INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Augmented Reality on Cloth with Realistic Illumination

By

Derek Bradley

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of
Master of Computer Science

Ottawa-Carleton Institute for Computer Science
School of Computer Science
Carleton University
Ottawa, Ontario

April 2005

Abstract

Augmented reality (AR) is the concept of adding computer generated objects to the real world using live video or a head-mounted display system in real-time. One of the main problems in AR systems is the correct alignment of virtual objects and real objects in order to create a believable mixed-reality environment. Often, this problem is solved by visually locating a rigid plane in the scene and then aligning the virtual objects with this plane. This thesis addresses the problem of performing real-time flexible augmentations aligned with *non-rigid* objects such as cloth. The techniques presented involve tracking a non-rigid object using computer vision to acquire a mesh representation of its surface. In addition, novel methods to establish common illumination between the real and virtual environments are presented. Our experiments confirm that non-rigid object tracking combined with the acquisition of realistic illumination provides an interactive system for flexible augmentations on non-rigid objects.

Acknowledgements

First and foremost I would like to thank my two thesis supervisors, Dr. Gerhard Roth and Dr. Prosenjit Bose for their excellent guidance throughout this research. A dual supervisor approach is not always the best scenario for everyone involved, but in this case my supervisors and I worked very well together as we shared a common focus and mutual goals at all times. Because of this fact, only the advantages of having two supervisors were evident, with few to no disadvantages, and I would like to acknowledge that.

I also wish to thank Dr. Mark Fiala and Dr. Chang Shu for the many discussions, the advice and the feedback that I received during this research.

I would like to thank my thesis defence committee for taking the time to review my work and attend my oral defence.

Finally, I wish to thank my friends and family for supporting me in my work at all times.

This research was funded in part by the Natural Sciences and Engineering Research Council of Canada.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The use of computer graphics in real world applications and everyday life is becoming more and more apparent. Many industries such as entertainment, medicine, tourism, military, and education (just to name a few) integrate virtual objects into real environments to help demonstrate situations that may not be possible in reality. These computer generated entities are often combined with the real world in a seamless way, so that we as humans find it difficult to perceive what is real and what is virtual. In the past there were two distinct and very specific environments, *reality* and *virtual reality*. Reality is obviously the physical world that we live in, and virtual reality is a world where every part of the environment is computer generated. In today's world this distinction is no longer valid, as there now exists a continuous range of environments between the real and the virtual [48]. This continuum, as illustrated in Fig. 1.1, introduces the concept of *mixed reality* to denote environments that are comprised of both real *and* virtual parts.

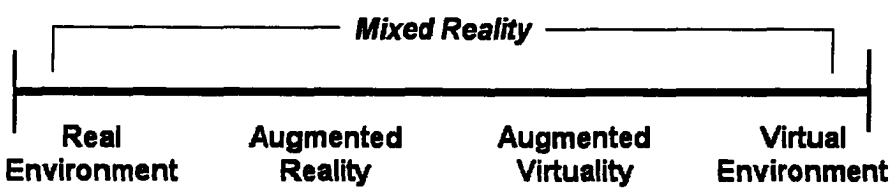


Figure 1.1: Continuum from real to virtual environments.

The two main fields of mixed reality are augmented reality (AR) and augmented virtuality (AV). In augmented reality, the main environment is the real world. It is said to be "augmented" because individual virtual objects are placed into the scene to provide an enhanced environment. Similarly, the main environment in augmented virtuality is computer generated, and individual real objects are placed into the scene. This thesis research is in the field of augmented reality.

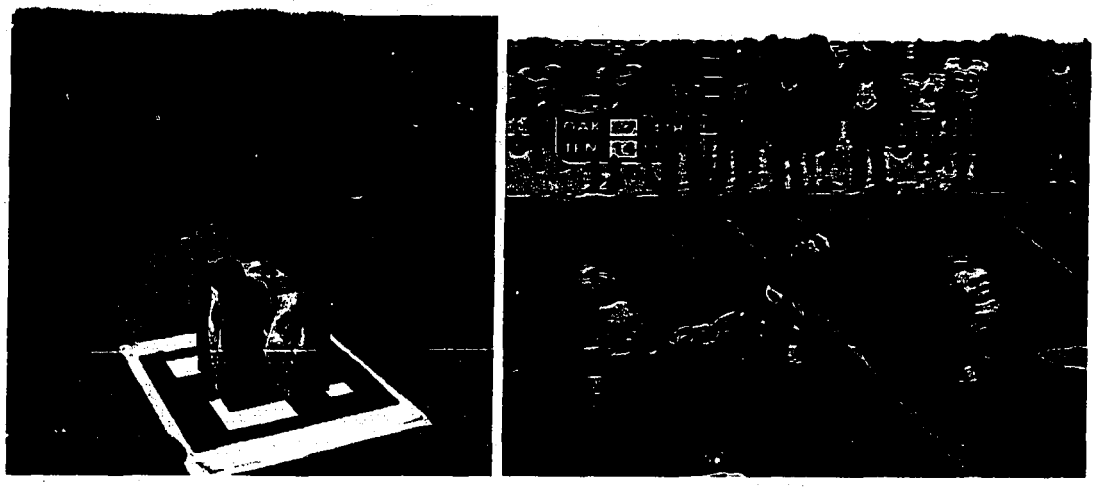
The main goal of augmented reality is to interactively insert virtual objects into a real scene by superimposing them onto a real-time video stream or head-mounted display (HMD) system. These virtual objects can be simple, such as a 2D textual label for a real object, or complex, like the 3D model of a building or biological structure. In all cases, it is essential to correctly align the virtual objects with the real world in order to establish seamless integration and the perception that the virtual objects actually exist. This so-called *registration* problem is solved by aligning the virtual objects with a rigid planar surface in the real world. The surface is typically located in the video stream using visual targets, and then the pose of the camera is established. Virtual objects are then rendered in the scene using the camera pose as a virtual viewpoint. Example AR augmentations are illustrated in Fig. 1.2.

Since augmentations are always aligned with a rigid plane in the real world, interaction with the virtual objects is somewhat restricted. This thesis concerns itself with performing flexible augmentations on non-rigid objects, such as a piece of cloth. In addition, novel techniques to increase the realism of augmentations by incorporating the real world illumination environment are explored.

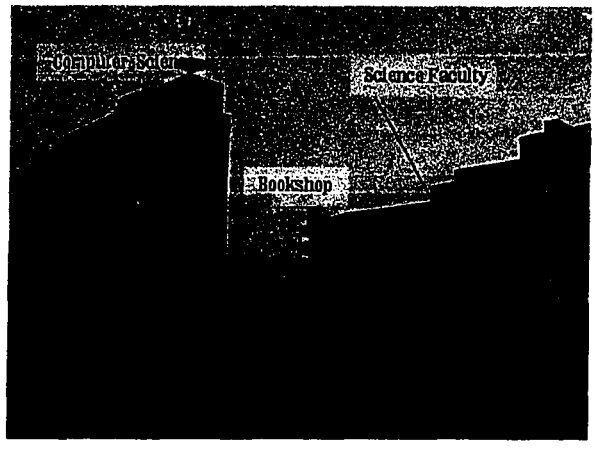
The next section describes motivation for this work. Section 1.2 outlines the scope of this thesis, and Section 1.3 identifies the contributions from this thesis to the AR research field. Finally, Section 1.4 provides an overview of the thesis chapters.

1.1 Motivation

The ability to insert virtual objects into a real scene is beneficial for many applications. Common uses of augmented reality include: overlaying critical data onto visual HMDs during military operations; augmented assistance during medical procedures;



(a) (b)



(c)

Figure 1.2: Example augmentations in the field of AR. a) A 3D cube is augmented on a 2D planar pattern; b) The yellow first-down line is augmented on a football field [3]; c) Textual annotations are augmented in a mobile AR system [72].

annotated textual cues in the tourism industry; interactive 3D model manipulation for education and research; advertising logos on televised sporting events; historical reconstruction of demolished sites; visualizing construction results before commencing; interactive games; and many more. However, these applications can make use of the rigid planar augmentation methods that exist today. We are interested in the problem of non-rigid augmentations that can be performed on cloth and other flexible objects. The motivation behind this research lies mainly in the entertainment, museum and fashion industries. The ability to perform realistic augmentations on cloth provides a system for interactively replacing the texture of clothing. Applications include a virtual fashion show, interactive clothing design, or entertainment for museums. The general novelty of this research is also a motivating factor.

1.2 Scope

Our hypothesis is that non-rigid objects can be visually tracked with enough precision to perform realistic augmentations. The scope of this research is the real-time tracking of a cloth surface in video. In addition, this thesis includes the acquisition of the real illumination environment for the purpose of 2D augmentations.

1.3 Contributions

To our knowledge, this thesis is the first research in augmented reality to include non-rigid object tracking. The contributions of this thesis are:

- Three different visual tracking methods for acquiring a sparse representation of a non-rigid object in a real-time video stream,
- Two different visual methods to establish common illumination in an augmented reality cloth environment for more realistic augmentations,
- System architecture and implementation of the above methods.

1.4 Thesis Overview

This thesis is organized as follows. Chapter 2 presents the related work in augmented reality, non-rigid object tracking and establishing common illumination in mixed reality scenes. Chapter 3 describes our three methods for tracking non-rigid objects in a real-time video stream. Chapter 4 explains how flexible augmentations are performed on the non-rigid object, including the acquisition and application of the real world illumination environment. The results of this thesis are outlined in Chapter 5, and finally conclusions are presented in Chapter 6.

Chapter 2

Related Work

This thesis falls into the growing research field of augmented reality. This chapter starts with an overview of AR systems and the augmentation environment, briefly mentioning previous work selected from the field. To the best of our knowledge, this is the first piece of research in augmenting non-rigid objects. Thus, direct related work is not available. However, this thesis includes research on two separate subtopics: vision-based non-rigid object tracking and establishing common illumination in mixed-reality scenes. Related work on these topics is also presented in this chapter.

2.1 Augmented Reality

Augmented reality is the concept of adding virtual objects to the real world by superimposing them onto a video stream or HMD in real-time. One of the main problems in AR systems is the registration problem, that is, aligning the virtual objects with the real world in a believable way. This alignment is accomplished by visually tracking real objects in the scene and computing the pose of the camera, which consists of its 3D position and orientation. Once the pose of the camera is known (relative to the tracked objects in the scene), virtual objects are rendered on top of the video at correct positions by using the camera pose as a virtual viewpoint. This creates two different categories of scene objects, real ones and virtual ones. The real objects exist in the video image and the virtual objects are created in a 3D rendering

environment. The rendering environment is set up much like a typical 3D graphics application, with the addition of the video frame as the background for the scene. The rendering environment consists of a viewpoint and view frustum. The viewpoint faces a 2D rendering plane that is at a distance equal to the camera focal length (converted to three-dimensional units) away from the viewpoint. This is where the video image is rendered as a texture to form the background. Then virtual objects are rendered in front of the background so that they appear to blend into the scene. Fig. 2.1 illustrates the rendering environment. Optionally, virtual objects can be rendered behind the video image plane, but then the depth buffer (or Z-buffer) must be manually controlled so that the objects behind the background are still drawn on top of it. Unlike other graphical applications, here the viewpoint remains fixed (along

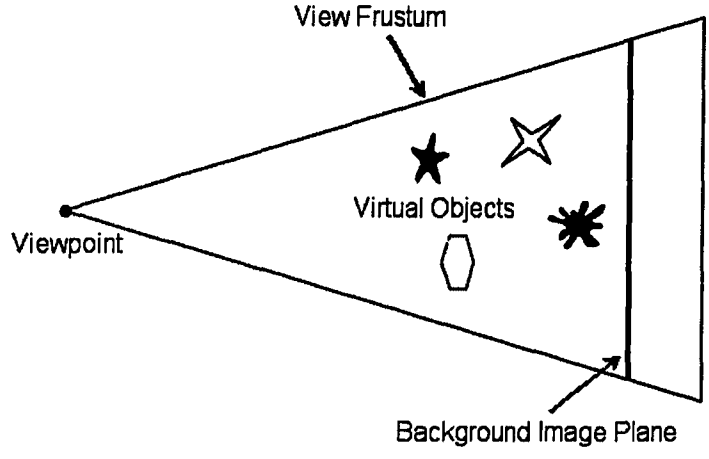


Figure 2.1: Top view of a typical AR rendering environment.

with the background plane). The scene is controlled by moving the camera and the real objects, which updates the background image and also the pose (and existence) of the virtual objects.

The rendering environment for augmented reality is a standard concept that does not receive much attention in the research community. Instead, researchers focus on the registration problem, different object tracking methods, interaction with the virtual objects, occlusion problems, hardware enhancements, applications of AR and

social acceptance studies. Azuma et al. [3] present a general overview of research in the AR field. To provide more background of AR research for this thesis, the most relevant and related research literature is reviewed below.

One of the most common ways to solve the registration problem is by tracking specific planar markers that are placed in the scene. A popular application of marker tracking for augmented reality is the "MagicBook", presented by Billinghurst et al. [6]. The MagicBook is a physical book with square markers on the pages, in addition to regular pictures and text. Users look through an augmented reality display and see virtual models appearing out of the pages. Since the models are aligned with the markers, users can see the AR scene from any perspective by moving themselves or the book. The virtual models are changed by turning pages in the physical book. In addition to AR, the authors include a virtual reality aspect by allowing users to "fly into" the virtual scene and become an avatar[1]. In this mode, the environment is completely virtual and the user controls the avatar from a first-person perspective. Another application of marker tracking for AR is the augmented chemistry workbench of Fjeld and Voegtli [24]. The authors create a tangible user interface (TUI) from a set of square markers to control an interactive mirror-based chemistry application. Virtual elements are chosen from a menu and composed into 3D molecular models. The elements and models are manipulated by hand using a tangible cube covered in markers. Malik et al. [44] present a fast and accurate vision-based corner tracker for marker-based AR. The goal of their research is to overcome the computational costs and lack of robustness in many marker tracking methods. Their tracking system predicts corner positions in video images and uses local search windows to find the corners with subpixel accuracy. Results of their system show robustness with respect to occlusion, scale, orientation and lighting.

A second approach to solve the registration problem for AR is to establish the alignment of virtual objects in a *markerless* environment. In this way, the scene is not modified by the addition of markers, and instead the tracking process determines the pose of the camera by locating real objects in the scene. Simon et al. [59] describe a markerless tracking system for environments that contain one or more planes. Their

[1]An image or object representing a user in a multi-user virtual environment.

system is based on a structure-and-motion estimation algorithm called automatic move-matching, which simultaneously estimates the camera motion and the planar structures in the scene. Their algorithm can "hand off" tracking from one plane to the next, providing a practical and reliable vision-based tracker. Comport et al. [15] develop a robust and efficient 3D model-based tracking algorithm. Non-linear pose computation is accomplished by means of a virtual visual servoing approach. Real-time tracking of different features including lines, circles, cylinders and spheres is presented using a local moving edges tracker. Virtual objects are placed in the scene relative to the rigid tracked objects. Their method is robust to occlusion, changes in illumination and mis-tracking.

Another area of AR research involves methods to interact with the virtual augmentations. Although almost all AR systems contain some form of user interaction, some research focuses entirely on techniques to naturally interact with the augmentations. Dorfmuller-Ulhaas and Schmalstieg [21] present a method to track a human finger using a marked glove, a stereoscopic tracking system and a kinematic 3D model of a finger. The system allows gestural interaction to grab, translate, rotate and release virtual objects in an AR environment. Their method is cheap, fast, accurate and robust against occlusions. The authors demonstrate their technique with an augmented reality chess game, allowing a user to interact with virtual chess pieces. Malik et al. [43] present an interaction method based on a planar pattern tracking system that is robust under partial occlusions. Their system includes a method to detect a hand over top of the pattern and use the hand position for interaction with the augmentation. The hand is also rendered on top of the virtual objects to increase realism. Another occlusion-based technique for interacting with augmentations is presented by Lee et al. [39]. The authors develop an intuitive method to allow one and two dimensional interaction with an AR user interface. Actions such as pressing buttons, changing slider values and making menu selections are performed by visually occluding physical markers.

The final area of AR research that is discussed here is the field of AR gaming. Interactive video games have always been very popular applications of computer graphics. With the current state of augmented reality technology, AR games are now

being proposed, expanding the gaming community into the real world. Jebara et al. [38] present a wearable augmented reality system to enhance the game of billiards. A vision-based algorithm uses probabilistic color models and symmetry operations to locate a billiard table, pockets and balls. Virtual assistance is then provided to aid a player in planning and aiming. Each possible shot is determined and ranked in order of usefulness, and trajectories are rendered onto a player's HMD in real-time. Thomas et al. [63] develop a mobile indoor/outdoor augmented reality version of an existing first person video game. Their system, called "ARQuake", includes a six degree-of-freedom tracking system based on GPS, a digital compass, and visual marker-based tracking. Woodward et al. [71] present an augmented table tennis game. Their system contains natural interaction with real rackets and a virtual ball and table. The pose of the racket is computed by visually detecting markers placed on the racket surface. The game is played over a network, where the video of each player is streamed in real-time to the opponent player and displayed at the other end of the table. Multicast implementation allows an audience on the network to view the game in a virtual environment.

2.2 Non-Rigid Object Tracking

In order to render augmentations on a flexible object, a virtual representation of the object must be acquired from the video image. Non-rigid object tracking is a difficult problem due to the flexibility of the object and its ability to self-occlude. Computer vision researchers have developed non-rigid object tracking algorithms for video sequences using different techniques. Combinations of single versus multiple viewpoints, cluttered versus non-cluttered backgrounds, and 2D contour retrieval versus 3D model recovery have all been attempted [64, 36, 57, 35, 46, 50, 9, 65, 62, 11, 12, 23, 37, 10, 56, 14, 52, 29, 30, 31, 32]. Previous work is divided into three categories, ranging from the least related to the most related to this thesis.

The first category is general non-rigid object tracking in an offline mode for the purpose of scene capturing, with no attempt to track cloth. Tiwari and Bhattacharya [64] recover 3D motion of a non-rigid object from a sequence of stereo images.

Assuming feature correspondence over multiple frames, they reduce the problem of S-dimensional motion recovery to the solution of a set of homogeneous polynomial equations. Huttenlocher et al. [36] describe a model-based method for tracking non-rigid objects in a complex scene. In this method, 2D models are extracted from the video sequence to decompose the 3D solid object into two components: motion and shape change. Motion and shape change are treated differently under the assumption that the shape of an object will change slowly from one frame to the next, however there is no restriction on the amount of motion between frames. Similarly, Qiuqi and Haiying [57] develop a system that also decomposes a non-rigid object into 2D shape change and motion components, under the same assumption. Their method is based on Hausdorff distances to extract templates of the object from one frame to the next. Experiments are conducted on human contour tracking. Heisele et al. [35] develop an algorithm for tracking non-rigid objects in a pre-recorded video sequence based on color. Object parts are determined in an initial clustering step, and then the clusters are iteratively adapted in each frame from the previous one, using a parallel k-means clustering algorithm. Only the cluster centroids are tracked, simplifying the correspondence problem and providing more robust tracking. Marcenaro et al. [46] present an automatic algorithm to build a statistical model of the shape of a non-rigid object using a multiple camera environment. The goal of their method is to learn the flexible model from a training sequence using a Principal Component Analysis (PCA) algorithm. Snakes and dynamic contours are then used to describe the shape of the object. Oberti and Regazzoni [50] demonstrate a low computational algorithm for tracking non-rigid objects in cluttered scenes. In this method, the 2D shape of the object is modeled using corners. A learning algorithm is applied to a short video sequence of the object without background clutter in order to extract the model, and then more objects are merged into the scene. Bregler et al. [9] develop a model-free approach that can recover non-rigid shape models from a single-view video sequence. The 3D shape in each frame is a linear combination of a set of basis shapes. Torresani et al. [65] also consider that a non-rigid object can be approximated using a linear combination of 3D basis shapes. In their work, the authors track deforming objects

in a monocular video sequence using the rank constraint to achieve optical flow estimation. The resulting flow matrix is then factored in an iterative manner to get the 3D pose of the object. The bound on the rank is exploited to handle occlusions. Tan and Ishikawa [62] recover 3D shape of non-rigid objects using a stereo approach. Static landmarks are used to calculate the orientation of the two cameras using a factorization technique, and then a set of linear equations are solved to obtain the 3D shape of non-rigid objects. Erdem et al. [11] develop a framework for non-rigid object tracking using performance evaluation measures as feedback. Their system can track the contour of a 2D object by dividing it into sub-contours, and then using the performance measures in a feedback loop to track the sub-contours of each frame. Simple objects can be tracked in real-time, however complex non-rigid objects require processing in an offline mode. Tracking results are accurate under significant occlusion and background clutter. Cen and Qi [12] propose a framework of geometric active contours to track non-rigid objects with a cluttered background. This method consists of motion detection and tracking stages, and results show the tracking of two hands undergoing an intersection movement. Feng and Zhao [23] present a technique for detecting and segmenting non-rigid objects in a video sequence using mean shift analysis to convert raw video data to corresponding 2D/3D region feature space. Objects are detected in successive frames using local motion estimation. Tracking is robust under partial occlusions and background clutter, and the result is a segmentation of the image indicating the object of interest. Jaffré and Crouzil [37] develop a method of non-rigid object localization in an image using object colors. A binary image is created and clusters are found using a mean shift procedure. Results show detection of soccer players in sport images.

The second category of previous work is more related to this thesis. This category contains work involving the capture of non-rigid cloth from a video sequence, however still in an offline method and for the purpose of scene capturing. Carceroni and Kutulakos [10] present a multiple view system that performs reconstructions based on *surfels* (surface elements [54]) to recover 3D shape, reflectance, and non-rigid motion of a dynamic 3D scene. Their algorithm called "Surfel Sampling" is used in complex scenes containing cloth, skin and shiny objects, and is able to explain pixel variations

in terms of their shape, reflectance, motion, illumination and visibility. Pritchard and Heidrich [56] use computer vision techniques to identify a printed pattern on cloth. They recover non-rigid geometry using stereo vision and then apply the Scale Invariant Feature Transform (SIFT) [41] to identify the pattern.

The last category of previous work is most related to this thesis. This is the category of real-time non-rigid object tracking. Comaniciu et al. [14] develop a real-time tracking system for non-rigid objects using mean shift. With a single camera, the most probable target position in each frame is computed using mean shift iterations. The system is capable of handling partial occlusions, clutter, and target scale variations in real-time. Examples show tracking of football players during a game and people at a subway station. However the system finds only the image region containing the object and does not attempt to re-build the 3D object structure. Okada and Hebert [52] also develop a method to track 3D non-rigid objects in real-time. This method processes 3D range data using a fast Iterative Closest Point (ICP) algorithm and a modified Robust Point Matching (RPM) algorithm to recover the motion of points on a 3D surface. They assume that the object is globally rigid and only parts of it are subjected to non-rigid deformations. Our tracking method is most similar to the flexible geometry tracking method of Guskov et al. [29, 30, 31, 32]. In their work, a grid of solid quads is placed on a flexible surface, and then the structure is captured by tracking the quads using spatial prediction and coherence in real-time. Originally, their algorithm processed black quads and was restricted by requiring user interaction to initiate the process. Further research led to improvements using colored quads and automatic feature correspondence, initialization and simplified 3D reconstruction. Quads are tracked individually to allow for occlusions. Results show efficient tracking of spheres, gloves and t-shirts.

Unlike our research, the motivation behind all of the above non-rigid object tracking methods is shape or contour reconstruction. Our goal is to track a 3D flexible piece of cloth for the purpose of real-time realistic augmentations.

2.3 Common Illumination

Establishing common illumination between the real and virtual environments is a problem that is generally divided into four categories. Shadows must be cast from real objects onto real objects, from virtual objects onto virtual objects, from virtual objects onto real objects, and from real objects onto virtual objects [33]. Shadows from real objects cast onto other real objects come for free, provided that the natural scene lighting is not modified virtually. Shadows from virtual objects cast onto other virtual objects can be dealt with using standard graphical techniques for real-time shading. However, the latter two cases are the ones that are necessary in order to achieve a high degree of realism in a mixed reality environment. A substantial amount of research has been performed by Drettakis et al. [22] and Loscos et al. [40] in the field of common illumination for AR systems and interactive virtual relighting of real scenes. Drettakis et al. [22] and Loscos et al. [40] virtually modify the lighting and geometry of a static scene by constructing a model from multiple photographs and then using a hierarchical radiosity algorithm with shaft data structures for illumination. Debevec [19] develops an algorithm to add virtual objects to a real photograph using a high dynamic range light-based model with a mirrored sphere light probe to illuminate the new objects. Virtual objects are rendered using global illumination and then composited onto the photograph using differential rendering. Stauder [61] estimates the intensity and direction of a distant point light and the ambient light in a video frame by examining the two succeeding images. Scene objects are modeled using ellipsoid-like 3D models, and the illumination parameters are estimated from the shape, motion and displacement of the models in the image sequence. Gibson and Murta [27] propose another solution to adding dynamic virtual objects to a static real scene in which the distant real world illumination is captured by an omni-directional image and then basis radiance maps are pre-computed. Virtual objects are added to the scene and illuminated using sphere mapping, which yields much faster frame-rates than global illumination algorithms. Gibson et al. [26] improve upon previous results by removing the assumption that light sources are distant. In the new system, hierarchical shaft data structures are used to subdivide light transport paths and determine

sources of light occluded by synthetic objects. Naemura et al. [49] introduce the concepts of a "Virtual Light" and a "Virtual Shadow" for mixed reality environments. In this context, a virtual light is a hand-held flashlight-like device that casts shadows of real objects onto virtual objects, virtual objects onto real objects and virtual objects onto other virtual objects. A recent work on establishing common illumination for mixed reality scenes is performed by Haller at al. [33]. In this research, the authors modify a real-time shadow volume algorithm that is typically used for computer generated scenes to work in mixed reality applications. Real objects are represented by virtual "phantom" objects, and they require markers to determine their position and orientation in the scene. Real-time results of shadow casting from all four shading categories mentioned above are demonstrated realistically.

Our goal is to perform augmentations on cloth, so common illumination can be achieved by solving only one of the four shading problems mentioned above, that of casting shadows from real objects onto the virtual augmentation. Therefore, we present two approaches to establish common illumination that are less complex than previous methods. Our methods require only the grayscale image of the input from the camera and they take into account multiple real light sources. We require no pre-processing or 3D model building, and create no shadow volumes or virtual lights. Despite their simplicity, our methods display augmentations with realistic lighting in a real-time interactive application.

Chapter 3

Real-Time Flexible Object Tracking

The first step in order to perform real-time graphical augmentations on any object is to determine where the object lies in the physical world. Our problem is to locate and track a flexible piece of cloth in a video image using a single camera and vision-based tracking techniques. As mentioned previously, cloth tracking is a difficult problem. However, a dense and accurate representation of the cloth is not necessary in order to render augmentations on the object. We demonstrate that believable 2D augmentations can be rendered onto a flexible piece of cloth using only a sparse representation of the object. This representation will be in the form of a virtual triangulated mesh. The problem is now reduced to that of locating and tracking a sufficient number of mesh points on the surface of a piece of cloth in order to interpolate a realistic augmentation.

Tracking a set of surface points on a flexible piece of cloth can be accomplished using target-based techniques. Targets (sometimes called "markers") are typically two-dimensional patterns that are placed at known locations in the scene. The targets are often self-identifiable so that when a target is found, the system will not only know the location of the target but also which target it is. In practice, an individual target can be located and identified in an input image using existing methods. Our approach is to use a set of these targets on the surface of the cloth, locate them simultaneously

and use this to acquire a sparse mesh representation of the cloth from the input image. By using visual targets, the successful tracking of a non-rigid object can be achieved using only vision-based techniques as required. Throughout this thesis, an open source computer vision library called OpenCV [8] is used for basic image processing tasks.

The ultimate goal of this thesis is to perform real-time augmentations on a piece of cloth. However, transitioning from tracking a rigid plane in space to tracking a flexible piece of cloth is a large step. For this reason, the tracking methods presented here are first demonstrated on a sheet of paper, which does not exhibit the same degree of flexibility as cloth. In addition, since our tracking methods are target-based, it is much easier to place targets on a sheet of paper and be able to modify the targets simply by printing a new sheet. Placing targets on cloth is more difficult and therefore this step is only performed once the tracking methods are established using the paper version. Throughout this thesis, cloth and paper are used interchangeably as the non-rigid object to be tracked.

Three different target tracking systems are developed to solve the real-time flexible object tracking problem. The first is a colored-circle system that is designed to track a small number of differently-colored circles on the surface of a non-rigid object. This system is designed as a proof-of-concept exercise in order to determine the feasibility of the theory of tracking targets on a flexible object for augmented reality. With successful results of the colored-circle system, the next two tracking systems are developed with the goal of performing real-time augmentations. The second tracking method is a square target system that locates a set of unique square targets on the cloth, and the third method is a coded-ring system that tracks an even larger set of targets that represent unique binary-encoded rings. These two systems have their advantages and disadvantages, as outlined in this thesis.

3.1 Colored-Circle System

The first system, developed to acquire information about the surface of a non-rigid object in a real-time video stream, is a colored-circle tracking system. This is an

experimental system intended to determine the possibility of simultaneously tracking multiple points on the surface of a sheet of flexible paper, for the purpose of building a virtual representation of the object. This system attempts to track a small number of large, uniquely-colored circle targets placed on the paper. The tracking process uses the shape, color and intensity of the targets.

3.1.1 Setup and Initialization

In the spectrum of visible light, any group of three independent wavelengths that can be mixed to produce all colors are called *primary colors.* The main primary colors are red, green and blue. These three colors are easy to distinguish and isolate in an input image, since color images have a separate channel for each of the red, green and blue components of each pixel. Another set of primary colors are cyan, magenta and yellow. Cyan is the combination of green and blue light, magenta is the combination of red and blue light, and yellow is the combination of red and green light (see Fig. 3.1).

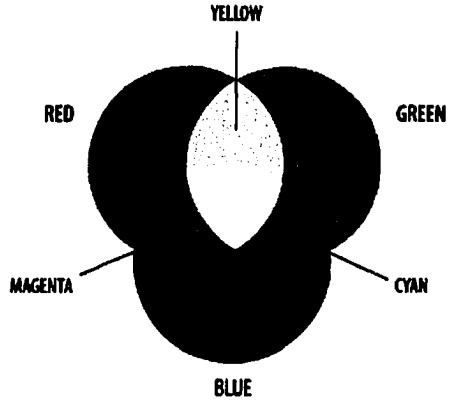


Figure 3.1: Primary colors. RGB and CMY both combine to make white light in an additive system.

These six primary colors are good candidates for target colors, since they are far apart from each other in the spectrum of visible light. During initial research for this tracking method, a series of other target colors were attempted, however tracking the targets became too difficult due to the similarity of the colors. The primary colors

are easier to uniquely identify in an input image, so six large circles are placed on a sheet of flexible paper, one of each color. By the same token, black and white are also good colors to use for identifying targets, however the background paper is chosen to be white, and in general there are too many natural small black objects in an indoor scene to effectively use that color for simple circular targets. With the white background, the targets are darkened (without changing the primary color) to increase contrast and simplify location. Unfortunately, the yellow target is too difficult to locate on the bright background, so it is removed from the system and the five remaining targets are spaced out accordingly. Fig. 3.2 shows the resulting sheet of paper with the five targets to be tracked.

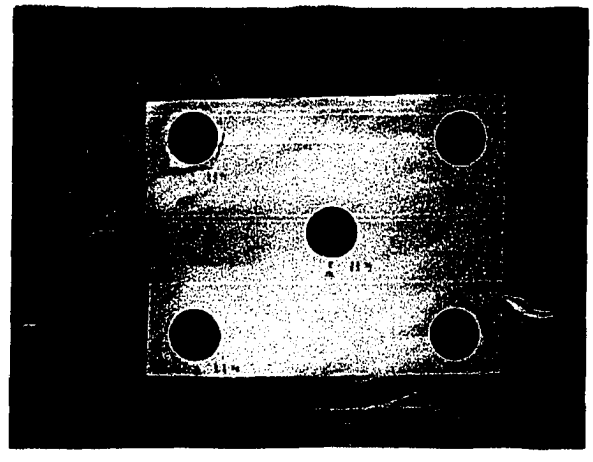


Figure 3.2: Dark colored-circles to be tracked on paper. Clockwise from top left: blue, cyan, magenta, green, red (in center).

3.1.2 Circle Tracking

The tracking process for the colored-circle system starts by creating a grayscale version of the color input image from the video stream. This may seem strange, since the system is based on the requirement of color to determine the identity of the targets. However, since the target colors are much darker than the white background, we first use the grayscale image to determine the location of possible targets. Once the locations are found, we then return to the color image to identify the targets.

The grayscale image is processed by comparing each pixel (in the range of 0-255) to a global threshold value, in order to highlight the contrast between the targets and the background. If a pixel is darker than the threshold then it is set to white, otherwise it is set to black. The threshold value is chosen to be lighter than all of the target circles, yet darker than the background sheet, even in the presence of shadows. This creates a binary image, consisting of only white and black pixels. The next step is to find all connected regions of white pixels (sometimes called *blobs*). The projection of a circle in 3D space onto an image plane in 2D space is an ellipse. Therefore, the blobs are processed individually to look for elliptical shapes. Blobs that are too small, do not have enough vertices (when approximated as a polygon), or are not convex enough to be ellipses are discarded. The remaining blobs are selected as potential targets and are passed on for further processing. Fig. 3.3 illustrates these first steps in the circle tracking process.

Now that we have determined possible target locations we can return to the color input image. For each blob representing a potential target, the pixel center is computed. Then the average red, green and blue values are calculated at this location. This is accomplished by summing the values for each channel in an eleven pixel by eleven pixel window centered at the pixel center of the blob. The last step is to determine a one-to-one mapping of the tracked colors to the known target colors. For each of the known target colors, the best matching potential target color is chosen. The matching is accomplished using color ratios. For instance, to find the blue target, the potential target with the best ratio of average blue to average red *and* average blue to average green is chosen. The reason is that the blue target will display a much higher blue value than red or green values. The other targets are chosen in a similar way, as illustrated in Table 3.1. The average red, green and blue values are R, G and B, respectively, and are clamped to the range of 1-255 (a value of zero is approximated to avoid dividing by zero).

To test the colored-circle tracking system, the located targets are statically connected with a small graph with a known topology. Graphical lines are drawn on the input image to indicate the graph and demonstrate the tracking result. Fig. 3.4 illustrates the final steps in tracking the colored-circle targets.

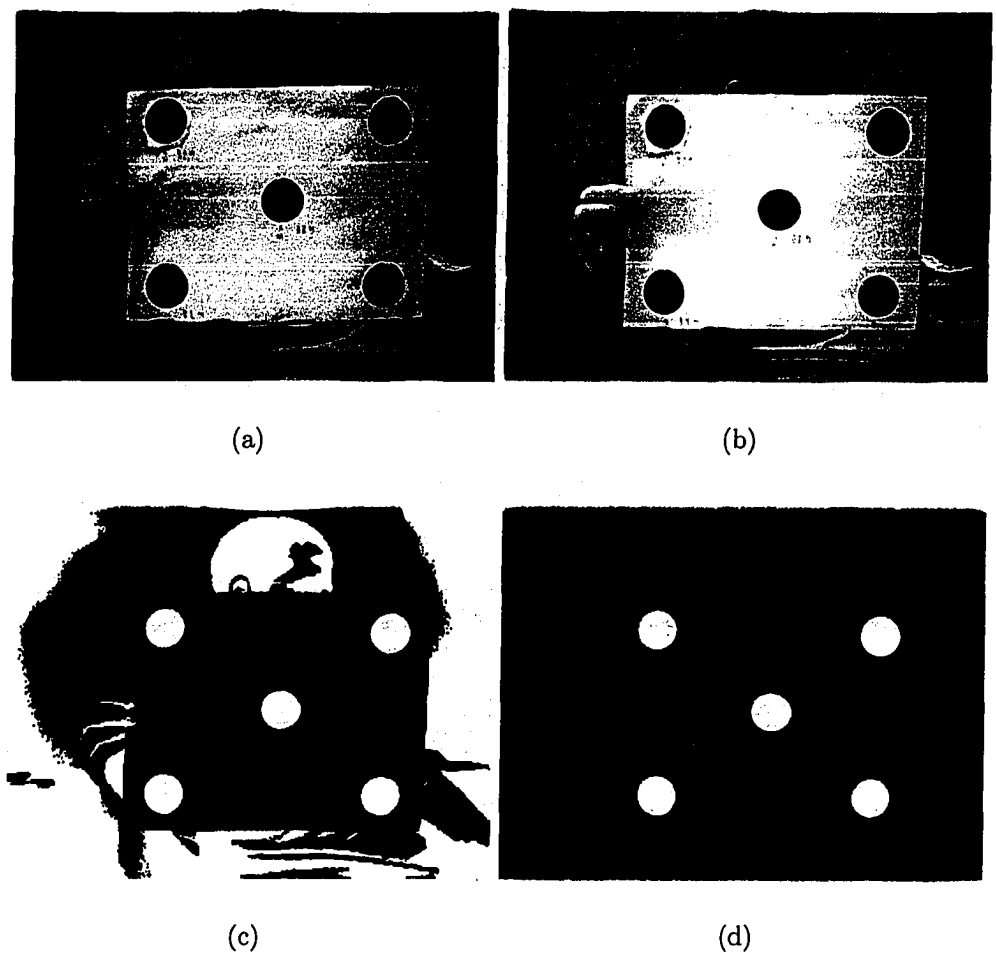
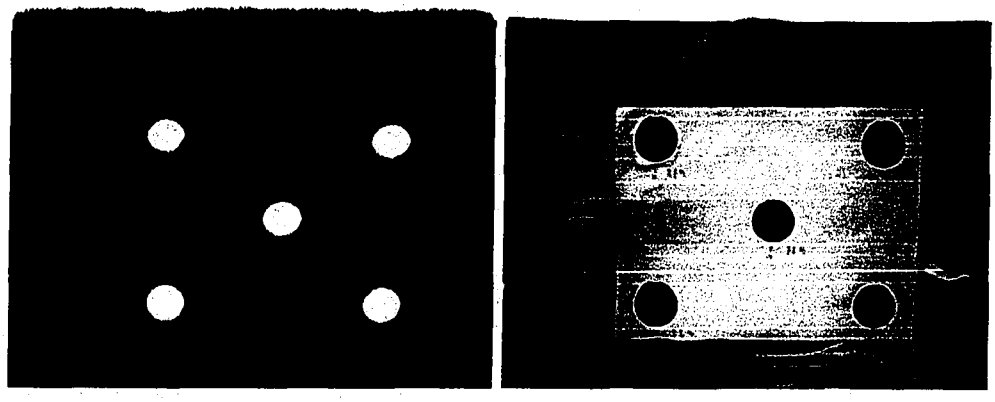


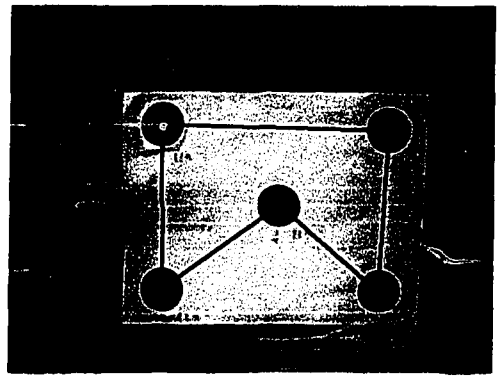
Figure 3.3: Initial steps in the colored-circle tracking process. a) Input image; b) Grayscale version of input; c) Binary threshold of grayscale image; d) Connected regions that pass as potential targets.

Color	Matching Function
Red	$\frac{R}{G} + \frac{R}{B}$
Green	$\frac{G}{R} + \frac{G}{B}$
Blue	$\frac{B}{R} + \frac{B}{G}$
Cyan	$\frac{B}{R} + \frac{G}{R}$
Magenta	$\frac{R}{G} + \frac{B}{G}$

Table 3.1: Matching colors in the colored-circle target system.



(a) (b)



(c)

Figure 3.4: Final steps in the colored-circle tracking process. a) Potential target blobs resulting from initial steps; b) Blob centers are computed on color image; c) Static graph connecting the targets to demonstrate results.

3.1.3 Discussion

The purpose of the colored-circle tracking method is to demonstrate the ability to sparsely track a non-rigid object by visually locating known points on its surface using unique targets. Fig. 3.5 shows the successful results of this method while moving and deforming the non-rigid sheet of paper.

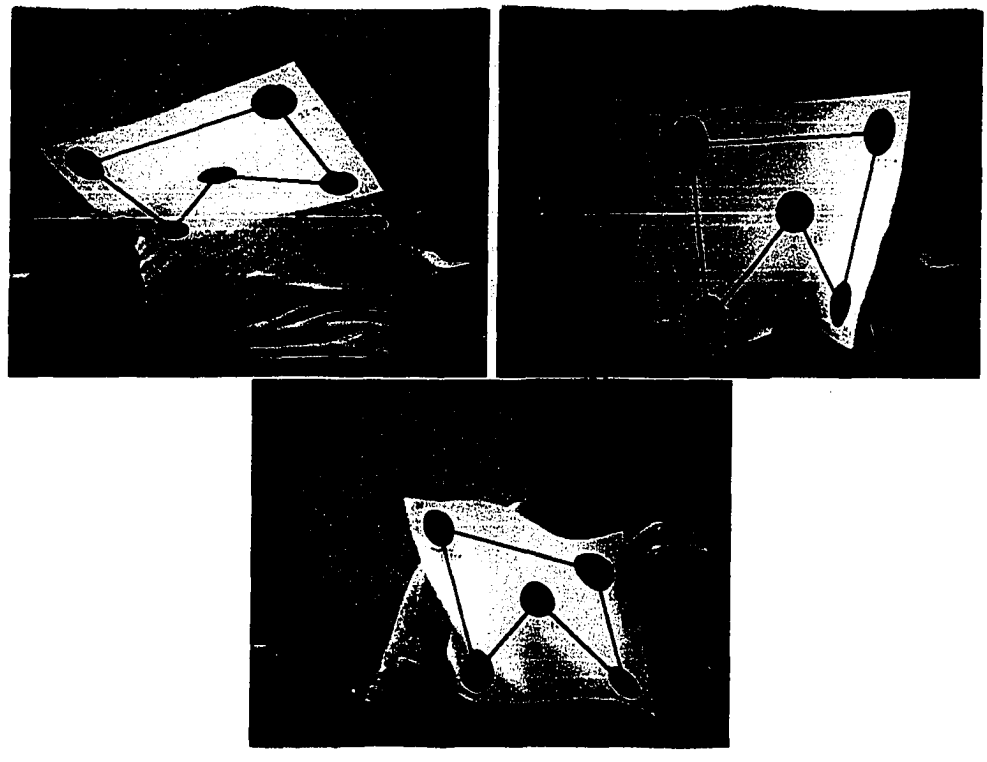


Figure 3.5: Results of colored-circle method under movement and deformation of the object.

The results of this tracking method indicate strong potential for vision-based tracking of non-rigid objects in the field of augmented reality. Unfortunately, it is not feasible to perform augmentations using this particular tracking method as only a small number of points are located on the surface of the object, resulting in a representation that is too sparse. Repeated patterns of the colored-circles could be used to obtain a more dense representation of the object, however this creates a dependency among target locations, removing the benefit of unique self-identifying targets. As well, it is too difficult to add more targets of different colors, since the

colors become too close to each other and tracking failures start to occur. For this reason, the colored-circle method is used only as a proof-of-concept method, and new non-rigid object tracking methods are explored.

3.2 Square Target System

The second system that is developed to track cloth in real-time is based on a set of self-identifiable square targets. This is a common approach for AR application developers. In fact, this approach is so common that an open-source library called ARToolKit [1] has been made available to provide tracking methods for these specific targets.

3.2.1 ARToolKit

ARToolKit is a software library that is used for building augmented reality applications. As mentioned previously, one of the main difficulties in augmented reality applications is the problem of computing the camera pose (ie: the user's viewpoint in the scene). ARToolKit solves this problem using computer vision algorithms to locate unique square targets in a real-time video stream. The algorithm to locate and identify the targets is described later. The square markers are comprised of a thick black border and a unique interior pattern. The border is used for locating the targets and the interior pattern is used for identification. Fig. 3.6 illustrates three typical ARToolKit markers. Notice that it is necessary for the interior pattern to be unique and deterministically identifiable in all four orientations of the marker.

The use of these self-identifying markers is common in AR applications, however it is not so common to require the correct identification of a large set of markers in one video frame simultaneously. The example ARToolKit markers in Fig. 3.6 are suitable when there are only a small number of markers to be detected. However, generating many patterns of this type can lead to false-positives and mis-identification in the tracking process. To establish more robust identification, we use markers with interior patterns constructed from orthogonal Discrete Cosine Transform (DCT) basis

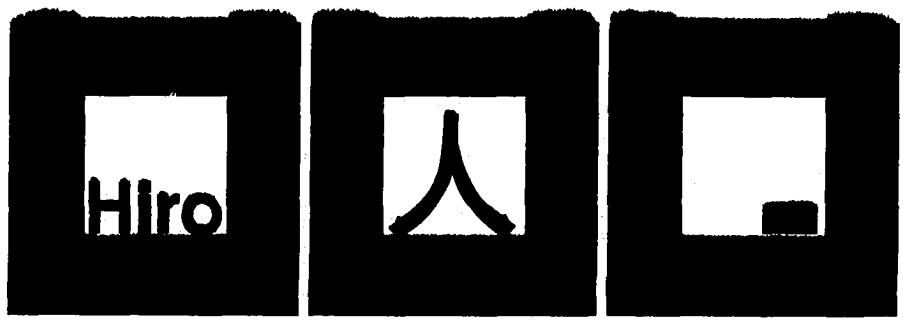


Figure 3.6: Typical ARToolKit markers.

images similar to Owen et al. [53]. The DCT patterns appear as repeated sinusoidal grayscale images with different horizontal and vertical wavelengths. A background gradient is also inserted to remove orientation ambiguity. This method allows us to generate tens of markers that can all be uniquely identified in a video image. Three examples of these markers are shown in Fig. 3.7.

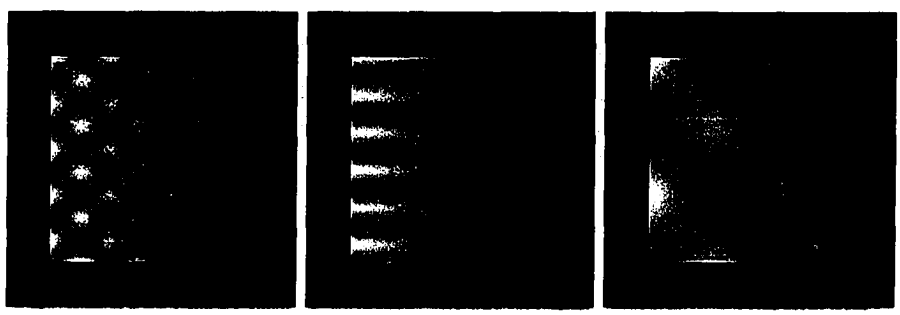
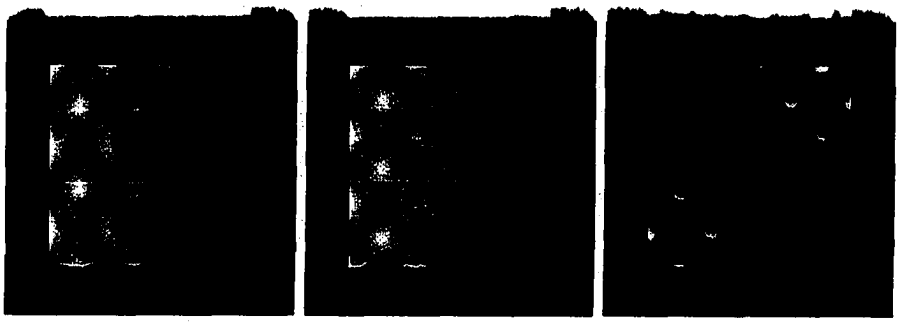


Figure 3.7: Example DCT markers.

To the human eye some of these patterns may look quite similar, however a quick computation of the pixel difference between two markers will highlight their individual uniqueness, due to the different sine wavelengths as illustrated in Fig. 3.8.

3.2.2 Setup and Initialization

In order to acquire a sparse mesh representation of the cloth we place sixteen targets on its surface, arranged in a 4 x 4 grid. To choose the sixteen DCT patterns that are most easily identifiable simultaneously, we attempted to track a uniform distribution of DCT patterns in one video frame (forty-seven targets to be exact) and then we



(a) (b) (c)

Figure 3.8: Pixel difference between DCT markers to illustrate uniqueness. a), b) Similar-looking DCT markers; c) Pixel difference between the markers (scaled by two for better illustration).

chose the sixteen patterns with the best tracking confidence values, as defined by ARToolKit. Fig. 3.9 shows the best sixteen markers arranged on paper.

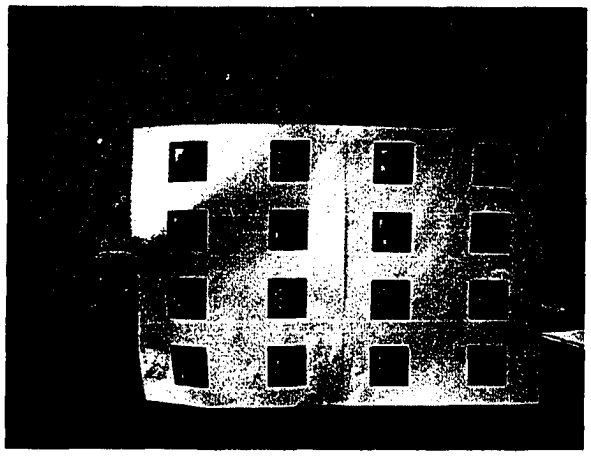


Figure 3.9: Square markers to track.

The application is then initialized by storing a 32 x 32 pixel sample of each interior pattern in the form of a vector, to be used when identifying possible targets later.

The physical size of the targets is determined by attempting to track multiple markers of different dimensions at a specific distance of one meter from the camera. Under a series of target movements and orientations, the smallest target size that is accurately tracked in all cases is selected. The target squares are chosen with a side

length of one and a half inches.

3.2.3 Locating and Identifying Targets

In the square target system, cloth tracking starts by locating all of the potential markers in the scene, based on their black square borders. The grayscale image of the input from the video stream is binarized by adaptively thresholding on pixel intensity values[1], and then connected regions of pixels are segmented into blobs. An adaptive thresholding method is chosen over a fixed threshold to provide more robust tracking in the presence of shadows and extreme illumination conditions. The segmentation is accomplished by examining the neighbourhood around each non-zero pixel and establishing a list of connected components. For each blob, the contour of the external boundary is extracted, and then a filter is applied to remove regions that do not have four-sided external boundaries. The result of this first processing stage is a set of possible marker locations. Fig. 3.10 illustrates these steps to locate potential markers in a scene.

Now each potential marker is examined individually. Markers are identified based on their unique interior pattern. The first step is to remove the perspective distortion of the pattern by applying a transformation to a 32 x 32 sample grid of the region. Specifically, the transformation is a one-to-one mapping called a *homography*[2], which is defined by a 3 x 3 matrix [34]. The homography is computed by mapping the four corners of the marker in screen coordinates to the four corners in *normalized pattern-space*. Normalized pattern-space is the coordinate space where the lower left corner of the marker has a value of (0,0) and the upper right corner has a value of (1,1). Standard mathematical techniques are used to compute the homography. Once computed, each pixel on the screen maps to exactly one point in pattern-space and vice-versa, providing an orthogonal view of the potential marker pattern. Each pattern is then compared to a set of stored images of the known patterns to be detected. The patterns are compared to all four possible orientations, and a confidence value for each comparison is computed as the normalized dot product between the 32 x

[1]The adaptive thresholding method is described in Appendix B.

[2]See Appendix A for more information on homographic transformations.

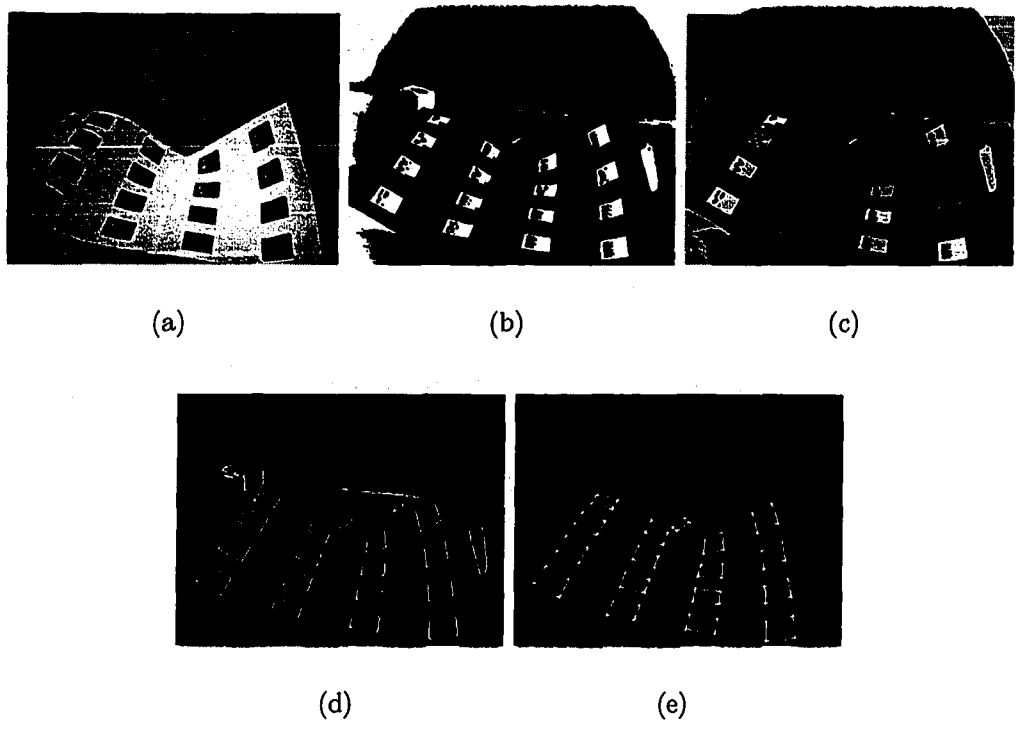


Figure 3.10: Establishing possible marker locations. a) Grayscale image of the input; b) Binarized image; c) Connected components (shaded differently for visualization); d) Region contours; e)Four-sided regions.

32 vector and the stored image. If any comparison generates a confidence value greater than a fixed ARToolKit threshold then it is claimed to be a match and the four corners of the marker in screen coordinates are returned. Since the cloth contains sixteen unique markers, the tracking procedure can find up to sixty-four pixel locations that map to known points on the surface of the cloth. These points are then used to render the augmentation, as described in Chapter 4.

3.2.4 Discussion

The square target tracking method is a fast approach that yields a reasonable number of identified points on the surface of the non-rigid object. There are, however, some disadvantages to this method. The first is that each target is responsible for four of the tracked surface points. This means that if a single target is occluded or otherwise not found in the video image, then four points will be missing from the final mesh, resulting in a large hole. Furthermore, since the tracking process requires that the targets are four-sided polygons, deforming or rippling the cloth can cause a tracking failure, even if the failing target is still in plain view. Though this problem can be minimized with smaller target sizes, the overall robustness would decrease from natural tracking failure due to an insufficient number of pixels comprising the interior patterns, and therefore using smaller targets would not be worthwhile. Despite the disadvantages, this tracking method shows impressive results for the first monocular non-rigid object tracker for the purpose of augmented reality.

3.3 Coded-Ring System

The coded-ring system is developed in order to overcome the shortcomings of the square target system and provide a more robust non-rigid object tracker. The theory behind this system is that a large number of very small targets can be accurately tracked, each yielding one point on the surface of the cloth. In order to robustly distinguish between many such targets simultaneously, a coded-ring marker system is adapted from the field of photogrammetry.

3.3.1 Photogrammetry Targets

Photogrammetry is the technique of measuring objects from photographs [2, 60]. Typically, photogrammetry applications use the 2D image of a 3D scene to reconstruct an accurate 3D model of the original scene. A popular photogrammetry tool is PhotoModeler Pro [55]. This application provides offline tracking of specific coded-ring targets in a series of images. The targets are generated by the tool, and then manually placed in the scene. The targets are self-identifiable, so locating the same target in multiple images allows the computation of the 3D location of that target. Reconstruction of the scene is accomplished by finding the 3D coordinates of all of the targets. The Photomodeler Pro technique to locate and identify the targets in a 2D image is a robust offline algorithm. We develop a slightly less reliable online algorithm for locating and identifying the coded-ring targets in a real-time video stream, allowing us to accurately track the surface of the cloth.

The coded-ring targets consist of an inner black circle surrounded by a white ring in the middle and a sequence of white and black ring segments on the outer border (see Fig. 3.11). The outer ring segments can be thought of as a 10-bit digital barcode wrapped around into a ring, where a 0 produces a white segment and a 1 produces a black segment. This ring is used to uniquely identify each target, independent of scale or orientation.

(a) (b) (c)

Figure 3.11: Coded ring markers. Outer ring is a 10-bit binary code wrapped around a black circle and inner white ring.

Photomodeler Pro can generate targets with 8-, 10-, 12- and 14-bit sequences. The easiest to identify are the 8-bit codes since the target segments are largest,

however only twenty-five unique targets are generated. With more bits, more unique targets are generated, however the outer ring segments become smaller and harder to locate in the image. The 10-bit coded-rings are selected since forty-five unique targets are generated, providing a sufficient number of identifiable points with an adequate minimum outer ring size.

3.3.2 Setup and Initialization

Of the forty-five generated targets, forty-two are selected to form a 6 x 7 grid of targets with known topology on the surface of the cloth. To choose the three unused targets, a test application is developed to determine the *false-positive rate* for each target. A false-positive occurs when the tracking system returns positive identification of a target that is not actually present at that location. The three targets with the highest false-positive rates are removed. This provides the most robust set of forty-two targets, which demonstrate better results than removing three targets at random. The false-positive rate is calculated over a series of video frames where no coded-rings are present. The number of positive tracking instances for each target (using our tracking algorithm described below) is recorded as the false-positive rate. Fig. 3.12 shows the best forty-two targets, arranged in the 6 x 7 grid, ready for tracking.

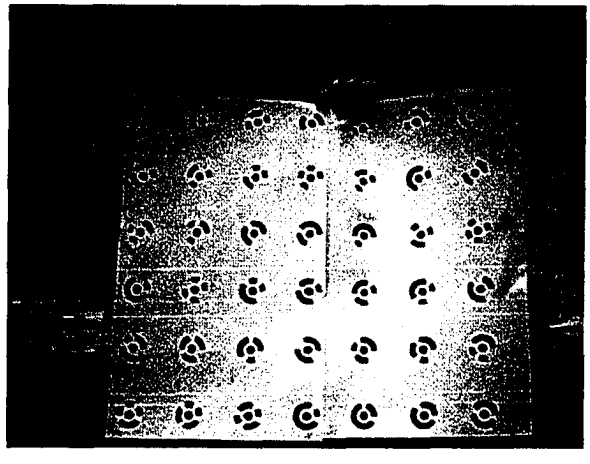


Figure 3.12: Grid of coded rings with known topology used to track the cloth (shown on paper).

It should be noted that an additional initialization step is to compute the ten binary strings that represent each target, one for each orientation. Actually there is only one identifying string for each target, and the other nine strings are generated by shifting the identifying string one bit each time. The integer values of the ten binary strings are stored in a look-up table, all pointing to the ID of the representative target.

3.3.3 Locating Coded-Rings

Tracking the coded ring markers starts by adaptively thresholding the grayscale image of the input to create a binary image[3]. As with the previous method, an adaptive thresholding technique is used to provide robust tracking in the presence of shadows and extreme illumination conditions. Contours of the connected regions in the binary image are found and a contour size filter is applied to remove unwanted noise. At this point we must determine which contours belong to the same marker, and then which specific contour represents the center circle of the marker. Simply processing the contours individually is not sufficient, so we perform a clustering step to group together contours belonging to the same marker. The clustering algorithm proceeds by calculating the pixel center of each contour to produce a set of 2D points on the image plane. Then the Delaunay triangulation [20] of the points is computed. The Delaunay triangulation is used since this triangulation seems to capture spatial relations within a point set [51]. We attempt to extract the clusters from the Delaunay Triangulation by using an approach similar to Kruskal's algorithm for computing the Minimum Spanning Tree of a graph [16]. Initially, each point is considered to be a single cluster. The edges of the triangulation are then processed from shortest to longest. When processing an edge, the two clusters that contain the end points of the edge are combined into a single cluster. This cluster merging is performed using a standard Union-Find data structure [69]. The Delaunay edges are processed until an application-specific termination condition is met, which indicates that the points are correctly clustered. For the case of our coded rings, the algorithm terminates when the number of clusters is at most half of the number of original points, and the length

[3]The adaptive thresholding method is described in Appendix B.

of the current Delaunay edge is more than 2% longer than the previously processed edge. At this point, each cluster represents exactly one coded ring target. Now for each cluster we compute its pixel center and then find the closest point in the cluster where the corresponding contour sufficiently resembles an ellipse. This contour is chosen as the center circle of the target. Fig. 3.13 illustrates these steps to find the target centers in a region of an input image.

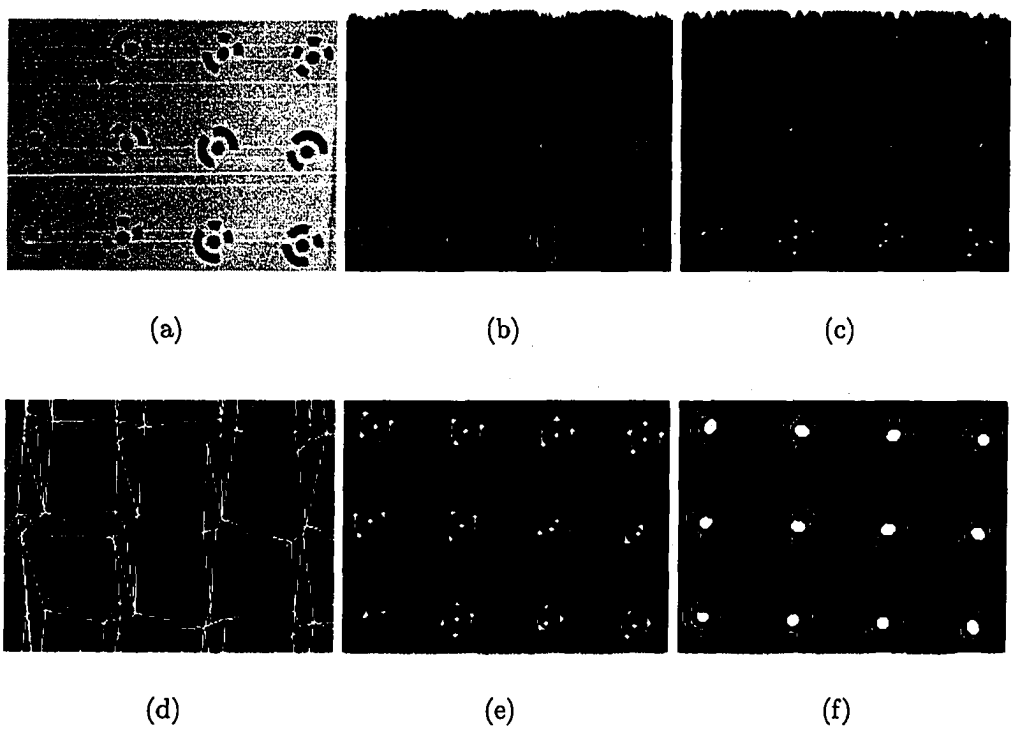


Figure 3.13: Finding the circular target centers (magnified region of an input image). a) Input; b) Contours are found; c) Center of each contour computed; d) Delaunay triangulation; e) Result of clustering (each cluster shown with a minimum enclosing circle of the cluster points); f) Target centers are found.

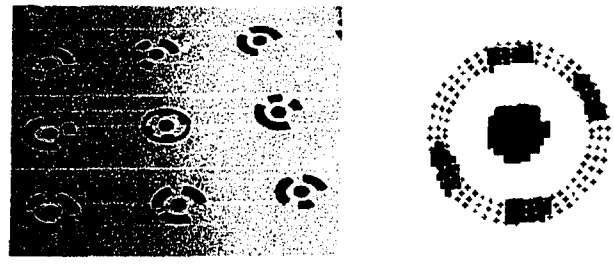
Let us briefly consider the processing time of the clustering algorithm, since clustering must be performed for each frame of the real-time video sequence. If there are N contours found in the image then N points are triangulated using a memory-efficient version of Fortune's $O(N \log N)$ sweepline algorithm for Voronoi diagrams [25, 7]. This produces $O(N)$ edges, and for each edge we perform cluster merging, which operates in $O(\log N)$ time per merge. The total running time of the clustering step is

therefore $O(N \log N)$. Although there are more theoretically efficient methods to perform the merge operation, we use the simple merging algorithm outlined in Weiss [69] since computing the Delaunay triangulation already has an $\Omega(N \log N)$ lower bound in the algebraic decision tree model of computation and the more complex merging algorithms tend to perform poorly on smaller inputs.

3.3.4 Identifying Coded-Rings

Now that the target locations have been found we can uniquely identify each target using the coded outer ring. To do this, we start by enlarging the elliptical projection of the target center circle by a constant factor to obtain an ellipse that represents the outer border of the target. Then we compute a homography[4] from that ellipse to a circle in order to remove the perspective distortion. The un-warped target is then sampled at multiple locations to determine the binary string representation of the outer ring segments. We sample at three different radius values (77%, 85% and 92% of the circle radius) and five orientations per radius, for a total of fifteen samples (one hundred and fifty sample points). Fig. 3.14 illustrates the application of the homography and the fifteen samples for one target. Each sample is a 10-bit binary string generated from the pixel values at the sampled radius starting from the specific orientation. The binary strings are used in the look-up table mentioned earlier to retrieve the corresponding target ID. As indicated, each ID produces ten entries in the table, one for each possible orientation. If the number of samples that agree on the same target ID is above an agreement threshold value then the target is labeled with that ID. If multiple targets are found to have the same ID, then the one with the most samples that agree on the ID is chosen. If the agreement threshold is not reached then the cluster is considered to be noise and is subsequently ignored. The result of the coded ring tracking method is a set of pixel locations, indicating the target centers of the uniquely identified targets. These points are then used to render the augmentation, as described in Chapter 4.

[4]See Appendix A for more information on homographic transformations.



(a) (b)

Figure 3.14: Removing perspective distortion and sampling to uniquely identify a target. a) Elliptical border of the target in question is shown in red; b) Homography is applied and the now circular target is sampled at the one hundred and fifty locations shown to produce fifteen 10-bit samples.

3.3.5 Discussion

The coded-ring system is the most robust of the three techniques to track a non-rigid piece of cloth in a real-time video stream. The main reason for this robustness is the fact that the targets are smaller than the square targets (approximately 33% smaller) and therefore are less susceptible to deformation when waving or rippling the cloth. Also, with a smaller size we can place more of the targets on the cloth and establish a one-to-one mapping of targets to surface points. This means that if the system fails to track one target then only one surface point is lost.

Unfortunately the disadvantage of the coded-ring system is the increase in the required amount of processing. This technique is noticeably slower than the square target system, so the typical trade-off between speed and accuracy is evident.

Real-time non-rigid object tracking is a very young field of research. This thesis provides important first results of different tracking methods for cloth, with the understanding that the optimal algorithms have yet to be discovered.

Chapter 4

Augmentation with Realistic Illumination

The main motivation of this thesis research is to be able to perform realistic virtual augmentations on a real piece of cloth, while interacting with the cloth in real-time. Chapter 3 describes the algorithms developed for tracking the cloth in a video-stream. These algorithms result in a set of pixel locations representing known surface points on the cloth. This chapter describes the techniques to perform augmentations on the cloth using those surface pixels. Although there are three tracking algorithms described in Chapter 3, the first technique is simply a proof-of-concept exercise, so augmentations are only being performed for the square target method and the coded-ring method. There are two distinct aspects of performing illumination-correct flexible augmentations. The first problem is to perform basic flexible augmentations while ignoring illumination, and then the second aspect is to increase the realism of the augmentation by incorporating the real world illumination environment.

4.1 Basic Flexible Augmentations

Flexible augmentations on non-rigid objects are presented here in the form of virtual 2D textures that appear to lie on the surface of the cloth or paper. In order for the augmentations to appear real they must adhere to the object during interaction,

which includes rippling and stretching the cloth under any arbitrary orientation. It is for this reason that the non-rigid object tracking methods require unique targets in a known topology, and produce a set of known locations on the cloth surface. These locations are used to build a virtual mesh representation of the cloth for the augmentation. Augmentations are performed slightly differently for each of the two tracking methods, however in both cases the rendering environment is set up as described in Section 2.1. Recall that in AR applications, there are two categories of objects: real ones and virtual ones. The real objects exist only in the video image and the virtual objects are individual entities, typically in a 3D scene. Even though we are only rendering 2D augmentations, the scene is still three-dimensional. The virtual mesh triangles are rendered onto the same plane as the background image. The specific method of rendering augmentations on the tracked non-rigid object is now discussed for each of the two tracking systems.

4.1.1 Square Target System

In the square target system, each target yields four points on the cloth surface, and therefore four vertices in the virtual mesh. The mesh is built statically by connecting the corner points of the grid of targets in a triangulation. Fig. 4.1 illustrates the static triangulation of the target corners.

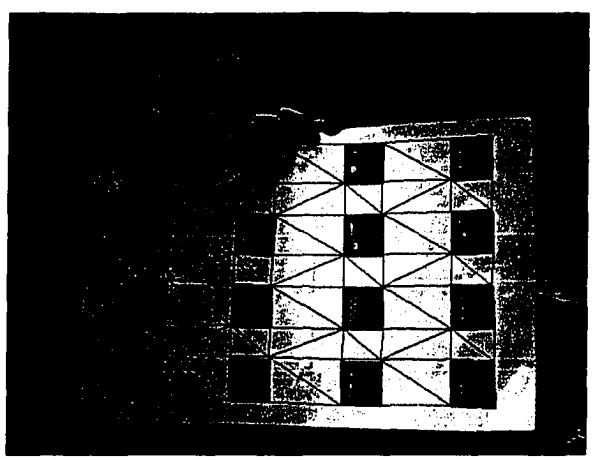


Figure 4.1: Static triangulation used in square target system.

The topology of the virtual mesh is defined in advance, so rendering the mesh is a simple matter of iterating through the triangles. For each triangle, we check to see if the required targets are found (the targets whose corner points are vertices of the triangle), and if so then the triangle is rendered at the pixel location of the corner points. If any one of the required targets is not found, then the triangle is skipped and a hole appears in the mesh.

The last step in rendering a basic augmentation is to add a texture to the mesh. A texture image is defined, and then the triangles are rendered with static texture coordinates, determined by the position of the corresponding target corner points with respect to the overall target grid. Fig. 4.2 shows a basic augmentation using the square target tracking method.

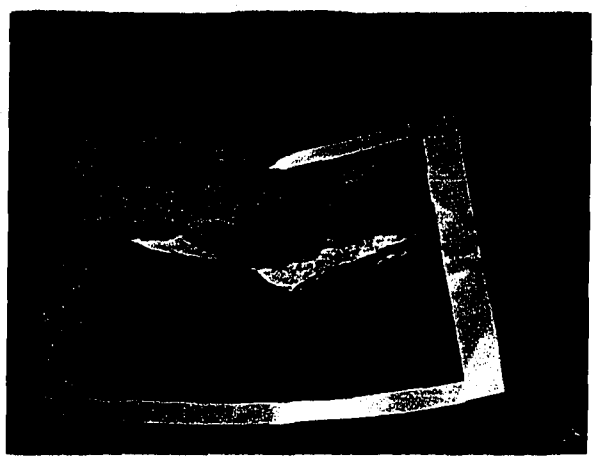
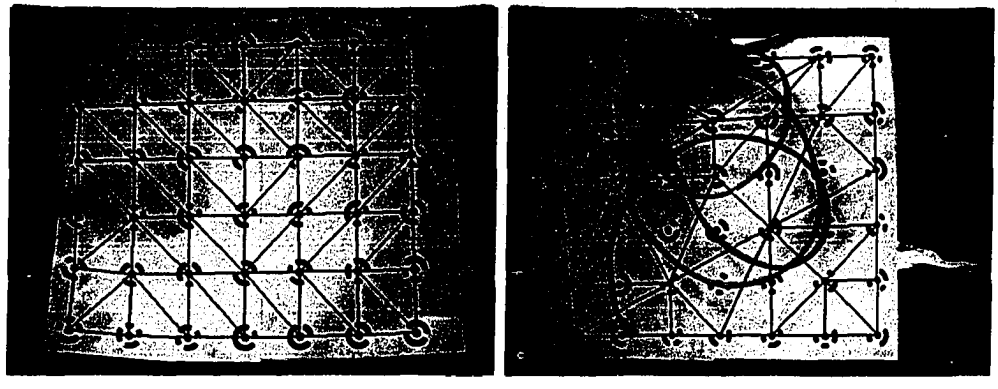


Figure 4.2: Basic augmentation using the square target tracking method.

4.1.2 Coded-Ring System

In the coded-ring system, each target maps to exactly one point on the surface of the cloth, and therefore one vertex in the virtual mesh. This means that if a single target is not found by the tracking process, then only one mesh vertex is missing (as opposed to four vertices in the square target method). Overall this means that fewer vertices are missing in the virtual mesh, or at least that the missing vertices are generally more spread out in this system, and not *always* adjacent to each other as is

the case in the square target system. For this reason, we are able to dynamically build the mesh from the vertices that are found and interpolate over the vertices that are missing each frame. Interpolation is not a viable option for the square target system because too much information is lost when four or more adjacent vertices are missing in the mesh. In the coded-ring system, interpolation is accomplished by dynamically computing the Delaunay triangulation of the pixel surface points that are found in the tracking process. Fig. 4.3 shows the dynamic triangulation of the located targets to form the augmentation mesh, when all the targets are located (a), and when some of the targets are occluded by a cable (b).



(a) (b)

Figure 4.3: Dynamic triangulation used in coded-ring system. a) When all targets are found; b) When some targets are not found (occluded by a cable).

The drawback to using the Delaunay triangulation algorithm to generate the augmentation mesh is that the algorithm outputs a mesh with a convex hull [18]. This is a problem along the border of the mesh when the cloth or paper forms a non-convex shape, as in Fig. 4.4. To solve this problem, a validity test is performed on each triangle before rendering. The test removes triangles formed by three vertices that lie on the same border edge.

As with the square target method, each coded-ring target is assigned static texture coordinates relative to the position of the target in the grid. This is possible because each target represents a specific point on the cloth surface and hence the

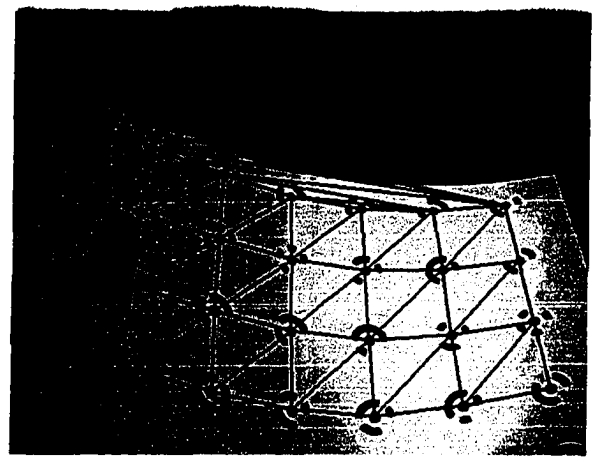


Figure 4.4: Drawback of using the Delaunay triangulation algorithm.

same augmented texture location no-matter where it is found in the image or what other targets are found in that vicinity. Fig. 4.5 shows a basic augmentation using the coded-ring tracking method.

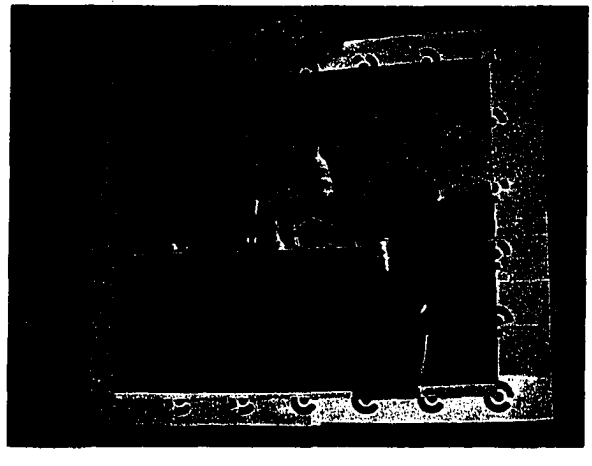


Figure 4.5: Basic augmentation using the coded-ring tracking method.

4.2 Realistic Illumination

We have now accomplished basic augmentations on a flexible object. However, the augmentations do not look very realistic in all illumination environments. For instance, consider Fig. 4.6. It is clear that the paper containing the augmentation is

brighter on the right side of the image and darker (due to shadows) on the left side. This is not reflected in the virtual part of the scene, and therefore the inconsistency results in an unrealistic augmentation. To solve this problem, common illumination

Figure 4.6: Lack of realism without proper illumination.

between the real and virtual parts of the scene must be established. The addition of realistic lighting to AR scenes increases perception of spatial relations between real and virtual objects [42]. As discussed in Section 2.3, common illumination for the case of this thesis is achieved by solving the problem of casting shadows of real objects onto the virtual object. The benefit of this approach is that the virtual object is a 2D mesh rendered directly onto the real object, so the white sheet of paper or cloth displays the exact shadows that are to be rendered onto the augmentation.

Two different methods of establishing common illumination are developed. In both methods, the illumination environment is acquired directly from the input image from the single camera. The first method is a very simple approach that approximates soft shadows which are viewed on the non-rigid object and incorporates them into the augmentation. The second method renders the exact hard and soft shadows with the augmentation at an additional cost of complexity and computation.

4.2.1 Approximate Soft Shadow Method

The approximate soft shadow method of incorporating realistic lighting is developed with the square target tracking system in mind. However, this method can be applied to the coded-ring tracking system with a little modification. The main idea behind this method is to sample the illumination condition at multiple points on the surface of the cloth and then interpolate the lighting across the augmentation. This is accomplished by acquiring an illumination intensity (a value between 0 and 255) at each vertex of the mesh, since those surface points are already known. Gouraud shading [28] is then used to blend the illumination smoothly over the mesh triangles. This method does not capture hard shadows correctly, however even rendering with an approximation of the soft shadows greatly increases the realism of the augmentation. In order to calculate a light intensity for each mesh vertex we use the fact that the cloth is white, and that it is already exhibiting the exact shadows we wish to approximate. Each vertex of the mesh corresponds to a marker corner on the cloth surface. Once a marker is found, we compute vectors from the center of the square marker to each corner location and then extend the vectors by a small constant value. This results in a surface position *just* outside the target corner, over a white area of the cloth. We then read the grayscale value of the input image at the computed locations and assign those values to the light intensities of the corresponding mesh vertices. Fig. 4.7 illustrates the locations on the input image from where the light intensity values are read for a specific marker. The actual intensity values are also displayed for visualization.

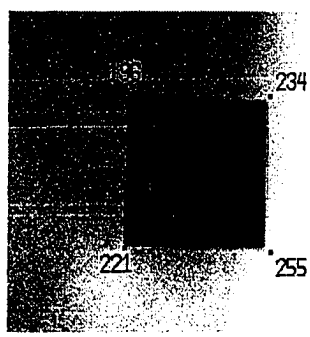


Figure 4.7: Sampling the grayscale input image for illumination.

The intensity values are used as a grayscale color for each vertex and blending with the textured triangles is accomplished with Gouraud shading. The 2D textured mesh is finally rendered with this approximate illumination, resulting in a realistic augmentation on a non-rigid object. Fig. 4.8 illustrates an example of the approximate soft shadow method for establishing common illumination when augmenting a non-rigid object.

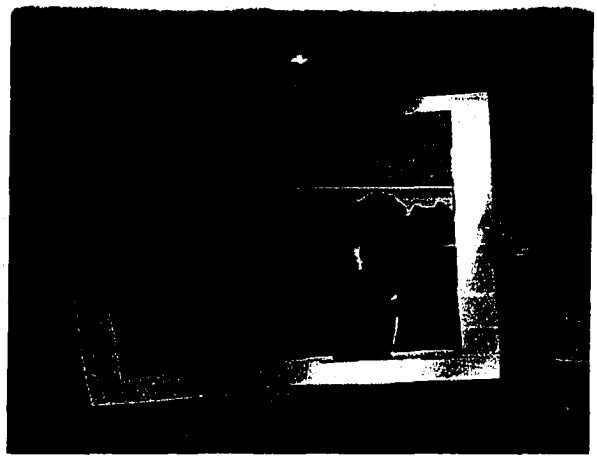


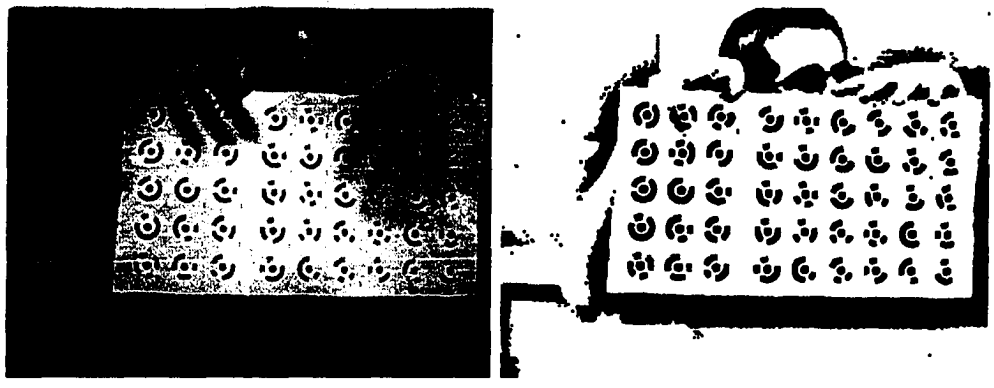
Figure 4.8: Approximate soft shadow method for common illumination.

4.2.2 Exact Hard and Soft Shadow Method

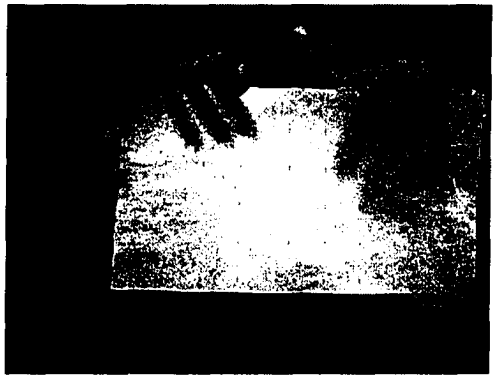
The second method that is developed to capture the real light environment and establish common illumination is an attempt to produce even more realistic augmentations. This is accomplished by rendering the exact hard and soft shadows that should appear on the virtual mesh. This method is developed for the coded-ring tracking system. In the previous method, the illumination on the cloth is sampled at each marker corner point and then interpolated over the augmented mesh. For this reason, hard shadow lines are not preserved. Our second method resolves this problem by capturing the exact shadows that are cast onto the cloth. The shadows are acquired directly from the input image. As with our previous method, we make use of the fact that the cloth is white in color, and that the shadows that are cast onto the cloth are the exact shadows that we wish to apply to the augmentation. We render the cloth

augmentation in two passes. The first pass renders the textured mesh without illumination. The second pass re-draws the mesh using a grayscale illumination texture with standard OpenGL blending to multiply the source pixels with the destination pixels and produce an illumination-correct textured mesh. The illumination texture is created dynamically each frame. The idea is that the illumination texture is exactly the grayscale image of the input without the coded ring targets. This is because the projection of the white cloth on the image plane is displaying the shadows that we wish to apply to the augmentation. However, we do not wish to add the black coded rings to the augmentation. We use *image inpainting* to remove the coded rings from the intensity image and replace the black areas with the surrounding illumination environment. Image inpainting is an ancient technique to make undetectable modifications to images [58]. The inpainting process is often used to remove unwanted objects from a scene, or to restore damaged paintings and photographs. Automatic digital inpainting methods have recently been developed to fill in a user-defined mask area using information from surrounding pixels [47, 5, 4]. However these inpainting techniques typically require many iterations and are executed offline for a single image. We implement a simple and fast image inpainting algorithm. Although our method is less accurate than traditional ones, our algorithm can be used on video frames as they are processed in real-time. The mask area for our inpainting method is defined by the contours of the coded rings, located in the target finding step. The contours are dilated by a 3 x 3 mask in order to slightly enlarge the individual mask areas, which accounts for small pixel errors in the target finding. The source image for our inpainting method is the grayscale image of the input. Inpainting proceeds by computing the bounding rectangle of each mask region. Rectangles are then individually scanned in a decreasing clockwise spiral, starting from the leftmost pixel in the top row. For each black pixel in the rectangle, the corresponding pixel value in the source image is set as the average of the surrounding eight pixel values whose corresponding mask pixels are white. After processing a black pixel in the mask image the pixel is set to white. This is a brute-force inpainting algorithm, so some artifacts do occur in the resulting image. However, since most of the source image consists of blurry shadows, the artifacts are less noticeable and results are adequate for our

application. Fig. 4.9 demonstrates the results of our image inpainting method. The inpainted image is used as the illumination texture during the second pass of the augmentation rendering. The projected screen coordinates for each mesh vertex are used as the texture coordinates for this illumination texture.



(a) (b)



(c)

Figure 4.9: Simple image inpainting. a) Source image; b) Dilated mask image; c) Result of inpainting algorithm.

Fig. 4.10 illustrates an example of the exact hard and soft shadow method for establishing common illumination when augmenting a non-rigid object. In this example, the exact shadow of a hand is correctly rendered onto the augmentation.

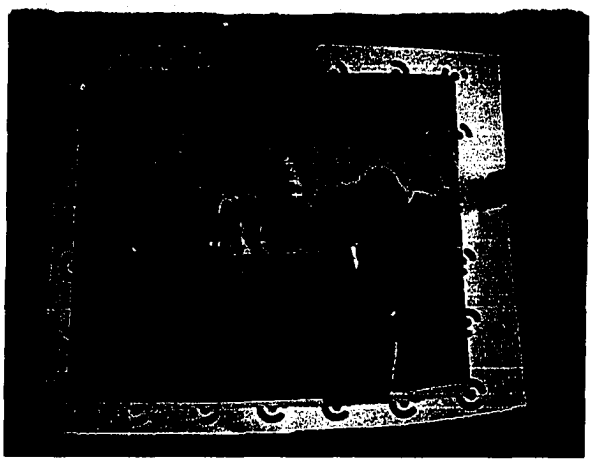


Figure 4.10: Exact hard and soft shadow method for common illumination.

4.3 Discussion

The approximate soft shadow method for augmenting cloth with realistic lighting is a very simple and fast solution. There is no pre-processing required or computationally expensive techniques involved, since the illumination is acquired simply by sampling the input image. However, the simplicity comes at the cost of accuracy. Specifically, hard shadow lines are not maintained in the rendered augmentation. On the other hand, the exact hard and soft shadow method does capture all types of shadows and renders them on the augmentation with almost no loss in accuracy (the image inpainting method is the only lossy step). However, a more complex and computationally expensive technique is required to create the exact shadow texture and rendering requires two passes instead of just one. Still, the second method yields more realistic augmentations and future research should focus on improving the efficiency of this method.

Although the approximate soft shadow method is developed for the square target tracking system, this technique can be modified slightly and used with the coded ring system as well, if desired. The one step in the process as described that is dependent on the square targets is when vectors are computed from the center of the targets to the corners, and then extended to indicate a location very near each mesh vertex that is over a white part of the cloth. This is required to sample the illumination for each

vertex. In the coded ring system, each target corresponds to one mesh vertex, so a technique would be required to sample the illumination on a white part of the cloth near the center of each target. Although the target centers are black in the coded-ring system, the middle ring of each target is white, and therefore the light intensity can be sampled uniformly within that ring and the approximate soft shadow method can be used to establish common illumination using the coded-ring tracking system as well. In the reverse case, the exact hard and soft shadow method is not as feasible when considering the square target system. The actual implementation would work without modification, however the image inpainting algorithm is well suited for the coded-ring system because the individual ring segments are small and easy to remove from the shadow texture image with few artifacts. On the other hand, the square targets are much larger and are not segmented into smaller regions, so the inpainting algorithm would attempt to fill large square regions in the shadow texture image, resulting in many artifacts and loss of accuracy.

Chapter 5

Results

This chapter outlines the results of this thesis. The first section describes the technical environment that is used for testing, including the running time of the methods presented. The remainder of this chapter contains images captured from the interactive application that accompanies this thesis. As mentioned earlier, the tracking and augmentations are performed on both paper and cloth objects. Results are shown for both objects individually, as the cloth can sag, stretch and wrinkle in a different way than the paper.

5.1 Technical Environment

The non-rigid object tracking and flexible augmentation systems are tested on a Pentium 4 processor running at 3.4GHz. The system contains 1GB of RAM, and is running Microsoft Windows XP. The capturing device is a Point Grey Dragonfly fire-wire camera, which captures color images at a resolution of 640 x 480 pixels.

The three tracking methods are measured individually over a sequence of five hundred frames each. Fig. 5.1 shows a graph of the frame rates of each method. The colored-circle tracking system runs at an average frame rate of 30.7 frames per second. The square target tracking system with the approximate soft shadow illumination method runs at an average frame rate of 14.4 frames per second. The coded-ring tracking system with the exact hard and soft shadow illumination method runs at an

average frame rate of 5.9 frames per second.

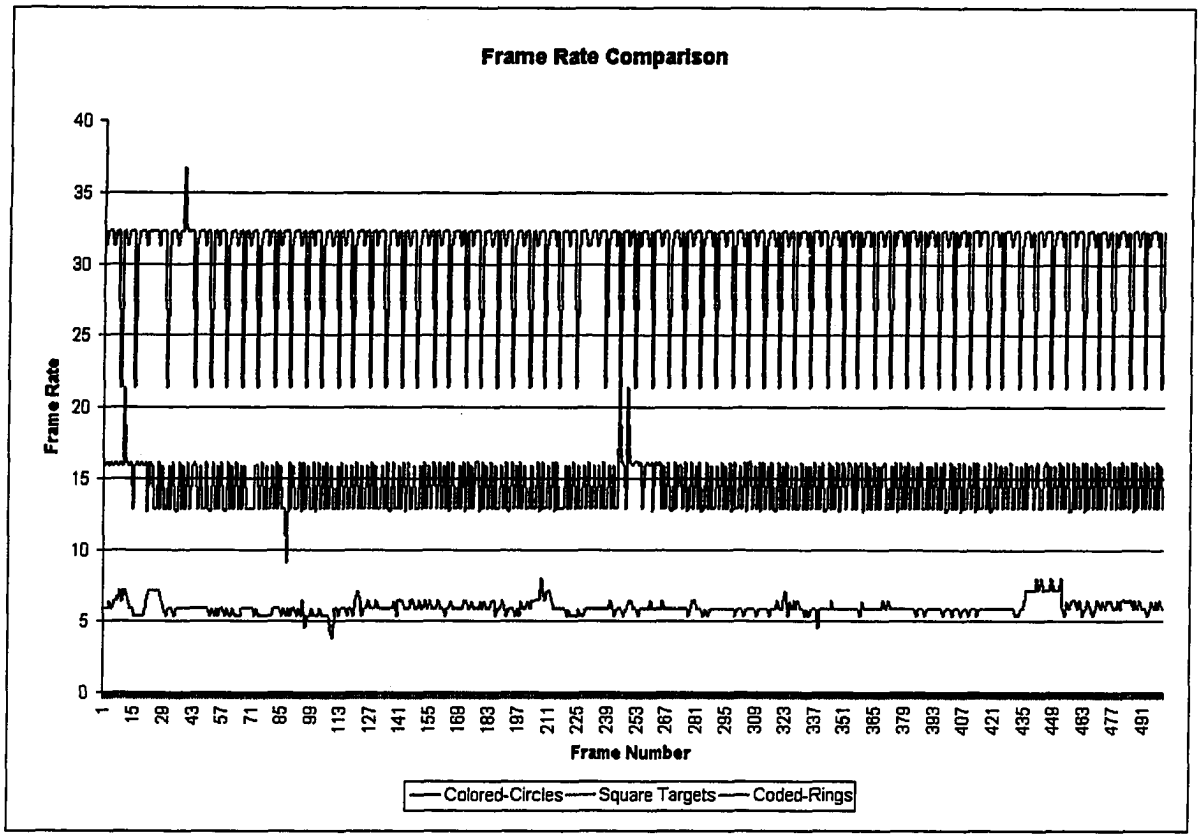


Figure 5.1: Graph of the frame rates of each method over five hundred frames.

Unfortunately, the coded-ring system with the exact shadows does not meet the standard frame rate of ten frames per second in order to qualify as officially real-time, however at nearly six frames per second the system is still quite interactive and results are impressive.

In order to determine the areas that use the most amount of processing time in the coded-ring method, the system is profiled using the standard Visual C++ profiling tool. The profile of interest is the *function timing* profile, which indicates the amount of processing time used in each function, along with the number of times each function is called in a given time period. Table 5.1 shows the profiling results for the twelve most expensive functions (in terms of processing time) for the coded-ring system during a 12.6 second execution of the program. Note that all functions with the "cv" prefix are part of the Intel Open Computer Vision library.

Func time (ms)	%	Func+Child time (ms)	%	Hit count	Function
3450	27.5	3450	27.5	20493758	cvGetReal2D
1933	15.4	1933	15.4	7667870	adaptiveThresholdTest
1639	13.0	3912	31.1	918045	inpaintOnePixel
1607	12.8	6149	49.0	17	findCodedRings
776	6.2	1124	9.0	2446007	boundedForwardHomography
705	5.6	5637	44.9	3360	inpaintOneRegion
348	2.8	348	2.8	2446007	forward_homography
338	2.7	338	2.7	1833296	cvSetReal2D
309	2.5	309	2.5	1392746	cvSetReal1D
239	1.9	239	1.9	34	cvResize
220	1.8	220	1.8	34	cvCvtColor
121	1.0	121	1.0	17	createIntegralImage

Table 5.1: Coded-ring system function profile.

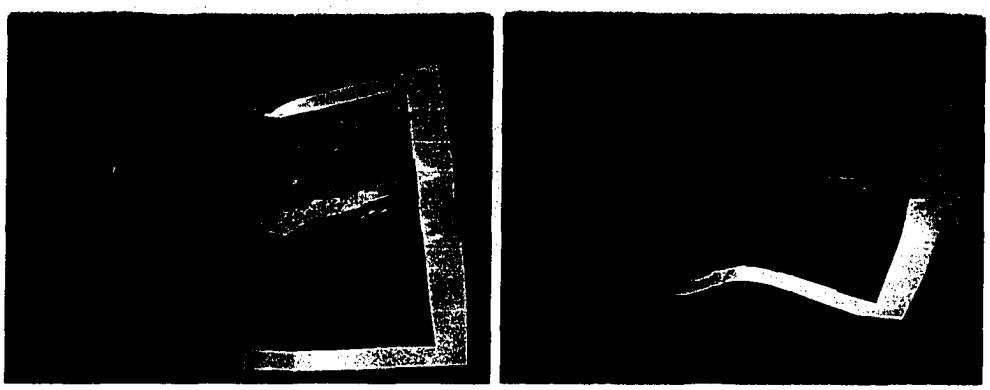
Table 5.1 provides quite a bit of insight regarding the efficiency of the coded-ring system. First and foremost is that nearly 30% of the processing time is spent in the function *cvGetReal2D*. This is a function to access the pixel value of an image given a two-dimensional index. This profile indicates that the function contains extra processing, likely to ensure the validity of the indices passed in. A significant improvement to the system can be made by replacing this function with code to access the image data directly from memory. Similarly, *cvSetReal2D* and *cvSetReal1D* can be replaced with direct memory manipulation routines. Table 5.1 also indicates, as expected, that adaptive thresholding, inpainting, homography computation and creation of integral images take up most of the processing time of the system. Further optimizing of these functions would yield the most benefits in terms of increasing the frame rate of the application. As a final step in this thesis, the optimizations mentioned here are implemented in the system and the resulting average frame rate of the coded-ring method is 10.2 frames per second. This is a 73% increase in speed.

5.2 Square Target System

Results of the square target tracking system with the approximate soft shadow illumination method are now illustrated for the paper and the cloth versions.

5.2.1 Paper Results

Flexible augmentations with approximate soft shadows are shown on the paper version in Fig. 5.2. Notice that self-shadows cast on the paper are rendered in a believable way with the augmentation.



(a) (b)

Figure 5.2: Flexible augmentations using the square target method with approximate soft shadows on paper, rendering self-shadows.

Further results of this common illumination method are shown in Fig. 5.3, where the soft shadow of a hand is approximated on the augmentation.

The final test of the approximate soft shadow method is to create an extreme illumination condition by dimming the ambient light and shining a flashlight on the paper. Results show impressive tracking of the targets under the radical conditions and believable shading of the augmentation, as seen in Fig. 5.4.

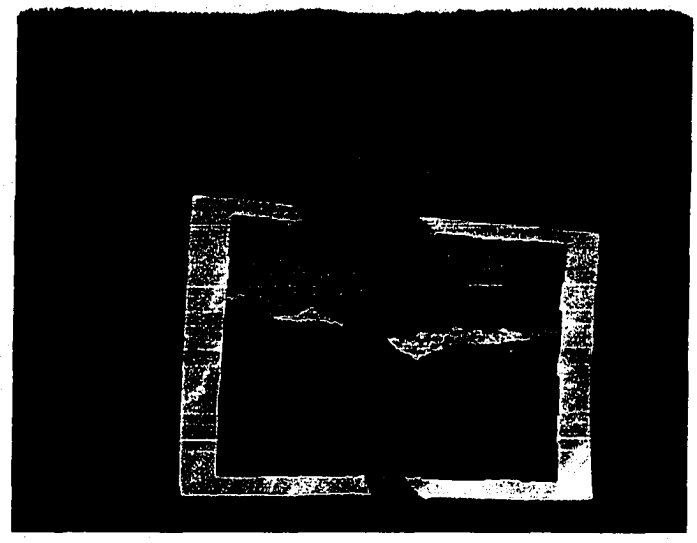
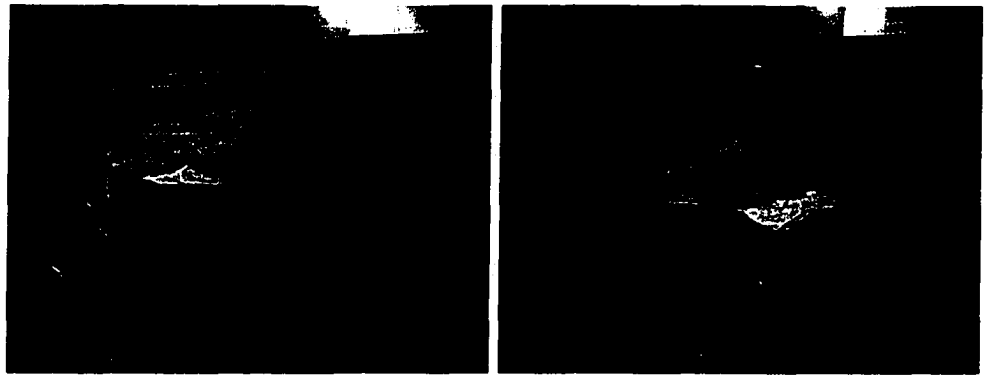


Figure 5.3: Approximating the soft shadow of a real object (a hand) on the augmentation.



(a) (b)

Figure 5.4: Extreme illumination conditions created by a flashlight and low ambient light.

5.2.2 Cloth Results

The results of the square target tracking system on the cloth object are not as good as the paper version. Specifically, tracking fails for more targets in each frame, resulting in holes in the augmentation. This is because the cloth is more flexible and the targets are more easily occluded or deformed by rippling the cloth. Some accuracy of the actual targets is also lost in the method to transfer the targets onto the surface of the object. The square targets are printed on special paper that is ironed onto the cloth. This creates targets with less contrast than the paper version, and some damage where the targets do not stick perfectly to the fabric. However, for the frames that do not contain a tracking failure, the results show realistic flexible augmentations on a non-rigid piece of cloth. Fig. 5.5 shows the augmentation on the cloth while it is sagging (Fig. 5.5(a)) and stretching (Fig. 5.5(b)).

(a) (b)

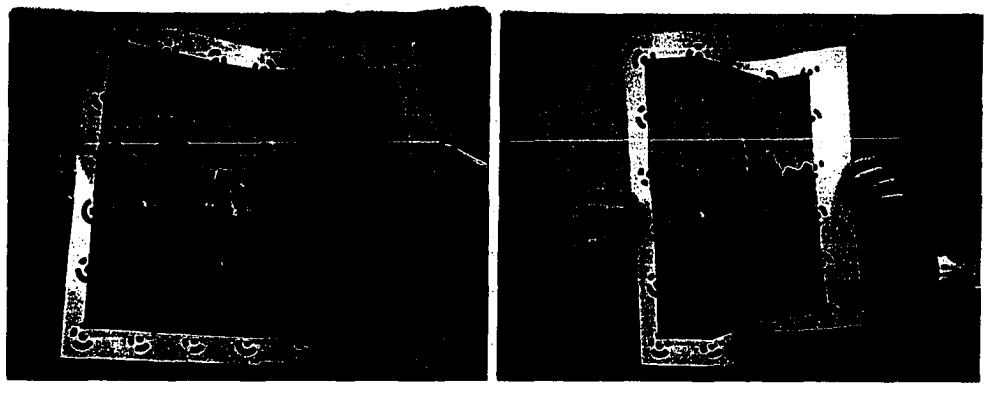
Figure 5.5: Square target tracking method on cloth. a) Sagging cloth; b) Stretching cloth.

5.3 Coded-Ring System

Results of the coded-ring tracking system with the exact hard and soft shadow illumination method are now illustrated for the paper and cloth versions.

5.3.1 Paper Results

Flexible augmentation results using the coded-ring tracking system are illustrated in Fig. 5.6. These results also demonstrate the acquisition and application of the exact hard and soft shadows that are present in the scene. Fig. 5.6(a) shows an augmentation of the non-rigid object displaying self-shadowing. Fig. 5.6(b) illustrates that the tracking procedure is robust under self-occlusion.



(a) (b)

Figure 5.6: Flexible augmentations using the coded-ring system and the exact shadows method on paper. a) Self-shadowing; b) Self-occlusion.

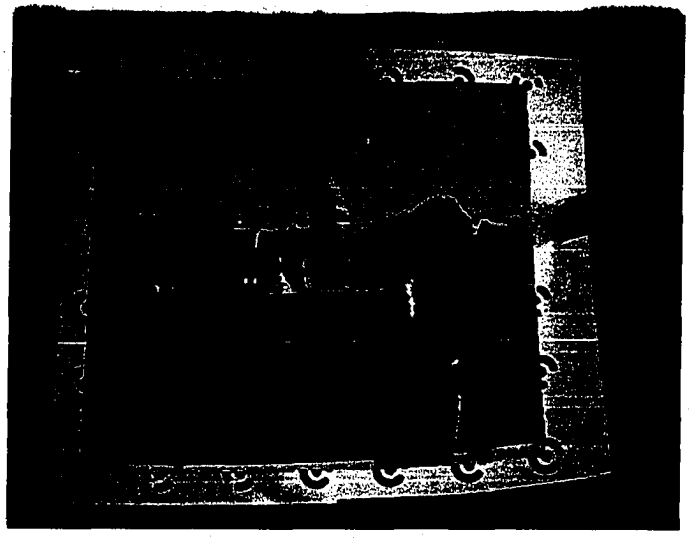
The ability to handle self-occlusions is an important factor when tracking a non-rigid object such as cloth. The coded-ring tracking system does not fail under the occlusion because each target is mapped to one virtual mesh vertex and the mesh is generated dynamically over all of the targets found. Unfortunately, the augmentation itself does not appear self-occluded, as all parts of the virtual texture remain visible. This is evident at the center vertical section of the augmentation that should be occluded, however the virtual texture is only compressed instead.

The exact shadow method of establishing common illumination is further illustrated in Fig. 5.7. Fig. 5.7(a) shows that the shadow of a hand is rendered with the augmentation exactly as it should appear, with the shadow of each individual finger separated. Fig. 5.7(b) demonstrates extreme illumination conditions that are created by dimming the ambient light and shining a flashlight onto the paper. As with the

previous method, the results are impressive, showing the correct contrast between light and dark areas in the augmentation. Again, there are no tracking failures from the poor illumination, due to the adaptive thresholding step in the tracking process. Fig. 5.7(c) shows the exact shadow of a coffee mug on the augmentation, also created with a flashlight.

5.3.2 Cloth Results

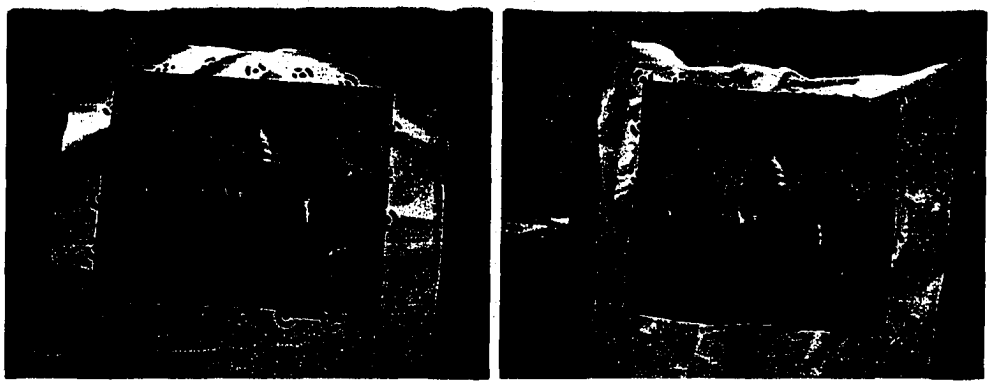
The results of the coded-ring tracking system and the exact illumination method on cloth surpass the results of the previous method on cloth. Most notably, the dynamic triangulation of the located targets to create the virtual mesh plays an important role. This step produces complete augmentations without holes, even if some of the targets are not located, making an observer oblivious to the fact that some inner targets may be lost each frame. The only visual artifacts that are seen in the augmentation occur when border targets are not located, and furthermore these artifacts are negligible unless the lost target is in a corner of the grid. The second factor that propels the results of this method past the previous method is the rendering of the exact illumination on the cloth. The cloth object undergoes a lot of rippling and wrinkling during interaction, and each ripple and wrinkle produces a shadow on the cloth surface. Rendering these shadows along with the augmentation produces very impressive results, as illustrated in Fig. 5.8.



(a)

(b) (c)

Figure 5.7: Exact hard and soft shadow method. a) Shadow of a hand showing each finger individually; b) Extreme illumination conditions; c) Exact shadow of a coffee mug.



(a) (b)

Figure 5.8: Results of the coded-ring system and the exact shadow method on flexible cloth. a) Rippled cloth; b) Wrinkled cloth.

Chapter 6

Conclusions

Augmented reality is an environment where computer generated objects are inserted into the real world. This is accomplished with a real-time video stream or HMD system. AR applications enhance many industries by creating interactive scenes that would not be possible without the addition of virtual entities. Medicine, military, education, tourism and entertainment are just a few industries that benefit from AR research.

One of the main difficulties in augmented reality systems is the problem of registering the virtual parts of the scene with the real world. Typically, this problem is solved by visually tracking physical objects in the scene in order to compute the position and orientation of the camera. This camera pose is then used as a virtual viewpoint to render computer generated objects at desired locations. Object tracking is often accomplished by manually placing rigid planar markers in the scene, which facilitates the computation of the camera pose and provides a three dimensional plane for aligning the virtual objects. Even the markerless tracking methods typically have the restriction that only rigid objects can be located, and they align augmentations to an implicit plane in the scene. Rendering virtual objects that are aligned to rigid objects can be restrictive if an application has a requirement for flexible manipulation. This thesis addresses the problem of rendering two dimensional augmentations on a non-rigid object for interactive augmented reality. We present two complete solutions, based on visually tracking a set of small targets that are placed on the

surface of the object. The target tracking methods are used to acquire a sparse representation of the object from the video image in each frame. The representation takes the form of a virtual mesh, which is rendered with a texture on top of the non-rigid object to provide interactive flexible manipulation. The square target solution makes use of a popular tracking method in the AR community and produces realistic flexible augmentations at interactive frame rates. The coded-ring solution includes a new tracking method to locate circular photogrammetry targets, and produces even more realistic augmentations. The coded-ring method is more robust under flexible movements of the non-rigid object, however it requires more processing time than the square target method. In addition to the non-rigid object tracking methods, this thesis includes two novel techniques to establish common illumination between the real and virtual scene objects. Common illumination is very important in increasing the realism of an augmentation. This is even more evident in the case of augmenting non-rigid objects, as flexible objects tend to cast dynamic self-shadows in addition to the shadows of other real objects. In this case, common illumination is established by rendering these shadows with the augmented mesh. The illumination methods presented are vision-based, requiring only the grayscale image of the camera frame. The methods exploit the fact that the non-rigid object is white in color and it therefore exhibits the exact shadows that should be included with the augmentation. Our first solution is demonstrated with the square target tracking system. We approximate the soft shadows in the scene by sampling the non-rigid object at a fixed number of locations and interpolate the illumination over the virtual mesh. Our second solution is demonstrated with the coded-ring tracking system. We acquire the exact hard and soft shadows that are displayed on the non-rigid object in the form of a shadow map, and then blend the illumination with the mesh texture. The results of this thesis are illustrated by rendering real-time augmentations on flexible paper and cloth in an interactive application.

The ability to render realistic augmentations on cloth leads to some interesting interactive applications. The texture of clothing can be virtually modified in a mirror-based system, allowing users to select different outfits in a virtual fashion show. A person would then be "trying on" multiple articles without ever changing their

physical clothing. A related application would be to interactively demonstrate and select t-shirt logos, prior to printing or ironing the logo onto a shirt. Museums can provide interactive AR cloth displays. Curtain colors and patterns can be virtually matched to surrounding designs in home decor. Medical augmentations of internal structures can be performed directly onto a human body. These applications are just some of the examples of how this thesis research can benefit multiple industries.

The future work for this research consists mainly of increasing the speed and robustness of the coded-ring tracking method. Alternatively, new tracking systems can be explored. Perhaps a markerless tracking method that locates a textured cloth surface using feature points would be applicable, if enough feature points could be determined in real-time. Another area of future research is dealing with self-occlusions. The coded-ring system successfully tracks a non-rigid object in the presence of self-occlusions, however the augmentation is not rendered to correctly display the occlusion. This feature would be beneficial to increase the realism of the augmentation. In addition, a future study should be performed to determine issues of social acceptance of augmentations on clothing and other non-rigid objects.

In summary, this research demonstrates impressive results of performing flexible augmentations on non-rigid objects while incorporating the real world illumination environment. To our knowledge, this is the first research in this particular area of augmented reality.

Bibliography

[1] ARToolKit. http://www.hitl.washington.edu/artoolkit.

[2] Keith Atkinson, editor. *Close Range Photogrammetry and Machine Vision.* Engineering and Science. Whittles Publishing, Scotland, UK, 1996. 12 chapters, 371 pages.

[3] Ronald Azuma, Yohan Baillot, Reinhold Behringer, Steven Feiner, Simon Julier, and Blair MacIntyre. Recent advances in augmented reality. *IEEE Computer Graphics and Applications*, 21(6):34–47, 2001.

[4] Coloma Ballester, Marcelo Bertalmio, Vincent Caselles, Guillermo Sapiro, and Joan Verdera. Filling-in by joint interpolation of vector fields and gray levels. *IEEE Transactions on Image Processing*, 10(8):1200–1211, Aug 2001.

[5] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 417–424. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[6] Mark Billinghurst, Hirkazu Kato, and Ivan Poupyrev. The magicbook - moving seamlessly between reality and virtuality. *IEEE Comput. Graph. Appl.*, 21(3):6–8, 2001.

[7] Derek Bradley. Memory efficient voronoi code. www.derekbradley.ca/voronoi.

[8] Gary Bradski. The OpenCV library. *Dr. Dobb's Journal of Software Tools*, 25(11):120, 122–125, nov 2000.

[9] Christoph Bregler, Aaron Hertzmann, and Henning Biermann. Recovering non-rigid 3d shape from image streams. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2000)*, pages II:690–696, 2000.

[10] Rodrigo L. Carceroni and Kiriakos N. Kutulakos. Multi-view scene capture by surfel sampling: From video streams to non-rigid 3d motion, shape & reflectance. In *International Conference on Computer Vision (ICCV)*, pages 60–67, 2001.

[11] Çigdem Eroglu Erdem, Bülent Sankur, and A. Murat Tekalp. Non-rigid object tracking using performance evaluation measures as feedback. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, pages II:323–330, 2001.

[12] Feng Cen and Feihu Qi. Tracking non-rigid objects in clutter background with geometric active contours. *Electronics Letters*, 38(12):550–551, June 2002.

[13] Francis Huo Yen Chan, F. K. Lam, and Hui Zhu. Adaptive thresholding by variational method. *IEEE Transactions on Image Processing*, 7(3):468–473, March 1998.

[14] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Real-time tracking of non-rigid objects using mean shift. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2000)*, pages II:142–149, 2000.

[15] Andrew I. Comport, Eric Marchand, and Francois Chaumette. A real-time tracker for markerless augmented reality. In *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2003)*, pages 36–45, Tokyo, Japan, October 2003.

[16] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001.

[17] Franklin C. Crow. Summed-area tables for texture mapping. *SIGGRAPH Comput. Graph.*, 18(3):207–212, 1984.

[18] Mark de Berg, Mark van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications.* Springer-Verlag, 2000. 2nd Ed.

[19] Paul Debevec. Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 189–198. ACM Press, 1998.

[20] Boris Delone. Sur la sphère vide. *Bulletin de L'Academie des Sciences de L'URSS, Classe des Sciences mathématiques et Naturelles*, 7(6):793–800, 1934.

[21] Klaus Dorfmüller-Ulhaas and Dieter Schmalstieg. Finger tracking for interaction in augmented environments. In *Proceedings of the IEEE and ACM International Symposium on Augmented Reality (ISAR'01)*, pages 55–64. IEEE Computer Society, 2001.

[22] George Drettakis, Luc Robert, and Sylvain Bougnoux. Interactive common illumination for computer augmented reality. In *Proceedings of the Eurographics Workshop on Rendering Techniques '97*, pages 45–56. Springer-Verlag, 1997.

[23] Wei Feng and Rong-Chun Zhao. Non-rigid objects detection and segmentation in video sequence using 3d mean shift analysis. In *International Conference on Machine Learning and Cybernetics*, pages 3134–3139, 2003.

[24] Morten Fjeld and Benedikt M. Voegtli. Augmented chemistry: An interactive educational workbench. In *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2002)*, pages 259–260, September 2002. Darmstadt, Germany.

[25] Steven Fortune. A sweepline algorithm for voronoi diagrams. *Algorithmica*, 2:153–174, 1987.

[26] Simon Gibson, Jon Cook, Toby Howard, and Roger Hubbold. Rapid shadow generation in real-world lighting environments. In *Proceedings of the 14th Eurographics workshop on Rendering*, pages 219–229. Eurographics Association, 2003.

[27] Simon Gibson and Alan Murta. Interactive rendering with real-world illumination. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, pages 365–376. Springer-Verlag, 2000.

[28] Henri Gouraud. Continuous shading of curved surfaces. *IEEE Transactions on Computers*, 20(6):623–628, 1971.

[29] Igor Guskov. Efficient tracking of regular patterns on non-rigid geometry. In *Proceedings of ICPR*, pages 1057–1060, 2002.

[30] Igor Guskov and Ben Bryant. Real-time tracking of quad-marked surfaces. In *IEEE Workshop on Motion and Video Computing*, 2002.

[31] Igor Guskov, Sergey Klibanov, and Benjamin Bryant. Trackable surfaces. In *ACM/EG Symposium on Computer Animation*, 2003.

[32] Igor Guskov and Leonid Zhukov. Direct pattern tracking on flexible geometry. In *Winter School of Computer Graphics*, 2002.

[33] Michael Haller, Stephan Drab, and Werner Hartmann. A real-time shadow approach for an augmented reality application using shadow volumes. In *ACM Symposium on Virtual Reality Software and Technology (VRST 2003)*, pages 56–65, October 2003.

[34] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521623049, 2000.

[35] Bernd Heisele, Ulrich KreBel, and Werner Ritter. Tracking non-rigid, moving objects based on color cluster flow. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 1997)*, pages 257–260, 1997.

[36] Daniel P. Huttenlocher, Jae J. Noh, and William J. Rucklidge. Tracking non-rigid objects in complex scenes. In *International Conference on Computer Vision (ICCV)*, pages 93–101, 1993.

[37] Gaël Jaffré and Alain Crouzil. Non-rigid object localization from color model using mean shift. In *International Conference on Image Processing*, pages III:317–320, 2003.

[38] Tony Jebara, Cyrus Eyster, Josh Weaver, Thad Starner, and Alex Pentland. Stochasticks: Augmenting the billiards experience with probabilistic vision and wearable computers. In *ISWC '97: Proceedings of the 1st IEEE International Symposium on Wearable Computers*, pages 138–145. IEEE Computer Society, 1997.

[39] Gun A. Lee, Mark Billinghurst, and Gerard Jounghyun Kim. Occlusion based interaction methods for tangible augmented reality environments. In *ACM Siggraph International Conference on Virtual Reality Continuum and its Applications in Industry (VRCAI 2004)*, pages 419–426, June 2004. Singapore.

[40] Céline Loscos, George Drettakis, and Luc Robert. Interactive virtual relighting of real scenes. *IEEE Transactions on Visualization and Computer Graphics*, 6(4):289–305, 2000.

[41] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[42] Claus B. Madsen, Mads K. D. Sørensen, and Michael Vittrup. The importance of shadows in augmented reality. In *Proceedings: 6th Annual International Workshop on Presence, Aalborg, Denmark*, October 2003.

[43] Shahzad Malik, Chris McDonald, and Gerhard Roth. Hand tracking for interactive pattern-based augmented reality. In *IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2002)*, pages 399–406, September 2002. Darmstadt, Germany.

[44] Shahzad Malik, Gerhard Roth, and Chris McDonald. Robust 2d tracking for real-time augmented reality. In *15th International Conference on Vision Interface*, May 2002. Calgary, Canada.

[45] Siddharth Manay and Anthony Yezzi. Anti-geometric diffusion for adaptive thresholding and fast segmentation. *IEEE Transactions on Image Processing*, 12(11):1310–1323, November 2003.

[46] Lucio Marcenaro, Carlo S. Regazzoni, and Gianni Vernazza. Automatic generation of the statistical model of a non-rigid object in a multiple-camera environment. In *International Conference on Pattern Recognition*, volume 1, pages 530–533, 2000.

[47] Simon Masnou and Jean-Michel Morel. Level lines based disocclusion. In *Proc. 1998 IEEE International Conference on Image Processing*, volume 3, pages 259–263, 1998.

[48] Paul Milgram and Fumio Kishino. A taxonomy of mixed reality visual displays. *IEICE Transactions on Information Systems*, E77-D(12), Dec. 1994.

[49] Takeshi Naemura, Takuya Nitta, Atsushi Mimura, and Hiroshi Harashima. Virtual shadows in mixed reality environment using flashlight-like devices. *Trans. Virtual Reality Society of Japan*, 7(2):227–237, 2002.

[50] Franco Oberti and Carlo Regazzoni. Adaptive tracking of multiple non rigid objects in cluttered scenes. In *International Conference on Pattern Recognition*, volume 3, pages 1096–1099, 2000.

[51] Atsuyuki Okabe, Barry Boots, Kokichi Sugihara, and Sung Nok Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, 2nd Edition.* John Wiley and Sons, 2000.

[52] Nobuhiro Okada and Martial Hebert. Fast 3d tracking of non-rigid objects. In *IEEE International Conference on Robotics and Automation*, pages 3497–3503, 2003.

[53] Charles B. Owen, Fan Xiao, and Paul Middlin. What is the best fiducial? In *IEEE International Augmented Reality Toolkit Workshop*, pages 98–105, 2002.

[54] Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, and Markus Gross. Surfels: Surface elements as rendering primitives. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 335–342. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[55] Photomodeler. http://www.photomodeler.com.

[56] David Pritchard and Wolfgang Heidrich. Cloth motion capture. *Computer Graphics Forum (Eurographics 2003)*, 22(3):263–271, Sep 2003.

[57] Ruan Qiuqi and Guan Haiying. New adaptive tracking algorithm of non-rigid objects. In *International Conference on Signal Processing*, volume 2, pages 940–943, 1998.

[58] Guillermo Sapiro. Image inpainting. *SIAM News*, 35(4), May 2002.

[59] Gilles Simon, Andrew W. Fitzgibbon, and Andrew Zisserman. Markerless tracking using planar structures in the scene. In *Proc. International Symposium on Augmented Reality*, pages 120–128, October 2000.

[60] Chester C. Slama, editor. *Manual of Photogrammetry.* American Society of Photogrammetry, Falls Church, VA, 1984. 4th Ed.

[61] Jurgen Stauder. Augmented reality with automatic illumination control incorporating ellipsoidal models. *IEEE Transactions on Multimedia*, 1(2):136–143, 1999.

[62] Joo Kooi Tan and Seiji Ishikawa. Shape recovery of non-rigid objects employing factorization-based stereo cameras. In *International Conference on Image Processing*, pages 145–148, 2001.

[63] Bruce Thomas, Ben Close, John Donoghue, John Squires, Phillip de Bondi, Michael Morris, and Wayne Piekarski. Arquake: An outdoor/indoor augmented reality first person application. In *ISWC '00: Proceedings of the 4th IEEE International Symposium on Wearable Computers*, pages 139–146. IEEE Computer Society, 2000.

[64] S. Tiwari and P. Bhattacharya. The recovery of non-rigid motion from stereo images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 1993)*, pages 758–759, 1993.

[65] Lorenzo Torresani, Danny B. Yang, Eugene J. Alexander, and Christoph Bregler. Tracking and modeling non-rigid objects with rank constraints. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, pages I:493–500, 2001.

[66] Emanuele Trucco and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision.* Prentice Hall PTR, 1998.

[67] Olga Veksler. Fast variable window for stereo correspondence using integral images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2003)*, pages 556–661, Madison, Wisconsin, June 2003.

[68] Paul Viola and Michael J. Jones. Robust real-time face detection. *Int. J. Comput. Vision*, 57(2):137–154, 2004.

[69] Mark Allen Weiss. *Data structures and algorithm analysis.* Benjamin-Cummings Publishing Co., Inc., 1992.

[70] Pierre D. Wellner. Adaptive thresholding for the digitaldesk. Technical Report EPC-93-110, EuroPARC, 1993.

[71] Charles Woodward, Petri Honkamaa, Jani Jäppinen, and Esa-Pekka Pyökkimies. Camball - augmented networked table tennis played with real rackets. In *ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE 2004)*, 2004.

[72] Ke Xu, Adrian David Cheok, Kar Wee Chia, and Simon Prince. Visual registration for geographical labeling in wearable computing. In *International Symposium on Wearable Computing*, pages 109–116, 2002.

[73] Xiao-Ping Zhang and Mita D. Desai. Segmentation of bright targets using wavelets and adaptive thresholding. *IEEE Transactions on Image Processing*, 10(7):1020–1030, July 2001.

Appendix A

Homographic Transformations

This thesis uses the mathematical concept of a homography. Homographic transformations are often used in augmented reality systems when tracking four-sided planar targets.

Consider the AR environment illustrated in Fig. A.1. The target to be tracked lies on a 2D plane in space, and similarly the image plane of the camera is a 2D plane in space. In order to identify a target in an image, we must sample the target at specific locations (in target space), however we have only the image and the tracked pixel locations of the corners of the target. We wish to define a transformation that allows us to compute the pixel location on the image plane for any given location on the target plane. Assume that the target lies on the plane $Z = 0$ in target space, centered at the origin. The projection of the target onto the image plane can be represented using a standard perspective projection matrix [66], taking into account the parameters of the camera:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix}$$

where p_{ij} are the elements of the perspective projection matrix, and $(x, y, 0)$ is a point on the target. Simplifying, we get:

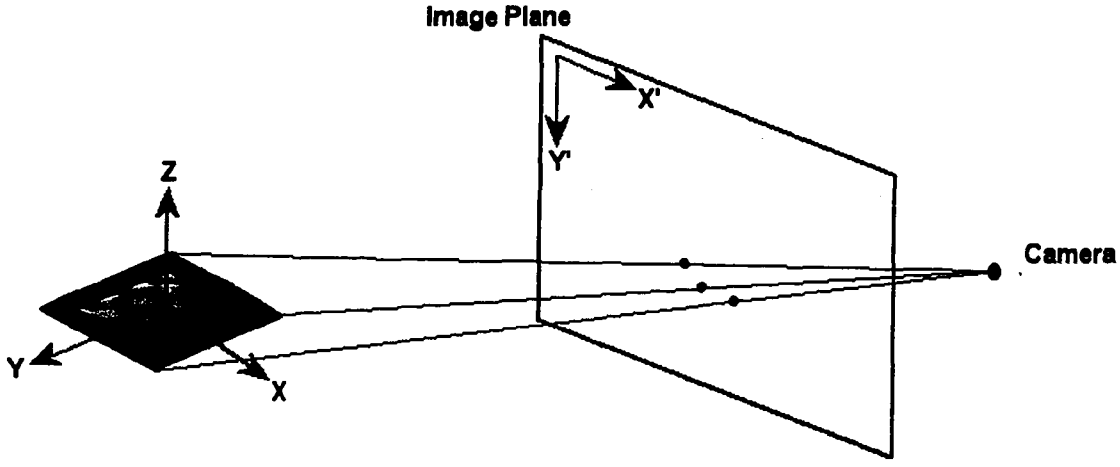


Figure A.1: AR environment using planar patterns.

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{14} \\ p_{21} & p_{22} & p_{24} \\ p_{31} & p_{32} & p_{34} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

where H is the homography [34]. If we now consider the image plane coordinates (x', y'), we have:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \lambda H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

where λ is a scale factor. This allows us to compute the image plane pixel location for any point on the target plane, if we know the homography H. We now focus on computing the homography.

The equation to compute (x', y') can be rewritten as:

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}$$

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

where h_{ij} define the elements of H. This leads to two linear equations:

$$x'(h_{31}x + h_{32}y + h_{33}) = h_{11}x + h_{12}y + h_{13}$$

$$y'(h_{31}x + h_{32}y + h_{33}) = h_{21}x + h_{22}y + h_{23}.$$

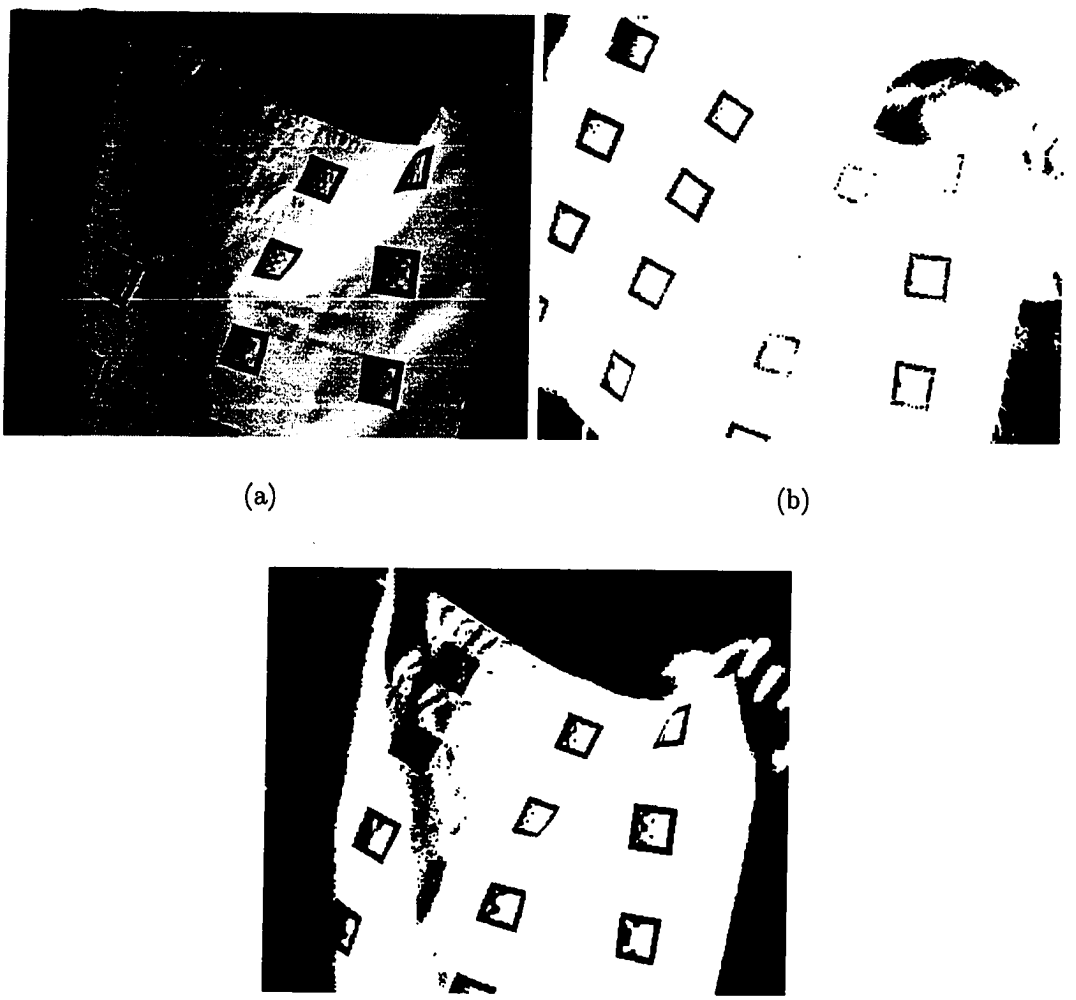
Now, if we know four non-collinear point correspondences from target space (x', y') to image space (x, y) then we can solve for all the elements of the homography. Fortunately, for square target tracking we do have the pixel locations of the four corners of the target, and hence the homography can be computed.

Appendix B

Adaptive Thresholding

One of the first steps in two of the tracking methods described in this thesis is to binarize an input image by thresholding the corresponding grayscale image. This can be accomplished by choosing a global threshold value, T, and then setting all pixels in the intensity image that are greater than T to 1 and all pixels that are less than T to 0. This is known as *global thresholding*. However, global thresholding can be problematic in many computer vision applications, especially under variable or extreme lighting conditions. For example, consider Fig. B.1. The input image (Fig. B.1(a)) shows a particular illumination condition where part of the cloth to be tracked is brightly lit and another part is dimly lit. Using global thresholding, the darker markers can be segmented correctly with a threshold value of $T_1 = 54$, however the brighter markers are lost completely (Fig. B.1(b)). Similarly, the brighter markers can be correctly segmented using a threshold value of $T_2 = 134$, however the darker markers are then lost (Fig. B.1(c)). This illustrates the need for a technique with better performance than global thresholding. The answer is *adaptive thresholding*.

Adaptive thresholding is similar to global thresholding except that the threshold value changes dynamically over the image. This method can accommodate variable lighting conditions such as strong illumination gradients or shadows. A number of different techniques exist to perform adaptive thresholding [13, 45, 70, 73]. In our system, we extend the method of Wellner [70]. In Wellner's algorithm, a moving average of the last s pixels is calculated while traversing the image. If the value of



(a) (b)

(c)

Figure B.1: Global thresholding. a) Input image; b) Threshold for dimly lit markers ($T_1 = 54$); c) Threshold for brightly lit markers ($T_2 = 134$).

the current pixel is t percent lower than the average then it is set to black, otherwise it is set to white. The advantage of this method is that only a single pass through the image is required. Wellner uses 1/8th of the image width for the value of s and 15 for the value of t. However, this technique yields different results depending on whether the image is scanned from left to right, from right to left or alternating from the left and from the right, often resulting in unwanted artifacts. Taking the previous line of approximate averages into account when processing each scan line will help remove the artifacts without requiring additional passes through the image. However the process becomes more complex and different outcomes will still be seen depending on whether the image is scanned from the top to the bottom or vice versa. We extend this solution to a very simple approach that produces the same output independently of how the image is processed, and we sacrifice only one additional pass through the image. Instead of computing a running average of the last s pixels seen, we pre-compute the average of an s x s window of pixels around each pixel, which is equivalent to blurring the input image. This average computation is accomplished in linear time by using the *integral image* of the input [17, 67, 68]. The integral image technique can be used whenever we have a function from pixels to real numbers $f(x, y)$ (for instance, pixel intensity), and we wish to compute the sum of this function over some rectangular region of the image. Without an integral image, this is accomplished in linear time per rectangle by calculating the value of the function for each pixel individually. However, if we need to compute the sum over multiple different rectangular windows, we can use an integral image and achieve a constant number of operations per rectangle with only a linear amount of pre-processing to compute the integral image. To compute the integral image, we store at each location, $I(x, y)$, the sum of all $f(x, y)$ terms to the left and above the pixel (x, y). Fig. B.2(a) illustrates the computation of the integral image. The integral image is computed in linear time using the following equation for each pixel, taking into account the image border cases;

$$I(x, y) = f(x, y) + I(x - 1, y) + I(x, y - 1) - I(x - 1, y - 1).$$

Once we have the integral image, the sum of the function for any arbitrary rectangle (x_1, y_1) to (x_2, y_2) is computed in constant time using the following equation,

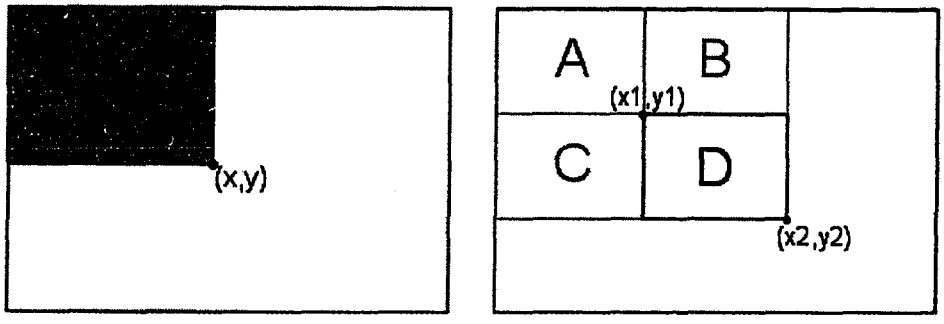


Figure B.2: Integral image. a) Computation; b) Usage to calculate sums.

again taking into account the image border cases;

$$\sum_{x=x1}^{x2}\sum_{y=y1}^{y2} f(x,y) = I(x_2,y_2) - I(x_2,y_1-1) - I(x_1-1,y_2) + I(x_1-1,y_1-1).$$

Fig. B.2(b) illustrates that the above equation to compute the sum of $f(x,y)$ over the rectangle D is equivalent to computing the sums over the rectangles (A+B+C+D)-(A+B)-(A+C)+A.

During the adaptive thresholding algorithm, the first pass through the input image calculates the integral image. During the second pass, the s x s average around each pixel is calculated in constant time using the integral image. If the value of the current pixel is t percent less than this average it is set to black, otherwise it is set to white. This results in a two-pass adaptive thresholding algorithm that is simple and independent of the scanning order. Fig. B.3 illustrates the result of our adaptive thresholding method on the input image from Fig. B.1.

Figure B.3: Adaptive thresholding result for input image of Fig. B.1.