の原画サイズである。

camera 66　　camera 67　　camera 68　　camera 69　　camera 70

図 6- 22　視差情報予測実験に用いたテスト画像

(a)　ブロック形状情報の符号化

先ず、ブロックの形状情報であるブロック幅とブロック傾き情報の符号化を検討した。予測に使う参照ブロックの幅補正や傾き補正を行うと、予測精度が改善される事は 6.9 節で確認したが、その為にはブロックの形状情報をデコーダに送る必要がある。ここでは、これらの情報を効率良く送る方法を検討し、その符号化効率をレート歪み特性によって確認した。検討に使用したアンカーコーデックの構成を下図に示す。

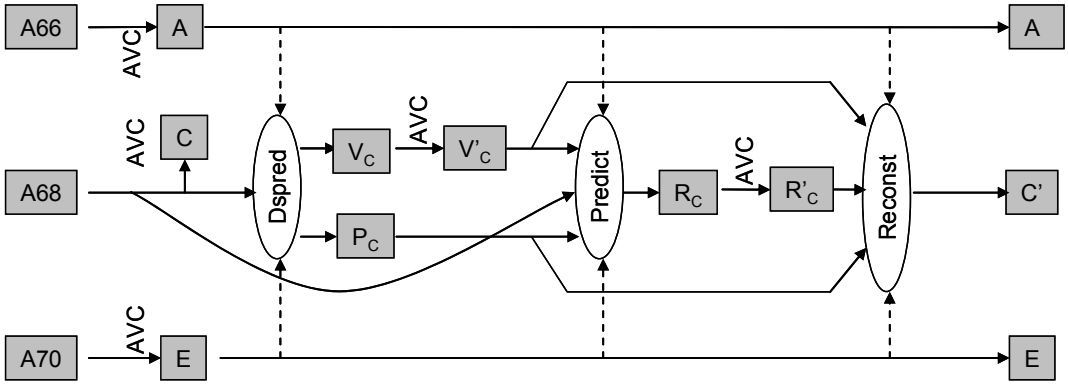


図 6- 23　双方向視差補償予測符号化の構造

先ず、各カメラ画像 A66, A68, A70 を夫々独立に JMVM でフレーム内符号化（AVC）し復号化したものをアンカー画像（A, C, E）とし、これに対して、両端のカメラ 66, 70 画像の再生画像 A, E から中央のカメラ画像 A68 を 8×8 画素のブロックで双方向視差補償予測（Dspred）し、得られた視差ベクトル V_c と予測誤差画像 R_c を夫々JMVM でフレーム内符号化（AVC）してビットストリームを得、これを復号化したもの(V'_c, R'_c)及び、参照画像ポインタ P_c を用いて予測画像の再生を行い（Reconst）、得られた再生画像 C' の PSNR とその時のトータルビット量（$V_c+R_c+P_c$）をレート歪カーブとしてプロットし

た。ここで、A, E どちらの参照画像を視差補償予測に用いたかのポインタ P_c が 1bit/ブロック発生するが、JMVM は 2 値データの圧縮効率が悪い為、非圧縮のままとしたが、ビット量は 144Kbps と僅かであり、全体の符号化効率に大きな影響は無かった。又、参照画像に再生画像を使ったが、デコーダ側で視差補償予測の復号を行う際に使用する参照画像が、量子化誤差を含んだ再生画像である為、エンコーダ側でも同じ再生画像を参照画像として使う事により、再生された参照画像に含まれる量子化誤差を解消出来、予測された画像の再生画質を参照画像の再生画質に無関係に自由に調節出来る様にする為である。同様に、視差ベクトルも非可逆圧縮され、量子化誤差が重畳するので、先ず、視差ベクトル V_c のみを求め（Dspred)、これを JMVM で符号化・復号化して得られる再生ベクトル V'_c を用いて、再度双方向視差補償予測を行い（Predict)、その時の予測誤差画像 R_c を JMVM で符号化してデコーダに送る構成にした。この様な構成にする事により、参照画像や、視差ベクトル、予測誤差画像のビットレートを夫々自由に調節出来、任意の画質やビットレートにおいて最適なレート歪み条件が得られる様になる。

又、視差ベクトルは 60×80 ブロック／フレームのバイトデータであるが、その圧縮に JMVM を利用する為には、輝度と色差成分から成る画像データと同じ構成にしてやる必要がある。又、JMVM のソフトウエアは画像サイズが 16 画素の整数倍しか受け付けない為、視差ベクトル画像は、底辺にその平均値 4 ラインを追加し 64×80 画素の輝度画像とし、色差成分には 0 を詰めた。その結果、視差ベクトルのビットストリームは、本来のエントロピーより若干多目のビット量となったが、480×640 画素の予測誤差画像のビット量が圧倒的に多く、符号化効率への影響は殆どなかった。

視差ベクトルの探索範囲は、テスト画像が rectification されており垂直視差は生じないので、水平方向探索のみ 1 画素精度の探索を行った。0.5 画素精度の探索も試みたが、データ量が 1bit/block 増加する割には、オクルージョン部の探索精度は上がらず、全体として有意な符号化ゲインは見られなかった。探索範囲は、先と同じく、+16～+56 画素の全探索とすると共に、カメラ 66 画像の探索範囲は右側のみ、カメラ 70 画像の探索範囲は左側のみとして、探索の高速化を図った。参照カメラまでの距離を等間隔に取る事により、オクルージョンがない場合、左右どちらの画像を参照しても視差ベクトルの絶対値は同じになるので、両者の平均値で視差ベクトルを決定する事により、視差ベクトル精度を高められると思われるが、実際には左右カメラの明るさや色のばらつきがあり、又オクルージョンが発生する部分は、片方の参照画像からしか正しい視差ベクトルが得られないので、ここでは、左右の参照画像の内、視差補償誤差の小さい方の画像のみを参照画像とした。

又、視差補償誤差を輝度値のみで評価すると、正しい視差量でなくても輝度の平均値さえ合えば良く、視差ベクトルに誤検出が多くなり、再生映像に違和感が出るので、輝度値と色差成分夫々の絶対値誤差の和に、既に探索の終わった部分であるブロックの左側から上部の 3 ブロックの視差ベクトルとの差分の絶対値に重みを掛けて加えたものを

評価関数とした。

$$Err = \sum_{block} | p(x) - q(x + dv) | + w \sum_{i=1}^{3} | dv_i - dv | \qquad (6.20)$$

ここで、Err は評価関数、p(x)は予測される画素値、q(x+dv)は参照画素値、dv は視差量、w は重み係数、dv_i は視差ベクトルを求めるブロックの右隣から上隣の 3 ブロックの視差量である。

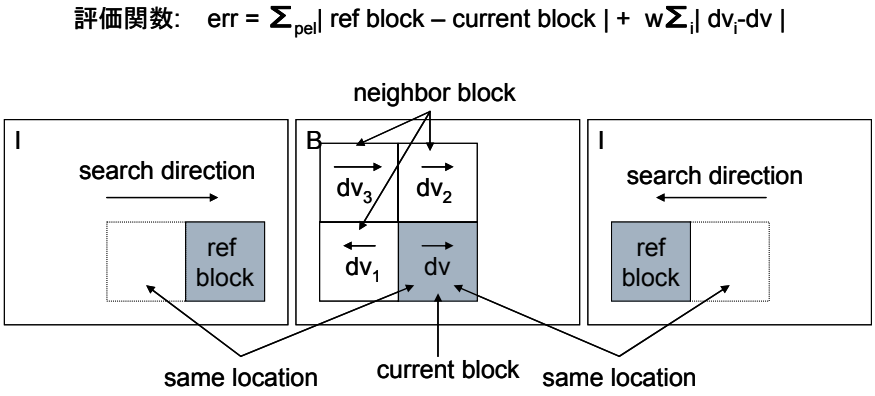
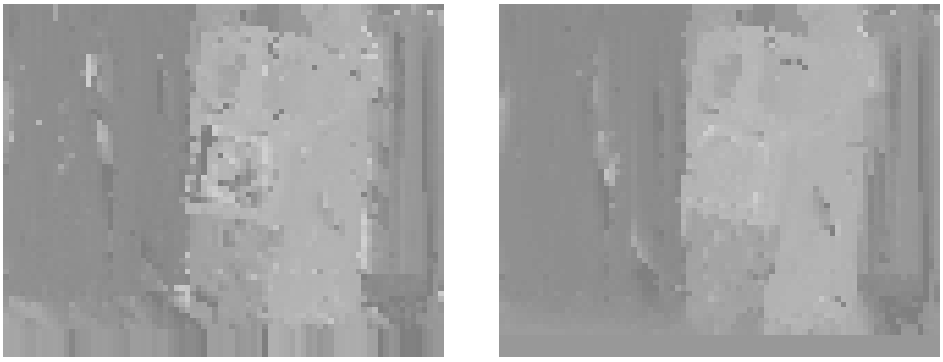


図 6- 24　視差ベクトルの探索

図 6-25 に輝度のみで探索を行った場合の視差ベクトルと、色差成分の差分を加えかつ、近隣ブロックの視差ベクトルとの差分も用いて求めた視差ベクトルを示すが、輝度のみの場合に対して、正しい視差ベクトルが得られている事が分かる。被写体のエッジ部分の形状が荒れているのは、ブロック内に異なる視差量を持つ被写体が含まれる為、その占める割合で視差量が決まった為である。視差量を画素単位で求めれば正しい外形が得られるが、視差ベクトルのデータ量が膨大になり符号化効率の観点からのメリットは少ない。



(a) 輝度値のみ　　(b)輝度・色差＋近隣ベクトル重み

図 6- 25　評価関数の違いによる視差ベクトル探索結果

このアンカーコーデックのレート歪み特性を、図 6-26 及び表 6-3 に示す。中央のカメラ画像を単独に符号化した場合（C）に対して、双方向視差補償予測を行う事により（C'）、4Mbps 付近では約 1dB 程度画質が高くなり、同じ画質なら約 20%程度ビットレートが節約出来ている事が確認され、6.7 節で示した双方向視差補償予測による画質改善効果は、双方向予測によるデータ量の増加を考慮したレート歪み特性の面からも有効である事が分かる。ビットレートの設定は、参照画像の量子化パラメータ QP を 22 から 30 まで変える事により調整し、夫々のビットレートで予測画像の画質が参照画像の画質とほぼ同じになる様に、予測誤差画像の量子化パラメータ Q は参照画像の量子化パラメータと同じ値を用い、視差ベクトルの量子化パラメータ V は、レート歪み特性が最大になる様に選択した結果、全てのビットレートで V=4 がほぼ最適値であったが、その値を倍又は半分にしても予測誤差画像のビットレートで吸収されて、総合特性に殆ど違いは出ずかなり柔軟性を持つ。これは、再生ベクトルで予測誤差画像を求める構造にした効果である。

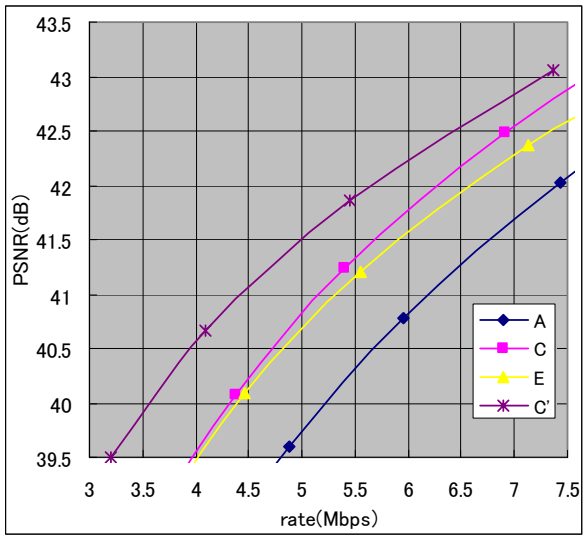


図 6- 26　双方向視差補償予測符号化のレート歪み特性

表 6- 3　双方向視差補償予測符号化のビットレートと PSNR

(a) 個別符号化結果

Quant. Param.	A(#66)		C(#68)		E(#70)	
	rate (Mbps)	PSNR (dB)	rate (Mbps)	PSNR (dB)	rate (Mbps)	PSNR (dB)
Q22	9.222	43.24	8.826	43.70	9.177	43.50
Q24	7.428	42.03	6.914	42.49	7.132	42.37
Q26	5.963	40.78	5.408	41.24	5.560	41.21
Q28	4.878	39.60	4.382	40.08	4.463	40.09
Q30	4.028	38.39	3.576	38.89	3.672	38.96

(b)双方向視差補償予測結果

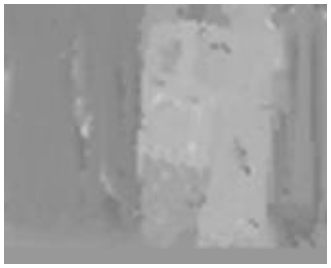
Quant. parameter	Vc		Pc	Rc		C'	
	rate (Mbps)	PSNR (dB)	rate (Mbps)	rate (Mbps)	PSNR (dB)	rate (Mbps)	PSNR (dB)
Q22V4	0.467	60.01	0.144	6.759	43.03	7.370	43.05
Q24V4	0.476	60.28	0.144	4.831	41.85	5.451	41.86
Q26V4	0.472	59.82	0.144	3.471	40.65	4.087	40.67
Q28V4	0.459	60.05	0.144	2.604	39.49	3.207	39.50

図 6-27 に、この符号化構造で得られた参照画像 A, C, E、視差ベクトル V_c, V'_c、予測誤差画像 R_c, R'_c、参照画像へのポインタ P_c、及び視差補償予測の再生画像 C'を示す。JMVM の圧縮をフレーム内符号化とした為、ビットレートが比較的高めであり、参照画像 C、再生予測画像 C'共に歪みやノイズは認められない。視差ベクトル V'_c は全体のビットレートの約 6%（467Kbps）で高精度（60.01dB）に圧縮出来た。参照ポインタ P_c は非圧縮であるが、全レートの約 2%（144Kbps）に収まった。ポインタの値が左側参照画像（図の白部分）に偏っているのは、左側カメラ（#66）の感度特性が予測画像のカメラ特性（#68）により近かった為と思われるが、左側カメラ画像からは予測出来ない画像の右端部分や、前景の被写体の右側部分は正しく右側カメラ画像が選ばれている（図の黒部分)。一部分ではオクルージョンよりはカメラ感度の違いの方が大きくなり、逆側の参照画像が選択された。画面下端（黒）は、平均値としてゼロを詰めた結果である。予測誤差画像 R_c は殆ど 0（レベル 128 のグレイ色）に近いが、画素数が多い為（480×640)、殆どのビットレート（約 92%）を占める。

(a)左参照画像 A(QP=22)
43.24dB(9.22Mbps)

(b)中央画像 C(QP=22)
43.70dB(8.83Mbps)

(c)右参照画像 E(QP=22)
43.50dB(9.18Mbps)



(d)視差ベクトル

(e)圧縮後(QP=4)
60.01dB(467Kbps)

(f)参照画像ポインタ
非圧縮(144Kbps)

(g)予測誤差画像 Rc　　(h)圧縮後 R'c(QP=22)　　(i)再生画像 C'

43.03dB(6.76Mbps)　　43.05dB(7.37Mbps)

図 6- 27　双方向視差補償予測の参照画像、視差ベクトル、ポインタ、予測誤差画像、及び再生画像

次に、このコーデックにブロック形状補償を追加した結果を述べる。図 6-28 は、ブロック形状補償を追加したコーデックの構成である。視差ベクトル V_c と参照画像ポインタ P_c を Dspred で求める際に、ブロックの幅探索と傾き探索を行い、得られた幅係数 K_w と傾き係数 K_s を、夫々JSVM で圧縮符号化・復号化し（AVC)、得られた再生係数 V'_c, K'_w, K'_s 及びポインタ P_c を用いて予測誤差画像 R_c を求め（Predict)、これを JMVM で圧縮して（AVC）デコーダに送る。デコーダでは、圧縮された参照画像、視差ベクトル、ポインタ、幅係数、傾き係数、予測誤差画像を再生して予測再生画像 C'を得る（Reconst)。予測誤差画像 R_c を求める時、圧縮・再生された予測情報を使うのは、先に述べた様に、再生画像 C'の画質を参照画像や予測情報の品質に対して独立に調整出来る様にする為である。

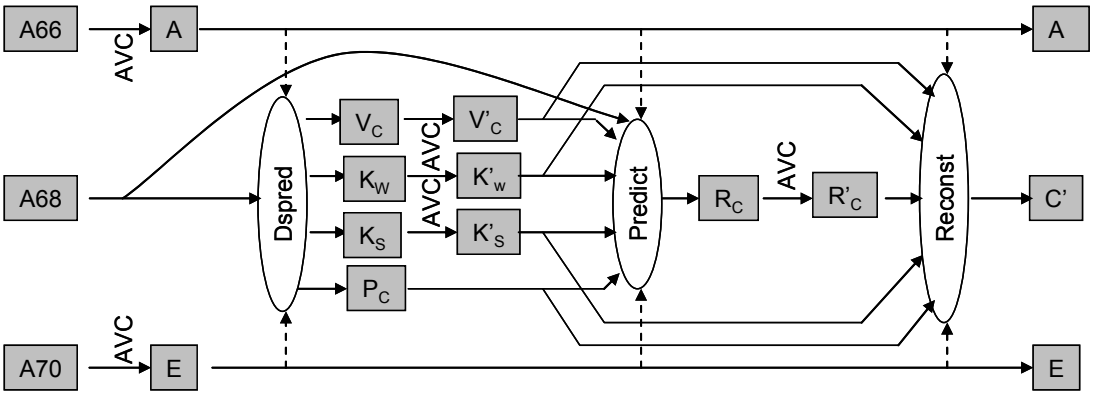
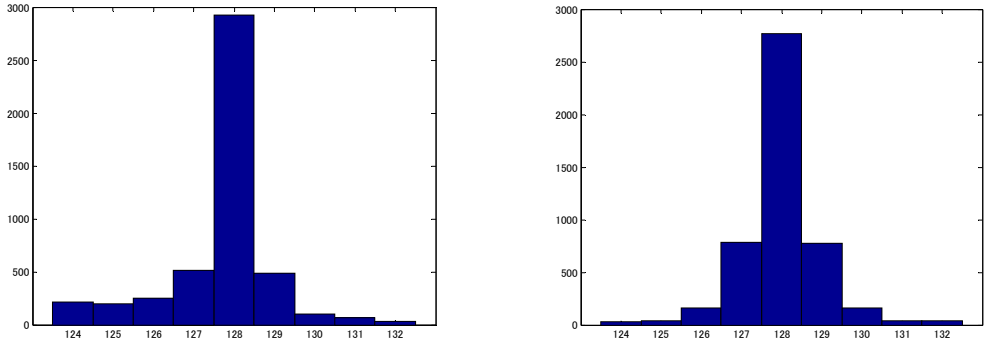


図 6- 28　ブロック形状補償付き双方向視差補償予測符号化の構成

ブロック幅の探索範囲は、(6, 6.5, 7, 7.5, 8, 8.5, 9, 9.5, 10pel）の 9 候補とし、幅係数は、K_w=(-4, -3, -2, -1, 0, 1, 2, 3, 4)で代表した。傾きの探索範囲は、(-2, -1.5, -1, -0.5, 0, 0.5, 1, 1.5, 2pel/block）の 9 候補とし、傾き係数は同様に K_s=(-4, -3, -2, -1, 0, 1, 2, 3, 4）で

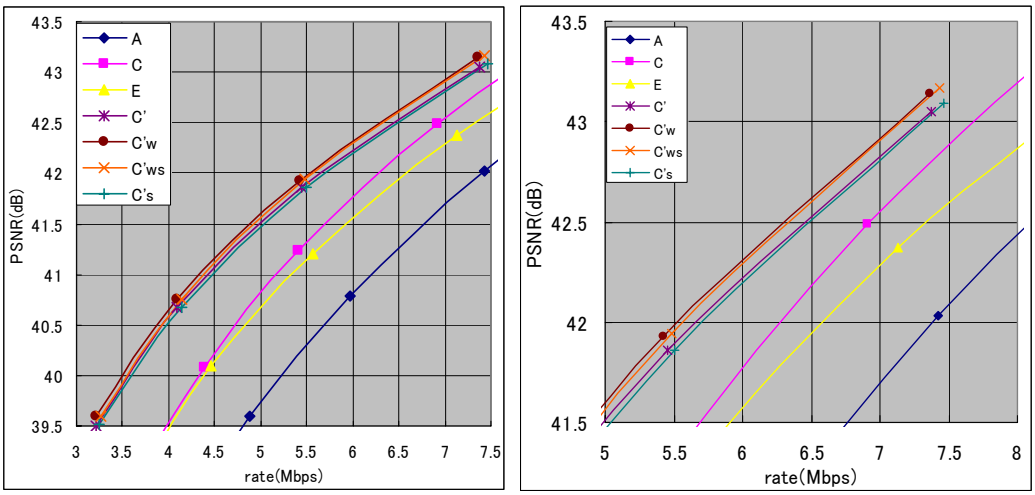
代表した。探索範囲を±両側で対称にしたのは、図 6-29 に示す様に、幅係数、傾き係数共にほぼ対象に分布する傾向があった為である。



(a) 幅係数分布（K_w=0：128）　　(b) 傾き係数分布（K_s=0：128）

図 6- 29　視差補償ブロックの幅係数と傾き係数の分布（カメラ 68 画像）

図 6-30 及び表 6-4 に、ブロック形状補償付き視差補償予測符号化のレート歪み特性を示す。ブロックの幅補償を行う事により（C'$_w$）、レート歪み特性が約 0.1dB 改善されている。ビットレートの増加を考慮しなければ、幅補償により画質は約 0.6dB 改善されたが、その為に、幅情報として約 80Kbps のビットレート増加が必要となり、それを含めた結果、総合のレート歪み特性では、約 0.1dB の改善になった。



(a) 全体図　　(b) 拡大図

図 6- 30　ブロック形状補償付き双方向視差補償予測符号化のレート歪み特性

又、傾き補償を行うと（C's）、画質は約 0.3dB 改善したが、その為に傾き情報として幅情報とほぼ同じく約 70Kbps のレート増加となり、それを含めた結果、総合のレート

歪み特性は改善するまでに至らなかった。その結果、ブロックの幅補償と傾き補償の両方を行った結果は（C'_{ws}）、幅補償のみを行った場合（C'_{w}）に比べて改善は見られなかった。この原因は、テスト画像中で傾き補償が有効な水平に近い被写体面である床部分のテキスチャが殆ど無い為、傾き補償の効果が少なかった為と思われる。

幅情報と傾き情報の最適ビットレートは、それらの量子化幅（W, S）を変えながらレート歪み特性を確認しながら探索した結果、表 6-4 左端に示す値がほぼ最適であった。

表 6- 4　ブロック形状補償付き双方向視差補償予測符号化のビットレートと PSNR

(a) 幅補償結果

Quant. parameter	Vc		Kw		Pc	Rc		C'w	
	rate (Mbps)	PSNR (dB)	rate (Mbps)	PSNR (dB)	rate (Mbps)	rate (Mbps)	PSNR (dB)	rate (Mbps)	PSNR (dB)
Q22V4W20	0.467	60.01	0.084	45.03	0.144	6.669	43.13	7.364	43.14
Q24V4W20	0.476	60.28	0.082	44.89	0.144	4.720	41.91	5.422	41.93
Q26V4W28	0.472	59.82	0.050	43.93	0.144	3.426	40.74	4.092	40.76
Q28V4W32	0.459	60.05	0.050	43.58	0.144	2.558	39.58	3.211	39.59

(b) 傾き補償結果

Quant. parameter	Vc		Ks		Pc	Rc		C's	
	rate (Mbps)	PSNR (dB)	rate (Mbps)	PSNR (dB)	rate (Mbps)	rate (Mbps)	PSNR (dB)	rate (Mbps)	PSNR (dB)
Q22V4S18	0.467	60.01	0.073	47.65	0.144	6.781	43.06	7.465	43.09
Q24V4S24	0.476	60.28	0.050	46.88	0.144	4.831	41.85	5.501	41.86
Q26V4S24	0.472	59.82	0.050	47.14	0.144	3.471	40.65	4.137	40.67
Q28V4S24	0.459	60.05	0.051	46.77	0.144	2.598	39.50	3.252	39.51

(c) 幅・傾き補償結果

Quant. parameter	Vc		Kw		Ks		Pc	Rc		C'ws	
	rate (Mbps)	PSNR (dB)	rate (Mbps)	PSNR (dB)	rate (Mbps)	PSNR (dB)	rate (Mbps)	rate (Mbps)	PSNR (dB)	rate (Mbps)	PSNR (dB)
Q22V4W20S16	0.467	60.01	0.084	45.03	0.088	49.27	0.144	6.655	43.15	7.438	43.17
Q24V4W20S20	0.476	60.28	0.082	44.89	0.056	48.00	0.144	4.725	41.92	5.483	41.94
Q26V4W28S24	0.472	59.82	0.050	43.93	0.050	47.89	0.144	3.424	40.74	4.140	40.76
Q28V4W32S28	0.459	60.05	0.050	43.58	0.050	47.39	0.144	2.560	39.59	3.263	39.60

以上の実験で得られたブロックの幅係数 K_w、傾き係数 K_s、予測誤差画像 R_c の圧縮前と圧縮後の画像、及びブロックの幅と傾き補償付き双方向視差補償予測符号化された画像の再生画像 C'_{ws} を以下に示す。図より、ブロックの幅係数と傾き係数の値が画面全体に散らばっている事から、被写体面の傾き適応の他に、視差ベクトルの誤差分の補正にも役立っていると思われる。本来は、ブロックの平行移動量の誤差を幅や傾きでは正しく補正出来ないが、ブロック内の被写体が完全な平面では無い為、画素毎に視差量が異なり、その大まかな傾向をブロックの幅補正や傾き補正で近似しているものと思われ

る。圧縮後のブロック幅係数 K'_w 及び傾き係数 K'_s は、圧縮時の量子化幅が比較的大きい為（QP=20, 16）細部の情報が失われているが有効に機能し、全体のレート歪み特性の改善に役立っている。

(a)ブロック幅係数 K_w　(b)圧縮後 K'_w(QP=20) 45.03dB(84Kbps)　(c)予測誤差画像 R_c

(d)ブロック傾き係数 K_s　(e)圧縮後 K'_s(QP=16) 49.27dB(88Kbps)　(f)圧縮後誤差画像 R'_c(QP=22) 43.15dB(6.66Mbps)

(g)視差ベクトル Vc(QP=4) 60.01dB(467Kbps)　(h)参照画像ポインタ Pc 非圧縮(144Kbps)　(i)再生画像 C'ws 43.17dB(7.44Mbps)

図 6- 31　ブロック形状補償付き双方向視差補償予測符号化の幅係数、傾き係数、予測誤差画像、視差ベクトル、参照画像ポインタ、及び再生画像

(b)　視差ベクトルの符号化

次に、視差情報の中で比較的大きなデータ量を占める視差ベクトルに関して、カメラ間の相関を利用して更なる圧縮を検討した結果を述べる。カメラ間の視差ベクトルの相

関を利用する為に、アンカーコーデックの符号化構造は、図 6-32 に示す様な、階層双方向視差補償予測符号化構造とした。ここでは、視差ベクトルの符号化効率を調べる事が目的である為、ブロックの形状補償は行なわず、符号化構造の単純化を図ったが、両者を組み合わせる事は可能である。

符号化構造は、両端のカメラ 66, 70 の画像 A66, A70 を夫々単独に JMVM でフレーム内符号化・復号化（AVC）して得られた再生画像 A, E を参照画像とし、カメラ 68 の画像 A68 を双方向視差補償予測符号化し、得られた再生画像 C'を参照画像に加えて、カメラ 67 及び 69 の画像 A67, A69 を同様にそれぞれ双方向視差補償予測符号化した。この様に階層的に双方向予測すると符号化効率が良くなる事は、MPEG MVC の提案評価結果にも示されている[20]。又、この様に階層化すると、カメラ 68 画像の予測で求められた視差ベクトル V_C の値は、カメラ 67 画像及び 69 画像の視差ベクトル V_B, V_D の値の 2 倍になっており、視差ベクトルの予測を行い易い。図中、再生された視差ベクトル V' から Predict 処理に矢印のフィードバックが掛かっているのは、先ず視差ベクトルのみの符号化を行いその再生ベクトルを用いて予測誤差画像 R を求める事を表している。

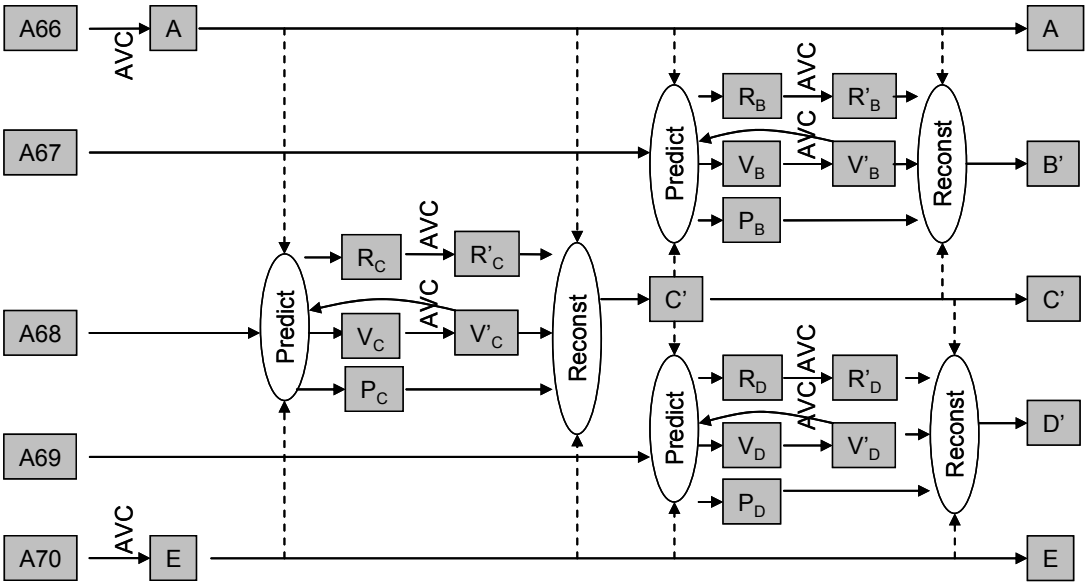


図 6- 32　双方向階層視差補償予測符号化の構成

図 6-33 と表 6-5 に、各画像のレート歪み特性を示す。図中、A, B, C, D, E は夫々独立に JMVM でフレーム内符号化したカメラ 66, 67, 68, 69, 70 の画像であり、C'は、カメラ 66, 70 の再生画像から双方向視差補償予測した画像である。この双方向予測により、同一ビットレートで約 1.5dB の画質向上、同一画質で約 20%のビットレート削減効果が見られることは、ブロック形状情報の符号化で述べた。図中、B', D'は、この再生画像 C'を参照画像に加えて、カメラ 67, 69 の画像を双方向視差補償予測した結果である。参

照画像までのカメラ間隔が半分になった為、更に符号化効率が上がり、同一画質で約 40%のビットレート削減、同一ビットレートでは約 2.5dB の画質向上が見られる。

表 6-5 では、参照画像ポインタや、視差ベクトル、予測誤差画像のレートと PSNR は示してないが、先の場合と同様の値が得られている。

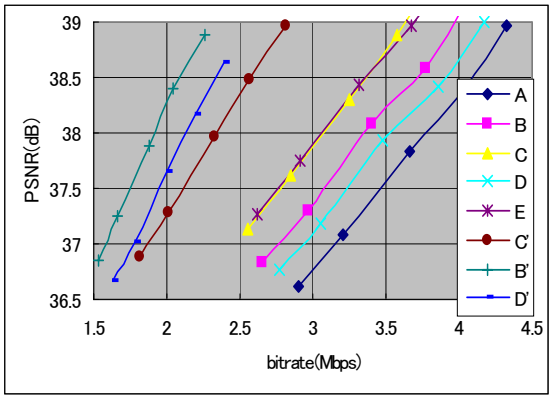


図 6- 33　視差補償予測符号化のレート歪み特性

表 6- 5　視差補償予測符号化のビットレートと PSNR

(a) アンカー画像

Quant. param	A(A66)		B(A67)		C(A68)		D(A69)		E(A70)	
	rate (Mbps)	PSNR (dB)	rate (Mbps)	PSNR (dB)	rate (Mbps)	PSNR (dB)	rate (Mbps)	PSNR (dB)	rate (Mbps)	PSNR (dB)
Q29	4.326	38.97	4.056	39.20	3.862	39.44	4.175	39.00	3.946	39.53
Q30	4.028	38.39	3.774	38.59	3.576	38.89	3.863	38.41	3.672	38.96
Q31	3.667	37.84	3.397	38.09	3.249	38.30	3.480	37.93	3.315	38.43
Q32	3.211	37.09	2.962	37.30	2.844	37.61	3.050	37.19	2.911	37.75
Q33	2.901	36.62	2.653	36.83	2.557	37.13	2.769	36.77	2.615	37.27

(b) 階層双方向視差補償予測画像

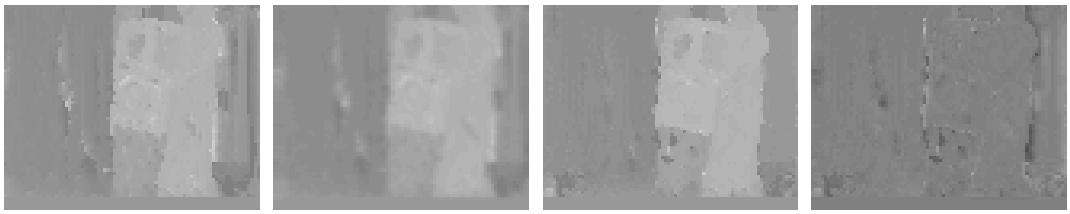
Quant. param.	C'		B'		D'	
	rate (Mbps)	PSNR (dB)	rate (Mbps)	PSNR (dB)	rate (Mbps)	PSNR (dB)
Q29V4	2.811	38.97	2.261	38.89	2.391	38.63
Q30V4	2.569	38.48	2.047	38.40	2.199	38.16
Q31V4	2.331	37.97	1.878	37.89	2.003	37.65
Q32V4	2.016	37.29	1.658	37.25	1.778	37.02
Q33V4	1.811	36.89	1.534	36.85	1.630	36.66

図 6-34 に再生された階層双方向視差補償予測画像を示すが、再生視差ベクトルで予測した結果の誤差画像をデコーダに送る事により、どのカメラ画像もビットレートの違いに関わらず、同じ画質が得られている事が確認される。

(a)再生画像 B' (Q29V4) 38.89dB(2.26Mbps)　(b) 再生画像 C' (Q29V4) 38.97dB(2.81Mbps)　(c) 再生画像 D' (Q29V4) 38.63dB(2.39Mbps)

図 6- 34　階層双方向視差補償予測符号化の復号画像

以上の結果をアンカーとし、次に視差ベクトルの予測符号化を検討した。先ず、カメラ 67, 69 画像の視差ベクトル V_B, V_D を、カメラ 68 画像の視差ベクトル V_C で予測する場合の効率を調べた。視差ベクトルは 40dB 程度の精度で予測出来たが、オクルージョン部の視差ベクトルの予測精度が上がらず、予測された視差ベクトルを用いた符号化では、予測誤差画像のデータ量が減らず、トータルのレート歪み性能は改善されなかった。



(a)視差ベクトル V_C ⇒(b)予測ベクトル V'_B -(c)視差ベクトル V_B =(d)予測誤差 V'_B -V_B (43.79dB)

図 6- 35　視差ベクトルの予測例

以上の結果より、視差ベクトルの予測で符号化効率を改善するには、更に高能率な視差ベクトル符号化の方法を開発する必要がある事が分かった。そこで、図 6-36 に示す様に、個々の視差ベクトルを予測するのではなく、隣接する複数のカメラ映像から共通の視差ベクトルを求め、この共通視差ベクトルを用いた複数のカメラ映像の予測符号化の効率を調べた。具体的には、カメラ 67, 68, 69 画像の視差ベクトル V_B, V_S, V_D を夫々の画像の両隣接画像から求めると同時に、カメラ 68 画像の視差ベクトル V_C をカメラ 66, 70 画像から求め、これら 4 ベクトルから共通視差ベクトル V_P を求め（DVCOM)、これを JMVM で符号化してデコーダに送ると共に（AVC)、エンコード側でもローカルデコードした V'_P から、3 つの視差ベクトル V'_B, V'_C, V'_D を再生し、これを用いて夫々の画像 A67, A68, A69 を階層双方向視差補償予測し（Predict)、予測誤差画像 R_B, R_C, R_D を JMVM でフレーム内符号化し(AVC)、デコーダに送る。この時、左右の参照画像の予測

誤差を調べ、誤差の少ない方の画像を参照画像として参照画像ポインタを付ける。左右の参照画像までの距離は同じであるので、視差ベクトルの値はどちらでも使える。このポインタ情報 P_B, P_C, P_D は、1bit/block で少量であるので非圧縮で送る。デコーダ側では、これらのポインタ情報と、共通視差ベクトルのビットストリームをデコードして得られる V'_P と、参照画像のデコード画像 A, B と各予測画像の予測誤差のビットストリームをデコードして得られる R'_B, R'_C, R'_D から、夫々の再生画像 b', c' d'を得る。

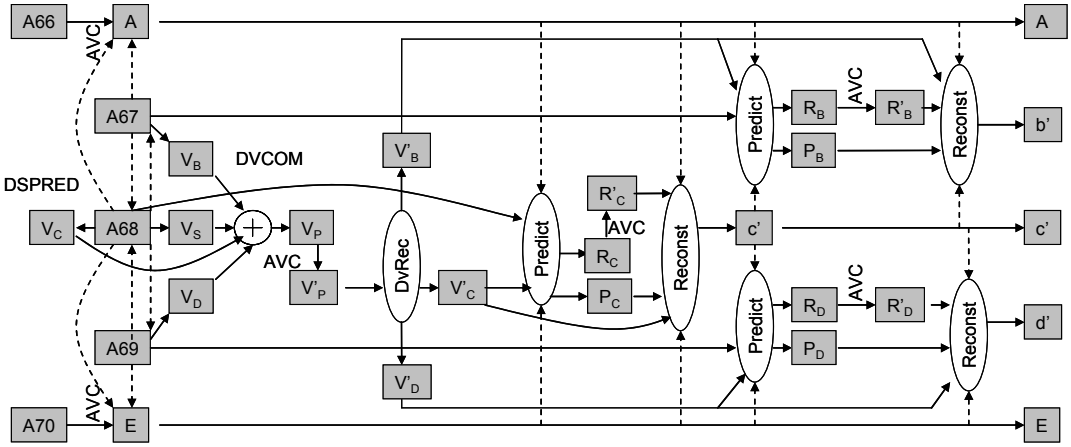


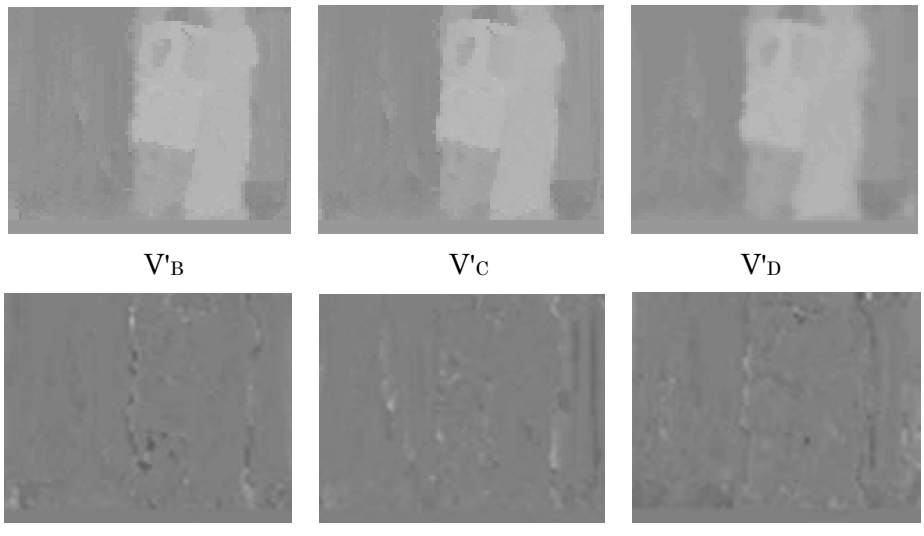
図 6- 36　共通視差ベクトルによる視差補償予測符号化構造

共通視差ベクトル V_P は、演算精度を落さない為に、絶対値が 2 倍大きい V_C をベースにし、これに各視差ベクトル V_B, V_C, V_D を 2 倍したものの位置を合わせて加算平均する。V_B, V_D の位置合わせは、画像 A68 との相対位置を考慮して各視差ベクトルの値だけブロック位置を右若しくは左にシフトすれば良い。V_C は位置が合っているのでシフトする必要はない。シフトする視差ベクトルの位置はブロック単位であるので、画素単位の視差ベクトルの値をブロックサイズ（8 画素）で割り、丸める事で求める。又、シフトすることにより、視差ベクトルがオーバーラップするオクルージョン部分では、絶対値の大きい視差ベクトルが前景の視差ベクトルであるのでこれを優先するが、オーバーラップが 1 ブロック分以下の場合は 1 ブロックの中に近景と遠景が含まれ、どちらの視差ベクトルも正しくないが、両者の平均値が比較的良い結果を与えた。シフトによりギャップが生じるオクルージョン部分は、前景に隠れていた遠景であるので、近傍の視差ベクトルの内、絶対値の小さい視差ベクトルで補間が可能であるが、ここは反対側の参照画像から正しい視差ベクトルが得られるので、補間は行わない。図 6-37 に、各視差ベクトル V_B, V_S, V_D, V_C とこれらを合成して作った V_P を示す。図では V_B, V_S, V_D の振幅を 2 倍にして V_C、V_P の振幅と合わせ見やすくしている。

V_B + V_S + V_D + V_C ⇒ V_P

図 6- 37　共通視差ベクトルの合成

デコード側では、復号した共通視差ベクトル V'_P を逆方向にシフトして、各視差ベクトル V'_B, V'_D を作る。オーバーラップ部分はエンコーダと同じ処理を行い、ギャップ部分は背景部分で構成される近傍ブロックの視差ベクトルで補間する。この時、視差ベクトルのシフト方向は画像内では同じであるので、ギャップの穴埋めは、シフト方向と同じ方向のブロックから視差ベクトルを持って来る事により、大小判定をする事なく自動的に絶対値の小さい視差ベクトルでの穴埋めが出来る。ギャップが 1 ブロック以下の場合は、上記と同様の理由により、ギャップの左右隣の視差ベクトルの平均値が比較的良い結果を与える。図 6-38 上段に再生された各視差ベクトルを示すが、V'_B と V'_D の振幅は 2 倍で表示している。下段は、再生誤差の振幅を 4 倍したものであり、グレイレベル 128 を誤差 0 として表示してある。図より、被写体のエッジ部分に誤差が集中しているのが分かる。この誤差は、3x3 タップ程度の緩い LPF を掛ける事により軽減する。



V'_B　　V'_C　　V'_D

E_B(46.45dB)　　E_C(40.89dB)　　E_D(45.25dB)

図 6- 38　再生された各視差ベクトル(V'_B,V'_C,V'_D)と誤差成分(E_B,E_C,E_D)

次に、これらの再生された視差ベクトルを用いて夫々の画像を階層双方向視差補償予測した結果の予測誤差画像を図 6-39 の上段に示し、中段に参照画像へのポインタを示し、下段に再生された画像の例を示す。ポインタ画像は白が左側、黒が右側の参照画像を予

測に使った事を表す。図より、ポインタ P_B、P_C、P_D の間にはかなりの相関が見られるので、更にビット量を削減可能と思われるが、ポインタのビット数は全体のビット量の 20 分の 1 程度であるので、例え圧縮してもその効果は薄く、圧縮は不要と思われる。

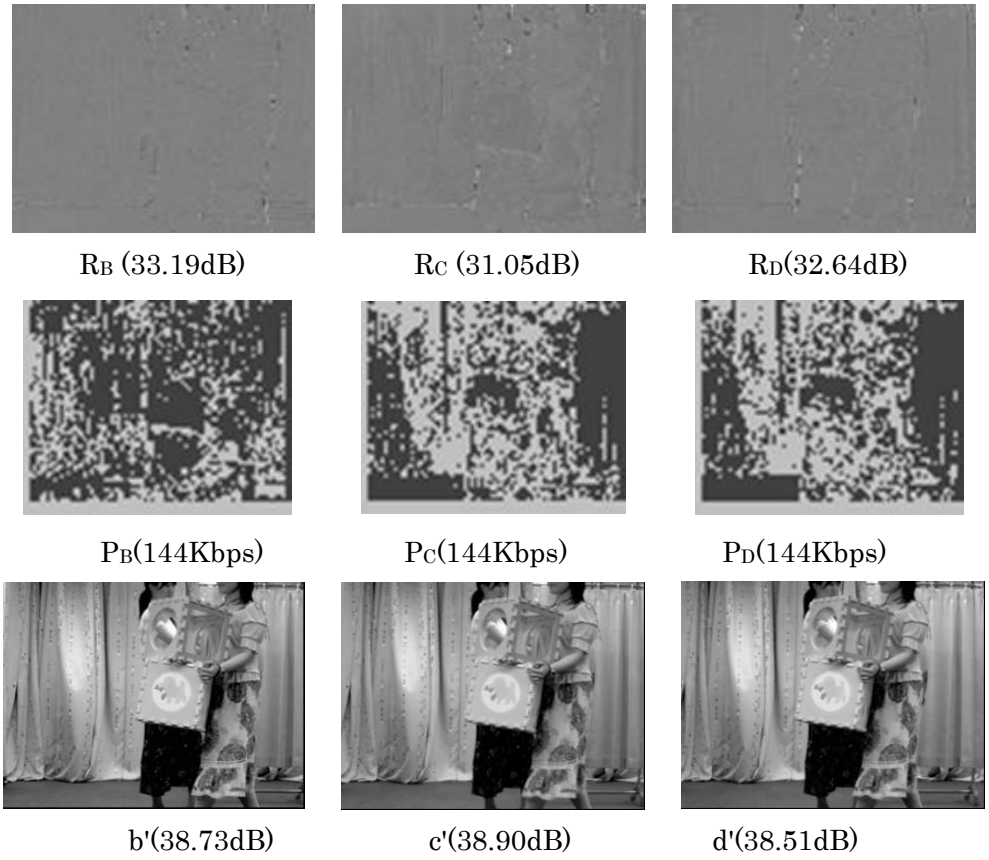
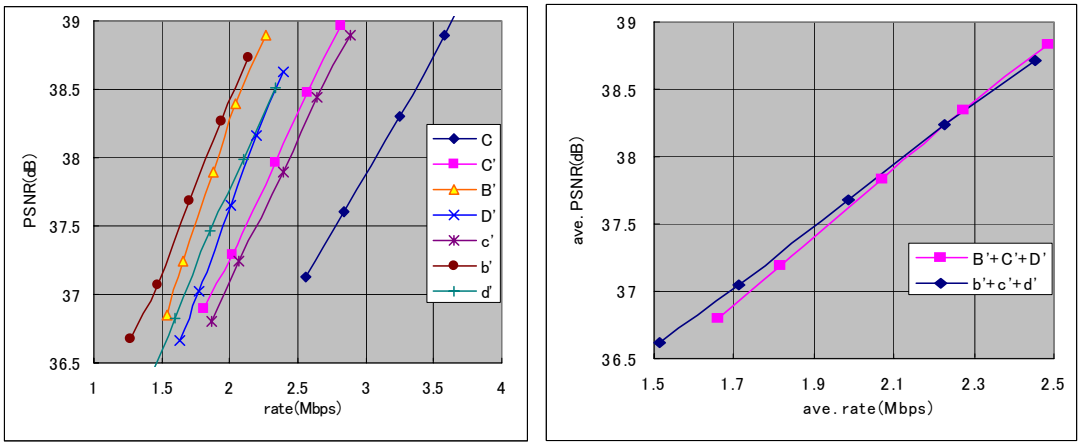


図 6- 39　共通視差ベクトルによる視差補償予測誤差(R_B,R_C,R_D)と参照画像ポインタ(P_B,P_C,P_D)及び再生画像(b', c', d')

これらのビットストリームの和と、再生画像の PSNR をレート歪み特性としてプロットしたものを、図 6-40 と、表 6-6 に示す。図 6-40 (a)は、参考としてセンター画像を単独に符号化した場合（C）と、個別の視差ベクトルを用いて階層視差補償予測した場合（B', C', D'）と、共通視差ベクトルを用いて階層視差補償予測した場合（b', c', d'）を示す。センタ画像 c'は、共通視差ベクトルのビット量を全てここに含めた為、若干レート歪み特性が悪くなったが、両隣画像 b', d'は、視差ベクトルを送る必要がなくなり、レート歪み特性が改善された。共通視差ベクトルの効果をより分かり易くする為に、個別視差補償予測画像 B', C', D'の PSNR とビットレートの平均値と、共通視差ベクトルによる予測画像の PSNR とビットレートの平均値をプロットしたものが、図の(b)である。

図より、ビットレートが低くなる程、共通視差ベクトルによる符号化の方がレート歪み特性が改善されてゆく事が分かる。これは、レートが低くなると、視差ベクトルが占

めるビット量の割合が大きくなり、個別に視差ベクトルを持つ方式では予測誤差に割けるビット量が低下して画質が劣化したのに対して、共通視差ベクトル方式では、視差ベクトルのビット量削減効果が大きく、予測誤差画像に割けるビット量に余裕があった為と思われる。



(a) 画像毎のレート歪み特性　　(b) 予測画像の平均レート歪み特性

図 6- 40　共通視差ベクトルによる階層視差補償予測符号化のレート歪み特性

表 6- 6　共通視差ベクトルによる階層視差補償予測符号化のビットレートと PSNR

(a)カメラ 67 画像

Quant. param.	V'_B	R'_B		P_B	b'	
	PSNR (dB)	rate (Mbps)	PSNR (dB)	rate (Mbps)	rate (Mbps)	PSNR (dB)
Q29V4	46.45	1.998	38.72	0.144	2.142	38.73
Q30V4	46.79	1.796	38.26	0.144	1.940	38.27
Q31V4	46.38	1.557	37.68	0.144	1.701	37.69
Q32V4	46.73	1.321	37.06	0.144	1.465	37.07
Q33V4	46.93	1.127	36.67	0.144	1.271	36.68

(b)カメラ 68 画像

Quant. param.	V'_P		V'_C	R'_C		P_C	c'	
	rate (Mbps)	PSNR (dB)	PSNR (dB)	rate (Mbps)	PSNR (dB)	rate (Mbps)	rate (Mbps)	PSNR (dB)
Q29V4	0.377	60.01	40.89	2.366	28.89	0.144	2.887	38.90
Q30V4	0.382	60.00	40.63	2.116	38.43	0.144	2.642	38.44
Q31V4	0.381	60.17	40.88	1.867	37.89	0.144	2.392	37.89
Q32V4	0.386	60.00	40.43	1.538	37.23	0.144	2.068	37.24
Q33V4	0.387	60.25	40.72	1.340	36.79	0.144	1.871	36.80

(c)カメラ 69 画像

Quant. param.	V'_D	R'_D		P_D	d'	
	PSNR (dB)	rate (Mbps)	PSNR (dB)	rate (Mbps)	rate (Mbps)	PSNR (dB)
Q29V4	45.25	2.188	38.5	0.144	2.332	38.51
Q30V4	45.34	1.958	37.98	0.144	2.102	37.99
Q31V4	45.38	1.714	37.47	0.144	1.858	37.47
Q32V4	45.26	1.455	36.81	0.144	1.599	36.82
Q33V4	45.19	1.261	36.37	0.144	1.405	36.38

(d)平均ビットレートと平均 PSNR（b'+c'+d', 参考：B'+C'+D'）

Quant. param.	b'+c'+d'	
	rate (Mbps)	PSNR (dB)
Q29V4	2.454	38.71
Q30V4	2.228	38.23
Q31V4	1.984	37.68
Q32V4	1.711	37.04
Q33V4	1.516	36.62

Quant. param.	B'+C'+D'	
	rate (Mbps)	PSNR (dB)
Q29V4	2.488	38.83
Q30V4	2.272	38.35
Q31V4	2.071	37.84
Q32V4	1.817	37.19
Q33V4	1.658	36.80

(c) 視差情報の符号化のまとめ

以上、本節では、多視点映像間に成立する規則性を利用して視差補償予測を行う為に必要な視差情報の効率の良い符号化方法を検討した。その結果、視差補償予測精度を改善出来る参照ブロックの幅や傾きの形状補償情報は、既存の MPEG-4 AVC 符号化手段を用いて効率良く圧縮出来、形状補正を行わない場合に比べて、レート歪み特性が上回る事を確認した。

又、視差補償を行う為の視差ベクトルの符号化検討では、隣接する複数カメラ画像の夫々の視差ベクトルの代わりに共通視差ベクトルを用いる事で、全体の符号化効率が向上し、平均のレート歪み特性が改善される事を確認した。

これらの符号化技術を組み合わせる事により、総合符号化効率の更なる向上が期待され、その確認は今後の課題である。

■ 第 7 章 ■

結論と今後の課題

第7章　結論と今後の課題

本論文では、立体映像配信に関する画像品質評価法及びシステム構成技術の検討を行った。そのアプローチは、理想的な立体映像の総合品質を評価する方法を先ず確立し、この評価法に基づいて現状の各種立体映像の品質を評価して、その評価結果から課題を抽出し、この課題を解決する手段として理想的な3次元立体映像をコストパフォーマンス良く生成する方法を考案し、その検証実験を行った。実験結果を検討した結果、更に理想立体映像に近付く為には、立体映像の多視点化が必要な事が判明したので、次に多視点化に伴う課題を抽出し、この課題を解決する高効率な圧縮符号化技術の検討を行った。この検討を通じて、平面映像を含む立体映像の総合品質を容易に評価出来る尺度として、映像の物理パラメータから計算される臨場感度を定義し、各種立体映像を用いた主観評価実験を通じて、定義式から得られる客観値と主観値の整合化が図れる事を確認した。又、品質評価結果から指摘された課題の1つである、全ての立体視要因を満たす立体映像方式の実現手段として、空間標本化法の提案を行い、これを実現させる為のデバイスの実現可能性を検討すると共に、標本化映像から立体映像オブジェクトを抽出する検証実験を行い、その可能性を確認した。更にこの検証実験結果の考察から指摘された立体映像の運動視差を増大させて、インターラクティブな立体映像とする為には、立体映像の多視点化が必要であるとの結論から、多視点化に伴う課題である膨大なデータを効率良く圧縮符号化する方法を検討した。そのアプローチは、先ずカメラに射影された被写体映像を解析し、被写体面の向きに応じてカメラ映像間に規則的に成立する関係式を導き、この関係式の規則性を利用した高速・高精度視差補償予測方式を提案し、これの実証実験を行い、提案された方式の画質改善効果を確認した。以下に、これらの研究内容のまとめを行い、今後の課題として、この技術の将来展望について提案技術を用いたシステムやサービスの提案を行う。

7.1 結論

本論文は、理想的な立体映像を配信する為の技術の確立を目標とし、この目標を達成する為に必要な要素技術の提案とその検証を行ったものである。

第１章では、序論として、これまでの映像技術全般の進化と課題をレビューすると共に、立体映像配信に関する技術の現状と課題を概観し、本研究で取り組む課題として、「理想的な立体映像の配信に関する技術の研究」を目標に設定した。次に、この目標を達成する手段として、立体映像の品質評価技術の確立と、理想的な立体映像の生成技術の確立

と、膨大なデータ量になる多視点立体映像の高能率圧縮符号化技術の確立を目的に掲げた。これらの目標・目的を達成する為のアプローチは、従来の２眼式立体映像ではなく、真に奥行を持つ３次元立体映像を基にしたアプローチとした。

第 2 章では、研究の背景と関連研究として、立体映像配信の為の要素技術に関して、従来研究の内容を調査・検討し、その課題を抽出した。その結果、理想的な立体映像生成方式としては、立体視の 5 大要因を全て満たして、大画面映像にも適用出来るコストパフォーマンスの良い方式が未確立である事が判明した。又、立体映像配信の実用化には立体映像の圧縮符号化が不可欠であるが、中でもインターラクティブな立体映像を実現する為に必要な、多視点化に伴う膨大な映像データを効率良く圧縮する符号化技術が未確立である事が判明した。又、立体映像の総合品質評価に関しては、立体感や没入感、信号対雑音比などを、個別に評価した前例はあるが、これらを有機的に統合した総合品質評価手段としては未確立である事が判明した。

第 3 章では、本研究の概要として、これらの調査結果を踏まえて、本研究の背景と、従来研究の概要を述べ、そこから導かれる課題の整理と、この課題の解決案の提案と、これら提案の検証結果を概説し、その有効性を述べた。

第 4 章では、３次元立体映像の評価技術と題し、立体映像の総合品質を物理パラメータで表す事を試みて臨場感度を定義し、この定義式を没入感度項と画質項と立体感度項で構成した。没入感度項は、映像の大きさを見込み角で現し、映像のきめの細かさを画素密度で現し、映像の動きの滑らかさをフレーム密度で表して、これらの積とする事で主観値との整合を図った。画質項は、映像の色のダイナミックレンジを量子化ビット数で決まる量子化信号対雑音比で表し、映像の圧縮等の信号処理で加わる雑音をピーク信号対雑音比で現し、これらの和の指数関数として主観値との整合化を図った。立体感項は、映像が提示する 5 大立体視要因を、それぞれの視覚感度で重み付け加算する事により主観値との整合を図った。5 大立体視要因の内、両眼視差項は、両眼の網膜に写る被写体映像の位置の微妙なズレ量が、被写体の奥行量によって変化する量として求めると共に、上限値を設ける事で主観値と整合させた。輻輳角項は、両眼と被写体の注視点で構成される輻輳角度が、被写体の奥行量によって変化する値として求め、焦点調節項は、被写体の奥行によって変化する、目の水晶体の焦点距離の変化量として求め、運動視差項は、立体映像の周りを回りこんだ時に、見る位置によって立体映像の見える向きが正しく変化出来る範囲を角度で表した。画像要因項は、映像中に存在する透視図法的な構図の有無や、コントラスト変化、被写体の大きさや明るさ変化等の、画像的効果があるか否かを 2 値で表す事で、平面映像の品質評価も行える様にした。次にこの定義式に基づいて、視差バリヤー型立体ディスプレイに各種実写映像を提示し、被験者による総合映像品質の主観評価値を計測すると同時に、各条件での客観値を計算し、両者の非線形回帰分析により、主観値と客観値の整合化を行った。次に、各種の大画面立体映像の品質評価を行ったが、具体的には、シャッターグラス方式のマルチ画面立体 CG 映像と、

分光眼鏡方式の大画面立体 CG 動画と、偏光眼鏡方式の大画面立体アニメ映画と、偏光眼鏡方式の大画面実写立体映像及び、実写と CG 映像の混合映像の、主観評価と客観評価を行い、両者の整合性を確認した。これらの計測実験を通じて、主観値測定項目の主成分分析を行い、客観値に対応する主観評価結果の総合品質値が、主観品質評価項目の平均値に近い事を確認してこれを整合性確認に用いた。又、主観評価値の相関分析を行い、立体映像を見る事に起因する疲労感や眼鏡意識等の心理要因が、年代や視力、映像品質項目とどの様な相関関係にあるかを分析した。更に、CG 画像などの合成映像の場合には、映像コンテンツの不完全さに起因するパラメータとして、合成映像の本物らしさ係数を導入する事で、主観値と客観値の整合化が行える事を確認した。

第５章では、３次元立体映像の生成技術と題して、先の評価結果から指摘される全ての立体感要因を効率良く満たす立体映像方式として、空間標本化法を提案した。この方法は、3 次元の線形な実空間をレンズにより非線形な立体映像空間に射影する事により、視覚特性にマッチしたコンパクトな立体映像を生成し、この立体映像を３次元に標本化・再生する手段を考案し、その原理の解析と、この方式に必要となる撮像・表示デバイスの構造とその実現可能性を検討すると共に、標本化された立体映像データから、合焦映像のみを抽出する方法を考案し、実証実験による確認を行った。又、立体映像の奥行分可能を上げる方法や、3D CG 映像等の合成映像と空間標本化法で得られる実写映像とを組み合わせたハイブリッド映像の生成方法を検討すると同時に、更に理想立体映像に近づく為には大きな運動視差確保が必要であり、その実現には立体映像の多視点化が有効であるとの考察を行った。

第６章では、３次元立体映像の符号化技術と題し、先の検討結果から指摘された立体映像の多視点化に伴う課題である、膨大な映像データを効率良く圧縮符号化する方法の検討をいった。そのアプローチは、冗長性の高い多視点映像間に成立する関係を、被写体面の傾きをパラメータにして解析して規則性を抽出し、その規則性を利用して高能率圧縮の可能性を探った。その結果、カメラが平行配列の場合は、各カメラ映像間の予測は、映像ブロックの平行移動の他に、ブロックの幅補償と傾き補償が必要な事を導出した。又、これらの補償量は、カメラ間で一定の関係を保つので容易に予測可能である事を見出した。次に、この解析結果を受けて、ブロックの片方向予測、両方向予測、ブロックサイズの細分割、ブロックの幅補償予測、ブロックの傾き補償予測の実験を行い、夫々、予測精度の改善量を確認すると共に、予測に必要な視差情報の符号化方法を検討し、レートひずみ特性による評価を行った。

第７章では、結論と今後の課題と題して、以上述べて来た本論文の総括を行い、今後の課題として、将来への展望を行った。

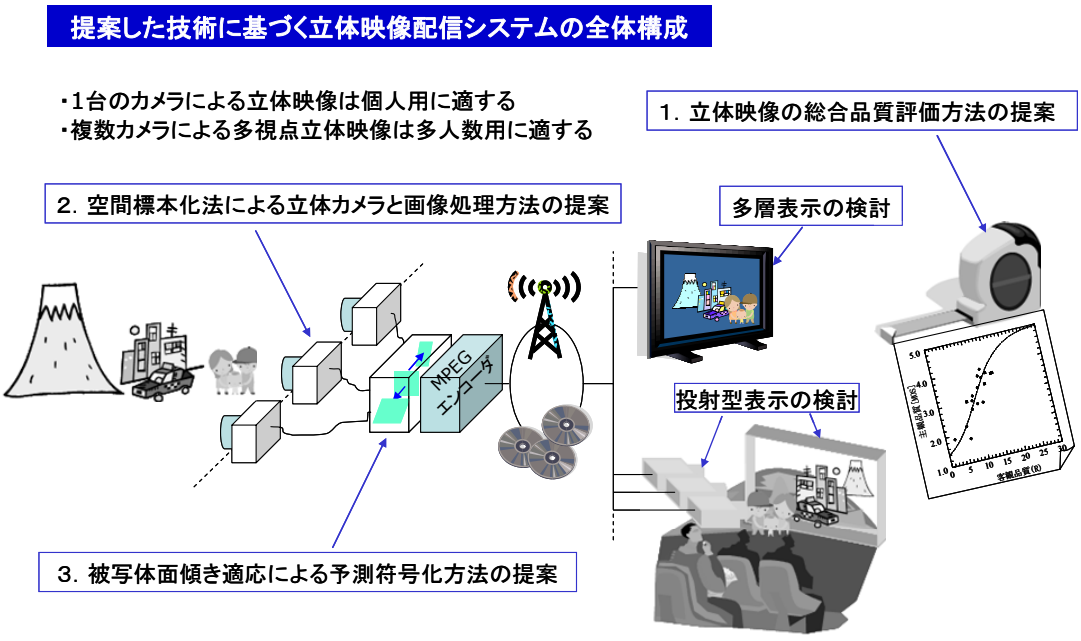


図 7- 1　本論文で報告した内容

7.2 今後の課題

今後の課題としては、本提案技術を実用化する為に周辺技術を含めた技術の総合的具現化が最優先課題である。

具体的には、臨場感度に関しては、更に多数の立体映像の品質評価を行うと同時に、評価式の改良を継続して行う事が挙げられる。

空間標本化法に関しては、これを実用化する為の、透明な撮像デバイスの試作及び、これの多層化による空間標本化デバイスの実現、これを用いた空間映像撮影カメラの試作、カメラから得られる空間映像データから合焦映像を実時間で抽出するシステムの開発、抽出された空間映像を表示する多層表示装置の開発などがある。

又、視差補償予測に関しては、更に高能率な符号化方式への符号化構造の考案と、その方式を用いた符号化実験検証と、そこから抽出される新たな課題の解決とアルゴリズムの改良などがある。

以下では更に、これらの技術をベースにした将来展望を述べると共に、それらに付随する将来の研究テーマについて述べる。

(a)　立体映像の品質評価技術の産業界への応用

本評価技術で提案した臨場感度による品質評価は、小画面から大画面までの各種方式による平面映像や立体映像の総合品質評価に使えるので、新技術の研究開発評価や、市場における製品の評価に応用出来る。更に評価映像は実写映像でも合成映像でも良いので、今後製作される幅広い映像コンテンツの評価にも応用出来、産業・技術・芸術の進歩・発展に貢献出来ると思われる。

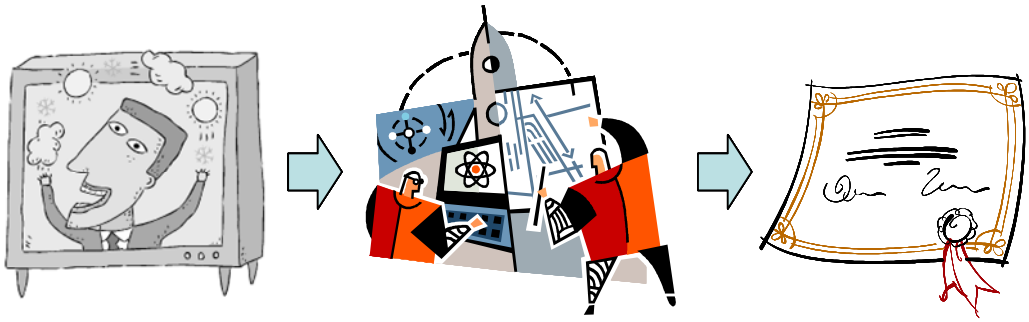


図 7- 2　臨場感度の測定サービス

この時に課題となるのは、今回定めた係数値を、より多種類の立体映像で継続して評価し直し、適宜新しいパラメータ値の導入を行って、常に更新して行く必要がある事である。その為には、臨場感度の継続的な測定を続け、市場の製品の品質評価を行うと共に、継続して評価技術の改良を行う機関が必要と思われる。この測定機関が発行した臨場感度値を、製品のカタログデータとして記載する事で、ユーザはどの製品を選べば良いかの目安とする事が出来る。

(b)　立体映像の生成技術の立体シネマへの応用

本生成技術で提案した空間標本化法は、映像の立体化の為の画素数の増加が殆ど無ないので、撮像板やディスプレイの大画面化・多画素化が容易で、立体シネマ等の高精細・大画面を必要とするアプリケーションに有効と思われる。

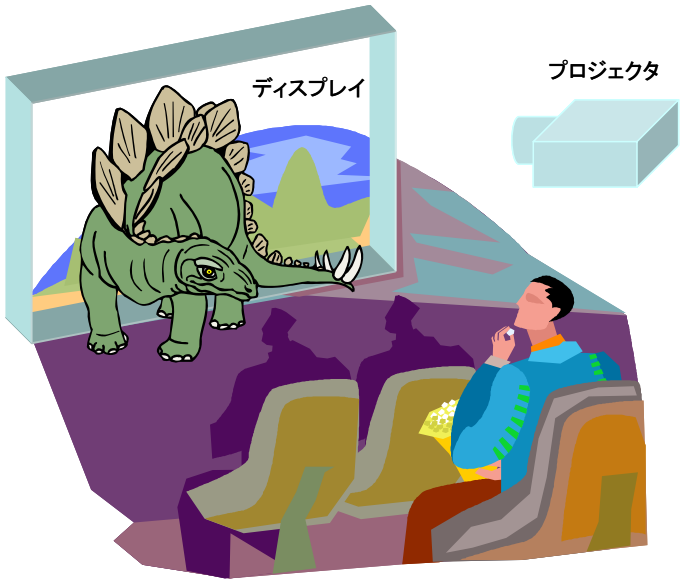


図 7- 3　空間標本化法の立体シネマへの応用

本方式を実用化するには、透明な撮像デバイスの開発が最も重要なテーマと思われるが、透明撮像デバイスの試作には、半導体メーカや、半導体プロセスを持った研究センターとの協力が必須であり、今後の課題である。多層表示デバイスは、液晶デバイスを用いたものが既に幾つかの機関・企業で試作・実用化されているが、オクルージョン部が透けて見える問題がある。この問題を解決する為の、背面に液晶シャッタを有した自己発光する透明有機 EL 素子の開発も今後の検討課題である。大画面の立体シネマへの応用には、視域の広い投影方式の実現が必須であり、屈折スクリーンの試作も含めて今後の課題である。一つの手段として、プロジェクタを複数配置して視域を分割確保する事が考えられる。この方式では、見る位置により立体映像の見える向きを変えられるので、多方向から立体映像を見ることが可能になり、立体シネマに適すると思われる。

(c)　立体映像の符号化技術の立体コンテンツ配信への応用

立体映像の符号化技術で提案した、多視点映像の視差補償予測符号化方式は、多視点映像間でオブジェクトの向きが変化する事に対して、精度良く予測符号化が出来るので、高画質符号化が行える。この特徴を生かして、コンテンツの高品質な配信･蓄積への応用が期待される。

図 7- 4　立体コンテンツの配信応用

この目的の為には、予測に使う視差ベクトルや参照画像の高能率圧縮符号化が不可欠であるが、既に開発されている MPEG-4 AVC（ITU-T H.264）をベースにする符号化技術が、うまく適合する事を示した。視差補償予測の為の参照画像を更に圧縮する方法として、時間軸方向のフレーム相関を利用する事が考えられるが、その目的にも MPEG-4 AVC は、有効であると思われる。図 7-5 に示す様に、最初にカメラ間の相関を利用した視差補償予測を行い、その結果に MPEG-4 AVC を適用して、参照カメラ映像の時間軸相関を利用した圧縮を行う。同時に、予測されたカメラ映像の視差情報も時間軸相関を持つので、MPEG-4 AVC のフレーム予測により効率の良い圧縮が可能と思われる。

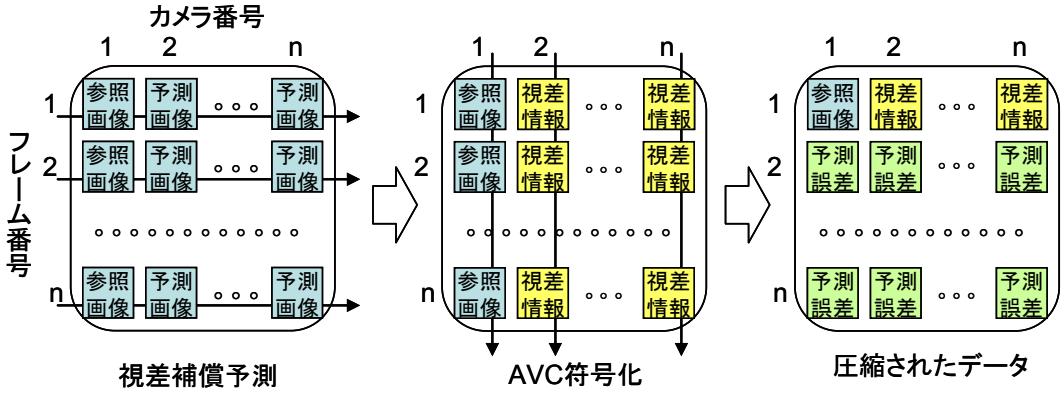


図 7- 5　視差補償予測結果の MPEG-4 AVC 圧縮

圧縮された立体映像データは、ファイルとしてネットワークを介しての伝送や、蓄積に用いられるが、ストリーミングの様な順次再生用途には、出来るだけ長い符号化単位で圧縮を行う事により、高能率な圧縮を行う必要がある。自由視点 TV の様な任意カメラ映像にランダムアクセスする用途には、出来るだけ短い符号化単位で符号化を行う必要がある。更にカメラ間の予測も、参照カメラ画像を増やし、予測カメラ画像を減らす事で、符号化効率は下がるが、ランダムアクセス能力は上がる。どの様な時空間符号化構造を採用するかは、そのアプリケーションの要求条件から最適構成を決める必要がある。

又、立体映像の特徴を利用した高耐性な透かしの挿入やコンテンツの強固なプロテクションメカニズムなどの著作権保護技術の組み込みも、立体映像配信の実用化に必要な技術であり、今後の重要な検討課題である。

謝辞

本論文をまとめるにあたり、大変多くの方々にご指導、ご助言、ご協力を頂きました。ここに、お世話になりました関係諸氏に感謝の意を表したいと思います。

本研究の遂行ならびに論文の作成にあたり、当研究室に温かく迎え入れて頂き、終始懇切なるご指導を賜りました東京大学　国際・産学共同研究センター　安田浩教授に謹んで感謝致します。又、研究テーマ設定の折から継続して有益なご助言とご教示を賜りました東京大学　先端科学技術研究センター　青木輝勝講師に深く感謝の意を表します。更に、この様な研究活動を開始する機会を設けて頂くと共に、折に触れ様々な面からご助言とご協力を頂いた東京大学　国際・産学共同研究センター　小暮拓世特任研究員、並びに小池真由美協力研究員に心より感謝致します。

又、本研究の遂行に当たり、熱心に討議して頂き、各種実験にも協力頂いた東京大学先端科学技術研究センター　安田・青木研究室の研究員・院生の方々及び、日頃の研究活動を支えて頂いた安田・青木研究室　秘書の皆様及び、デジタルシネマ事務局の方々に篤く感謝の意を表します。

更に、3D 映像フォーラムや 3D シネマフォーラムの活動を通じて、研究内容のご助言と討議をして頂き、各種の計測実験の機会を設けて頂いた東京大学　大学院　情報学環　羽倉弘之特任研究員に深く感謝致します。又、本研究の遂行に当たり、実験に参加頂くと共に、研究活動にご協力頂いた(有)エクセリードテクノロジー　沼田秀穂代表取締役、並びに池田佳代ディレクターに感謝致します。

最後に、当大学での研究生活を全面的に支え続けてくれた妻に心より感謝します。

本論文をまとめるに当たり、多くの方々にご指導、ご助言、ご協力を頂きました。ここにお世話になった方々全員に深く感謝の意を表します。

参考文献

[1] ISO/IEC 11172-2, 1992

[2] ISO/IEC 13818-2, 1995

[3] ISO/IEC 14496-2, 1999

[4] ISO/IEC 14496-10, 2003

[5] 原島 博 監修,”三次元画像と人間の科学”，オーム社，pp. 2-33，2000 年

[6] 大口 孝之,”新たなる立体映画ブームの到来”，3D 映像，Vol.19，No.4，pp.29-35，2005 年

[7] Vide & Test group, "Call for Proposal on Multi-view Video Coding", ISO/IEC JTC1/SC29/WG11 N7327, Jul. 2005

[8] Kazuyoshi Itoh, Wataru Watanabe, Hidenobu Arimoto & Keisuke Isobe, "Coherence-based 3-D and Spectral Imaging and Laser-Scanning Microscopy", Proceedings of The IEEE, Vol.94, No.3, pp.608-628, March 2006

[9] 岡野文男，星野春男，洗井淳,“インテグラルフォトグラフィの撮像系に関する検討と実験”，信学技報 IE95-146，pp.45-51，1996 年 3 月

[10] V. Javidi, R. Martinetz-Cuenca, G. Saavedra, M. Martinetz-Corral, "Orthoscopic long-focal-depth integral imaging by hybrid method", Proc. SPIE, Vol.6392, 639203, Oct. 2006

[11] Yasuhiro Takagi, "High-Density Directional Display for Generating Natural Three-Dimensional Images", Proceedings of The IEEE, Vol.94, No.3, pp.654-663, March 2006

[12] Alan C. Traub: “Three-Dimensional Display", US Pat. 3493290, Feb. 1970.

[13] 本田武士，惣司浩史，宮崎大介，向井孝彰,“デジタルマイクロミラーデバイスを用いた高解像度な傾斜型体積走査ディスプレイ”，3D イメージコンファレンス 2006, 1-2, July 2006

[14] 芝健介，宮崎大介，惣司浩史，松下賢二,“傾斜像面を用いた体積走査型 3 次元ディスプレイ”，3 次元画像コンファレンス 2005，3-4, July 2005

[15] 大塚理恵子,“TRANSPOST：360 度立体映像ディスプレイシステム”，3 次元画像コンファレンス 2005，S-2，July 2005

[16] W. Chun, J. Napoli, O. Cossiart, R. Dorval, D. Hall, T. Puttell, J. Schooler, Y. Banker, G. Favalora, "Spatial 3D Infrastructure: Display-Independent Software Framework, High-Speed Rendering Electronics, & Several New Displays", Proc. SPIE, Vol.5664, pp.302-312, 2005

[17] 陶山史郎：“高速な二周波液晶レンズによる新たな可変焦点型三次元表示方式の提

案”，3 次元画像コンファレンス’98 講演論文集，pp.10-15，1998.
[18] Benno Hendriks, Stein Kuiper: “Through A Lens Sharply”，IEEE Spectrum, pp.20-24, Dec. 2004.
[19] Bruno Berge : “機構部品ゼロ量産近付く液体レンズの実力”，日経エレ, 2005.10.24, pp.129-135, 2005
[20] Video group, “Description of Core Experiments in MVC”, ISO/IEC JTC1/SC29/WG11 N8019, Apr. 2006
[21] Aljoscha Smolic, Hideaki Kimata, “Report on 3D AV Exploration”, ISO/IEC JTC1/SC29/WG11 N5878, Jul. 2003
[22] 須佐見憲史，畑田豊彦，“３次元ディスプレイと高臨場感”，映情メディア学技報 IDY2001-124，pp.1-6, 2001.
[23] 成田長人，金沢勝，湯山一郎，“ハイビジョン画像と立体ハイビジョン画像の心理要因分析”，信学技報 EID98-174，IE98-165，pp.63-68，1999 年 2 月
[24] 増田千尋，“3 次元ディスプレイ”，産業図書，1990 年 5 月
[25] C. Fehn, E. LaBarre, S. Pastoor, "Interactive 3DTV concepts & Key technologies", Proc. IEEE, Vol.94, Issue3, pp.524-538, Mar. 2006
[26] K. Hopf, P. Chojecki, F. Newmannn, D. Przewozny, " Novel autostereoscopic single-user displays with user interaction", Proc. SPIE, Vol.6392, 639207, Oct. 2006
[27] 高梨，“高精細 3D システムオングラス液晶ディスプレイ”，3D コンソーシアム　コンファレンス 2005　予稿，2005 年 9 月
[28] 結城明正，“2”QVGA スキャンバックライト立体 LCD”，3D コンソーシアム　コンファレンス 2005　予稿，2005 年 9 月
[29] 矢野澄男，井出真司，“立体映像の見やすさと調節変動からみた視覚疲労”，信学技報 MVE99-75, Feb. 20
[30] 本田捷夫監修 : “平成 16 年度立体映像表示に関する調査研究報告書”，日本機械工業連合会/日本オプトメカトロニクス協会，日機連 16 先端-8，Mar. 2005.
[31] 安東孝久，“多視点眼鏡なし 3D ディスプレイの開発例”，3D コンソーシアム　コンファレンス 2005　予稿，2005 年 9 月
[32] S. Zwart, etal., "Switchable Auto-Stereoscopic 2D/3D Display", IDW'04, p.1459, 2004
[33] Y. Hirayama, H. Nagatani, T. Saishu, R. Fukushima, K. Taira, "Flatbed-type 3D display systems using integral imaging method", Proc. SPIE, Vol. 6392, 639209, Oct. 2006
[34] 勝間ひでとし : “6-12 3D 画像・ホログラフィ”，画像電子学会誌，Vol.33，No.6, pp.991-994, Nov. 2004.

[35] Y. Flauel, T.J. Naughton, O. Matoba, E. Tajahuerce, and B. Javidi, “Three-Dimmensional Imaging and Processing Using Computational Holographic Imaging”, Proceedings of The IEEE, Vol.94, No.3, pp.636-653, Mar. 2006.

[36] 山口健，吉川浩，“インタラクティブホログラフィックテレビジョン”，映情メディア学誌，Vol.60，No.5，pp.813-818，2006 年

[37] K. Nitta, N. Nishikawa, O. Matoba, T. Yoshimura, "Three-dimensional imaging system with stereo vision capturing and wavefromt reconstruction", Proc. SPIE, Vol.6392, 639206, Oct. 2006

[38] 永井大輔，本田捷夫，“投影光学系扇形配列による立体映像表示装置の開発”，信学技報 EID2000-233，pp.13-18，2000 年 11 月

[39] 高木康博,“変形 2 次元配置した多重テレセントリック光学系を用いた 3 次元ディスプレイ”，映情メディア学誌，Vo.57, No.2, pp.293-300, Feb. 2003.

[40] 高木康博，“高密度指向性表示 3 次元ディスプレイによる質感表示”，3 次元画像コンファレンス 2005，3-3，July 2005

[41] 中沼寛，亀井浩之，高木康博，“128 指向性画像を高密度表示する自然な 3 次元ディスプレイの開発”，3 次元画像コンファレンス 2004，2-1，June 2004

[42] Y. Takaki, K. Kikuta, "3D Images with Enhanced DOF Produced by 128-Directional Display", IDW'06, 3D4-2, Dec. 2006

[43] 榛葉俊彦，高木康博，“大画面 3 次元ディスプレイを構成する 3 次元ピクセルモジュールの開発”，映情メディア学誌，Vol.60，No.6，pp.920-927，2006 年

[44] A. Sullivan, “3 Deep”, IEEE Spectrum, pp. 22-27, Apr. 2005

[45] H. Yamaguchi, Y. Tatehira, K. Akiyama and Y. Kobayashi, “Stereo-Scopic Images Disparity for Predictive Coding”, Proc. ICASSP 1989, pp.1976-1979, 1989

[46] H. Aydinoglu, M.N. Hayes, “Stereo Image Coding: A Projection Approach”, IEEE Tans on Image Proc., Vol.7, No.4, pp.506-516, Apr. 1998

[47] T. Fujii, H. Harashima, “3-D Image Coding based on Affine Transform”, ICASSP-94, Vo.5, pp.V-577-580, April. 1994

[48] 苗村健，原島博，“自己相似モデリングによる多眼 3 次元画像の補間と情報圧縮”，TV 学誌，Voi.48，No.10，pp.1215-1221，1994 年

[49] 柳沢健之，苗村健，金子正秀，原島博，“光線空間を用いた 3 次元物体の操作”，TV 学誌，Vol.50，No.9，pp.1345-1351，1996 年

[50] 苗村健，柳原健之，金子正秀，原島博，“領域分割に基づく多眼画像の 3 次元レイヤ表現”，TV 学会誌，Vol.50，No.9，pp.1335-1344，1996 年

[51] T. Fujii, T. Kimoto and M. Tanimoto, “A New Flexible Aquisition System of Ray-Space Data for Arbitrary Objects”, IEEE Trans on Circuits & System for Video

Coding, Vol.10, No.2, pp.218-224, 2000
[52] 高野隆英, 苗村健, 原島博, “仮想オブジェクト面を用いた空間符号化”, 信学論 D-II, Vol.J82-D-II, No.10, pp.1804-1815, 1999 年
[53] Y. Taguchi, K. Takahashi, T. Naemura, "View-dependent scalable coding of light fields using ROI-based tequniques", Proc. SPIE, Vol.6392, 63920C, Oct. 2006
[54] T. Naemura, H. Harashima, "Ray-based approach to Integrated 3D Visual Communication", SPIE, 3D Video & Display: Devices & Systems, Vol.CR76, pp.282-305, 2000.11
[55] 谷本正幸, “自由視点テレビ FTV-多視点画像処理を使って”, 映情メディア学誌, Vol.58, No. 7 , pp.898-901, 2004
[56] N. Fukushima, T. Yendo, T. Fujii, M. Tanimoto, "Real-time arbitrary view interpolation & rendering system using ray-space", Proc. SPIE, Vol.6016, 60160Q, Nov. 2005
[57] 木全英明, 北原正樹, 上倉一人, 八島由幸, 藤井俊彰, 谷本正幸, “自由視点映像通信の為の低遅延多視点映像符号化”, 信学論誌 D, Vol.J89-D, No.1, pp.40-55, 2006 年
[58] 佐藤淳 : “コンピュータビジョン―視覚の幾何学―”, コロナ社, pp.30-35, May 1999.
[59] A. Smolic, H. KImata, “ Report on 3D AV Exploration”, ISO/IEC JTC1/SC29/WG11 N5878, pp.24-34, July 2003
[60] M. Magnor, P. Ramanathan and B. Girod, “Multi-View Coding for Image-Based Rendering Using 3-D Scene Geometry”, IEEE Trans on Circuits & System for Video Tech., Vol.13, No.11, pp.1092-1106, Nor. 2003
[61] 谷千束, “マルチメディア時代における高臨場感ディスプレイ”, 映情メディア学会研究会資料 IDY99-15, pp.83-89, 1999
[62] 山之上裕一, 井出真司, 奥井誠人, 湯山一郎, 尾藤峯夫, 寺島信義, “立体画像の臨場感度・見易さと視差ベクトル分布についての一考察”, 映情メディア学技報 HIR2000-146, Vol.24, No.63, pp.29-34, Oct. 2001
[63] 成田長人, 金沢勝, 湯山一郎, “ハイビジョン画像と立体ハイビジョン画像の心理分析”, 信学技報 EID98-174, IE98-165, pp.63-68, 1999 年 2 月
[64] 堀田裕弘, 河合良直, 南陽子, 村井忠邦, 中嶋芳雄, “符号化されたステレオ静止画像における画質評価モデル”, AV 複合情報処理研究会 28-6, pp.31-36, 2000 年 3 月
[65] 柳在鎬, 橋本直己, 佐藤誠, “没入型ディスプレイの映像提示領域による没入感への影響”, 映情メディア学誌, Vol.59, No.7, pp.1051-1058, 2005 年
[66] 氏家弘裕, 鵜飼一彦, 斎田真也, “映像酔いに対する運動パタンと映像コンテンツの

影響”，日本バーチャルリアリティ学界論文誌，Vol.9，No.4，pp.377-385，2004 年
[67] Akiyoshi Kitaoka : “Anomalous motion illusion and steropsis”，3D 映像，Vol.20，No.3，Oct. 2006
[68] 陶山史郎，“新たな立体錯視減少に基づく DFD(Depth-Fused-3D)ディスプレイ”，3D 映像，Vol.9，No.3，pp.20-24，Sep. 2005
[69] M. Date, H. Takada, S. Suyama, K. Tanaka, K. Nakazawa, "Projection-Type Depth Fuse 3-D(DFD) Display", IDW'06, 3Dp-4, Dec. 2006
[70] 栗林英範，石榑康雄，陶山史郎，高田英明，伊達宗和，石川和夫，畑山豊彦，“DFD(Depth-fused-3-D)表示の奥行知覚に与えるぼけの効果”, 映情メディア学誌，Vol.60，No.3，pp.431-438，2006
[71] 江本正喜，“広視野・大画面映像の視野角と臨場感”，平成 18 年度技研公開 研究発表 予稿集，pp.32-37, May 2006
[72] G. Nyman, J. Radun, T. Leisti, J. Oja, H. Ojanen, JL. Olives, T. Vuori, J. Hakkinen, "What do users really perceive: probing the subjective imaging quality", Proc. SPIE, Vol.6059, 605902, Jan. 2006
[73] F. Massidda, C. Perra, D.D. Guisto, " A human visual system model for no-reference digital video quality estimation", Proc. SPIE, Vol.5668, pp.302-313, Jan. 2005
[74] 長田昌次郎，“視覚の奥行き距離情報とその奥行感度”，テレビ誌，Vol.31，No.8，pp.649-655, 1977
[75] Xin Zhang，James Ashton-Miller，Christian Stohler : “A Closed-Loop System for Maintaining Constant Experimental Muscle Pain in Man”，IEEE Trans on Biomedical Engineering，40，4，pp. 344-352（Apr. 1993）
[76] 成橋和正，野村政明，亀井浩行，小野俊介，松下良，清水栄，横川弘一，山田清文，鈴木永雄，宮本謙一，木村和子 : “大学院修士課程臨床薬学講義ならびに実務実習の Visual Analog Scale 法による客観的評価”， 薬学雑誌，123，11，pp. 973-980，Nov. 2003
[77] 廣野元久，林俊克, "JMP による多変量データ活用術", 開文堂, June, 2004
[78] FUJIFILM，“DATA SHEET 163Ai099A フジクローム TREBI 100C”，富士写真フィルム プロフェッショナル写真部，営技・01.6-HB・10-1
[79] S. Ahn, S. Lee, A. Meyyappan, P. Schenker, "Experiments on Depth from Magnification & Blur", IEEE Proc. IROS'97, pp.733-739, 1997
[80] S. Lee, S. Ahn, A. Mayyappan, "Depth from Magnification and Blur", IEEE Proc. ICRA 1997, pp.137-142, April 1997
[81] A. Yokota, T. Yoshida, H. Kashiyama, T. Hamamoto, "High Speed Depth Estimation by Smart Image Sensor System", International Symposium on

Communication & Information Technology ISCIT2004, pp.963-968, Oct. 2004
[82] T. Yoshida, A. Yokota, H. Kashiyama, T. Hamamoto, "Smart Image Sensor for High Speed In-Focus Detection", IEEE ICIP2004, pp.2837-2840, 2004
[83] S. Nayer, Y. Nakagawa, "Shape from Focus", IEEE Trans Pattern Analysis & Machine Intelligence", Vol.16, No.8, pp.824-831, Aug. 1994
[84] M. Armad, T. Choi, " Fast & Accurate 3D Shape From Focus using Dynamic Programming Optimization Technique", IEEE ICASSP2005, pp.II-969-972, 2005
[85] 井上弘, “立体視の不思議を探る”, オプトロニクス, Feb. 1999.
[86] “Quality”, http://cweb.canon.jp/camera/ixyd/l/2_ixyl.html Japan 2005.
[87] 藤枝一郎, “画像入出力デバイスの基礎”, 森北出版 Jun. 2005.
[88] 細野秀雄, 古賀明嗣 : “室温プロセスで作製したアモルファス酸化物半導体を用いたフレキシブル薄膜トランジスタ”, Nature 2004.11.25, 2004
[89] Tadatsugu Minami, "Transparent conducting oxide semiconductors for transparent electrodes", 2005 Semicond. Sci. Tech. 20, S35-S44, 2005.
[90] 大槻智洋, “有機撮像素子で撮影成功 富士写が感度向上に道開く”, 日経エレ 2006.1.30 号, pp.36-37, Jan. 2006.
[91] H. Takada S. Suyama & K. Nakazawa: “A new 3D display method using visual illusion produced by overlapping two luminance division displays”, IEICE trans on Electron E88-C (3) p445, 2005
[92] 小林道哉, “フルカラー有機 EL ディスプレイ”, 信学誌, Vo.88, No.8, pp.646-652, 2005
[93] 三浦登, “無機 EL ディスプレイ”, 信学誌, Vol.88, No.8, pp.653-658, 2005
[94] 涌井良幸, 涌井貞実, “回帰分析”, 日本実業出版, 2002 年 6 月
[95] 廖洪恩, 岩原誠, 小池崇文, 桃井康行, 波多伸彦, 佐久間一郎, 土肥健純, “マルチプロジェクション Integral VideoGraphy 三次元画像表示装置の開発”, 信学論 D-II, Vol.J87-D-II, No.12, pp.2198-2208, 2004 年 12 月
[96] T. Koike, M. Oikawa, N. Kimura, F. Beniyama, T. Moriya, & M. Yamasaki, "Integral Videography of high-density light field with spherical layout camera array", Proc. SPIE, Vol.6055, 605510, Jan. 2006
[97] 小池崇文, 及川道雄, 宇都木契, 山崎眞見, “カラーフィルタ配置を変更した Integral Videography ディスプレイ”, 映像メディア処理シンポジウム(IMPS2006), I6-03, pp.109-110, 2006 年 11 月

研究業績

＜学会誌論文＞

(1) 妹尾 孝憲, 青木 輝勝, 安田 浩, 小暮 拓世, “立体映像のパラメータが主観に及ぼす影響”, 画像電子学会誌, （2006 年 8 月投稿済み）

(2) 妹尾 孝憲, 青木 輝勝, 安田 浩, 小暮 拓世, “多視点映像の視差補償予測の検討”, 画像電子学会誌, Vol.35, No.5, pp.488-496, 2006.

(3) 妹尾 孝憲, 青木 輝勝, 安田 浩, 小暮 拓世, “空間標本化法による立体映像の検討”, 画像電子学会誌, Vol.35, No.4, pp.317-325, 2006.

(4) Takanori Senoh & Takuyo Kogure, “Multimedia Technology Trend in MPEG4,” IEICE Trans. Electronics, Vol.E81-C, No.5, pp.642-650, May 1998.

(5) Takanori Senoo & Bernd Girod, “Vector Quantization for Entropy Coding of Image Subbands,” IEEE Tans. Image Processing, Vol.1, No.4, pp.526-533, October 1992.

＜共著学会誌論文＞

(6) Ming Ji, ShengMei. Shen, Wenjun. Zeng, Taka Senoh, Takafumi Ueno, Aoki, Terumasa Aoki, Hiroshi Yasuda, Takuyo Kogure, “MPEG-4 IPMP Extension for Interoperable Protection of Multimedia Content”, EURASIP Journal on Applied Signal Processing 2004:14, pp.2001-2013, 2004.

(7) Akiyama, T.; Aono, H.; Aoki, K.; Ler, K.W.; Wilson, B.; Araki, T.; Morishige, T.; Takeno, H.; Sato, A.; Nakatani, S.; Senoh, T.;” MPEG2 video codec using image compression DSP”, IEEE Transactions on Consumer Electronics, Volume 40, No.3, pp. 466 - 472, Aug 1994.

(8) 中村和彦, 神野一平, 妹尾孝憲, “シリアルデジタルインターフェース用チップセットの開発,” テレビジョン学会誌, Vol.47, No.10, pp1351-1357, 1993.

＜技術解説等＞

(9) 妹尾孝憲, ″3D 関連の MPEG 標準化動向″, 画像ラボ, 日本工業出版, Vol.18, No.1, 2007 年 2 月.

(10) 妹尾 孝憲, 羽倉 弘之, 青木 輝勝, 安田 浩, 小暮 拓世, “大画面立体映像の品質・心理評価”, 3D 映像, Vol.20, No.1, 2006 年 5 月.

(11) 妹尾孝憲, 青木輝勝, 安田浩, "MPEG での 3D 関連標準化状況", 3D 映像, Vol.20, No.1, pp.65-73, 2006 年 3 月.

(12)妹尾孝憲，“4.4 ディジタル時代の符号化と標準化,” 画像電子学会誌，Vol.33, No.2，pp.262-273，2004.

(13)妹尾孝憲，“1. MPEG-4 仕様レビュー” 映像情報メディア学会誌，Vol.55，No.4, pp490-491，2001.

(14)妹尾孝憲，“画像圧縮の巧妙な技術,” 日経エレクトロニクス，1992 年 5 月号, pp.5-10，1992.

＜共著技術解説等＞

(15)坂本，松居，原田，中村，妹尾，二階堂，井上，“SD メモリーカードの諸規格および著作権保護技術,” Matsushita Tech. Journal，Vol.48，No.2，pp.108-114，2002.

(16)江村恒一，妹尾孝憲，“MPEG-7 標準化動向,” 映像情報メディア学会誌，Vol.54, No.3，pp.351-355，2000.

(17)八木下，江間，秋山，長田，妹尾，“コンパクトディスクプレーヤ用信号処理 LSI,” National Tech. Report，Vol.32，No.1，pp. 81-87，1986.

(18)広田，阿部，難波江，丹羽，妹尾，“コンパクトディスクプレーヤ用半導体素子,” National Tech. Report，Vol.29，No.2，pp.278-288，1983.

(19)小田木，小坂，佐野，広田，妹尾，“ディジタルオーディオカセットレコーダ,” National Tech. Report，Vol.26，No.6，1980.

＜国際学会＞

(20) Takanori Senoh, Terumasa Aoki, Hiroshi Yasuda & Takuyo Kogure, “Reality of High-Quality Three-Dimensional Images”, IEEE Int. Conf. on Signal Processing 2006, Nov, 2006.

(21) Takanori Senoh, Terumasa Aoki, Hiroshi Yasuda & Takuyo Kogure, “Disparity-Compensated Picture-Prediction for Multiview Video Coding”, IAPR/EURASIP/TUBITAK/ITU, Int. Workshop on Multimedia Content Representation, Classification & Security, Lecture Notes in Computer Science, Springer, Vol.4105/2006, pp.699-705, Sep, 2006

(22) Takanori Senoh, Terumasa Aoki, Hiroshi Yasuda & Takuyo Kogure, “Space-Sampling Method for 3D Cinemas”, IEEE Proc of CVPRW'06 (3DCINE'06), pp.170-177, June, 2006.
（採用論文は、関連誌[Vis. & CG, Circuits & Sys. for Video Tech.等]に提出される）

(23) Takanori Senoh, Terumasa Aoki, Hiroshi Yasuda, Takuyo Kogure & Mayumi Koike, “Measurement Criteria for 3D Content Reproduction System”, 芸術科学会, NICOGRAPH Int.2005, April, 2005.

(24) Takanori. Senoh, Takafumi Ueno, Takuyo Kogure, ShengMei Shen, Ming Ji, Jing Liu, ZongYang Huang, Craig Schultz,“DRM Renewability & Interoperability,” IEEE

CCNC2004, pp.424-429, January, 2004.

(25) Takanori Senoo, Bernd Girod & Andy Lippman, et al., “Optimization of Subband Vector Quantization for Moving Images,” Proc. SMPTE Int. Conf. Sound & Vision ’90, Australia, 24, 1991.

＜共同発表国際学会＞

(26) T. Ueno, T. Senoh, et al., “Renewable Security System for Content Delivery Service,” Telecom2003, 2003.

(27) Akiyama, T.; Aono, H.; Aoki, K.; Ler, K.W.; Wilson, B.; Araki, T.; Takahashi, T.; Takeno, H.; Boon, C.; Sato, A.; Nakatani, S.; Horiike, K.; Senoh, T.; “MPEG2 Video Codec Using Image Compression DSP”, IEEE 1994 International Conference on Consumer Electronics, 1994 Digest of Technical Papers., pp.150-151, JUNE 21-23, 1994.

(28) Akiyama, T.; Aono, H.; Aoki, K.; Ler, K.W.; Wilson, B.; Araki, T.; Takahashi, T.; Takeno, H.; Boon, C.; Sato, A.; Nakatani, S.; Horiike, K.; Senoh, T.; “MPEG2 Video Codec Using Image Compression DSP”, IEEE Tans on Consumer Electronics, Vo.40, Issue 3, pp.3449-3451, JUNE 21-23, 1994.

(29) K. Nakamura, I. Kanno & T. Seno, “A Chip Set for Serial Digital Video Interface,” 135th SMPTE Tech. Conference, 1993.

(30) B. Girod & T. Senoo, “Entropy-Coded Vector Quantization of Image Subbands,” IEEE Workshop on Visual Signal Processing & Communications, September 1992.

＜研究会＞

(30) 妹尾 孝憲, 青木 輝勝, 安田 浩, 小暮 拓世, “多視点映像の視差補償予測”, 画像電子学会 第 226 回研究会, 2006 年 7 月 28 日

(31) 妹尾 孝憲, 青木 輝勝, 安田 浩, 小暮 拓世, “大画面 3D CG 映像の臨場感度：主観値と客観値の整合化”, NICOGRAPH 2006 春季大会, 2006 年 5 月

(32) 妹尾 孝憲, 青木 輝勝, 安田 浩, 小暮 拓世, “空間標本化法による立体映像の検討”, 画像電子学会 第 224 回研究会, 2006 年 3 月 17 日

(33) 妹尾 孝憲, 羽倉 弘之, 青木 輝勝, 安田 浩, 小暮 拓世, “大画面立体映像の品質・心理評価”, 3D 映像, Vol.19, No.4, pp.87-94, 2005 年 12 月

(34) 妹尾 孝憲, 青木 輝勝, 安田 浩, 小暮 拓世, “3D 映像の臨場感度計測”, 3D 映像, Vol.19, No.3, pp.14-19, 2005 年 9 月

(35) 妹尾 孝憲, 青木 輝勝, 安田 浩, 小暮 拓世, “3D 画像の臨場感度”, 画像電子学会・映像情報メディア学会共催, 映像表現&コンピュータグラフィックス研究会, 2005 年 7 月

(36) 妹尾 孝憲, 青木 輝勝, 安田 浩, 小暮 拓世, 小池 真由美, “3D マルチメディアコン

テンツ再生システムの性能評価尺度の考察", 情報処理学会, オーディオビジュアル複合処理研究会, vol.2005, no.48, 2005 年 3 月
(37)妹尾孝憲, “マルチメディアとパッケージ,” TV 学会関西支部, 専門講習会講演論文集, pp. 25-31, 1992.

＜共同発表研究会＞
(38)秋山, 青木, 荒木, Wilson, Ler, 竹野, 佐藤, 中谷, 中村, 田仲, 妹尾, “MPEG2 画像圧縮コーデック,” 信学技報 CS94-84, 1984.

＜共著書籍＞
(39)妹尾孝憲, “第 9 章 デジタル映像符号化技術,” デジタル情報流通システム, 東京電機大学出版局, pp. 126-144, 2004.
(40)妹尾孝憲, "H.323/MPEG-4 教科書　第 10 章 1, 第 12 章 7, 10," IE インスティチュート, 2001 年.
(41)妹尾孝憲, “第 2 章 MPEG-4 の概要と特徴,” MPEG-4 のすべて, pp. 19-35, 工業調査会, 1998
(42)妹尾孝憲, “第 2 章 画像圧縮技術と適用分野,” 画像圧縮技術のはなし, pp. 15-35, 工業調査会, 1993

＜共編国際規格＞
(43)ISO/IEC 13818-4:200X/AMD1, IPMPX Conformance Extensions, 2005.
(44)ISO/IEC 13818-5:200X/AMD1, IPMPX Reference Software Extensions, 2005.
(45)ISO/IEC 14496-4:2003/AMD4, IPMPX Conformance Extensions, 2004. Co-editor.
(46)ISO/IEC 14496-5:2001/AMD4, IPMPX Reference Software Extensions, 2004. Co-editor.

＜共同出願特許＞
(47)妹尾 孝憲, 沼田 秀穂, 池田 佳代, 青木 輝勝, 安田 浩, “立体映像成生システム”, 特願 2005-269901(基礎出願), 2005-352247(優先権主張出願)
(48)青木 輝勝, 妹尾 孝憲,安田 浩, “多視点画像の符号化方法及び復号化方法”, TLO 承継 4305X017-1
(49)青木 輝勝, 妹尾 孝憲, 安田 浩, 沼田 秀穂, 池田 佳代, "再生像浮遊型映像通信システム", TLO 承継 43057010-1

付録

１．空間標本化法の合焦オブジェクト抽出プログラム（MATLAB）

```
% Laplacian + Peak detection
A1 = imread('r04bw.bmp')
A2 = imread('r06bw.bmp')
A3 = imread('r12bw.bmp')
A4 = imread('r30bw.bmp')
B1 = single(A1(:,:,1))
B2 = single(A2(:,:,1))
B3 = single(A3(:,:,1))
B4 = single(A4(:,:,1))
% Laplacian Filter 7x7
F1 =[0 0 -1 -1;
       0 -1 -3 -3;
       -1 -3 0 7;
       -1 -3 7 24]
% Laplacian Filter 5x5
% F1 =[0 0 -1;
%          0 -1 -2;
%          -1 -2 16]
% Laplacian Filter 3x3
% F1 =[-1 -1;
%           -1 8]
% Laplacian Filter 5 tap
% F1 =[0 -1;
%        -1 4]
% Extend filter
F2 = F1
F2(:,4) = []
F2 = fliplr(F2)
F3 = [F1 F2]
F4 = F3
F4(4,:) = []
F4 = flipud(F4)
F = [F3;F4]
 % HF det for L1
C1 = abs(conv2(B1,F,'same'))
Min = min(min(C1))
Max = max(max(C1))
imwrite(uint8(255*(C1-Min)/(Max-Min)),'r04f3.bmp','bmp')
% HF det for L2
```

```
C2 = abs(conv2(B2,F,'same'))
Min = min(min(C2))
Max = max(max(C2))
imwrite(uint8(255*(C2-Min)/(Max-Min)),'r06f3.bmp','bmp')
% HF det for L3
C3 = abs(conv2(B3,F,'same'))
Min = min(min(C3))
Max = max(max(C3))
imwrite(uint8(255*(C3-Min)/(Max-Min)),'r12f3.bmp','bmp')
% HF det for L4
C4 = abs(conv2(B4,F,'same'))
Min = min(min(C4))
Max = max(max(C4))
imwrite(uint8(255*(C4-Min)/(Max-Min)),'r30f3.bmp','bmp')
% un-Focused Part Detection
D1 = (C1>=C2)&(C1>C3)&(C1>C4)
D2 = (C2>=C1)&(C2>=C3)&(C2>C4)
D3 = (C3>=C2)&(C3>=C4)&(C3>C1)
D4 = (C4>=C3)&(C4>C2)&(C4>C1)
D5 = (C1==C2)&(C2==C3)&(C3==C4)
imwrite(255*D1,'D04f3b.bmp','bmp')
imwrite(255*D2,'D06f3b.bmp','bmp')
imwrite(255*D3,'D12f3b.bmp','bmp')
imwrite(255*D4,'D30f3b.bmp','bmp')
imwrite(255*D5,'Duf7b.bmp','bmp')
% stop here

% Low pass filter
A1 = imread('D04f3b.bmp','bmp')
A2 = imread('D06f3b.bmp','bmp')
A3 = imread('D12f3b.bmp','bmp')
A4 = imread('D30f3b.bmp','bmp')
B1 = single(A1(:,:))
B2 = single(A2(:,:))
B3 = single(A3(:,:))
B4 = single(A4(:,:))
%   LPF 43tap
F1 = [6 6 6 7 7 7 8 8 8 9 9 9 10 10 10 10 10 10 10 11 11;
      6 6 7 7 7 8 8 8 9 9 9 10 10 10 10 11 11 11 11 11 11;
      6 7 7 7 8 8 9 9 9 10 10 10 11 11 11 11 11 12 12 12 12;
      7 7 7 8 8 9 9 9 10 10 10 11 11 11 12 12 12 12 12 12 13;
      7 7 8 8 9 9 9 10 10 11 11 11 12 12 12 12 13 13 13 13 13;
      7 8 8 9 9 10 10 10 11 11 12 12 12 13 13 13 13 13 14 14 14;
      8 8 9 9 9 10 10 11 11 12 12 12 13 13 13 14 14 14 14 14 14;
      8 8 9 9 10 10 11 11 12 12 13 13 13 14 14 14 15 15 15 15 15;
```

```
        8 9 9 10 10 11 11 12 12 13 13 14 14 14 15 15 15 15 15 16 16 16;
        9 9 10 10 11 11 12 12 13 13 14 14 14 15 15 15 16 16 16 16 16 16;
        9 9 10 10 11 12 12 13 13 14 14 15 15 15 16 16 16 16 17 17 17 17;
        9 10 10 11 11 12 12 13 14 14 15 15 15 16 16 16 17 17 17 17 17 17;
        9 10 11 11 12 12 13 13 14 14 15 15 16 16 17 17 17 17 18 18 18 18;
        10 10 11 11 12 13 13 14 14 15 15 16 16 17 17 17 18 18 18 18 18 18;
        10 10 11 12 12 13 13 14 15 15 16 16 17 17 17 18 18 18 18 19 19 19;
        10 11 11 12 12 13 14 14 15 15 16 16 17 17 18 18 18 19 19 19 19 19;
        10 11 11 12 13 13 14 15 15 16 16 17 17 18 18 18 19 19 19 19 19 19;
        10 11 12 12 13 13 14 15 15 16 16 17 17 18 18 19 19 19 19 19 20 20;
        10 11 12 12 12 14 14 15 15 16 17 17 18 18 18 19 19 19 19 20 20 20;
        10 11 12 12 13 14 14 15 16 16 17 17 18 18 19 19 19 19 20 20 20 20;
        11 11 12 12 13 14 14 15 16 16 17 17 18 18 19 19 19 20 20 20 20 20;
        11 11 12 13 13 14 14 15 16 16 17 17 18 18 19 19 19 20 20 20 20 20]
  F2 = F1
  F2(:,22) = []
  F2 = fliplr(F2)
  F3 = [F1 F2]
  F4 = F3
  F4(22,:) = []
  F4 = flipud(F4)
  F = [F3;F4]
  S = sum(sum(F))
  C1 = conv2(B1,F,'same')/S
Max = max(max(C))
Min = min(min(C))
C1 = 255*(C-Min)/(Max-Min)
imwrite(uint8(C1),'r04f3bnL43.bmp','bmp')
C2 = conv2(B2,F,'same')/S
Max = max(max(C))
Min = min(min(C))
C2 = 255*(C-Min)/(Max-Min)
imwrite(uint8(C2),'r06f3bnL43.bmp','bmp')
C3 = conv2(B3,F,'same')/S
Max = max(max(C))
Min = min(min(C))
C3 = 255*(C-Min)/(Max-Min)
imwrite(uint8(C3),'r12f3bnL43.bmp','bmp')
C4 = conv2(B4,F,'same')/S
Max = max(max(C))
Min = min(min(C))
C4 = 255*(C-Min)/(Max-Min)
imwrite(uint8(C4),'r30f3bnL43.bmp','bmp')
% LPF ome more time
C1 = conv2(C1,F,'same')/S
```

```
Max = max(max(C))
Min = min(min(C))
C1 = 255*(C-Min)/(Max-Min)
imwrite(uint8(C1),'r04f3bnLL43.bmp','bmp')
C2 = conv2(C2,F,'same')/S
Max = max(max(C))
Min = min(min(C))
C2 = 255*(C-Min)/(Max-Min)
imwrite(uint8(C2),'r06f3bnLL43.bmp','bmp')
C3 = conv2(C3,F,'same')/S
Max = max(max(C))
Min = min(min(C))
C3 = 255*(C-Min)/(Max-Min)
imwrite(uint8(C3),'r12f3bnLL43.bmp','bmp')
C4 = conv2(C4,F,'same')/S
Max = max(max(C))
Min = min(min(C))
C4 = 255*(C-Min)/(Max-Min)
imwrite(uint8(C4),'r30f3bnLL43.bmp','bmp')
% stop here

% Blur object deletion
A1 = imread('r04m.bmp','bmp')
A2 = imread('r06m.bmp','bmp')
A3 = imread('r12m.bmp','bmp')
A4 = imread('r30m.bmp','bmp')
% Input blur data
B1 = single(imread('r04f3cnLL43.bmp','bmp'))
B2 = single(imread('r06f3bnLL43.bmp','bmp'))
B3 = single(imread('r12f3bnLL43.bmp','bmp'))
B4 = single(imread('r30f3enLL43.bmp','bmp'))
% Blur data shaping
B = uint8(255./(1+exp(B1/16-3.5)))
imwrite(B,'w04f3cnLL43n35.bmp','bmp')
C1 = repmat(B,[1 1 3])
B = uint8(255./(1+exp(B2/16-3.5)))
imwrite(B,'w06f3bnLL43n35.bmp','bmp')
C2 = repmat(B,[1 1 3])
B = uint8(255./(1+exp(B3/16-3.5)))
imwrite(B,'w12f3bnLL43n35.bmp','bmp')
C3 = repmat(B,[1 1 3])
B = uint8(255./(1+exp(B4/16-3.5)))
imwrite(B,'w30f3enLL43n35.bmp','bmp')
C4 = repmat(B,[1 1 3])
% Whitening
```

```
O1 = A1 + C1
imwrite(uint8(O1),'r04m3cnLL43n35.bmp','bmp')
O2 = A2 + C2
imwrite(uint8(O2),'r06m3bnLL43n35.bmp','bmp')
O3 = A3 + C3
imwrite(uint8(O3),'r12m3bnLL43n35.bmp','bmp')
O4 = A4 + C4
imwrite(uint8(O4),'r30m3enLL43n35.bmp','bmp')
% stop here
```

２．多視点映像の形状補償付視差補償予測プログラム（MATLAB）

```
% One directional prediction (A46 -> A48)
A1 = single(imread('A46f161q.bmp','bmp'))
A2 = single(imread('A48f161q.bmp','bmp'))
B81 = zeros(240,320)
C = zeros(1,40)
MB = zeros(8,8)
D81 = zeros(30,40)
E81 = 2500*ones(30,40)
for i = 1:30
    ii = 8*i-7
    for j = 1:33
        jj = 8*j-7
        for k = 1:40
            kk = jj+15+k
            C(k) = sum(sum(abs(A1(ii:ii+7,kk:kk+7)-A2(ii:ii+7,jj:jj+7))))
        end
        [E81(i,j),D81(i,j)] = min(C)
        D81(i,j) = D81(i,j) +15
        temp = jj + D81(i,j)
        B81(ii:ii+7,jj:jj+7) = A1(ii:ii+7,temp:temp+7)
    end
end
imwrite(uint8(B81),'B81.bmp','bmp')
Emax = max(max(E81))
Dis = 255*D81/max(max(D81))
imwrite(uint8(255*(1-E81/Emax)),'E81.bmp','bmp')
imwrite(uint8(Dis),'D81.bmp','bmp')
save('dispdat81.mat')
% stop here

% One directional prediction (A50 -> A48)
load('dispdat81')
```

```
A3 = single(imread('A50f161q.bmp','bmp'))
B82 = zeros(240,320)
MB = zeros(8,8)
C = zeros(1,40)
D82 = zeros(30,40)
E82 = 2500*ones(30,40)
for i = 1:30
    ii = 8*i-7
    for j = 8:40
        jj = 8*j-7
        for k = 1:40
            kk = jj-15-k
            C(k) = sum(sum(abs(A3(ii:ii+7,kk:kk+7)-A2(ii:ii+7,jj:jj+7))))
        end
        [E82(i,j),D82(i,j)] = min(C)
        D82(i,j) = -D82(i,j) - 15
        temp = jj + D82(i,j)
        B82(ii:ii+7,jj:jj+7) = A3(ii:ii+7,temp:temp+7)
    end
end
imwrite(uint8(B82),'B82.bmp','bmp')
Emax = max(max(E82))
Err = 255*(1-E82/Emax)
imwrite(uint8(Err),'E82.bmp','bmp')
imwrite(uint8(-255*D82/max(max(-D82))),'D82.bmp','bmp')
save('dispdat81.mat')
% stop here

% Bi-directiona prediction
load('dispdat81.mat')
B83 = B81
E83 = E81
D83 = D81
E = zeros(30,40)
for i=1:30
    i2=8*i
    i1=i2-7
    for j=1:40
        j2=8*j
        j1=j2-7
        E(i,j) = (E81(i,j) > E82(i,j))
        if (E(i,j))
            B83(i1:i2,j1:j2) = B82(i1:i2,j1:j2)
            E83(i,j) = E82(i,j)
            D83(i,j) = D82(i,j)
```

```
            end
        end
end
imwrite(uint8(B83),'B83.bmp','bmp')
Emax = max(max(E83))
imwrite(uint8(255*(1-E83/Emax)),'E83posi.bmp','bmp')
imwrite(uint8(2.318*(D83+55)),'D83.bmp','bmp')
save('dispdat81.mat')
%stop here

% Block width compensation
load('dispdat81.mat')
B85 = B83
E85 = E83
M1 = zeros(8,8)
M2 = zeros(8,8)
C = zeros(1,3)
W = ones(30,40)
for i =1:30
    i2 = 8*i
    i1 = i2 - 7
    for j =1:40
        j2 = 8*j
        j1 = j2 - 7
%           if (E4(i,j) > 300)
            C(2) = E83(i,j)
            k = j1+D83(i,j)
            if (D83(i,j) > 0)
                M3 = A1(i1:i2,k:k+7)
                % check 6/8 width
                M1(:,1:3:4) = A1(i1:i2,k+1:2:k+3)
                M1(:,5:3:8) = A1(i1:i2,k+4:2:k+6)
                M1(:,2:4:6) = (A1(i1:i2,k+1:3:k+4) + 2*A1(i1:i2,k+2:3:k+5))/3
                M1(:,3:4:7) = (2*A1(i1:i2,k+2:3:k+5) + A1(i1:i2,k+3:3:k+6))/3
                C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
                % check 12/8 width
                M2(:,1:2:7) = (2*A1(i1:i2,k-2:3:k+7) + A1(i1:i2,k-1:3:k+8) )/3
                M2(:,2:2:8) = (A1(i1:i2,k-1:3:k+8) + 2*A1(i1:i2,k:3:k+9))/2
                C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
                [Err,Ind] = min(C)
                C(2) = Err
                if (Ind ==1)
                    M3 = M1
                    % check 5/8 & width
                    M1(:,1:3:7) = A1(i1:i2,k+1:2:k+5)
```

```
M1(:,2:3:8) = A1(i1:i2,k+2:2:k+6)
M1(:,3) = (2*A1(i1:i2,k+2) + 3*A1(i1:i2,k+3))/5
M1(:,6) = (3*A1(i1:i2,k+4) + 2*A1(i1:i2,k+5))/5
C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
% check 7/8 width
M2(:,4:5) = A1(i1:i2,k+3:k+4)
M2(:,1) = (6*A1(i1:i2,k) + 3*A1(i1:i2,k+1))/9
M2(:,2) = (7*A1(i1:i2,k+1) + 2*A1(i1:i2,k+2))/9
M2(:,3) = (8*A1(i1:i2,k+2) + A1(i1:i2,k+3))/9
M2(:,6) = (A1(i1:i2,k+4) + 8*A1(i1:i2,k+5))/9
M2(:,7) = (2*A1(i1:i2,k+5) + 7*A1(i1:i2,k+6))/9
M2(:,8) = (3*A1(i1:i2,k+6) + 6*A1(i1:i2,k+7))/9
C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
[Err,Ind] = min(C)
C(2) = Err
if (Ind ==1)
    M3 = M1
    % check 4.5/8 width
    M1(:,2:2:4) = A1(i1:i2,k+2:k+3)
    M1(:,5:2:7) = A1(i1:i2,k+4:k+5)
    M1(:,1:7:8) = (A1(i1:i2,k+1:4:k+5) + A1(i1:i2,k+2:4:k+6))/2
    M1(:,3) = (A1(i1:i2,k+2) + 4*A1(i1:i2,k+3))/5
    M1(:,6) = (4*A1(i1:i2,k+4) + A1(i1:i2,k+5))/5
    C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
    % check 5.5/8 width
    M2(:,1:3:4) = A1(i1:i2,k+1:2:k+3)
    M2(:,5:3:8) = A1(i1:i2,k+4:2:k+6)
    M2(:,2) = (A1(i1:i2,k+1) + 6*A1(i1:i2,k+2))/7
    M2(:,3) = (4*A1(i1:i2,k+2) + 3*A1(i1:i2,k+3))/7
    M2(:,6) = (3*A1(i1:i2,k+4) + 4*A1(i1:i2,k+5))/7
    M2(:,7) = (6*A1(i1:i2,k+5) + A1(i1:i2,k+6))/7
    C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
    [Err,Ind] = min(C)
    E85(i,j) = Err
    switch (Ind)
        case (1)
            B85(i1:i2,j1:j2) = M1(:,:)
            W8(i,j) =0.5625
        case (2)
            B85(i1:i2,j1:j2) = M3(:,:)
            W8(i,j) =0.625
        case (3)
            B85(i1:i2,j1:j2) = M2(:,:)
            W8(i,j) =0.6875
    end
```

```
elseif (Ind ==2)
    % check 5.5/8 width
    M1(:,1:3:4) = A1(i1:i2,k+1:2:k+3)
    M1(:,5:3:8) = A1(i1:i2,k+4:2:k+6)
    M1(:,2) = (A1(i1:i2,k+1) + 6*A1(i1:i2,k+2))/7
    M1(:,3) = (4*A1(i1:i2,k+2) + 3*A1(i1:i2,k+3))/7
    M1(:,6) = (3*A1(i1:i2,k+4) + 4*A1(i1:i2,k+5))/7
    M1(:,7) = (6*A1(i1:i2,k+5) + A1(i1:i2,k+6))/7
    C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
    % check 6.5/8 width
    M2(:,4:5) = A1(i1:i2,k+3:k+4)
    M2(:,2:5:7) = (A1(i1:i2,k+1:4:k+5) + A1(i1:i2,k+2:4:k+6))/2
    M2(:,1) = (A1(i1:i2,k) + 2*A1(i1:i2,k+1))/3
    M2(:,3) = (3*A1(i1:i2,k+2) + A1(i1:i2,k+3))/4
    M2(:,6) = (A1(i1:i2,k+4) + 3*A1(i1:i2,k+5))/4
    M2(:,8) = (2*A1(i1:i2,k+6) + A1(i1:i2,k+7))/3
    C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
    [Err,Ind] = min(C)
    E85(i,j) = Err
    switch (Ind)
        case (1)
            B85(i1:i2,j1:j2) = M1(:,:)
            W8(i,j) =0.6875
        case (2)
            B85(i1:i2,j1:j2) = M3(:,:)
            W8(i,j) =0.75
        case (3)
            B85(i1:i2,j1:j2) = M2(:,:)
            W8(i,j) =0.8125
    end
elseif (Ind ==3)
    M3 = M2
    % check 6.5/8 width
    M1(:,4:5) = A1(i1:i2,k+3:k+4)
    M1(:,2:5:7) = (A1(i1:i2,k+1:4:k+5) + A1(i1:i2,k+2:4:k+6))/2
    M1(:,1) = (A1(i1:i2,k) + 2*A1(i1:i2,k+1))/3
    M1(:,3) = (3*A1(i1:i2,k+2) + A1(i1:i2,k+3))/4
    M1(:,6) = (A1(i1:i2,k+4) + 3*A1(i1:i2,k+5))/4
    M1(:,8) = (2*A1(i1:i2,k+6) + A1(i1:i2,k+7))/3
    C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
    % check 7.5/8 width
    M2(:,3:6) = A1(i1:i2,k+2:k+5)
    M2(:,1) = (4*A1(i1:i2,k) + A1(i1:i2,k+1))/5
    M2(:,2) = (8*A1(i1:i2,k+1) + A1(i1:i2,k+2))/9
    M2(:,7) = (A1(i1:i2,k+5) + 8*A1(i1:i2,k+6))/9
```

```
            M2(:,8) = (A1(i1:i2,k+6) + 4*A1(i1:i2,k+7))/5
            C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
            [Err,Ind] = min(C)
            E85(i,j) = Err
            switch (Ind)
                case (1)
                    B85(i1:i2,j1:j2) = M1(:,:)
                    W8(i,j) =0.8125
                case (2)
                    B85(i1:i2,j1:j2) = M3(:,:)
                    W8(i,j) =0.875
                case (3)
                    B85(i1:i2,j1:j2) = M2(:,:)
                    W8(i,j) =0.9375
            end
        end
    elseif (Ind ==2)
        % check 7/8 width
        M1(:,4:5) = A1(i1:i2,k+3:k+4)
        M1(:,1) = (5*A1(i1:i2,k) + 3*A1(i1:i2,k+1))/8
        M1(:,2) = (7*A1(i1:i2,k+1) + 2*A1(i1:i2,k+2))/9
        M1(:,3) = (8*A1(i1:i2,k+2) + 2*A1(i1:i2,k+3))/9
        M1(:,6) = (A1(i1:i2,k+4) + 8*A1(i1:i2,k+5))/9
        M1(:,7) = (2*A1(i1:i2,k+5) + 7*A1(i1:i2,k+6))/9
        M1(:,8) = (3*A1(i1:i2,k+6) + 5*A1(i1:i2,k+7))/8
        C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
        % check 10/8 width
        M2(:,1) = (4*A1(i1:i2,k-1) + A1(i1:i2,k))/5
        M2(:,2) = (3*A1(i1:i2,k) + 2*A1(i1:i2,k+1))/5
        M2(:,3) = (2*A1(i1:i2,k+1) + 3*A1(i1:i2,k+2))/5
        M2(:,4) = (A1(i1:i2,k+2) + 4*A1(i1:i2,k+3))/5
        M2(:,5) = (4*A1(i1:i2,k+4) + A1(i1:i2,k+5))/5
        M2(:,6) = (3*A1(i1:i2,k+5) + 2*A1(i1:i2,k+6))/5
        M2(:,7) = (2*A1(i1:i2,k+6) + 3*A1(i1:i2,k+7))/5
        M2(:,8) = (A1(i1:i2,k+7) + 4*A1(i1:i2,k+8))/5
        C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
        [Err,Ind] = min(C)
        C(2) = Err
        if (Ind ==1)
            M3 = M1
            % check 6.5/8 width
            M1(:,4:5) = A1(i1:i2,k+3:k+4)
            M1(:,2:5:7) = (A1(i1:i2,k+1:4:k+5) + A1(i1:i2,k+2:4:k+6))/2
            M1(:,1) = (A1(i1:i2,k) + 2*A1(i1:i2,k+1))/3
            M1(:,3) = (3*A1(i1:i2,k+2) + A1(i1:i2,k+3))/4
```

```
        M1(:,6) = (A1(i1:i2,k+4) + 3*A1(i1:i2,k+5))/4
        M1(:,8) = (2*A1(i1:i2,k+6) + A1(i1:i2,k+7))/3
        C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
        % check 7.5/8 width
        M2(:,3:6) = A1(i1:i2,k+2:k+5)
        M2(:,1) = (4*A1(i1:i2,k) + A1(i1:i2,k+1))/5
        M2(:,2) = (8*A1(i1:i2,k+1) + A1(i1:i2,k+2))/9
        M2(:,7) = (A1(i1:i2,k+5) + 8*A1(i1:i2,k+6))/9
        M2(:,8) = (A1(i1:i2,k+6) + 4*A1(i1:i2,k+7))/5
        C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
        [Err,Ind] = min(C)
        E85(i,j) = Err
        switch (Ind)
            case (1)
                B85(i1:i2,j1:j2) = M1(:,:)
                W8(i,j) =0.8125
            case (2)
                B85(i1:i2,j1:j2) = M3(:,:)
                W8(i,j) =0.875
            case (3)
                B85(i1:i2,j1:j2) = M2(:,:)
                W8(i,j) =0.9375
        end
    elseif (Ind ==2)
        % check 7.5/8 width
        M1(:,3:6) = A1(i1:i2,k+2:k+5)
        M1(:,1) = (4*A1(i1:i2,k) + A1(i1:i2,k+1))/5
        M1(:,2) = (8*A1(i1:i2,k+1) + A1(i1:i2,k+2))/9
        M1(:,7) = (A1(i1:i2,k+5) + 8*A1(i1:i2,k+6))/9
        M1(:,8) = (A1(i1:i2,k+6) + 4*A1(i1:i2,k+7))/5
        C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
        % check 9/8 width
        M2(:,4:5) = A1(i1:i2,k+3:k+4)
        M2(:,1) = (4*A1(i1:i2,k-1) + 5*A1(i1:i2,k))/9
        M2(:,2) = (A1(i1:i2,k) + 2*A1(i1:i2,k+1))/3
        M2(:,3) = (2*A1(i1:i2,k+1) + 7*A1(i1:i2,k+2))/9
        M2(:,6) = (7*A1(i1:i2,k+5) + 2*A1(i1:i2,k+6))/9
        M2(:,7) = (2*A1(i1:i2,k+6) + A1(i1:i2,k+7))/3
        M2(:,8) = (5*A1(i1:i2,k+7) + 4*A1(i1:i2,k+8))/9
        C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
        [Err,Ind] = min(C)
        E85(i,j) = Err
        switch (Ind)
            case (1)
                B85(i1:i2,j1:j2) = M1(:,:)
```

```
                W8(i,j) =0.9375
            case (2)
                % keep B5(i1:i2,j1:j2) = M3(:,:)
                W8(i,j) =1.0
            case (3)
                B85(i1:i2,j1:j2) = M2(:,:)
                W8(i,j) =1.125
        end
    elseif (Ind ==3)
        M3 = M2
        % check 9/8 width
        M1(:,4:5) = A1(i1:i2,k+3:k+4)
        M1(:,1) = (4*A1(i1:i2,k-1) + 5*A1(i1:i2,k))/9
        M1(:,2) = (A1(i1:i2,k) + 2*A1(i1:i2,k+1))/3
        M1(:,3) = (2*A1(i1:i2,k+1) + 7*A1(i1:i2,k+2))/9
        M1(:,6) = (7*A1(i1:i2,k+5) + 2*A1(i1:i2,k+6))/9
        M1(:,7) = (2*A1(i1:i2,k+6) + A1(i1:i2,k+7))/3
        M1(:,8) = (5*A1(i1:i2,k+7) + 4*A1(i1:i2,k+8))/9
        C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
        % check 11/8 width
        M2(:,1) = (5*A1(i1:i2,k-2) + 8*A1(i1:i2,k-1))/13
        M2(:,2) = (A1(i1:i2,k-1) + 5*A1(i1:i2,k) + A1(i1:i2,k+1))/7
        M2(:,3) = (4*A1(i1:i2,k+1) + 3*A1(i1:i2,k+2))/7
        M2(:,4) = (2*A1(i1:i2,k+2) + 5*A1(i1:i2,k+3))/7
        M2(:,5) = (5*A1(i1:i2,k+4) + 2*A1(i1:i2,k+5))/7
        M2(:,6) = (3*A1(i1:i2,k+5) + 4*A1(i1:i2,k+6))/7
        M2(:,7) = (A1(i1:i2,k+6) + 5*A1(i1:i2,k+7) + A1(i1:i2,k+8))/7
        M2(:,8) = (8*A1(i1:i2,k+8) + 5*A1(i1:i2,k+9))/13
        C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
        [Err,Ind] = min(C)
        E85(i,j) = Err
        switch (Ind)
            case (1)
                B85(i1:i2,j1:j2) = M1(:,:)
                W8(i,j) =1.125
            case (2)
                B85(i1:i2,j1:j2) = M3(:,:)
                W8(i,j) =1.25
            case (3)
                B85(i1:i2,j1:j2) = M2(:,:)
                W8(i,j) =1.375
        end
    end
elseif (Ind ==3)
    M3 = M2
```

```
            % check 10/8 width
            M1(:,1) = (4*A1(i1:i2,k-1) + A1(i1:i2,k))/5
            M1(:,2) = (3*A1(i1:i2,k) + 2*A1(i1:i2,k+1))/5
            M1(:,3) = (2*A1(i1:i2,k+1) + 3*A1(i1:i2,k+2))/5
            M1(:,4) = (A1(i1:i2,k+2) + 4*A1(i1:i2,k+3))/5
            M1(:,5) = (4*A1(i1:i2,k+4) + A1(i1:i2,k+5))/5
            M1(:,6) = (3*A1(i1:i2,k+5) + 2*A1(i1:i2,k+6))/5
            M1(:,7) = (2*A1(i1:i2,k+6) + 3*A1(i1:i2,k+7))/5
            M1(:,8) = (A1(i1:i2,k+7) + 4*A1(i1:i2,k+8))/5
            C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
            % check 14/8 width
M2(:,1:4:5) = (4*A1(i1:i2,k-3:7:k+4) + 3*A1(i1:i2,k-2:7:k+5))/7
M2(:,2:4:6) = (A1(i1:i2,k-2:7:k+5) + 4*A1(i1:i2,k-1:7:k+6) + 2*A1(i1:i2,k:7:k+7))/7
M2(:,3:4:7) = (2*A1(i1:i2,k:7:k+7) + 4*A1(i1:i2,k+1:7:k+8) + A1(i1:i2,k+2:7:k+9))/7
M2(:,4:4:8) = (3*A1(i1:i2,k+2:7:k+9) + 4*A1(i1:i2,k+3:7:k+10))/7
            C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
            [Err,Ind] = min(C)
            C(2) = Err
            if (Ind ==1)
                M3 = M1
                % check 9/8 width
                M1(:,4:5) = A1(i1:i2,k+3:k+4)
                M1(:,1) = (4*A1(i1:i2,k-1) + 5*A1(i1:i2,k))/9
                M1(:,2) = (A1(i1:i2,k) + 2*A1(i1:i2,k+1))/3
                M1(:,3) = (2*A1(i1:i2,k+1) + 7*A1(i1:i2,k+2))/9
                M1(:,6) = (7*A1(i1:i2,k+5) + 2*A1(i1:i2,k+6))/9
                M1(:,7) = (2*A1(i1:i2,k+6) + A1(i1:i2,k+7))/3
                M1(:,8) = (5*A1(i1:i2,k+7) + 4*A1(i1:i2,k+8))/9
                C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
                % check 11/8 width
                M2(:,1) = (5*A1(i1:i2,k-2) + 8*A1(i1:i2,k-1))/13
                M2(:,2) = (A1(i1:i2,k-1) + 5*A1(i1:i2,k) + A1(i1:i2,k+1))/7
                M2(:,3) = (4*A1(i1:i2,k+1) + 3*A1(i1:i2,k+2))/7
                M2(:,4) = (2*A1(i1:i2,k+2) + 5*A1(i1:i2,k+3))/7
                M2(:,5) = (5*A1(i1:i2,k+4) + 2*A1(i1:i2,k+5))/7
                M2(:,6) = (3*A1(i1:i2,k+5) + 4*A1(i1:i2,k+6))/7
                M2(:,7) = (A1(i1:i2,k+6) + 5*A1(i1:i2,k+7) + A1(i1:i2,k+8))/7
                M2(:,8) = (8*A1(i1:i2,k+8) + 5*A1(i1:i2,k+9))/13
                C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
                [Err,Ind] = min(C)
                E85(i,j) = Err
                switch (Ind)
                    case (1)
                        B85(i1:i2,j1:j2) = M1(:,:)
                        W8(i,j) =1.125
```

```
            case (2)
                B85(i1:i2,j1:j2) = M3(:,:)
                W8(i,j) =1.25
            case (3)
                B85(i1:i2,j1:j2) = M2(:,:)
                W8(i,j) =1.375
        end
    elseif (Ind ==2)
        % check 11/8 width
        M1(:,1) = (5*A1(i1:i2,k-2) + 8*A1(i1:i2,k-1))/13
        M1(:,2) = (A1(i1:i2,k-1) + 5*A1(i1:i2,k) + A1(i1:i2,k+1))/7
        M1(:,3) = (4*A1(i1:i2,k+1) + 3*A1(i1:i2,k+2))/7
        M1(:,4) = (2*A1(i1:i2,k+2) + 5*A1(i1:i2,k+3))/7
        M1(:,5) = (5*A1(i1:i2,k+4) + 2*A1(i1:i2,k+5))/7
        M1(:,6) = (3*A1(i1:i2,k+5) + 4*A1(i1:i2,k+6))/7
        M1(:,7) = (A1(i1:i2,k+6) + 5*A1(i1:i2,k+7) + A1(i1:i2,k+8))/7
        M1(:,8) = (8*A1(i1:i2,k+8) + 5*A1(i1:i2,k+9))/13
        C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
        % check 13/8 width
        M2(:,2:5:7) = (A1(i1:i2,k-1:8:k+7) + A1(i1:i2,k:8:k+8))/2
        M2(:,1) = (5*A1(i1:i2,k-3) + 10*A1(i1:i2,k-2) + 2*A1(i1:i2,k-1))/17
        M2(:,3) = (A1(i1:i2,k) + 5*A1(i1:i2,k+1) + 2*A1(i1:i2,k+2))/8
        M2(:,4) = (3*A1(i1:i2,k+2) + 5*A1(i1:i2,k+3))/8
        M2(:,5) = (5*A1(i1:i2,k+4) + 3*A1(i1:i2,k+5))/8
        M2(:,6) = (3*A1(i1:i2,k+5) + 4*A1(i1:i2,k+6))/7
        M2(:,8) = (2*A1(i1:i2,k+8) + 10*A1(i1:i2,k+9) + 5*A1(i1:i2,k+10))/17
        C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
        [Err,Ind] = min(C)
        E85(i,j) = Err
        switch (Ind)
            case (1)
                B85(i1:i2,j1:j2) = M1(:,:)
                W8(i,j) =1.375
            case (2)
                B85(i1:i2,j1:j2) = M3(:,:)
                W8(i,j) =1.5
            case (3)
                B85(i1:i2,j1:j2) = M2(:,:)
                W8(i,j) =1.625
        end
    elseif (Ind ==3)
        M3 = M2
        % check 13/8 width
        M1(:,2:5:7) = (A1(i1:i2,k-1:8:k+7) + A1(i1:i2,k:8:k+8))/2
        M1(:,1) = (5*A1(i1:i2,k-3) + 10*A1(i1:i2,k-2) + 2*A1(i1:i2,k-1))/17
```

```
                M1(:,3) = (A1(i1:i2,k) + 5*A1(i1:i2,k+1) + 2*A1(i1:i2,k+2))/8
                M1(:,4) = (3*A1(i1:i2,k+2) + 5*A1(i1:i2,k+3))/8
                M1(:,5) = (5*A1(i1:i2,k+4) + 3*A1(i1:i2,k+5))/8
                M1(:,6) = (3*A1(i1:i2,k+5) + 4*A1(i1:i2,k+6))/7
                M1(:,8) = (2*A1(i1:i2,k+8) + 10*A1(i1:i2,k+9) + 5*A1(i1:i2,k+10))/17
                C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
                % check 15/8 width
                M2(:,1) = (5*A1(i1:i2,k-4) + 10*A1(i1:i2,k-3) + 3*A1(i1:i2,k-2))/18
                M2(:,2) = (7*A1(i1:i2,k-2) + 10*A1(i1:i2,k-1) + 2*A1(i1:i2,k))/19
                M2(:,3) = (8*A1(i1:i2,k) + 10*A1(i1:i2,k+1) + A1(i1:i2,k+2))/19
                M2(:,4) = (9*A1(i1:i2,k+2) + 10*A1(i1:i2,k+3))/19
                M2(:,5) = (10*A1(i1:i2,k+4) + 9*A1(i1:i2,k+5))/19
                M2(:,6) = (A1(i1:i2,k+5) + 10*A1(i1:i2,k+6) + 8*A1(i1:i2,k+7))/19
                M2(:,7) = (2*A1(i1:i2,k+7) + 10*A1(i1:i2,k+8) + 7*A1(i1:i2,k+9))/19
                M2(:,8) = (3*A1(i1:i2,k+9) + 10*A1(i1:i2,k+10) + 5*A1(i1:i2,k+11))/18
                C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
                [Err,Ind] = min(C)
                E85(i,j) = Err
                switch (Ind)
                    case (1)
                        B85(i1:i2,j1:j2) = M1(:,:)
                        W8(i,j) =1.625
                    case (2)
                        B85(i1:i2,j1:j2) = M3(:,:)
                        W8(i,j) =1.75
                    case (3)
                        B85(i1:i2,j1:j2) = M2(:,:)
                        W8(i,j) =1.875
                end
            end
        end
    else
        M3 = A3(i1:i2,k:k+7)
        % check 6/8 width
        M1(:,1:3:4) = A3(i1:i2,k+1:2:k+3)
        M1(:,5:3:8) = A3(i1:i2,k+4:2:k+6)
        M1(:,2:4:6) = (A3(i1:i2,k+1:3:k+4) + 2*A3(i1:i2,k+2:3:k+5))/3
        M1(:,3:4:7) = (2*A3(i1:i2,k+2:3:k+5) + A3(i1:i2,k+3:3:k+6))/3
        C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
        % check 12/8 width
        M2(:,1:2:7) = (2*A3(i1:i2,k-2:3:k+7) + A3(i1:i2,k-1:3:k+8) )/3
        M2(:,2:2:8) = (A3(i1:i2,k-1:3:k+8) + 2*A3(i1:i2,k:3:k+9))/2
        C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
        [Err,Ind] = min(C)
        C(2) = Err
```

```
if (Ind ==1)
    M3 = M1
    % check 5/8 & width
    M1(:,1:3:7) = A3(i1:i2,k+1:2:k+5)
    M1(:,2:3:8) = A3(i1:i2,k+2:2:k+6)
    M1(:,3) = (2*A3(i1:i2,k+2) + 3*A3(i1:i2,k+3))/5
    M1(:,6) = (3*A3(i1:i2,k+4) + 2*A3(i1:i2,k+5))/5
    C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
    % check 7/8 width
    M2(:,4:5) = A3(i1:i2,k+3:k+4)
    M2(:,1) = (6*A3(i1:i2,k) + 3*A3(i1:i2,k+1))/9
    M2(:,2) = (7*A3(i1:i2,k+1) + 2*A3(i1:i2,k+2))/9
    M2(:,3) = (8*A3(i1:i2,k+2) + A3(i1:i2,k+3))/9
    M2(:,6) = (A3(i1:i2,k+4) + 8*A3(i1:i2,k+5))/9
    M2(:,7) = (2*A3(i1:i2,k+5) + 7*A3(i1:i2,k+6))/9
    M2(:,8) = (3*A3(i1:i2,k+6) + 6*A3(i1:i2,k+7))/9
    C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
    [Err,Ind] = min(C)
    C(2) = Err
    if (Ind ==1)
        M3 = M1
        % check 4.5/8 width
        M1(:,2:2:4) = A3(i1:i2,k+2:k+3)
        M1(:,5:2:7) = A3(i1:i2,k+4:k+5)
        M1(:,1:7:8) = (A3(i1:i2,k+1:4:k+5) + A3(i1:i2,k+2:4:k+6))/2
        M1(:,3) = (A3(i1:i2,k+2) + 4*A3(i1:i2,k+3))/5
        M1(:,6) = (4*A3(i1:i2,k+4) + A3(i1:i2,k+5))/5
        C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
        % check 5.5/8 width
        M2(:,1:3:4) = A3(i1:i2,k+1:2:k+3)
        M2(:,5:3:8) = A3(i1:i2,k+4:2:k+6)
        M2(:,2) = (A3(i1:i2,k+1) + 6*A3(i1:i2,k+2))/7
        M2(:,3) = (4*A3(i1:i2,k+2) + 3*A3(i1:i2,k+3))/7
        M2(:,6) = (3*A3(i1:i2,k+4) + 4*A3(i1:i2,k+5))/7
        M2(:,7) = (6*A3(i1:i2,k+5) + A3(i1:i2,k+6))/7
        C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
        [Err,Ind] = min(C)
        E85(i,j) = Err
        switch (Ind)
            case (1)
                B85(i1:i2,j1:j2) = M1(:,:)
                W8(i,j) =0.5625
            case (2)
                B85(i1:i2,j1:j2) = M3(:,:)
                W8(i,j) =0.625
```

```
            case (3)
                B85(i1:i2,j1:j2) = M2(:,:)
                W8(i,j) =0.6875
        end
    elseif (Ind ==2)
        % check 5.5/8 width
        M1(:,1:3:4) = A3(i1:i2,k+1:2:k+3)
        M1(:,5:3:8) = A3(i1:i2,k+4:2:k+6)
        M1(:,2) = (A3(i1:i2,k+1) + 6*A3(i1:i2,k+2))/7
        M1(:,3) = (4*A3(i1:i2,k+2) + 3*A3(i1:i2,k+3))/7
        M1(:,6) = (3*A3(i1:i2,k+4) + 4*A3(i1:i2,k+5))/7
        M1(:,7) = (6*A3(i1:i2,k+5) + A3(i1:i2,k+6))/7
        C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
        % check 6.5/8 width
        M2(:,4:5) = A3(i1:i2,k+3:k+4)
        M2(:,2:5:7) = (A3(i1:i2,k+1:4:k+5) + A3(i1:i2,k+2:4:k+6))/2
        M2(:,1) = (A3(i1:i2,k) + 2*A3(i1:i2,k+1))/3
        M2(:,3) = (3*A3(i1:i2,k+2) + A3(i1:i2,k+3))/4
        M2(:,6) = (A3(i1:i2,k+4) + 3*A3(i1:i2,k+5))/4
        M2(:,8) = (2*A3(i1:i2,k+6) + A3(i1:i2,k+7))/3
        C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
        [Err,Ind] = min(C)
        E85(i,j) = Err
        switch (Ind)
            case (1)
                B85(i1:i2,j1:j2) = M1(:,:)
                W8(i,j) =0.6875
            case (2)
                B85(i1:i2,j1:j2) = M3(:,:)
                W8(i,j) =0.75
            case (3)
                B85(i1:i2,j1:j2) = M2(:,:)
                W8(i,j) =0.8125
        end
    elseif (Ind ==3)
        M3 = M2
        % check 6.5/8 width
        M1(:,4:5) = A3(i1:i2,k+3:k+4)
        M1(:,2:5:7) = (A3(i1:i2,k+1:4:k+5) + A3(i1:i2,k+2:4:k+6))/2
        M1(:,1) = (A3(i1:i2,k) + 2*A3(i1:i2,k+1))/3
        M1(:,3) = (3*A3(i1:i2,k+2) + A3(i1:i2,k+3))/4
        M1(:,6) = (A3(i1:i2,k+4) + 3*A3(i1:i2,k+5))/4
        M1(:,8) = (2*A3(i1:i2,k+6) + A3(i1:i2,k+7))/3
        C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
        % check 7.5/8 width
```

```
            M2(:,3:6) = A3(i1:i2,k+2:k+5)
            M2(:,1) = (4*A3(i1:i2,k) + A3(i1:i2,k+1))/5
            M2(:,2) = (8*A3(i1:i2,k+1) + A3(i1:i2,k+2))/9
            M2(:,7) = (A3(i1:i2,k+5) + 8*A3(i1:i2,k+6))/9
            M2(:,8) = (A3(i1:i2,k+6) + 4*A3(i1:i2,k+7))/5
            C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
            [Err,Ind] = min(C)
            E85(i,j) = Err
            switch (Ind)
                case (1)
                    B85(i1:i2,j1:j2) = M1(:,:)
                    W8(i,j) =0.8125
                case (2)
                    B85(i1:i2,j1:j2) = M3(:,:)
                    W8(i,j) =0.875
                case (3)
                    B85(i1:i2,j1:j2) = M2(:,:)
                    W8(i,j) =0.9375
            end
        end
    elseif (Ind ==2)
        % check 7/8 width
        M1(:,4:5) = A3(i1:i2,k+3:k+4)
        M1(:,1) = (5*A3(i1:i2,k) + 3*A3(i1:i2,k+1))/8
        M1(:,2) = (7*A3(i1:i2,k+1) + 2*A3(i1:i2,k+2))/9
        M1(:,3) = (8*A3(i1:i2,k+2) + 2*A3(i1:i2,k+3))/9
        M1(:,6) = (A3(i1:i2,k+4) + 8*A3(i1:i2,k+5))/9
        M1(:,7) = (2*A3(i1:i2,k+5) + 7*A3(i1:i2,k+6))/9
        M1(:,8) = (3*A3(i1:i2,k+6) + 5*A3(i1:i2,k+7))/8
        C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
        % check 10/8 width
        M2(:,1) = (4*A3(i1:i2,k-1) + A3(i1:i2,k))/5
        M2(:,2) = (3*A3(i1:i2,k) + 2*A3(i1:i2,k+1))/5
        M2(:,3) = (2*A3(i1:i2,k+1) + 3*A3(i1:i2,k+2))/5
        M2(:,4) = (A3(i1:i2,k+2) + 4*A3(i1:i2,k+3))/5
        M2(:,5) = (4*A3(i1:i2,k+4) + A3(i1:i2,k+5))/5
        M2(:,6) = (3*A3(i1:i2,k+5) + 2*A3(i1:i2,k+6))/5
        M2(:,7) = (2*A3(i1:i2,k+6) + 3*A3(i1:i2,k+7))/5
        M2(:,8) = (A3(i1:i2,k+7) + 4*A3(i1:i2,k+8))/5
        C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
        [Err,Ind] = min(C)
        C(2) = Err
        if (Ind ==1)
            M3 = M1
            % check 6.5/8 width
```

```
        M1(:,4:5) = A3(i1:i2,k+3:k+4)
        M1(:,2:5:7) = (A3(i1:i2,k+1:4:k+5) + A3(i1:i2,k+2:4:k+6))/2
        M1(:,1) = (A3(i1:i2,k) + 2*A3(i1:i2,k+1))/3
        M1(:,3) = (3*A3(i1:i2,k+2) + A3(i1:i2,k+3))/4
        M1(:,6) = (A3(i1:i2,k+4) + 3*A3(i1:i2,k+5))/4
        M1(:,8) = (2*A3(i1:i2,k+6) + A3(i1:i2,k+7))/3
        C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
        % check 7.5/8 width
        M2(:,3:6) = A3(i1:i2,k+2:k+5)
        M2(:,1) = (4*A3(i1:i2,k) + A3(i1:i2,k+1))/5
        M2(:,2) = (8*A3(i1:i2,k+1) + A3(i1:i2,k+2))/9
        M2(:,7) = (A3(i1:i2,k+5) + 8*A3(i1:i2,k+6))/9
        M2(:,8) = (A3(i1:i2,k+6) + 4*A3(i1:i2,k+7))/5
        C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
        [Err,Ind] = min(C)
        E85(i,j) = Err
        switch (Ind)
            case (1)
                B85(i1:i2,j1:j2) = M1(:,:)
                W8(i,j) =0.8125
            case (2)
                B85(i1:i2,j1:j2) = M3(:,:)
                W8(i,j) =0.875
            case (3)
                B85(i1:i2,j1:j2) = M2(:,:)
                W8(i,j) =0.9375
        end
    elseif (Ind ==2)
        % check 7.5/8 width
        M1(:,3:6) = A3(i1:i2,k+2:k+5)
        M1(:,1) = (4*A3(i1:i2,k) + A3(i1:i2,k+1))/5
        M1(:,2) = (8*A3(i1:i2,k+1) + A3(i1:i2,k+2))/9
        M1(:,7) = (A3(i1:i2,k+5) + 8*A3(i1:i2,k+6))/9
        M1(:,8) = (A3(i1:i2,k+6) + 4*A3(i1:i2,k+7))/5
        C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
        % check 9/8 width
        M2(:,4:5) = A3(i1:i2,k+3:k+4)
        M2(:,1) = (4*A3(i1:i2,k-1) + 5*A3(i1:i2,k))/9
        M2(:,2) = (A3(i1:i2,k) + 2*A3(i1:i2,k+1))/3
        M2(:,3) = (2*A3(i1:i2,k+1) + 7*A3(i1:i2,k+2))/9
        M2(:,6) = (7*A3(i1:i2,k+5) + 2*A3(i1:i2,k+6))/9
        M2(:,7) = (2*A3(i1:i2,k+6) + A3(i1:i2,k+7))/3
        M2(:,8) = (5*A3(i1:i2,k+7) + 4*A3(i1:i2,k+8))/9
        C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
        [Err,Ind] = min(C)
```

```
    E85(i,j) = Err
    switch (Ind)
        case (1)
            B85(i1:i2,j1:j2) = M1(:,:)
            W8(i,j) =0.9375
        case (2)
            % keep B85(i1:i2,j1:j2) = M3(:,:)
            W8(i,j) =1.0
        case (3)
            B85(i1:i2,j1:j2) = M2(:,:)
            W8(i,j) =1.125
    end
elseif (Ind ==3)
     M3 = M2
    % check 9/8 width
    M1(:,4:5) = A3(i1:i2,k+3:k+4)
    M1(:,1) = (4*A3(i1:i2,k-1) + 5*A3(i1:i2,k))/9
    M1(:,2) = (A3(i1:i2,k) + 2*A3(i1:i2,k+1))/3
    M1(:,3) = (2*A3(i1:i2,k+1) + 7*A3(i1:i2,k+2))/9
    M1(:,6) = (7*A3(i1:i2,k+5) + 2*A3(i1:i2,k+6))/9
    M1(:,7) = (2*A3(i1:i2,k+6) + A3(i1:i2,k+7))/3
    M1(:,8) = (5*A3(i1:i2,k+7) + 4*A3(i1:i2,k+8))/9
    C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
     % check 11/8 width
    M2(:,1) = (5*A3(i1:i2,k-2) + 8*A3(i1:i2,k-1))/13
    M2(:,2) = (A3(i1:i2,k-1) + 5*A3(i1:i2,k) + A3(i1:i2,k+1))/7
    M2(:,3) = (4*A3(i1:i2,k+1) + 3*A3(i1:i2,k+2))/7
    M2(:,4) = (2*A3(i1:i2,k+2) + 5*A3(i1:i2,k+3))/7
    M2(:,5) = (5*A3(i1:i2,k+4) + 2*A3(i1:i2,k+5))/7
    M2(:,6) = (3*A3(i1:i2,k+5) + 4*A3(i1:i2,k+6))/7
    M2(:,7) = (A3(i1:i2,k+6) + 5*A3(i1:i2,k+7) + A3(i1:i2,k+8))/7
    M2(:,8) = (8*A3(i1:i2,k+8) + 5*A3(i1:i2,k+9))/13
    C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
    [Err,Ind] = min(C)
    E85(i,j) = Err
    switch (Ind)
        case (1)
            B85(i1:i2,j1:j2) = M1(:,:)
            W8(i,j) =1.125
        case (2)
            B85(i1:i2,j1:j2) = M3(:,:)
            W8(i,j) =1.25
        case (3)
            B85(i1:i2,j1:j2) = M2(:,:)
            W8(i,j) =1.375
```

```
            end
        end
    elseif (Ind ==3)
        M3 = M2
        % check 10/8 width
        M1(:,1) = (4*A3(i1:i2,k-1) + A3(i1:i2,k))/5
        M1(:,2) = (3*A3(i1:i2,k) + 2*A3(i1:i2,k+1))/5
        M1(:,3) = (2*A3(i1:i2,k+1) + 3*A3(i1:i2,k+2))/5
        M1(:,4) = (A3(i1:i2,k+2) + 4*A3(i1:i2,k+3))/5
        M1(:,5) = (4*A3(i1:i2,k+4) + A3(i1:i2,k+5))/5
        M1(:,6) = (3*A3(i1:i2,k+5) + 2*A3(i1:i2,k+6))/5
        M1(:,7) = (2*A3(i1:i2,k+6) + 3*A3(i1:i2,k+7))/5
        M1(:,8) = (A3(i1:i2,k+7) + 4*A3(i1:i2,k+8))/5
        C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
        % check 14/8 width
M2(:,1:4:5) = (4*A3(i1:i2,k-3:7:k+4) + 3*A3(i1:i2,k-2:7:k+5))/7
M2(:,2:4:6) = (A3(i1:i2,k-2:7:k+5) + 4*A3(i1:i2,k-1:7:k+6) + 2*A3(i1:i2,k:7:k+7))/7
M2(:,3:4:7) = (2*A3(i1:i2,k:7:k+7) + 4*A3(i1:i2,k+1:7:k+8) + A3(i1:i2,k+2:7:k+9))/7
M2(:,4:4:8) = (3*A3(i1:i2,k+2:7:k+9) + 4*A3(i1:i2,k+3:7:k+10))/7
        C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
        [Err,Ind] = min(C)
        C(2) = Err
        if (Ind ==1)
            M3 = M1
            % check 9/8 width
            M1(:,4:5) = A3(i1:i2,k+3:k+4)
            M1(:,1) = (4*A3(i1:i2,k-1) + 5*A3(i1:i2,k))/9
            M1(:,2) = (A3(i1:i2,k) + 2*A3(i1:i2,k+1))/3
            M1(:,3) = (2*A3(i1:i2,k+1) + 7*A3(i1:i2,k+2))/9
            M1(:,6) = (7*A3(i1:i2,k+5) + 2*A3(i1:i2,k+6))/9
            M1(:,7) = (2*A3(i1:i2,k+6) + A3(i1:i2,k+7))/3
            M1(:,8) = (5*A3(i1:i2,k+7) + 4*A3(i1:i2,k+8))/9
            C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
             % check 11/8 width
            M2(:,1) = (5*A3(i1:i2,k-2) + 8*A3(i1:i2,k-1))/13
            M2(:,2) = (A3(i1:i2,k-1) + 5*A3(i1:i2,k) + A3(i1:i2,k+1))/7
            M2(:,3) = (4*A3(i1:i2,k+1) + 3*A3(i1:i2,k+2))/7
            M2(:,4) = (2*A3(i1:i2,k+2) + 5*A3(i1:i2,k+3))/7
            M2(:,5) = (5*A3(i1:i2,k+4) + 2*A3(i1:i2,k+5))/7
            M2(:,6) = (3*A3(i1:i2,k+5) + 4*A3(i1:i2,k+6))/7
            M2(:,7) = (A3(i1:i2,k+6) + 5*A3(i1:i2,k+7) + A3(i1:i2,k+8))/7
            M2(:,8) = (8*A3(i1:i2,k+8) + 5*A3(i1:i2,k+9))/13
            C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
            [Err,Ind] = min(C)
            E85(i,j) = Err
```

```
        switch (Ind)
            case (1)
                B85(i1:i2,j1:j2) = M1(:,:)
                W8(i,j) =1.125
            case (2)
                B85(i1:i2,j1:j2) = M3(:,:)
                W8(i,j) =1.25
            case (3)
                B85(i1:i2,j1:j2) = M2(:,:)
                W8(i,j) =1.375
        end
    elseif (Ind ==2)
        % check 11/8 width
        M1(:,1) = (5*A3(i1:i2,k-2) + 8*A3(i1:i2,k-1))/13
        M1(:,2) = (A3(i1:i2,k-1) + 5*A3(i1:i2,k) + A3(i1:i2,k+1))/7
        M1(:,3) = (4*A3(i1:i2,k+1) + 3*A3(i1:i2,k+2))/7
        M1(:,4) = (2*A3(i1:i2,k+2) + 5*A3(i1:i2,k+3))/7
        M1(:,5) = (5*A3(i1:i2,k+4) + 2*A3(i1:i2,k+5))/7
        M1(:,6) = (3*A3(i1:i2,k+5) + 4*A3(i1:i2,k+6))/7
        M1(:,7) = (A3(i1:i2,k+6) + 5*A3(i1:i2,k+7) + A3(i1:i2,k+8))/7
        M1(:,8) = (8*A3(i1:i2,k+8) + 5*A3(i1:i2,k+9))/13
        C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
        % check 13/8 width
        M2(:,2:5:7) = (A3(i1:i2,k-1:8:k+7) + A3(i1:i2,k:8:k+8))/2
        M2(:,1) = (5*A3(i1:i2,k-3) + 10*A3(i1:i2,k-2) + 2*A3(i1:i2,k-1))/17
        M2(:,3) = (A3(i1:i2,k) + 5*A3(i1:i2,k+1) + 2*A3(i1:i2,k+2))/8
        M2(:,4) = (3*A3(i1:i2,k+2) + 5*A3(i1:i2,k+3))/8
        M2(:,5) = (5*A3(i1:i2,k+4) + 3*A3(i1:i2,k+5))/8
        M2(:,6) = (3*A3(i1:i2,k+5) + 4*A3(i1:i2,k+6))/7
        M2(:,8) = (2*A3(i1:i2,k+8) + 10*A3(i1:i2,k+9) + 5*A3(i1:i2,k+10))/17
        C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
        [Err,Ind] = min(C)
        E85(i,j) = Err
        switch (Ind)
            case (1)
                B85(i1:i2,j1:j2) = M1(:,:)
                W8(i,j) =1.375
            case (2)
                B85(i1:i2,j1:j2) = M3(:,:)
                W8(i,j) =1.5
            case (3)
                B85(i1:i2,j1:j2) = M2(:,:)
                W8(i,j) =1.625
        end
    elseif (Ind ==3)
```

```
                        M3 = M2
                        % check 13/8 width
                        M1(:,2:5:7) = (A3(i1:i2,k-1:8:k+7) + A3(i1:i2,k:8:k+8))/2
                        M1(:,1) = (5*A3(i1:i2,k-3) + 10*A3(i1:i2,k-2) + 2*A3(i1:i2,k-1))/17
                        M1(:,3) = (A3(i1:i2,k) + 5*A3(i1:i2,k+1) + 2*A3(i1:i2,k+2))/8
                        M1(:,4) = (3*A3(i1:i2,k+2) + 5*A3(i1:i2,k+3))/8
                        M1(:,5) = (5*A3(i1:i2,k+4) + 3*A3(i1:i2,k+5))/8
                        M1(:,6) = (3*A3(i1:i2,k+5) + 4*A3(i1:i2,k+6))/7
                        M1(:,8) = (2*A3(i1:i2,k+8) + 10*A3(i1:i2,k+9) + 5*A1(i1:i2,k+10))/17
                        C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
                        % check 15/8 width
                        M2(:,1) = (5*A3(i1:i2,k-4) + 10*A3(i1:i2,k-3) + 3*A3(i1:i2,k-2))/18
                        M2(:,2) = (7*A3(i1:i2,k-2) + 10*A3(i1:i2,k-1) + 2*A3(i1:i2,k))/19
                        M2(:,3) = (8*A3(i1:i2,k) + 10*A3(i1:i2,k+1) + A3(i1:i2,k+2))/19
                        M2(:,4) = (9*A3(i1:i2,k+2) + 10*A3(i1:i2,k+3))/19
                        M2(:,5) = (10*A3(i1:i2,k+4) + 9*A3(i1:i2,k+5))/19
                        M2(:,6) = (A3(i1:i2,k+5) + 10*A3(i1:i2,k+6) + 8*A3(i1:i2,k+7))/19
                        M2(:,7) = (2*A3(i1:i2,k+7) + 10*A3(i1:i2,k+8) + 7*A3(i1:i2,k+9))/19
                        M2(:,8) = (3*A3(i1:i2,k+9) + 10*A3(i1:i2,k+10) + 5*A3(i1:i2,k+11))/18
                        C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
                        [Err,Ind] = min(C)
                        E85(i,j) = Err
                        switch (Ind)
                            case (1)
                                B85(i1:i2,j1:j2) = M1(:,:)
                                W8(i,j) =1.625
                            case (2)
                                B85(i1:i2,j1:j2) = M3(:,:)
                                W8(i,j) =1.75
                            case (3)
                                B85(i1:i2,j1:j2) = M2(:,:)
                                W8(i,j) =1.875
                        end
                    end
                end
            end
%           end
    end
end
imwrite(uint8(B85),'B85width.bmp','bmp')
imwrite(uint8(128*W8),'width8.bmp','bmp')
imwrite(uint8(255*(1-E85/max(max(E85)))),'E85posi.bmp','bmp')
E85t = sum(sum(E85))
save('dispdat81.mat')
% stop here
```

```
% Block slant compensation
load('dispdat81.mat')
B84 = B83
E84 = E83
M1 = zeros(8,8)
M2 = zeros(8,8)
C = zeros(1,3)
S = ones(30,40)
for i =1:30
    i2 = 8*i
    i1 = i2 - 7
    for j =1:40
        j2 = 8*j
        j1 = j2 - 7
            C(2) = E83(i,j)
            k = j1+D83(i,j)
            if (D83(i,j) > 0)
                M3 = A1(i1:i2,k:k+7)
                % check -1.5pel slant
                M1(1,:) = (2*A1(i1,k-1:k+6) + A1(i1,k:k+7))/3
                M1(2,:) = (A1(i1+1,k-1:k+6) + A1(i1+1,k:k+7))/2
                M1(3,:) = (2*A1(i1+2,k-1:k+6) + 5*A1(i1+2,k:k+7))/7
                M1(4,:) = A1(i1+3,k:k+7)
                M1(5,:) = A1(i1+4,k:k+7)
                M1(6,:) = (5*A1(i1+5,k:k+7) + 2*A1(i1+5,k+1:k+8))/7
                M1(7,:) = (A1(i1+6,k:k+7) + A1(i1+6,k+1:k+8))/2
                M1(8,:) = (A1(i2,k:k+7) + 2*A1(i2,k+1:k+8))/3
                C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
                % check +1.5pel slant
                M2(1,:) = (2*A1(i1,k+1:k+8) + A1(i1,k:k+7))/3
                M2(2,:) = (A1(i1+1,k+1:k+8) + A1(i1+1,k:k+7))/2
                M2(3,:) = (2*A1(i1+2,k+1:k+8) + 5*A1(i1+2,k:k+7))/7
                M2(4,:) = A1(i1+3,k:k+7)
                M2(5,:) = A1(i1+4,k:k+7)
                M2(6,:) = (5*A1(i1+5,k:k+7) + 2*A1(i1+5,k-1:k+6))/7
                M2(7,:) = (A1(i1+6,k:k+7) + A1(i1+6,k-1:k+6))/2
                M2(8,:) = (A1(i2,k:k+7) + 2*A1(i2,k-1:k+6))/3
                C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
                [Err,Ind] = min(C)
                C(2) = Err
                if (Ind ==1)
                    M3 = M1
                    % check -2pel slant
                    M1(1,:) = (7*A1(i1,k-1:k+6) + A1(i1,k:k+7))/8
```

```
        M1(2,:) = (5*A1(i1+1,k-1:k+6) + 3*A1(i1+1,k:k+7))/8
        M1(3,:) = (3*A1(i1+2,k-1:k+6) + 5*A1(i1+2,k:k+7))/8
        M1(4,:) = (A1(i1+3,k-1:k+6) + 7*A1(i1+3,k:k+7))/8
        M1(5,:) = (7*A1(i1+4,k:k+7) + A1(i1+4,k+1:k+8))/8
        M1(6,:) = (5*A1(i1+5,k:k+7) + 3*A1(i1+5,k+1:k+8))/8
        M1(7,:) = (3*A1(i1+6,k:k+7) + 5*A1(i1+6,k+1:k+8))/8
        M1(8,:) = (A1(i2,k:k+7) + 7*A1(i2,k+1:k+8))/8
        C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
        % check -1pel slant
        M2(1,:) = (4*A1(i1,k-1:k+6) + 5*A1(i1,k:k+7))/9
        M2(2,:) = (A1(i1+1,k-1:k+6) + 2*A1(i1+1,k:k+7))/3
        M2(3,:) = (A1(i1+2,k-1:k+6) + 4*A1(i1+2,k:k+7))/5
        M2(4,:) = A1(i1+3,k:k+7)
        M2(5,:) = A1(i1+4,k:k+7)
        M2(6,:) = (4*A1(i1+5,k:k+7) + A1(i1+5,k+1:k+8))/5
        M2(7,:) = (2*A1(i1+6,k:k+7) + A1(i1+6,k+1:k+8))/3
        M2(8,:) = (5*A1(i2,k:k+7) + 4*A1(i2,k+1:k+8))/9
        C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
        [Err,Ind] = min(C)
        E84(i,j) = Err
        switch (Ind)
            case (1)
                B84(i1:i2,j1:j2) = M1(:,:)
                S8(i,j) =-2
            case (2)
                B84(i1:i2,j1:j2) = M3(:,:)
                S8(i,j) =-1.5
            case (3)
                B84(i1:i2,j1:j2) = M2(:,:)
                S8(i,j) =-1
        end
    elseif (Ind ==2)
        % check -0.5pel slant
        M1(1,:) = (2*A1(i1,k-1:k+6) + 7*A1(i1,k:k+7))/9
        M1(2,:) = (2*A1(i1+1,k-1:k+6) + 11*A1(i1+1,k:k+7))/13
        M1(3,:) = (A1(i1+2,k-1:k+6) + 10*A1(i1+2,k:k+7))/11
        M1(4,:) = A1(i1+3,k:k+7)
        M1(5,:) = A1(i1+4,k:k+7)
        M1(6,:) = (10*A1(i1+5,k:k+7) + A1(i1+5,k+1:k+8))/11
        M1(7,:) = (11*A1(i1+6,k:k+7) + 2*A1(i1+6,k+1:k+8))/13
        M1(8,:) = (7*A1(i2,k:k+7) + 2*A1(i2,k+1:k+8))/9
        C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
        % check +0.5pel slant
        M2(1,:) = (2*A1(i1,k+1:k+8) + 7*A1(i1,k:k+7))/9
        M2(2,:) = (2*A1(i1+1,k+1:k+8) + 11*A1(i1+1,k:k+7))/13
```

```
        M2(3,:) = (A1(i1+2,k+1:k+8) + 10*A1(i1+2,k:k+7))/11
        M2(4,:) = A1(i1+3,k:k+7)
        M2(5,:) = A1(i1+4,k:k+7)
        M2(6,:) = (10*A1(i1+5,k:k+7) + A1(i1+5,k-1:k+6))/11
        M2(7,:) = (11*A1(i1+6,k:k+7) + 2*A1(i1+6,k-1:k+6))/13
        M2(8,:) = (7*A1(i2,k:k+7) + 2*A1(i2,k-1:k+6))/9
        C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
        [Err,Ind] = min(C)
        E84(i,j) = Err
        switch (Ind)
            case (1)
                B84(i1:i2,j1:j2) = M1(:,:)
                S8(i,j) =-0.5
            case (2)
                B84(i1:i2,j1:j2) = M3(:,:)
                S8(i,j) =0
            case (3)
                B84(i1:i2,j1:j2) = M2(:,:)
                S8(i,j) =0.5
        end
    elseif (Ind ==3)
        M3 = M2
        % check +1pel slant
        M1(1,:) = (4*A1(i1,k+1:k+8) + 5*A1(i1,k:k+7))/9
        M1(2,:) = (A1(i1+1,k+1:k+8) + 2*A1(i1+1,k:k+7))/3
        M1(3,:) = (A1(i1+2,k+1:k+8) + 4*A1(i1+2,k:k+7))/5
        M1(4,:) = A1(i1+3,k:k+7)
        M1(5,:) = A1(i1+4,k:k+7)
        M1(6,:) = (4*A1(i1+5,k:k+7) + A1(i1+5,k-1:k+6))/5
        M1(7,:) = (2*A1(i1+6,k:k+7) + A1(i1+6,k-1:k+6))/3
        M1(8,:) = (5*A1(i2,k:k+7) + 4*A1(i2,k-1:k+6))/9
        C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
        % check +2pel slant
        M2(1,:) = (7*A1(i1,k+1:k+8) + A1(i1,k:k+7))/8
        M2(2,:) = (5*A1(i1+1,k+1:k+8) + 3*A1(i1+1,k:k+7))/8
        M2(3,:) = (3*A1(i1+2,k+1:k+8) + 5*A1(i1+2,k:k+7))/8
        M2(4,:) = (A1(i1+3,k+1:k+8) + 7*A1(i1+3,k:k+7))/8
        M2(5,:) = (7*A1(i1+4,k:k+7) + A1(i1+4,k-1:k+6))/8
        M2(6,:) = (5*A1(i1+5,k:k+7) + 3*A1(i1+5,k-1:k+6))/8
        M2(7,:) = (3*A1(i1+6,k:k+7) + 5*A1(i1+6,k-1:k+6))/8
        M2(8,:) = (A1(i2,k:k+7) + 7*A1(i2,k-1:k+6))/8
        C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
        [Err,Ind] = min(C)
        E84(i,j) = Err
        switch (Ind)
```

```
                case (1)
                    B84(i1:i2,j1:j2) = M1(:,:)
                    S8(i,j) =1
                case (2)
                    B84(i1:i2,j1:j2) = M3(:,:)
                    S8(i,j) =1.5
                case (3)
                    B84(i1:i2,j1:j2) = M2(:,:)
                    S8(i,j) =2
            end
        end
    else
        M3 = A3(i1:i2,k:k+7)
        % check -1.5pel slant
        M1(1,:) = (2*A3(i1,k-1:k+6) + A3(i1,k:k+7))/3
        M1(2,:) = (A3(i1+1,k-1:k+6) + A3(i1+1,k:k+7))/2
        M1(3,:) = (2*A3(i1+2,k-1:k+6) + 5*A3(i1+2,k:k+7))/7
        M1(4,:) = A3(i1+3,k:k+7)
        M1(5,:) = A3(i1+4,k:k+7)
        M1(6,:) = (5*A3(i1+5,k:k+7) + 2*A3(i1+5,k+1:k+8))/7
        M1(7,:) = (A3(i1+6,k:k+7) + A3(i1+6,k+1:k+8))/2
        M1(8,:) = (A3(i2,k:k+7) + 2*A3(i2,k+1:k+8))/3
        C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
        % check +1.5pel slant
        M2(1,:) = (2*A3(i1,k+1:k+8) + A3(i1,k:k+7))/3
        M2(2,:) = (A3(i1+1,k+1:k+8) + A3(i1+1,k:k+7))/2
        M2(3,:) = (2*A3(i1+2,k+1:k+8) + 5*A3(i1+2,k:k+7))/7
        M2(4,:) = A3(i1+3,k:k+7)
        M2(5,:) = A3(i1+4,k:k+7)
        M2(6,:) = (5*A3(i1+5,k:k+7) + 2*A3(i1+5,k-1:k+6))/7
        M2(7,:) = (A3(i1+6,k:k+7) + A3(i1+6,k-1:k+6))/2
        M2(8,:) = (A3(i2,k:k+7) + 2*A3(i2,k-1:k+6))/3
        C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
        [Err,Ind] = min(C)
        C(2) = Err
        if (Ind ==1)
            M3 = M1
            % check -2pel slant
            M1(1,:) = (7*A3(i1,k-1:k+6) + A3(i1,k:k+7))/8
            M1(2,:) = (5*A3(i1+1,k-1:k+6) + 3*A3(i1+1,k:k+7))/8
            M1(3,:) = (3*A3(i1+2,k-1:k+6) + 5*A3(i1+2,k:k+7))/8
            M1(4,:) = (A3(i1+3,k-1:k+6) + 7*A3(i1+3,k:k+7))/8
            M1(5,:) = (7*A3(i1+4,k:k+7) + A3(i1+4,k+1:k+8))/8
            M1(6,:) = (5*A3(i1+5,k:k+7) + 3*A3(i1+5,k+1:k+8))/8
            M1(7,:) = (3*A3(i1+6,k:k+7) + 5*A3(i1+6,k+1:k+8))/8
```

```
        M1(8,:) = (A3(i2,k:k+7) + 7*A3(i2,k+1:k+8))/8
        C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
        % check -1pel slant
        M2(1,:) = (4*A3(i1,k-1:k+6) + 5*A3(i1,k:k+7))/9
        M2(2,:) = (A3(i1+1,k-1:k+6) + 2*A3(i1+1,k:k+7))/3
        M2(3,:) = (A3(i1+2,k-1:k+6) + 4*A3(i1+2,k:k+7))/5
        M2(4,:) = A3(i1+3,k:k+7)
        M2(5,:) = A3(i1+4,k:k+7)
        M2(6,:) = (4*A3(i1+5,k:k+7) + A3(i1+5,k+1:k+8))/5
        M2(7,:) = (2*A3(i1+6,k:k+7) + A3(i1+6,k+1:k+8))/3
        M2(8,:) = (5*A3(i2,k:k+7) + 4*A3(i2,k+1:k+8))/9
        C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
        [Err,Ind] = min(C)
        E84(i,j) = Err
        switch (Ind)
            case (1)
                B84(i1:i2,j1:j2) = M1(:,:)
                S8(i,j) =-2
            case (2)
                B84(i1:i2,j1:j2) = M3(:,:)
                S8(i,j) =-1.5
            case (3)
                B84(i1:i2,j1:j2) = M2(:,:)
                S8(i,j) =-1
        end
    elseif (Ind ==2)
        % check -0.5pel slant
        M1(1,:) = (2*A3(i1,k-1:k+6) + 7*A3(i1,k:k+7))/9
        M1(2,:) = (2*A3(i1+1,k-1:k+6) + 11*A3(i1+1,k:k+7))/13
        M1(3,:) = (A3(i1+2,k-1:k+6) + 10*A3(i1+2,k:k+7))/11
        M1(4,:) = A3(i1+3,k:k+7)
        M1(5,:) = A3(i1+4,k:k+7)
        M1(6,:) = (10*A3(i1+5,k:k+7) + A3(i1+5,k+1:k+8))/11
        M1(7,:) = (11*A3(i1+6,k:k+7) + 2*A3(i1+6,k+1:k+8))/13
        M1(8,:) = (7*A3(i2,k:k+7) + 2*A3(i2,k+1:k+8))/9
        C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
        % check +0.5pel slant
        M2(1,:) = (2*A3(i1,k+1:k+8) + 7*A3(i1,k:k+7))/9
        M2(2,:) = (2*A3(i1+1,k+1:k+8) + 11*A3(i1+1,k:k+7))/13
        M2(3,:) = (A3(i1+2,k+1:k+8) + 10*A3(i1+2,k:k+7))/11
        M2(4,:) = A3(i1+3,k:k+7)
        M2(5,:) = A3(i1+4,k:k+7)
        M2(6,:) = (10*A3(i1+5,k:k+7) + A3(i1+5,k-1:k+6))/11
        M2(7,:) = (11*A3(i1+6,k:k+7) + 2*A3(i1+6,k-1:k+6))/13
        M2(8,:) = (7*A3(i2,k:k+7) + 2*A3(i2,k-1:k+6))/9
```

```
        C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
        [Err,Ind] = min(C)
        E84(i,j) = Err
        switch (Ind)
            case (1)
                B84(i1:i2,j1:j2) = M1(:,:)
                S8(i,j) =-0.5
            case (2)
                B84(i1:i2,j1:j2) = M3(:,:)
                S8(i,j) =0
            case (3)
                B84(i1:i2,j1:j2) = M2(:,:)
                S8(i,j) =0.5
        end
    elseif (Ind ==3)
        M3 = M2
        % check +1pel slant
        M1(1,:) = (4*A3(i1,k+1:k+8) + 5*A3(i1,k:k+7))/9
        M1(2,:) = (A3(i1+1,k+1:k+8) + 2*A3(i1+1,k:k+7))/3
        M1(3,:) = (A3(i1+2,k+1:k+8) + 4*A3(i1+2,k:k+7))/5
        M1(4,:) = A3(i1+3,k:k+7)
        M1(5,:) = A3(i1+4,k:k+7)
        M1(6,:) = (4*A3(i1+5,k:k+7) + A3(i1+5,k-1:k+6))/5
        M1(7,:) = (2*A3(i1+6,k:k+7) + A3(i1+6,k-1:k+6))/3
        M1(8,:) = (5*A3(i2,k:k+7) + 4*A3(i2,k-1:k+6))/9
        C(1)= sum(sum(abs(M1(:,:)-A2(i1:i2,j1:j2))))
        % check +2pel slant
        M2(1,:) = (7*A3(i1,k+1:k+8) + A3(i1,k:k+7))/8
        M2(2,:) = (5*A3(i1+1,k+1:k+8) + 3*A3(i1+1,k:k+7))/8
        M2(3,:) = (3*A3(i1+2,k+1:k+8) + 5*A3(i1+2,k:k+7))/8
        M2(4,:) = (A3(i1+3,k+1:k+8) + 7*A3(i1+3,k:k+7))/8
        M2(5,:) = (7*A3(i1+4,k:k+7) + A3(i1+4,k-1:k+6))/8
        M2(6,:) = (5*A3(i1+5,k:k+7) + 3*A3(i1+5,k-1:k+6))/8
        M2(7,:) = (3*A3(i1+6,k:k+7) + 5*A3(i1+6,k-1:k+6))/8
        M2(8,:) = (A3(i2,k:k+7) + 7*A3(i2,k-1:k+6))/8
        C(3)= sum(sum(abs(M2(:,:)-A2(i1:i2,j1:j2))))
        [Err,Ind] = min(C)
        E84(i,j) = Err
        switch (Ind)
            case (1)
                B84(i1:i2,j1:j2) = M1(:,:)
                S8(i,j) =1
            case (2)
                B84(i1:i2,j1:j2) = M3(:,:)
                S8(i,j) =1.5
```

```
                                case (3)
                                    B84(i1:i2,j1:j2) = M2(:,:)
                                    S8(i,j) =2
                            end
                        end
                    end
                % end
            end
end
imwrite(uint8(B84),'B84.bmp','bmp')
imwrite(uint8(64*(S8+2)),'Slant8.bmp','bmp')
imwrite(uint8(255*(1-E84/max(max(E84)))),'E84posi.bmp','bmp')
E84t = sum(sum(E84))
save('dispdat81.mat')
% stop here
```

３．多視点映像の視差補償予測符号化プログラム（C++）

```
// dspred_main
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct
{
   int                  width;   int                  height;
   unsigned char*   data;
} ColorComponent;
typedef struct
{
   ColorComponent lum;   ColorComponent cb;   ColorComponent cr;
} YuvFrame;
void createColorComponent( ColorComponent* cc )
{
   if( ! ( cc->data = new unsigned char[cc->width * cc->height]))
   {
      fprintf(stderr, "¥nERROR: memory allocation failed!¥n¥n");
      exit(1);
   }
}
void deleteColorComponent( ColorComponent* cc )
{
   delete[] cc->data;
```

```
  cc->data = NULL;
}
void createFrame( YuvFrame* f, int width, int height )
{
  f->lum.width = width;      f->lum.height   = height;       createColorComponent( &f->lum );
  f->cb .width = width/2;   f->cb .height   = height/2;    createColorComponent( &f->cb   );
  f->cr .width = width/2;   f->cr .height   = height/2;    createColorComponent( &f->cr   );
}
void deleteFrame( YuvFrame* f )
{
  deleteColorComponent( &f->lum );   deleteColorComponent( &f->cb   );
  deleteColorComponent( &f->cr   );
}
void readColorComponent( ColorComponent* cc, FILE* file )
{
  unsigned int size     = cc->width*cc->height;   unsigned int rsize;
  rsize = fread( cc->data, sizeof(unsigned char), size, file );
  if( size != rsize )
  {
    fprintf(stderr, "¥nERROR: while reading from input file!¥n¥n");
    exit(1);
  }
}
void writeColorComponent( ColorComponent* cc, FILE* file   )
{
  int outwidth   = cc->width;   int outheight = cc->height;   int wsize;
  for( int i = 0; i < outheight; i++ )
  {
    wsize = fwrite( cc->data+i*cc->width, sizeof(unsigned char), outwidth, file );
    if( outwidth != wsize )
    {
      fprintf(stderr, "¥nERROR: while writing to output file!¥n¥n");
      exit(1);
    }
  }
}
double psnr( ColorComponent& rec, ColorComponent& org )
{
  unsigned char*   pOrg   = org.data;      unsigned char*   pRec   = rec.data;
  double               ssd    = 0;                int                   diff;
  for   ( int r = 0; r < rec.height; r++ )
  {
    for( int c = 0; c < rec.width;   c++ )
    {
      diff   = pRec[c] - pOrg[c];
```

```
            ssd    += (double)( diff * diff );
        }
        pRec     += rec.width;       pOrg     += org.width;
    }
    if( ssd == 0.0 )
    {
        return 99.99;
    }
    return ( 10.0 * log10( (double)rec.width * (double)rec.height * 65025.0 / ssd ) );
}
void getPSNR( double& psnrY, double& psnrU, double& psnrV, YuvFrame& rcFrameOrg,
YuvFrame& rcFrameRec )
{
    psnrY = psnr( rcFrameRec.lum, rcFrameOrg.lum );
    psnrU = psnr( rcFrameRec.cb,    rcFrameOrg.cb );
    psnrV = psnr( rcFrameRec.cr,    rcFrameOrg.cr );
}
void hdet( int preb[128], int tgt[64], int dspv, int min, int hrslt[112], int& dspvh)
{
            int can[120];           int diff1=0, diff2=0;
            for ( int i=0; i<8; i++ )
            {
                        int ii = 16*i;              int ij = 15*i;
                        for ( int j=0; j<15; j++ )
                        {
                                    int jj = j + ii;
                                    can[j+ij] = (preb[jj] + preb[1+jj])/2;
                        }
            }
            for ( int i=0; i<8; i++ )
            {
                        int ii = 8*i;               int ij = 15*i;
                        for ( int j=0; j<8; j++ )
                        {
                                    diff1 += abs( can[3+j+ij] - tgt[j+ii] );
                                    diff2 += abs( can[4+j+ij] - tgt[j+ii] );
                        }
            }
            int offset = 0;
            if ( diff1 <= diff2 ) // (-1/2) <= (+1/2)
            {
                        if ( diff1 < min ) // (-1/2) < (0), (+1/2)
                        {
                   for ( int i=0; i<8; i++ )
                                    {
```

```
                                int ii = 15*i;          int ij = 14*i;
                                for ( int j=0; j<14; j++ )
                                {
                                        hrslt[j+ij] = can[j+ii];
                                }
                        }
                        dspvh = 2*dspv - 1;
                }
                else // (0) <= (-1/2) <= (+1/2)
                {
    for ( int i=0; i<8; i++ )
                        {
                                int ii = 1+16*i;        int ij = 14*i;
                                for ( int j=0; j<14; j++ )
                                {
                                        hrslt[j+ij] = preb[j+ii];
                                }
                        }
                        dspvh = 2*dspv;
                }
}
else // (+1/2) < (-1/2)
{
                if ( diff2 < min ) // (+1/2) < (0), (-1/2)
                {
    for ( int i=0; i<8; i++ )
                        {
                                int ii = 15*i+1;        int ij = 14*i;
                                for ( int j=0; j<14; j++ )
                                {
                                        hrslt[j+ij] = can[j+ii];
                                }
                        }
                        dspvh = 2*dspv + 1;
                }
                else // (0) <= (+1/2) < (-1/2)
                {
  for ( int i=0; i<8; i++ )
                        {
                                int ii = 1+16*i;        int ij = 14*i;
                                for ( int j=0; j<14; j++ )
                                {
                                        hrslt[j+ij] = preb[j+ii];
                                }
                        }
```

```
                        dspvh = 2*dspv;
                }
        }
}
// check width = 9, 9.5, 10 for wdet (2nd-Right Stage)
void wchk10(int cand[112], int tgt[64], int wgt, int rslt[80], int diff3, int& kw)
{
        int can1[80], can2[80];         int diff1=0, diff2=0;
        for ( int i=0; i<8; i++ )
        {
                int ij = 10*i;          int ii = 14*i;
                // w=9
can1[ij]  =(5*cand[1+ii]+ 4*cand[2+ii]+4)/9, can1[9+ij]=(5*cand[12+ii]+ 4*cand[11+ii]+4)/9;
can1[1+ij]=(4*cand[2+ii]+ 5*cand[3+ii]+4)/9, can1[8+ij]=(4*cand[11+ii]+ 5*cand[10+ii]+4)/9;
can1[2+ij]=(5*cand[3+ii]+11*cand[4+ii]+8)/16,can1[7+ij]=(5*cand[10+ii]+11*cand[ 9+ii]+8)/16;
can1[3+ij]=(2*cand[4+ii]+ 9*cand[5+ii]+5)/11,can1[6+ij]=(2*cand[ 9+ii]+ 9*cand[ 8+ii]+5)/11;
can1[4+ij]=    cand[6+ii],                  can1[5+ij]=    cand[ 7+ii];
                // w=10
can2[ij]  =(  cand[  ii]+ 7*cand[1+ii]+4)/8, can2[9+ij]=(  cand[13+ii]+7*cand[12+ii]+4)/8;
can2[1+ij]=(7*cand[2+ii]+   cand[3+ii]+4)/8, can2[8+ij]=(7*cand[11+ii]+  cand[10+ii]+4)/8;
can2[2+ij]=(5*cand[3+ii]+ 3*cand[4+ii]+4)/8, can2[7+ij]=(5*cand[10+ii]+3*cand[ 9+ii]+4)/8;
can2[3+ij]=(3*cand[4+ii]+ 5*cand[5+ii]+4)/8, can2[6+ij]=(3*cand[ 9+ii]+5*cand[ 8+ii]+4)/8;
can2[4+ij]=(  cand[5+ii]+ 7*cand[6+ii]+4)/8, can2[5+ij]=(  cand[ 8+ii]+7*cand[ 7+ii]+4)/8;
        }
        for (int i=0; i<8; i++ )
        {
                int i8 = 8*i;           int ij = 1+10*i;
                for ( int j=0; j<8; j++ )
                {
                        diff1 += abs(can1[j+ij] - tgt[j+i8]); // check w=9 (kw=2)
                        diff2 += abs(can2[j+ij] - tgt[j+i8]); // check w=10(kw=4)
                }
        }
// weight
        diff1 = diff1 + 40*abs( wgt - 3*2 );
        diff2 = diff2 + 40*abs( wgt - 3*4 );
        if ( diff1 < diff3 ) // (w=9 < w=9.5)
        {
                if ( diff1 < diff2 ) // (w=9 < w=10, w=9.5)
                {
                        for ( int i=0; i<8; i++ )
                        {
                                int ij = 10*i;
                                for (int j=0; j<10; j++)
                                {
```

```
                                        rslt[j+ij]= can1[j+ij]; //(w=9)
                                }
                        }
                        kw = 2; // (w=9)
                }
                else // (diff2 (w=10) <= diff1 (w=9) < diff3 (w=9.5))
                {
                        for ( int i=0; i<8; i++ )
                        {
                                int ij = 10*i;
                                for (int j=0; j<10; j++)
                                {
                                        rslt[j+ij]= can2[j+ij]; //(w=10)
                                }
                        }
                        kw = 4; // (w=10)
                }
        }
        else // (diff3 (w=9.5) < diff1 (w=9))
        {
                if ( diff3 < diff2 ) // (w=9.5 < w=10, w=9)
                {
                        // rslt[ ] = (w=9.5), // kw = 3
                }
                else // (diff2 (w=10) <= diff3 (w=9.5) < diff1 (w=9))
                                {
                        for ( int i=0; i<8; i++ )
                        {
                                int ij = 10*i;
                                for (int j=0; j<10; j++)
                                {
                                        rslt[j+ij]= can2[j+ij]; // (w=10)
                                }
                        }
                        kw = 4; // (w=10)
                }
        }
}
// width search
void wdet( int cand[112], int tgt[64], int wgt, int rslt[80], int& kw )
{
        int can1[80], can2[80];         int diff1=0,    diff2=0,    diff3=0;
        // check w=6.5(can1), 8(cand), 9.5(can2): 1st Stage (center 8)
        for ( int i=0; i<8; i++ )
        {
```

```
                int ij = 10*i;          int ii = 14*i;
                // w=6.5
can1[ij]  =(  cand[2+ii] +5*cand[3+ii]+3)/6, can1[9+ij]=(  cand[11+ii]+ 5*cand[10+ii]+3)/6;
can1[1+ij]=(  cand[3+ii] +2*cand[4+ii]+1)/3, can1[8+ij]=(  cand[10+ii]+ 2*cand[ 9+ii]+1)/3;
can1[2+ij]=(9*cand[4+ii]+10*cand[5+ii]+9)/19,can1[7+ij]=( 9*cand[9+ii]+10*cand[ 8+ii]+9)/19;
can1[3+ij]=(5*cand[5+ii] +2*cand[6+ii]+3)/7, can1[6+ij]=( 5*cand[8+ii]+ 2*cand[ 7+ii]+3)/7;
can1[4+ij]=(10*cand[6+ii]+  cand[7+ii]+5)/11,can1[5+ij]=(10*cand[7+ii]+   cand[ 6+ii]+5)/11;
                // w=9.5
can2[ij]  =(5*cand[1+ii]+  cand[2+ii]+3)/6,        can2[9+ij]=(5*cand[12+ii]  +cand[11+ii]+3)/6;
can2[1+ij]=(2*cand[2+ii]+  cand[3+ii]+1)/3,        can2[8+ij]=(2*cand[11+ii]  +cand[10+ii]+1)/3;
can2[2+ij]=(8*cand[3+ii]+7*cand[4+ii]+7)/15,can2[7+ij]=(8*cand[10+ii]+7*cand[9+ii] +7)/15;
can2[3+ij]=(2*cand[4+ii]+5*cand[5+ii]+3)/7,        can2[6+ij]=(2*cand[ 9+ii]+5*cand[8+ii] +3)/7;
can2[4+ij]=(  cand[5+ii]+9*cand[6+ii]+5)/10,can2[5+ij]=(  cand[ 8+ii]+9*cand[7+ii] +5)/10;
        }
        for (int i=0; i<8; i++ )
        {
                int i8 = 8*i;           int ij = 1+10*i;        int ii = 3+14*i;
                for ( int j=0; j<8; j++ )
                {
            diff1 += abs(can1[j+ij] - tgt[j+i8]); //check w=6.5 (Kw=-3)
                        diff2 += abs(can2[j+ij] - tgt[j+i8]); //check w=9.5 (kw= 3)
                        diff3 += abs(cand[j+ii] - tgt[j+i8]); //check w=8     (kw= 0)
                }
        }
// weight
        diff1 = diff1 + 40*abs( wgt + 3*3 );
        diff2 = diff2 + 40*abs( wgt - 3*3 );
        diff3 = diff3 + 40*abs( wgt );
        if ( diff1 < diff3) // (w=6.5 < w=8)
        {
                if (diff1 < diff2 ) // (w=6.5 < w=9.5, w=8)
                {
                        for ( int i=0; i<8; i++ )
                        {
                                int ij = 10*i;
                                for (int j=0; j<10; j++)
                                {
                                        rslt[j+ij]= can1[j+ij]; //(w=6.5)
                                }
                        }
                        kw = -3; // (w=6.5)
                        diff3 = diff1;
                        // check w=5/6(can1), 6.5(rslt), 7(can2): 2nd-Left Stage (center 6.5)
                        for ( int i=0; i<8; i++ )
                        {
```

```
                                int ij = 10*i;          int ii = 14*i;
                                // w=5
// can1[ij]   =(5*cand[3+ii]+11*cand[4+ii]+8)/16,can1[9+ij]=(5*cand[10+ii]+11*cand[9+ii]+8)/16;
// can1[1+ij]=(9*cand[4+ii]+5*cand[5+ii]+7)/14, can1[8+ij]=(9*cand[9+ii]   +5*cand[8+ii]+7)/14;
// can1[2+ij]=( cand[4+ii]+15*cand[5+ii]+8)/16, can1[7+ij]=(   cand[9+ii] +15*cand[8+ii]+8)/16;
// can1[3+ij]=(7*cand[5+ii]+9*cand[6+ii]+8)/16, can1[6+ij]=(7*cand[8+ii]   +9*cand[7+ii]+8)/16;
// can1[4+ij]=(13*cand[6+ii]+3*cand[7+ii]+8)/16,can1[5+ij]=(13*cand[7+ii] +3*cand[6+ii]+8)/16;
                                // w=6
   can1[ij]   =(5*cand[3+ii]+   cand[4+ii]+3)/6,can1[9+ij]=(5*cand[10+ii]+   cand[9+ii]+3)/6;
   can1[1+ij]=(   cand[3+ii]+7*cand[4+ii]+4)/8,can1[8+ij]=(   cand[10+ii]+7*cand[9+ii]+4)/8;
   can1[2+ij]=(3*cand[4+ii]+5*cand[5+ii]+4)/8,can1[7+ij]=(3*cand[ 9+ii]+5*cand[8+ii]+4)/8;
   can1[3+ij]=(5*cand[5+ii]+3*cand[6+ii]+4)/8,can1[6+ij]=(5*cand[ 8+ii]+3*cand[7+ii]+4)/8;
   can1[4+ij]=(7*cand[6+ii]+   cand[7+ii]+4)/8,can1[5+ij]=(7*cand[ 7+ii]+   cand[6+ii]+4)/8;
                                // w=7
can2[ij]   =(5*cand[2+ii]+4*cand[3+ii]+4)/9,can2[9+ij]=(5*cand[11+ii]+4*cand[10+ii]+4)/9;
can2[1+ij]=(4*cand[3+ii]+3*cand[4+ii]+3)/7,can2[8+ij]=(4*cand[10+ii]+3*cand[ 9+ii]+3)/7;
can2[2+ij]=(2*cand[4+ii]+   cand[5+ii]+1)/3,can2[7+ij]=(2*cand[ 9+ii]+   cand[ 8+ii]+1)/3;
can2[3+ij]=(4*cand[5+ii]+   cand[6+ii]+2)/5,can2[6+ij]=(4*cand[ 8+ii]+   cand[ 7+ii]+2)/5;
can2[4+ij]=    cand[6+ii],                  can2[5+ij]=    cand[ 7+ii];
                        }
                        diff1 = 0;              diff2 = 0;
                        for (int i=0; i<8; i++ )
                        {
                                int i8 = 8*i;           int ij = 1+10*i;
                                for ( int j=0; j<8; j++ )
                                {
diff1 += abs(can1[j+ij] - tgt[j+i8]); // check W=6/5 (kw=-4/-5)
diff2 += abs(can2[j+ij] - tgt[j+i8]); // check W=7 (kw=-2)
                                }
                        }
// weight
//                      diff1 = diff1 + 40*abs( wgt + 3*5 );   // w= 5 (kw=-5)
                        diff1 = diff1 + 40*abs( wgt + 3*4 );   // w= 6 (kw=-4)
                        diff2 = diff2 + 40*abs( wgt + 3*2 );   // w= 7 (kw=-2)
                        if ( diff1 < diff3 ) // (w=6/5 < w=6.5)
                        {
                                if ( diff1 < diff2 ) // (w=6/5 < w=7, w=6.5)
                                {
                                        for ( int i=0; i<8; i++ )
                                        {
                                                int ij = 10*i;
                                                for (int j=0; j<10; j++)
                                                {
                                                        rslt[j+ij]=     can1[j+ij];
//(w=6/5)
```

```
                                        }
                                }
//                              kw = -5; // (w=5)
                                kw = -4; // (w=6)

// no check for kw =4,5,6
                        if ( kw != kw )
                        {
                                // check kw = 4,5,6
                                diff3 = diff1;
                                for ( int i=0; i<8; i++ )
                                {
                                        int ij = 10*i;          int ii = 14*i;
                                        // w=4
can1[ij]  =(3*cand[4+ii]+  cand[5+ii]+2)/4,can1[9+ij]=(3*cand[9+ii]+  cand[8+ii]+2)/4;
can1[1+ij]=(  cand[4+ii]+3*cand[5+ii]+2)/4,can1[8+ij]=(  cand[9+ii]+3*cand[8+ii]+2)/4;
can1[2+ij]=(3*cand[5+ii]+  cand[6+ii]+2)/4,can1[7+ij]=(3*cand[8+ii]  +cand[7+ii]+2)/4;
can1[3+ij]=(  cand[5+ii]+3*cand[6+ii]+2)/4,can1[6+ij]=(  cand[8+ii]+3*cand[7+ii]+2)/4;
can1[4+ij]=(3*cand[6+ii]+  cand[7+ii]+2)/4,can1[5+ij]=(3*cand[7+ii]  +cand[6+ii]+2)/4;
                                        // w=6
can2[ij]  =(5*cand[3+ii]+  cand[4+ii]+3)/6,can2[9+ij]=(5*cand[10+ii]+  cand[9+ii]+3)/6;
can2[1+ij]=(  cand[3+ii]+7*cand[4+ii]+4)/8,can2[8+ij]=(  cand[10+ii]+7*cand[9+ii]+4)/8;
can2[2+ij]=(3*cand[4+ii]+5*cand[5+ii]+4)/8,can2[7+ij]=(3*cand[ 9+ii]+5*cand[8+ii]+4)/8;
can2[3+ij]=(5*cand[5+ii]+3*cand[6+ii]+4)/8,can2[6+ij]=(5*cand[ 8+ii]+3*cand[7+ii]+4)/8;
can2[4+ij]=(7*cand[6+ii]+  cand[7+ii]+4)/8,can2[5+ij]=(7*cand[ 7+ii]+  cand[6+ii]+4)/8;
                                }
                                diff1 = 0;              diff2 = 0;
                                for (int i=0; i<8; i++ )
                                {
                                        int i8 = 8*i;           int ij = 1+10*i;
                                        for ( int j=0; j<8; j++ )
                                        {
diff1 += abs(can1[j+ij] - tgt[j+i8]); // check W=4 (kw=-6)
diff2 += abs(can2[j+ij] - tgt[j+i8]); // check W=6 (kw=-4)
                                        }
                                }
// weight
                                diff1 = diff1 + 40*abs( wgt + 3*6 );
                                diff2 = diff2 + 40*abs( wgt + 3*4 );
                                if ( diff1 < diff3 ) // (w=4 < w=5)
                                {
                                  if ( diff1 < diff2 ) // (w=4 < w=6, w=5)
                                  {
                                        for ( int i=0; i<8; i++ )
                                        {
```

```
                                        int ij = 10*i;
                                        for (int j=0; j<10; j++)
                                        {
                                                rslt[j+ij]= can1[j+ij]; //(w=4)
                                        }
                                    }
                                    kw = -6; // (w=4)
                                }
                                else    // (w=6 < w=4 < w=5)
                                {
                                    for ( int i=0; i<8; i++ )
                                    {
                                        int ij = 10*i;
                                        for (int j=0; j<10; j++)
                                        {
                                                rslt[j+ij]= can2[j+ij]; //(w=6)
                                        }
                                    }
                                    kw = -4; // (w=6)
                                }
                            }
                            else    // (w=5 < w=4)
                            {
                                if ( diff3 < diff2 ) // (w=5 < w=6, w=4)
                                {
                                    // rslt[j+ij]= (w=4), kw = -5 (w=5)
                                }
                                else    // (w=6 < w=5 < w=4)
                                {
                                    for ( int i=0; i<8; i++ )
                                    {
                                        int ij = 10*i;
                                        for (int j=0; j<10; j++)
                                        {
                                                rslt[j+ij]= can2[j+ij]; //(w=6)
                                        }
                                    }
                                    kw = -4; // (w=6)
                                }
                            }
// end of nocheck for kw=4,5,6
                        }
                    }
                    else // (diff2 (w=7) <= diff1 (w=6/5) < diff3 (w=6.5))
                    {
```

```
                                        for ( int i=0; i<8; i++ )
                                        {
                                                int ij = 10*i;
                                                for (int j=0; j<10; j++)
                                                {
                                                        rslt[j+ij]= can2[j+ij]; //(w=7)
                                                }
                                        }
                                        kw = -2; // (w=7)
                                }
                        }
                        else // ( diff3(w=6.5) <= diff1(w=6/5) )
                        {
                                if ( diff3 <= diff2 ) // (w=6.5 <= w=7, w=6/5)
                                {
                                        // rslt[ ] = (w=6.5), // kw = -3 (w=6.5)
                                }
                                else // (diff2 (w=7) < diff3 (w=6.5) <= diff1 (w=6))
                                {
                                        for ( int i=0; i<8; i++ )
                                        {
                                                int ij = 10*i;
                                                for (int j=0; j<10; j++)
                                                {
                                                        rslt[j+ij]= can2[j+ij]; //(w=7)
                                                }
                                        }
                                        kw = -2; // (w=7)
                                }
                        }
                }
                else // (diff2 (w=9.5) <= diff1 (w=6.5) < diff3 (w=8))
                {
                        for ( int i=0; i<8; i++ )
                        {
                                int ij = 10*i;
                                for (int j=0; j<10; j++)
                                {
                                        rslt[j+ij]= can2[j+ij]; //(w=9.5)
                                }
                        }
                        kw = 3; // (w=9.5) kw=3
                        diff3 = diff2; // (w=9.5)
                        // check w=9(can1), 9.5(rslt) ,10(can2): 2nd-Right Stage (center 9.5)
                        wchk10(cand, tgt, wgt, rslt, diff3, kw);
```

```
                }
            }
            else // (diff3(w=8) <= diff1(W=6.5))
            {
               if ( diff3 <= diff2 ) // (w=8 <= w=9.5 <= w=6.5)
               {
                        // check w=7.5(can1), 8(can3), 8.5(can2): 2nd-Center Stage (center 8)
                        for ( int i=0; i<8; i++ )
                        {
                           int ij = 10*i;    int ii = 14*i;
                           // w=7.5
   can1[ij]   =( 4*cand[2+ii]+   cand[3+ii]+2)/5, can1[9+ij]=( 4*cand[11+ii]+ cand[10+ii]+2)/5;
   can1[1+ij]=(11*cand[3+ii]+3*cand[4+ii]+7)/14,can1[8+ij]=(11*cand[10+ii]+3*cand[9+ii]+7)/14;
   can1[2+ij]=( 5*cand[4+ii]+   cand[5+ii]+3)/6, can1[7+ij]=( 5*cand[ 9+ii]+   cand[8+ii]+3)/6;
   can1[3+ij]=(10*cand[5+ii]+   cand[6+ii]+5)/11,can1[6+ij]=(10*cand[ 8+ii]+   cand[7+ii]+5)/11;
   can1[4+ij]=      cand[6+ii],                                          can1[5+ij]=      cand[ 7+ii];
                           // w=8.5
   can2[ij]   =(2*cand[1+ii]+5*cand[2+ii]+3)/7, can2[9+ij]=(2*cand[12+ii]+5*cand[11+ii]+3)/7;
   can2[1+ij]=(2*cand[2+ii]+7*cand[3+ii]+4)/9, can2[8+ij]=(2*cand[11+ii]+7*cand[10+ii]+4)/9;
   can2[2+ij]=(   cand[3+ii]+5*cand[4+ii]+3)/6, can2[7+ij]=(   cand[10+ii]+5*cand[ 9+ii]+3)/6;
   can2[3+ij]=(   cand[4+ii]+9*cand[5+ii]+5)/10,can2[6+ij]=(   cand[ 9+ii]+9*cand[ 8+ii]+5)/10;
   can2[4+ij]=    cand[6+ii],                                          can2[5+ij]=    cand[ 7+ii];
                        }
                                 diff1 = 0;              diff2 = 0;
                                 for (int i=0; i<8; i++ )
                                 {
                                          int i8 = 8*i;          int ij = 1+10*i;
                                          for ( int j=0; j<8; j++ )
                                          {
                                          diff1 += abs(can1[j+ij] - tgt[j+i8]); // check w=7.5 (kw=-1)
                                          diff2 += abs(can2[j+ij] - tgt[j+i8]); // check w=8.5 (kw= 1)
                                          }
                                 }
// weight
                                 diff1 = diff1 + 40*abs( wgt + 3 );
                                 diff2 = diff2 + 40*abs( wgt - 3 );
                                 if ( diff1 < diff3 ) // (w=7.5 < w=8)
                                 {
                                          if ( diff1 < diff2 ) // (w=7.5 < w=8.5, w=8)
                                          {
                                                   for ( int i=0; i<8; i++ )
                                                   {
                                                      int ij = 10*i;
                                                      for (int j=0; j<10; j++)
                                                      {
```

```
                                        rslt[j+ij]= can1[j+ij]; //(w=7.5)
                                    }
                                }
                                kw = -1; // (w=7.5)
                        }
                        else // (diff2(w=8.5) <= diff1(w=7.5) < diff3(w=8))
                        {
                                for ( int i=0; i<8; i++ )
                                {
                                    int ij = 10*i;
                                    for (int j=0; j<10; j++)
                                    {
                                        rslt[j+ij]= can2[j+ij]; //(w=8.5)
                                    }
                                }
                                kw = 1; // (w=8.5)
                        }
                }
                else // (diff3 (w=8) <= diff1 (w=7.5)
                {
                        if ( diff3 < diff2 ) // (w=8 < w=8.5, w=7.5)
                        {
                                for ( int i=0; i<8; i++ )
                                {
                                        int ij = 10*i;          int ii = 2+14*i;
                                        for (int j=0; j<10; j++)
                                        {
                                                rslt[j+ij]= cand[j+ii]; //(w=8)
                                        }
                                }
                                kw = 0; // w=8
                        }
                        else // ( diff2 (w=8.5) <= diff3 (w=8) <= diff1 (w=7.5))
                        {
                                for ( int i=0; i<8; i++ )
                                {
                                        int ij = 10*i;
                                        for (int j=0; j<10; j++)
                                        {
                                            rslt[j+ij]= can2[j+ij]; //(w=8.5)
                                        }
                                }
                                kw = 1; // w=8.5
                        }
                }
```

```
                }
                else // (diff2 (w=9.5) <= diff3 (w=8) <= diff1 (w=6.5))
                {
                        for ( int i=0; i<8; i++ )
                        {
                                int ij = 10*i;
                                for (int j=0; j<10; j++)
                                {
                                        rslt[j+ij]= can2[j+ij]; //(w=9.5)
                                }
                        }
                        kw = 3; // (w=9.5)
                        diff3 = diff2; // (w=9.5)
                        // check w=9(can1), 9.5(rslt) ,10(can2): 2nd-Right Stage (center 9.5)
                        wchk10(cand, tgt, wgt, rslt, diff3, kw);
                }
        }
}
// check slant = -0.5, 0, 0.5
void schk0( int cand[80], int tgt[64], int wgt, int diff3, int rslt[64], int& kslnt )
{
        int can1[64],   can2[64];       int diff1 = 0,   diff2 = 0;
        for ( int j=0; j<8; j++ )
        {
                int j1= 1+j, j2 = 2+j;
                // s=-0.5
can1[j]   =(  cand[j]   +3*cand[j1]   +2)/4, can1[j+56]=(3*cand[j1+70]+  cand[j2+70]+2)/4;
can1[j+ 8]=(2*cand[j+10]+9*cand[j1+10]+5)/11,can1[j+48]=(9*cand[j1+60]+2*cand[j2+60]+5)/11;
can1[j+16]=(  cand[j+20]+8*cand[j1+20]+4)/9, can1[j+40]=(8*cand[j1+50]+  cand[j2+50]+4)/9;
can1[j+24]=   cand[j1+30],                   can1[j+32]=   cand[j1+40];
                // s=0.5
can2[j]   =(3*cand[j1]   +  cand[j2]   +2)/4, can2[j+56]=(  cand[j+70]+3*cand[j1+70]+2)/4;
can2[j+ 8]=(9*cand[j1+10]+2*cand[j2+10]+5)/11,can2[j+48]=(2*cand[j+60]+9*cand[j1+60]+5)/11;
can2[j+16]=(8*cand[j1+20]+  cand[j2+20]+4)/9, can2[j+40]=(  cand[j+50]+8*cand[j1+50]+4)/9;
can2[j+24]=   cand[j1+30],                   can2[j+32]=   cand[j1+40];
        }
        for ( int i=0; i<8; i++ )
        {
                int ii = 8*i;
                for ( int j=0; j<8; j++ )
                {
                        int jj = j+ii;
                        diff1 += abs( can1[jj] - tgt[jj] );   // s=-0.5 (ks=-1)
                        diff2 += abs( can2[jj] - tgt[jj] );   // s= 0.5 (ks= 1)
                }
```

```
		}
// weight
		diff1 = diff1 + 40*abs( wgt + 3 );
		diff2 = diff2 + 40*abs( wgt - 3 );
		if ( diff1 < diff2 )	// s=-0.5 < s=0.5
		{
			if ( diff1 < diff3 )	// s=-0.5 < s=0, s=0.5
			{
				for ( int i=0; i<8; i++ )
				{
					int ii = 8*i;
					for ( int j=0; j<8; j++ )
					{
						int jj = j+ii;
						rslt[jj] = can1[jj];	// s=-0.5
					}
				}
				kslnt = -1;
			}
			else	// s=0 <= s=-0.5 < s=0.5
			{
				// rslt[ij] = (s=0), kslnt = 0
			}
		}
		else	// s=0.5 <= s=-0.5
		{
			if ( diff2 < diff3 )	// s=0.5 < s=0 , s=-0.5
			{
				for ( int i=0; i<8; i++ )
				{
					int ii = 8*i;
					for ( int j=0; j<8; j++ )
					{
						int jj = j+ii;
						rslt[jj] = can2[jj];	// s=0.5
					}
				}
				kslnt = 1;
			}
			else	// s=0 <= s=0.5 < s=-0.5
			{
				// rslt[ij] = (s=0), kslt = 0
			}
		}
}
```

```
// slant search
void sdet( int cand[80], int tgt[64], int wgt, int rslt[64], int& kslnt )
{
        int can1[64], can2[64];         int diff1 = 0, diff2 = 0, diff3 = 0;
        // check s= -1.5, 0, +1.5
        for ( int j=0; j<8; j++ )
        {
                int j1= 1+j,   j2 = 2+j;
                // s=-1.5
can1[j]   =(3*cand[j]  +   cand[j1]   +2)/4, can1[j+56]=(  cand[j1+70]+3*cand[j2+70]+2)/4;
can1[j+ 8]=(8*cand[j+10]+7*cand[j1+10]+7)/15,can1[j+48]=(7*cand[j1+60]+8*cand[j2+60]+7)/15;
can1[j+16]=(  cand[j+20]+2*cand[j1+20]+1)/3, can1[j+40]=(2*cand[j1+50]+  cand[j2+50]+1)/3;
can1[j+24]=(  cand[j+30]+8*cand[j1+30]+4)/9, can1[j+32]=(8*cand[j1+40]+  cand[j2+40]+4)/9;
                // s=1.5
can2[j]   =(  cand[j1]  + 3*cand[j2]   +2)/4, can2[j+56]=(3*cand[j+70]+  cand[j1+70]+2)/4;
can2[j+ 8]=(7*cand[j1+10]+8*cand[j2+10]+7)/15,can2[j+48]=(8*cand[j+60]+7*cand[j1+60]+7)/15;
can2[j+16]=(2*cand[j1+20]+  cand[j2+20]+1)/3, can2[j+40]=(  cand[j+50]+2*cand[j1+50]+1)/3;
can2[j+24]=(8*cand[j1+30]+  cand[j2+30]+4)/9, can2[j+32]=(  cand[j+40]+8*cand[j1+40]+4)/9;
        }
        for ( int i=0; i<8; i++ )
        {
                int ii = 8*i;           int ij = 10*i+1;
                for ( int j=0; j<8; j++ )
                {
                        int jj = j+ii;
                        diff1 += abs( can1[jj] - tgt[jj] );   // s=-1.5 (ks=-3)
                        diff2 += abs( can2[jj] - tgt[jj] );   // s=+1.5 (ks= 3)
                        diff3 += abs( cand[j+ij] - tgt[jj]); // s=0      (ks= 0)
                }
        }
// weight
        diff1 += 40*abs( wgt + 3*3 );
        diff2 += 40*abs( wgt - 3*3 );
        diff3 += 40*abs( wgt );
        if ( diff1 < diff2 ) // s=-1.5 < s=1.5
        {
                if ( diff1 < diff3 ) // s=-1.5 < s=0, s=1.5
                {
                        for ( int i=0; i<8; i++ )
                        {
                                int ii = 8*i;
                                for ( int j=0; j<8; j++ )
                                {
                                        int jj = j+ii;
                                        rslt[jj] = can1[jj];   // s=-1.5
```

```
                    }
                }
                kslnt = -3;
                diff3 = diff1;
                //check s= -2, -1.5, -1
                for ( int j=0; j<8; j++ )
                {
                  int j1= 1+j, j2 = 2+j;
                  // s=-2
can1[j]  =  cand[j],                    can1[j+56]=  cand[j2+70];
can1[j+ 8]=(5*cand[j+10]+2*cand[j1+10]+3)/7,can1[j+48]=(2*cand[j1+60]+5*cand[j2+60]+3)/7;
can1[j+16]=(3*cand[j+20]+4*cand[j1+20]+3)/7,can1[j+40]=(4*cand[j1+50]+3*cand[j2+50]+3)/7;
can1[j+24]=(  cand[j+30]+6*cand[j1+30]+3)/7,can1[j+32]=(6*cand[j1+40]+  cand[j2+40]+3)/7;
                  // s=-1
can2[j]  =(  cand[j]  +  cand[j1]  +1)/2, can2[j+56]= (  cand[j1+70]+  cand[j2+70]+1)/2;
can2[j+ 8]=(5*cand[j+10]+ 9*cand[j1+10]+7)/14,can2[j+48]= (9*cand[j1+60]+5*cand[j2+60]+7)/14;
can2[j+16]=(3*cand[j+20]+11*cand[j1+20]+7)/14,can2[j+40]=(11*cand[j1+50]+3*cand[j2+50]+7)/14;
can2[j+24]=(  cand[j+30]+13*cand[j1+30]+7)/14,can2[j+32]=(13*cand[j1+40]+  cand[j2+40]+7)/14;
                }
                diff1 =0;  diff2 =0;
                for ( int i=0; i<8; i++ )
                {
                    int ii = 8*i;
                    for ( int j=0; j<8; j++ )
                    {
                        int jj = j+ii;
                        diff1 += abs( can1[jj] - tgt[jj] );   // s=-2 (ks=-4)
                        diff2 += abs( can2[jj] - tgt[jj] );   // s=-1 (ks=-2)
                    }
                }
// weight
                diff1 += 40*abs( wgt + 3*4 );
                diff2 += 40*abs( wgt + 3*2 );
                if ( diff1 < diff2 )   // s=-2 < s=-1
                {
                    if ( diff1 < diff3 )   // s=-2 < s=-1.5
                    {
                        for ( int i=0; i<8; i++ )
                        {
                            int ii = 8*i;
                            for ( int j=0; j<8; j++ )
                            {
                                int jj = j+ii;
                                rslt[jj] = can1[jj];   // s=-2
                            }
```

```
                                        }
                                        kslnt = -4;
                                }
                                else    // s=-1.5 <= s=-2 < s=-1
                                {
                                        // rslt[ij] = (s=-1.5), kslnt = -3
                                }
                        }
                        else    // s=-1 <= s=-2
                        {
                                if ( diff2 < diff3 )    // s=-1 < s=-1.5, s=-2
                                {
                                        for ( int i=0; i<8; i++ )
                                        {
                                                int ii = 8*i;
                                                for ( int j=0; j<8; j++ )
                                                {
                                                        int jj = j+ii;
                                                        rslt[jj] = can2[jj];    // s=-1
                                                }
                                        }
                                        kslnt = -2;
                                }
                                else    // s=-1.5 <= s=-1 <= s=-2
                                {
                                        // rslt[ij] = (s=-1.5), kslnt = -3
                                }
                        }
                }
                else    // s=0 <= s=-1.5 < s=1.5
                {
                        for ( int i=0; i<8; i++ )
                        {
                                int ii = 8*i;           int ij = 10*i+1;
                                for ( int j=0; j<8; j++ )
                                {
                                        int jj = j+ii;
                                        rslt[jj] = cand[j+ij];    // s=0
                                }
                        }
                        kslnt = 0;    // diff3 = 0
                        // call schk0
                        schk0( cand, tgt, wgt, diff3, rslt, kslnt );
                }
        }
```

```
        else    // s=1.5 <= s=-1.5
        {
                if ( diff2 < diff3 )    // s=1.5 < s=0, s=-1.5
                {
                        for ( int i=0; i<8; i++ )
                        {
                                int ii = 8*i;
                                for ( int j=0; j<8; j++ )
                                {
                                        int jj = j+ii;
                                        rslt[jj] = can2[jj];    // s=1.5
                                }
                        }
                        kslnt = 3;
                        diff3 = diff2;
                        //check s= 1, 1.5, 2
                        for ( int j=0; j<8; j++ )
                        {
                           int j1= 1+j, j2 = 2+j;
                           // s=1
can1[j]     = (    cand[j1]+       cand[j2]     +1)/2, can1[j+56]=(    cand[j+70]+     cand[j1+70]+1)/2;
can1[j+ 8]= (9*cand[j1+10]+5*cand[j2+10]+7)/14,can1[j+48]=(5*cand[j+60]+ 9*cand[j1+60]+7)/14;
can1[j+16]=(11*cand[j1+20]+3*cand[j2+20]+7)/14,can1[j+40]=(3*cand[j+50]+11*cand[j1+50]+7)/14;
can1[j+24]=(13*cand[j1+30]+   cand[j2+30]+7)/14,can1[j+32]=(    cand[j+40]+13*cand[j1+40]+7)/14;
                           // s=2
can2[j]     =     cand[j2],                           can2[j+56]=     cand[j+70];
can2[j+ 8]=(2*cand[j1+10]+5*cand[j2+10]+3)/7,can2[j+48]=(5*cand[j+60]+2*cand[j1+60]+3)/7;
can2[j+16]=(4*cand[j1+20]+3*cand[j2+20]+3)/7,can2[j+40]=(3*cand[j+50]+4*cand[j1+50]+3)/7;
can2[j+24]=(6*cand[j1+30]+   cand[j2+30]+3)/7,can2[j+32]=(    cand[j+40]+6*cand[j1+40]+3)/7;
                        }
                        diff1 =0;   diff2 =0;
                        for ( int i=0; i<8; i++ )
                        {
                                int ii = 8*i;
                                for ( int j=0; j<8; j++ )
                                {
                                        int jj = j+ii;
                                        diff1 += abs( can1[jj] - tgt[jj] );    // s=1 (ks=2)
                                        diff2 += abs( can2[jj] - tgt[jj] );    // s=2 (ks=4)
                                }
                        }
// weight
                        diff1 += 40*abs( wgt - 3*2 );
                        diff2 += 40*abs( wgt - 3*4 );
                        if ( diff1 < diff2 )    // S=1 < S=2
```

```
                {
                        if ( diff1 < diff3 )    // s=1 < s=1.5
                        {
                                for ( int i=0; i<8; i++ )
                                {
                                        int ii = 8*i;
                                        for ( int j=0; j<8; j++ )
                                        {
                                                int jj = j+ii;
                                                rslt[jj] = can1[jj];    // s=1
                                        }
                                }
                                kslnt = 2;
                        }
                        else    // s=1.5 <= s=1 < s=2
                        {
                                // rslt[ij] = (s=1.5), kslnt = 3
                        }
                }
                else    // s=2 <= s=1
                {
                        if ( diff2 < diff3 )    // s=2 < s=1.5, s=1
                        {
                                for ( int i=0; i<8; i++ )
                                {
                                        int ii = 8*i;
                                        for ( int j=0; j<8; j++ )
                                        {
                                                int jj = j+ii;
                                                rslt[jj] = can2[jj];    // s=2
                                        }
                                }
                                kslnt = 4;
                        }
                        else    // s=1.5 <= s=2 <= s=1
                        {
                                // rslt[ij] = (s=1.5), kslnt = 3
                        }
                }
        }
        else    // s=0 < s=1.5 < s=-1.5
        {
                for ( int i=0; i<8; i++ )
                {
                        int ii = 8*i;           int ij = 10*i+1;
```

```
                                        for ( int j=0; j<8; j++ )
                                        {
                                                int jj = j+ii;
                                                rslt[jj] = cand[j+ij];   // s=0
                                        }
                                }
                                kslnt = 0;   // diff3 = abs(cand[1+ij] -tgt[ij])
                                // check s=-0.5, 0, 0.5
                                // call schk0
                                schk0( cand, tgt, wgt, diff3, rslt, kslnt );
                        }
                }
        }
}
// disparity search
void dispredY( ColorComponent& orgy, ColorComponent& orgb, ColorComponent& orgr,
ColorComponent& recy, ColorComponent& recb, ColorComponent& recr, ColorComponent& refy,
ColorComponent& refb, ColorComponent& refr, ColorComponent& prp, ColorComponent& prd,
ColorComponent& lrp, ColorComponent& dvc, ColorComponent& kwc, ColorComponent& ksc )
{
  unsigned char*  pOrgy  = orgy.data;  unsigned char*  pOrgb  = orgb.data;
  unsigned char*  pOrgr  = orgr.data;  unsigned char*  pRecy  = recy.data;
  unsigned char*  pRecb  = recb.data;  unsigned char*  pRecr  = recr.data;
  unsigned char*  pRefy  = refy.data;  unsigned char*  pRefb  = refb.data;
  unsigned char*  pRefr  = refr.data;  unsigned char*  pPrp  = prp.data;
  unsigned char*  pPrd  = prd.data;  unsigned char*  pLrp  = lrp.data;
  unsigned char*  pDvc  = dvc.data;  unsigned char*  pKwc  = kwc.data;
  unsigned char*  pKsc  = ksc.data;
  int       difl[48], difr[48];          int       sign = 0;
  // for dvl or dvr
//  int dofset = 8;  int dmax = 20;
  // for dvc
  int dofset = 16;  int dmax = 40;
  for  ( int r=0; r<60; r++ )
  {
        int rc = 80*r;       int rr = 640*8*r;    int rrh = 320*4*r;
        for  ( int c=0; c<80; c++ )
    {
                int cc = 8*c;        int cch = 4*c;       int pix = rr + cc;
                int pixh = rrh + cch;
                for ( int d=0; d<dmax; d++ )
                {
                        int dd = d + dofset;  int ddh = dd / 2;
                        if ( cc + dd + 8 > 640 )
                        {
                                difl[d] =  16384;
```

```
if ( cc - dd < 0 )
                {
                        difr[d] =16384;
                }
else // (cc - dd) >= 0
                {
                        difr[d] = 0;
                        for ( int i=0; i<8; i++ )    // add luma error
                        {
                                int ii = pix + 640*i;
                                for ( int j=0; j<8; j++ )
                                {
                int add = ii+j;
                difr[d] += abs((int) pRecy[add] - (int) pRefy[add-dd]);
                                }
                        }
                        for ( int i=0; i<4; i++ )    // add chroma error
                        {
                                int ii = pixh + 320*i;
                                for ( int j=0; j<4; j++ )
                                {
                int add = ii+j;
                int addh = add - ddh;
                difr[d] += abs((int) pRecb[add] - (int) pRefb[addh]);
                difr[d] += abs((int) pRecr[add] - (int) pRefr[addh]);
                                }
                        }
                }
        }
        else // (cc + dd + 8) <= 640)
        {
                difl[d] = 0;
                if ( cc - dd < 0 )
                {
                        difr[d] =16384;
                        for ( int i=0; i<8; i++ )    // add luma error
                        {
                                int ii = pix + 640*i;
                                for ( int j=0; j<8; j++ )
                                {
                int add = ii+j;
                difl[d] += abs((int) pRecy[add] - (int) pOrgy[add+dd]);
                                }
                        }
                        for ( int i=0; i<4; i++ )    // add chroma error
```

```
                        {
                                int ii = pixh + 320*i;
                                for ( int j=0; j<4; j++ )
                                {
                        int add = ii+j;
                        difl[d] += abs((int) pRecb[add] - (int) pOrgb[add+ddh]);
                        difl[d] += abs((int) pRecr[add] - (int) pOrgr[add+ddh]);
                                }
                        }
                }
                else // (cc - dd) >= 0)
                {
                        difr[d] = 0;
                        for ( int i=0; i<8; i++ )    // add luma error
                        {
                                int ii = pix + 640*i;
                                for ( int j=0; j<8; j++ )
                                {
                        int add = ii+j;
                        difl[d] += abs((int) pRecy[add] - (int) pOrgy[add+dd]);
                        difr[d] += abs((int) pRecy[add] - (int) pRefy[add-dd]);
                                }
                        }
                        for ( int i=0; i<4; i++ )    // add chroma error
                        {
                                int ii = pixh + 320*i;
                                for ( int j=0; j<4; j++ )
                                {
                        int add = ii+j;
                        difl[d] += abs((int) pRecb[add] - (int) pOrgb[add+ddh]);
                        difl[d] += abs((int) pRecr[add] - (int) pOrgr[add+ddh]);
                        difr[d] += abs((int) pRecb[add] - (int) pRefb[add-ddh]);
                        difr[d] += abs((int) pRecr[add] - (int) pRefr[add-ddh]);
                                }
                        }
                }
        }
}
int minl = 16384,    minr = 16384;
int dspl = 0,        dspr = 0;
int rcc = rc + c;    int Ev = 0;
if ( r == 0 )    // no upper space
{
        if ( c == 0 ) // no upper nor left space
        {
```

```
                                for ( int d=0; d<dmax; d++ )
                                {
                                        int diffl = difl[d];    // no weight
                                        int diffr = difr[d];    // no weight
                                        if ( diffl < minl )
                                        {
                                                minl = diffl;           dspl = d;
                                        }
                                        if ( diffr < minr )
                                        {
                                                minr = diffr;           dspr = d;
                                        }
                                }
                        }
                        else // only left space
                        {
                                if ( pLrp[rcc-1] < 129 ) sign = -1;
                                else sign = 1;
//                              Ev = sign * ((int)pDvc[rcc-1] - 64); // 2* left disparity
                                Ev = sign * ((int)pDvc[rcc-1] - 96); // left disparity
                                for ( int d=0; d<dmax; d++ )
                                {
                                        int dd = d + dofset;
//                                      int diffl = difl[d] + 4*abs( Ev - 2*dd    );
//                                      int diffr = difr[d] + 4*abs( Ev + 2*dd    );
                                        int diffl = difl[d] + 16*abs( Ev - dd    );
                                        int diffr = difr[d] + 16*abs( Ev + dd    );
                                        if ( diffl < minl )
                                        {
                                                minl = diffl;           dspl = d;
                                        }
                                        if ( diffr < minr )
                                        {
                                                minr = diffr;           dspr = d;
                                        }
                                }
                        }
                }
                else // upper space
                {
                        if ( c == 0 ) // only upper space
                        {
                                if ( pLrp[rcc-80] < 129 ) sign = -1;
                                else sign = 1;
//                              Ev = sign * ((int)pDvc[rcc-80] -64);    // 2* upper disparity
```

```
                    Ev = sign * ((int)pDvc[rcc-80] -96);    // upper disparity
                    if ( pLrp[rcc-79] < 129 ) sign = -1;
                    else sign = 1;
//                  Ev += sign * ((int)pDvc[rcc-79] -64);   // 2* upper right disparity
                    Ev += sign * ((int)pDvc[rcc-79] -96);   // upper right disparity
                    for ( int d=0; d<dmax; d++ )
                    {
                        int dd = d + dofset;
//                      int diffl = difl[d] + 2*abs( Ev - 4*dd );
//                      int diffr = difr[d] + 2*abs( Ev + 4*dd );
                        int diffl = difl[d] + 8*abs( Ev - 2*dd );
                        int diffr = difr[d] + 8*abs( Ev + 2*dd );
                        if ( diffl < minl )
                        {
                            minl = diffl;       dspl = d;
                        }
                        if ( diffr < minr )
                        {
                            minr = diffr;       dspr = d;
                        }
                    }
                }
                if ( c > 0 && c < 79 ) // upper, left and right space
                {
                    if ( pLrp[rcc-81] < 129 ) sign = -1;
                    else sign = 1;
//                  Ev   = sign * ((int)pDvc[rcc-81] -64); // 2* up-left disparity
                    Ev   = sign * ((int)pDvc[rcc-81] -96); // up-left disparity
                    if ( pLrp[rcc-80] < 129 ) sign = -1;
                    else sign = 1;
//                  Ev += (sign * ((int)pDvc[rcc-80] -64)); // 2* upper disparity
                    Ev += (sign * ((int)pDvc[rcc-80] -96)); // 2* upper disparity
                    if ( pLrp[rcc-79] < 129 ) sign = -1;
                    else sign = 1;
//                  Ev += (sign * ((int)pDvc[rcc-79] -64)); // 2* up-right disparity
                    Ev += (sign * ((int)pDvc[rcc-79] -96)); // 2* up-right disparity
                    if ( pLrp[rcc-1] < 129 ) sign = -1;
                    else sign = 1;
//                  Ev += (sign * ((int)pDvc[rcc- 1] -64)); // 2* left disparity
                    Ev += (sign * ((int)pDvc[rcc- 1] -96)); // left disparity
                    for ( int d=0; d<dmax; d++ )
                    {
                        int dd = d + dofset;
//                      int diffl = difl[d] + abs( Ev - 8*dd );
//                      int diffr = difr[d] + abs( Ev + 8*dd );
```

```
                                        int diffl = difl[d] + 4*abs( Ev - 4*dd );
                                        int diffr = difr[d] + 4*abs( Ev + 4*dd );
                                        if ( diffl < minl )
                                        {
                                                minl = diffl;           dspl = d;
                                        }
                                        if ( diffr < minr )
                                        {
                                                minr = diffr;           dspr = d;
                                        }
                                }
                        }
                        if ( c == 79 ) // no right space
                        {
                                if ( pLrp[rcc-81] < 129 ) sign = -1;
                                else sign = 1;
//                              Ev    = sign * ((int)pDvc[rcc-81] -64); // 2* up-left disparity
                                Ev    = sign * ((int)pDvc[rcc-81] -96); // up-left disparity
                                if ( pLrp[rcc-80] < 129 ) sign = -1;
                                else sign = 1;
//                              Ev += (sign * ((int)pDvc[rcc-80] -64)); // 2* upper disparity
                                Ev += (sign * ((int)pDvc[rcc-80] -96)); // upper disparity
                                if ( pLrp[rcc-1] < 129 ) sign = -1;
                                else sign = 1;
//                              Ev += (sign * ((int)pDvc[rcc- 1] -64)); // 2* left disparity
                                Ev += (sign * ((int)pDvc[rcc- 1] -96)); // left disparity
                                for ( int d=0; d<dmax; d++ )
                                {
                                        int dd = d + dofset;
//                                      int diffl = difl[d] + abs( Ev - 6*dd );
//                                      int diffr = difr[d] + abs( Ev + 6*dd );
                                        int diffl = difl[d] + 5*abs( Ev - 3*dd );
                                        int diffr = difr[d] + 5*abs( Ev + 3*dd );
                                        if ( diffl < minl )
                                        {
                                                minl = diffl;           dspl = d;
                                        }
                                        if ( diffr < minr )
                                        {
                                                minr = diffr;           dspr = d;
                                        }
                                }
                        }
                }
                int dspv = 0;
```

```
//                  int min    = 0;
                    if ( minl <= minr )
                    {
//                              min = 16 * difl[dspl];
                                dspv = dspl + dofset;
                    }
                    else
                    {
//                              min = 16 * difr[dspr];          dspv = -dspr - dofset;
                    }
                    // copy ref block to preb[8x16]
                    int         preb[128];          int         add = pix + dspv;
                    if ( dspv >= 0 )
                    {
                                for ( int i=0; i<8; i++ )
                                {
                                            int ii = add - 4 + 640*i;
                                            for ( int j=0; j<12; j++ )
                                            {
                                                        preb[j+16*i] = 16* (int) pOrgy[j+ii];
                                            }
                                }
                                int redge = cc+dspv+8;
                                if ( redge >= 640 )
                                {
                                            for (int i=0; i<8; i++)
                                            {
                                                        int ii = 11+16*i;
                                                        preb[1+ii] = preb[ii]; preb[2+ii] = preb[ii];
                                                        preb[3+ii] = preb[ii];    preb[4+ii] = preb[ii];
                                            }
                                }
                                if ( redge == 639 )
                                {
                                            int add8 = add + 8;
                                            for (int i=0; i<8; i++)
                                            {
                                                        int ii = 12+16*i;
                                                        preb[ii]     = 16* (int) pOrgy[add8+640*i];
                                                        preb[1+ii] = preb[ii];    preb[2+ii] = preb[ii];
                                                        preb[3+ii] = preb[ii];
                                            }
                                }
                                if ( redge == 638 )
                                {
```

```
                int add8 = add + 8;
                for (int i=0; i<8; i++)
                {
                        int ii = 13+16*i;       int ij = 640*i + add8;
                        preb[ii-1] = 16* (int) pOrgy[    ij];
                        preb[ii    ] = 16* (int) pOrgy[1+ij];
                        preb[ii+1] = preb[ii]; preb[ii+2] = preb[ii];
                }
        }
        if ( redge == 637 )
        {
                int add8 = add + 8;
                for (int i=0; i<8; i++)
                {
                        int ii = 14+16*i;       int ij = 640*i + add8;
                        preb[ii-2] = 16* (int) pOrgy[    ij];
                        preb[ii-1] = 16* (int) pOrgy[1+ij];
                        preb[ii    ] = 16* (int) pOrgy[2+ij];
                        preb[ii+1] = preb[ii];
                }
        }
        if ( redge <= 636 )
        {
                int add8 = add + 8;
                for (int i=0; i<8; i++)
                {
                        int ii = 12+16*i;       int ij = add8+640*i;
                        preb[    ii] = 16* (int) pOrgy[    ij];
                        preb[1+ii] = 16* (int) pOrgy[1+ij];
                        preb[2+ii] = 16* (int) pOrgy[2+ij];
                        preb[3+ii] = 16* (int) pOrgy[3+ij];
                }
        }
}
else // ( dspv < 0 )
{
        for ( int i=0; i<8; i++ )
        {
                int ii = add + 640*i;                   int ij = 16*i + 4;
                for ( int j=0; j<12; j++ )
                {
                        preb[ij+j] = 16* (int) pRefy[ii+j];
                }
        }
        int ledge = cc+dspv;
```

```
if ( ledge <= 0 )
{
        for ( int i=0; i<8; i++ )
        {
                int ii = 16*i;
                preb[   ii] = preb[4+ii]; preb[1+ii] = preb[ii];
                preb[2+ii] = preb[ii];    preb[3+ii] = preb[ii];
        }
}
if ( ledge == 1 )
{
        int add1 = add - 1;
        for ( int i=0; i<8; i++ )
        {
                int ii = 3+16*i;
                preb[ii   ] = 16* (int) pRefy[add1+640*i];
                preb[ii-1] = preb[ii]; preb[ii-2] = preb[ii];
                preb[ii-3] = preb[ii];
        }
}
if ( ledge == 2 )
{
        int add1 = add - 1;
        for ( int i=0; i<8; i++ )
        {
                int ii = 2+16*i;        int ij = add1+640*i;
                preb[ii+1] = 16* (int) pRefy[ij   ];
                preb[ii   ] = 16* (int) pRefy[ij-1];
                preb[ii-1] = preb[ii];    preb[ii-2] = preb[ii];
        }
}
if ( ledge == 3 )
{
        int add1 = add - 1;
        for ( int i=0; i<8; i++ )
        {
                int ii = 1+16*i;        int ij = add1+640*i;
                preb[ii+2] = 16* (int) pRefy[ij   ];
                preb[ii+1] = 16* (int) pRefy[ij-1];
                preb[ii   ] = 16* (int) pRefy[ij-2];
                preb[ii-1] = preb[ii];
        }
}
if ( ledge >=4 )
{
```

```
                        int add1 = add - 1;
                        for ( int i=0; i<8; i++ )
                        {
                                int ii = 16*i;          int ij = add1+640*i;
                                preb[ii+3] = 16* (int) pRefy[ij   ];
                                preb[ii+2] = 16* (int) pRefy[ij-1];
                                preb[ii+1] = 16* (int) pRefy[ij-2];
                                preb[ii   ] = 16* (int) pRefy[ij-3];
                        }
                }
        }
        // minute block adjustment
        int target[64];     int hreslt[112];    int wreslt[80];     int rslt[64];
        int dspvh = 0;      int kwidth= 0;      int kslant= 0;      int wgt= 0;
        unsigned char sign = 0;
        for ( int i=0; i<8; i++ )
        {
                int ii = pix+640*i;             int ij = 8*i;
                for ( int j=0; j<8; j++ )
                {
                        target[ij+j] = 16* (int) pRecy[ii+j];
                }
        }
        // call half-pel search
//      hdet(preb, target, dspv, min, hreslt, dspvh);
        // without half pel search
//      dspvh = 2*dspv;
        for ( int i=0; i<8; i++)
        {
                int ii = 1+16*i;        int ij = 14*i;
                for ( int j=0; j<14; j++ )
                {
                        hreslt[j+ij] = preb[j+ii];
                }
        }
        // output disparity
//      if ( dspvh < 0 ) pLrp[rcc] = 128;
        if ( dspv < 0 ) pLrp[rcc] = 128;
        else pLrp[rcc] = 129;
//      pDvc[rcc] = (unsigned char) (abs(dspvh) + 64);
        pDvc[rcc] = (unsigned char) (abs(dspv) + 96);
        // calculate width weight
        if ( r == 0 )   // no upper sapce
        {
                if ( c == 0 )     // no upper nor left space
```

```
            {
                    wgt = 0;
            }
            else    // only left space
            {
                    wgt = 3*((int)pKwc[rcc-1] -128); // left kw(-4,,,+4)
            }
}
else   // upper space
{
            if ( c == 0 ) // only upper space
            {
                    wgt = 3*((int)pKwc[rcc-80] -128); // upper kw(-4,,,+4)
            }
            else   // upper and left space
            {
                    wgt = (int)pKwc[rcc-81] -128; // up-left kw(-4,,,+4)
                    wgt += ((int)pKwc[rcc-80] -128);// upper kw
                    wgt += ((int)pKwc[rcc-1] -128);    // left kw
            }
}
// call width search
wdet( hreslt, target, wgt, wreslt, kwidth );
// without width search
kwidth = 0;
for ( int i=0; i<8; i++)
{
            int i14 = 2+14*i;      int i10 = 10*i;
            for ( int j=0; j<10; j++ )
            {
                    wreslt[j+i10] = hreslt[j+i14];
            }
}
// out put width parameter
pKwc[rcc] = (unsigned char) (kwidth +128);
// calculate slant weight
if ( r == 0 )   // no upper sapce
{
            if ( c == 0 )     // no upper nor left space
            {
                    wgt = 0;   // defort ks=0
            }
            else    // only left space
            {
                    wgt = 3*((int)pKsc[rcc-1] -128); // left ks(-4,,,+4)
```

```
            }
}
else   // upper space
{
            if ( c == 0 ) // only upper space
            {
                        wgt = 3*((int)pKsc[rcc-80] -128); // upper ks(-4,,,+4)
            }
            else   // upper and left space
            {
                        wgt   = (int)pKsc[rcc-81] -128; // up-left ks(-4,,,+4)
                        wgt += ((int)pKsc[rcc-80] -128); // upper ks
                        wgt += ((int)pKsc[rcc-1] -128);   // left ks
            }
}
// call slant search
sdet( wreslt, target, wgt, rslt, kslant );
// without slant search
kslant =0;
for ( int i=0; i<8; i++)
{
            int ii = 1+10*i;        int ij = 8*i;
            for ( int j=0; j<8; j++ )
            {
                        rslt[j+ij] = wreslt[j+ii];
            }
}
// output slant parameter
pKsc[rcc] = (unsigned char) (kslant +128);
// copy predicted block with filter
if ( r == 0 )   // no upper space
{
            if ( c == 0 )   // no upper nor left sapaces
            {
                        for ( int i=0; i<8; i++ )
                        {
                                    int ii = pix + 640*i;  int ij = 8*i;
                                    for ( int j=0; j<8; j++ ) // no filter
                                    {
                                    int jj = j + ii;
                                    pPrp[jj] = (unsigned char) (rslt[j+ij]/16);
                                    }
                        }
            }
            else     // (c > 0) only left sapce
```

```
                    {
                        int difdl = abs( (int)pDvc[rcc-1] - (int)pDvc[rcc] );
//                      if ( difdl <= 1 )   // no left disparity difference
                        if ( difdl == 0 )   // no left disparity difference
                        {
                            for ( int i=0; i<8; i++ ) // no filter
                            {
                                int ii = pix +640*i;   int ij = 8*i;
                                for ( int j=0; j<8; j++ )
                                {
                            pPrp[j+ii]= (unsigned char) (rslt[j+ij]/16);
                                }
                            }
                        }
//                    if ( difdl > 1 && difdl <= 5 ) // left has small disp diff
                    if ( difdl > 0 && difdl < 3 ) // left has small disp diff
                        {
                            for ( int i=0; i<8; i++ )
                            {
                                int ii = pix +640*i;   int ij = 8*i;
                        pPrp[ii-1]=(unsigned char)((16*pPrp[ii-2]+32*pPrp[ii-1]+rslt[ij])/64);
                        pPrp[ii]   =(unsigned char)((16*pPrp[ii-1]+ 2*rslt[ij]+rslt[ij+1])/64);
                                for ( int j=1; j<8; j++ )
                                {
                            pPrp[j+ii]= (unsigned char) (rslt[j+ij]/16);
                                }
                            }
                        }
//                      if ( difdl > 5 ) // left has large disp diff
                        if ( difdl >= 3 ) // left has large disp diff
                        {
                            for ( int i=0; i<8; i++ )
                            {
                                int ii = pix -1 +640*i; int ij = 8*i;
pPrp[ii-1]=(unsigned char)((4*pPrp[ii-3]+16*pPrp[ii-2]+24*pPrp[ii-1]+16*pPrp[ii]+rslt[ij]/4)/64);
pPrp[ii]   =(unsigned char)((4*pPrp[ii-2]+16*pPrp[ii-1]+24*pPrp[ii]   +rslt[ij] +rslt[ij+1]/4)/64);
pPrp[ii+1]=(unsigned char)((4*pPrp[ii-1]+16*pPrp[ii]+6*rslt[ij]/4+ rslt[ij+1] +rslt[ij+2]/4)/64);
pPrp[ii+2]=(unsigned char)((4*pPrp[ii]   +rslt[ij] +6*rslt[ij+1]/4 +rslt[ij+2] +rslt[ij+3]/4)/64);
                                for ( int j=2; j<8; j++ )
                                {
                            pPrp[1+j+ii]= (unsigned char) (rslt[j+ij]/16);
                                }
                            }
                        }
                    }
```

```
                    }
                    else    // (r > 0)    upper space
                    {
                            if ( c == 0 )    // only upper space
                            {
                                    int difdu = abs( (int)pDvc[rcc-80] - (int)pDvc[rcc] );
//                                  if ( difdu <= 1 )    //upper block has same disparity
                                    if ( difdu == 0 )    //upper block has same disparity
                                    {
                                            for ( int i=0; i<8; i++ ) // no filter
                                            {
                                                    int ii = pix +640*i;   int ij = 8*i;
                                                    for ( int j=0; j<8; j++ )
                                                    {
                                            pPrp[j+ii]= (unsigned char) (rslt[j+ij]/16);
                                                    }
                                            }
                                    }
//                                  if ( difdu > 1 && difdu <= 5 )    // upper has small disp diff
                                    if ( difdu > 0 && difdu < 3 )    // upper has small disp diff
                                    {
                                            for ( int j=0; j<8; j++ )
                                            {
                                                    int jj = pix +j;
            pPrp[jj-640]=(unsigned char)((16*pPrp[jj-1280]+32*pPrp[jj-640]+rslt[j])/64);
            pPrp[jj]      =(unsigned char)((16*pPrp[jj-640] + 2*rslt[j]     +rslt[j+8])/64);
                                            }
                                            for ( int i=1; i<8; i++ )
                                            {
                                                    int ii = pix +640*i;   int ij = 8*i;
                                                    for ( int j=0; j<8; j++ )
                                                    {
                                            pPrp[j+ii]= (unsigned char) (rslt[j+ij]/16);
                                                    }
                                            }
                                    }
//                                  if ( difdu > 5 )    // upper has large dispdiff
                                    if ( difdu >= 3 )    // upper has large dispdiff
                                    {
                                            for ( int j=0; j<8; j++ )
                                            {
int jj = pix -640 +j;
pPrp[jj-640]=(unsigned
char)((4*pPrp[jj-1920]+16*pPrp[jj-1280]+24*pPrp[jj-640]+16*pPrp[jj]+rslt[j]/4)/64);
pPrp[jj]=(unsigned char)((4*pPrp[jj-1280]+16*pPrp[jj-640]+24*pPrp[jj] +rslt[j] +rslt[j+8]/4)/64);
```

```
pPrp[jj+640]=(unsigned char)((4*pPrp[jj-640]+16*pPrp[jj] +6*rslt[j]/4 +rslt[j+8] +rslt[j+16]/4)/64);
pPrp[jj+1280]=(unsigned char)((4*pPrp[jj]+rslt[j] +6*rslt[j+8]/4 +rslt[j+16]+rslt[j+24]/4)/64);
                                                }
                                                for ( int i=2; i<8; i++ )
                                                {
                                                        int ii = pix +640*i;   int ij = 8*i;
                                                        for ( int j=0; j<8; j++ )
                                                        {
                                                pPrp[j+ii]= (unsigned char) (rslt[j+ij]/16);
                                                        }
                                                }
                                        }
                                }
                                else    // (c > 0) // upper & left spaces
                                {
                                        int difdu = abs( (int)pDvc[rcc-80] - (int)pDvc[rcc] );
//                                      if ( difdu <= 1 )    // upper has same disparity
                                        if ( difdu == 0 )    // upper has same disparity
                                        {
                                        int difdl = abs( (int)pDvc[rcc-1] - (int)pDvc[rcc] );
//                                      if ( difdl <= 1 )    // upper & left have same disparity
                                        if ( difdl == 0 )    // upper & left have same disparity
                                                {
                                                        for    ( int i=0; i<8; i++ ) // no filter
                                                        {
                                                           int ii = pix+640*i; int ij = 8*i;
                                                           for ( int j=0; j<8; j++ )
                                                           {
                                                pPrp[j+ii]= (unsigned char) (rslt[j+ij]/16);
                                                           }
                                                        }
                                                }
//                                      if ( difdl > 1 && difdl <= 5 ) // left has small disp diff
                                        if ( difdl > 0 && difdl < 3 ) // left has small disp diff
                                                {
                                                        for ( int i=0; i<8; i++ )
                                                        {
                                                           int ii = pix +640*i;    int ij = 8*i;
                        pPrp[ii-1]=(unsigned char)((16*pPrp[ii-2]+32*pPrp[ii-1]+rslt[ij])/64);
                        pPrp[ii]    =(unsigned char)((16*pPrp[ii-1]+ 2*rslt[ij]+rslt[ij+1])/64);
                                                                for ( int j=1; j<8; j++ )
                                                                {
                                                pPrp[j+ii]= (unsigned char) (rslt[j+ij]/16);
                                                                }
                                                        }
```

```
                                        }
//                                      if ( difdl > 5 ) // left has large disp diff
                                        if ( difdl >= 3 ) // left has large disp diff
                                        {
                                                for ( int i=0; i<8; i++ )
                                                {
                                                   int ii = pix -1 +640*i; int ij = 8*i;
pPrp[ii-1]=(unsigned char)((4*pPrp[ii-3]+16*pPrp[ii-2]+24*pPrp[ii-1]+16*pPrp[ii]+rslt[ij]/4)/64);
pPrp[ii]   =(unsigned char)((4*pPrp[ii-2]+16*pPrp[ii-1]+24*pPrp[ii]    +rslt[ij]+rslt[ij+1]/4)/64);
pPrp[ii+1]=(unsigned char)((4*pPrp[ii-1]+16*pPrp[ii] +6*rslt[ij]/4 +rslt[ij+1]+rslt[ij+2]/4)/64);
pPrp[ii+2]=(unsigned char)((4*pPrp[ii]   +rslt[ij]   +6*rslt[ij+1]/4 +rslt[ij+2]+rslt[ij+3]/4)/64);
                                                                for ( int j=2; j<8; j++ )
                                                                {
                                        pPrp[1+j+ii]= (unsigned char) (rslt[j+ij]/16);
                                                                }
                                                }
                                        }
                                }
//                              if ( difdu > 1 && difdu <= 5 ) // upper has small disp deff
                                if ( difdu > 0 && difdu < 3 ) // upper has small disp deff
                                {
                                        for ( int j=0; j<8; j++ )
                                        {
        int jj = pix -640 +j;
        pPrp[jj]      =(unsigned char)((16*pPrp[jj-640]+32*pPrp[jj]+rslt[j])/64);
        pPrp[jj+640]=(unsigned char)((16*pPrp[jj]    +2*rslt[j] +rslt[j+8])/64);
                                        }
                                        int difdl = abs( (int)pDvc[rcc-1] - (int)pDvc[rcc] );
//                                      if ( difdl <= 1 )   // only upper has small disp diff
                                        if ( difdl == 0 )   // only upper has small disp diff
                                        {
                                                for   ( int i=1; i<8; i++ ) // no left filter
                                                {
                                                   int ii = pix + 640*i;   int ij = 8*i;
                                                   for ( int j=0; j<8; j++ )
                                                   {
                                        pPrp[j+ii]= (unsigned char) (rslt[j+ij]/16);
                                                   }
                                                }
                                        }
//                                      if ( difdl > 1 && difdl <= 5 ) // left has small disp diff
                                        if ( difdl > 0 && difdl < 3 ) // left has small disp diff
                                        {
                                                for ( int i=0; i<8; i++ )
                                                {
```

```
                                                int ii = pix -1 +640*i;    int ij = 8*i;
                    pPrp[ii]    =(unsigned char)((16*pPrp[ii-1]+32*pPrp[ii]+rslt[ij])/64);
                    pPrp[ii+1]=(unsigned char)((16*pPrp[ii]+2*rslt[ij]+rslt[ij+1])/64);
                                                }
                                                for    ( int i=1; i<8; i++ )
                                                {
                                                   int ii = pix + 640*i;    int ij = 8*i;
                                                   for ( int j=1; j<8; j++ )
                                                   {
                                            pPrp[j+ii]= (unsigned char) (rslt[j+ij]/16);
                                                   }
                                                }
                                            }
//                                          if ( difdl > 5 ) // left has large disp diff
                                            if ( difdl >= 3 ) // left has large disp diff
                                            {
                                                for ( int i=0; i<8; i++ )
                                                {
int ii = pix -1 +640*i;             int ij = 8*i;
pPrp[ii-1]=(unsigned char)((4*pPrp[ii-3]+16*pPrp[ii-2]+24*pPrp[ii-1]+16*pPrp[ii]+rslt[ij]/4)/64);
pPrp[ii]    =(unsigned char)((4*pPrp[ii-2]+16*pPrp[ii-1]+24*pPrp[ii]    +rslt[ij] +rslt[ij+1]/4)/64);
pPrp[ii+1]=(unsigned char)((4*pPrp[ii-1]+16*pPrp[ii] +6*rslt[ij]/4 +rslt[ij+1]+rslt[ij+2]/4)/64);
pPrp[ii+2]=(unsigned char)((4*pPrp[ii]    +rslt[ij]    +6*rslt[ij+1]/4 +rslt[ij+2]+rslt[ij+3]/4)/64);
                                                }
                                                for    ( int i=1; i<8; i++ )
                                                {
                                                   int ii = pix + 640*i;    int ij = 8*i;
                                                   for ( int j=2; j<8; j++ )
                                                   {
                                            pPrp[j+ii]= (unsigned char) (rslt[j+ij]/16);
                                                   }
                                                }
                                            }
                                        }
//                                      if ( difdu > 5 )    // upper has large disp diff
                                        if ( difdu >= 3 )    // upper has large disp diff
                                        {
                                                for ( int j=0; j<8; j++ )
                                                {
int jj = pix -640 +j;
pPrp[jj-640]=(unsigned
char)((4*pPrp[jj-1920]+16*pPrp[jj-1280]+24*pPrp[jj-640]+16*pPrp[jj]+rslt[j]/4)/64);
pPrp[jj]=(unsigned char)((4*pPrp[jj-1280]+16*pPrp[jj-640]+24*pPrp[jj] +rslt[j] +rslt[j+8]/4)/64);
pPrp[jj+640]=(unsigned char)((4*pPrp[jj-640] +16*pPrp[jj]+6*rslt[j]/4 +rslt[j+8] +rslt[j+16]/4)/64);
pPrp[jj+1280]=(unsigned char)((4*pPrp[jj]+rslt[j] +6*rslt[j+8]/4 +rslt[j+16] +rslt[j+24]/4)/64);
```

```
                                        }
                                        int difdl = abs( (int)pDvc[rcc-1] - (int)pDvc[rcc] );
//                                      if ( difdl <= 1 )   // only upper has large disp diff
                                        if ( difdl == 0 )   // only upper has large disp diff
                                        {
                                                for   ( int i=2; i<8; i++ ) // no left filter
                                                {
                                                   int ii = pix + 640*i;   int ij = 8*i;
                                                   for ( int j=0; j<8; j++ )
                                                   {
                                        pPrp[j+ii]= (unsigned char) (rslt[j+ij]/16);
                                                   }
                                                }
                                        }
//                                  if ( difdl > 1 && difdl <= 5 ) // left has small disp diff
                                    if ( difdl > 0 && difdl < 3 ) // left has small disp diff
                                        {
                                                for ( int i=0; i<8; i++ )
                                                {
                        int ii = pix +640*i;   int ij = 8*i;
                        pPrp[ii-1]=(unsigned char)((16*pPrp[ii-2]+32*pPrp[ii-1]+rslt[ij])/64);
                        pPrp[ii]   =(unsigned char)((16*pPrp[ii-1]+2*rslt[ij] +rslt[ij+1])/64);
                                                }
                                                for   ( int i=2; i<8; i++ )
                                                {
                                                   int ii = pix + 640*i; int ij = 8*i;
                                                   for ( int j=1; j<8; j++ )
                                                   {
                                        pPrp[j+ii]= (unsigned char) (rslt[j+ij]/16);
                                                   }
                                                }
                                        }
//                                      if ( difdl > 5 ) // left has large disp diff
                                        if ( difdl >= 3 ) // left has large disp diff
                                        {
                                                for ( int i=0; i<8; i++ )
                                                {
int ii = pix -1 +640*i;             int ij = 8*i;
pPrp[ii-1]=(unsigned char)((4*pPrp[ii-3]+16*pPrp[ii-2]+24*pPrp[ii-1]+16*pPrp[ii]+rslt[ij]/4)/64);
pPrp[ii]   =(unsigned char)((4*pPrp[ii-2]+16*pPrp[ii-1]+24*pPrp[ii]   +rslt[ij] +rslt[ij+1]/4)/64);
pPrp[ii+1]=(unsigned char)((4*pPrp[ii-1]+16*pPrp[ii] +6*rslt[ij]/4 +rslt[ij+1]+rslt[ij+2]/4)/64);
pPrp[ii+2]=(unsigned char)((4*pPrp[ii]    +rslt[ij] +6*rslt[ij+1]/4 +rslt[ij+2]+rslt[ij+3]/4)/64);
                                                }
                                                for   ( int i=2; i<8; i++ )
                                                {
```

```
                                                    int ii = pix + 640*i;    int ij = 8*i;
                                                            for ( int j=2; j<8; j++ )
                                                            {
                                                pPrp[j+ii]= (unsigned char) (rslt[j+ij]/16);
                                                            }
                                                    }
                                                }
                                            }
                                    }
                            }
// no filter
//                          for ( int i=0; i<8; i++ )
//                          {
//                                  int ii = pix + 640*i;  int ij = 8*i;
//                                  for ( int j=0; j<8; j++ ) // no filter
//                                  {
//                                          int jj = j + ii;
//                                          pPrp[jj] = (unsigned char) ((rslt[j+ij] +8)/16);
//                                  }
//                          }
                    }
            }
    // calculate prediction error
    for    ( int r=0; r<60; r++ )
    {
            int rr = 640*8*r;
            for    ( int c=0; c<80; c++ )
            {
                    int pix = rr + 8*c;
                    for ( int i=0; i<8; i++ )
                    {
                            int ii = pix + 640*i;
                            for ( int j=0; j<8; j++ )
                            {
                                    int jj = j + ii;
                                    int prde = (int) pRecy[jj] - (int) pPrp[jj];
                                    if ( prde < -128 )      prde = -128;
                                    if ( prde > 127 )       prde = 127;
                                    pPrd[jj] = (unsigned char) ( prde +128 );
                            }
                    }
            }
    }
    int rr = 80*59;
    for (int r=60; r<64; r++)
```

```
  {
            int rc = 80*r;
            for (int c=0; c<80; c++)
            {
                  int rcc = rc + c;       int cc   = rr + c;       pLrp[rcc] = 128;       pDvc[rcc] = 124;
                  pKwc[rcc] = 128;      pKsc[rcc] = 128;
            }
  }
}
// chroma width compensation
void wcomp( int cand[32], int kw, int can[24] )
{
            if ( kw == -6 ) // w=2
            {
                  for ( int i=0; i<4; i++ )
                  {
                              int ii = 6*i;              int ij = 8*i;
            can[   ii]=(3*cand[2+ij]+   cand[3+ij]+2)/4,can[5+ii]= (3*cand[5+ij]+   cand[4+ij]+2)/4;
            can[1+ii]=(   cand[2+ij]+3*cand[3+ij]+2)/4,can[4+ii]= (   cand[5+ij]+3*cand[4+ij]+2)/4;
            can[2+ii]=(3*cand[3+ij]+   cand[4+ij]+2)/4,can[3+ii]= (3*cand[4+ij]+   cand[3+ij]+2)/4;
                  }
            }
            if ( kw == -5 ) // w=2.5
            {
                  for ( int i=0; i<4; i++ )
                  {
int ii = 6*i;              int ij = 8*i;
can[   ii]=(    cand[1+ij]+15*cand[2+ij]+8)/16,can[5+ii]=(    cand[6+ij]+15*cand[5+ij]+8)/16;
can[1+ii]=( 7*cand[2+ij] +9*cand[3+ij]+8)/16,can[4+ii]=( 7*cand[5+ij] +9*cand[4+ij]+8)/16;
can[2+ii]=(13*cand[3+ij] +3*cand[4+ij]+8)/16,can[3+ii]=(13*cand[4+ij] +3*cand[3+ij]+8)/16;
                  }
            }
            if ( kw == -4 ) // w=3
            {
                  for ( int i=0; i<4; i++ )
                  {
int ii = 6*i;              int ij = 8*i;
can[   ii]=(3*cand[1+ij]+5*cand[2+ij] +4)/8,can[5+ii]= (3*cand[6+ij]+ 5*cand[5+ij] +4)/8;
can[1+ii]=(5*cand[2+ij]+3*cand[3+ij] +4)/8,can[4+ii]= (5*cand[5+ij]+ 3*cand[4+ij] +4)/8;
can[2+ii]=(7*cand[3+ij]+   cand[4+ij] +4)/8,can[3+ii]= (7*cand[4+ij]+    cand[3+ij] +4)/8;
                  }
            }
            if ( kw == -3 ) // w=3.25
            {
                  for ( int i=0; i<4; i++ )
```

```
                    {
int ii = 6*i;           int ij = 8*i;
can[   ii]=(10*cand[1+ij]+9*cand[2+ij]+9)/19,can[5+ii]=(10*cand[6+ij]+9*cand[5+ij]+9)/19;
can[1+ii]=( 5*cand[2+ij]+2*cand[3+ij]+3)/7, can[4+ii]=( 5*cand[5+ij]+2*cand[4+ij]+3)/7;
can[2+ii]=(10*cand[3+ij]+   cand[4+ij]+5)/11,can[3+ii]=(10*cand[4+ij]+   cand[3+ij]+5)/11;
                    }
          }
          if ( kw == -2 ) // w=3.5
          {
                    for ( int i=0; i<4; i++ )
                    {
                              int ii = 6*i;           int ij = 8*i;
can[   ii]=(11*cand[1+ij]+5*cand[2+ij]+8)/16,can[5+ii]=(11*cand[6+ij]+5*cand[5+ij]+8)/16;
can[1+ii]=( 9*cand[2+ij]+2*cand[3+ij]+5)/11,can[4+ii]=( 9*cand[5+ij]+2*cand[4+ij]+5)/11;
can[2+ii]=      cand[3+ij],                 can[3+ii]=      cand[4+ij];
                    }
          }
          if ( kw == -1 ) // w=3.75
          {
                    for ( int i=0; i<4; i++ )
                    {
                              int ii = 6*i;           int ij = 8*i;
can[   ii]=( 5*cand[1+ij]+cand[2+ij]+3)/6, can[5+ii]=( 5*cand[6+ij]+cand[5+ij]+3)/6;
can[1+ii]=(10*cand[2+ij]+cand[3+ij]+5)/11,can[4+ii]=(10*cand[5+ij]+cand[4+ij]+5)/11;
can[2+ii]=      cand[3+ij],                 can[3+ii]=      cand[4+ij];
                    }
          }
          if (   kw == 0 ) //   w=4
          {
                    for ( int i=0; i<4; i++ )
                    {
                              int ii = 6*i;           int ij = 8*i+1;
                              for ( int j=0; j<6; j++ )
                              {
                                        can[j+ii] = cand[j+ij];
                              }
                    }
          }
          if ( kw == 1 ) // w=4.25
          {
                    for ( int i=0; i<4; i++ )
                    {
                              int ii = 6*i;           int ij = 8*i;
can[   ii]=(cand[   ij]+5*cand[1+ij]+3)/6,can[5+ii]=(cand[7+ij]+5*cand[6+ij]+3)/6;
can[1+ii]=(cand[1+ij]+8*cand[2+ij]+4)/9,can[4+ii]=(cand[6+ij]+8*cand[5+ij]+4)/9;
```

```
can[2+ii]= cand[3+ij],                    can[3+ii]= cand[4+ij];
                }
        }
        if ( kw == 2 ) // w=4.5
        {
                for ( int i=0; i<4; i++ )
                {
                        int ii = 6*i;           int ij = 8*i;
can[  ii]=(5*cand[  ij]+11*cand[1+ij]+8)/16,can[5+ii]=(5*cand[7+ij]+11*cand[6+ij]+8)/16;
can[1+ii]=(  cand[1+ij]+ 4*cand[2+ij]+2)/5, can[4+ii]=(  cand[6+ij]+ 4*cand[5+ij]+2)/5;
can[2+ii]=   cand[3+ij],                  can[3+ii]=   cand[4+ij];
                }
        }
        if ( kw == 3 ) // w=4.75
        {
                for ( int i=0; i<4; i++ )
                {
                        int ii = 6*i;           int ij = 8*i;
can[  ii]=(9*cand[  ij]+10*cand[1+ij]+9)/19,can[5+ii]=(9*cand[7+ij]+10*cand[6+ij]+9)/19;
can[1+ii]=(2*cand[1+ij]+ 5*cand[2+ij]+3)/7, can[4+ii]=(2*cand[6+ij]+ 5*cand[5+ij]+3)/7;
can[2+ii]=(  cand[2+ij]+10*cand[3+ij]+5)/11,can[3+ii]=(  cand[5+ij]+10*cand[4+ij]+5)/11;
                }
        }
        if ( kw == 4 ) // w=5
        {
                for ( int i=0; i<4; i++ )
                {
                        int ii = 6*i;           int ij = 8*i;
can[  ii]=(5*cand[  ij]+3*cand[1+ij]+4)/8,can[5+ii]=(5*cand[7+ij]+3*cand[6+ij]+4)/8;
can[1+ii]=(3*cand[1+ij]+5*cand[2+ij]+4)/8,can[4+ii]=(3*cand[6+ij]+5*cand[5+ij]+4)/8;
can[2+ii]=(  cand[2+ij]+7*cand[3+ij]+4)/8,can[3+ii]=(  cand[5+ij]+7*cand[4+ij]+4)/8;
                }
        }
}
// slant compensation
void scomp( int cand[24], int ks, int rslt[16] )
{
        // cb/cr are 1pel down from y position (This works. Half-pel down works half.)
        if ( ks == -4 )
        {
                for ( int j=0; j<4; j++ )
                {
int j1 = j+1,   j2 = j+2;
rslt[j]  =(5*cand[j]  + 9*cand[j1]  +7)/14,rslt[j+12]=(   cand[j1+18]+  cand[j2+18]+1)/2;
rslt[j+4]=(  cand[j+6]+13*cand[j1+6]+7)/14,rslt[j+8] =(11*cand[j1+12]+3*cand[j2+12]+7)/14;
```

```
                }
        }
        if ( ks == -3 )
        {
                for ( int j=0; j<4; j++ )
                {
int j1 = j+1,   j2 = j+2;
rslt[j]  =(3*cand[j]  +8*cand[j1]+5)/11,rslt[j+12]=(5*cand[j1+18]+3*cand[j2+18]+4)/8;
rslt[j+4]=    cand[j1+6],                rslt[j+8] =(5*cand[j1+12]+ cand[j2+12]+3)/6;
                }
        }
        if ( ks == -2 )
        {
                for ( int j=0; j<4; j++ )
                {
int j1 = j+1,   j2 = j+2;
rslt[j]  =(2*cand[j]    +9*cand[j1]+5)/11,rslt[j+12]=(3*cand[j1+18]+cand[j2+18]+2)/4;
rslt[j+4]=    cand[j1+6],                rslt[j+8] =(8*cand[j1+12]+cand[j2+12]+4)/9;
                }
        }
        if ( ks == -1 )
        {
                for ( int j=0; j<4; j++ )
                {
int j1 = j+1,   j2 = j+2;
rslt[j]  =(cand[j]  +10*cand[j1]+5)/11,rslt[j+12]=(7*cand[j1+18]+cand[j2+18]+4)/8;
rslt[j+4]= cand[j1+6],             rslt[j+8] =    cand[j1+12];
                }
        }
        if ( ks == 0 )
        {
                for ( int i=0; i<4; i++ )
                {
                        int ii = 4*i,   ij = 6*i+1;
                        for ( int j=0; j<4; j++ )
                        {
                                rslt[j+ii] = cand[j+ij];
                        }
                }
        }
        if ( ks == 1 )
        {
                for ( int j=0; j<4; j++ )
                {
int j1 = j+1,   j2 = j+2;
```

```
rslt[j]  =(10*cand[j1]  +cand[j2]+5)/11,rslt[j+12]=(cand[j+18]+7*cand[j1+18]+4)/8;
rslt[j+4]=      cand[j1+6],                    rslt[j+8] = cand[j1+12];
                        }
            }
            if ( ks == 2 )
            {
                        for ( int j=0; j<4; j++ )
                        {
int j1 = j+1,   j2 = j+2;
rslt[j] = (9*cand[j1] +2*cand[j2]+5)/11,rslt[j+12]=(cand[j+18]+3*cand[j1+18]+2)/4;
rslt[j+4]=    cand[j1+6],                    rslt[j+8] =(cand[j+12]+8*cand[j1+12]+4)/9;
                        }
            }
            if ( ks == 3 )
            {
                        for ( int j=0; j<4; j++ )
                        {
int j1 = j+1,   j2 = j+2;
rslt[j]  =(8*cand[j1]  +3*cand[j2]+5)/11,rslt[j+12]=(3*cand[j+18]+5*cand[j1+18]+4)/8;
rslt[j+4]=    cand[j1+6],                    rslt[j+8] =(   cand[j+12]+5*cand[j1+12]+3)/6;
                        }
            }
            if ( ks == 4 )
            {
                        for ( int j=0; j<4; j++ )
                        {
int j1 = j+1,   j2 = j+2;
rslt[j]  =( 9*cand[j1]  +5*cand[j2]+7)/14,rslt[j+12]=(   cand[j+18]+cand[j1+18]+1)/2;
rslt[j+4]=(13*cand[j1+6]+cand[j2+6]+7)/14,rslt[j+8] =(3*cand[j+12]+11*cand[j1+12]+7)/14;
                        }
            }
 }
void dispredUV( ColorComponent& org, ColorComponent& rec, ColorComponent& ref,
ColorComponent& prp, ColorComponent& prd, ColorComponent& lrp, ColorComponent& dvc,
ColorComponent& kwc, ColorComponent& ksc )
{
   unsigned char*   pOrg   = org.data;   unsigned char*   pRec   = rec.data;
   unsigned char*   pRef   = ref.data;   unsigned char*   pPrp   = prp.data;
   unsigned char*   pPrd   = prd.data;   unsigned char*   pLrp   = lrp.data;
   unsigned char*   pDvc   = dvc.data;   unsigned char*   pKwc   = kwc.data;
   unsigned char*   pKsc   = ksc.data;
   int cand[36];   int hrslt[32];   int wrslt[24];   int rslt[16];   int sign = 0;
   for   ( int r=0; r<60; r++ )
   {
            int rd = 80*r;
```

```
int rr = 320*4*r;
for  ( int c=0; c<80; c++ )
{
                int pix = rr + 4*c;    int rdc = rd + c;
                if ( pLrp[rdc] < 129 ) sign = -1;
                else sign = 1;
//              int dspy = sign * ((int) pDvc[rdc] -64);    int dspv =   dspy / 4;
                int dspy = sign * ((int) pDvc[rdc] -96);    int dspv =   dspy / 2;
                int add = pix + dspv;

                // copy candidate block
                if ( dspv > 0 )
                {
                        for ( int i=0; i<4; i++ )
                        {
                                int ii = 9*i;           int ij = add +320*i -2;
                                for ( int j=0; j<6; j++ )
                                {
                                        cand[j+ii] = 16* (int) pOrg[j+ij];
                                }
                        }
                        int redge = 4*c+dspv+4;
                        if ( redge >=320 )
                        {
                                for ( int i=0; i<4; i++ )
                                {
                                        int ii = 5+9*i;
                                        cand[1+ii] = cand[ii]; cand[2+ii] = cand[ii];
                                        cand[3+ii] = cand[ii];
                                }
                        }
                        if ( redge == 319 )
                        {
                                for ( int i=0; i<4; i++ )
                                {
                                        int ii = 6+9*i;
                                        cand[   ii] = 16* (int)   pOrg[add+4+320*i];
                                        cand[1+ii] = cand[ii];   cand[2+ii] = cand[ii];
                                }
                        }
                        if ( redge == 318 )
                        {
                                int add4 = add + 4;
                                for ( int i=0; i<4; i++ )
                                {
```

```
                        int ii = 7+9*i;         int ij = add4+320*i;
                        cand[ii-1] = 16* (int) pOrg[ij];
                        cand[ii   ] = 16* (int) pOrg[ij+1];
                        cand[ii+1] = cand[ii];
                }
        }
        if ( redge <=317 )
        {
                int add4 = add + 4;
                for ( int i=0; i<4; i++ )
                {
                        int ii = 6+9*i;         int ij = add4+320*i;
                        cand[ii   ] = 16* (int) pOrg[ij];
                        cand[ii+1] = 16* (int) pOrg[ij+1];
                        cand[ii+2] = 16* (int) pOrg[ij+2];
                }
        }
}
else // (dspv < 0)
{
        for ( int i=0; i<4; i++ )
        {
                int ij = add + 320*i; int ii = 9*i+3;
                for ( int j=0; j<6; j++ )
                {
                        cand[ii+j] = 16* (int) pRef[ij+j];
                }
        }
        int ledge = 4*c+dspv;
        if ( ledge <= 0 )
        {
                for ( int i=0; i<4; i++ )
                {
                        int ii = 3+9*i;
                        cand[ii-1] = cand[ii];   cand[ii-2] = cand[ii];
                        cand[ii-3] = cand[ii];
                }
        }
        if ( ledge == 1 )
        {
                int add1 = add - 1;
                for ( int i=0; i<4; i++ )
                {
                        int ii = 2+9*i;
                        cand[ii   ] = 16* (int) pRef[add1+320*i];
```

```
                                cand[ii-1] = cand[ii]; cand[ii-2] = cand[ii];
                        }
                }
                if ( ledge == 2 )
                {
                        int add1 = add - 1;
                        for ( int i=0; i<4; i++ )
                        {
                                int ii = 1+9*i;         int ij = add1+320*i;
                                cand[ii+1] = 16* (int) pRef[ij];
                                cand[ii   ] = 16* (int) pRef[ij-1];
                                cand[ii-1] = cand[ii];
                        }
                }
                if ( ledge >= 3 )
                {
                        int add1 = add - 1;
                        for ( int i=0; i<4; i++ )
                        {
                                int ii = 9*i;           int ij = add1+320*i;
                                cand[ii+2] = 16* (int) pRef[ij];
                                cand[ii+1] = 16* (int) pRef[ij-1];
                                cand[ii   ] = 16* (int) pRef[ii-2];
                        }
                }
}
// full pel
dspy = 2*dspy;
//half pel compensation
int subpel = dspy % 4;
if ( subpel == 1 || subpel == -3 )
{
        for ( int i=0; i<4; i++ )
        {
                int ii = 8*i;           int ij = 9*i;
                for ( int j=0; j<8; j++ )
                {
                        int jj = ij+j;
                        hrslt[ii+j] = (3*cand[jj] + cand[jj+1] +2)/4;
                }
        }
}
if ( subpel == 3 || subpel == -1 )
{
        for ( int i=0; i<4; i++ )
```

```
		{
			int ii = 8*i;		int ij = 9*i;
			for ( int j=0; j<8; j++ )
			{
				int jj = ij+j;
				hrslt[ii+j] = (cand[jj] + 3*cand[jj+1] +2)/4;
			}
		}
	}
	if ( subpel == 2 || subpel == -2 )
	{
		for ( int i=0; i<4; i++ )
		{
			int ii = 8*i;		int ij = 9*i;
			for ( int j=0; j<8; j++ )
			{
				int jj = ij+j;
				hrslt[ii+j] = (cand[jj] + cand[jj+1] +1)/2;
			}
		}
	}
	if ( subpel == 0 )
	{
		int offset =0;
		if ( dspv < 0 ) offset = 1;
		for ( int i=0; i<4; i++ )
		{
			int ii = 8*i;		int ij = 9*i + offset;
			for ( int j=0; j<8; j++ )
			{
				hrslt[ii+j] = cand[ij+j];
			}
		}
	}
	// call width compensation
	int kw = (int)pKwc[rdc] -128;
	wcomp( hrslt, kw, wrslt );
	// call slant compensation
	int ks = (int)pKsc[rdc] -128;
	scomp( wrslt, ks, rslt );
	// copy predicted block with filter
	if ( r == 0 )	// no upper space
	{
		if ( c == 0 )	// no left sapaces
		{
```

```
                                        for ( int i=0; i<4; i++ )
                                        {
                                                int ii = pix + 320*i;  int ij = 4*i;
                                                for ( int j=0; j<4; j++ )
                                                {
                                        int jj = j + ii;
                                        pPrp[jj] = (unsigned char) ((rslt[j+ij]+8)/16);
                                                }
                                        }
                                }
                                else    // (c > 0) only left space
                                {
                                        int difdl = abs( (int)pDvc[rdc-1] - (int)pDvc[rdc] );
//                                      if ( difdl < 4 )   // no left disparity difference
                                        if ( difdl < 2 )   // no left disparity difference
                                        {
                                                for ( int i=0; i<4; i++ )
                                                {
                                                        int ii = pix +320*i;   int ij = 4*i;
                                                        for ( int j=0; j<4; j++ )
                                                        {
                                                pPrp[j+ii]= (unsigned char) ((rslt[j+ij]+8)/16);
                                                        }
                                                }
                                        }
 //                      if ( difdl >= 4 ) // left has disp diff
                        if ( difdl >= 2 ) // left has disp diff
                                        {
                                                for ( int i=0; i<4; i++ )
                                                {
int ii = pix +320*i;   int ij = 4*i;
pPrp[ii-1]=(unsigned char)((16*pPrp[ii-2]+32*pPrp[ii-1]+rslt[ij]+32)/64);
pPrp[ii]   =(unsigned char)((16*pPrp[ii-1]+ 2*rslt[ij]+rslt[ij+1]+32)/64);
                                                        for ( int j=1; j<8; j++ )
                                                        {
                                        pPrp[j+ii]= (unsigned char) ((rslt[j+ij]+8)/16);
                                                        }
                                                }
                                        }
                                }
                        }
                        else    // (r > 0)   upper space
                        {
                                if ( c == 0 )   // only upper space
                                {
```

```
                                        int difdu = abs( (int)pDvc[rdc-80] - (int)pDvc[rdc] );
//                                      if ( difdu < 4 )    //upper block has same disparity
                                        if ( difdu < 2 )    //upper block has same disparity
                                        {
                                                for ( int i=0; i<4; i++ )
                                                {
                                                        int ii = pix +320*i;   int ij = 4*i;
                                                        for ( int j=0; j<4; j++ )
                                                        {
                                        pPrp[j+ii]= (unsigned char) ((rslt[j+ij]+8)/16);
                                                        }
                                                }
                                        }
//                                      if ( difdu >= 4 )    // upper has disp diff
                                        if ( difdu >= 2 )    // upper has disp diff
                                        {
                                                for ( int j=0; j<4; j++ )
                                                {
        int jj = pix +j;
        pPrp[jj-320]=(unsigned char)((16*pPrp[jj-640]+32*pPrp[jj-320]+rslt[j]+32)/64);
        pPrp[jj]      =(unsigned char)((16*pPrp[jj-320] + 2*rslt[j]     +rslt[j+4]+32)/64);
                                                }
                                                for ( int i=1; i<4; i++ )
                                                {
                                                        int ii = pix +320*i;   int ij = 4*i;
                                                        for ( int j=0; j<4; j++ )
                                                        {
                                        pPrp[j+ii]= (unsigned char) ((rslt[j+ij]+8)/16);
                                                        }
                                                }
                                        }
                                }
                                else    // (c > 0) // upper & left spaces
                                {
                                        int difdu = abs( (int)pDvc[rdc-80] - (int)pDvc[rdc] );
//                                      if ( difdu < 4 )    // upper has same disparity
                                        if ( difdu < 2 )    // upper has same disparity
                                        {
int difdl = abs( (int)pDvc[rdc-1] - (int)pDvc[rdc] );
// if ( difdl < 4 )    // upper & left have same disparity
if ( difdl < 2 )    // upper & left have same disparity
                                                {
                                                        for    ( int i=0; i<4; i++ )
                                                        {
                                                           int ii = pix + 320*i; int ij = 4*i;
```

```
                                                for ( int j=0; j<4; j++ )
                                                {
                                        pPrp[j+ii]= (unsigned char) ((rslt[j+ij]+8)/16);
                                                }
                                            }
                                    }
//                                  if ( difdl >= 4 ) // left has disp diff
                                    if ( difdl >= 2 ) // left has disp diff
                                    {
                                            for ( int i=0; i<4; i++ )
                                            {
                    int ii = pix +320*i;   int ij = 4*i;
                    pPrp[ii-1]=(unsigned char)((16*pPrp[ii-2]+32*pPrp[ii-1]+rslt[ij]+32)/64);
                    pPrp[ii]   =(unsigned char)((16*pPrp[ii-1]+ 2*rslt[ij]+rslt[ij+1]+32)/64);
                                                    for ( int j=1; j<4; j++ )
                                                    {
                    pPrp[j+ii]= (unsigned char) ((rslt[j+ij]+8)/16);
                                                    }
                                            }
                                    }
                                }
//                              if ( difdu   >= 4) // upper has disp deff
                                if ( difdu   >= 2) // upper has disp deff
                                {
                                    for ( int j=0; j<4; j++ )
                                    {
                                            int jj = pix -320 +j;
        pPrp[jj]      =(unsigned char)((16*pPrp[jj-320]+32*pPrp[jj]+rslt[j]+32)/64);
        pPrp[jj+320]=(unsigned char)((16*pPrp[jj]     +2*rslt[j] +rslt[j+4]+32)/64);
                                    }
                                int difdl = abs( (int)pDvc[rdc-1] - (int)pDvc[rdc] );
//                              if ( difdl < 4 )   // only upper has small disp diff
                                if ( difdl < 2 )   // only upper has small disp diff
                                    {
                                            for   ( int i=1; i<4; i++ )
                                            {
                                               int ii = pix + 320*i;   int ij = 4*i;
                                               for ( int j=0; j<4; j++ )
                                               {
                                        pPrp[j+ii]= (unsigned char) ((rslt[j+ij]+8)/16);
                                               }
                                            }
                                    }
//                                  if ( difdl >= 4 ) // left has disp diff
                                    if ( difdl >= 2 ) // left has disp diff
```

```
                                        {
                                                for ( int i=0; i<4; i++ )
                                                {
                                                   int ii = pix -1 +320*i;    int ij = 4*i;
                        pPrp[ii]    =(unsigned char)((16*pPrp[ii-1]+32*pPrp[ii]+rslt[ij]+32)/64);
                        pPrp[ii+1]=(unsigned char)((16*pPrp[ii]+2*rslt[ij]+rslt[ij+1]+32)/64);
                                                }
                                                for    ( int i=1; i<4; i++ )
                                                {
                                                   int ii = pix + 320*i;    int ij = 4*i;
                                                   for ( int j=1; j<4; j++ )
                                                   {
                                                pPrp[j+ii]= (unsigned char) ((rslt[j+ij]+8)/16);
                                                   }
                                                }
                                        }
                                }
                        }
// no filter
//                      for ( int i=0; i<4; i++ )
//                      {
//                              int ii = pix + 320*i;  int ij = 4*i;
//                              for ( int j=0; j<4; j++ )
//                              {
//                                      int jj = j + ii;
//                                      pPrp[jj] = (unsigned char) ((rslt[j+ij]+8)/16);
//                              }
//                      }
                }
        }
        // calculate prediction error
        for ( int r=0; r<60; r++ )
        {
                int rr = 320*4*r;
                for ( int c=0; c<80; c++ )
                {
                        int pix = rr + 4*c;
                        for ( int i=0; i<4; i++ )
                        {
                                int ii = pix + 320*i;  int ij = 4*i;
                                for ( int j=0; j<4; j++ )
                                {
                                        int jj = ii + j;
                                        int prde = (int) pRec[jj] - (int) pPrp[jj];
```

```
                                        if ( prde < -128 )      prde = -128;
                    if ( prde > 127 )           prde = 127;
                                        pPrd[jj] = (unsigned char) ( prde +128 );
                                }
                        }
                }
        }
}
void fillzDsp( ColorComponent& dspcb, ColorComponent& dspcr )
{
   unsigned char*   pCb   = dspcb.data;   unsigned char*   pCr   = dspcr.data;
   for   ( int r=0; r<32; r++ )
   {
            int rr = 40*r;
     for   ( int c=0; c<40; c++ )
     {
            pCb[rr+c] = 0;          pCr[rr+c] = 0;
     }
   }
}
void dispred( YuvFrame& rcFrameOrg, YuvFrame& rcFrameRec, YuvFrame& rcFrameRef,
YuvFrame& rcFramePrp, YuvFrame& rcFramePrd, YuvFrame& rcFrameLrp, YuvFrame&
rcFrameDvc, YuvFrame& rcFrameKwc, YuvFrame& rcFrameKsc )
{
            dispredY   (   rcFrameOrg.lum,   rcFrameOrg.cb,   rcFrameOrg.cr,   rcFrameRec.lum,
rcFrameRec.cb, rcFrameRec.cr, rcFrameRef.lum, rcFrameRef.cb, rcFrameRef.cr, rcFramePrp.lum,
rcFramePrd.lum, rcFrameLrp.lum, rcFrameDvc.lum, rcFrameKwc.lum, rcFrameKsc.lum );
     dispredUV ( rcFrameOrg.cb, rcFrameRec.cb, rcFrameRef.cb, rcFramePrp.cb, rcFramePrd.cb,
rcFrameLrp.lum, rcFrameDvc.lum, rcFrameKwc.lum, rcFrameKsc.lum );
            dispredUV   (   rcFrameOrg.cr,   rcFrameRec.cr,   rcFrameRef.cr,   rcFramePrp.cr,
rcFramePrd.cr, rcFrameLrp.lum, rcFrameDvc.lum, rcFrameKwc.lum, rcFrameKsc.lum );
            fillzDsp ( rcFrameLrp.cb, rcFrameLrp.cr ); fillzDsp ( rcFrameDvc.cb, rcFrameDvc.cr );
            fillzDsp ( rcFrameKwc.cb, rcFrameKwc.cr ); fillzDsp ( rcFrameKsc.cb, rcFrameKsc.cr );
}
void readFrame( YuvFrame* f, FILE* file )
{
   readColorComponent( &f->lum, file );   readColorComponent( &f->cb,   file );
   readColorComponent( &f->cr,   file );
}
void writeFrame( YuvFrame* f, FILE* file   )
{
   writeColorComponent( &f->lum, file );   writeColorComponent( &f->cb,   file   );
   writeColorComponent( &f->cr,   file   );
}
void print_usage_and_exit( int test, char* name, char* message = 0 )
```

```
{
  if( test )
  {
    if( message )
    {
      fprintf ( stderr, "¥nERROR: %s¥n", message );
    }
fprintf (    stderr, "¥nUsage: %s <w> <h> <org> <rec> <ref> <prp> <prd> <lrp> <dvc> <kwc>
<ksc> ¥n¥n", name );
    fprintf (    stderr, "¥t      w: original width   (luma samples)¥n" );
    fprintf (    stderr, "¥t      h: original height (luma samples)¥n" );
    fprintf (    stderr, "¥t   org: left reference file¥n" );
    fprintf (    stderr, "¥t   rec: center original file¥n" );
    fprintf (    stderr, "¥t   ref: right reference file¥n" );
    fprintf (    stderr, "¥t   prp: predicted picture file¥n" );
    fprintf (    stderr, "¥t   prd: prediction error file¥n" );
    fprintf (    stderr, "¥t   lrp: left-right pointer file¥n" );
    fprintf (    stderr, "¥t   dvc: disparity vector file¥n" );
    fprintf (    stderr, "¥t   kwc: width coefficient file¥n" );
    fprintf (    stderr, "¥t   ksc: slant coefficient file¥n" );
    fprintf (    stderr, "¥n" );
    exit     (    1 );
  }
}
int main(int argc, char *argv[])
{
  int       acc = 10000;
  #define    OUT "%d,%04d"
  //===== input parameters =====
  unsigned int   width          = 0;   unsigned int   height    = 0;
  FILE*          org_file       = 0;   FILE*          rec_file      = 0;
  FILE*          ref_file       = 0;   FILE*          prp_file      = 0;
  FILE*          prd_file       = 0;   FILE*          lrp_file      = 0;
  FILE*          dvc_file       = 0;   FILE*          kwc_file      = 0;
  FILE*          ksc_file       = 0;
  //===== variables =====
  unsigned int   index, sequence_length;
  int            py, pu, pv;   double         psnrY, psnrU, psnrV;
  YuvFrame   cOrgFrame, cRecFrame, cRefFrame;
  YuvFrame   cPrpFrame, cPrdFrame, cLrpFrame;
  YuvFrame   cDvcFrame, cKwcFrame, cKscFrame;
  double      AveragePSNR_Y = 0.0,      AveragePSNR_U = 0.0,      AveragePSNR_V = 0.0;
  //===== read input parameters =====
  print_usage_and_exit      ( argc < 11, argv[0] );
  int ind  = 1;
```

```
width   = atoi   ( argv[ind++] );
height  = atoi   ( argv[ind++] );
org_file = fopen ( argv[ind++], "rb" );
rec_file = fopen ( argv[ind++], "rb" );
ref_file  = fopen ( argv[ind++], "rb" );
prp_file= fopen ( argv[ind++], "wb" );
prd_file= fopen ( argv[ind++], "wb" );
lrp_file  = fopen ( argv[ind++], "wb" );
dvc_file= fopen ( argv[ind++], "wb" );
kwc_file= fopen ( argv[ind++], "wb" );
ksc_file = fopen ( argv[ind++], "wb" );
//===== check input parameters =====
print_usage_and_exit   ( ! org_file, argv[0], "Cannot open left reference file!" );
print_usage_and_exit   ( ! rec_file, argv[0], "Cannot open reconstructed(center) file!" );
print_usage_and_exit   ( ! ref_file, argv[0], "Cannot open right reference file!" );
print_usage_and_exit   ( ! prp_file, argv[0], "Cannot open predicted picture file!" );
print_usage_and_exit   ( ! prd_file, argv[0], "Cannot open prediction difference file!" );
print_usage_and_exit   ( ! lrp_file, argv[0], "Cannot open prediction direction file!" );
print_usage_and_exit   ( ! dvc_file, argv[0], "Cannot open disparity vector file!" );
print_usage_and_exit   ( ! kwc_file, argv[0], "Cannot open width file!" );
print_usage_and_exit   ( ! ksc_file, argv[0], "Cannot open slant file!" );
//======= get number of frames and stream size =======
fseek( org_file, 0, SEEK_END );
size_t osize = ftell( org_file );
fseek( org_file, 0, SEEK_SET );
fseek( rec_file, 0, SEEK_SET );
fseek( ref_file, 0, SEEK_SET );
sequence_length = osize*4/(6*width)/height;
//===== initialization ======
        createFrame( &cOrgFrame, width, height );
        createFrame( &cRecFrame, width, height );
        createFrame( &cRefFrame, width, height );
        createFrame( &cPrpFrame, width, height );
        createFrame( &cPrdFrame, width, height );
        createFrame( &cLrpFrame, (unsigned int) width/8, (unsigned int) (height/8 +4) );
        createFrame( &cDvcFrame, (unsigned int) width/8, (unsigned int) (height/8 +4) );
        createFrame( &cKwcFrame, (unsigned int) width/8, (unsigned int) (height/8 +4) );
        createFrame( &cKscFrame, (unsigned int) width/8, (unsigned int) (height/8 +4) );
//===== loop over frames =====
for( index = 0; index < sequence_length; index++ )
{
                readFrame       ( &cOrgFrame, org_file );
                readFrame       ( &cRecFrame, rec_file );
                readFrame       ( &cRefFrame, ref_file );
                dispred             ( cOrgFrame, cRecFrame, cRefFrame, cPrpFrame,
```

```
cPrdFrame, cLrpFrame, cDvcFrame, cKwcFrame, cKscFrame );
                        writeFrame          ( &cPrpFrame, prp_file );
                        writeFrame          ( &cPrdFrame, prd_file );
                        writeFrame          ( &cLrpFrame, lrp_file );
                        writeFrame          ( &cDvcFrame, dvc_file );
                        writeFrame          ( &cKwcFrame, kwc_file );
                        writeFrame          ( &cKscFrame, ksc_file );
                        getPSNR             ( psnrY, psnrU, psnrV, cRecFrame, cPrpFrame );
                        AveragePSNR_Y +=    psnrY;
                        AveragePSNR_U +=    psnrU;
                        AveragePSNR_V +=    psnrV;
                        py = (int)floor( acc * psnrY + 0.5 );
                        pu = (int)floor( acc * psnrU + 0.5 );
                        pv = (int)floor( acc * psnrV + 0.5 );
fprintf(stdout,"%d¥t"OUT"¥t"OUT"¥t"OUT"¥n",index,py/acc,py%acc,pu/acc,pu%acc,pv/acc,pv%ac
c);
   }
   fprintf(stdout,"¥n");
   py = (int)floor( acc * AveragePSNR_Y / (double)sequence_length + 0.5 );
   pu = (int)floor( acc * AveragePSNR_U / (double)sequence_length + 0.5 );
   pv = (int)floor( acc * AveragePSNR_V / (double)sequence_length + 0.5 );
   fprintf(stderr,"total¥t"OUT"¥t"OUT"¥t"OUT"¥n",py/acc,py%acc,pu/acc,pu%acc,pv/acc,pv%acc);
   fprintf(stdout,"total¥t"OUT"¥t"OUT"¥t"OUT"¥n",py/acc,py%acc,pu/acc,pu%acc,pv/acc,pv%acc);
   fprintf(stdout, "¥n");
   //===== finish =====
            deleteFrame( &cOrgFrame   );          deleteFrame( &cRecFrame   );
            fclose      ( org_file    );          fclose      ( rec_file    );
            deleteFrame( &cRefFrame   );          deleteFrame( &cPrpFrame   );
            deleteFrame( &cPrdFrame   );          deleteFrame( &cLrpFrame   );
            deleteFrame( &cDvcFrame   );          deleteFrame( &cKwcFrame   );
            deleteFrame( &cKscFrame   );
            fclose      ( ref_file    );      fclose      ( prp_file    );
            fclose      ( prd_file    );      fclose      ( lrp_file    );
            fclose      ( dvc_file    );      fclose      ( kwc_file    );
            fclose      ( ksc_file    );
   return 0;
}
```