Robot Painter: High-Level Planning Based on Visual Perception

描画ロボット：　視覚に基づく高レベルプランニング

Miti Ruchanurucks

A Doctoral Dissertation
博士論文

Submitted to
The Graduate School of Information Science and Technology
The University of Tokyo
in Partial Fulfillment of the Requirements
for the Doctoral Degree of Information and Communication Engineering

Thesis Supervisor: Katsushi Ikeuchi 池内 克史
Professor of Graduate School of Interdisciplinary Information Studies

THE UNIVERSITY OF TOKYO

Abstract

Recently, many areas of research on humanoid robots have been studied, such as motion control, man-machine interfaces, artificial intelligence (AI), and so on. Among them many research projects have tried to create artist robots, with the common objective of exploring new sensing, artificial intelligence, and manipulation techniques.

The research described in this thesis explores new vision and manipulation techniques through painting tasks. The ultimate goal is to create a robot painter that has capabilities similar to those of human artists.

Regarding vision, the key problems of 2D/3D object segmentation, color perception, orientation mapping, and geometric edge processing are directly addressed by our method.

This research focuses, first, on an effective 2D segmentation scheme using local and global classifiers. Our proposed method can effectively deal with a foreground cut, multiple cuts, and cut before matting. Then it is shown how to exploit normal stereo cameras to roughly extract the object automatically, based on 3D background subtraction and other vision techniques, and how to use our 2D segmentation to extract the object area correctly.

The robot must analyze color distribution of the object to select the best set of colors to use. Normally, clustering colors face the tendency to produce colors with low contrast. We solve this problem by incorporating two clustering methods: maximum distance clustering and K-means.

Then, in order to draw brush strokes meaningfully, the robot senses the orientation of the object. To smooth the orientation of the whole object, we apply global orientation that exploits the radial basis function to generate a style similar to Van Gogh, for instance, for the entire object.

Furthermore, some human artists usually use edges to enhance their paintings. Technically, many researchers use gradient information to represent edges of objects. However, this would be affected by the color information on the surface. Hence, we decided to use 3D geometric edges as an input. We then extract 2D edges from the 3D model. Finally, the 2D edges are processed into brush strokes.

We show how to apply these methods to high-level manipulation using a robot platform that consists of two arms and multi-fingered hands.

The robot also has a stereo vision system. Based on the derived information, the robot then performs a visual feedback drawing. First, it detects a brush and grabs it using cameras and force sensors. Second, it calculates the position of the brush tip using principal component analysis (PCA). Third, it then draws and compares the canvas with the picture produced by the stereo cameras.

Finally, as the trajectories planned by the robot may not be realized on the real robot platform because of its physical limitations, this research presents a method to filter and optimize trajectories targeting offline and online applications. All physical attributes, namely angle, collision, velocity, and dynamic force, are considered as a set of constraints to be met and represented as B-spline coefficients, making the limits guaranteed.

The proposed method will be shown to outperform the current methods in the sense of correctness and minimal user interaction, and it does so in a reasonable computation time.

Acknowledgment

First, I would like to express my sincere gratitude to my advisor, Prof. Katsushi Ikeuchi, for his consideration and direction. He kindly took me on as his student from my master's degree through my doctoral degree. He also granted me sufficient freedom to pursuit my own research goals. In addition, the research environment provided by him enabled me to concentrate on testing many ideas in both theory and applications. Finally, his management strategy taught me how to deal with people. These gifts and skills will be invaluable to my future.

I am very grateful to Dr. Koichi Ogawara, Dr. Jun Takamatsu, and Dr. Shunsuke Kudoh. They worked very hard to build our robot platform and spent a great deal of time guiding me not only in programming but also regarding concepts in computer vision and robotics. Furthermore, they improved my presentations on various occasions. Dr. Koichi Ogawara often asked me about underlying concepts. Dr. Jun Takamatsu's comments were frank and helpful. Dr. Shunsuke Kudoh taught me a great deal about how to be a good programmer.

I would like to thank Prof. Atsushi Nakazawa for giving me many breakthrough ideas during my master's program as well as frank comments when we made presentations at conferences. I am very grateful to Dr. Shinichiro Nakaoka who was my mentor during my master's program. He provided a great deal of guidance, especially about humanoid robot technology. Some of his ideas changed my perspective about robotics. I would like to thank Dr. Takaaki Shiratori who supported me in everyday life in Japan, teaching me how to survive in the Japanese culture.

I also would like to express my sincere gratitude to Dr. Atsuhiko Banno, Dr. Shintaro Ono, Dr. Rei Kawakami, Mr. J.H. Manoj Vincent Perera, Mr. Bjoern Rennhak, Mr. Phongtharin Vinayavekhin, and other members of the Ikeuchi laboratory.

I am also very grateful to all members of Humanoid Robotics Group of AIST that have supported this research in various ways.

I also would like to thank Mr. Tanas Sirangul, Ms. Kuniko Urakawa, Dr. Atit Tamtrakarn, Ms. Siriluck Sujiwarodom, and other friends too numerous to name for their support, guidance, helpful conversation, and friendship.

I also would like to express my sincere gratitude to my defense committee who gave me valuable guidance on how to improve my research. They provided ideas on how to evaluate research projects and presentations, which will be particularly valuable to my future.

I would like to thank Dr. Joan Knapp and her son Mr. Robert Knapp for proofreading my papers as well as giving me guidance on how to improve my English.

Also, I could not finish this research without love from my father, my mother, and my wife.

Content

List of Figures

List of Tables

Chapter1

Introduction

Development of the humanoid robot has recently been progressing, and the technologies surrounding it have progressed simultaneously. Applications for the humanoid robot are expanding as well as its hardware and software development. A number of research studies have been conducted in this area for various purposes. Current studies on humanoid robots focus on such techniques as motion control, human-machine interfaces, artificial intelligence (AI), and so on.

In order to improve such technologies as well as to study human beings' mechanisms, among many approaches, a number of projects are devoted to the creation of artist robots. ISAC, for example, is a robot that can track an artist's hand trajectory and mimic it [SCN*98]. AARON is a unique robot that creates artworks by itself based on adapting a geometrical model of a subject [Mon97]. The Teleoperated Drawing Robot considers the possibility of a teleoperated robot that can track markers in the human hand online to generate brush trajectories [Dtr]. Draw-Bot focuses on force feedback to draw a simple pre-programmed shape [Db].

The research described in this dissertation attempts to create a robot that can mimic painting activities. The ultimate goal is to have a robot that can paint using its own creativity. Among the differences from the previous works are, first, our robot can plan "what" as well as "how" to paint automatically, by using cameras (shown in Fig. 1.1) to select the subject and choose colors and orientation in drawing. Second, the robot can move a paintbrush in a manner similar to that of human beings by using a multi-fingered hand shown in the figure to grasp the paintbrush and draw on a canvas.

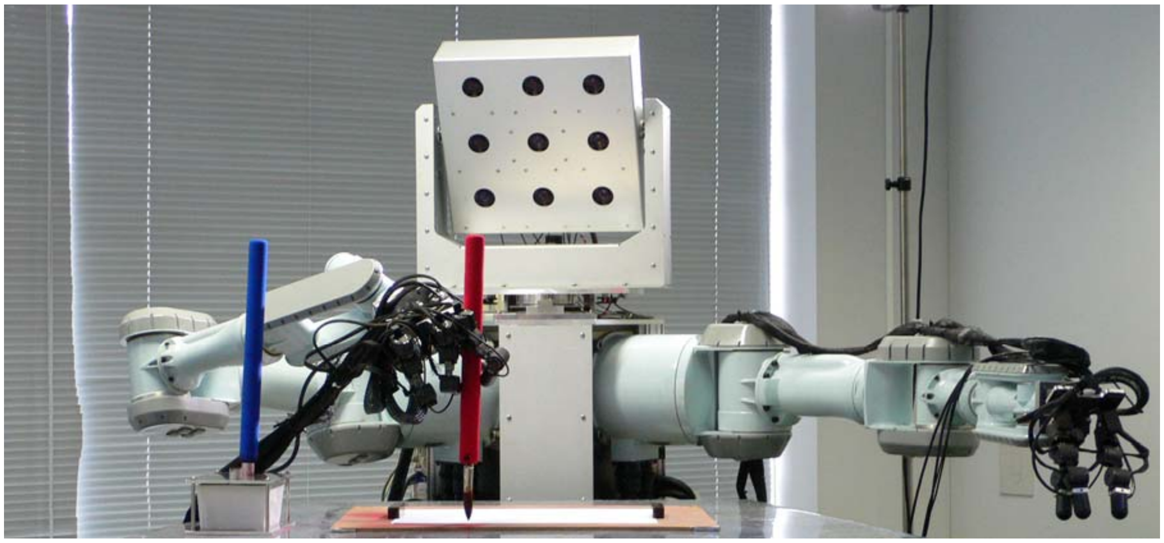


Fig. 1.1 Our robot painter.

1.1 Background

What is a robot painter?

Our robot painter is a system that can perform painting activities similar to those of human beings. The abilities can be divided into three parts, namely, observation/planning, physical interaction/execution, and verification/revision. This research forms a hypothesis that these three parts consist of several internal mechanisms as shown in Fig. 1.2. We attempt to develop a robot based on these mechanisms.

The observation/planning part consists of visual information analysis units A and B, a verbal information analysis unit, a strategy unit, and a tactical unit. Unit A captures and processes the 3D object to derive 3D/2D data, for example, edges and color. Unit B compares a 2D image with a pre-installed database to alter the image using computer vision techniques, for example, computer techniques related to impressionism. The verbal unit deals with information regarding listening and speaking, interactive conversation between people that could affect the strategy of the painter. Then, the strategy unit creates some form of picture model, which can be called a desired outcome. The desired outcome does not need to be exactly like the subject, for example, there could be model boundaries using edges or model areas using colors. After this, the tactical unit then transforms the picture model into brush strokes. Furthermore, the tactic depends on the strategy. For example, if the edge picture model is used, the tactic is how to extract meaningful edges. If the color picture model is used, the tactic is how to select the colors and trace their orientation.

The physical interaction/execution part then applies the brush strokes to draw on the canvas by task and skill models, referred to here as what to do with the paintbrush and how do it, respectively. Among the task models are grasping the paintbrush, calculating the position of the tip, detecting brush-canvas contact, and drawing a brush stroke. The skill model deals with techniques to accomplish such tasks, using vision and force sensors. Furthermore, the skill model of painting can be simulated without having to perform a real painting experiment.

The verification/revision part consists of unit B, visual information analysis unit C, the strategy unit, and the tactical unit. Unit B observes the canvas after drawing each brush stroke, and then can change the strategies or tactics. For example, it could adapt the original picture model to the actual canvas or adapt the brush stroke's size, respectively. Unit C observes the tip of the paintbrush in real time and informs the tactical unit to change the position of the paintbrush on-the-fly.

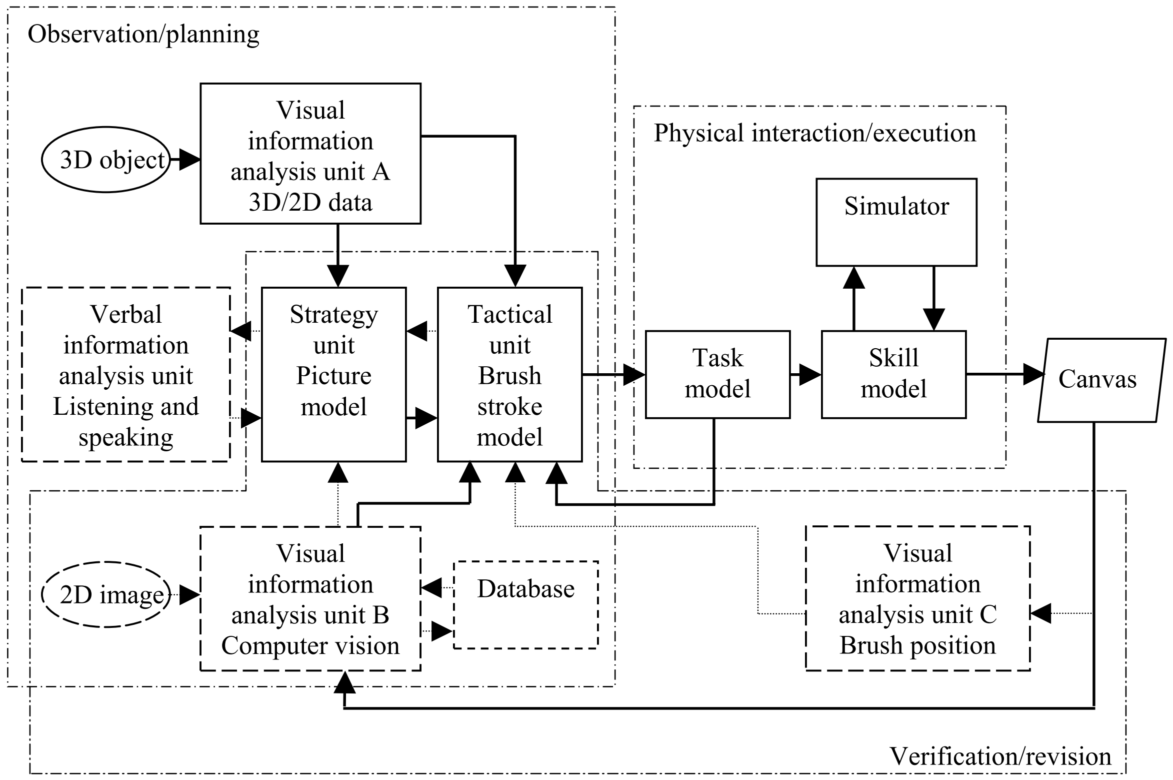

Observation/planning
3D object
Visual information analysis unit A 3D/2D data
Physical interaction/execution
Simulator
Verbal information analysis unit Listening and speaking
Strategy unit Picture model
Tactical unit Brush stroke model
Task model
Skill model
Canvas
2D image
Visual information analysis unit B Computer vision
Database
Visual information analysis unit C Brush position
Verification/revision

Fig. 1.2 Analysis of painting tasks. Our present robot painter consists of solid-line boxes. [This diagram is made by Masaki Fujihata, the project leader of this robot painter project.]

Why do we need to develop a robot painter?

This research achieves painting tasks by a robot, not a simulation program, based on the following reasons. First, dynamic characteristics of painting activities are sources of inspiration for artists. When a human draws, first, he or she observes the subject, using units A and B in Fig. 1.2. Then, the painter produces the picture model and the brush stroke model in his or her brain. Based on the models, the painter starts to paint using the physical interaction/execution part. However, the canvas might be different from the picture model. For example, colors on canvas can spread out in an unpredictable manner due to the deformable characteristics of the tip of the brush as well as the difference in thickness of colors, or a drop of color may unintentionally fall on the canvas. So the painter must refine strategies and tactics to cope with such disturbances using the verification/revision part. Often, artists consider the disturbances as a stimulus and a source of creativity. This final part is an essential element that is lacking in simulation programs.

Furthermore, the resulting technology of this achievement is also interesting from the viewpoint of art. This technology enables a robot to be a medium for reproducing human-like painting activities and the paintings. This medium is far more impressive for users than typical media based on simulation programs because users can appreciate real painting of real objects.

In addition, this medium has a potential to provide ways to improve current technology of computer vision and robotics. These technologies enable researchers to improve our everyday life by using robots in unwanted or complex tasks. Until now, most of such development is still limited to static tasks, which the robot knows the state of activities. Unfortunately, in reality, uncertainty is inevitable. Hence, it is important to enable a robot to handle tasks dynamically. The painting process involves uncertainty and is thus good training for dynamic solutions.

What are the technical challenges in making a robot painter?

Constructing all the elements presented in Fig. 1.2 requires a great deal of effort. At the present time, this research focuses on a compact scale of the robot painter shown in solid-line boxes in Fig. 1.2, which uses the visual unit A, the strategy unit, the tactical unit, the task model, the skill model, and the simulator. Based on this simplified method, we will explain the technical challenges of the three different parts, namely, observation/planning, physical interaction/execution, and verification/revision.

In the observation/planning part, the robot must be able to distinguish the foreground and the background robustly in reasonable time.

Then, the tactical unit needs to decide how to use the paintbrush. The decisions include the direction of brush strokes, how to paint areas, and how to draw outlines. These require techniques to sense orientation, select colors, and extract meaningful edges.

In the physical interaction/execution part, among the challenges are how to locate and move the paintbrush. Techniques to locate the paintbrush in noisy images are important. Grasping and moving the paintbrush to draw on a canvas is not straightforward.

In the verification/revision part, a visual feedback painting is performed using a simulator, and the grasping is omitted. At this stage, this research still does not deal with actual physical uncertainty effectively. Even so, the routine that generates brush strokes is not just a rendering program; it can be integrated into the robot in the future. An underlying difficulty is how to decide where to start drawing each brush stroke.

How do we overcome the technical challenges?

This research focuses, first, on how to extract the foreground robustly in reasonable time. It starts by using 3D segmentation techniques to roughly extract the object. Then we propose to use this noisy foreground/background as an initial input to a 2D segmentation program, which is more robust since it features more information such as that of color and gradient.

Furthermore, since current 2D segmentation technology has not been perfectly solved, this research proposes an approach to segment the foreground area correctly. The benefit is twofold: first, it can deal with foreground segmentation robustly, and second, it requires little input when there are many holes in the foreground and/or background as well as when there are thin or long objects on the boundary between foreground and background, for example, hair.

Next, to generate a color palette, the robot clusters the color distribution of the foreground. We propose to solve the low contrast problem of previous clustering routines by using Maximum Distance Clustering (MDC) to generate initial cluster positions for K-means clustering. Moreover, as MDC is comparatively slow when the number of desired colors is high, a suboptimal algorithm is therefore proposed.

Then, to draw brush strokes smoothly, the robot senses the orientation of the object. To generate smooth orientation for the whole object, we interpolate the orientation using a technique called Radial Basis Function (RBF).

To further draw the outlines, the robot extracts meaningful edges. We propose a concept of geometric edges that more precisely represents the outlines. A clustering method is used to transform the geometric edges into brush strokes.

Implementation related to the physical interaction/execution part and the verification/revision part is explained in the next subsection.

What do we implement?

Experiments are performed using the robot painter in Fig. 1.1. It has a stereo vision system with nine cameras and is used for obtaining an object's positions in 3D space. The camera head is equipped with a pan-tilt structure that allows its head to pan vertically and horizontally. The robot has two robot arms with seven Degrees Of Freedom (DOFs) each, and a multi-fingered hand on each arm. The right hand functions to handle the brush and has four fingers as shown in Fig. 6.1. The first finger has four DOFs while the other fingers each have three DOFs. On the tip of each finger, there are force sensors. We use a multi-fingered hand in order to undertake delicate paintbrush techniques that human painters use.

Regarding skills to detect and move the paintbrush, first, the robot detects a brush and grasps it using the cameras and the force sensors. Then, the orientation of the brush and the position of the brush tip are determined using principal component analysis (PCA). Verifying of brush-canvas contact is performed using force sensors. The robot uses the edge picture model as a goal for painting.

For visual feedback simulation, this research uses the color picture model. The decision to draw brush strokes is based on comparing the virtual canvas and the color picture model, in which a threshold of color difference is used to decide whether an area is already painted. The visual feedback is then examined in a hierarchical manner, which begins with large brushes and subsequently moves to small brushes to add details.

1.2 Related Work

1) Foreground Segmentation

Human painters often concentrate on specific areas of an image. This tendency is applicable for the robot painter to make paintings appear more beautiful. To accomplish this, object segmentation can be used to segment the subject area in a 3D image, for example, [TW05] and [MGS*05]. However, the methods cannot return the boundary of the subject correctly. Hence, this research focuses on how to segment the object area correctly and automatically, based on 3D techniques and a 2D segmentation method. By incorporating the 2D segmentation, the 3D extraction becomes more robust since it exploits features differently from 3D techniques.

Recent approaches to 2D segmentation attempt to extract the foreground using color information, edge information, or both, for example, Bayes matting [CCSS01], snakes [HBS99], or graph cut [BJ01]. However, the first scheme is not robust when the color distribution is not well separated. The second scheme is sensitive to gradient information. The third scheme cannot segment small or thin objects.

For methods that apply 2D segmentation to extract 3D objects, some works show how to apply graph cut to segment 3D images, such as medical images [BK04]. Some works regard video images as 3D objects and use graph cut to segment the foreground [WBC*05]. Among these methods, the drawback is that they require human interaction to mark the foreground and the background roughly. There is another group of works that uses special video sensors to capture and extract the foreground automatically [MMP05]. However, they need special, and expensive, hardware, whereas the robot painter can extract the foreground automatically using regular stereo cameras.

2) Color Perception

After extraction and segmentation, the robot must analyze color distribution of the object to select a small set of colors to formulate the picture model. Color perception can use splitting, clustering, AI, a spatial characteristic-based method, or a region-based method. Among the first group that applies a splitting-based algorithm are median-cut [Hec82] and octree [GP90]. However, they tend to produce false colors. The second group uses a clustering-based algorithm, such as K-means [YK04] and C-means [SK87]. This group faces the problem of low contrast. The third group uses AI for better classification results, for example, the Kohenen network [Dek94] and Neural Gas [AP06]. However, AI is time-consuming and difficult to implement. The fourth group tries to enhance the quality of the output image by considering spatial characteristics such as dithered color quantization [BFH*98], which cannot produce a color for each area effectively. The final group is region-based color segmentation, for example, watershed [VS91] and mean shift [CM02]. However, this approach often cannot reduce the number of colors effectively, and instead is applied in the field of region segmentation.

3) Global Orientation Calculation

For the local orientation used to guide brush strokes, gradient information can be used to guide brush strokes that are more robust to texture [Her98], whereas image moment is more robust to shape orientation [SY00]. However, such methods do not produce good brush strokes because the gradient or moment that a robot preserves could be noisy. To interpolate the orientation for the whole object, global orientation that exploits a radial basis function [Bor] to generate an orientation similar in style to Van Gogh can be used [HE04]. However, this consumes a great deal of time. It means that after the subject is designated, the robot must analyze the orientation for minutes before starting to draw. Such a long delay is not acceptable on many occasions, for instance, when using the robot to paint human subjects in an exhibition.

4) Brush Manipulation

Based on the picture model, the robot then performs brush strokes painting, which requires manipulation techniques. Some researchers have tried to manipulate by a multi-fingered hand [Nap56], [Cut89], [KI97]. However, they expect that their method would be used for rigid objects, and therefore few studies have been carried out that require manipulating deformable objects, like a brush in this study. The contact states of such objects do not rely only on a kinematics model, but also on other parameters such as force acting on fingers.

5) Robot Constraints

Finally, as robots have physical limitations, a motion generation routine must be applied to prevent limit violations. The limit violations lead to error in trajectories, and the error might result in collision and damage to the hardware. This must be prevented by applying a motion generation routine that guarantees angle, velocity, force, and collision limits.

Among motion generation methods, a filter-based approach is a fast way to retarget motion [PHRA02]. However, forces and collision avoidance are not considered. Trajectory optimization is another approach [UAR04]. The objective function can minimize physical characteristics; however, it is not guaranteed that such values will satisfy limits. A real-time approach also presents problems [RUWA03]. Attempts to limit physical characteristics rely on kinematics-based optimization to reduce values, rather than applying specific limitations [DVS01]. Thus, it faces the same problem of guaranteeing limit values.

1.3 Organization of the Thesis

Our method directly addresses these key problems: foreground segmentation, color perception, orientation mapping, geometric edge processing, skill level sensing and manipulation, and constraints to the trajectory. In Chapter 2, our 2D/3D foreground segmentation is explained. Chapter 3 explains the color perception method. Chapter 4 demonstrates multi-scale painting using an orientation map. Chapter 5 explains how to extract and process geometric edges of an object. Chapter 6 shows paintbrush manipulation by a camera and force sensors. Finally, Chapter 7 contains our conclusions. Robot constraint functions are shown in Appendix A with their online and offline applications. Appendix B explains Chebyshev's inequality, which is used as an underlying theory for foreground segmentation.

Chapter 2

Foreground Segmentation

In order to extract the subject area, the object is placed in front of the robot. The robot then automatically extracts the foreground. The question is: how does the robot accomplish that? Attempts to extract objects using 3D information to gain adequate information and resolution of images have been relying on high quality hardware, for example, the medical capturing device [BJ01], the laser range finder [ROI06], or special stereo cameras [MMP05]. At present, one challenge in this field is whether it is possible to extract such information using normal stereo cameras.

Because conventional 3D techniques always produce a noisy image that is not suitable to be used as a model for painting, whereas 2D segmentation methods often give a clear boundary, in this research, 3D segmentation is used first to extract the boundary roughly and then a novel 2D segmentation is used to extract the correct boundary. This chapter emphasizes the 2D segmentation method and then discusses its application to 3D material.

2D image segmentation has been extensively researched and is useful in image processing, in computer vision, and in many other aspects of computer graphics. It plays an important role not only in editing still images but also in editing video images [LSS05] [WBC*05], disparity cut [KYO*07] [RKOSI07], 3D image enhancement [BK04], 3D modeling [ROI06], etc.

This work focuses on a method that optimizes local similarity and global fitness with iterative edge constraint and that can deal with the problems of a foreground cut, multiple cuts, and cut before matting. Hence, it is called "Comprehensive Iterative Foreground Segmentation (CIFS)." Specifically, it will be shown here that this method answers questions that have become growing concerns for computer vision researchers: What are the conditions for robust segmentation? And why is it that many previous methods that are robust for a foreground cut cannot deal with multiple cuts effectively? In these cases, based on existing schemes, how should the algorithm be revamped? Why are many high-end matting approaches comparatively weak in dealing with data distribution that is not well separated? Then, how can matting be performed effectively? Furthermore, as many of the current high-end segmentation methods are more or less robust when enough input is provided, we are well aware that any novelty that could reduce user interaction warrants serious consideration. Our method can effectively deal with many of these problems.

It is widely known that many segmentation approaches have had different limitations mainly because of their interfaces, underlying mathematics, and algorithms. Among many approaches, we observe that the supervised region-based approach with an edge constraint scheme has not been widely applied to difficult images. Research in other fields has discovered that considering some parameters as a constraint as opposed to a cost function yields extra performance; hence, it is worthwhile to ascertain what benefit could be gained if the optimization is constrained.

With this in mind, this research considers how to integrate and improve graph cut and region growing based on the following reasons. Whereas graph cut can segment a foreground well as its cost function deals with both local and global similarity, its optimization scheme is unconstrained. We will show that using a similar cost function, constrained optimization improves user interaction, while the robustness of graph cut is preserved. On the other hands, region growing's use is very limited since its cost function is too simple. However, region growing has states, which means it stops before growing further. The benefit of this notion of state is not clearly addressed earlier, and we will consider this as a constrained optimization problem.

Our novelty is, first, we propose to parameterize the state based on a statistical value of Chebyshev's inequality. In doing so, the proposed method can constrain the timing before searching for a remote area as well as before matting. This reduces user interaction a great deal either when there are multiple holes in the images or when matting is required in difficult images, as the user does not have to mark every hole or the boundary for matting. Second, the cost function used in this work consists of local and global similarity, similar to that of graph cut, which leads to robustness. Conventional region growing programs cannot exploit such cost function. This is accomplished by our notion of subjecting each local and global term to each constraint.

When the behavior differences are studied, the proposed method is an improvement compared to previous approaches, namely, graph cut, region growing, object recognition, and matting.

In 3D segmentation, the method is then applied to allow the robot's stereo cameras to extract the boundary of the foreground correctly. The success of this procedure relies on how good the quality of the 3D foreground is. Then, a number of dilations and erosions are performed on the 3D foreground to produce a rough foreground for 2D segmentation.

The rest of this chapter is organized as follows. Section 1 explains the differences between previous approaches and our CIFS. In Section 2, the core notions of the proposed system are explained. The system is then applied to a foreground cut. Section 3 explains how to perform multiple cuts. Section 4 extends the notion to tackle the problem of cut before matting. Section 5 shows how to apply the method for 3D object extraction.

Extensive discussion is provided in Section 6. To better understand the notion, the algorithm is provided in Section 7. Finally, there is a summary in Section 8.

2.1 Related Work

Despite the value of image segmentation, the segmentation problem is still imperfectly solved. We first discuss several current approaches to segmentation and their limitations.

Most current segmentation methods require human input: a user marks areas as being foreground vs. background or marks a contour for use as a guideline. Interestingly, it is also shown that not only the mathematics and the algorithm that defines performance are different, but user interfaces are also different. For example, marking areas as being foreground or background is different from drawing the guideline contour. In the former interface, users can directly specify features to be used for data distribution. Controlling topology is also more straightforward. Our program uses this interface.

1) Unsupervised Edge-based Approach

Active contour models or "snakes" are among the most famous methods in the unsupervised edge-based approach to segmentation. A user marks one or more contours for use as initial boundaries. The notion of snakes is used to evolve the contour or contours L in image X based on internal and external measures of edge "energy," as expressed in the following equation

$$E(L) = \varsigma \int \left|L_s\right|^2 ds + \xi \int \left|L_{ss}\right|^2 ds - \int \left|\nabla X(L)\right|^2 ds \tag{2.1}$$

where the first and second terms are internal forces that measure the length of the contour and its stiffness or rigidity, weighted by parameters ς and ξ, whereas the third term is external force, which is generally based on image gradient. L_s and L_{ss} denote the first and second derivative with respect to the curve parameter s.

Since the seminal paper on snakes [KWT88], many variants of snakes have been proposed. Generally, snakes function can be divided into two main categories, explicit and implicit.

Explicit function such as splines can be used to represent the contour. The curve can be propagated based on different energy terms. However, explicit function faces difficulty when the curves need to be split or merged for multiple section segmentation. Some studies try to overcome the local minima problem, for example, using balloon force [Coh91]. Even so, the active contour cannot be made to extrude through any significant protrusions that the shape may possess.

Another breakthrough in the edge-based approach that can effectively deal with multiple cuts is the exploitation of implicit functions. "Level set" [MSV95] is one such solution in which snakes become "geodesic active contour" [CKS97]. The use of implicit representation of contour simplifies the process of evolving so that there is no longer a need to carefully control the splitting or merging of explicit function.

However, the edge-based approach is seriously criticized as segmentations depend heavily on initialization and gradient information. Recently, edge tracking and hybrid methods have also been proposed, namely, intelligent scissors [MB95] and a graph-cut based active contour [XBA03], respectively. The former can be considered as a supervised edge-based approach although it is affected by irrelevant gradient information.

Additionally, users may find it is not easy to mark or revise an appropriate snakes guideline contour for difficult images.

2) Unsupervised Region-based Approach

In contrast to using gradient information, many works calculate external energy based on other features instead. This is shown to be more robust for initialization [CRD06].

Nevertheless, by drawing the guideline contour as input, users may find it is not easy to mark the guideline contour. Moreover, they cannot directly specify pixels to be used as input data distribution.

On the contrary, for works that automatically divide images into multiple regions [SbFAZ00] [VC02] the decision is based on grouping similar features directly, which can be considered as clustering [BW04]. However, the decision would not work well if the data distributions are not well separated.

3) Supervised Region-based Approach

Supervised region-based methods interact with a user naturally. A user marks foreground vs. background, and these marked regions can be used to compute the image's data distribution. This means that users not only control a region's topology but can also specify its feature samples. There are four famous methods for this approach, namely, "region growing," "object recognition," "matting," and "graph cut."

3.1) Region Growing

In this type of method, a region can expand based on local characteristics such as color similarity, and there is always some criterion to stop the searching process such as heuristic strong edge information or optimal threshold between foreground and background [FSA07].

However, region growing is not a sophisticated method and is generally used for labeling rather than for segmentation. Attempts to improve region growing fail because the issue of global data distribution is not well addressed.

3.2) Object Recognition

In problems of object recognition, in which color patterns are often unknown or change only slightly, color-distribution-based separation methods are of limited use. Instead, machine learning dominates this field. [FWF02] focuses on combining belief and neural networks to apply a set of labels (e.g., sky, vegetation) to images. [WJ05] uses a probability model to extract an object of interest based on input training data without human interaction. These techniques, however, focus on unsupervised object recognition more than on how to effectively segment the foreground.

On the other hand, supervised learning techniques such as [MFM04] detect meaningful edges in images by exploiting brightness, color, and texture. They compare the use of density estimation, decision trees, logistic regression, hierarchical mixtures of expert systems, and support vector machines. [FO03] compares various machine-learning algorithms such as neural network, nearest-neighbor, and decision trees with the objective being to recognize a hand in moving images. However, a great deal of user interaction is required during the initial training.

Another problem with object recognition is that it is usually noisy and does not give a clear enough boundary for image/video editing.

3.3) Matting

An example of a recent method that exploits color distribution to produce a soft boundary, in other words, matting, is Bayes matting [CCSS01], which is an improvement on the algorithms presented in [Mis93], [BVD00], and [RT00]. From a user-marked foreground/background, an "alpha" value, which is the method's descriptor of being foreground vs. background, is computed over the remaining region. [LLW06] proposes a close form solution to the alpha matting problem. It can be seen that using color information extensively applies to a problem of alpha value approximation. However, the notion often fails when color distributions between foreground and background are not well separated.

Often, users need to mark an area to be matted manually, which is a tedious task. [WC05] tries to address the problem by introducing an iterative approximation approach. Their recent result can be found in [WC07]. However, the method shares a similar weakness to normal matting for images when the foreground/background colors are too ambiguous. This is the downside of considering only the color information. In addition, segmentations depend on initialization.

3.4) Graph Cut

Optimizing both smoothness between neighboring pixels and fitness to the model derived from data distribution given by users seems to offer an advantage. Graph cut [BJ01] [GPS89] is one such technique, and is presented in the following equation.

$$E(X) = \sum_{i,j} E_1(x_i, x_j) + \lambda \sum_{i} E_2(x_i, \psi) \tag{2.2}$$

where the image is an array $X = (x_1, ..., x_N)$ for which optimized cut energy can be derived by maximizing the smoothness E_1 and the fitness E_2; given the model Ψ, weighted by a parameter λ.

Since the original graph cut algorithm was developed, there have been various improvements. [BJ01] focuses on a multidimensional graph cut. [RKB04] improves the graph cut method by allowing interactive estimation and incomplete labeling, both of which reduce the amount of needed user interaction. [LSTS04] incorporates interactive boundary editing by representing the border as a set of vertices.

One clear advantage of graph cut is that it is less sensitive to initialization and color distribution, as both local and global information are considered at the same time.

However, the tradeoff generally makes graph cut incapable of extracting small or thin objects, in contrast to the matting method. [RKB04] proposed to cut before matting by performing graph cut and then erosion to generate unknown areas for the matting algorithm. However, the number of erosions is ad hoc, which is one problem [WC05] tries to address as mentioned.

Another problem with graph cut is the high degree of user interaction required in some cases of multiple cuts.

4) The Proposed Segmentation Method

This research tries to generate a method that is robust against sensitivity to initialization, ambiguous color distribution, and the loss of thin or small sections. When considering user interaction, the topics extend to how easily the user can guide the initial mark, how convenient it is to revise the marked data, how to give small input for images with multiple sections, and how to perform matting without having users further mark areas to be matted.

In light of the limitations of former approaches, we propose a comprehensive iterative foreground segmentation “CIFS” that can segment the foreground based on optimizing a cost function of feature similarity, locally and globally, with a constraint of edge strength. Unlike “graph cut,” which tries to trade off between local and global information using a weighting parameter, CIFS exploits “region growing” and “alpha value calculation” to deal with local and global optimization, respectively. The weighting is directly done using the constraint.

First, do:

$$E_1(X, a_n) = \sum_{i,k} E_l(x_i, x_k \mid a_n) \tag{2.3}$$

For yet-to-be-decided pixels, do:

$$E_2(X, \Psi, Grad, a_n) = \sum_{i} E_g(x_i, \Psi \mid C(Grad_i, a_n)) \tag{2.4}$$

where E_1 maximizes the smoothness of known pixel x_k to unknown neighboring pixel x_i; given the constraint value a_n, E_2 maximizes the applicability to the model Ψ; given the constraint function $C(Grad_i, a_n)$ based on the gradient of that pixel $Grad_i$ and the constraint value a_n.

In other words, first, local region growing is performed, given a statistical constraint value used as a threshold. Then only for pixels that cannot be decided using region growing, global classification is checked. In contrast with region growing, the global scheme has no pixel difference to be used as a threshold, and classification of each pixel is constrained by its gradient, given the same constraint value (its relation will be explained in the next section). The higher the gradient is, the stricter the decision is. In this way, optimizations are independent but linked to each other by the constraint, leading to computationally efficient algorithms.

We first proposed the notion for a foreground cut in [ROI06], targeting 3D modeling, and used it for our robot painter program [RKO*07]. However, at that time only global data distribution was considered and the constraint setting was not effective, which resulted in a time of about 10 seconds for a 640*480 image on a laptop with a 1.8GHz CPU. CIFS generalizes and extends [ROI06], and is used for the disparity image cut in [RKOSI07]. At present, the computation time is reduced by more than ten times.

It will be shown here that this scheme is not sensitive to initialization. The constraint of edge is used as a state indicator to perform multiple cuts automatically. More importantly, to perform cut and matting, timing can be used to stop the search process effectively before matting. The constraint, which was not effective in previous methods, is made possible by "Chebyshev's inequality" (see Appendix A).

Apart from the core idea of optimizing a cost function of feature similarity and using a constraint of edge strength, another key aspect of the proposed method is the use of the algorithm that can consider local and global optimization at the same time. This research proposes to employ the tradeoff between local and global constraints.

Hence, first, the next section explains how to estimate and update the edge constraint as well as how to weight between local and global information, and how to use the proposed algorithm to cut a foreground.

2.2 Theory and an Application for a Foreground Cut

Now, we shall see how to consider both local and global cost function at the same time with edge constraint.

From marked input, the search process will try to identify unknown pixels. Since the algorithm considers edge information as a constraint to stop the search process, if the constraint is too large, foreground/background regions can grow incorrectly into each other. So Chebyshev's inequality, which covers all data distribution models, is applied to initialize the constraint.

As both the local and global information are used, one can imagine that for constrained optimization, there must be two constraints for each of them. This section, first, explains how to derive "constraint value" for "local region growing" from color distributions. Although the best color domain is CIEL*a*b*, many works use the RGB domain. Hence, in order to benchmark against them, this research uses the RGB domain. Also, we tested and found that the segmentations do not depend very much on the choice between these two color models.

On the other hand, "constraint function" for "global classification" and its relation to the constraint value will be described later in this section. Global optimization could be "any classification methods" that can give continuous alpha value between 1 (absolute foreground) and 0 (absolute background). One important point that is different from other energy optimization schemes is how to trade off between local and global information. This "smart weighting" is embedded into the constraint function.

Regarding the algorithm, first, local region growing will be used. If region growing is unable to decide on some unmarked pixel, global classification is then performed. After searching forward and backward on the image array $X = (x_1, ..., x_N)$, the present constraint value a_n will be updated to a_{n+1}. This updating is modeled by a mixture of Gaussians. The process continues until there are no remaining unknown pixels.

1) Constraint Initialization

In order to derive the initial constraint value for region growing so that the foreground and background grow correctly, a statistical method is used to model the input image's foreground/ background distribution. As opposed to conventional methods that try only to form a threshold based on first order moment [FSA07] that is weak when the image is complex, classical Chebyshev's inequality is appropriate to be used for calculating how far an arbitrary point in the color domain can grow to neighboring colors, by directly considering the distribution.

First, we calculate the color standard deviation of the foreground, the background, and of both areas. After that the constraint value can be calculated from (2.6).

Proof. Based on Chebyshev's inequality, at least 99% of all samples will fall within +/-

10 of the standard deviation. This range can be considered to be like a box in the RGB domain with a width, height, and depth of $20\sigma_R$, $20\sigma_G$, and $20\sigma_B$. Next, further consider the distribution boundary to be a sphere with a diameter determined in equation (2.5). The claim that at least 99% of all samples are within the box will hold for this sphere. Although the sphere is larger than the box, a small difference from 99% is not statistically important, as the actual population needed is 100%.

$$\sqrt{400\sigma_R^2 + 400\sigma_G^2 + 400\sigma_B^2} = 20ssd \tag{2.5}$$

Based on this spherical standard deviation, ssd, of foreground, background, and both areas, calculate the region growing constraint from (2.6). Note that the factor of 20 can be ignored since the relation between each parameter is linear.

$$a_0 = \max(k_{scale} * (ssd_{fb} - \max(ssd_f, ssd_b)), k_{\min}) \tag{2.6}$$

In (2.6), constant $k_{\min}$ enables a region to grow for the case in which the intersection between foreground and background is too tight, and is set to unity as this represents the smallest color difference in a digital image. There is also a scaling factor, k_{scale}, whose value is important. Too small a k_{scale} will make the growing process unnecessarily slow, while too large a k_{scale} can cause erroneous growing of the foreground into the background, or vice versa. According to Chebyshev's inequality, it is possible that almost none of the samples will fall within +/- 1 of the standard deviation, here ssd_{fb}, so that the ideal optimum constraint from foreground that would not intrude into the background, and vice versa, would be $1 * ssd_{fb}$. Note that generally the outcome of (2.6) would be biased to a smaller value than $k_{scale} * ssd_{fb}$ statistically to compensate for non-ideal distribution. As we are going to use the timing that $k_{scale} * ssd_{fb}$ equal to ssd_{fb} and $2 * ssd_{fb}$ for multiple cuts and cut before matting, as will be shown in Sections 4 and 5, respectively, k_{scale} must be lower than 1, which will not affect the performance much as it only results in more iterations. In our experiments, it is set to be 0.5, which proved to be robust for all 40 test images.

2) Local Classification

Based on the optimum constraint value, region growing is performed by calculating pixel differences from the user's marked core data. If the Euclidean distance in the color domain between the known foreground/background and its surrounding unmarked pixels is smaller than the constraint value a_n, an unknown pixel will be changed to foreground or background.

3) Global Classification

Translating our global optimization idea into a classification criterion, since the alpha value ranges between 1 and 0 for foreground and background, respectively, any yet-to-be-marked pixel is judged as foreground or background using the criterion of (2.7).

$$x_i \text{ be} \begin{cases} \text{foreground} & ;\alpha_i \geq 0.5 + C(Grad_i, a_n) \\ \text{background} & ;\alpha_i \leq 0.5 - C(Grad_i, a_n) \end{cases} \tag{2.7}$$

where α_i is the alpha value of pixel i, and $C(Grad_i, a_n)$ is the constraint function calculated from gradient information of pixel i, given the region growing's "constraint value" for iteration *n* is a_n.

For alpha value calculation, presently, in order to calculate the alpha value, the foreground cut uses statistical clustering functions such as the Gaussian Mixture Model (GMM) in [RKB04] or k-means clustering as in [LSTS04] or strong nonlinear classifier as in our previous work [ROI06]. In the present work, k-means is used to compare with graph cut implementation of [LSTS04], and neural network is used for the case that input consists of multiple features. For now, alpha value from k-means is calculated from:

$$\alpha_i = \frac{\left\| I_i - \mu_b^c \right\|}{\left\| I_i - \mu_f^c \right\| + \left\| I_i - \mu_b^c \right\|} \tag{2.8}$$

in which I_i is the RGB value of pixel i and μ_b^c is the closest cluster mean of the background (or foreground) RGB.

For constraint function, as opposed to constraint value, the two constraints must contain some relation, as region growing compares two neighboring pixels whereas classification compares the pixel and the data distributions. This will be explained in the next section.

4) Tradeoff Between Local and Global Data

In previous works that consider complex cost function, the decision to weight between inside parameters faces the problem of inconsistency from cost function to cost function. There have been attempts to generate such a weighting parameter [CKS97] [DM00]. On the contrary, if the optimization is constrained, it is possible to trade off each and every constraint directly. This is unexpectedly robust and very easy to implement.

By considering priority, global classification should be given lower priority than local region growing. Hence, the constraint for global classification in (2.7), $C(Grad_i, a_n)$, should be: 1) 0.5 when a gradient value is strong, indicating that the unknown pixel is probably around the boundary. Regarding priority, the constraint should be high when the gradient is higher than the region growing's constraint. 2) Gradually relax to 0 as the gradient value become smaller than the region growing's constraint.

At first, we used Haar-like function as the constraint and found that it cannot produce a good result, evidently because it violates the second condition mentioned above. It is discovered from experiments that, similar to classical nonlinear classification, the effective group of function is variants of sigmoid.

In order to make the relation to region growing's constraint linear so that increasing one will affect another equally, sigmoid is adapted to be a linear function of a_n.

$$C(Grad_i, a_n) = \max(\frac{1}{1+\exp(a_n - Grad_i - w)} - 0.5),0) \tag{2.9}$$

Equation (2.9) is a scalable version of a sigmoid function in which the constant w is the point where constraint function approximately reaches maximum value, here 6.75. This is actually our method's weighting parameter. The higher w is, the more weight is given to local similarity. The effect of increasing a_n or reducing w in the gradient-constraint domain is similar to shifting the sigmoid function rightward as shown in Fig. 2.1.

Among the advantages of this weighting are the following five: 1) It is easy to implement. 2) It is compatible with existing schemes. 3) Prioritizing is straightforward; it tries not to decide on pixels where the gradient is stronger than a_n, which means the global classification has lower priority than local region growing, and passing the decision if the gradient is smaller than $a_n - w$. Between these two milestones, the decision is based on nonlinear function, here sigmoid. 4) Both constraint value a_n and constraint function $C(Grad_i, a_n)$ are always consistent as they are derived from data distribution. The reason that both can be based on Chebyshev's inequality is that the initial constraint a_0 reflects both color difference and gradient. 5) It is extendable to other features without any adaptation. In multidimensional features, region growing should use alpha value instead of color, as will be shown in the discussion section. Consequently, a_0 will be calculated from alpha distribution which then reflects alpha difference and gradient in alpha domain.

Note that when the feature is only color, region growing does not have to use an alpha value instead of color. Also this can reduce the inconsistency of alpha value calculation due to tight color distribution or not-well-structured classifier. This advantage can be seen from the second row of Fig. 2.3.

In this scheme, one important thing to stress is the difference from merely weighting the cost function, as the latter is difficult to derive, especially when the cost function is complex, as when weighting between local and global information.

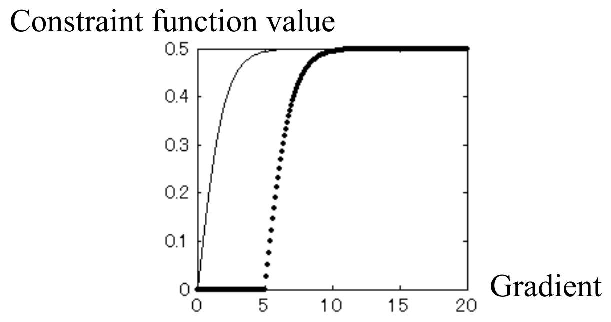


Fig. 2.1 Constraint function. $a_n - w$ = 0 and 5 shown in two lines, respectively.

5) Constraint Update

After the region cannot grow further using the present constraint value and constraint function, the constraint value will be relaxed from a_n to a_{n+1}. As opposed to the constraint initialization that does not model the distribution, the relaxing process is modeled based on the mixture of truncated symmetry Gaussians in features domain with the width of 6σ. The Gaussians are symmetrical and have a fixed size, as the constraint is unbiased on axis and actual distribution in the features domain.

At the end of iteration, assuming the worst scenario, the Gaussians expand to the width of 6σ and touch each other perfectly, meaning some Gaussians of the foreground are touching those of the background. In the next iteration where the constraint will be relaxed and the Gaussians will likely expand further, the optimal intruding would be σ, according to the less insignificant part of the distribution. So the optimal relaxing factor is $1.3\dot{3}$. This is iteratively done at every iteration end.

Note that by performing a segmentation experiment using only local region growing, this factor also serves the method very well.

In global optimization, the constraint function $C(Grad_i, a_n)$, which has lower priority, is relaxed for the same portion. The process continues until there are no remaining unknown pixels.

6) Experimental Result

In order to compare our proposed method with the current technology of foreground segmentation, a method that exploits both local and global information, graph cut, is used as a benchmark. To make the comparison fair, both methods apply the watershed algorithm [VS91] to generate pre-segment input images and use k-means as the classification agent with the same number of clusters.

For the case that the boundary is clear and the foreground and background data distributions are well separated, both graph cut and CIFS can segment the foreground effectively, as can be seen in the first row of Fig. 2.2. On the other hand, region growing easily fails.

If the data distributions are ambiguous and there are strong edges that are not the boundary, graph cut often fails whereas our smart constraint exhibits improvement as can be seen in the second and third rows of Fig. 2.2. This does not mean that our method is better than graph cut in all aspects as we are still relying on the choice of the constraint function $C(Grad_i, a_n)$. A better outcome of graph cut can be expected if more weight is given to global information.

In any case, even when the marks for foreground and background are close to each other, Chebyshev's inequality proves to be robust in preventing the foreground from incorrectly growing into the background, and vice versa, as can be seen from column (d).

The overall process, without the pre-segmentation step of watershed, typically consumes less than one second for a 640*480 image on a laptop with a 1.8GHz CPU. The time consumption is linear to the dimension of the images. Although the time consumption is reasonable, this is not faster than graph cut. Since even the notion of independent local and global classification leads to algorithmic efficiency, the iterative constraint produces delay in segmentation. However, this constraint will be shown to have outstanding merits in the next two sections.

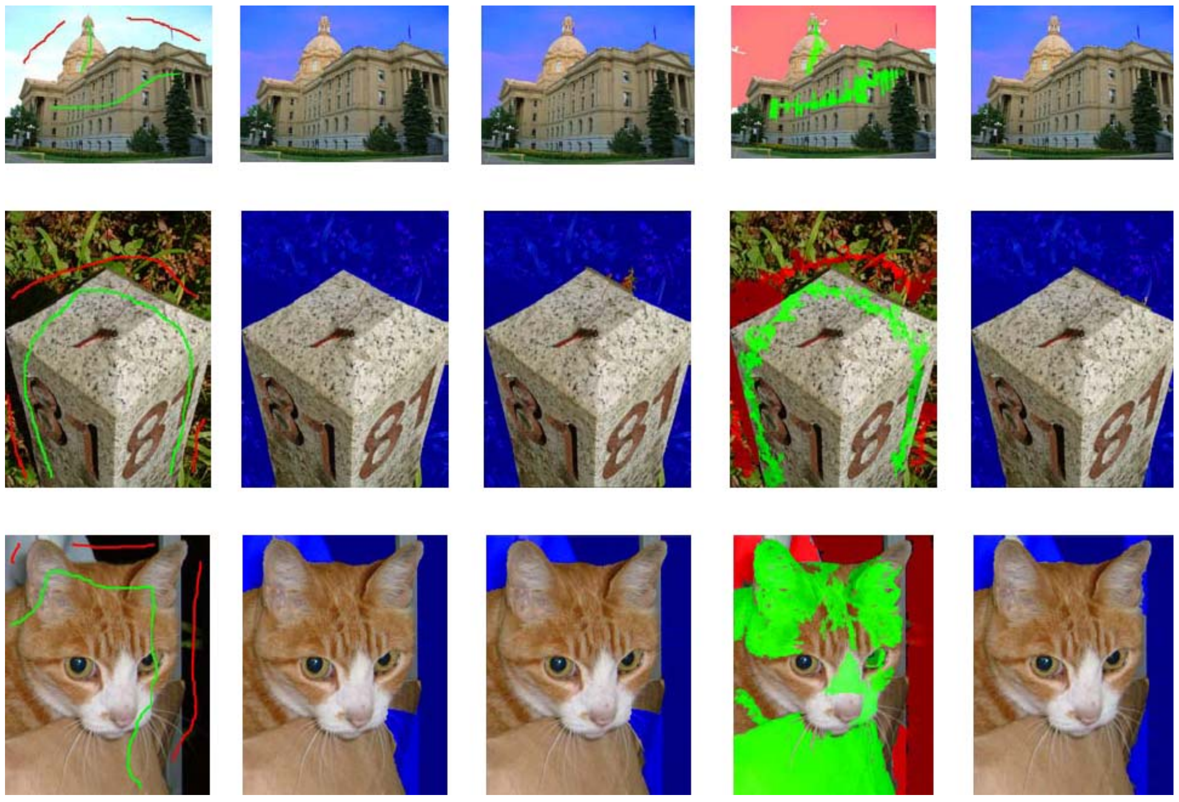

(a) (b) (c) (d) (e)

Fig. 2.2 Comparison of foreground cut tools.
(a) Input image. (b) Graph cut. (c) Applied region growing with Chebyshev's inequality.
(d) At constraint based on Chebyshev's inequality. (e) CIFS.

2.3 Multiple Cuts

One advantage of the proposed method is the use of edge information as a constraint. Not only does this improve the problem of trading off between color similarity and edge strength but this also enables multiple object searching without having the user mark all the separate objects in an image. Searching too early in remote areas results in false positive objects, as can be seen in Fig. 2.3 column (b); searching too late might miss a chance to distinguish remote sections if they were already incorrectly detected as shown in column (c). This indicates that previous approaches are inferior to our system in the sense that they do not possess the notion of growing to some edge constraint before splitting.

1) Remote Search Criterion

Automatic searching is possible in our algorithm by letting the region grow to a statistically certain iteration. Since it is possible that there are remote sections that should be judged to be foreground (or background) but are blocked by background (or foreground), without multiple section mode these areas would never connect to the foreground (background), as shown in Fig. 2.3. So instead of letting the region grow further, we start searching in remote unmarked areas for pixels that fall into the global data distribution of the foreground (or background) but not vice versa. These detected pixels are used in the next iteration as additional cores.

Statistically, the timing limits for searching can be obtained by considering the lower bound of Chebyshev's inequality. According to the theory, the ideal optimum constraint from foreground that would not grow into background, and vice versa, is when k_{scale} equals 1. As k_{scale} was set to 0.5, so the remote section search process started when $a_n \geq 2a_0$. We can use this to search for remote sections, and must do so, because if we do not constrain the timing, remote foreground (background) objects will be incorrectly turned into background (foreground) in the consecutive iterations.

At the end of this iteration, all the remote sections are detected and marked as new cores. Then the process continues as usual until no unknown pixel is left.

2) Experimental Result

Multiple cuts mode is benchmarked against an object recognition scheme similar to [MFM04] and [FO03]. Although [FO03] exploits high dimensional training input that forces the use of a comparatively fast but weak classification method, we have found that this larger window is slow and does not always produce good recognition results. Especially when the foreground/background consists of many different regions, using a larger window usually confuses the classifier because it encounters patterns that are too complex. Hence, recognition is done on a pixel basis here. To simplify the comparison, both methods apply the watershed algorithm to generate pre-segment input images and use k-means as the classification agent.

From the result in Fig. 2.3, first, the first row shows a well-separated distributions case where both approaches perform equally. The second row presents a not-well-structured classifier (including underfitting and overfitting). Here, the background contains clear color but the structure of the classifier is over-complex so that the result of classification is prone to error. On the other hand, using the same classifier, multiple cuts mode gives a robust result due to exploitation of local region growing.

Even when the color distribution is not well separated, as can be observed in column (b), multiple cuts mode exhibits robustness to some extent. The reason is as mentioned: exploiting Chebyshev's inequality as a constraint enables the region to grow to the critical statistical limit shown in column (d) before searching for remote sections resulting in column (e). Note that the method inevitably shares similar weakness with object recognition and the contour evolving approach when the foreground/background features are too ambiguous, as can be seen in the third row. However, considering the final row, in which the intersection between foreground and background is also tight but not as complex as that of the third row, CIFS shows substantial improvement.

Regarding time complexity, multiple cuts mode consumes approximately the same as that of a foreground cut mode.

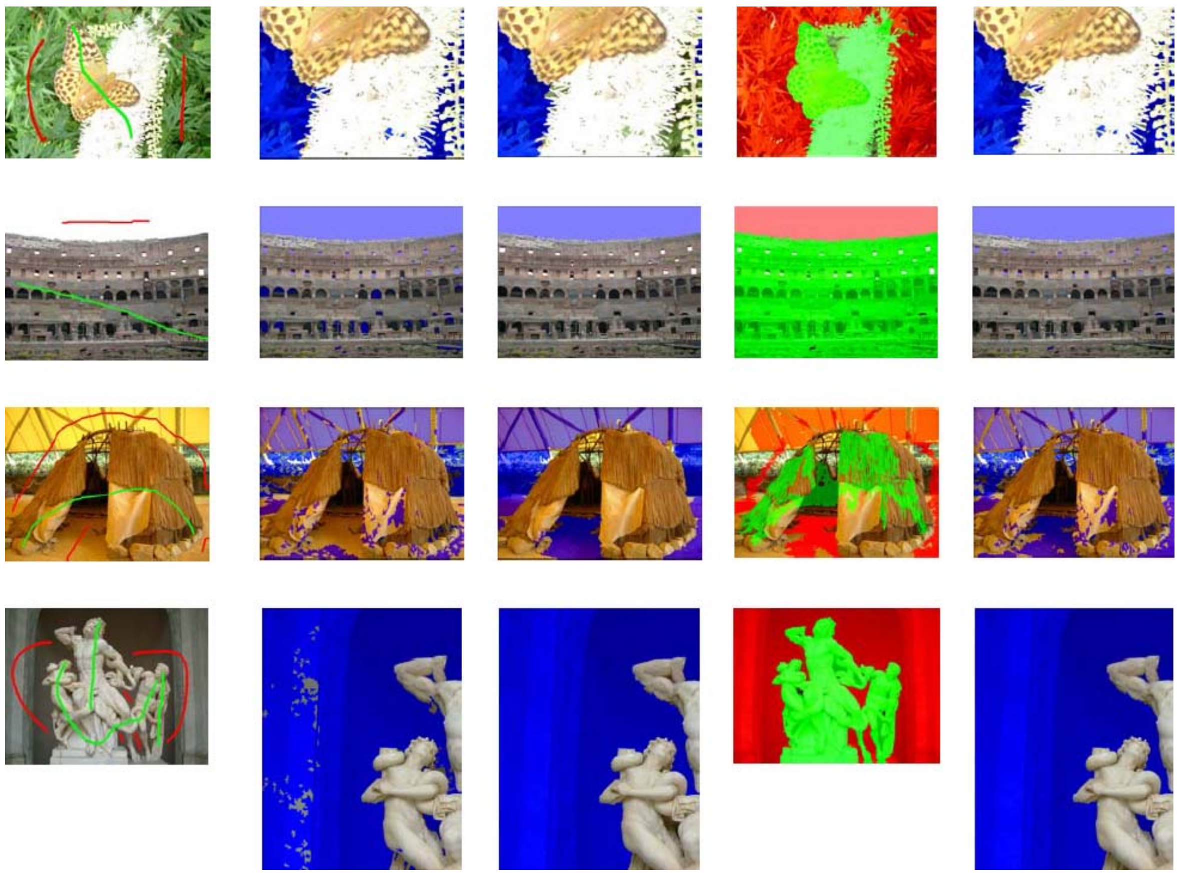


Fig. 2.3 Comparison of multiple cuts tools.
(a) Input image. (b) Object recognition. (c) Foreground cut mode. (d) At twice the constraint based on Chebyshev's inequality. (e) Multiple cuts mode.

2.4 Cut Before Matting

Presently, matting is often required in image editing, as subjects to be segmented from original images often contain small or thin sections. However, a great deal of user interaction is usually required to mark areas around the boundary manually. Hence, this problem is important because it attacks such considerations as whether it is possible to, first, use high-end foreground segmentations to generate pre-cut images, and then, use matting tools to generate a soft boundary. In other words, can the process be automatic? This would reduce user interaction, which would be a great benefit to people in media fields.

1) Cut before Matting Criterion

Although many matting tools are available in both commercial and research format, using pre-cut images is often required for difficult images where data distribution is not well separated. [RKB04] propose to use their graph cut implementation to cut, and then perform erosion to some heuristic iterations, before using a matting tool like [CCSS01] to generate a soft boundary. [WC05] sees this approach as an ad-hoc decision, and proposes the use of belief propagation to iteratively revise the matting process. Our work, though not perfect, shows improvement over the mentioned schemes even when the distribution is ambiguous.

Similar to multiple cuts mode, cut before matting mode relies on the process of growing to some certain statistical constraint. Naturally, the pre-segmentation step of watershed is omitted since it would generate undesired blobs in the thin sections. Consequently, region growing can produce undesired growth due to noises in the original image (without watershed smoothing). Hence, region growing is omitted in the early iterations where $a_n < 2a_0$, based on the previous section analysis.

Again, the timing limits for searching can be obtained by considering the lower bound of Chebyshev's inequality. Whereas multiple cuts strictly stops before search at the constraint that "would not allow foreground and background to grow into each other" (ideally equal to $1 * s_{fb}$), cut before matting can tolerate more since it is not concerned with losing remote sections. Hence, it is designed to stop before matting at the constraint that just "would not allow foreground to grow into existing background's data distribution, and vice versa " (ideally equal to $2 * s_{fb}$). As k_{scale} was set to 0.5, so the remote section search process started when $a_n \geq 4a_0$.

After this iteration, constraint function (2.9) is set to be zero, meaning that decision is based on local region growing with a constraint value or on alpha value directly (without constraint function). Even when a pixel is judged using region growing, its alpha value is recorded to represent composite images automatically.

Additionally, as opposed to some works that use a fixed size large window for searching, CIFS increases the size of such a window at the end of iteration. This is done because using a fixed size window gives a similar result but consumes considerably more time.

2) Experimental Result

In this experiment, three types of images are used, namely, uniformly distributed colors, well separated colors, and tightly separated colors.

For all these kinds of images, CIFS shows a better result than robust matting [WC07], which is a state-of-the-art matting program, in the sense that it is less sensitive to initialization, as shown in Fig. 2.4. However, it can be observed that in the sense of "softness" of the boundary, a specific matting program can do its job slightly better.

However, when the color distributions are not well separated, as can be seen in the third and fourth rows, CIFS gives substantially better results. Chebyshev's inequality does not fail us regardless of how dense the distribution is. This kind of result would not be obtained if a conventional erosion before matting scheme were used, as a small number of erosions would not cover the size of the boundaries, and a large number would give pre-cut images that are too far from the boundaries and, in the case of ambiguous color distributions, would result in completely wrong cuts.

Considering the difficulty in marking the images, it can be seen that users do not have to carefully trace the hairs. They can just roughly place a mark around the subjects.

Time consumption of this mode is, of course, larger than the previous two modes since a large search window is used. The overall process typically consumes less than ten seconds for a 640*480 image on a laptop with a 1.8GHz CPU. This is generally as fast as robust matting [WC07], which also tries to address the problem of automatic matting.

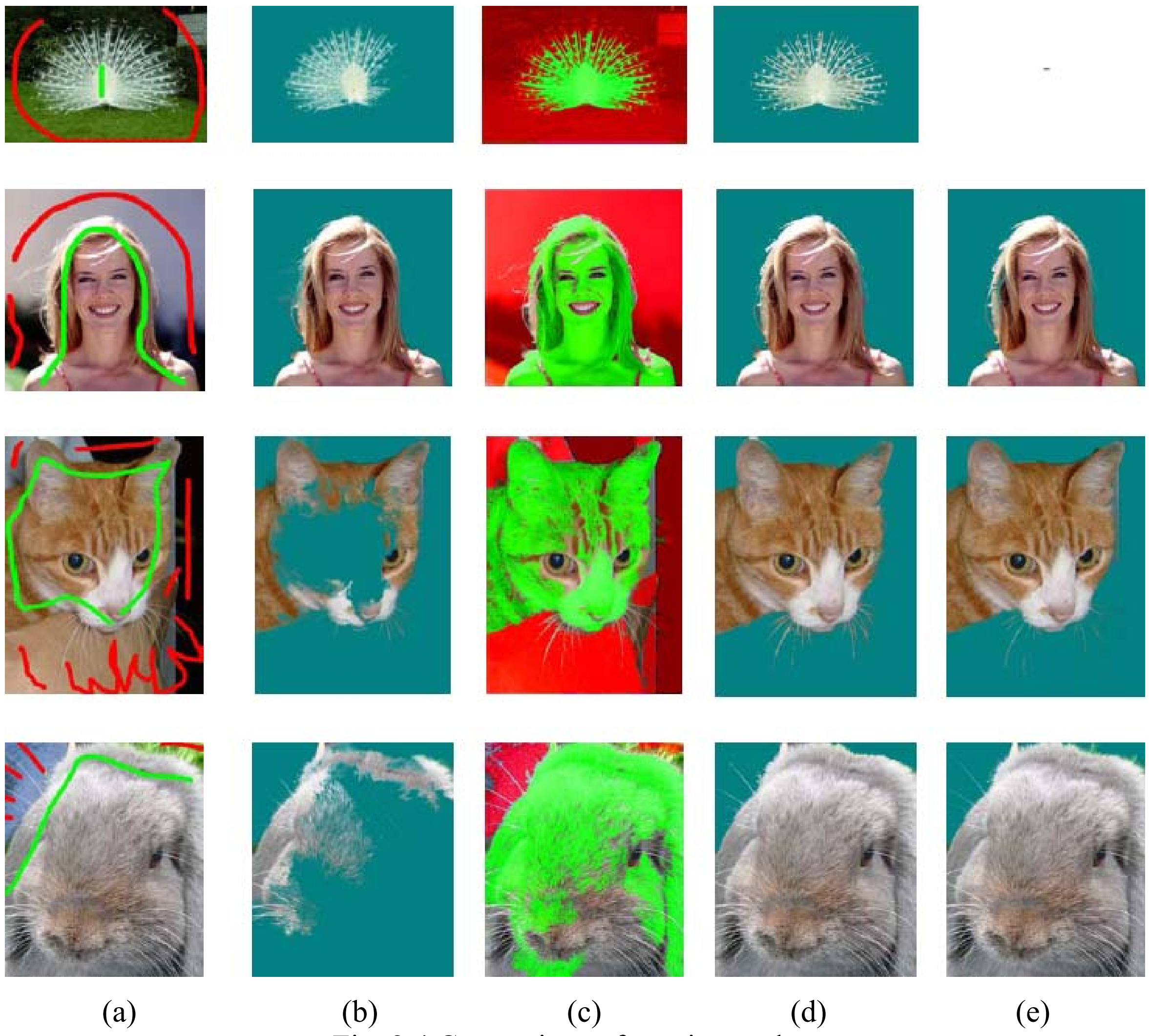


Fig. 2.4 Comparison of matting tools.
(a) Input image. (b) Robust matting. (c) At four times the constraint based on Chebyshev's inequality. (d) Cut before matting mode. (e) Cut before matting in (c) used as input for other high-end matting method.

2.5 3D Object Segmentation

Using normal cameras, for example, 280x200-pixel cameras in a robot's head, after disparity map and 3D background subtraction, the extracted foreground is usually, if not always, noisy, as shown in Fig. 2.5.

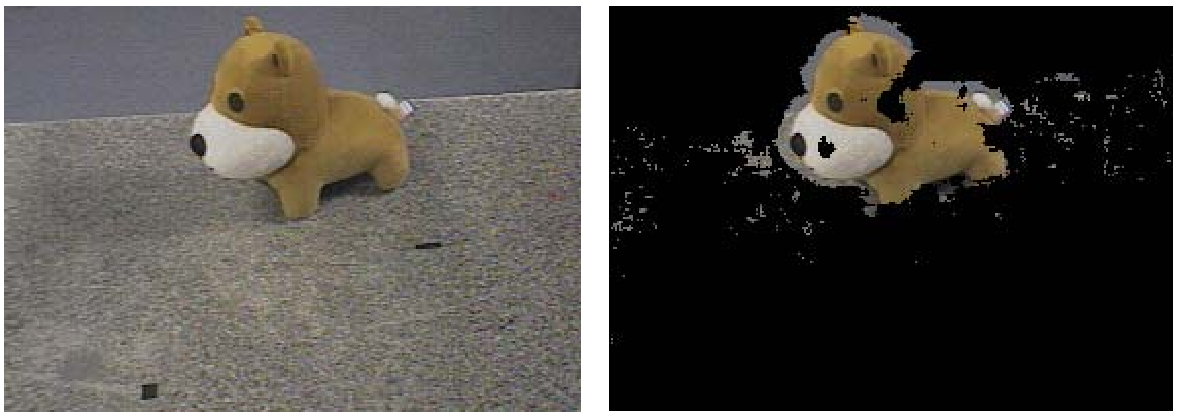
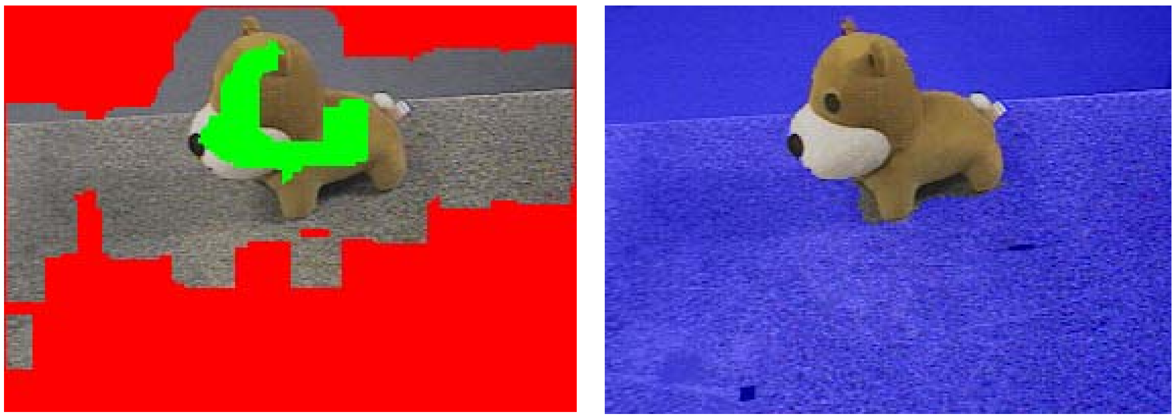


Fig. 2.5 Conventional stereo segmentation method.
(a) An image captured by a camera attached to the robot in Fig. 1.1. (b) Disparity map and background subtraction.

We then notice that interactive foreground segmentation can be used to solve this problem, as described in [BK04] [WBC*05] [LSS05] and [ROI06], which require a user to roughly mark foreground and background. Subsequently, the correct foreground can then be extracted using their algorithms.

In contrast with such methods, instead of letting a user mark foreground and background, this research directly exploits the extracted foreground shown in Fig. 2.5 (b). As the data is noisy, dilations and erosions are performed on foreground and background. The number of dilations and erosions is fixed for each calibration. This automatically marked image is shown in Fig. 2.6 (a).

Then, this image is used as an input for our 2D CIFS, the result is shown in Fig. 2.6.



(a) (b)

Fig. 2.6 Stereo segmentation using 3D data as an input for our method.
(a) Input from dilations and erosions on Fig. 2.5 (b). (b) Object segmentation result.

2.6 Discussion

This section describes comparative studies from the aspects of mathematics and algorithms. Among the important aspects considered here are sensitivity to initialization, robustness, noises in multiple cuts, cut area before matting, matting itself, multidimensional features, and user interaction.

1) Sensitivity to Initialization

CIFS, similar to graph cut, is evidently robust against initialization. The reason is quite obvious; both of them exploit local and global similarity at the same time. On the contrary, matting and edge-based approaches are often criticized for sensitivity to initialization. Matting is robust and capable of generating a soft area locally, but when it comes to large areas, not-well-separated feature distributions in images force the alpha classification to the corner. For the latter, having the flow process rely on gradient information is weak when the boundary area contains a large number of edges. Unsupervised region-based approach is also robust against initialization, [CRD06], but it is difficult to make this comparison since the user input is different.

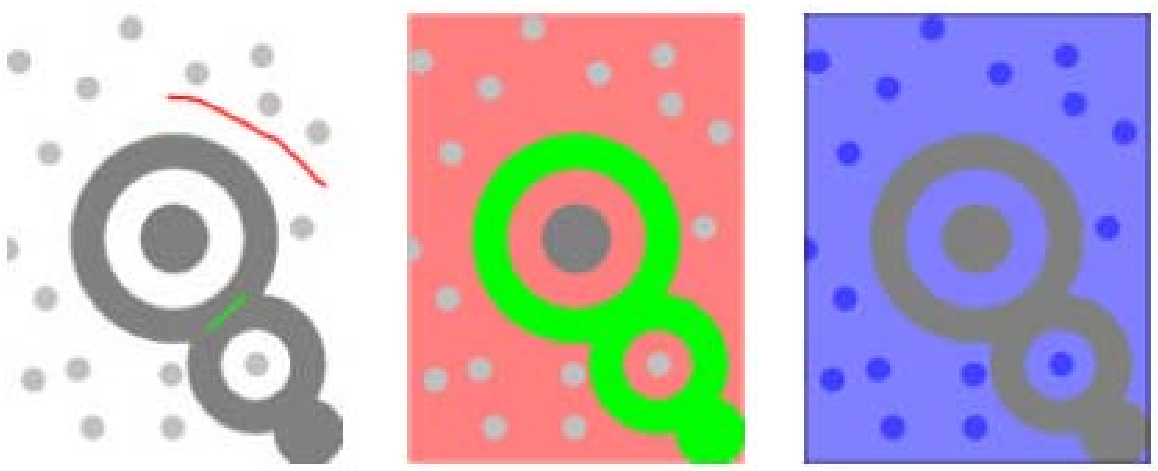
2) Robustness

Furthermore, the use of both local and global similarity also leads to robustness against tight data distributions, which is one drawback in the matting and unsupervised edge-/region-based approaches. While the reason that such a drawback exists in the matting approach is immediately understandable, as the method relies considerably on feature distribution, the cause for the unsupervised approach is not so clear. For the boundary to evolve, a user is required to draw a guideline roughly. This is an unforeseen defect. Users cannot mark areas as being foreground and background, which means that they more or less lose control of the topology. Also, the initial data distribution would never be given, which also means that the cost function cannot exploit similarity to data distribution, since this is not given.

3) Multiple Cuts

Whereas the unsupervised edge-based approach is weak in images with noisy gradient information, CIFS, fortunately, is very similar to the unsupervised region-based approach. The latter two give noisy results when the feature distributions are not well separated. The unsupervised region-based approach can reduce this noise by imposing a cost function on the geometry of an evolving boundary whereas our approach grows to reduce noise before searching for remote areas.

Although the unsupervised region-based approach is more flexible since the cost function can be adapted whereas our method relies considerably on the data distribution, a more important feature than flexibility is robustness. Although applying internal force in such approach may reduce noise, it can also limit geometric flexibility and prevent the representation of thin objects [MT95]. It is difficult to say which one performs better since the input is different. Ultimately, in the case that feature distributions are well separated, multiple cuts can segment the foreground correctly without having a user mark all areas, as shown in Fig. 2.7. Even in not-well-separated cases, it shows robustness to noise as can be seen in Fig. 2.3.



(a) (b) (c)

Fig. 2.7 Multiple cuts mode with noise.
(a) Input image. (b) At twice the constraint based on Chebyshev's inequality. (c) Result of multiple cuts.

4) Cut Before Matting

It can be clearly seen that the proposed method gives a substantially better cut since the regions grow from both core foreground and core background, before stopping at the timing based on Chebyshev's inequality. Combined with the notion of local and global optimization, this is a great benefit not available in previous works.

Moreover, this mode can be used as a standalone matting program as well as an input generation for other high-end matting methods. In this regard, evidently, in difficult images from Fig. 2.4, if users prefer correctness to softness, cut before matting is recommended as a standalone matting program.

5) Multidimensional Features

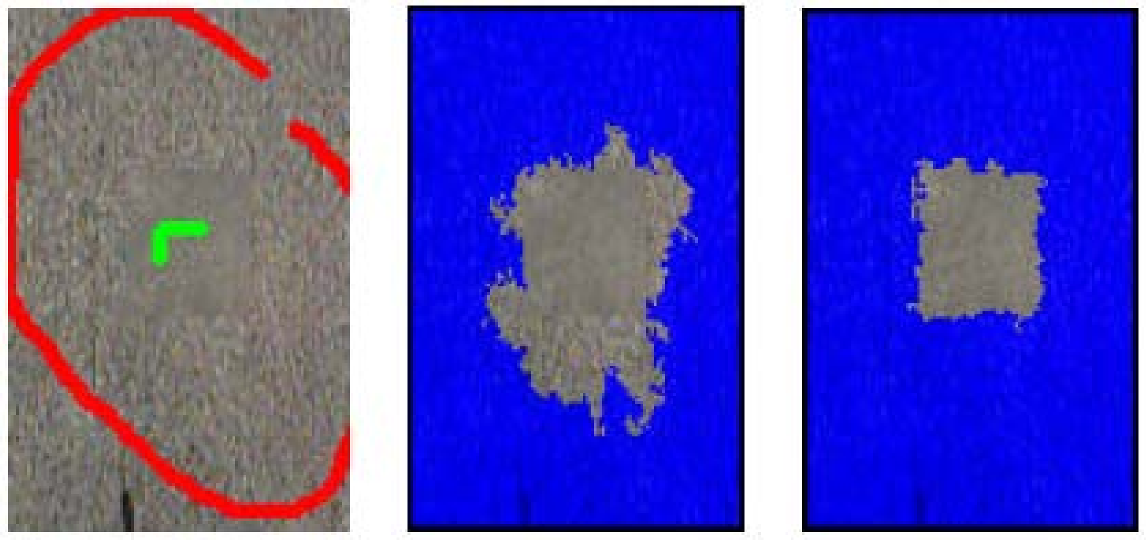
When exploiting other input along with color, for example, texture, there are generally two ways to classify these complex features. The first scheme is by weighting between color similarity and other features' similarity. The difficulty in this scheme is how to decide the weighting parameter. Many previous works select it by trial-and-error and face the problem of inconsistency. The second scheme lets the pattern classifier do the job, but this scheme relies on the performance of the chosen classifier. CIFS employs this latter scheme.

In order to take other features into account, whereas global optimization already relies on an alpha value that can be generated from all features, the input for local region growing must be changed from color value to scaled alpha value, without loss in generality as explained in Section 3. Also, the pre-segmentation step is not applied here as it would filter out texture information undesirably.

Based on its main strength, effectiveness in classification, a neural network with one hidden layer is used in this work, instead of clustering schemes that cannot provide meaningful discrimination power in such complex input domain. The Levenberg-Marquardt training algorithm is chosen since it converges well. As alpha value ranges between 0 and 1, sigmoid function is used as a transfer function. The user-marked data are then used to train the network for 100 epochs.

In the experiment, images with very tight color distributions are used. For example, in Fig. 2.8, even a human can hardly see the rectangle in the middle. By exploiting texture here, using conventional local variance of a five-by-five window, a gross area of the object can be detected.

In spite of the fact that the extracted boundary is not smooth since CIFS does not have an internal cost function, one can clearly see the effectiveness of the method to enhance an object.



(a) (b) (c)

Fig. 2.8 Using a neural network with various features to generate alpha value.
(a) Input image. (b) Using RGB. (c) Using RGB and variance.

6) User Interaction

Finally, as mentioned earlier, user interaction also plays an important role in the segmentation algorithm. If the segmentation is not perfect, it can be corrected more easily in the interface. "Mark areas as being foreground vs. background" is an easier task than "mark a guideline contour."

In addition, CIFS offers two important options that can greatly reduce user interaction: multiple cuts and cut before matting.

2.7 Algorithm

```
TrainClassifier();
r = 1;
an = a0 = ConstraintInitialization();
if (modeMatting)
        doRegionGrowing = false;
else
        doRegionGrowing = true;

// In each iteration, running forward or backward in image domain //
while (n != 0)
{
        if (mark[i][j] == surroundedByUnknownPixel)
        {
                surround = false;
                for (x = i-r; x< = i+r; x++)
                {
                        for (y = j-r; y< = j+r; y++)
                        {
                                if (mark[x][y] == unknownPixel)
                                {
                                        surround = true;
                                        undecided = false;

                                        // Region growing //
                                        if (doRegionGrowing)
                                                undecided = CriterionRegionGrowing();

                                        // Global classification //
                                        if (!doRegionGrowing || undecided)
                                        {
                                                alpha = FindAlphaValue();
                                                CriterionGlobalClassifiacation();
                                        }

                                }
                        }
                }
                if (surround == false)
                        mark[i][j] = !surroundedByUnknownPixel;
        }

        // Stop criterion //
        n0 = n;
        n = 0;
        for (i = 0; i<row; i++)
        {
                for (j = 0; j<col; j++)
                {
                        if (mark[i][j] == unknownPixel)
                                n++;
                }
        }
        if (n == 0) break;

        // Special mode //
        if (evenIteration || n == n0)
        {
                if (modeMultipleCuts && an >= 2*a0)
                {
                        SearchRemoteArea();
                        modeMultipleCuts = false;
                }
                else if (modeMatting){
                        r++;
                        if (an >= 4*a0)
                                doRegionGrowing = true;
                }
                an *= relaxedGain;
        }
}
```

2.8 Summary

This research presents a supervised comprehensive iterative foreground segmentation (CIFS) based on local and global optimization with edge constraint that can effectively deal with a foreground cut, multiple cuts, and cut before matting. It uses the cost function similarly to graph cut but extends the algorithms by the notion of edge constraint. The constraint that was not effective in conventional region growing is made possible by Chebyshev's inequality and proves to be robust. This is the first novelty.

Since both local and global cost functions have their own constraints, a novel weighting method is shown to perform in the constraint domain instead of in the cost function. Although it is difficult to compare this notion with conventional weighting, our scheme is more straightforward if there are local and global components in the cost function. This notion can also deal with multidimensional features.

Considering a foreground cut, CIFS is as robust as graph cut, since their cost functions are quite similar. Nevertheless, graph cut, in difficult images, applies only to the foreground cut problem.

The third novelty is automatic multiple cuts that are capable of reducing a great deal of user interaction. This approach is robust against a not well-structured classifier as well as ambiguous data distribution. The notion of growing to an edge constraint based on Chebyshev's inequality before searching is novel and can reduce a great deal of error. This is completely different from the contour evolving approach. Compared to this approach, the main differences are among the algorithms: the proposed method does not split or merge to perform multiple cuts. CIFS relies on data distribution and thus is more straightforward, whereas the other approach seems to be more flexible due to its internal and external cost function.

For cut before matting, the proposed method can generally gives substantially better pre-cut images since it uses Chebyshev's inequality as opposed to heuristic erosion. The scheme can be used as a standalone matting program as well as an input generation for other high-end matting methods.

The final novelty is that the method is then applied to perform 3D object segmentation automatically. In order to extract the subject area, this research focuses on how to exploit normal stereo cameras to roughly extract the object automatically using disparity map and 3D background subtraction, and then uses CIFS to extract the object area correctly. 3D background subtraction is usually noisy; thus, dilations and erosions are required. CIFS, then, exploits the local and global color similarity optimization with a constraint of edge to extract the boundary correctly. This is a promising paradigm, which can be applied in wide varieties of 3D segmentation/detection tasks.

Chapter 3

Color Perception

After the object boundary is derived, the robot must represent the object's color using an appropriate set of colors. This chapter attacks the problem of whether it is possible for a robot to perceive colors in an image in the same way a human does.

The question falls into the domain of color reduction. Usually, digital color images consist of up to 16 million different colors in a 24-bit color space. However, in many applications, namely compression, presentation, and transmission, it is preferred to have as small a number of colors as possible. Color reduction is a process that transforms a full color image to an image with a smaller number of colors, by grouping similar colors and replacing them with a representative color.

Our research proposes a color reduction scheme that incorporates two clustering methods, maximum distance clustering (MDC) and K-means. It shows that, using MDC + an iteration of K-means, in RGB color space, the result is better than using K-means alone. This results in high-quality, well-contrasted output images; and even the reduction in the number of colors is very low. We also solve the speed problem of MDC using a proposed sub-optimal algorithm. Then it is shown that behavior of clustering schemes in CIEL*a*b* color space is different from that in RGB color space.

Another objective of this research is an algorithm that anyone could easily implement, and this is already achieved since our improvement is based on well-known and easy algorithms, MDC and K-means.

For benchmarking, we use Photoshop's perceptual-based palette generation [AP7], region-based color segmentation of watershed [VS91], and clustering-based method of normal K-means. It will be shown that, in RGB color space, although MDC + an iteration K-means performs better than K-means, however, in CIEL*a*b* color space, both approaches are comparable if K-means is performed forward and backward on image coordinate.

This key problem of high-quality interactive color reduction is what our method can directly address. Section 1 shows some related works. In Section 2, the incorporation of MDC and K-means is described with some experimental result. The performance in CIEL*a*b* domain is analyzed in Section 3. Finally, Section 4 contains a summary.

3.1 Related Work

Presently, several paradigms for color reduction have been proposed. The first scheme processes colors by splitting the color space into smaller regions. Among the methods for accomplishing this are median-cut [Hec82], octree [GP90], and variance-based algorithm [WPW90]. However, their disadvantage is that the resulting image often contains regions with colors largely different from the original ones, when viewed by human eyes.

The second paradigm falls into the domain of clustering. K-means [Ver95] and [YK04], C-means [SK87], and fuzzy C-means [LL90] are among the practical clustering-based color reduction methods. A general drawback of clustering is that it tends to produce colors with low contrast.

Another more complex approach makes use of a neural network for better classification results, for example, the Kohenen network [Dek94], adaptive color reduction [PAS02], and Neural Gas [AP06]. Nevertheless, exploitation of the neural net is time consuming and difficult to implement.

The fourth paradigm tries to enhance the quality of the output image by considering spatial characteristics, for example, dithered color quantization [BFH*98]. The method can produce good quality images by performing color reduction along with dithering at the same time. However, this cannot produce a color for each area effectively.

The final scheme is region-based color segmentation, such as [VS91], which automatically finds color segments based on the minimum acceptable size of a region specified by a user or [CM02], which allows a user to specify not only a color difference threshold but also the spatial radius of a filter and the minimum acceptable size of a region for a given color. However, this approach often cannot reduce the number of colors effectively, and, instead, is applied in the field of region segmentation. Ultimately, although many techniques are available, an acceptable interactive method that can produce a higher-quality result is still needed.

3.2 Maximum Distance Clustering + 1 K-means

In order to solve the contrast problem of clustering-based color reduction, maximum distance clustering (MDC), considered a weak clustering method, is initially used to specify the highest contrasted colors. The set of maximum contrast colors is then used to initialize K-means clustering. K-means clustering is robust and widely used in various fields of research. The drawback is that it tends to lessen the contrast in the output image. Hence, we [RKO*07] propose a method that weights the contrasted colors obtained by MDC and the statistically calculated colors obtained by K-means.

1) Maximum Distance Clustering

To capture the maximum contrast of colors of an image, MDC is applied to the image in the RGB color space. The first iteration starts by sampling a pixel from the input image and then identifying a color with the largest Euclidean distance from the sampled pixel's color. For all consecutive iteration, the criterion for a new color is (3.1).

$$d_{\max,k+1} = \min_{k} \max_{i,j} \left\| cc_k - ce_{i,j} \right\| \tag{3.1}$$

where $d_{\max,k+1}$ is the maximum distance of cluster *k+1*, *k* is the number of existing clusters, *i,j* is the coordinate of a pixel, cc_k is the existing cluster's colors, and $ce_{i,j}$ represents colors at each coordinate, and is a candidate for a new cluster.

Since the colors would be used to initialize K-means clustering, the algorithm continues until the number of clusters equals the desired number of colors, or there is no other candidate color. In other words, the method requires running *k* iterations on an image to find *k* initial cluster means.

2) K-means Clustering

The derived highest contrasted colors are used as initial means for K-means clustering. Then, each new piece of data is used to compute the new mean of the closest cluster derived from (3.2).

$$\min_{k} \left\| cc_k - ce_{i,j} \right\| \tag{3.2}$$

In order to prevent K-means from dominating the clustering process, it must not be run until converged. Practically, running K-means for only one iteration gives the best result. Finally, each pixel of an image is rendered based on the closest derived cluster mean.

3) Sub-Optimal Maximum Distance Clustering

Since it is preferable that color reduction returns the output image as fast as possible, MDC is the bottleneck of our previous algorithm [RKO*07] as it consumes a considerable amount of time when the number of colors is high (see table 3.1). This is due to the fact that searching for one maximum distance cluster requires an iteration search on an image. So we [RKOSI07, Ruc07] propose a sub-optimal MDC.

The algorithm starts similarly by identifying a color with the largest Euclidean distance from a sampled pixel's color. The difference is that every new cluster color is derived within the second iteration search by a criterion explained as follows. First, calculate a minimum Euclidean distance of a pixel from the existing cluster(s) using (3.3).

$$d_{\min,i,j} = \min_{k} \left\| cc_k - ce_{i,j} \right\| \tag{3.3}$$

```
if $d_{\min,i,j} > d_{\max,k}$ for any cluster $k$,
    replace the existing cluster k with $p_{i,j}$.
else if $d_{\min,i,j} \neq 0 \,\&\, d_{\min,i,j} \leq d_{\max,k}$ for all clusters,
    generate a new cluster with value equal to $p_{i,j}$ if the present
    number of clusters is still lower than the desired number.
```

Using this new algorithm, running sub-optimal MDC consumes approximately the same time as running an iteration of K-means. In other words, running sub-optimal MDC + an iteration of K-means consumes time approximately equal to that of two iterations K-means.

3.3 Experimental Result

As shown in Fig. 3.2, it can be seen that the (sub-optimal) MDC + an iteration of K-means outperforms the region-based algorithm of [VS91], the commercial perceptual-based color reduction by Photoshop [AP7], and the normal K-means clustering. Note that we discover that running K-means using the same order of input will result in even lower contrasted image, so K-means, here, is performed forward for one iteration and backward for the next iteration, and vice versa, on image coordinate domain.

Thus, using sub-optimal MDC + 1 K-means is preferable to running K-means alone since the former give better result in shorter time, in RGB color space.

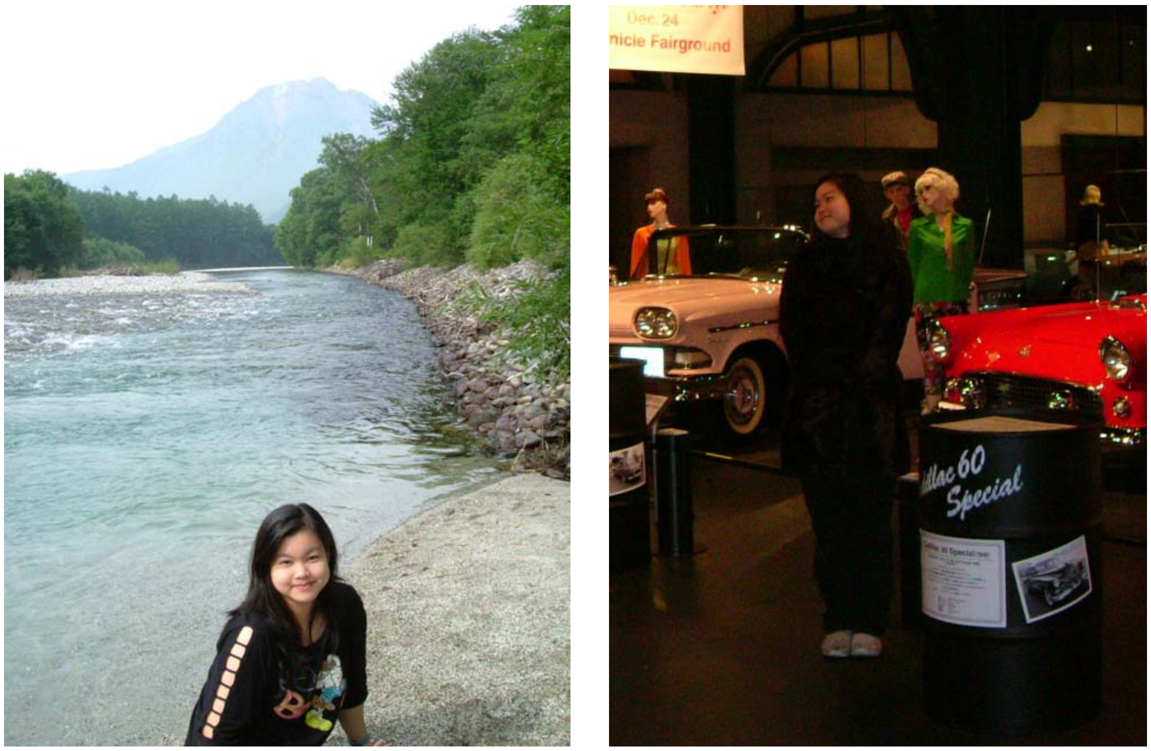
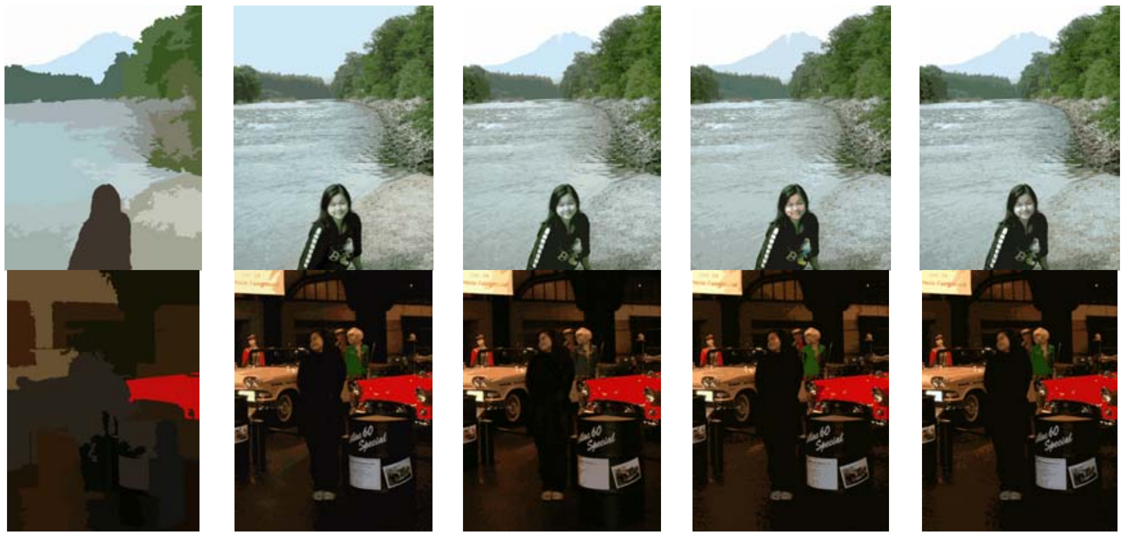


Fig. 3.1 Input images for color reduction.



(a) (b) (c) (d) (e)

Fig. 3.2 Comparison of color reduction tools when performing color reduction to 16 colors in RGB color space.
(a) Watershed. (b) Photoshop. (c) 15 iterations of K-means forward and backward. (d) MDC + an iteration of K-means. (e) Sub-optimal MDC + an iteration of K-means.

Method	Time complexity	16 colors	64 colors
K-means	$O(N \cdot k \cdot itr)$	2.33	7.64
MDC + an iteration of K-means	$O(N \cdot k^2) + O(N \cdot k)$	1.21	15.83
Sub-optimal MDC + an iteration of K-means	$O(N \cdot k) + O(N \cdot k)$	0.29	1.03

Table 3.1 Color reduction algorithm time comparison.
N is the image size (here 640x480), *k* is the number of colors, and *itr* is the number of iterations until convergence (here 15). Time is in seconds. Average is based on 20 images using a laptop with 1.8 GHz CPU.

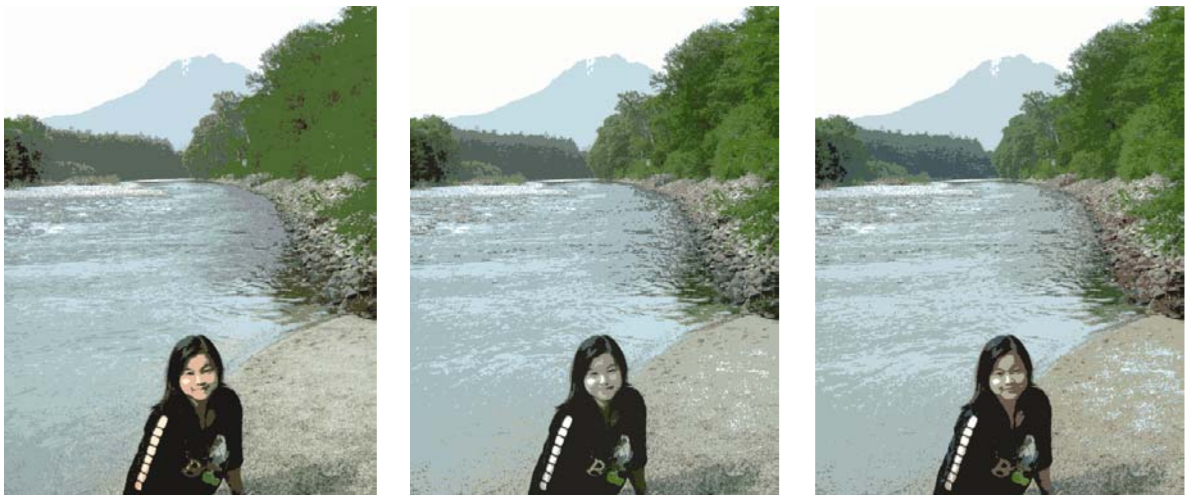
3.4 Performance in CIEL*a*b* Color Space

According to our previous finding, using MDC with K-means gives a better result than using K-means alone. In this research, we also discover that such a claim is trivial if the clustering is performed on CIEL*a*b* color space, instead of other color spaces like, for example, RGB.

CIEL*a*b* is designed to produce a color that is more perceptually linear than other color space, meaning that a change of the same amount in color value should result in a change of about the same visual importance. Nevertheless, it might be assumed that the comparison result in the former section should hold even if the color space is changed from RGB to CIEL*a*b*. Surprisingly, it is not.

This research shows that, using CIEL*a*b*, two powerful yet easy-to-implement color reduction methods can be achieved. The best choice is the sub-optimal MDC + an iteration of K-means. The second one is running K-means for two iterations, forward and backward.

Comparing Fig. 3.2 and 3.3, it can be seen that if CIEL*a*b* is used, both K-means and sub-optimal MDC + an iteration of K-means can generate images that can preserve contrast and objects in scene well.



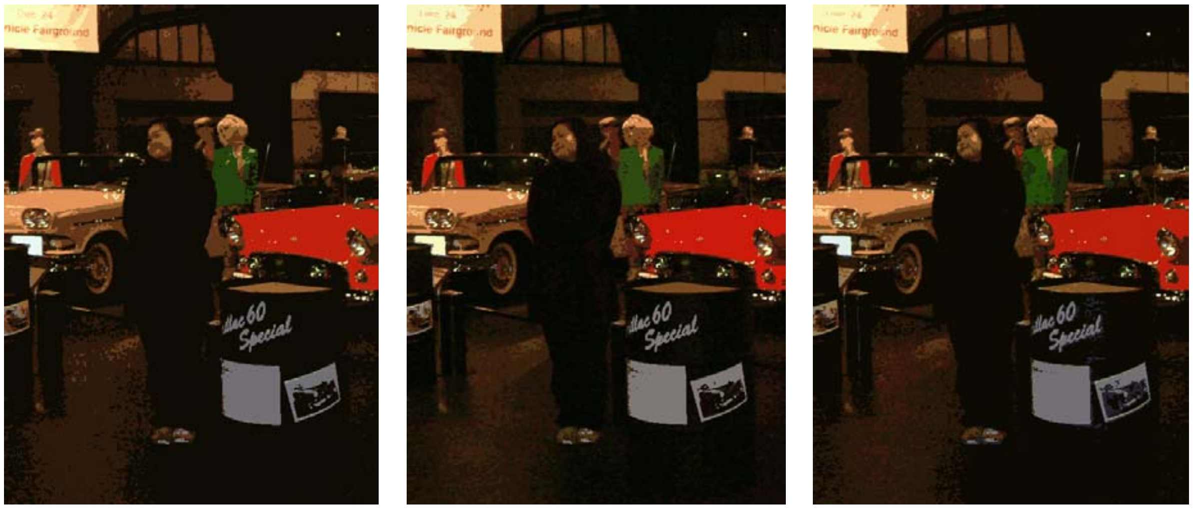


Fig. 3.3 Comparison of clustering methods when performing color reduction to 16 colors in CIEL*a*b* color space.
(a) 2 iterations of K-means forward and backward. (b) 15 iterations of K-means forward and backward. (c) Sub-optimal MDC + an iteration of K-means.

Method	MSE	
Photoshop	86.121	
K-means	RGB	CIEL*a*b*
1 iteration	168.581	115.22
2 iterations	131.606	90.499
2 iterations forward backward	110.640	67.156
15 iterations	95.913	70.699
15 iterations forward backward	86.895	60.762
MDC + an iteration of K-means	84.769	62.345
Sub-optimal MDC + an iteration of K-means	87.626	63.718

Table 3.2 Mean square error (MSE) from original images measured in CIEL*a*b* color space.
Average on 20 images reduced to 16 colors.

It can be seen from table 3.2 that, first, using RGB color space, error of (sub-optimal) MDC + an iteration of K-means is as low as that of K-means with a high number of forward and backward iterations, as well as that of Photoshop's perceptual color palette. However, it would be a misinterpretation to conclude that they are comparable. Photoshop cannot preserve contrast in scene well whereas K-means is slow and cannot preserve the colors with a small number of pixels, as can be seen in Fig. 3.2. On the other hands, our method can preserve the contrast in an image best in reasonable time.

Second, it can be seen that although the more iterations K-means uses, the less the error would be, the contrast sometimes gets worse when the number of iterations goes too high. Also, practically, the number of iterations is limited by the interactive requirement.

Third, running K-means forward and backward helps reduce a great deal of error.

Finally, it can be clearly seen that CIEL*a*b* offers higher quality in color reduction than RGB. In CIEL*a*b*, running only two iterations of K-means, one forward and one backward, is comparable to sub-optimal MDC + an iteration of K-means.

We also tested another color space, HSL. The results are not good, as expected, since the distance between points in such color domain does not provide meaningful measurement.

3.5 Summary

Using maximum distance clustering (MDC) to generate initial cluster positions for K-means can solve the general problem of clustering-based color reduction methods. It is required to run K-means for only one iteration to prevent it from dominating the process. Thus, the convergence-speed problem of K-means is not present in our algorithm.

As MDC is comparatively slow when the number of desired colors is high, a sub-optimal algorithm is proposed and shown to be extremely fast and able to generate a higher quality image than many existing interactive color reduction methods, in RGB color space.

However, in human perception color space, CIEL*a*b*, K-means alone is comparable to MDC + an iteration of K-means, provided that K-means is run for two iterations or higher, one iteration forward and another iteration backward. Considering time, running MDC + an iteration of K-means consumes approximately the same as that of two iterations of K-means.

Chapter 4

Brush Stroke Planning Using Global Orientation

After the foreground and its appropriate number of colors are derived, a method that mimics human painting style is used.

In this chapter, Section 1 shows related works. The next section shows how to calculate global orientation. Then hierarchical painting is shown in Section 3. Section 4 shows some of the experimental results based on 2D and 3D input. A summary is in Section 5.

4.1 Related Work

For the local orientation used to guide brush strokes, [Her98] uses a gradient to guide brush strokes that are more robust to texture, whereas [SY00] uses image moment that is more robust to shape orientation. However, such methods do not produce good brush strokes because the gradient or moment that a robot preserves could be noisy. Recently, methods that consider global gradient include [HE04], which selects only a strong gradient and applies a linear radial basis function (RBF) to interpolate a gradient on other areas. This seems to match an artist's perception. In fact, results shown in [HE04] resemble Van Gogh's style. This research modifies the method of [HE04] to generate practical end-effector trajectories for the robot, by making it faster using a certain size of RBF window.

4.2 Global Orientation Calculation

From an original image, the robot would then calculate the normal orientation. Orientation, (4.1), is calculated using the gradient operator.

$$\theta = \tan^{-1}(Grad_y / Grad_x) \tag{4.1}$$

where θ has a range between $\{-\pi/2, \pi/2\}$.

This normal vector is not directly used to guide the brush because it would be noisy as shown in Fig. 4.1 (a). Instead, the robot uses linear basis RBF [Bor] to interpolate the gradient field. Differing from [HE04], where only strong edges are used as an input for RBF, and which could fail to capture some subtle but important edge such as human hair, this research exploits all gradient values as an input for RBF, which is possible by weighting function of (4.2).

$$w = \sqrt{Grad_x^{\ 2} + Grad_y^{\ 2}} \tag{4.2}$$

Furthermore, the orientation image is not calculated totally globally since doing so would make an orientation of any pixel affected by the whole image's orientation, making straight lines bend. Also, globally computing consumes a lot of time; for each pixel, the number of operations required is shown in (4.3).

$$R * C \tag{4.3}$$

where R is the number of rows and C is the number of columns of an image.

For example, a 640x480 image would require overall 9.4372e+010 sets of operations.

In order to enable the robot to calculate this in a reasonable time, instead, the robot just focuses around an area and calculates global orientation in this area. By doing this, the processing time would be reduced as shown in (4.4).

$$r * c \tag{4.4}$$

where r is the number of rows and c is the number of columns of a mask.

For example, a 640x480 image with a 10x10 RBF mask, used in this work, would require only around 3.0720e+007 sets of operations. In the case that there is no information of input orientation in the mask, as can be seen from the input figure, the mask size for searching is doubled until orientation information is found. In other words, the area that has no orientation information would use the orientation of the surrounding area. The result is shown in Fig. 4.1 (b).

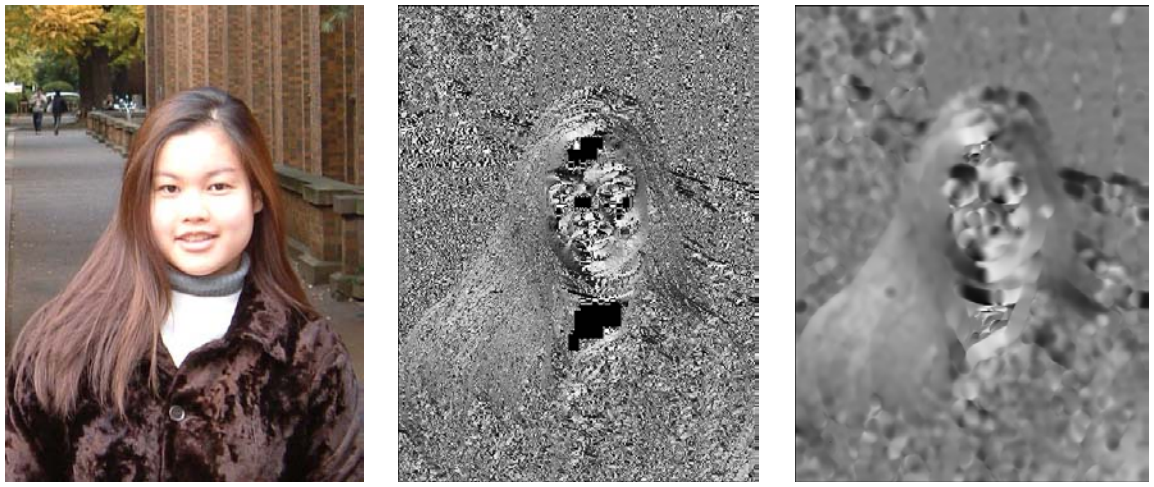


Fig. 4.1 Orientation of an image scaled from {-π/2, π/2} to {0, 255}.
(a) Original image. (b) Original orientation of each pixel. (c) Orientation after RBF is applied.

As can be clearly seen on the subject hair and on the building, the result is very much the same as when humans perceive orientation in a scene, even with the presence of a noisy gradient.

Whereas the above image uses texture to guide brush strokes, we observe that using the gradient of the object may or may not generate good trajectories. Hence, if the gradient information of the boundary between foreground and background is used, the result is shown in the below image.

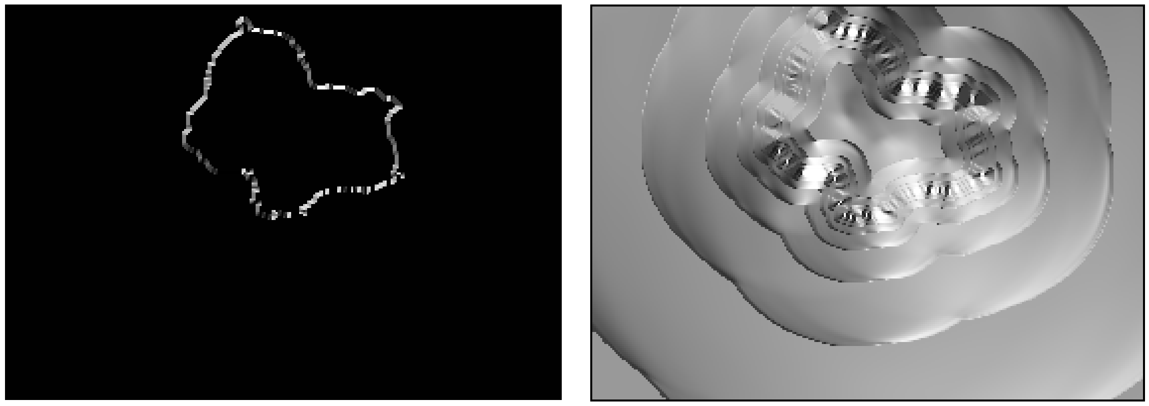


Fig. 4.2 Orientation of an image's boundary scaled from {-π/2, π/2} to {0, 255}.
(a) Original orientation of boundary. (b) Orientation after RBF is applied.

It is also possible to use range data instead of gradient information. In doing so, range data from stereo cameras is also input into the RBF. The result is shown in Fig. 4.3.

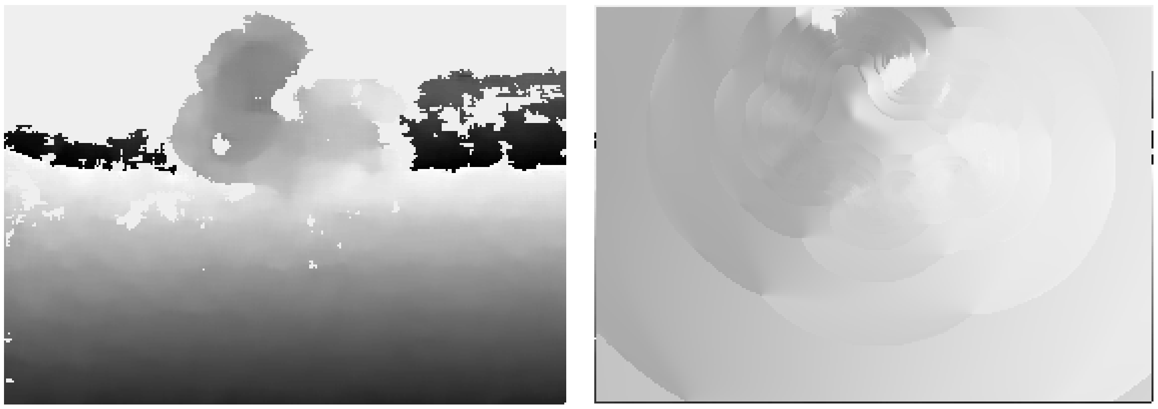

(a) (b)

Fig. 4.3 Range image.
(a) Original range image. (b) Range after RBF is applied.

4.3 Hierarchical Painting

The robot then scans each color segment derived from chapter 3 to find an area that contains a number of pixels larger than (4.5).

$$th_{region} = k_{region} \times r^2 \tag{4.5}$$

where r is the pre-computed radius of the brush, and k_{region} is the constant depends on drawing style.

If such an area is found, the robot starts to draw. To check whether the brush tip touches the canvas or not, the force sensor is then used along with the position of the brush tip detected in real time.

For the next movement, the robot then focuses along the normal direction of orientation derived earlier, and counts all yet-to-be-drawn pixels with the same color. If the number of pixels is higher than the threshold described in (4.5), the robot would decide to move the brush.

Each stroke is considered finished if an area found in the normal orientation direction is smaller than (4.5). After all the colors in each hierarchy are painted, the next hierarchy starts, where the robot then selects a smaller brush. The number of hierarchies depends on the error criterion between the picture in the robot's mind and the painting on the canvas.

4.4 Experimental Result

At the present time, due to insufficient drawing equipment and other resources, the artworks the robot created are subjected only to drawing of the geometry edge, as can be seen in the next chapter. In color filling, simulation on a laptop is done, k_{region} is set to be 0.5 for the first iteration and 0.3 for consecutive iterations, and brush size is reduced by the ratio of 0.5 every hierarchy. Also, prior object segmentation allows for the background to usually require less detail than the foreground, so it can be drawn with a smaller number of hierarchies. This usually makes the foreground stand out and also leads to considerable time reduction, an advantage that can be seen in Fig. 4.4, or we can draw only the foreground as shown in Figs. 4.5 and 4.6.

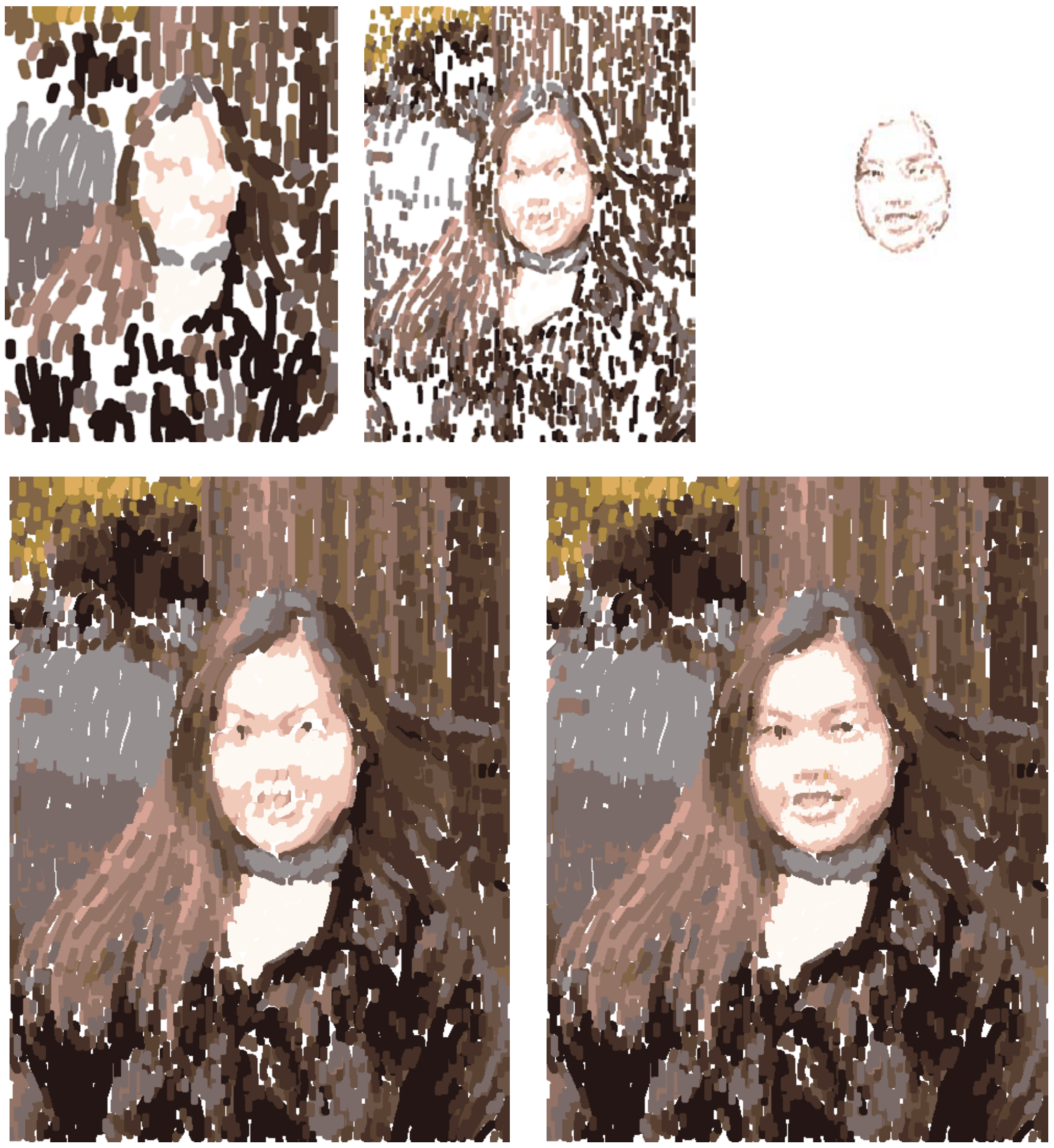


Fig. 4.4 Visual feedback painting with object segmentation (subject's face here). The first row shows each hierarchy, and the second row shows the overall result.

(a) (b)

Fig. 4.5 Visual feedback painting with object segmentation using global gradient information with 1 brush size.
(a) Trajectories, (b) Area filling result.

(a) (b)

Fig. 4.6 Visual feedback painting with object segmentation using global range information with 1 brush size.
(a) Trajectories, (b) Area filling result.

4.5 Summary

For the local orientation used to guide brush strokes, using a gradient to guide brush strokes is more robust to texture, whereas using image moment is more robust to area orientation. However, local orientation is usually noisy and thus results in inappropriate brush strokes. In this sense, global orientation, which can be described as a smoothed version of local orientation, is applied.

In this research, a linear radial basis function (RBF) is selected as the smoothing method, as previous works suggested. We approach the problem differently by focusing also on the speed of the algorithm. Speedup is accomplished by calculating RBF using a fixed size window. For an area that has no orientation information the RBF window is increased until it covers the surrounding area that contains orientation. This substantially reduces the time consumption so that RBF can be performed in real time.

It can be seen that the RBF can be applied not only in the gradient domain but also in any other similar domains to generate smooth output.

After the foreground, color, and orientation information are derived, the robot can perform brush stroke planning automatically based only on the brush size information.

At this point, the result of area filling is verified using a computer simulation. This type of visual feedback simulation consumes around 1 second per hierarchy, for a 640x480 image, on a laptop with 1.8GHz CPU; thus, it should not be any load at all for the real robot platform. The problem to be aware of in the real drawing by the robot is how to consider the color mixing on canvas, and how it would affect the drawing as a whole.

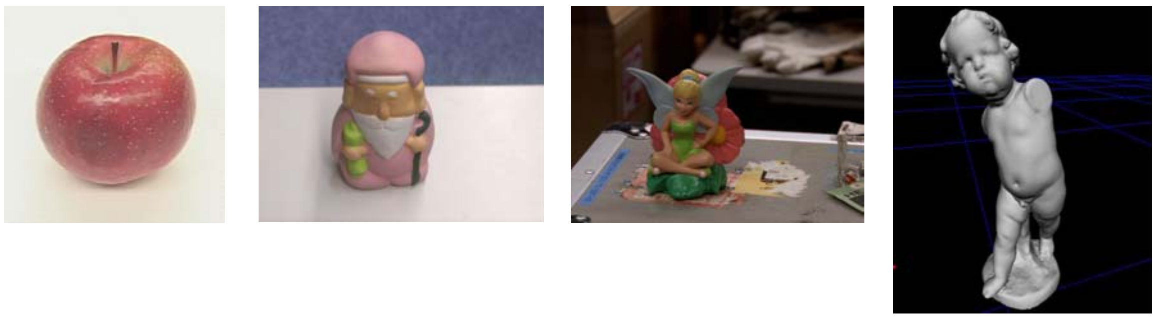
Chapter 5

Geometric Edges Processing

Some human artists use edges to enhance their paintings. Small children are also taught to do so in their early school period. The idea is to see an object and select an appropriate set of edges to represent the object in the painting. This is one fundamental way human sense an object. We use geometric edges of an object to represent these edges.

This chapter is divided into 4 sections. Related work and 3D sensor hardware is explained in Section 1. Generating a 3D model is described in Section 2. How to extract 2D feature lines from 3D shapes is briefly shown in Section 3. As these lines contain redundancy, they cannot be used by the robot painter directly, so Section 4 describes how to process them into brush trajectories. Section 5 is the summary.

Test objects used in this work are shown below.



(a) (b) (c) (d)

Fig. 5.1 Test objects for processing geometric edges.
(a) Apple. (b) Old man doll. (c) Tinker Bell doll. (d) Cupid doll.

5.1 Related Work

Technically, many researchers use gradient information to represent edges of objects. However, this information can be distorted by the color information on the surface. For example, even though a ball contains only 1 circular edge, color on its surface will generate unnecessary gradient information.

Hence, we decided to use 3D geometric edges of an object as input for the robot to draw as they represent the object well. As painting is done in 2D space, we describe how to extract 2D edges from the 3D model. Then, we explain how the 2D edges are processed into brush strokes.

In the first phase, the 3D shape is reconstructed from multi-view range images or multi-view camera images. In this research, we use a 3D model sensor called Vivid 900 as it can capture 3D geometric edges in high detail. The sensor is shown in Fig. 5.2.

Fig. 5.2 Vivid 900 sensor.

5.2 3D Model Generation

Examples of the range images captured by the Vivid 900 sensor are shown in Fig. 5.3. An adequate number of local images, which overlap each other, are captured from several viewpoints, and they are aligned and merged, resulting in the image shown in Fig. 5.4.

Note that outliers in the range images are deleted by a user manually to achieve the results shown in Fig. 5.3. Generally, the range image of each view consists of foreground and background, and the user must delete the background manually before exploiting the foreground images to generate the 3D model. This is a tedious task. Hence, we propose the use of foreground segmentation to delete these noises more effectively. The foreground segmentation described in Chapter 2 can be used directly to assist the current technology of multi-sensor modeling by segmenting the color image of each view and applying the result to the corresponding range image.

We use another object as a test subject in Fig. 5.5. Here the foreground contains color similar to that of the background. One might propose to use range information as an input for a segmentation program to overcome the not-well-separated color distribution problem. However, as we explained in Chapter 2, our foreground segmentation has components of both local and global classifiers. As a result, it is robust even when the foreground and background contain similar features.

Furthermore, even though our foreground segmentation scheme can deal with multidimensional features, it is not always appropriate to use range information as an input to the segmentation scheme since it could result in bad training data, as shown in Fig. 5.6, which would confuse the classifier. In these images, we just use color information.

An example of a 3D model generated from segmented color and range images is shown in Fig. 5.7.

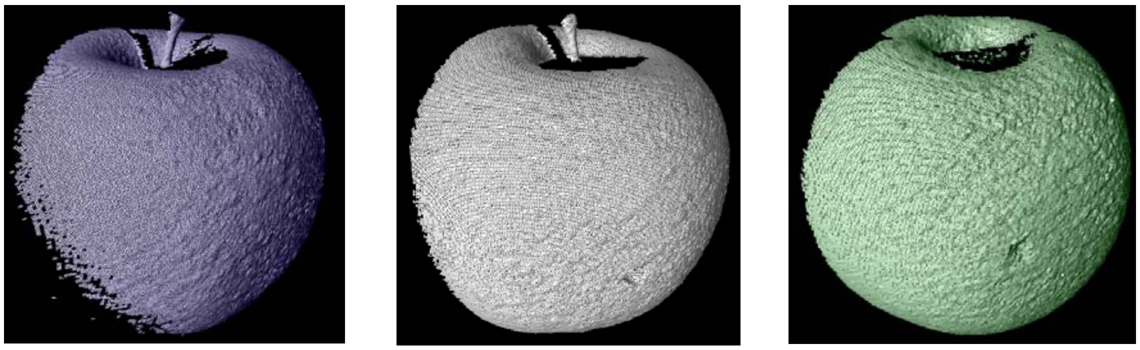


Fig. 5.3 Range images of an apple captured by the Vivid 900 sensor from different viewpoints.

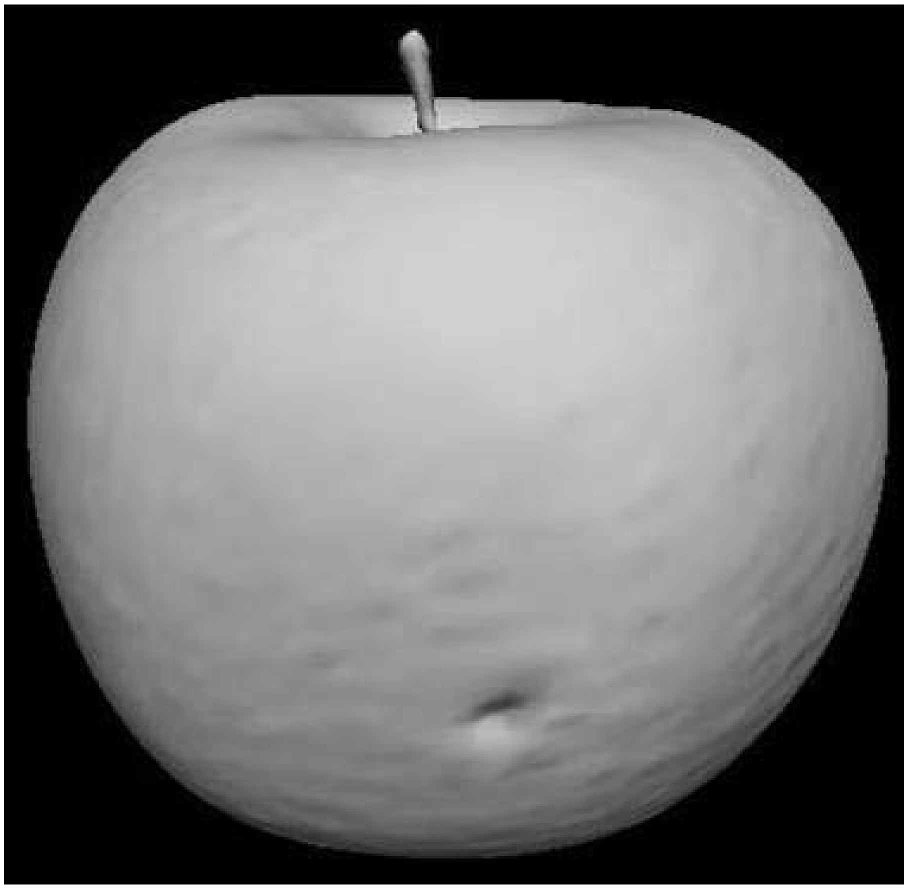
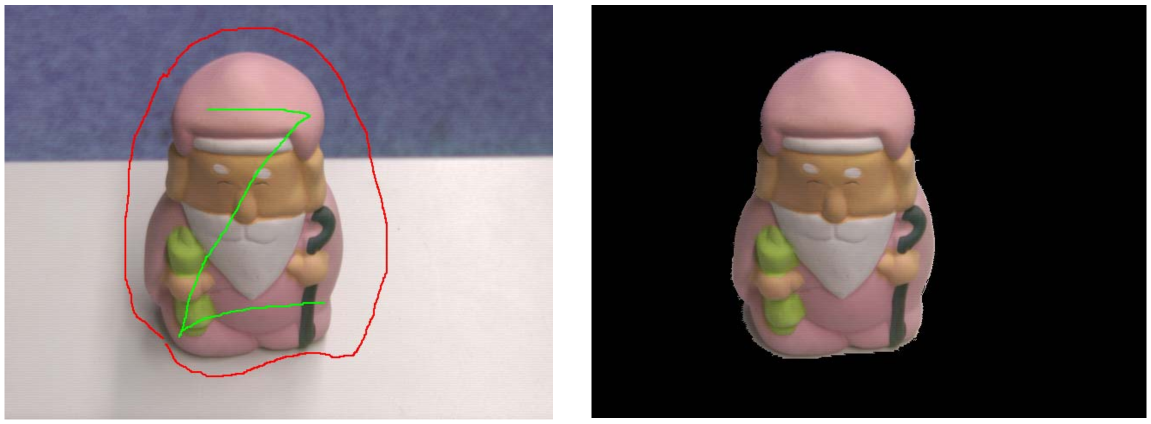


Fig. 5.4 3D model of Fig. 5.3.



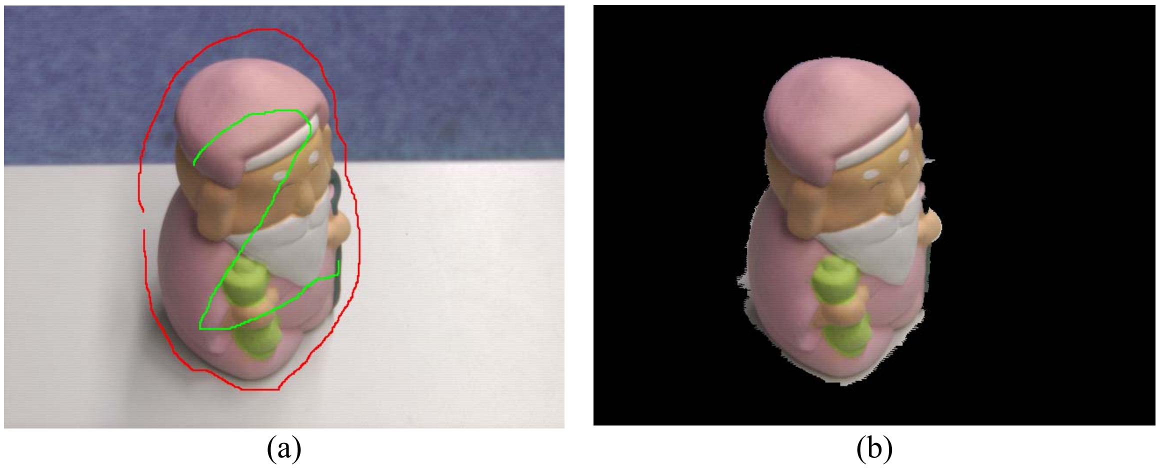

(a) (b)

Fig. 5.5 Foreground segmentation of each view.
(a) User input. (b) Segmentation result.

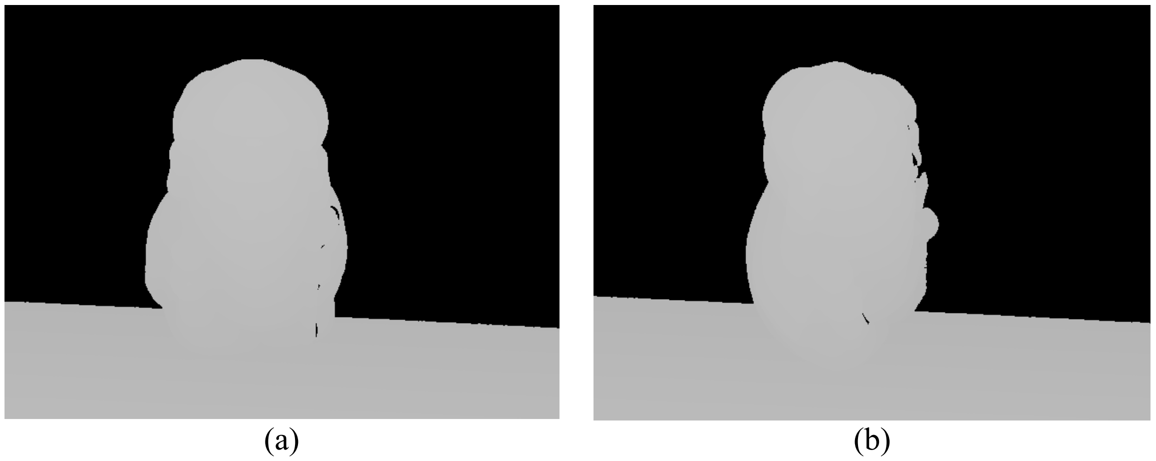

(a) (b)

Fig. 5.6 Range images of Fig. 5.5.

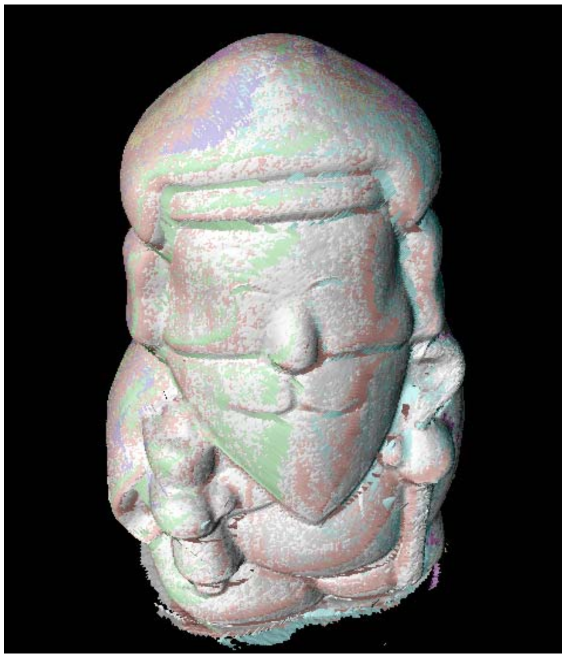


Fig. 5.7 3D model of Fig. 5.6.

5.3 Extracting Geometric Edges

After the 3D model is derived, we can proceed to the next step of geometric edge extraction.

A 3D model consists of triangular patches. A geometric edge is defined as the common edge at which the sign of inner product of the normal vector of a patch and the viewing direction changes, as shown in Fig. 5.8.

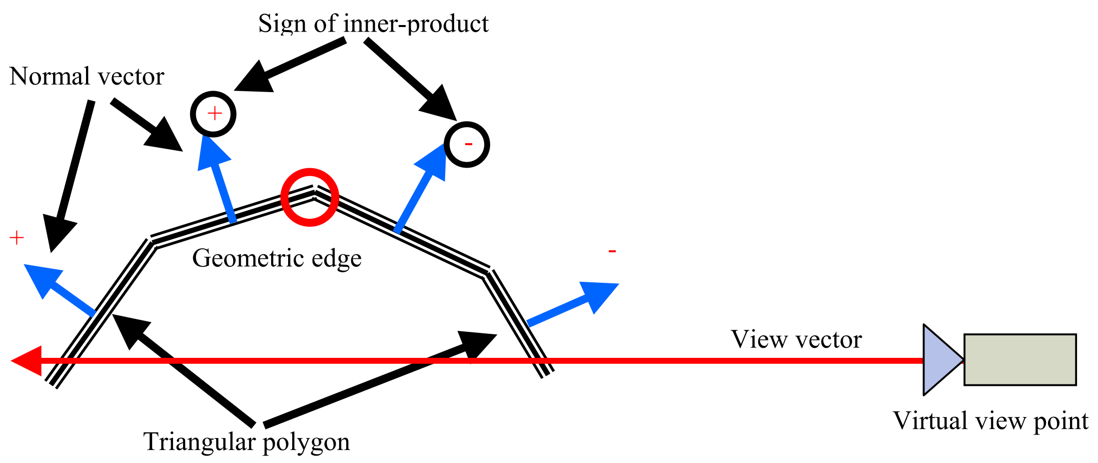


Fig. 5.8 Geometric edges extraction.

Actually, the geometric edges extracted in this way, Fig. 5.9, include hidden edges and many very short edges that are not appropriate for the painting task. These will be reduced in the next section.

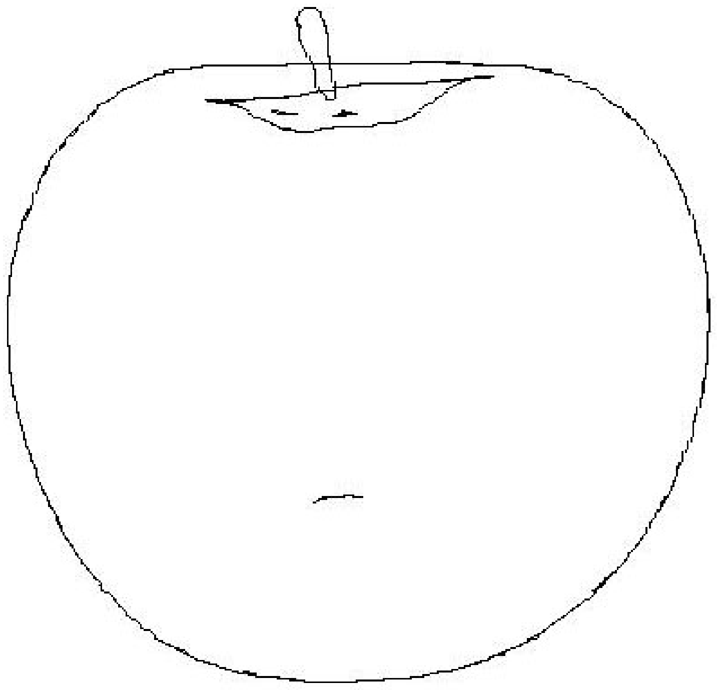


Fig. 5.9 Original geometric edges of the apple.

5.4 Conversion to Brush Strokes

After 2D geometric edge extraction, still, original data possess redundant points and links that are not suitable to be used as brush strokes. As an example, the input to this section, the Tinker Bell doll, is shown in Fig. 5.10. This image has redundancies as can be seen in Fig. 5.11 (a).

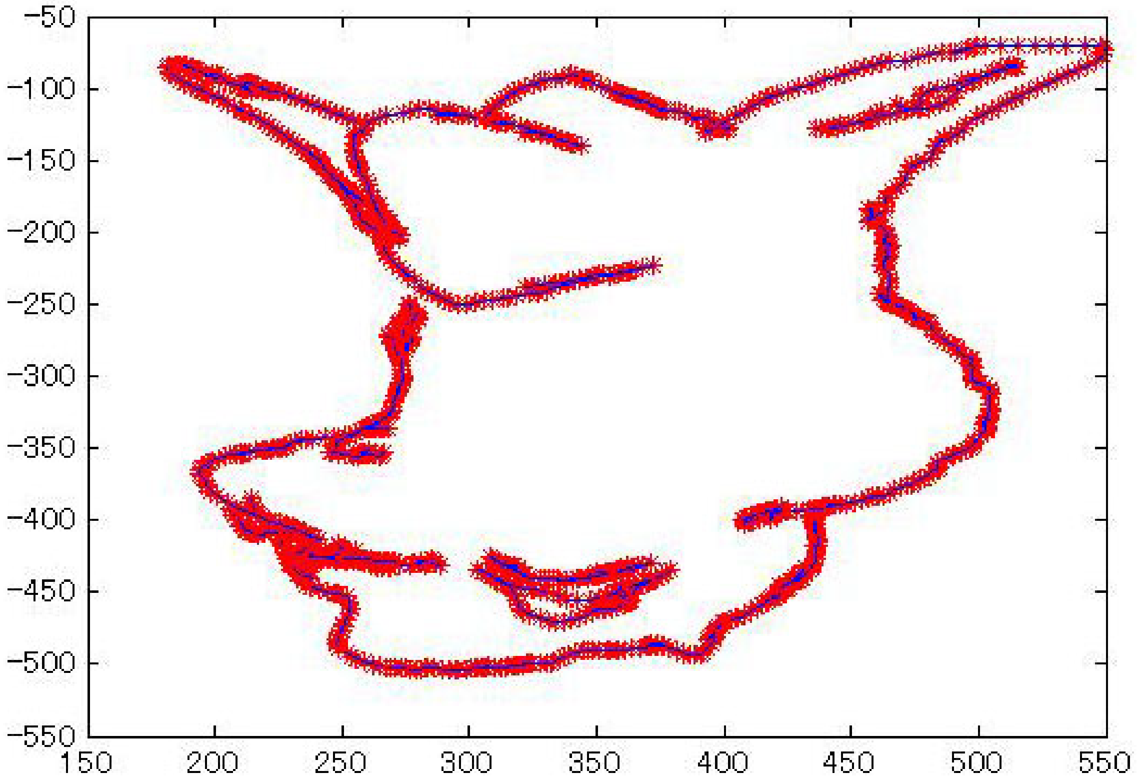


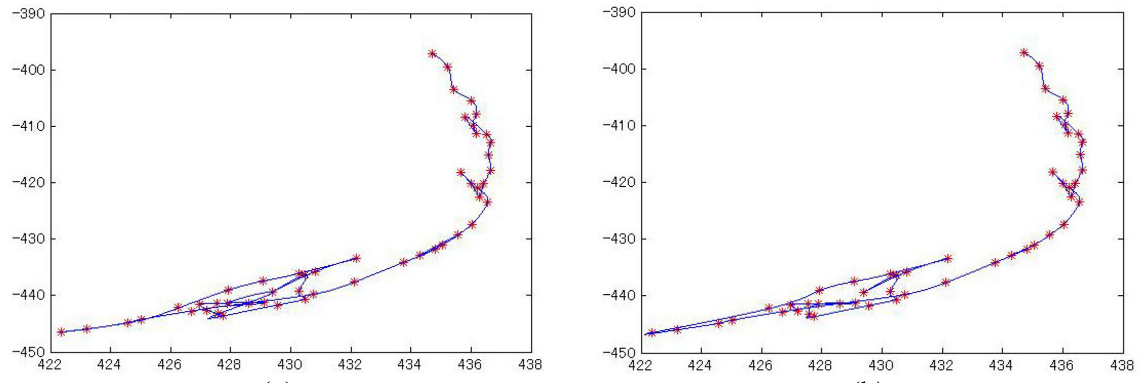
Fig. 5.10 Original geometric edges of Tinker Bell doll[a].

[a] Red color represents vertexes of edges, blue color represents spline curves that show connection between these vertexes.

The redundant points and links are reduced using methods described below.

1) Reduce Redundant Points

This is a case where there are 2 links between point A and B instead of 1 link. In other words, link A-B and link B-A. A simple search method is used to find this linkage false. If a redundant link is found, one link will be deleted.

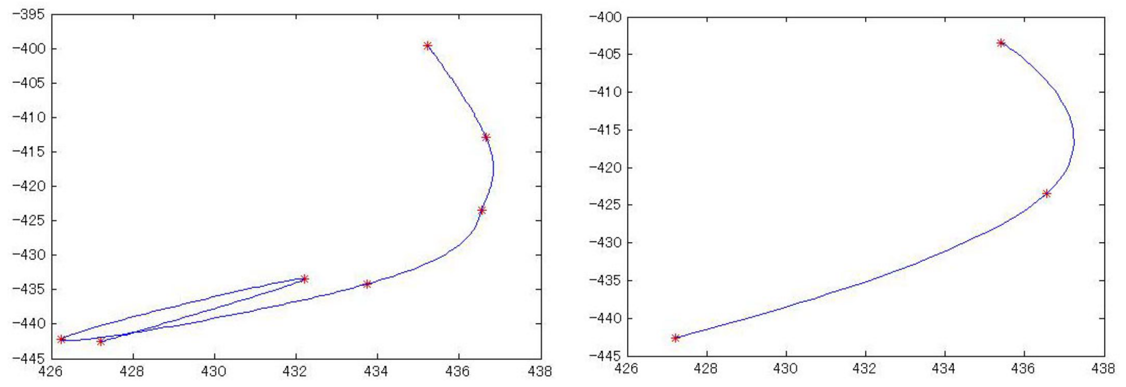


(a) (b)

Fig. 5.11 Redundancy points reduced[a].
(a) Original coordinates. (b) After processing.

2) Cluster Points

Too many points do not provide a robot with good brush trajectories to draw. Using the above redundancy-points-reduced coordinate, the brush strokes will still be noisy and meaningless. Also, they consume a lot of drawing time. So, simple clustering based on spatial difference is performed on the trajectory coordinate. Here are some examples of such clustering schemes with various clustering radii.



(a) (b)

Fig. 5.12 Clustered points with different radii[a].
(a) Radius equals 10. (b) Radius equals 20.

With the radius equal to 20, here is an example of the processed trajectories.

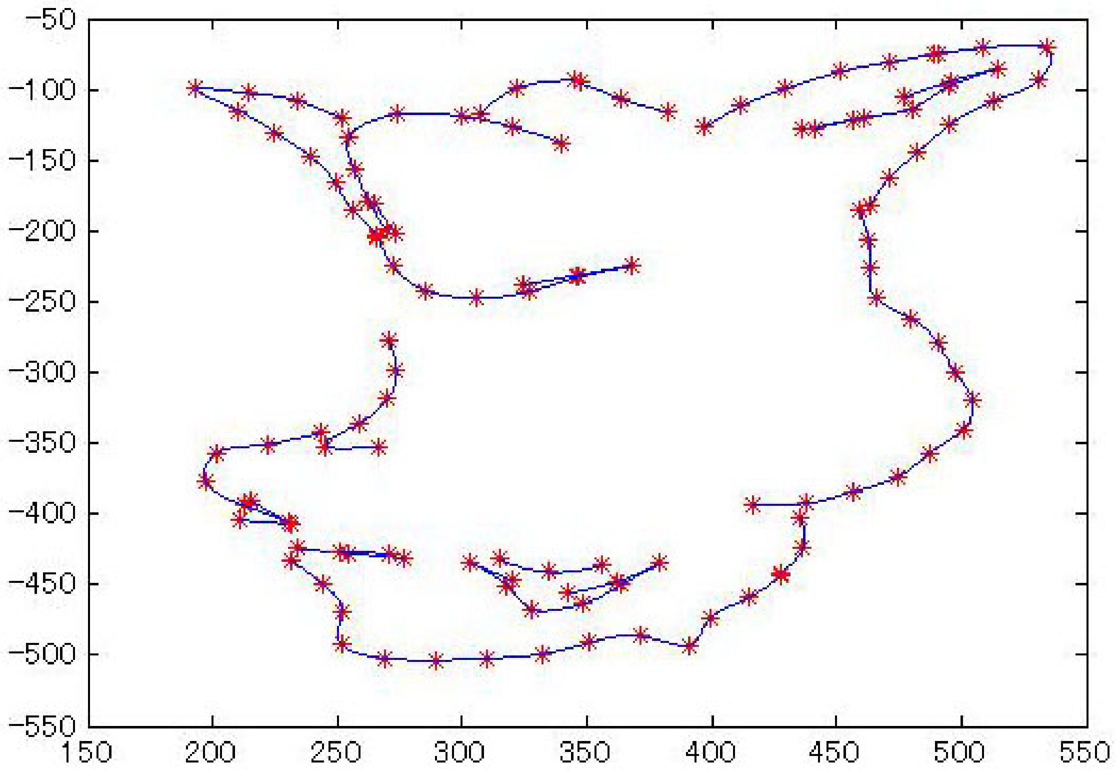


Fig. 5.13 Clustered points with radius equal to 20[a].

By looking at the above image, one can still see there are redundant links in the sense that they are too similar to be draw by two brush strokes. It is necessary to eliminate this coordinate to save manipulation time. The process is divided into steps 3 and 4.

3) Reduce Redundant Links 1/2: Comparing a Link to Previous Trajectories' Links

This step is based on a practical point of view that the robot should not draw similar trajectories onto finished brush strokes. To accomplish this, a link is compared to the previously processed links. If two links are too close, the new one will be deleted.

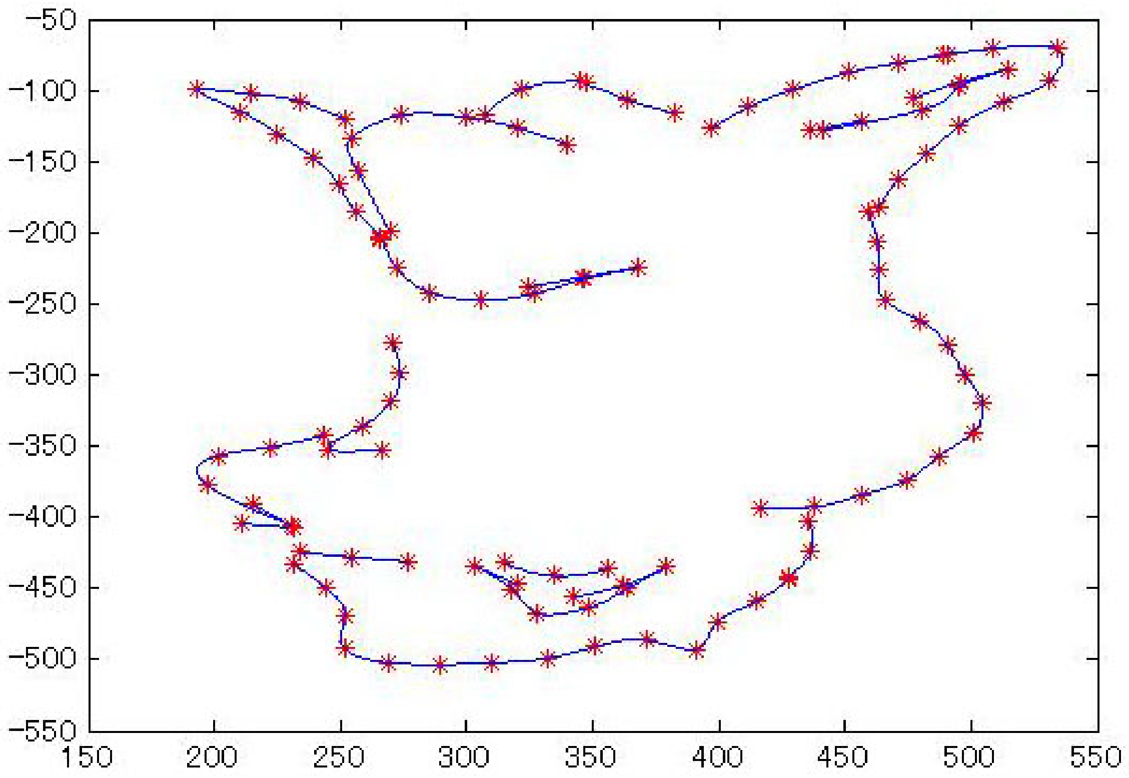


Fig. 5.14 Redundant links reduced: Comparing a link to previous trajectories' links[a].

Based on the above image, redundancy links are still present within the same brush stroke. This is resolved in step 4.

4) Reduce Redundant Links 2/2: Comparing a Link to the Present Trajectory's Links

Hence, a further step to eliminate similar links requires comparing two links in the same brush stroke. This is performed in forward and backward manner to be able to best eliminate such links.

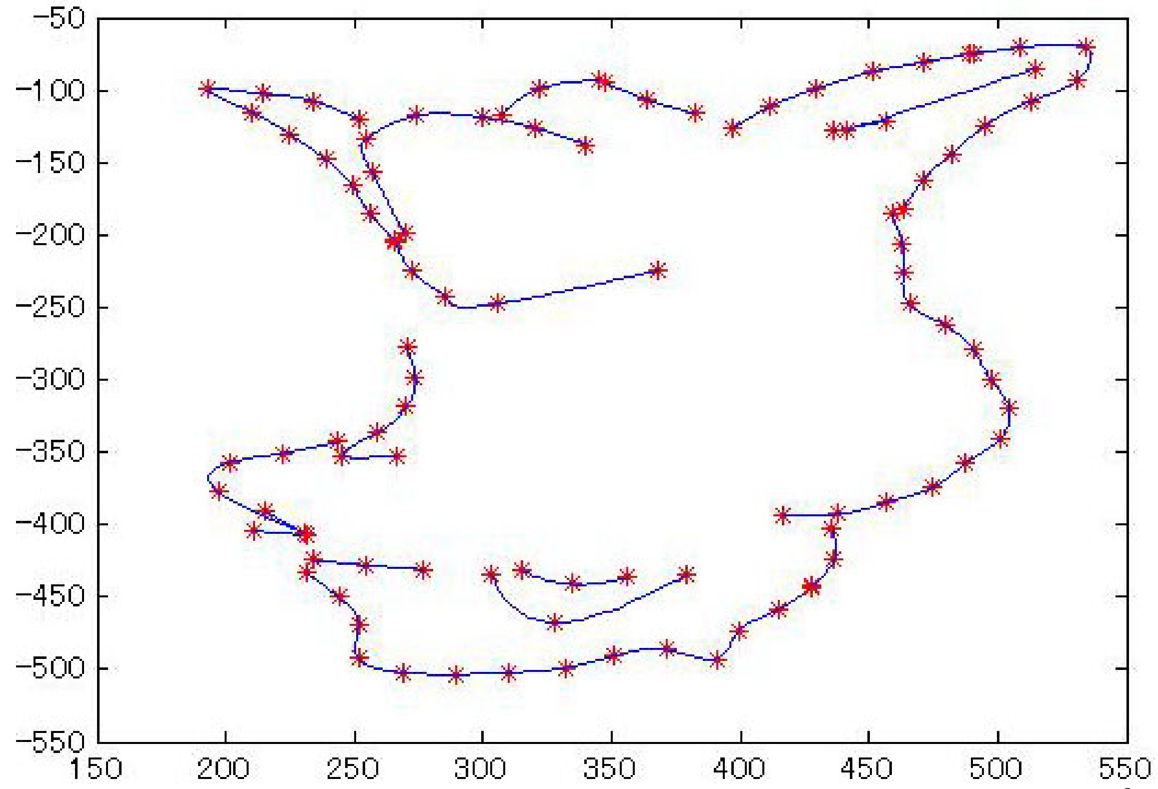


Fig. 5.15 Redundant links reduced: Comparing a link to the present trajectory's links[a].

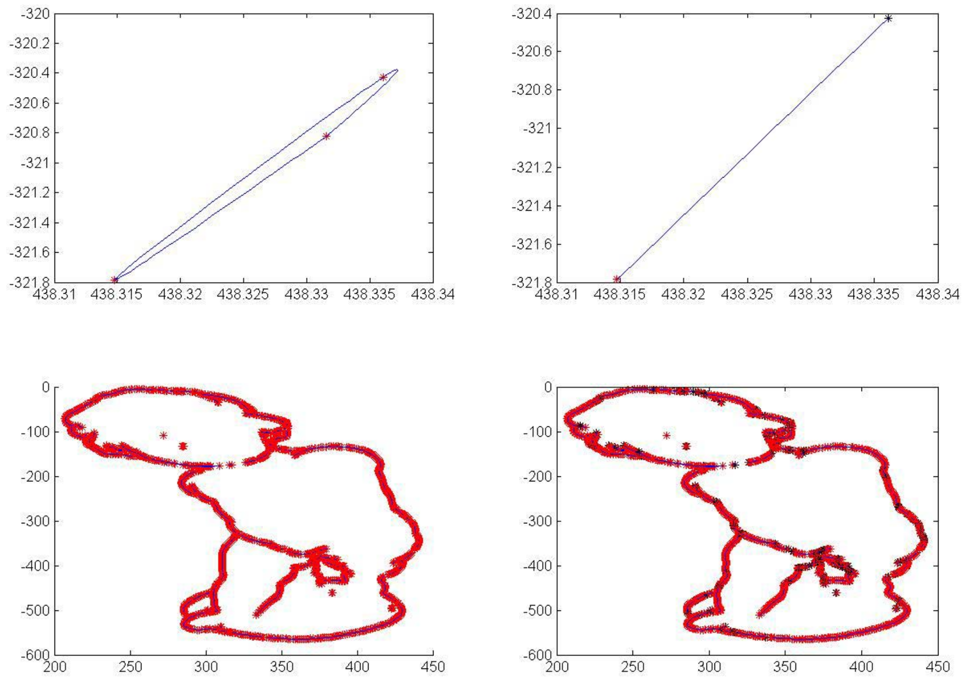


Fig. 5.16 Planned brush strokes of the cupid doll[a].

Note:
Upper-left image is considered the original trajectory, while the upper-right image is a processed trajectory. The lower-left image is the result up to the present stage, while the lower-right image is the result after adding the processed trajectory.

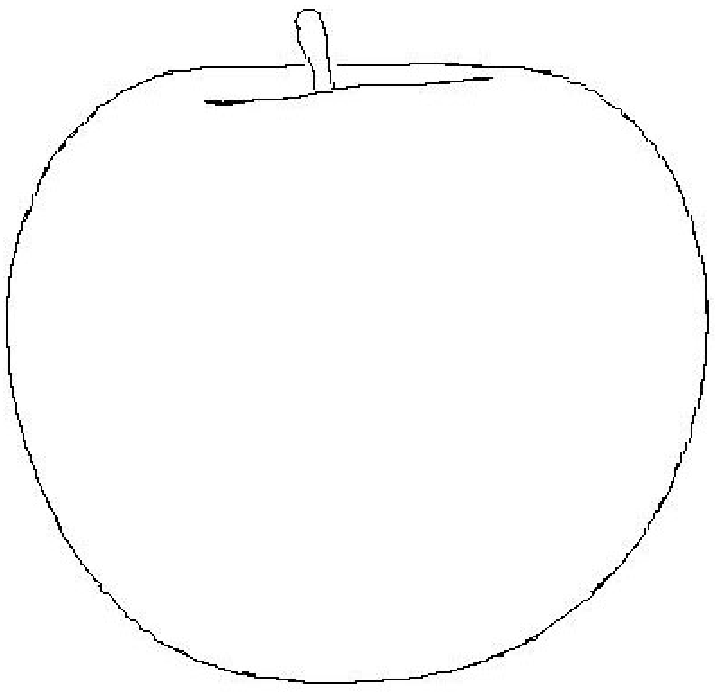


Fig. 5.17 Planned brush strokes of the apple.

5.5 Summary

Some human artists use edges to enhance their paintings. Although many researchers use gradient information to represent edges of objects, this information can be distorted by the color information on the surface. Hence, we decided to use 3D geometric edges of an object as input for the robot to draw, since they represent the object well.

After using a range sensor to capture multiple-viewpoint images, we align and merge the images to generate a 3D model. Because painting is done in 2D space, 2D edges are extracted from the 3D model. A geometric edge is defined as the common edge at which the sign of the inner product of the normal vector of a patch and the viewing direction changes.

Finally, the noisy 2D edges are processed into brush strokes by reducing various kinds of redundancy.

Chapter 6

Brush Manipulation and Experimental Result

In the last phase, painting by a robot with multi-fingered hands is achieved. One of the important challenges here is manipulation of a paintbrush by a multi-fingered hand. This process is definitely different from drawing by an XY-plotter in which a paintbrush is fixed on the arm by bolts. In grasping by a multi-fingered hand, we need to overcome problems such as the fact that the state of the grasp changes with motion and the grasp becomes unstable against a particular direction of force. The reason why we use a multi-fingered hand in spite of the difficulty is that we believe fingers are necessary to reproduce precise human techniques for manipulating a paintbrush. In this phase, visual feedback using stereo cameras is also performed.

In the remainder of this chapter, first, we describe some of the related works in Section 1. Next, grasping technique is presented in Section 2. Section 3 shows how to detect the tip of the brush. Detecting brush-canvas contact is explained in Section 4. Section 5 explains how to plan brush direction to prevent brush slide. Section 6 shows some of our techniques in actual painting skills. An experimental result is shown in Section 7. Section 8 gives a summary.

6.1 Related Work

In most studies about manipulating objects by a robot, the objects are fixed on the robot arm and they only the relationship among the objects is studied. Some researchers have tried manipulation by a multi-fingered hand [Nap56], [Cut89], [KI97]. They classified grasp and developed an algorithm to manipulate objects based on the classification. However, they expected that their method would be used for a manufacturing application, and therefore few studies have been executed about manipulating daily objects, which are often deformable like a brush in this study.

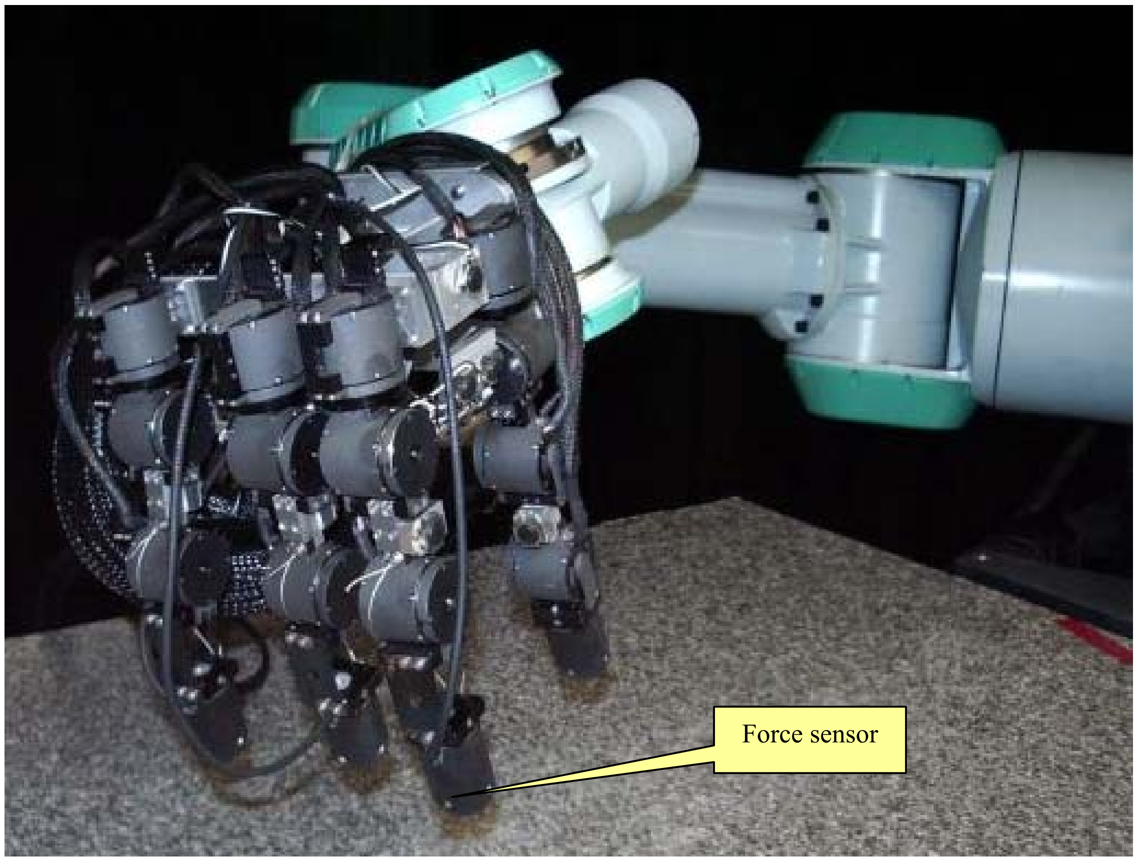


Fig. 6.1 Multi-fingered hand.

6.2 Grasping a Paintbrush

In this system, an ordinary commercial paintbrush is used. However, because the fingers of the robot are much thicker than the fingers of people, a thick handle is attached to the brush (Fig. 1.1). In grasping, the paintbrush is supported by four points, which are three fingers (thumb, forefinger, and second finger) and the root of the thumb (Fig. 6.2). This is almost an imitation of human grasping of a paintbrush. This grasping produces stable support of a paintbrush against forces from all direction except one direction. The one direction occurs when using the right hand to draw outward from right to left. In order to avoid this instability, the brush trajectory is planned so as to avoid the weak direction of brush motion when the picture is painted.

The process of grasping is explained as follows. First, the position of the handle of a paintbrush is detected using the vision system. The detail of the algorithm is described in the following section. Next, the right hand approaches the paintbrush. When the force sensor in the middle finger detects the contact between the finger and the paintbrush, the system decides that the hand is adequately close to the paintbrush for grasping, and stops the hand motion. Finally, power is gradually added to the fingers. The force on the fingers is watched during this phase, and power is added until a firm grasp is realized. Then the paintbrush is pulled up. These steps can be seen in Fig. 6.3.

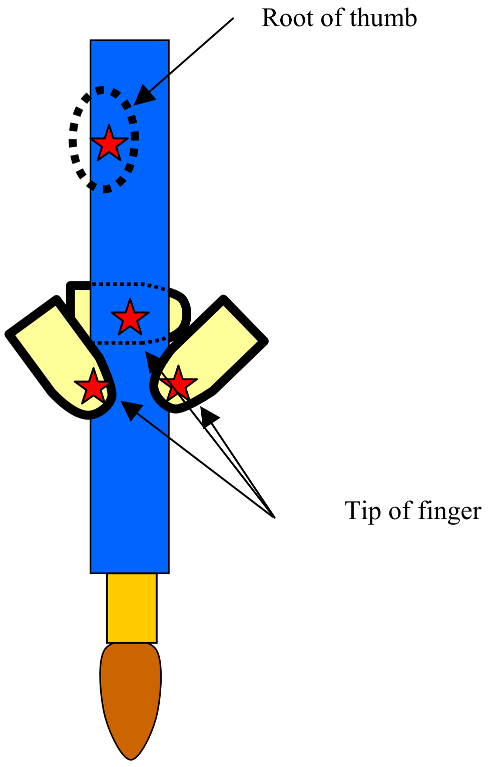


Fig. 6.2 Grasping using force sensor.

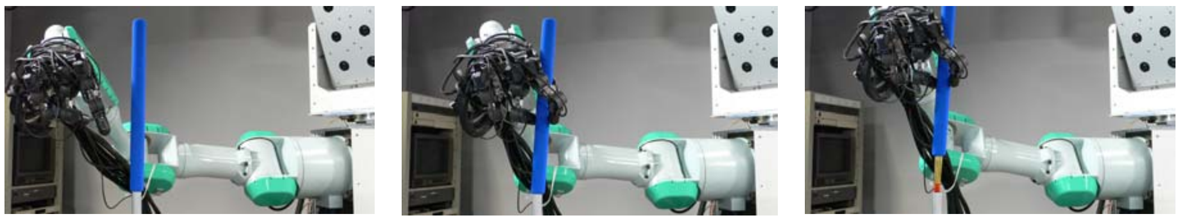


Fig. 6.3 Grasp states.
(a) Approaching. (b) Grip. (c) Pull up.

6.3 Detecting the Tip of a Brush

The detection of the tip of a brush is performed using the stereo vision system shown in Fig. 6.4. First, the position of the handle of a paintbrush is detected, and then the position of the brush's tip is derived from it. In this system, because the paintbrush is grasped by the multi-fingered hand, the relative position of the tip to the arm changes a bit with every trial of grasping. Moreover, the position can change by an unexpected slip of the paintbrush during painting. Therefore, it is required that the tip position is frequently detected in order that the system always knows the correct position. The reason why the tip position is not detected directly is that direct detection is difficult because the brush is deformable and its color changes according to the color of the paint.

First, pixels belonging to the handle are extracted from the captured image (Fig. 6.5). The color of the handle is given a parameter here, and the hue and saturation are used for the extraction. The 3D position of each pixel is calculated by stereo calculation. For improving accuracy, the area to detect is limited to the possible area determined from the hand position and rotation. Next, principal component analysis (PCA) is performed on the extracted pixels (in 3D space), and used to calculate the axis of the handle (Fig. 6.6). The pixel in the lowest position is regarded as the lowest point of the handle. Finally, the position of the tip is derived from these two factors: the axis and the lower point of the handle (Fig. 6.7).

Table 6.1 shows the result of measuring the accuracy of tip detection. The tip position was calculated ten times, keeping the paintbrush in the same position. Because the paintbrush does not move, the result must always be the same, but some error appears due to the vision system. The table shows the standard deviation of the detected positions. Three trials, in each of which the tip position is detected ten times, are performed in various positions and rotations. In every trial, the standard deviation of the error is within about 1.0 mm. However, although the brush deforms during painting, this algorithm does not consider it. In the actual case, an error of about 10 mm appears because of the deformation.

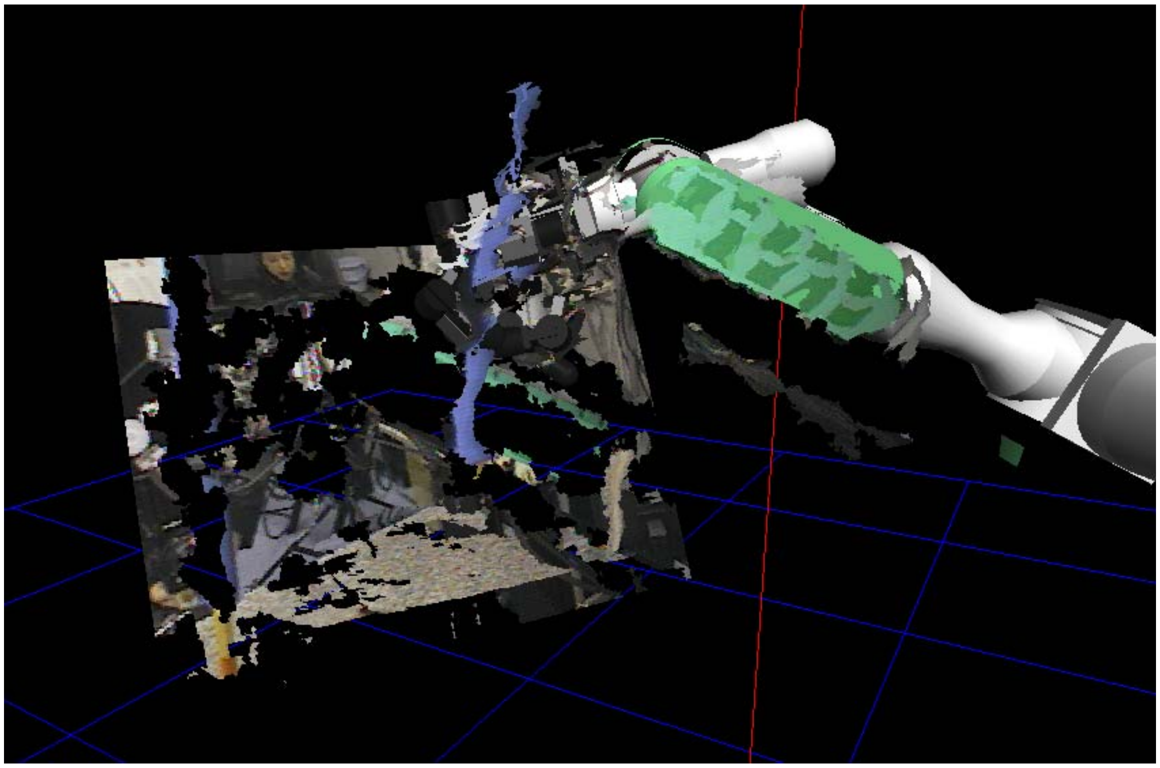


Fig. 6.4 Stereo vision system’s output.

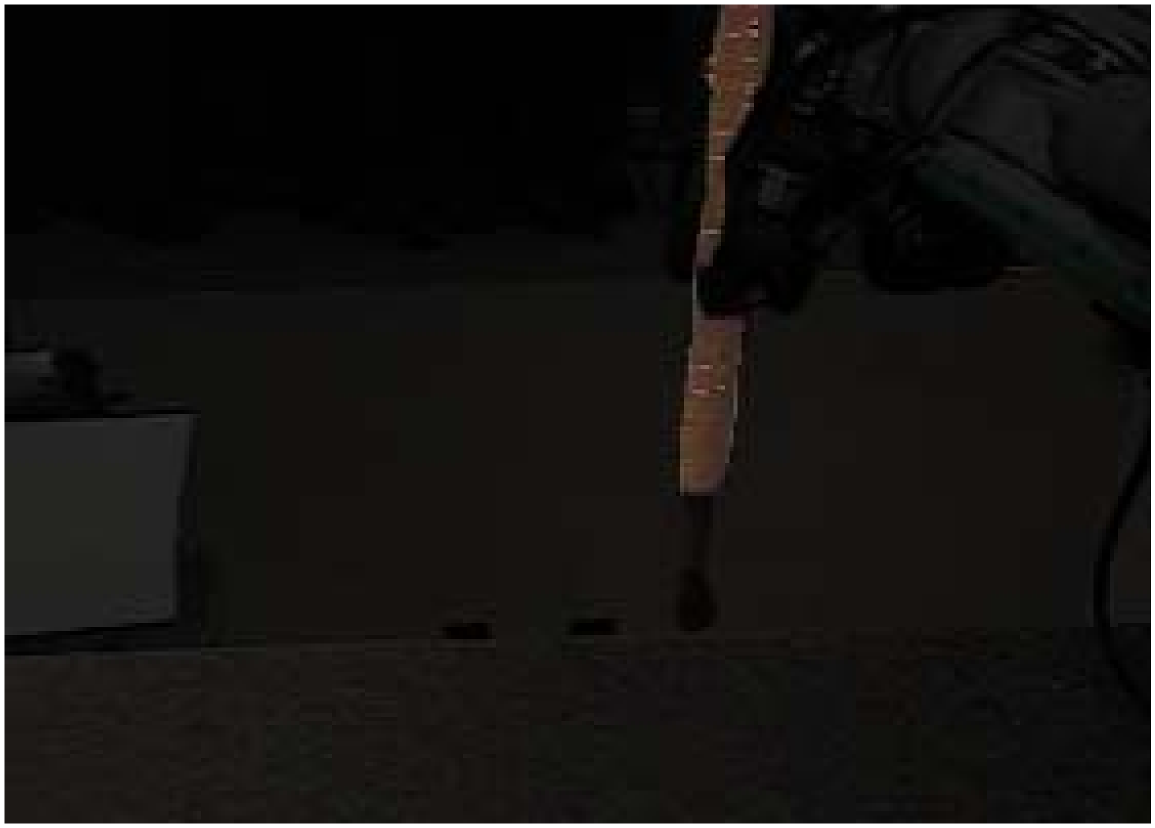


Fig. 6.5 Detection of the tip of a brush: Color extraction.

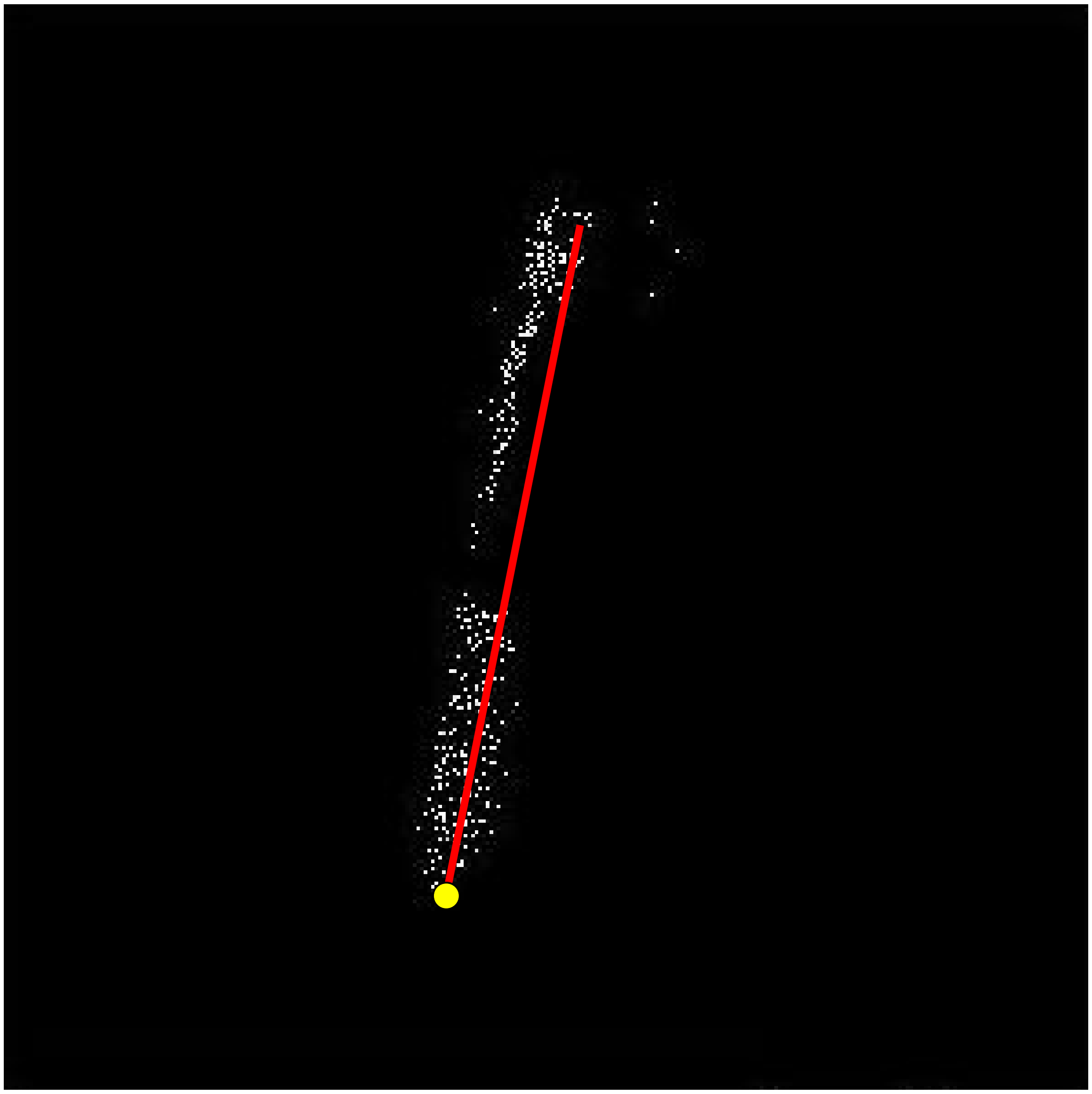


Fig. 6.6 Detection of the tip of a brush: Calculating 3D position.

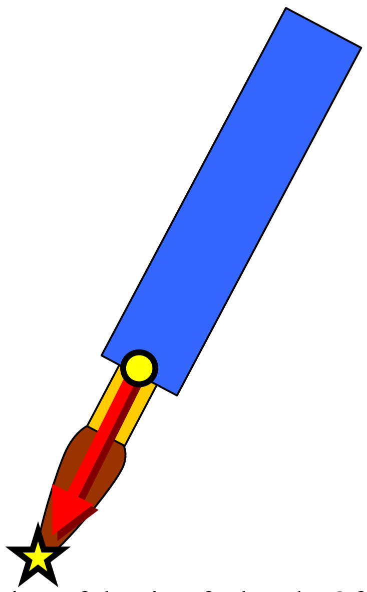


Fig. 6.7 Detection of the tip of a brush: Offsetting.

Trial	x-axis	y-axis	z-axis
1	0.64	0.89	0.66
2	0.77	1.50	0.80
3	0.90	1.60	0.49

Table 6.1 Evaluation of tip detection [mm]. This table shows the standard deviation of the detected positions.

6.4 Detecting Brush-Canvas Contact

Force sensors in the fingers are used for the detection of the contact between a brush and a canvas. The integration of force acting on the three fingers that support a paintbrush (F_{finger}) is focused here. According to Newton's equation, if a paintbrush is regarded as moving in a quasi-static process, F_{finger} is formulated as (6.1).

$$F_{finger} = -mg + F_{ground} \tag{6.1}$$

where m is the mass of the paintbrush, g is the gravity acceleration, and F_{ground} is the ground reaction acting on the paintbrush. From this equation, it can be said that F begins to decrease when the brush contacts a canvas.

Fig. 6.8 shows the change of F while pushing down a paintbrush, drawing a line, and pulling up the paintbrush. It is seen that F suddenly decreases when contact between brush and canvas occurs. Therefore, we set a threshold to detect this contact.

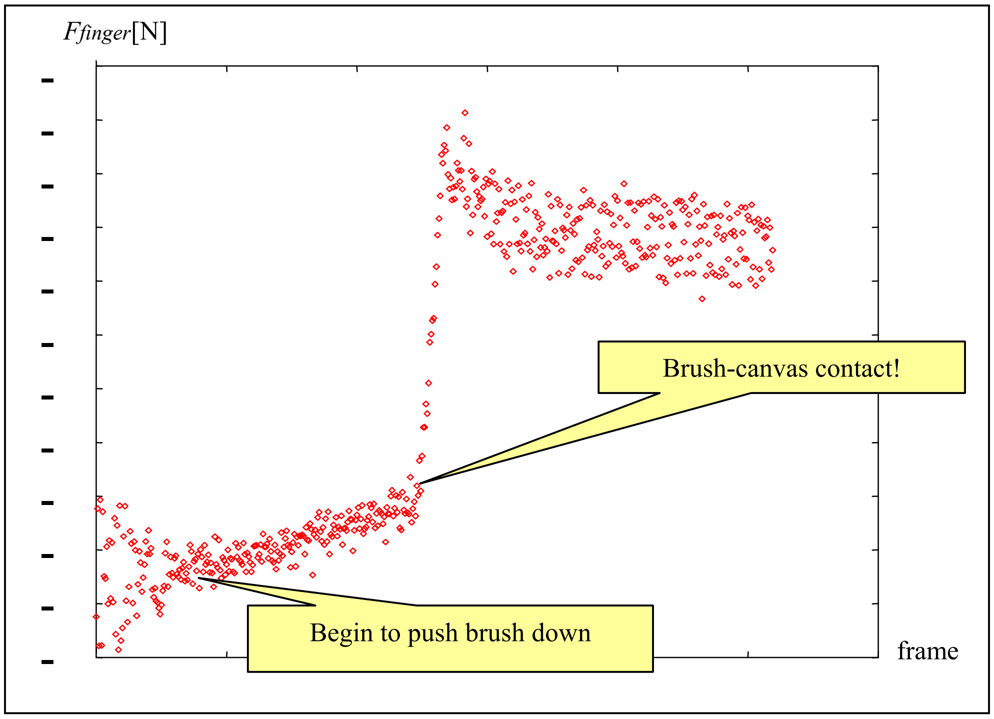


Fig. 6.8 Force acting on finger. The integration of force acting on the three fingers that support a paintbrush.

6.5 Reducing Brush Slide

Because the robot hands are designed to mimic human hands, there are movements that allow a brush to easily slip out of the hand, for instance, when using the right hand to draw outward from right to left. A human artist naturally avoids drawing at these angles.

For the robot, the best way to tackle the brush slide problem is to plan the trajectories carefully so that the brush avoids drawing at an angle that would tend to slide. To do this, the direction from point to point is reversed. For example, if drawing from A to B is in the range of angle that might cause the brush to slip, this trajectory is inverted to draw from B to A, as shown in Fig. 6.9.

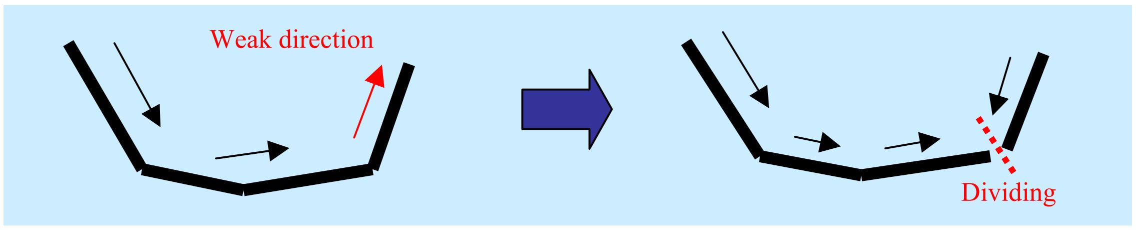


Fig. 6.9 Inverting the direction of a brush stroke.

Let the weak direction be from θ_l to θ_u, counterclockwise, here $\pi/4$ to $3\pi/4$, the result would be like Fig. 6.10.

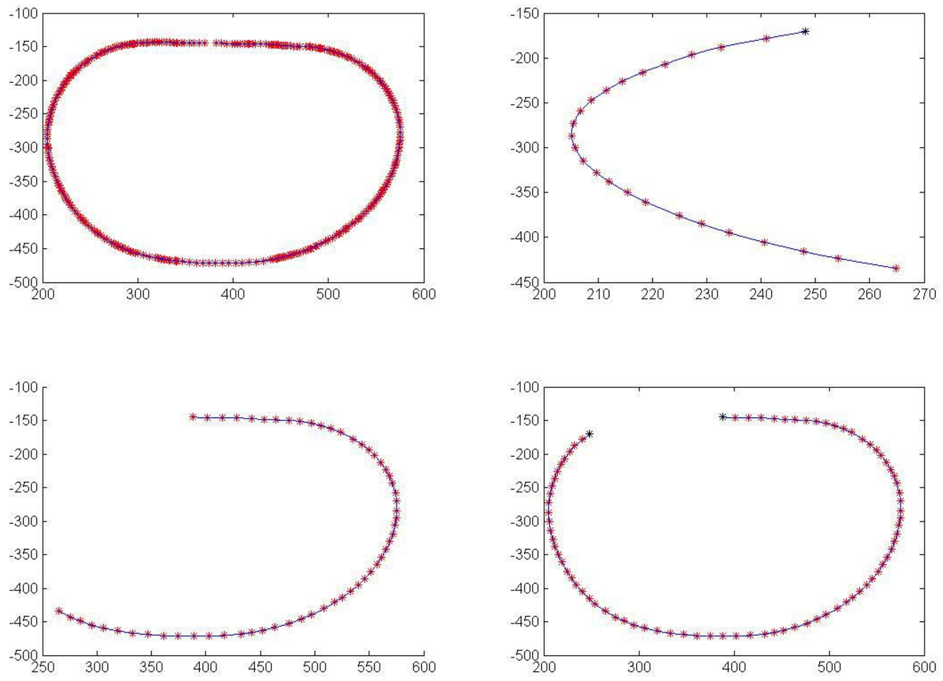


Fig. 6.10 Planned brush strokes of the apple.

Note:
Upper-left image is considered the original trajectory, while the upper-right image is a processed trajectory. The lower-left image is the result up to the present stage, while the lower-right image is the result after adding the processed trajectory.

6.6 Parameterized Paintbrush Technique

Now, the following three paintbrush techniques are parameterized (see Fig. 6.11):
. Leaning a paintbrush to the drawing direction (a lean angle)
. Pushing a paintbrush onto a canvas (an amount of pressure)
. Pulling a paintbrush up gradually (an amount of sweep)

The first parameter is a lean angle of a paintbrush. The drawn lines and curves, especially curves, vary according to the lean angle.

The second parameter is an amount of pressure. When a paintbrush is pushed down upon a canvas, it does not stop moving downward as soon as it contacts the canvas, but it keeps moving for a while. The duration to keep moving is varied as the parameter. It controls painting pressure.

The third parameter is an amount of sweeping performed by a paintbrush at the end of drawing. In drawing a curve with a paintbrush, the tip motion of the brush tends to lag behind the motion of the arm because the brush deforms during drawing. Therefore, if the arm stops when it reaches its target position, the tip often still remains behind its target position. In order to avoid this situation, a paintbrush is pulled up while gradually moving to the target direction. The parameter determines an amount of this motion.

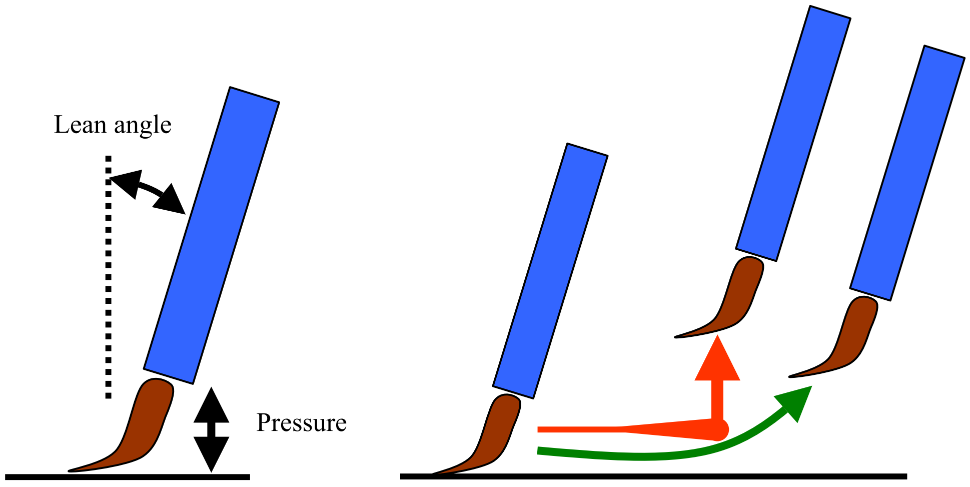


Fig. 6.11 Parameterized paintbrush technique.

6.7 Experimental Result

First, let's consider the effect of the brush parameter on painting. Fig. 6.12 shows the difference between pulling brush up gradually and abruptly, respectively.

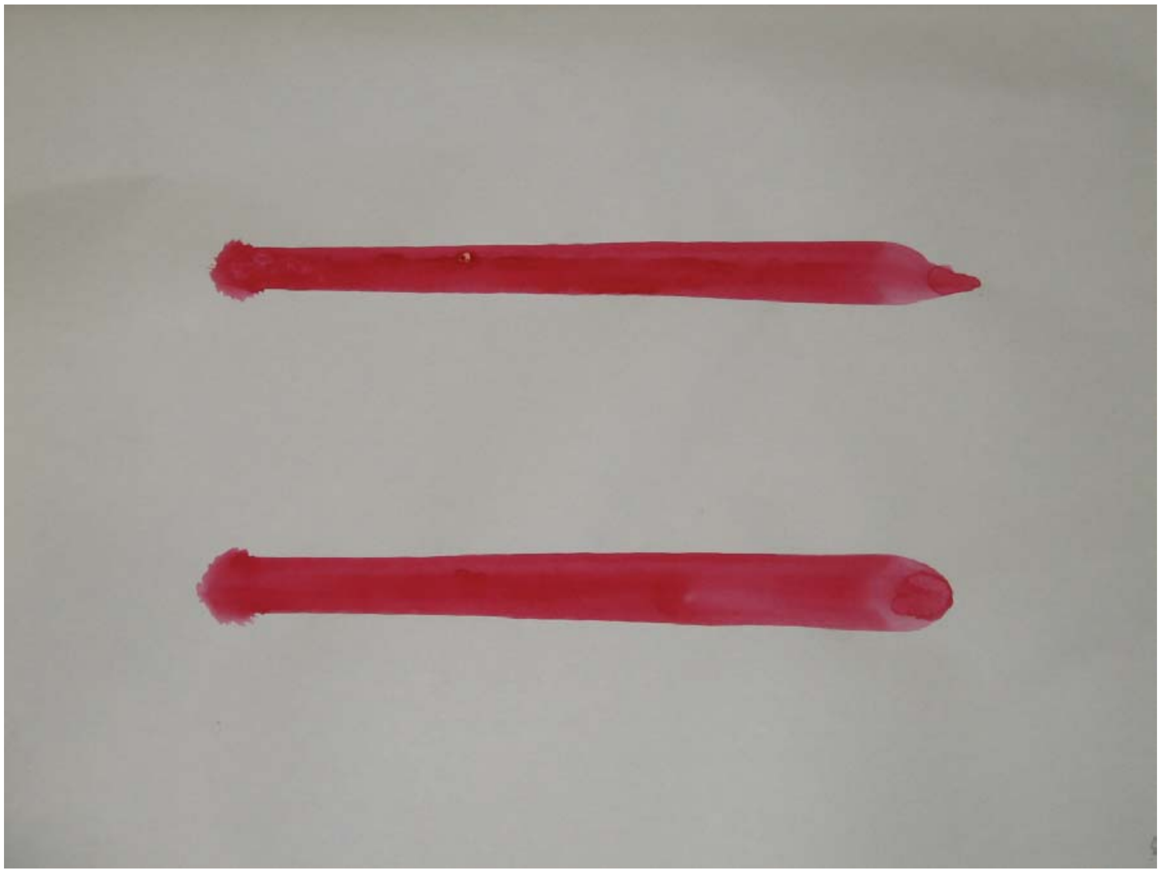


Fig. 6.12 Parameterized paintbrush technique: Pulling brush up gradually and abruptly, respectively.

Next, consider the pressure on the canvas. This can be represented by the distance between the brush and the canvas.

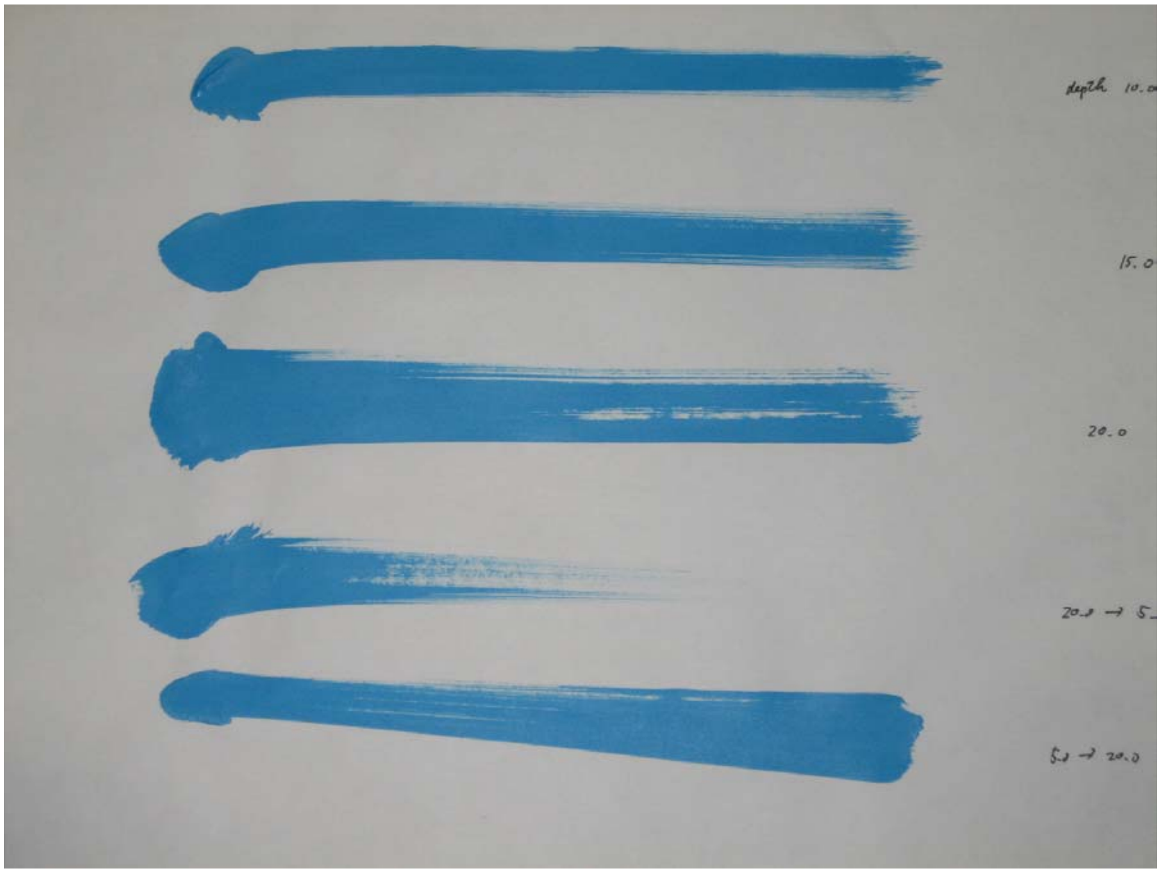


Fig. 6.13 Parameterized paintbrush technique: Changing the depth of vertical axis [mm]; 10, 15, 20, 20 to 5, 5 to 20, respectively.

Human artists use these same techniques in painting as well.

Fig. 6.14 is the result of painting an apple where the picture model is obtained previously. The results are different for every trial because of the error included in detecting the position of the brush by the vision system. This is actually interesting, as it shows the same imperfections that occur when human artists paint.

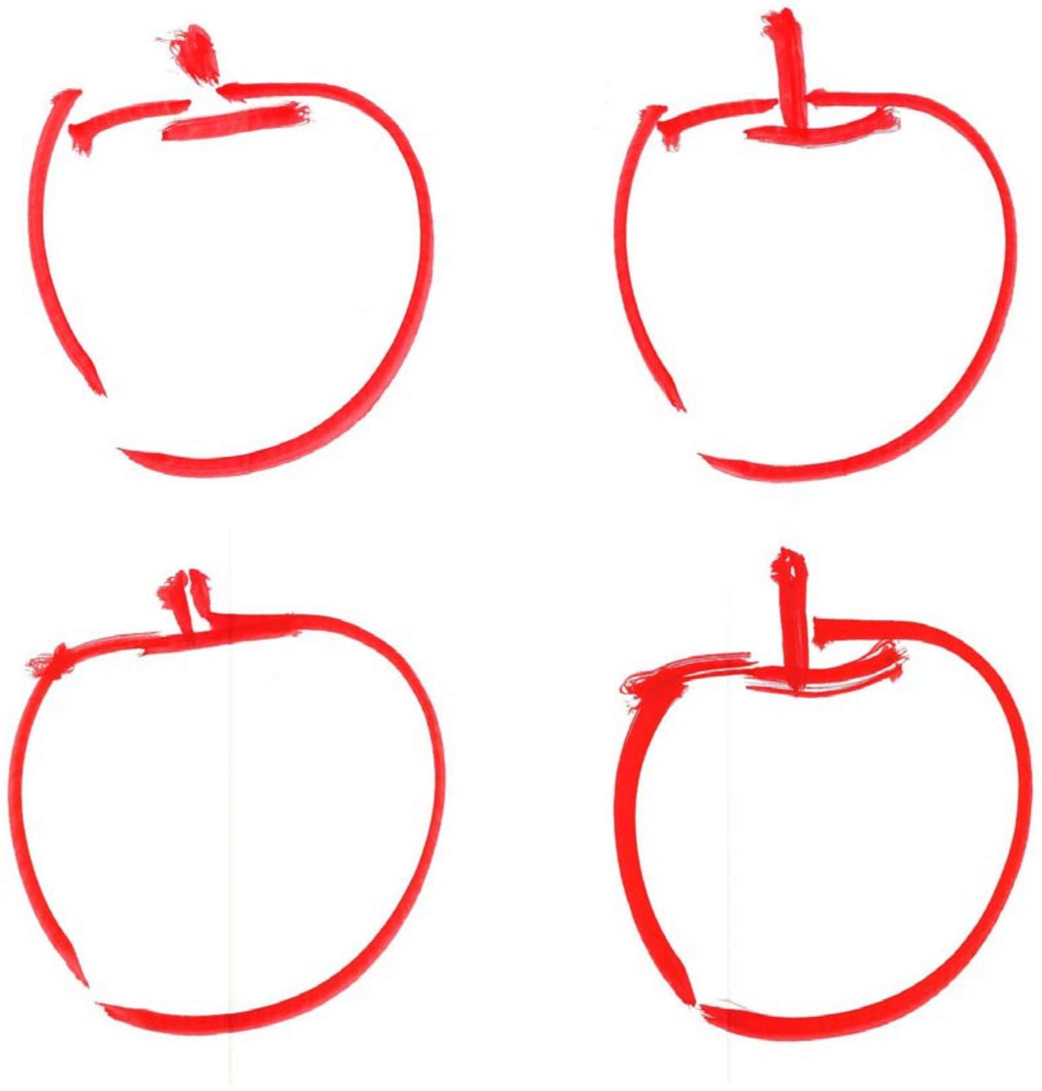


Fig. 6.14 Experimental result.

For our future work, our first priority is filling a region. Since we have already developed a method to generate a picture model for filling, in order to have it performed by a robot with a multi-fingered hand, we need to introduce an additional operation, such as changing paintbrushes and dipping a paintbrush into water color. We are expecting to have a painting like that shown in Fig. 6.15.

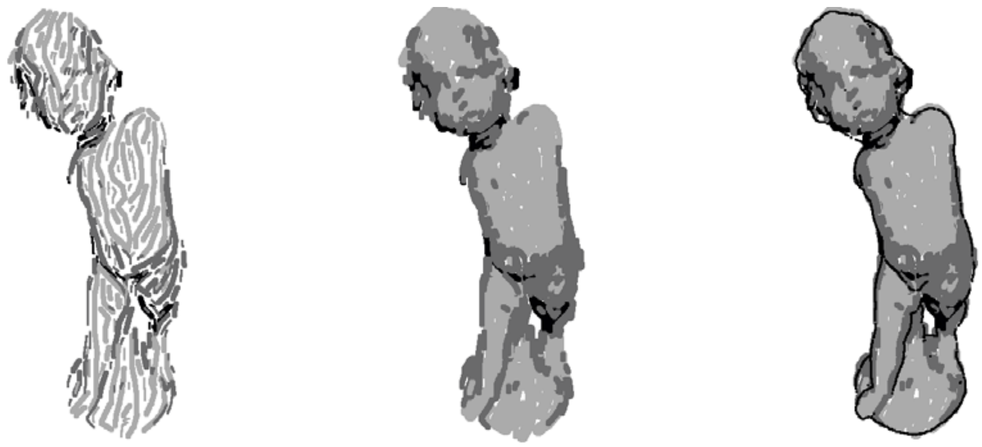

(a) (b) (c)

Fig. 6.15 Combining area filling and edge drawing (simulation).
(a) Brush trajectories. (b) Area filling. (c) Geometry edge drawing.

6.8 Summary

The steps used for brush manipulation are described in this chapter. Stereo cameras on the robot's head are used to locate the brush. Then the robot approaches and grasps the brush, with force sensors in its hand playing an important role in ascertaining the grip. The position of the brush tip is then pre-computed by finding the PCA of the handle and projecting this to a known distance.

When drawing, the robot checks whether the brush touches the canvas or not by calculating the force that needs to be applied to its fingers. A technique to prevent the brush from slipping out of the robot's hand and painting techniques adapted to it are shown, and the experimental result of these techniques is presented.

It is interesting to note that even when the same picture model is used, the result shows that all of the paintings are quite different.

Chapter 7

Conclusion

This research presents vision and manipulation techniques applied to a robot painter, namely, object segmentation, color perception, orientation mapping, geometric edge processing, and then shows how to apply these methods to high-level manipulation.

This chapter presents the contribution of this research (Section 1) as well as extensive discussion (Section 2). As each part of the project can be considered as separate research, contribution and discussion are divided into states.

7.1 Contribution

1) Foreground Segmentation

The first novelty is a supervised comprehensive iterative foreground segmentation (CIFS) based on local and global optimization with edge constraint. In other words, it is a region growing with the cost function similar to graph cut. Then, this concept is used in 3D images to enable the robot to extract the foreground automatically.

The second novelty is our constraint based on Chebyshev's inequality, which was proved to be robust. Such constraint setting was not effective in conventional region growing.

The third novelty is, since both local and global cost functions have their own constraints, a weighting method to perform in the constraint domain instead of in the cost function. This method can also deal with multidimensional features.

The fourth novelty is automatic multiple cuts that are capable of reducing a great deal of user interaction. The notion of growing to an edge constraint based on Chebyshev's inequality before searching is novel and can reduce error significantly.

The final novelty is cut before matting, where some previous works tried to heuristically erode the boundary before matting, the notion of growing to a certain statistical constraint before matting gives a better result than state-of-the-art matting programs.

2) Color Perception Based on Two Clustering Schemes

Since normal K-means clustering usually produces low contrast color and consumes considerable time, we use maximum distance clustering (MDC) prior to an iteration of K-means to solve both efficiency and time consumption problems.

As maximum distance clustering (MDC) is comparatively slow when the number of desired colors is high, a sub-optimal algorithm is proposed and shown to be extremely fast.

Another objective of this research is an algorithm that anyone could easily implement, and this is already achieved since our improvement is based on well-known and easy algorithms, MDC and K-means.

MDC + an iteration of K-means can be applied to clustering in general.

3) Global Orientation

In this research, a linear radial basis function (RBF) is used to smooth the orientation domain, as previous works suggested. We approach the problem differently by focusing also on the speed of the algorithm. Speedup is accomplished by calculating RBF using a fixed size window. For the area that has no orientation information the RBF window is increased until it covers the surrounding area that contains orientation. This substantially reduces the time consumption so that RBF can be performed in real time.

It can be seen that the RBF can be applied not only in a gradient domain but also in other similar domains to generate smooth output.

4) Paint Brush Manipulation

Stereo cameras on the robot's head are used to locate the brush. Then the robot approaches and grasps the brush, with force sensors in its hand playing an important role in ascertaining the grip. Although the position of the brush tip is pre-computed, by finding PCA of the handle and projecting down to a known distance, force sensors are also used for checking whether the brush touch the canvas or not. This process is similar to that used by a human artist.

When looking at the overall manipulation process, one sees that this is the integration and realization of many robotics concepts into one system.

7.2 Discussion

1) Foreground Segmentation

Considering a foreground cut, comprehensive iterative foreground segmentation (CIFS) is as robust as graph cut, since their cost functions are quite similar. Nevertheless, graph cut, in difficult images, applies only to the foreground cut problem.

Using the notion of growing to a certain statistical constraint, before searching for a remote area or before matting, the program can effectively deal with a foreground cut, multiple cuts, and cut before matting.

The multiple cuts method is robust against a not well-structured classifier as well as ambiguous data distribution. This is completely different from the contour evolving approach. Compared to this approach, the main differences are among the algorithms: the proposed method does not split or merge to perform multiple cuts. CIFS relies on data distribution and thus is more straightforward, whereas the other approach seems to be more flexible due to its internal and external cost function.

For cut before matting, the proposed method can generally gives substantially better pre-cut images since it uses Chebyshev's inequality as opposed to heuristic erosion. The scheme can be used as a standalone matting program as well as an input generation for other high-end matting methods.

Furthermore, our novel weighting method, between local and global information, is shown to perform in the constraint domain instead of in the cost function. Although it is difficult to compare this notion with conventional weighting, our scheme is more straightforward if there are local and global components in the cost function.

The method is then applied to perform 3D object segmentation automatically. In order to extract the subject area, this thesis focuses on how to exploit normal stereo cameras to roughly extract the object automatically using a disparity map and 3D background subtraction, and then using CIFS to extract the object area correctly. 3D background subtraction is usually noisy, thus dilations and erosions are required. CIFS, then, exploits the local and global color similarity optimization with a constraint of edge to extract the boundary correctly. This is a promising paradigm, which can be applied to a wide variety of 3D segmentation/detection tasks.

2) Color Perception Based on Two Clustering Schemes

Using maximum distance clustering (MDC) prior to an iteration of K-means benefits not only color reduction application but also to clustering methods in general. K-means should be run for only one iteration to prevent it from dominating the process. Thus, the convergence-speed problem of K-means is not present in our algorithm.

The method can generate a higher quality image than many existing interactive color reduction methods in RGB color space.

However, in human perception color space, CIEL*a*b*, K-means alone is comparable to MDC + an iteration of K-means, provided that K-means is run for two iterations or more, one iteration forward and another iteration backward. Considering time, running MDC + an iteration of K-means consumes approximately the same time as that of two iterations of K-means.

3) Global Orientation

For the local orientation used to guide brush strokes, using a gradient to guide brush strokes is more robust to texture, whereas using image moment is more robust to area orientation. However, local orientation is usually noisy and thus results in inappropriate brush strokes. In this sense, global orientation, which can be described as a smoothed version of local orientation, is applied. In this research, a linear radial basis function (RBF) is selected as the smoothing method.

4) Area Filling

After the foreground, color, and orientation information are derived, the robot can perform brush stroke planning automatically based only on the brush size information. Actually this is the process of visual feedback where the robot:

- Selects a color
- Decides which area to start drawing
- Starts moving based on orientation information
- Pulls brush up
- Visually verifies the canvas, comparing it to the picture model the robot processed initially
- Starts next brush strokes or changes color

If there is no area left to draw, then the painting is considered finished.

At this point, the result of area filling is verified using a computer simulation. This type of visual feedback simulation consumes around 1 second per hierarchy, for a 640x480 image, on a laptop with 1.8GHz CPU; thus, it should be a minimal load for the real robot platform. The problem to be aware of in the real drawing by the robot is how to consider the color mixing on canvas, and how it would affect the drawing as a whole.

5) Geometric Edge Processing

Some human artists use edges to enhance their paintings. Although many researchers use gradient information to represent edges of objects, this distorts the color information on the surface. Hence, we decide to use 3D geometric edges of an object as input for the robot to draw, as these represent the object well.

After range sensors capture multiple-viewpoint images, they are aligned and merged to generate a 3D model.

In this process of 3D model generation, where outliers must be deleted before merging into the final model, we also propose the use of foreground segmentation to delete outliers in range images semi-automatically.

As painting is done in 2D space, 2D edges are extracted from the 3D model. A geometric edge is defined as the common edge at which the sign of the inner product of the normal vector of a patch and the viewing direction changes. Then, the noisy 2D edges are processed into brush strokes by reducing various kinds of redundancy.

6) Paint Brush Manipulation

The brush is used in a manner similar to that used by a human artist. The position of the brush tip is pre-computed by stereo cameras. The robot checks whether the brush touch the canvas or not by calculating the force applied to its fingers.

Furthermore, just as a human artist avoids moving a hand in some awkward direction, a robot's hand needs to avoid an awkward direction. Hence, a technique to prevent a brush from slipping out of a robot's hand is presented. Some painting techniques are shown along with the experimental result.

Seeing the paintings, it is interesting to note that even when the same picture model is used, all of the paintings are quite different. This variation mimics the same quality in human artists.

Appendix A

Robot Constraints

This appendix focuses on space-time motion generation methods with the robot's physical constraints.

It is widely known that motion generation methods for robots have had problems because the physical capabilities of humanoid robots are limited. There are limits relating to physical attributes, such as angle, collision, velocity, force, and balance consistency.

For an industrial robot, the physical constraint problem can be solved by allowing the robot a longer time to finish a task, to satisfy the limits. The main requirement is the precision in the world coordinate, the so-called "space constraint."

With the emergence of the humanoid robot, there is a new problem called "space-time constraint" in which the robot is required to finish a task at a certain time. This is a requirement when the robot is used as a demonstrator (offline), when it is used during plans before moves (offline), when it is teleoperated (online), or when it interacts with other entities (online).

Currently, without considering balance control, many research groups are trying to solve the space-time problem with physical limits. The significance of such motion generation is clear, because in many works balance control is achieved but methods to deal with other physical limits are not effective. If such limits are not satisfied, the trajectories will have a large error, which leads to collision problems and balance inconsistency. If these are not prevented, the robot will be damaged. However, no research was successful in limiting all physical characteristics.

This paper presents the first space-time method that can effectively guarantee angle, collision, velocity, and force limits, based on the B-spline function. The method can be used for both an offline and an online environment.

As our robot platform in Fig. 1.1 is not designed to deal with abrupt movement, we use HRP-2 as the test bed. The test data are the Japanese traditional dances captured from professional dancers as test motions as shown in Fig. A.1. These dances are very complex; their motions exceed many physical limits of our HRP-2 robot and cannot be easily performed by any humanoid robot using existing methods. If our algorithm could deal with such dances, it would benefit other kinds of motion research as well.

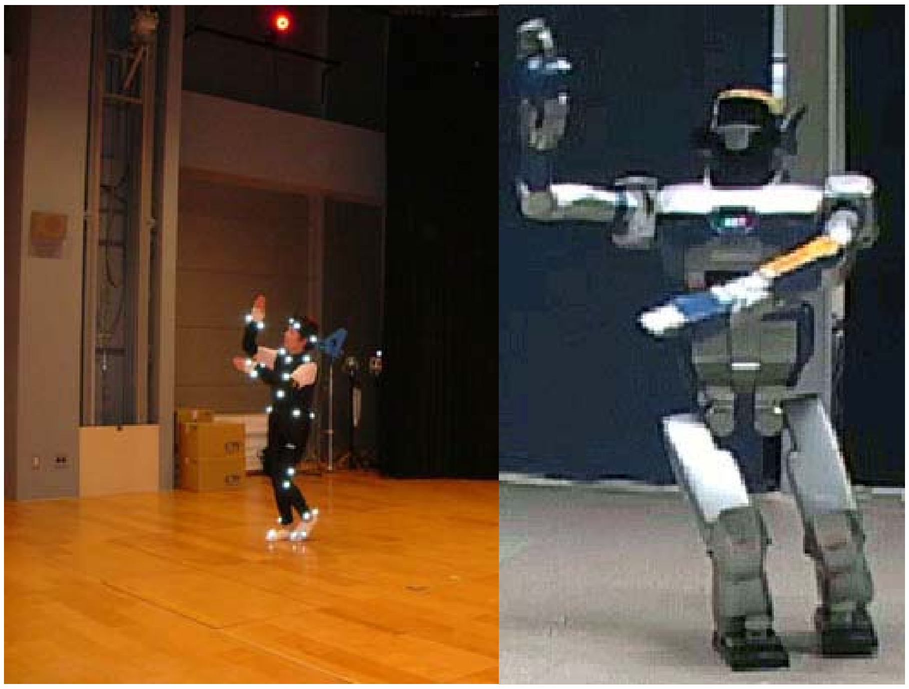


Fig. A.1 Japanese traditional dance performed by dancer vs. HRP-2.

First, let us look at present space-time research.

Among motion generation methods, a filter-based approach is a fast way to retarget motion. [PHRA02] realizes upper-body dances using a humanoid robot by scaling the angle and filtering velocity. However, first, since each joint is scaled separately, the overall motion may be different than the desired path [SPH03]. Second, collision avoidance and dynamic forces are not considered. Third, the method cannot be applied for trajectory optimization since it processes each data input directly while optimization usually considers trajectory as a function. Finally, it cannot be used in real time since the velocity filter requires running it forward and backward.

Trajectory optimization is another approach. [LS99] uses a hierarchical B-spline whereas [UAR04] uses B-spline wavelets. However, their algorithm does not deal with physical limits effectively. The objective function can minimize physical characteristics; however, it is not guaranteed that such values will satisfy limits. Limit violation can occur, requiring a user to manually adjust trajectories, which is very tedious work. On the other hand, if the cost function is set too strictly, the result may be ineffective movement or large errors in end effectors.

A real-time approach, as described in [RUWA03], also presents problems. Attempts to limit physical characteristics rely on kinematics-based optimization to reduce values, rather than applying specific limitations, as described in [DVS01].

These problems have posed very difficult challenges that have caused considerable discussion. This paper solves this challenge by representing all physical limits in the term of B-spline coefficients. Also, most of our constraints are totally different from those of a conventional industrial robot.

The proposed constraints can be used as an offline filtering approach. Furthermore, since the constraints are applied on B-spline curve rather than on raw data, we propose an effective method to decompose a trajectory to a B-spline.

For the offline optimization approach, the proposed constraints can be used directly as the constraint function for this approach. In addition, it is usually necessary to locate the problem period that should be re-optimized in higher hierarchies. To do this effectively, not only the traditional trajectory's error detector but also our preemptive knot density detector [RNKI05] are used.

In real-time approach, our angle, collision, and velocity constraints can be applied directly. Unfortunately, force constraint, which requires iteratively running, might not be suited to this approach. A different technique to limit force is presented.

The remainder of this appendix is organized as follows. We describe the physical constraint functions in Section 1. Section 2 explains how to decompose a trajectory into a B-spline curve and how to use the constraints to filter the curve. Section 3 focuses on issues related to using the constraints in an optimization approach, and motion refinement based on a hierarchical B-spline. Section 4 explains a real-time approach and the issues related to adaptation of the constraints along with some drawbacks. Finally, discussion and conclusion are presented in Section 5.

A.1 Physical Constraint Functions

Presently various problems occur when attempts are made to limit physical attributes. Four physical attributes govern the movement of the robot, namely, angle, collision, velocity, and force. The attributes must be limited for many vital reasons.

For angle and force, if these attributes are not well limited, the robot trajectories would be different from the planned trajectories as the robot does not have the capability to follow the planned motion. At first glance, this seems to be a negligible problem; however, it is not. Error in trajectories could lead to two serious problems, collision and balance inconsistency. Whereas the collision problem due to a violated angle is clear and present, and inadequate force also leads to an incorrect angle, balance inconsistency is more obscure. Balance inconsistency is related to the whole body control of the humanoid robot. There are various methods to control the balance of the robot. These include [HZS06], which shows use of the spline function in biped gate optimization based on zero moment point (ZMP); driving torque analysis, [HKKH03], which analyzes ZMP for arm/leg coordination tasks of humanoid robots; [Kaj02], which involves real-time biped pattern generation; and [TNMY05], which deals with biped walking patterns on slopes. Practically, these methods usually require the upper body motion to be planned prior to adjusting the whole body posture, such as by re-calculating the waist angle [NNK*07]. However, if the upper body motion is different from the planned one, the center of gravity and zero moment would be different from the planned whole body motion, and the robot could collapse. This must be prevented by applying a method that effectively limits angle and force. Though angle limit can be achieved easily, force is often not well limited due to the complexity of the dynamic equation. Most previous works can only reduce the force attribute, rather than giving it definite limits, as our method does.

Velocity limits pose another problem. Usually, for most kinds of electric actuators used in humanoid robots, there is a back electromotive force (emf) that increases proportionally with velocity. Excessive emf could lead to undesired actuator wear. Although [PHRA02] proposed a method to filter velocity trajectories, this method requires running forward and backward iteratively; hence, it cannot be used in a real-time approach. Also, the method does not consider force filtering or collision avoidance.

As mentioned before, collision is a clear and present problem that must be avoided to prevent any damage to the robot. Various collision avoidance methods can be found in literature, such as [KL06], which focuses on a cylindrical body, [YES*06], which shows whole body collision avoidance of a humanoid robot, and [SKH05], which contains the notion of check points. The idea of check points placed on the robot's body is adopted in this work. In this paper, the problem of collision avoidance is not considered as an objective function to be reduced but as a constraint to be limited. Our experiment proved this to be effective, compared with methods such as [ZN02], which only uses an objective function.

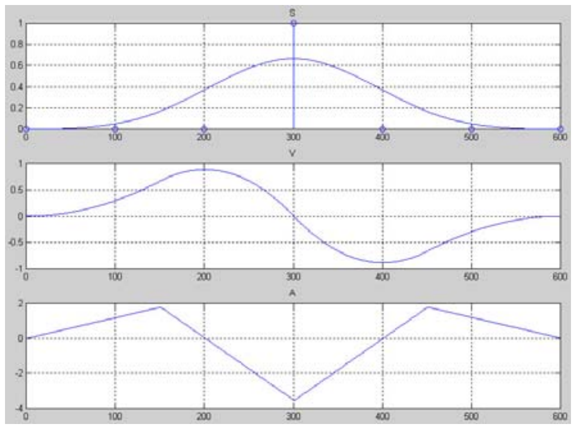
Another must-be-considered aspect is a data representation method. In order to ensure the smoothness of trajectories and to create the constraint functions that are compatible with various motion generation methods, namely filtering, optimization, and real-time-approaches, a curve representation method is required. When employed, instead of processing a trajectory as a set of points that could lead to a jerky motion, a trajectory would be altered by adapting parameters of the curve representation function. The choice of curve representation method is very important as it determines how the constraint functions work. Constraints are required to limit physical attributes of such curves directly, which means that constraints are to be implemented based on the curve representation method's parameters. Traditionally, even though a curve representation function is used, its role is to ensure smoothness or to perform successive refinement only, as shown in [LS99], [UAR04], and [AMH01], whereas in this work the physical attributes are also limited based on a curve representation function. This is a major advantage of this method, which differs from other methods in this respect.

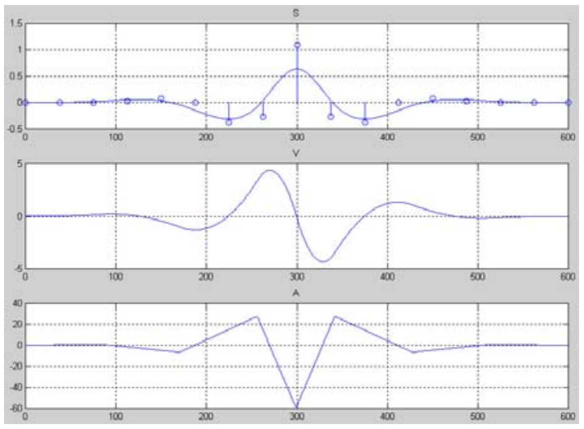
The following portions of this section start with a discussion of curve representation. Then, based on the best representation method, the physical constraints of angle, collision, velocity, and force from [RNKI06] are explained. The velocity and force constraints are influenced by an iterative soft-constraint paradigm [RNKI06] that makes limiting very effective.

1) Data Representation

In the robotics field, B-spline is widely used for manipulator motion-planning [KT03] and even for intelligent control [FZK99]. The important characteristics of B-spline are, first, changing a parameter of the B-spline function affects only the limit range of a curve and, second, the method involves hierarchical refinement. For the space-time problem of a human-like figure, [LS99] has used a hierarchical structure of B-spline to generate trajectories.

Recently, wavelet [CDF92] is used in many fields. As wavelets share the two important desirable properties with B-spline, some previous works [Got95] have compared the advantages of these techniques. [UAR00] proposes to use B-spline wavelets for trajectory optimization. Their comprehensive work can be found in [UAR04]. It is understandable that the authors are trying to enhance the convergence characteristics of B-spline by adding wavelets, as the latter converge faster than the former if the trajectory contains an inadequate space constraint [Got95]. This, however, is not the case for motion generation, as a large amount of data can be derived from a human trainer or even from the trajectories the robot plans itself.





(a) (b)

Fig. A.2 Scale, velocity, and acceleration of B-spline and wavelet.
(a) B-spline. (b) Wavelet.

With the criterion in mind that the method must be exploitable for constraint functions, B-spline is the method of choice as its angle, velocity, and acceleration functions (also implying the force function) have a clear structure compared to wavelets [RNKI05]. Furthermore, B-spline is superior to wavelets for its usability in a real-time approach. As will be shown in Section 4, using an online force limit faces a problem due to B-spline's acceleration function. But since a wavelet's velocity shares a physical structure similar to a B-spline's acceleration, it would be ineffective to use wavelets in real time for both velocity and force limiting.

Let's look back a little on the use of B-spline for physical constraint. Industrial robots have long been using B-spline with physical constraints in trajectory planning, [SM85], [SY89]; however, such methods apply limits by scaling the time domain, which is not applicable for a space-time problem. In this work, space-time B-spline based constraints are proposed as described in the following sections.

2) Angle Limits

Cubic B-spline has a characteristic that its amplitude will not be higher than the magnitude of a control point. Hence, angle limiting can be done directly by applying bounded constraints to the magnitude of control points in a B-spline function (of one knot period) as shown below.

$$\begin{aligned} q(t) = \frac{1}{6}\{&(-t^3 + 3t^2 - 3t + 1)p_0 + (3t^3 - 6t^2 + 4)p_1 \\ &+ (-3t^3 + 3t^2 + 3t + 1)p_2 + (t^3)p_3\} \end{aligned} \tag{A.1}$$

where q is angle, t is time, and p_n is control point.

3) Self-Collision Avoidance

In order to decide whether collision occurs or not, check points are placed on or inside a robot body and an arm is considered as a link of cylinders. Collision is detected if the distance between a cylinder and any check point is lower than a certain value.

All angles responsible for moving such a cylinder are then searched to see which one requires the least angle change to avoid collision.

$$j = \arg\max_{j}(d / \Delta p_{n,j}) \tag{A.2}$$

where j is the joint number, d is the distance from the collision point, $\Delta p_{n,j}$ is the change of value in control point number n of joint j, where n is the closest control point to the collision period.

For example, to avoid collision on the lower arm, three joints in the shoulder and a joint in the elbow are explored. The result of avoiding collision of the head is shown in Fig. A.3.

Actually, we have tried to do collision avoidance using a cost function proposed in [ZN02], and found that solving it using a constraint is more effective and straightforward for choosing critical distances from collision check points.

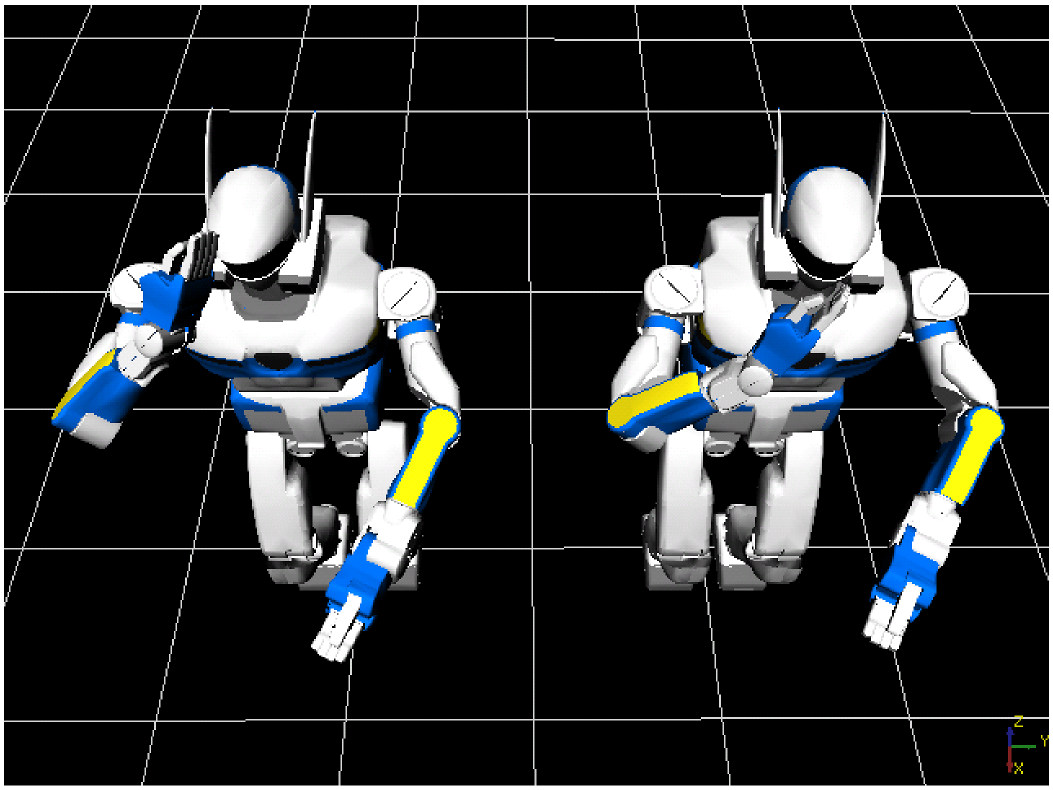


Fig. A.3 A collision check point is placed at the center of mass of a robot's head. (left) Without collision avoidance. (right) With collision avoidance.

4) Velocity Limits

Our velocity limit checking is done on B-spline function at the beginning and the middle of each period as well as at the peak velocity. From (A.1), velocities at the beginning of a period and at the middle of a previous period are:

$$\dot{q}(present : begin) = \frac{p_2 - p_0}{2T} \tag{A.3}$$

$$\dot{q}(previous : middle) = \frac{p_2 + 5p_1 - 5p_0 - p_{-1}}{8T} \tag{A.4}$$

where T is the length of the knot period.

Interestingly enough, (A.3) must be checked before (A.4) to avoid divergence of trajectory, by altering p_2 and p_1, respectively. This is based on two criteria. The first criterion is the sensitivity of altering the control point (a higher sensitivity control point affects the shape of a curve more). This must be high for checking the present period and low for past or future periods.

It can be seen that there are similar ways that meet the criterion; however, these will also lead to divergence or ineffective limiting. So the second criterion is required:

after changing p_2 in (A.3), altering p_1 in (A.4) does not affect (A.3) and affects previous checking only in a supportive way.

It is likely that the peak velocity would not occur at the beginning or the middle of a knot period. So we also check the peak velocity which is in the term of (A.5).

$$\dot{q}_{\max} = \frac{1}{2T}[(p_1 - p_{-1}) + \frac{(p_1 - 2p_0 + p_{-1})^2}{-p_2 + 3p_1 - 3p_0 + p_{-1}}] \tag{A.5}$$

To limit, it has to be determined which control point provides the highest sensitivity for the function, which is p_1.

However since p_1 is in a second-order term, solving it directly may yield complex numbers. Hence, a search method is required, and a method like hill climbing is adequate.

5) Dynamic Force Limits

The force in each joint can be calculated from this inverse dynamics equation:

$$F_i = \sum_j M_{ij}(Q)\ddot{Q}_j + I\ddot{Q}_i + \sum_{j,k} C_{ijk}(Q)\dot{Q}_j\dot{Q}_k + G_i(Q) \tag{A.6}$$

where *F* is the applied force, *M* is the inertia matrix, *I* is the actuator's inertia of the present joint, *C* is a centripetal and Coriolis forces matrix, *G* is gravitational loading, and *Q* is a set of all joint angles.

M greatly influences the dynamics equation, and the acceleration multiplies *M*. Since the acceleration is a straight line, its peak values are located at the beginning of each knot period; therefore, limiting force only at the beginning of a knot period is sufficient.

The above equation is used to limit values by altering a control point to alter *Q* and its derivatives, first inserting velocity and acceleration in the form of a B-spline. Note that since the dynamic equation's parameters are nonlinear in *Q*, it can be considered to be constants that are updated recursively.

$$\begin{aligned} F_i = &\sum_j M_{ij}(\frac{p_2 - 2p_1 + p_0}{T^2})_j + I(\frac{p_2 - 2p_1 + p_0}{T^2})_i \\ &+ \sum_{j,k} C_{ijk}(\frac{p_2 - p_0}{2T})_j(\frac{p_2 - p_0}{2T})_k + G_i \end{aligned} \tag{A.7}$$

From (A.7), it can be seen that since the term multiplies *M*, changing p_1 provides the highest sensitivity. So if one rearranges the function to isolate terms that contain p_1 from those that do not, the following equations result.

(A.8)

$$F_i = \sum_j (M_{ij} + I)(\frac{-2p_1}{T^2})_j + \sum_j (M_{ij} + I)(\frac{p_2 + p_0}{T^2})_j$$
$$+ \sum_j \sum_k C_{ijk} (\frac{p_2 - p_0}{2T})_j (\frac{p_2 - p_0}{2T})_k + G_i$$

This is a set of nonlinear equations that cannot be solved explicitly. So they are solved by recursively searching for the value p_1 of joint *i* alone that satisfies a limit. The hill climbing method is used for this recursive search.

A new problem arises upon the use of this method: changing p_1 often results in the force of a previous period being larger than the limit, making force constraint ineffective. This can be solved using the method described next.

6) Iterative Soft Limits

Suppose that a value that needs to be reduced is 1. The easiest way to do this is 1-1 = 0; however, this poses the problem stated above, in case that such value represents force. Another way to do this is:

$$1 - y\sum_{i=0}^{\inf}(1-y)^i = 1 - 1 = 0; y < 1 \quad \text{(A.9)}$$

In the case of force constraints, instead of limiting force to the desired value, reduce it only *y* of the needed amount (present force minus force limit value) iteratively, say, 0.5 with 46 iterations.

$$1 - 0.5\sum_{i=0}^{45}(1-0.5)^i = 1 - 0.99999999999999 \approx 0 \quad \text{(A.10)}$$

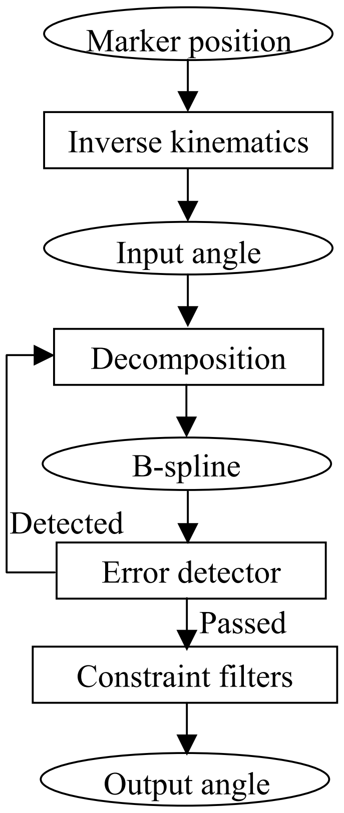
Although the reduced value does not exactly equal the portion needed, such difference is extremely low and clearly negligible, as shown in (A.10).

Furthermore, if both velocity and force constraints are to be used separately, the resulting trajectories would meet force limits while velocity limits are often violated. Fortunately, if both are put under the iterative soft constraint, for offline approaches, they will gradually converge to limit values. For an online approach, the soft constraint, which requires running forward and backward iteratively, cannot be applied. Hence adaptation of force constraint function is required and will be explained in Section 4.

A.2 Filtering Approach

The fastest and easiest way to retarget original motion to the joint angles that can be represented by a robot with physical limits is filtering. Previous filtering approach work, such as described in [PHRA02], is not effective enough in the sense that it does not consider all important physical attributes, while our work does.

Since all of the constraints operate on a B-spline function in order to ensure smoothness, it is required to decompose a joint trajectory to a B-spline curve prior to limiting physical attributes.



Overview of filtering approach.

1) Decomposition From Raw Data

Unfortunately, it is inappropriate to use B-spline hierarchical decomposition, [CQ92] or [FB88], in cases where it is extensively used for an optimization approach such as [LS99] or as described in Section 3 of this appendix. Let us explain the differences. Differing from the optimization approach, where the appropriate value of control points is unknown at first, the filtering approach needs only a set of control points that can precisely represent the original trajectory. It is more like one-shot filtering on an already decomposed B-spline curve that first represents the original data.

On the other hand, if a trajectory is downsampled and then the derived samples are used as control points for a B-spline, the decomposed curve will have a large error around the end of it as shown in Fig. A.6 (a).

We observed that the downsampling error occurs due to lack of synchronization between the number of control point periods and the number of knot periods. From (A.1), it is shown that a knot period consists of three control point periods instead of only one control point period.

To synchronize these two domains, a redundant control point is placed at each end point. From Fig. A.5, it can be imagined that at each end of the curve there will be the same two control points, so the result is like a control point period, between a and b, corresponding to a knot period, generated from the four control points, as required. The result of this decomposition method is shown in Fig. 6 (b).

In order to acquire an appropriate sampling rate, we use the error detector shown below.

$$e(n) = | q(n,P) - \theta(n) | > thErr \tag{A.11}$$

where n represents discrete time, P is a set of the present angle's control points, $q(n,P)$ is the curve magnitude of B-spline, $\theta(n)$ is the angle from inverse kinematics, and *thErr* is the threshold for error.

If the curve is not precise enough, we increase the sampling rate.

2) Filtering Based on The Proposed Constraints

After the B-spline curve is derived, a set of constraints in Section 1 can be applied to the B-spline curve directly. For a combination of constraints, after angle limit, collision avoidance is checked prior to the other constraints. This order is adopted because collision avoidance may pose discontinuity in trajectories, which can be solved by force and velocity constraints. Collision may occur, but it can be solved automatically by increasing the critical distance.

Users might prefer to set the initial and final postures of joint trajectories to be the same as the original data. This can be achieved by fixing the first and the last three control points. Angle and collision constraints must be given higher priority than fixing the end point, as can be seen in Fig. A.7. Furthermore, due to forcing of the end point position, an abrupt surge in the force domain can be seen. Practically, actuators characteristic usually allows instantaneous force to be more or less substantially higher than the constant force limit. Since three control points are fixed, the surge would occur only for two knot periods, which is not a problem since the duration is very short.

If the end point is allowed to change, the result of an open-end curve is shown in Fig. A.8.

The effect of approximating the dynamic equation to be linear, (A.7), can be seen more clearly when the limit is set to be stricter, for example, in Fig. A.9. Some small spikes that occur are a little higher than the force limit, which is not a problem since actuators' characteristic allows instantaneous force.

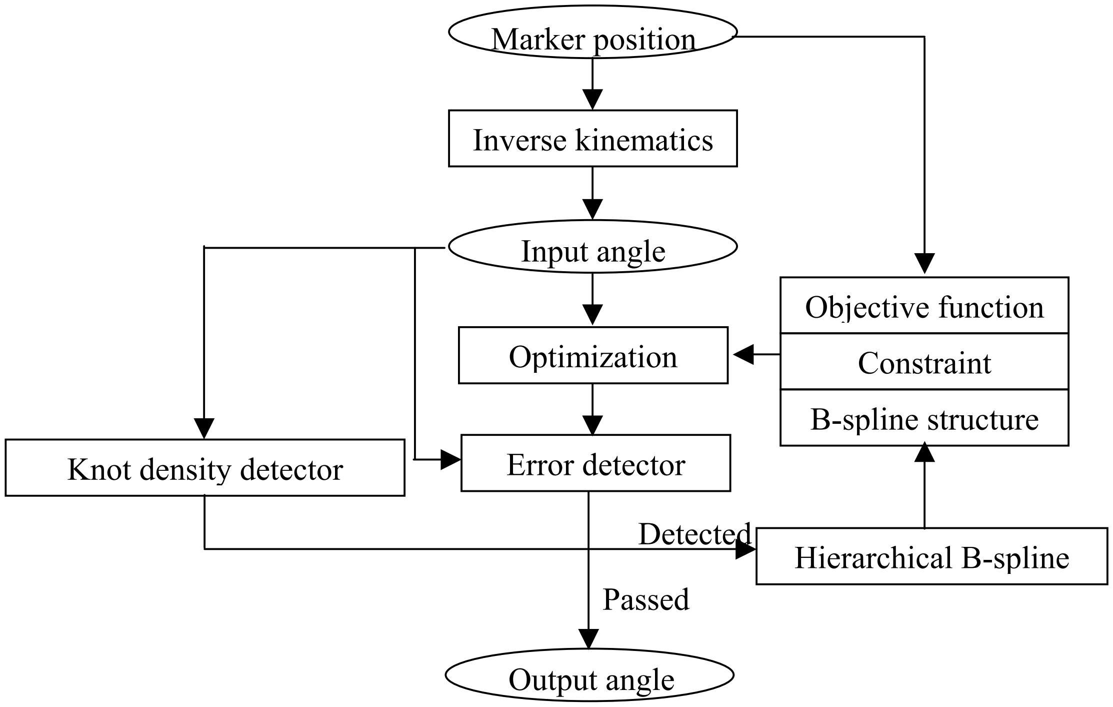
Compared to the optimization result in the next section, the filtering approach gives a more precise trajectory since there is no influence due to the optimization process. However, the disadvantage of this approach is that the multiple objectives may not be realized if any of them cannot be formed in term of constraint. In that case, the optimization method is required.

A.3 Optimization Approach

Traditionally, using a cost function to reduce physical characteristics, as described in [UAR04] or [RGBC96], is inadequate since it does not apply specific limits to those values. Limit violation can occur, especially in a multiple-objectives task, requiring a user to manually adjust trajectories, which is very tedious work. On the other hand, if the cost function is set too strictly, it cannot exploit all the capacity of the robot, which often results in ineffective movement or a large error in end effectors.

Hence in this study, physical attributes are limited using constraints, as shown in Section 1, while a cost function is used to preserve the essence of original input motion and will be explained in this section.

Although a cost function is optimized and constraints are checked, the curve may still not be as precise as expected. In that case, optimization is done hierarchically, first using a small number of control points to represent a curve, then if the precision is not satisfied, decomposing the original set of control points to a larger number of control points and re-optimizing it. Prior to decomposition, two methods are used to detect the problem of the present hierarchy curve, novel knot density approximation [RNKI05] and error detection (A.11).



Overview of optimization approach.

1) Optimization

Based on the B-spline function, optimizing the joint angles is usually enough for a robot to represent human motion, so an acceptable objective function is:

$$f(P)=\sum_{n=1}^{N}\| Q(n,P)-\Theta(n)\|_2 \tag{A.12}$$

where N is the sampling length of a trajectory, and Θ is a set of all angles derived from inverse kinematics.

From Fig. A.1, the markers attached to the body of the human via an elastic suit may be not balanced, or they may slip during the capture process, so some of the sequences of angles computed by inverse kinematics may have large errors. Therefore, another objective function that considers both angles and end-effector is used for such motions:

$$f(P)=\sum_{n=1}^{N}\{\| Q(n,P)-\Theta(n)\|_2 + w\| FK(Q(n,P))-h(n)\|_2\} \tag{A.13}$$

where w is weight derived from experiment, FK is forward kinematics , and h is the hand position of the trainer.

One may choose to use only (A.12), trading off time to calculate forward kinematics. In this research, the decision to use (A.12) or (A.13) depends on the designer's subjective judgment of the data after inverse kinematics. Adding more cost functions is possible.

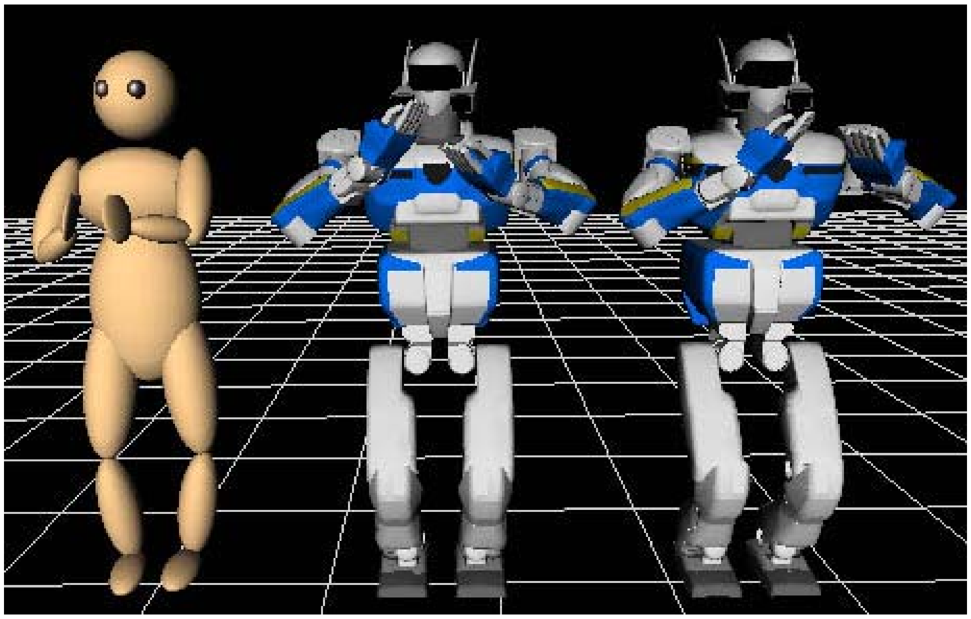


Fig. A.4 Positions of markers, inverse kinematics, and optimization using (A.13).

Constraints can be applied very much like the filtering approach, except that there is no redundancy control point at the end of each curve, as in Section 2.

Note that the global optimization program described in [HN98] is used in this research with some adaptation so that many constraints can be used.

2) Motion Refinement

For many reasons, optimized trajectories may contain errors, caused by such factors as the fact that the number of control points is not appropriate, for example.

Fortunately, the B-spline function allows a hierarchical structure so that a trajectory can be re-optimized, using a larger number of B-spline bases by the process called knot insertion. Prior to knot insertion, there must be criteria to decide whether the present hierarchy's curve needs to be re-optimized or not.

First, the convergence could be made faster with less error by assigning an appropriate hierarchy to each part of a trajectory using our proposed knot density detector [RNKI05].

Furthermore, even if the density of a control point is appropriate, error can occur for reasons such as the optimization process does not fully converge. Hence, along with knot density approximation, a traditional error detector (A.11) is applied.

Now, if a density problem or error is detected, that set of B-spline coefficients must be fed into a hierarchical B-spline decomposition, [CQ92] or [FB88], to generate a new B-spline that has a greater number of control points.

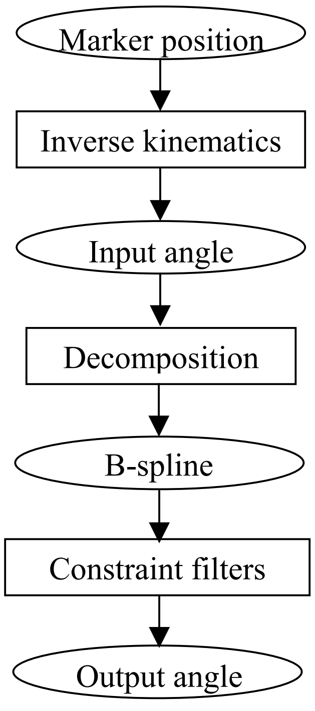
This new set of control points, at the period of the joint angle trajectory that has a problem, is optimized again to find a new optimum set of points for the B-spline. The cost functions and constraints can be applied in a higher hierarchy without any adaptation required.

A.4 Real-Time Approach

As opposed to filtering or optimization approaches where the entire input trajectories could be derived a priori from a human trainer or from a robot's self-generated path, real-time motion generation focuses on how to teleoperate the robot with time delay [LS06], plan its path in a dynamic environment [LL95], or on force-feedback [MO06]. Attempts to limit physical characteristics rely on kinematics-based optimization to reduce values, rather than supplying specific limitations, [DVS01].

This research focuses on how to control the robot in real time using constraint functions. Among the novelties are, first, the fact that, since the trajectory is represented by a B-spline, the smoothness of the output trajectories will always be ensured. Second, many physical limits are guaranteed. However, since all the constraint functions operate on a B-spline parameter, new input time frames must be decomposed into the B-spline, and this introduces delay into output trajectories, which will be explained next.

Even though angle, collision, and velocity constraints can be used directly, force constraint requires running forward and backward iteratively, and cannot be used in a real-time scheme. A method to limit force at each time frame is proposed in this section.



Overview of real-time approach.

1) Real-Time Decomposition From Raw Data

In a manner much like the decomposition technique presented in Section 2, new time frame data are downsampled and their amplitude values are used as control point values.

There are two interesting aspects different from offline processing. First is sampling rate. Offline decomposition can choose the appropriate sampling rate from an error criterion (A.11). However, an online method does not know the data in advance. So the sampling rate is subjectively set based on the trainer's capabilities. For example, in the case that original data is derived from a human, a sampling rate of 30 control points per second would be appropriate. In our experiment, a sampling rate of 20 control points is set to compare with offline approaches.

Second, it is important to consider what kind of model of decomposition would be appropriate. Formerly in a filtering approach, a redundant control point is inserted at the beginning and at the end of the trajectory. In a real-time approach, this introduces delay in processing. Fig. A.11 (a) shows that it is necessary to wait for the third sampled control point, c, because a knot of B-spline consists of 4 control points. Hence, the processing delay introduced is a sampling rate lag. However, the original curve is well preserved, as can be seen in Fig. A.12 (a).

On the other hand, if the two redundant control points are placed only at the beginning of the curve as shown in Fig. A.11 (b), after the second control point, b, arrives, filtering could start immediately. It is not surprising that these inserts introduce a delay in output trajectory as shown in Fig. A.12 (b). The delay, in this case, is not a processing delay as in the former decomposition method but is a physical delay that occurs in output trajectory and must be avoided.

Hence, the approach of inserting a control point at the beginning and the end would be used.

2) Real-Time Limiting Based on The Proposed Constraints

After the B-spline curve is derived, most of the constraints in Section 1 can be applied to the B-spline curve directly, namely, angle, collision, and velocity. The result after passing through these filters is shown in Fig. A.13.

However, for force equation (A.8), changing the highest sensitivity control point affects the former period in an unwanted manner. This can be solved in an offline case using the iterative soft limit. Unfortunately, it cannot be applied in real time. So, if (A.8) is solved directly, the resulting force trajectory is not well limited as shown in Fig. A.14.

Hence, it is necessary to modify the force constraint function. From (A.7), considering the term that multiplies *M*, instead of changing the highest sensitivity control point, p_1, we choose to change the control point that does not affect the force of the previous period, which is p_2. Then we rearrange the function to isolate terms that contain p_2 from those that do not. As a result, (A.8) would be superseded by (A.14).

$$F_i = \sum_j [\frac{2M_{ij}}{T} + \sum_k C_{ijk} (\frac{p_2 - p_0}{2T})_k](\frac{p_2 - p_0}{2T})_j$$
$$+ \frac{2I}{T}(\frac{p_2 - p_0}{2T})_i$$
$$+ \sum_j M_{ij} (\frac{-2p_1 + 2p_0}{T^2})_j + I(\frac{-2p_1 + 2p_0}{T^2})_i + G_i \quad \text{(A.14)}$$

Using this new constraint function, force can be well limited as shown in Fig. A.15 (c). Nevertheless, since p_2 is not the highest sensitivity control point, a considerable change in its value is needed, so that joint trajectory (or velocity trajectory) violates its limit, as can be seen in Fig. A.15 (a). This is the drawback of real-time force constraint.

A.5 Discussion

The proposed physical constraints based on the B-spline function bring many benefits into motion generation, not only at the offline level, but also at the online level. It is the first space-time method that can ensure angle, collision, velocity, and force limits. This is possible by using our proposed B-spline-based equations, which, on the whole, are totally different from those for industrial robots.

Considering using the constraints offline, the proposed iterative soft constraint is one key to success in the comprehensive constraints. First, it makes force limiting, which was not available before, possible. Second, it prevents conflict between velocity and force constraints when all constraints are used simultaneously.

Then, for the offline filtering approach, a novel decomposition from raw data is shown to generate a B-spline curve that resembles the original trajectory with less error than existing methods. After that, all the proposed constraints mentioned above can be used as filtering.

Furthermore, the offline/online optimization algorithm can use the proposed constraints as its constraint function directly. For offline optimization, the proposed density scheme and error in the trajectory are used as criteria before decomposing a B-spline to a larger number of control points that can be readapted. For online consideration, the best way to decompose the B-spline is shown. Although it produces a delay equal to one sampling rate lag compared to existing methods that use kinematics-based optimization, the sample rate can be set high enough so that the delay is negligible.

The only factor that could deter progress toward a generalization of using B-spline-based constraints is the problem of real-time force constraint that could violate other limits. However, our online method can ensure a smooth trajectory with angle, collision, and velocity limits. Actually, this is not a drawback since, in a dynamic environment, it may be better to put force in a cost function and optimize the force of all joints as a whole. If the force is too great, the robot cannot move to the desired position, similar to a situation when a human attempts to lift objects that are too heavy. Such cost function is beyond the scope of this paper.

To summarize, among the offline manipulation possibilities offered by our approach are 1) using the robot as a demonstrator, 2) picking up objects whose static parameters can be approximated prior to the action (to be used in force dynamic equation (A.8)). For online cases, with the appropriate cost function, the method can be used 3) for smooth teleoperation as well as 4) when the robot interacts with objects or other entities.

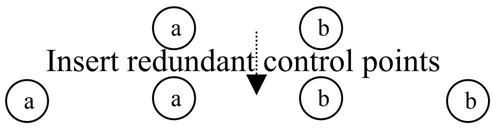


Fig. A.5 Inserting a redundant control point at the beginning and the end of the trajectory.

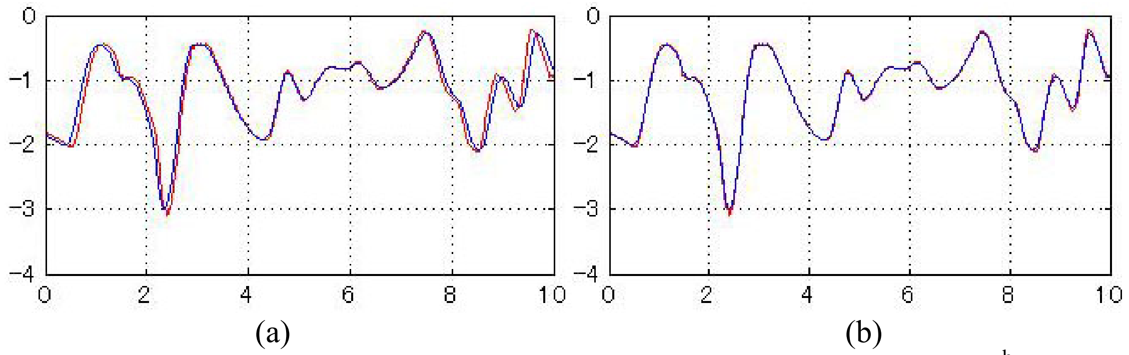


Fig. A.6 Decomposition from 200 fps to 20 control points per second[b].
(a) Conventional method. (b) Our method.

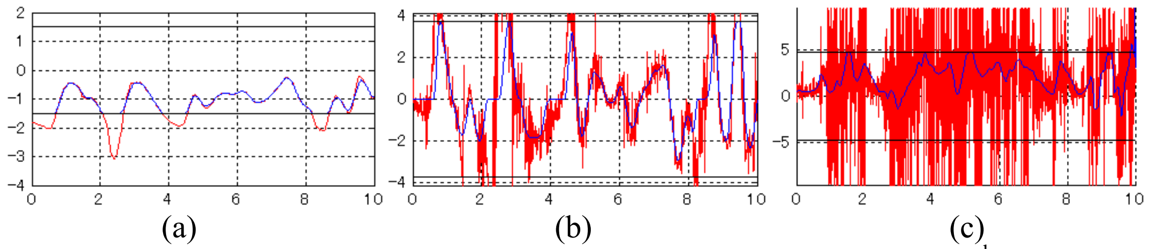


Fig. A.7 Apply constraints simultaneously with fixed end posture[b].
(a) Angle [+/-1.5184]. (b) Velocity [+/-3.74129]. (c) Force [+/-4.8112].

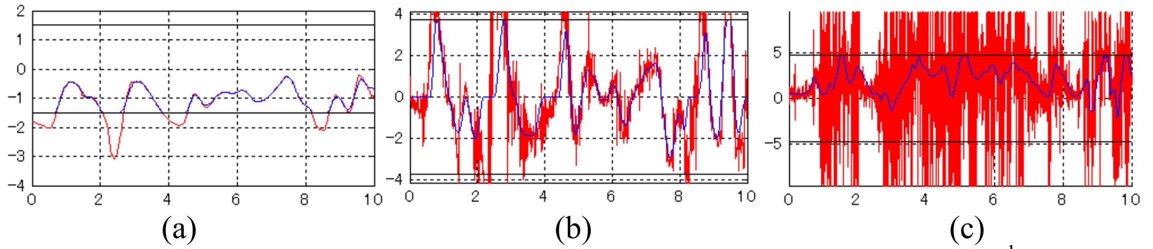
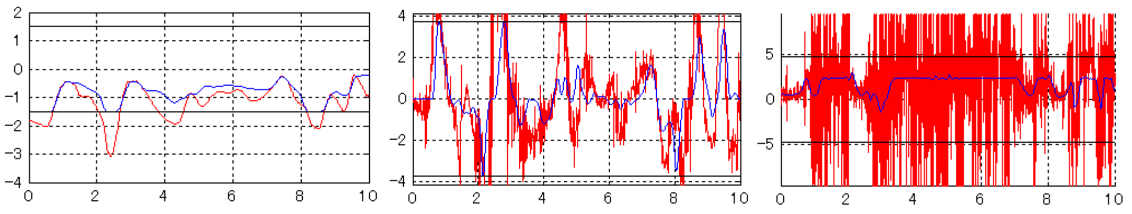
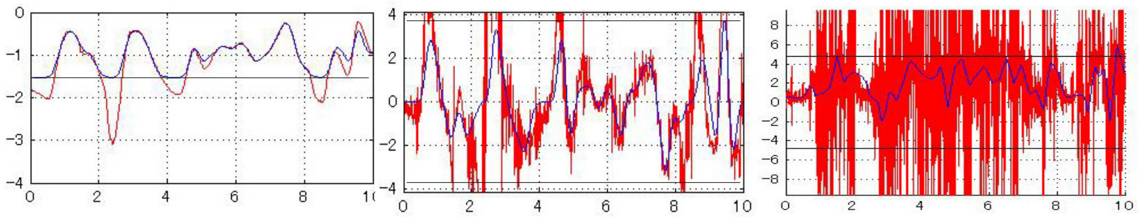


Fig. A.8 Apply constraints simultaneously without fixing end posture[b].
(a) Angle [+/-1.5184]. (b) Velocity [+/-3.74129]. (c) Force [+/-4.8112].



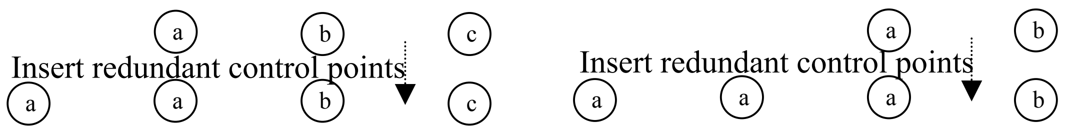
(a) (b) (c)

Fig. A.9 Apply constraints simultaneously with strict force limit[b].
(a) Angle [+/-1.5184]. (b) Velocity [+/-3.74129]. (c) Force half of [+/-4.8112].



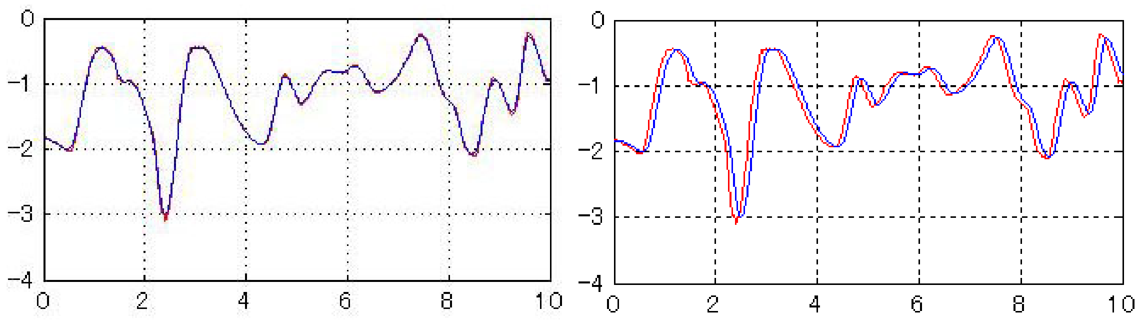
(a) (b) (c)

Fig. A.10 Optimization result with fixed end posture[b].
(a) Angle [+/-1.5184]. (b) Velocity [+/-3.74129]. (c) Force [+/-4.8112].


Insert redundant control points
a
a
b
c
a
a
b
c
Insert redundant control points
a
b
a
a
a
b

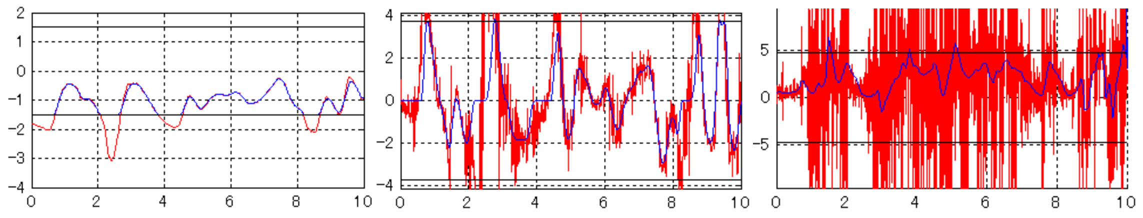
(a) (b)

Fig. A.11 Inserting redundant control points.
(a) A control point at the beginning (and, although not shown here, at the end. (b) Two control points at the beginning.



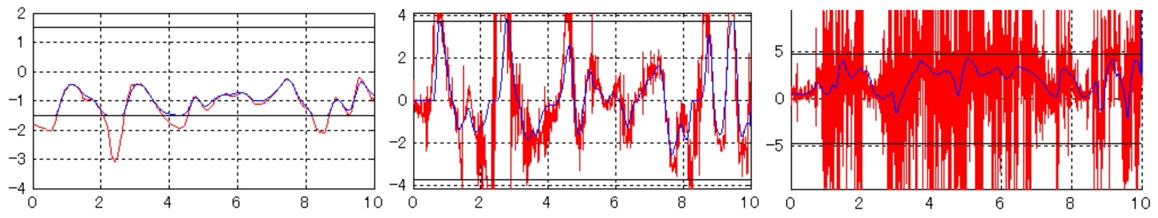
(a) (b)

Fig. A.12 Real-time decomposition from 200 fps to 20 control points per second[b].
(a) When a control point is added at the beginning and the end. (b) When two control points are added at the beginning.



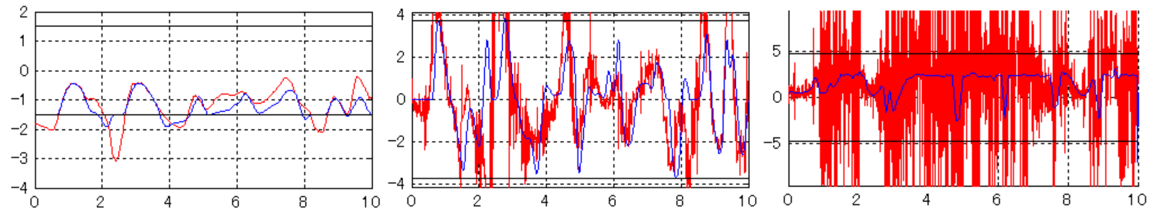
(a) (b) (c)

Fig. A.13 On-the-fly application of angle and velocity constraints simultaneously[b].
(a) Angle [+/-1.5184]. (b) Velocity [+/-3.74129]. (c) Force.



(a) (b) (c)

Fig. A.14 On-the-fly application of angle, velocity, and force constraints simultaneously[b].
(a) Angle [+/-1.5184]. (b) Velocity [+/-3.74129]. (c) Force half of [+/-4.8112] cannot be well limited.



(a) (b) (c)

Fig. A.15 On-the-fly application of angle, velocity, and real-time force constraints simultaneously[b].
(a) Angle [+/-1.5184] is violated since real-time force limiting. (b) Velocity [+/-3.74129].
(c) Force half of [+/-4.8112].

[b]Red, blue, and black lines represent original data (velocity is calculated from the angle, force is calculated from the velocity), generated data, and limit lines, respectively.

Appendix B

Chebyshev's Inequality

Pafnuty Chebyshev, a Russian mathematician (1821-1894), proved the so-called Chebyshev's inequality that in any data sample or probability distribution, nearly all the values are close to the mean value, which can be described mathematically as follows:

Theorem. *Let Y be a random variable with expected value μ and finite variance σ^2. Then for any positive real number k:*

$$\Pr(|Y - \mu| \geq k\sigma) \leq 1/k^2 \ . \tag{B.1}$$

Note that only the cases $k > 1$ provide useful information.

Reference

[AMH01] A. Ahmed, F. Mokhtarian, and A. Hilton, “Parametric Motion Blending through Wavelet Analysis,” *Proceedings of Eurographics*, 2001.

[AP06] A. Atsalakis and N. Papamarkos, “Color Reduction and Estimation of the Number of Dominant Colors by Using a Self-Growing and Self-Organized Neural Gas,” *Engineering Applications of Artificial Intelligence 19*, pp. 769−786, 2006.

[AP7] Adobe Photoshop 7.

[BFH*98] J.M. Buhmann, D.W. Fellner, M.Held, J.Ketterer, and J. Puzicha, “Dithered Color Quantization,” *Proceedings of Eurographics*, vol. 17, no. 3., pp. 219–231, 1998.

[BJ01] Y. Boykov and M. P. Jolly, “Interactive Graph Cuts for Optimal Boundary and Region Segmentation of Objects in N-D Images,” *IEEE International Conference on Computer Vision*, 2001.

[BK04] Y. Boykov and V. Kolmogorov, “An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no.9, pp. 1124-1137, 2004.

[BKT*05] C. Breazeal, C. Kidd, A. L. Thomaz, G. Hoffman, and M. Berlin, “Effects of Nonverbal Communication on Efficiency and Robustness in Human-Robot Teamwork,” *IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2005.

[Bor] A. G. Bors, Introduction of Radial Basis Function (RBF) Networks, *Online Symposium for Electronics Engineers*, vol.1 of *DSP Algorithms: Multimedia*.

[BVD00] A. Berman, P. Vlahos, and A. Dadourian, “Comprehensive Method for Removing From an Image the Background Surrounding a Selected Object,” U.S. Patent 6,134,345, 2000.

[BW04] T. Brox and J. Weickert, “Level Set Based Image Segmentation with Multiple Regions,” *Proceedings of 26th DAGM*, pp.415-423, 2004.

[CCSS01] Y. Y. Chuang, B. Curless, D. Salesin, and R. Szeliski, “A Bayesian Approach to Digital Matting,” *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001.

[CDF92] E. Cohen, I. Daubechies, and J. C. Feauveau, “Biorthogonal Bases of Compactly Supported Wavelets,” *Communication on Pure and Applied Mathematics 45*, pp. 485–560, 1992.

[CKS97] V. Caselles, R. Kimmel, and G. Sapiro, “Geodesic Active Contours,” *International Journal of Computer Vision*, 22(1):61-79, 1997.

[CM02] D. Comaniciu and P. Meer, “Mean Shift: A Robust Approach Toward Feature Space Analysis,” *IEEE Transactions on Pattern Analysis and Machine Intelligence 24,* 2002.

[Coh91] L. D. Cohen, “On Active Contour Models and Balloons,” *Computer Vision, Graphics, and Image Processing*, vol. 53, no. 2, pp. 211-218, 1991.

[CQ92] C. K. Chui and E. Quak, An Introduction to Wavelets, *Wavelet Analysis and its Application*, Academic Press, vol. 1, 1992.

[CRD06] D. Cremers, M. Rousson, and R. Deriche, "Review of Statistical Approaches to Level Set Segmentation: Integrating Color, Texture, Motion and Shape," *International Journal of Computer Vision*, 2006.

[Cut89] M. R. Cutkosky, "On Grasp Choice, Grasp Models, and the Design of Hands for Manufacturing Tasks," *IEEE Transactions on Robotics and Automation*, 5(3):269–279, 1989.

[Db] http://robix.com/drawbot.htm

[Dek94] A. H. Dekker, "Kohonen Neural Networks for Optimal Color Quantization," *Network: Computat. Neural Syst.*, vol. 5, pp. 351–367, 1994.

[DM00] H. Delingette and J. Montagnat, "New Algorithms for Controlling Active Contours Shape and Topology," *European Conference on Computer Vision*, 2000.

[Dtr] http://techhouse.brown.edu/~neel/drawing_telerobot/

[DVS01] A. D'Souza, S. Vijayakumar, and S. Schaal, "Learning Inverse Kinematics," *IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2001.

[FB88] D. Forsey and R. Bartels, "Hierarchical B-Spline Refinement," *Proceedings of Computer Graphics and Interactive Techniques*, 1988.

[FO03] J. A. Fails and D. R. Olsen, "A Design Tool for Camera-Based Interaction," *Conference on Human Factors in Computing Systems*, 2003.

[FSA07] I. Fondon, C. Serrano, and B. Acha, "Segmentation of Skin Cancer Images Based on Multistep Region Growing," *IAPR Conference on Machine Vision Applications*, pp.339-342, 2007.

[FWF02] X. Feng, C. K. I. Williams, and S. N. Felderhof, "Combining Belief Networks and Neural Network for Scene Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24, 2002.

[FZK99] M. Ferch, J. Zhang, and A. Knoll, "Robot Skill Transfer Based on B-Spline Fuzzy Controllers For Force-Control Tasks" *IEEE International Conference on Robotics and Automation,* 1999.

[Got95] S. J. Gotler, "Hierarchical and Variational Geometric Modeling with Wavelets," *ACM Symposium on Interactive 3D Graphics*, pp. 35-42, 1995.

[GP90] M. Gervautz and W. Purgathofer, "A Simple Method for Color Quantization: Octree Quantization," *Graphics Gems*, A. S. Glassner, Ed. New York: Academic, pp. 287–293, 1990.

[GPS89] D. Greig, B. Porteous, and A. Seheult, "Exact MAP Estimation for Binary Images," *J. Roy. Stat. Soc. B.*, 51:271–279, 1989.

[HBS99] J. Hug, C. Brechbuhler, and G. Szekely, "Tamed Snake: A Particle System for Robust Semi-Automatic Segmentation," *Proceedings of Intl. Conf. on Medical Image Computing and Computer-Assisted Intervention*, 106-115, 1999.

[HBZ06] M. Hueser, T. Baier, and J. Zhang, "Learning Demonstrated Grasping Skills by Stereoscopic Tracking of Human Hand Recognition," *IEEE International Conference on Robotics and Automation*, 2006.

[HE04] J. Hays and I. Essa, "Image and Video Based Painterly Animation," *International Symposium on Non-Photorealistic Animation and Rendering*, 2004.

[Hec82] P. Heckbert, "Color Image Quantization for Frame Buffer Display," *Comput. Graph.*, vol. 16, pp. 297–307, 1982.

[Her98] A. Hertzmann, "Painterly Rendering with Curved Brush Strokes of Multiple Sizes," *Proceedings of ACM SIGGRAPH* , 1998.

[HKKH03] K. Harada, S. Kajita, K. Kaneko, and H. Hirukawa, "ZMP Analysis for Arm/Leg Coordination," *IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2003.

[HKY05] Y. Hirano, K. Kitahama, and S. Yoshizawa, "Image-Based Object Recognition and Dexterous Hand/Arm Motion Planning Using RRTs for Grasping in Cluttered Scene," *IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2005.

[HN98] W. Huyer and A. Neumaier, Multilevel coordinate search, Kluwer Academic Publishers, 1998. Available: http://www.mat.univie.ac.at/~neum/ software/ls/

[HZS06] L. Hu, C. Zhou, and Z. Sun, "Biped Gait Optimization Using Spline Function Based Probability Model," *IEEE International Conference on Robotics and Automation,* 2006.

[Kaj02] S. Kajita, "A Realtime Pattern Generator for Biped Walking," *IEEE International Conference on Robotics and Automation,* 2002.

[KI97] S. B. Kang and K. Ikeuchi. Toward automatic robot instruction from perception — mapping human grasps to manipulator grasps. *IEEE Transactions on Robotics and Automation*, 13(1):81–95, 1997.

[KL06] J. Ketchel and P. Larochelle, "Collision Detection of Cylindrical Rigid Bodies for Motion Planning," *IEEE International Conference on Robotics and Automation,* 2006.

[KORI06] S. Kudoh, K. Ogawara, M. Ruchanurucks, and K. Ikeuchi, "Painting Robot with Multi-Fingered Hands and Stereo Vision," *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 2006**.**

[KT03] W. Korb and I. Troch, "Data Reduction for Manipulator Path Planning," *Robotica*, vol. 21, pp. 605-614, 2003.

[KWT88] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, 1:321-331, 1988.

[KYO*07] T. Kang, J. Yu, J. Oh, Y. Seol, K. Choi and M. Kim, "Object Based Contour Detection by Using Graph-Cut on Stereo Image," *IAPR Conference on Machine Vision Applications*, pp.319-322, 2007.

[LL90] Y. W. Lim and S. U. Lee, "On the Color Image Segmentation Algorithm Based on the Thresholding and the Fuzzy C-Means Techniques," *Pattern Recognit.*, vol. 23, no. 9, pp. 935–952, 1990.

[LL95] T.-Y. Li and J.-C. Latombe, "Online Manipulation Planning for Two

Robot Arms in a Dynamic Environment," *IEEE International Conference on Robotics and Automation*, 1995.

[LLW06] A. Levin, D. Lischinski, and Y. Weiss, "A Closed Form Solution to Natural Image Matting," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006.

[LS06] D. Lee and M. W. Spong, "Passive Bilateral Teleoperation with Constant Time Delays" *IEEE International Conference on Robotics and Automation*, 2006.

[LS99] J. Lee and S. Y. Shin, "A Hierarchical Approach to Interactive Motion Editing for Human-Like Figures," *Proceedings of ACM SIGGRAPH*, pp. 39-48, 1999.

[LSS05] Y. Li, J. Sun, and H. Y. Shum, "Video Object Cut and Paste," *Proceedings of ACM SIGGRAPH*, 2005.

[LSTS04] Y. Li, J. Sun, C. K. Tang, and H. Y. Shum, "Lazy Snapping," *Proceedings of ACM SIGGRAPH*, 2004.

[MB95] E. Mortensen and W. Barrett, "Intelligent Scissors for Image Composition," *Proceedings of ACM SIGGRAPH*, 1995.

[MFM04] D. R. Martin, C. C. Fowlkes, and J. Malik, "Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26, 2004.

[MGS*05] M. Marrn, J. C. Garca, M. A. Sotelo, D. Fernndez, and D. Pizarro, "XPFCP: An Extended Particle Filter for Tracking Multiple and Dynamic Objects in Complex Environment," *IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2005.

[Mis93] Y. Mishima, "Soft Edge Chroma-Key Generation Based upon Hexoctahedral Color Space," U.S. Patent 5,355,174, 1993.

[MMP05] M. McGuire, W. Matusik, H. Pfister, J. F. Hughes, and F. Durand, "Defocus video matting," *Proceedings of ACM SIGGRAPH*, 2005.

[MO06] M. Mahvash and A. M. Okumura, "Friction Compensation for a Force-Feedback Telerobotic System", *IEEE International Conference on Robotics and Automation*, 2006.

[Mon97] P. Monaghan, "An Art Professor Uses Artificial Intelligence to Create a Computer That Draws and Paints," *The Chronicle of Higher Education*, 1997.

[MSV95] R. Malladi, J.A. Sethian, and B.C. Vemuri, "Shape Modeling with Front Propagation: A Level Set Approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):158-174, 1995.

[MT95] T. McInerney and D. Terzopoulos, "Topologically Adaptable Snakes," *IEEE International Conference on Computer Vision*, 1995.

[Nap56] J. Napier, "The Prehensile Movements of the Human Hand," *Journal of Bone and Joint Surgery*, 38B(4):902–913, 1956.

[NNK*07] S. Nakaoka, A. Nakazawa, F. Kanehiro, K. Kaneko, M. Morisawa, H. Hirukawa, K. Ikeuchi, "Learning from Observation Paradigm: Leg Task Models for Enabling a Biped Humanoid Robot to Imitate Human Dances," *The International Journal of Robotics Research*, vol. 26, no.

8, pp. 829-844, 2007.

[PAS02] N. Papamarkos, A. E. Atsalakis, and C. P. Strouthopoulos, “Adaptive color reduction,” *IEEE Transactions on System, Man, and Cybernatics*, vol. 32, no. 1, pp. 44-56, 2002.

[PHRA02] N. S. Pollard, J. K. Hodgins, M. J. Riley, and C. G. Atkeson, “Adapting Human Motion for the Control of a Humanoid Robot,” *IEEE International Conference on Robotics and Automation,* 2002.

[RGBC96] C. Rose, B. Guenter, B. Bodenheimer, and M. F. Cohen, “Efficient Generation of Motion Transitions Using Spacetime Constraints,” *Proceedings of ACM SIGGRAPH*, 1996.

[RKB04] C. Rother, V. Kolmogorov, and A. Blake, “GrabCut–Interactive Foreground Extraction Using Iterated Graph Cut,” *Proceedings of ACM SIGGRAPH*, 2004.

[RKO*07] M. Ruchanurucks, S. Kudoh, K. Ogawara, T. Shiratori, and K. Ikeuchi, “Humanoid Robot Painter: Visual Perception and High-Level Planning,” *IEEE International Conference on Robotics and Automation*, 2007.

[RKOSI07] M. Ruchanurucks, S. Kudoh, K. Ogawara, T. Shiratori, and K. Ikeuchi, “Robot Painter: From Object to Trajectory,” *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007.

[RNKI05] M. Ruchanurucks, S. Nakaoka, S. Kudoh, and K. Ikeuchi, “Generation of Humanoid Robot Motions with Physical Constraints Using Hierarchical B-Spline,” *IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2005.

[RNKI06] M. Ruchanurucks, S. Nakaoka, S. Kudoh, and K. Ikeuchi, “Humanoid Robot Motion Generation with Sequential Physical Constraints,” *IEEE International Conference on Robotics and Automation,* 2006.

[ROI] (submitted) M. Ruchanurucks, K. Ogawara, and K. Ikeuchi, “Region Growing Graph Cut,” *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

[ROI06] M. Ruchanurucks, K. Ogawara, and K. Ikeuchi, “Neural Network Based Foreground Segmentation with an Application to Multi-Sensor 3D Modeling,” *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 2006**.**

[RT00] M. A. Ruzon and C. Tomasi, “Alpha Estimation in Natural Images,” *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 18–25, 2000.

[Ruc07] M. Ruchanurucks, “Clustering Based Color Reduction: Improvement and Tips,” *Meeting on Image Recognition and Understanding*, 2007.

[RUWA03] M. Riley, A. Ude, K. Wade, and C. G. Atkeson., “Enabling Real-Time Full Body Imitation: A Natural Way of Transferring Human Movements to Humanoids,” *IEEE International Conference on Robotics and Automation,* 2003.

[SbFAZ00] C. Samson, L. Blanc-Feraud, G. Aubert, and J. Zerubia, “A Level Set Model for Image classification,” *International Journal of Computer Vision*, 40(3):187–197, 2000.

[SCN*98] A. Srikaew, M. E. Cambron, S. Northrup, R. A. Peters II, M. Wilkes, and K. Kawamura, “Humanoid Drawing Robot,” *IASTED International Conference on Robotics and Manufacturing*, 1998.

[SK87] S. A. Shafer and T. Kanade, “Color vision,” *Encyclopedia of Artificial Intelligence*, S. C. Shapiro and D. Eckroth, Eds. New York:Wiley, pp. 124–131, 1987.

[SKH05] F. Seto, K. Kosuge and Y. Hirata, “Self-Collision Avoidance Motion Control for Human Robot Cooperation System Using RoBE,” *IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2005.

[SM85] K.G. Shin and N.D. McKay, “Minimum-Time Control of Robotic Manipulators with Geometric Path Constraints,” *IEEE Transactions on Automatic Control*, vol. 30, no.6, pp. 531-541, June 1985.

[SPH03] A. Safonova, N. S. Pollard, and J. K. Hodgins, “Optimizing Human Motion for the Control of a Humanoid Robot,” *2nd International Symposium on Adaptive Motion of Animals and Machines*, 2003.

[SY00] M. Shiraishi and Y. Yamaguchi, “An Algorithm for Automatic Painterly Rendering Based on Local Source Image Approximation,” *International Symposium on Non-Photorealistic Animation and Rendering*, 2000.

[SY89] J. -J. E. Slotine and H. S. Yang, “Improving the Efficiency of Time-Optimal Path-Following Algorithms,” *IEEE Transactions on Robotics and Automation*, vol.5, no.1, pp. 118-124, February 1989.

[TNMY05] M. Takahashi, T. Narukawa, K. Miyakawa and K. Yoshida, “Combined Control of CPG and Torso Attitude Control for Biped Locomotion,” *IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2005.

[TW05] D. R. Thompson and D. Wettergreen, “Multi-object Detection in Natural Scenes with Multiple-view EM Clustering,” *IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2005.

[UAR00] A. Ude, C. G. Atkeson, and M. Riley, “Planning of Joint Trajectories for Humanoid Robots Using B-Spline Wavelets,” *IEEE International Conference on Robotics and Automation*, 2000.

[UAR04] A. Ude, C. G. Atkeson, and M. Riley, “Programming Full Body Movements for Humanoid Robot by Observation,” *Robotics and Autonomous Systems*, vol. 47, pp. 93-108, 2004.

[VC02] L. Vese and T. Chan, “A Multiphase Level Set Framework for Image Segmentation Using the Mumford and Shah Model,” *International Journal of Computer Vision*, 50(3):271–293, 2002.

[Ver95] O. Verevka, “The Local K-Means Algorithm for Color Image Quantization,” M.Sc. dissertation, Univ. Alberta, Canada, 1995.

[VS91] L. Vincent and P. Soille, “Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13, 1991.

[VS91] L. Vincent and P. Soille, “Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations,” *IEEE Transactions on*

Pattern Analysis and Machine Intelligence 13, 1991.

[WBC*05] J. Wang, P. Bhat, R. A. Colburn, M. Agrawala, and M. F. Cohen, "Interactive Video Cutout," *Proceedings of ACM SIGGRAPH*, 2005.

[WC05] J. Wang and M. F. Cohen, "An Iterative Method Approach for Unified Image Segmentation and Matting," *IEEE International Conference on Computer Vision*, 2005.

[WC07] J. Wang and M. F. Cohen, "Optimized Color Sampling for Robust Matting," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

[WJ05] J. Win and N. Jojic, "LOCUS: Learning Object Classes with Unsupervised Segmentation," *IEEE International Conference on Computer Vision,* 2005.

[WPW90] S. J. Wan, P. Prusinkiewicz, and S. K. M. Wong, "Variance Based Color Image Quantization for Frame Buffer Display," *Color Res. Applicat.*, vol. 15, no. 1, pp. 52–58, 1990.

[XBA03] N. Xu, R. Bansal, and N. Ahuja, "Object Segmentation Using Graph Cuts Based Active Contours," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003.

[YES*06] E. Yoshida, C. Esteves, T. Sakaguchi, J.-P. Laumond, and K. Yokoi, "Smooth Collision Avoidance: Practical Issues in Dynamic Humanoid Motion," *IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2006.

[YK04] K. J. Yoon and I. S. Kweon, "Human Perception Based Color Image Quantization," *IEEE International Conference on Pattern Recognition 17*, 2004.

[ZN02] L. Zlajpah and B. Nemec, "Kinematic Control Algorithms for Online Obstacle Avoidance for Redundant Manipulators," *IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2002.

List of Publications

Journal Paper

M. Ruchanurucks, K. Ogawara, and K. Ikeuchi, "Comprehensive Iterative Foreground Segmentation Using Region Growing Graph Cut," *IEEE Transactions on Pattern Analysis and Machine Intelligence.* (submitted)

M. Ruchanurucks, S. Nakaoka, S. Kudoh, and K. Ikeuchi, "Space-Time Physical Constraints for Offline and Online Motion Generation," *IEEE Transactions on Robotics.* (submitted)

IEEE International Conference

M. Ruchanurucks, S. Kudoh, K. Ogawara, T. Shiratori, and K. Ikeuchi, "Robot Painter: From Object to Trajectory," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007.

M. Ruchanurucks, S. Kudoh, K. Ogawara, T. Shiratori, and K. Ikeuchi, "Humanoid Robot Painter: Visual Perception and High-Level Planning," *IEEE International Conference on Robotics and Automation*, 2007.

M. Ruchanurucks, K. Ogawara, and K. Ikeuchi, "Neural Network Based Foreground Segmentation with an Application to Multi-Sensor 3D Modeling," *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 2006**.**

S. Kudoh, K. Ogawara, M. Ruchanurucks, and K. Ikeuchi, "Painting Robot with Multi-Fingered Hands and Stereo Vision," *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 2006**.**

M. Ruchanurucks, S. Nakaoka, S. Kudoh, and K. Ikeuchi, "Humanoid Robot Motion Generation with Sequential Physical Constraints," *IEEE International Conference on Robotics and Automation,* 2006.

M. Ruchanurucks, S. Nakaoka, S. Kudoh, and K. Ikeuchi, "Generation of Humanoid Robot Motions with Physical Constraints Using Hierarchical B-Spline," *IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2005.

Domestic Conference

M. Ruchanurucks, "Clustering Based Color Reduction: Improvement and Tips," *Meeting on Image Recognition and Understanding*, 2007.