Practical Color-Based Motion Capture

by

Robert Yuanbo Wang

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Computer Science and Engineering
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
February 2011

© Massachusetts Institute of Technology 2011. All rights reserved.
Practical Color-Based Motion Capture

by

Robert Yuanbo Wang

Submitted to the Department of Electrical Engineering and Computer Science
on January 7th, 2011, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science and Engineering

Abstract

Motion capture systems track the 3-D pose of the human body and are widely used for high quality content creation, gestural user input and virtual reality. However, these systems are rarely deployed in consumer applications due to their price and complexity. In this thesis, we propose a motion capture system built from commodity components that can be deployed in a matter of minutes. Our approach uses one or more webcams and a color garment to track either the user’s upper body or hands for motion capture and user input.

We demonstrate that custom designed color garments can simplify difficult computer vision problems and lead to efficient and robust algorithms for hand and upper body tracking. Specifically, our highly descriptive color patterns alleviate ambiguities that are commonly encountered when tracking only silhouettes or edges, allowing us to employ a nearest-neighbor approach to track either the hands or the upper body at interactive rates. We also describe a robust color calibration system that enables our color-based tracking to work against cluttered backgrounds and under multiple illuminants.

We demonstrate our system in several real-world indoor and outdoor settings and describe proof-of-concept applications enabled by our system that we hope will provide a foundation for new interactions in computer aided design, animation control and augmented reality.

Thesis Supervisor: Jovan Popović
Title: Associate Professor
Acknowledgments

Thanks goes to Jiawen Chen for his generosity and technical advice, especially on software tools and development; Yeuhi Abe, Ilya Baran and Marco da Silva for being a reliable sounding board on research ideas and techniques; and to all the members of the MIT Computer Graphics Lab for their encouragement and discussions throughout this process.

Special thanks goes to Sylvain Paris for being an excellent collaborator and sage researcher who never stopped nagging me to go the extra mile; and my advisor Jovan Popović for being a wise, patient and well-balanced advisor.
# Contents

1 Introduction 19  
1.1 Hand Tracking 20  
1.2 Upper Body Tracking 21  
1.3 Dissertation Outline 23  

2 Related work 25  
2.1 Hand Tracking 25  
2.1.1 Marker-Based Motion Capture 25  
2.1.2 Instrumented Gloves 25  
2.1.3 Color-Marker-Based Motion Capture 26  
2.1.4 Bare-Hand Tracking 26  
2.1.5 Depth Sensing Cameras and Shadows 26  
2.2 Body-Tracking 27  
2.2.1 Markerless Full-Body Motion Capture 27  
2.2.2 Data-Driven Pose Estimation 29  
2.3 Deformable Surface Tracking 29  
2.4 Color-Based Image Analysis 29  

3 Garment Design for Color-Based Tracking 31  
3.1 Color Selection 32  
3.2 Pattern Layout 34
### CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3 Color Assignment</td>
<td>35</td>
</tr>
<tr>
<td>3.4 Discussion</td>
<td>38</td>
</tr>
</tbody>
</table>

#### 4 Database-Driven Pose Estimation

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Database Construction</td>
<td>42</td>
</tr>
<tr>
<td>4.1.1 Database Sampling</td>
<td>43</td>
</tr>
<tr>
<td>4.1.2 Image Normalization</td>
<td>44</td>
</tr>
<tr>
<td>4.1.3 Tiny Image Distance</td>
<td>44</td>
</tr>
<tr>
<td>4.1.4 Blending Nearest Neighbors</td>
<td>46</td>
</tr>
<tr>
<td>4.2 Fast Lookup Using Boosting</td>
<td>47</td>
</tr>
<tr>
<td>4.3 Inverse Kinematics</td>
<td>48</td>
</tr>
<tr>
<td>4.3.1 Temporal Smoothness</td>
<td>50</td>
</tr>
<tr>
<td>4.4 Discussion</td>
<td>51</td>
</tr>
</tbody>
</table>

#### 5 Robust Color and Pose Estimation

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Histogram-Based Localization</td>
<td>53</td>
</tr>
<tr>
<td>5.2 Color Model and Initialization</td>
<td>55</td>
</tr>
<tr>
<td>5.3 Online Analysis</td>
<td>58</td>
</tr>
<tr>
<td>5.3.1 Step 1: Color classification</td>
<td>59</td>
</tr>
<tr>
<td>5.3.2 Step 2: Pose estimation</td>
<td>60</td>
</tr>
<tr>
<td>5.3.3 Step 3: Color classification refinement</td>
<td>61</td>
</tr>
<tr>
<td>5.3.4 Step 4: Pose refinement with inverse kinematics</td>
<td>62</td>
</tr>
<tr>
<td>5.3.5 Step 5: Estimating the white balance for the next frame</td>
<td>63</td>
</tr>
<tr>
<td>5.4 Discussion</td>
<td>64</td>
</tr>
</tbody>
</table>

#### 6 Results

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1 Experimental Setup</td>
<td>66</td>
</tr>
<tr>
<td>6.1.1 Hand Tracking</td>
<td>66</td>
</tr>
<tr>
<td>6.1.2 Upper Body Tracking</td>
<td>68</td>
</tr>
<tr>
<td>6.2 Hand Tracking Results</td>
<td>69</td>
</tr>
<tr>
<td>6.3 Body-Tracking Vicon Validation</td>
<td>71</td>
</tr>
</tbody>
</table>
CONTENTS

6.4 Discussion ................................................................. 74

7 Applications ................................................................. 79
  7.1 Applications for Hand Tracking ..................................... 79
    7.1.1 Driving an Animated Character ................................. 79
    7.1.2 Manipulating Rigid Bodies with Physics ...................... 80
    7.1.3 Sign Language Finger Spelling ................................ 81
    7.1.4 Flight Simulator .................................................. 81
    7.1.5 Trackball Controller ............................................. 81
  7.2 Upper-Body Tracking for Sports Analysis and Gaming ............ 84

8 Conclusion ................................................................. 87
  8.1 Contributions .......................................................... 87
  8.2 Future Directions ..................................................... 88
    8.2.1 Bare-Hand Tracking .............................................. 88
    8.2.2 Usability Optimization ........................................... 89
    8.2.3 Computer Aided Design .......................................... 89
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>We describe a system that can reconstruct the pose of the hand from a single image of the hand wearing a multi-colored glove. We demonstrate our system as a user-input device for desktop virtual reality applications.</td>
</tr>
<tr>
<td>1-2</td>
<td>We describe a lightweight color-based motion capture system that uses one or two commodity webcams and a color shirt to track the upper body. Our system can be used in a variety of natural lighting environments such as this squash court, a basketball court or outdoors.</td>
</tr>
<tr>
<td>3-1</td>
<td>Our glove design consisting of large color patches accounts for camera limitations, self-occlusion, and algorithm performance. The length of our glove is 24 cm.</td>
</tr>
<tr>
<td>3-2</td>
<td>We print a 6 x 6 grid of saturated colors for our calibration pattern. We then observe the pattern held at various angles from the camera.</td>
</tr>
<tr>
<td>3-3</td>
<td>We experimented with various patch sizes and shapes.</td>
</tr>
<tr>
<td>3-4</td>
<td>The palm down and palm up poses map to similar images for a bare hand. These same poses map to very different images for a gloved hand.</td>
</tr>
<tr>
<td>3-5</td>
<td>We measured retrieval accuracy as a function of the number of patches on a glove. Performance peaks at around 25 patches after which there are no benefits to using a higher patch count.</td>
</tr>
</tbody>
</table>
3-6 As we try more random assignments, the best assignment score \( S(\alpha) \) quickly converges to a value close to \( \sigma_{\text{max}} = 5 \). ...

3-7 Our glove design. Note that the finger tips, which can reach any part of the palm, are seldom assigned the same colors as the palm, since the fingers often bend towards the palm. On the other hand, the finger tips are often assigned the same colors as patches on the back of the hand since the fingers can never bend backwards. ...

3-8 Our shirt design. Note that the location of the wrists, the most kinematically flexible part of the body, are assigned the same colors as their opposite shoulders. ...

4-1 Pose database. We sample a large set of hand poses which are indexed by their rasterized tiny images. ...

4-2 Our pose estimation process. The camera input image is transformed into a normalized tiny image. We use the tiny image as the query for a nearest neighbor search of our pose database. The pose corresponding to the nearest database match is retrieved. ...

4-3 We denoise the input image before using a mixture-of-Gaussians color classifier to segment the hand. Our normalization step consists of cropping and rescaling the hand region into a \( 40 \times 40 \) tiny image. ...

4-4 Hausdorff-like image distance. A database image and a query image are compared by computing the divergence from the database to the query and from the query to the database. We then take the average of the two divergences to generate a symmetric distance. ...

4-5 Database coverage evaluation (log scale). As the database size increases, pose estimation becomes more accurate. ...

4-6 The rank of the true nearest neighbor (log scale) according to the Hamming distance approximation decreases quickly with longer (more descriptive) codes. ...
4-7 Correspondences for inverse kinematics. We compute centroids of each patch in the query image and the nearest neighbor pose. We establish correspondences between the two sets of points and use IK to penalize differences between the two. .......................... 49

5-1 The measured values of the color patches can shift considerably from frame to frame. Each row shows the measured value of two identically colored patches in two frames from the same capture session. .......................... 54

5-2 The colorful shirt has a more uniform chromaticity (2-D normalized RGB) histogram (b) with many non-zero entries whereas most other regions have a peakier histogram (a) dominated by one or two colors. We visualize our uniformity measure $u(h(x, y, s)) = \sum_i \min(h_i, 0.1)$ with scale $s = 80$ as a heat map. .......................... 55

5-3 We show several frames of a hand tracking sequence with their uniformity heat maps. Notice that the system is not confused by the colorful mousepad. Only a Rubik's cube comes close to approaching the color uniformity of the glove. .......................... 56

5-4 Ahead of time, we build a color model of our shirt by scribbling on 5 white-balanced images. We model each color with a Gaussian distribution in RGB space. .......................... 57

5-5 We crop and compute the normalized RGB histogram of the shirt region. The peaks of the NRGB histogram are distinctive and easy to extract. We brute force align the observed peaks (squares) with the NRGB coordinates of the color patches of our Gaussian color model (circles). This gives us a coarse initial classification of the image, which we use to bootstrap our white balance estimate to produce the final classification. .......................... 59
5-6 After cropping the image to the shirt region, we white balance and classify the image colors. The classified image is used to estimate the upper-body pose by querying a precomputed pose database. We take the pose estimate to be a weighted blend of these nearest neighbors in the database. The estimated pose can be used to refine our color classification, which is converted into a set of patch centroids. These centroids drive the inverse kinematics (IK) process to refine our pose. Lastly, the final pose is used to estimate the white balance matrix for the next frame.

5-7 Our pose-assisted classifier classifies more correct pixels at a lower false-positive rate than the baseline Gaussian classifier discussed in Step 1.

5-8 For each camera, we compute the centroids of the color-classified id map $\hat{r}$ and correspond them to centroids of the blended nearest neighbor to establish inverse kinematics constraints.

6-1 In addition to color calibration, we calibrate the desk plane by asking the user to click on four points of an 8.5 × 11 inch sheet of letter paper. We also calibrate the scale of the hand by asking the user to hold his hand flat against the known desk plane.

6-2 We tested several subjects performing the same sequence of motions. The image error of the tracked sequences varied by less than 5% across the subjects.

6-3 Representative frames from the rigid hand tracking sequence. Our prediction is superimposed on the input image.

6-4 Representative frames from the free jiggling hand tracking sequence.

6-5 Representative frames from the alphabet hand tracking sequence.

6-6 Representative frames from the medley hand tracking sequence.

6-7 Reprojection error and jitter on several tracking sequences (lower is better). Inverse kinematics reduces the reprojection error of the scene while temporal filtering reduces the jitter without sacrificing accuracy.
6-8 Even when our estimate closely matches the input frame (left), monocular depth ambiguities remain a problem, as shown from a different camera perspective (right).

6-9 We demonstrate motion capture in a basketball court, inside and outside of a squash court, at the office, outdoors and while using another (Vicon) motion capture system. The skeleton overlay is more visible in the accompanying video.

6-10 The pose prior and white balancing described in Chapter 5 have a significant effect in natural scenes with mixed lighting, like this basketball court. Here, we show the color classification mistakes that arise when we drop the pose prior, the white balance, and both.

6-11 We show one frame of the validation sequence, where we have ground truth from a Vicon motion capture system. The pose prior and white balance steps produce a better color classification and pose estimate.

6-12 We compare the accuracy (RMS of all mesh vertices) between a simple system without pose prior nor adaptive white balance akin to [68] and our approach. In absence of occlusion, both methods perform equivalently but on more complex sequences with faster motion and occlusions, our approach can be nearly twice more precise. On average, our method performs 20% better.

6-13 We plot the angle of a shoulder joint for ground-truth data captured with a Vicon system, our method, and our method without the pose prior or white balance steps (akin to [68]). Our results are globally closer to the ground truth and less jittery than those of the simple method. This is better seen in the companion video.

6-14 Despite the significant motion blur in this frame, we are still able to estimate the patch centroids and the upper-body pose.

7-1 We demonstrate animation control (a) and a sign language finger spelling application with our hand tracking interface.
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-2</td>
<td>We demonstrate a virtual yoke controller for a flight simulator.</td>
<td>82</td>
</tr>
<tr>
<td>7-3</td>
<td>We provide a trackball-like interface for the user to rotate and scale a 3-D model.</td>
<td>83</td>
</tr>
<tr>
<td>7-4</td>
<td>We combine motion with clicking to define rotation, scaling, and translation.</td>
<td>83</td>
</tr>
<tr>
<td>7-5</td>
<td>We show two frames of a simple goalkeeper game where the player controls an avatar to block balls. We show the view from each camera alongside the game visualization as seen by the player.</td>
<td>84</td>
</tr>
<tr>
<td>7-6</td>
<td>We demonstrate a motion analytics tool for squash that tracks the speed and acceleration of the right wrist while showing the player from two perspectives different from those captured by the cameras.</td>
<td>85</td>
</tr>
<tr>
<td>8-1</td>
<td>An early prototype of a system that incorporates several ideas for future direction of our work. We use a wide baseline stereo camera system to obtain two views of each hand. The working area of this system is relatively small and the arms are supported at all times. We intend to apply this system to 3-D manipulation tasks for CAD.</td>
<td>90</td>
</tr>
</tbody>
</table>
List of Tables
Chapter 1

Introduction

Over the past two decades, a revolution in computer graphics output has brought us photorealistic special effects and computer-generated animated films. Today, immersive video games are powered by dedicated rendering hardware in every modern PC or console system. A similar revolution is now occurring in graphics input. We have seen a spate of new input devices ranging from inertial motion controllers to depth sensing cameras. Within the next two years, these new input devices will be integrated in every major video game console system.

In this thesis, we propose another type of input device, a color-garment-based tracking system that uses commodity cameras to capture the motions of the hands or the upper body. Our system is designed to be higher fidelity than the next generation of gaming input devices but retain the properties of easy setup and robust input. In addition to gaming applications, our system is suitable for virtual and augmented reality.

Our proposed system is particularly low cost and easy to deploy—requiring only commodity cameras and a garment imprinted with a special color pattern. Because our garments do not embed sensors or use exoskeletons, they are easy to put on and comfortable to wear. We also use only one or two webcams to simplify calibration, minimize setup time, and reduce cost. Our algorithms are designed for real-world environments, explicitly addressing mixed lighting, motion blur and cluttered backgrounds. The basis of our technique is a data-driven pose estimation algorithm that
can determine pose from a single image and an inverse-kinematics pose refinement algorithm that can improve a coarse pose estimate. For a cluttered environment, we describe a robust histogram-based localization algorithm for detecting colorful objects. For environments with mixed or varying illumination, we also present an alternating color and pose estimation algorithm designed. In summary, this thesis proposes a practical and robust color-based motion capture system.

We demonstrate two applications of our technique through hand tracking with a color glove and upper body tracking with a color shirt.

1.1 Hand Tracking

Recent trends in user-interfaces have brought on a wave of new consumer devices that can capture the motion of our hands. These include multi-touch interfaces such as the iPhone and the Microsoft Surface as well as camera-based systems such as the Sony EyeToy and the Wii Remote. These devices have enabled new interactions in applications as diverse as shopping, gaming, and animation control [52]. In this thesis, we introduce a new user-input device that captures the freeform, unrestricted motion of the hands for desktop virtual reality applications. Our motion-capture system tracks both the individual articulation of the fingers and the 3-D global hand motion.

Interactive 3-D hand tracking has been employed effectively in virtual reality applications including collaborative virtual workspaces, 3-D modeling, and object selection [1, 21, 69, 51, 41]. We draw with our hands [28]. We are accustomed to physically-based interactions with our hands [31, 71]. Hand-tracking can even be combined with multi-touch to provide a combination of 2-D and 3-D input [6].

While a large body of research incorporates hand tracking systems for user-input, their deployment has been limited due to the price and setup cost of these systems. We introduce a consumer hand tracking technique requiring only inexpensive, commodity components. Our prototype system provides useful hand pose data to applications at interactive rates.
1.2. UPPER BODY TRACKING

Figure 1-1: We describe a system that can reconstruct the pose of the hand from a single image of the hand wearing a multi-colored glove. We demonstrate our system as a user-input device for desktop virtual reality applications.

We achieve these results by proposing a novel variation of the hand tracking problem: we require the user to wear a glove with a color pattern. We contribute a data-driven technique that robustly determines the configuration of a hand wearing such a glove from a single camera. In addition, we employ a Hamming-distance-based acceleration data structure to achieve interactive speeds and use inverse kinematics for added accuracy. Despite the depth ambiguity that is inherent to our monocular setup, we demonstrate our technique on tasks in animation, virtual assembly and gesture recognition.

1.2 Upper Body Tracking

Motion capture data has revolutionized feature films and video games. However, the price and complexity of existing motion capture systems have restricted their use to research universities and well-funded movie and game studios. Typically, mocap systems are setup in a dedicated room and are difficult and time-consuming to relo-
cate. In this thesis, we propose a simple mocap system consisting of a laptop and one or more webcams. The system can be set up and calibrated within minutes. It can be moved into an office, a gym or outdoors to capture motions in their natural environments.

Figure 1-2: We describe a lightweight color-based motion capture system that uses one or two commodity webcams and a color shirt to track the upper body. Our system can be used in a variety of natural lighting environments such as this squash court, a basketball court or outdoors.

Our system uses a robust color calibration technique and a database-driven pose estimation algorithm to track a multi-colored object. Color-based tracking has been used before for garment capture [49] and hand tracking [68]. However these techniques are typically limited to studio settings due to their sensitivity to the lighting environment. Working outside a carefully controlled setting raises two major issues.
1.3. DISSERTATION OUTLINE

The color of the incident light may change, thereby altering the apparent color of the garment in nontrivial ways. One may also have to work in dimly lit scenes that require slower shutter speeds. However, longer exposure times increase motion blur, which perturbs tracking. We contribute a method that continuously compensates for light color variations and is robust to motion blur. We demonstrate in the results section that this enables our system to track activities in real-world settings that are challenging for existing garment-based techniques.

Our tracking approach is complementary to more sophisticated setups such as those that use infrared cameras and markers. Without achieving the same accuracy, our system is sufficiently precise to perform tasks such as motion analysis and contact detection, which makes it usable for augmented reality and human-computer interaction. Its low cost and ease of deployment make it affordable to the masses and we believe that it can help spread mocap as an input device for games and virtual reality applications. Furthermore, since our system enables tracking at interactive rates, it enables instant feedback for previsualization purposes.

1.3 Dissertation Outline

Chapter 2 begins with an overview of related work.

Chapter 3 addresses the issue of designing a color garment. We explore different possible patterns for a patchwork garment (§3.2), deciding on a pattern with twenty relative large patches. We optimize the set of ten colors assigned to the patches by calibrating the fabric printing and image acquisition process (§3.1). We also optimize the assignment of the colors to the patches based on avoiding collisions between patches of the same color for all possible poses (§3.3).

Chapter 4 discusses our database driven pose estimation algorithm. We describe the construction of our database by sampling an overcomplete collection of hand poses (§4.1.1). Each pose in the database is rasterized and normalized to produce a tiny image. A similar normalization process is performed on each image from the camera (§4.1.2). We compare the input image from the camera with the images in the
database using a tiny image distance (§4.1.3), blending the closest matches. We also describe an acceleration data structure for the database lookup based on hashing the database entries into short binary codes (§4.2). Section 4.3 describes how we refine our pose estimate after the data-driven step. We establish inverse-kinematics constraints for each visible patch in the camera image and use a gradient-based optimization to determine a pose that satisfies them. Temporal coherency is also preserved in the optimization to reduce jitter.

Chapter 5 deals with locating the color garment in a cluttered background and pose estimation in realistic environments with mixed lighting. We propose a fast method of detecting and cropping the color garment by searching for an image region with the appropriate color histogram. Our histogram search technique allows us to pick out the glove or shirt with only a coarse background subtraction and white balance estimate. We improve the robustness of the pose estimation step described in Chapter 4 by iteratively estimating both the color and pose of the cropped region.

We present results in Chapter 6 for hand tracking and upper body tracking in both clean and cluttered environments. Processing is performed at interactive rates on a laptop, enabling a wide range of applications (Chapter 7).

Finally, Chapter 8 summarizes our contributions and outlines future work on color-garment-based tracking.
Chapter 2

Related work

2.1 Hand Tracking

2.1.1 Marker-Based Motion Capture

Marker-based motion-capture has been demonstrated in several interactive systems and prototypes. However, these systems require obtrusive retro-reflective markers or LEDs [42] and expensive many-camera setups. They focus on accuracy at the cost of ease of deployment and configuration. While our system cannot provide the same accuracy as high-end optical mocap, our solution is simpler, less expensive, and requires only a single camera.

2.1.2 Instrumented Gloves

Instrumented glove systems such as the P5 Data Glove and the CyberGlove [39] have demonstrated precise capture of 3-D input for real-time control. However, these systems are expensive and unwieldy. The P5 Data Glove relies on an exoskeleton, which can be restrictive to movement, while the CyberGlove embeds more than a dozen sensors into a glove, which makes the glove very warm and uncomfortable to wear for an extended amount of time. Our approach uses a completely passive glove, made of a lightweight (10 g) Lycra and polyester weave.
2.1.3 Color-Marker-Based Motion Capture

Previous work in color-based hand tracking have demonstrated applications in limited domains or for short motion sequences. Theobalt and colleagues track a baseball pitcher’s hand motion with color markers placed on the back of a glove using four cameras and a stroboscope [64]. Dorner uses a glove with color-coded rings to recognize (6 to 10 frame) sequences of the sign language alphabet [19]. The rings correspond to the joints of each finger. Once the joint positions are identified, the hand pose is obtained with inverse kinematics. We use a data-driven approach to directly search for the hand pose that best matches the image.

2.1.4 Bare-Hand Tracking

Bare-hand tracking continues to be an active area of research. Edge detection and silhouettes are the most common features used to identify the pose of the hand. While these cues are general and robust to different lighting conditions, reasoning from them requires computationally expensive inference algorithms that search the high-dimensional pose space of the hand [61, 58, 16]. Such algorithms are still far from real-time, which precludes their use for control applications.

Several bare-hand tracking systems achieve interactive speeds at the cost of resolution and scope. They may track the approximate position and orientation of the hand and two or three additional degrees of freedom for pose. Successful gesture recognition applications [48, 17, 33] have been demonstrated on these systems. We propose a system that can capture more degrees-of-freedom, enabling direct manipulation tasks and recognition of a richer set of gestures.

2.1.5 Depth Sensing Cameras and Shadows

Another recent avenue of research uses depth sensing cameras to detect hand gestures. Hilliges and colleagues combine an FTIR multi-touch surface and the 3DV depth sensing “ZSense” camera to provide interactions above the surface [24]. Benko and Wilson use a short throw projector, a transparent vertical display screen, and a ZSense
camera to simulate a window into a 3D virtual scene [7]. Several depth cameras and projectors can be used to produce an intelligent workspace that allows interaction between surfaces [74].

The advantage of depth sensing cameras is that 3-D gestures may be easier to recognize with rgb-z data [73]. Background subtraction is more straightforward, and the depth data augments edge and silhouette-based tracking. However, inferring articulation of the fingers is still a difficult problem even with depth data. As depth cameras become more widely available, we hope to extend our proposed data-driven pose estimation system to work with rgb-z images.

Another approach to hand tracking uses the projected shadows of a user’s hands on a surface. In the PlayAnywhere system [72], Wilson uses a high-powered IR LED and a camera with an IR pass filter to illuminate and observe the hands from above. The PlayAnywhere system can precisely detect contacts with a surface by analyzing the shadows projected by the IR illuminant. Starner and colleagues [56] use several IR illuminants and cameras to obtain several shadow projections on a surface. The shadow volumes corresponding to the projections are intersected to obtain a visual hull of the user’s arm. In general, reconstructing geometry from shadows requires careful setup of both illuminants and cameras, complicating the calibration process. Our system forgoes illuminants for this reason, trading off the extra information from predictable shadows for faster setup time and less hardware.

2.2 Body-Tracking

2.2.1 Markerless Full-Body Motion Capture

The accuracy of markerless motion capture systems typically depend on the number of cameras used. Monocular or stereo systems are portable but less accurate and limited by the complexity of the motion [38].

Hasler and colleagues [23] use four high-definition handheld cameras for markerless motion capture. Their approach applies RANSAC structure-from-motion and non-
linear bundle adjustment per camera per frame to stabilize the cameras and build a 3-D model of the background. The computational cost of these steps precludes use in interactive applications such as virtual reality and gaming.

Furthermore, the pose estimation algorithm [47] used by Hasler relies on a motion database to assist segmentation and tracking. Similarly, Sidenbladh and colleagues [53] also employ a motion database. This is a fundamental difference with our approach, which only needs a dataset of static poses. Even with a simple model of motion where one considers only the position and speed of each joint, the number of degrees of freedom to describe motion is double that of static poses defined only by joint positions. Since the size of a space grows exponentially with its dimension, the space of motions is several orders of magnitude larger than the space of static poses. Because of this, sampling all possible motions is impossible and the references above consider only specific activities e.g. walking, running, dancing. In contrast, our approach exhaustively samples the space of poses and is agnostic to the captured activity. All the sequences demonstrated in our video, including the complex sports motions, have been processed using the same dataset.

Our approach also affords several advantages over recent work on markerless performance capture in the graphics literature [14, 67]. We use two cameras instead of eight or twelve, which facilitates portability as well as faster setup and calibration. We can capture in natural environments without controlled lighting or a green screen. Furthermore, our pose estimation algorithm is fast enough for interactive applications such as virtual reality and gaming.

Commercial systems such as Microsoft Xbox Kinect [37] and iMocap [9] also aim for on-site and easy-to-deploy capture. Although little is publicly available about these techniques, we believe that our approach is complementary to them. Kinect relies on infrared illumination and is most likely limited to indoor and short-range use while iMocap is marker-based which probably requires a careful setup. Beyond these differences, our work and these methods share the same motivations of developing mocap for interactive applications such as games [26], augmented reality, and on-site previsionalization.
2.2.2 Data-Driven Pose Estimation

Our work builds upon the techniques of data-driven pose estimation. Shakhnarovich and colleagues [50] introduced an upper body pose estimation system that searches a database of synthetic, computer-generated poses. Athitsos and colleagues [4, 3] developed fast, approximate nearest-neighbor techniques in the context of hand pose estimation. Ren and colleagues [46] built a database of silhouette features for controlling swing dancing. Our system imposes a pattern on the garment designed to simplify the database lookup problem. The distinctive pattern unambiguously gives the pose of the upper body, improving retrieval accuracy and speed.

2.3 Deformable Surface Tracking

Our technique is also related to deformable surface tracking. Hilsmann and Eisert [25] track and re-texture a garment for visualization. Pilet and colleagues [43] solve for the deformation of a reference image on a non-rigid surface. Bradley and colleagues [11] introduce a marker-based approach for augmented reality on a t-shirt. All of these techniques are primarily concerned with capturing fine-scale details such as wrinkles and folds on a shirt, while we solve for the pose of an articulated model.

2.4 Color-Based Image Analysis

The histogram tracking aspect of our work is also related to image analysis techniques that rely on colors. Swain and Ballard showed that color histograms are effective for identifying and locating objects in a cluttered scene [62]. Comaniciu et al. [12] track an object in image space by locally searching for a specific color histogram. In comparison, we locate the shirt without assuming a specific histogram, which make our approach robust to illumination changes. Furthermore, our algorithm is sufficiently fast to perform a global search. It does not rely on temporal smoothness and can handle large motions. Dynamic color models have been proposed to cope with illumination changes, e.g. [36, 27, 54]. The strong occlusions that appear with our
shirt would be challenging for these models because one or several color patches can disappear for long periods of time. In comparison, we update the white balance using the a priori knowledge of the shirt color. We can do so even if only a subset of the patches is visible, which makes the process robust to occlusions. Generic approaches have been proposed for estimating the white balance, e.g. [20], but these are too slow to be used in our context. Our algorithm is more efficient with the help of a color shirt as a reference.
Garment Design for Color-Based Tracking

Our use of a color garment is inspired by advances in cloth motion capture, where dense patterns of colored markers have enabled precise deformation capture [70, 49, 22]. A variety of color patterns may be appropriate for tracking. Our particular design is motivated by the limitations of consumer-grade cameras, the significant amount of self-occlusion for both hand tracking and upper body tracking, and the speed requirements of the inference algorithm.

Figure 3-1: Our glove design consisting of large color patches accounts for camera limitations, self-occlusion, and algorithm performance. The length of our glove is 24 cm.

We describe a garment with twenty patches colored with a set of ten distinct colors (See Figure 3-1). We chose to use a few large patches rather than numerous small patches because smaller features are less robust to occlusion and require more complex patch identification algorithms for pose inference [70, 49, 22]. Our color set
is limited by our computer vision system, which looks for fully saturated colors to segment the hand from the background. Our camera could reliably distinguish only ten of these colors due to shadows and shading. Each of the ten colors is assigned to two of the twenty patches, and we optimize the assignment to minimize ambiguity.

3.1 Color Selection

We used a set of highly saturated colors to distinguish the garment from the background, since a desktop environment is typically composed of relatively unsaturated colors such as gray and brown. However, due to the limitations of the camera and printing process, uniformly sampling fully-saturated colors does not always achieve good results. A printer may be able to print a more saturated range of blues than reds, while a commodity webcam may be more capable of perceiving differences in greens versus blues. Thus, we developed a systematic means of calibrating the printing and perception process to sample a tailored set of colors.

We sampled a set of 36 colors in the HSV color space, uniformly distributed in hue, with full saturation and value. We printed these colors in a $6 \times 6$ grid on a square of size 8in $\times$ 8in and observed the printed grid with our webcam. We observed several images of the printed sheet in various orientations to simulate the colors with different shading (see Figure 3-2). Next we manually mark the ground truth labels of each of the 36 colors by scribbling on top of the images. We built a normalized-RGB Gaussian model for each labeled color $(\mu_i, \Sigma_i)$.

We sought a set of colors that were maximally different from a classification perspective, and so used the same Mahalanobis distance as we do for classification. For each pair of colors, we chose the point $p^*$ closest to the two Gaussians with respect to Mahalanobis distance,

$$p^* = \arg \min_{p} ||p - \mu_1||_\Sigma_1 + ||p - \mu_2||_\Sigma_2$$
3.1. COLOR SELECTION

Figure 3-2: We print a 6 x 6 grid of saturated colors for our calibration pattern. We then observe the pattern held at various angles from the camera.

We set the distance to $p^*$ from each Gaussian as the distance between the colors.

\[ d(c_1, c_2) = ||p^* - \mu_1||_{\Sigma_1} + ||p^* - \mu_2||_{\Sigma_2} \]

Next, we tested each combination of ten colors out of the thirty-six, choosing the set with the maximum minimum distance between any two pairs of Gaussians.

\[ S^* = \arg\max_S \min_{i,j \in S} d(c_i, c_j) \]

The choice of ten colors affords us a generous distance of $S = 5.7$ standard deviations between the two closest colors. While over 5 standard deviations may seem excessive, we have not yet accounted for effects such as mixed lighting and inter-reflections which can make colors appear more similar to each other.
3.2 Pattern Layout

Before deciding on a twenty-patch pattern, we experimented with several patch sizes (See Figure 3-3). We found that larger patches are easier to detect because they occupy more pixels of the image. They are more robust to motion blur because only the boundary of the patches are distorted. However larger patches are also less informative than smaller patches. A patch covering an entire finger is not discriminative enough to localize the finger tip or describe how it is bent, while a collection of (correctly identified) smaller patches can more precisely describe the deformation of a finger or an arm.

However smaller patches are more difficult to detect and identify, especially on thin structures like the fingers. They are especially prone to failure when the fingers are partially-occluded and only a fraction of the patches are visible. Because our camera can only robustly identify a small set of colors, more complex patch identification algorithms such as belief propagation are employed to tolerate repetition of colors amongst the patches [70]. There is also a limit on the minimum patch size due to the placement of the camera. To enable a large capture workspace, the garment can occupy only a small part of the image, and patches must span at least a few pixels to be identified.

First and foremost, our pattern is designed for efficient database-driven pose estimation. One of the challenges of bare-hand pose estimation is that two very different poses can map to very similar images. One of the goals of the gloved hand is to ensure that very different poses always map to very different images (See Figure 3-4). This property forms the basis for the simple image lookup approach which we describe in the next chapter.

We quantitatively measured the effect of patch size on retrieval accuracy for a database-driven pose estimation algorithm (See Chapter 4). We considered patch densities of between 1 and 75 patches per glove. We evaluated retrieval accuracy on a database of 100,000 hand poses using 500 synthetically generated random poses. We suspected that up to a certain extent, a higher number of patches would reduce
3.3 COLOR ASSIGNMENT

Figure 3-3: We experimented with various patch sizes and shapes.

ambiguity for a database-driven pose estimation algorithm. We found that we could improve retrieval accuracy by increasing patches density up until 25 patches (See Figure 3-5). At this point, performance stays roughly constant. That is, we can reduce ambiguity (See Figure 3-4) of a naked hand by using more multi-colored patches up until about 25 patches per glove.

3.3 Color Assignment

Once we have chosen ten colors, we assign them to twenty patches, duplicating each color once. By having twenty patches, we are able to obtain finer resolution deformations of the fingers. However, duplicating colors also means that sometimes we will
Figure 3-4: The palm down and palm up poses map to similar images for a bare hand. These same poses map to very different images for a gloved hand.

Figure 3-5: We measured retrieval accuracy as a function of the number of patches on a glove. Performance peaks at around 25 patches after which there are no benefits to using a higher patch count.

confuse one patch for another when we establish inverse kinematics constraints for refining our pose estimate (See §4.3). Fortunately, we can also optimize the assignment to minimize the impact of color duplication.

Specifically, we seek to maximize the image-space distance between patches assigned the same color for all hand poses. We sample \( N = 10^6 \) synthetic rasterized \( 40 \times 40 \) hand-images \( r^i \) that span common hand poses (see Section 4.1). The patch distance between each pair of patches is computed as the average minimal distances between pixels belonging to each patch over the set of all hand-images,
3.3. COLOR ASSIGNMENT

\[ AMD(p_1, p_2) = \frac{1}{N} \sum_{i}^{N} \min_{(u,v) \in P_1} \min_{(x,y) \in P_2} \min(\sqrt{(u-x)^2 + (v-y)^2}, \sigma_{\max}) \]

where \( P_j = \{(x,y)|r^j = p_j\} \)

We cap the distance between two patches to \( \sigma_{\max} = 5 \) pixels, the threshold where two patches are completely unambiguous. To compute the score for an assignment \( a \), we take the minimum of the average minimal distance between patches assigned the same color,

\[ S(a) = \min_{\{(p_i, p_j)|a(p_i) = a(p_j)\}} AMD(p_i, p_j) \]

The higher the score \( S(a) \), the less likely the color assignment will lead to a collision of patch colors for any hand pose.

![Best assignment score vs assignments tried](image)

**Figure 3-6:** As we try more random assignments, the best assignment score \( S(a) \) quickly converges to a value close to \( \sigma_{\max} = 5 \).

We tested \( 10^7 \) random assignments of colors to the garment and chose the assignment with the best assignment score (See Figure 3-6). While we do not claim that
our particular assignment is optimal, it is a sufficiently good assignment of colors that we largely avoid ambiguities (See Figures 3-7 and 3-8).

![Glove for Left Hand](image)

**Figure 3-7:** Our glove design. Note that the finger tips, which can reach any part of the palm, are seldom assigned the same colors as the palm, since the fingers often bend towards the palm. On the other hand, the finger tips are often assigned the same colors as patches on the back of the hand since the fingers can never bend backwards.

### 3.4 Discussion

In this chapter, we have presented guidelines for designing a color garment for tracking. First, we presented a method for selecting distinguishable colors. We described a procedure to pick colors that were maximally far apart from the point of view of our camera. Next we discussed the trade-offs between gloves with different patch densities. We showed that a glove of our design can significantly improve database retrieval accuracy over a bare hand. Finally, we optimized the assignment of colors to patches to minimize ambiguity.

Our garment design is sufficiently distinctive by design that we can reliably infer the pose of the hand from a single frame. This compares favorably to hand tracking approaches that rely on an accurate pose from the previous frame to constrain the search for the current pose. When these approaches lose track of the hand, they have
3.4. DISCUSSION

Figure 3-8: Our shirt design. Note that the location of the wrists, the most kinematically flexible part of the body, are assigned the same colors as their opposite shoulders.

no means of recovery [15]. Our pose estimation (versus tracking) approach effectively "recovers" at each frame.

While we have explored the space of equally-sized colored patches for use in database-driven pose estimation, the unexplored design space is large. For instance, it could be advantageous to allocate patches of different sizes around the hand—smaller patches near smaller joints, and larger patches elsewhere. We do not make use of texture cues to introduce higher-frequency content on the glove design. For instance, stripes could provide orientation cues in addition to position cues.

More generally, we've limited ourselves to a very specific database-driven algorithm with which to test our patterns. We have gravitated towards retrieval techniques that rely on low-resolution tiny images. For future work, we would like to
explore sharp corners, gradients or blobs that could be detected quickly with other specialized feature detectors such as SIFT.
Database-Driven Pose Estimation

The core of our approach is to infer pose (of the hand or the upper body) from an image of the color garment. We design our garment so that this inference task amounts to looking up the image in a database. We generate this database by sampling an overcomplete set of natural poses and index it by rasterizing images of the garment in these poses. A (noisy) input image from the camera is first transformed into a normalized query. It is then compared to each entry in the database according to a robust distance metric. We evaluate our data-driven pose estimation algorithm and show a steady increase in retrieval accuracy with the size of the database.

While our tiny image database look up is fast, it is not fast enough for real-time performance. We describe an acceleration data structure based on boosting and hamming distance that increases the speed of look up by an order of magnitude.

Our inference algorithm produces a roughly accurate estimate of the hand pose. We can refine this estimate by establishing correspondences between colored patches in our image and color patches predicted by our hand model. From these correspondences, we apply inverse kinematics to constrain the model to better match the image, thus producing a more accurate estimate.

Without loss of generality, we will focus on the application of hand tracking in this chapter.
4.1 Database Construction

We construct a database of hand poses $\Lambda$ consisting of a large set of hand configurations $\{q^i\}$, indexing each entry by a tiny $(40 \times 40)$ rasterized image of the pose $r^i$ (See Figure 4-1). Given a normalized query image from the camera, pose estimation amounts to searching a database of tiny images [65]. The pose corresponding to the nearest neighbor is likely to be close to the actual pose of the hand (See Figure 4-2). To complete this process, we describe a means of constructing a database, normalizing an image from the camera to query our database, and judging distance between two tiny images.

Figure 4-1: Pose database. We sample a large set of hand poses which are indexed by their rasterized tiny images.

Figure 4-2: Our pose estimation process. The camera input image is transformed into a normalized tiny image. We use the tiny image as the query for a nearest neighbor search of our pose database. The pose corresponding to the nearest database match is retrieved.
4.1. DATABASE CONSTRUCTION

4.1.1 Database Sampling

Ideally, we would like a small database that uniformly samples all natural hand configurations. An overcomplete database consisting of many redundant samples would be inefficient. Alternatively, a database that does not cover all natural hand configurations would result in poor retrieval accuracy. Our approach uses low-dispersion sampling to select a sparse database of samples from a dense collection of natural hand poses.

We collected a set of 18,000 finger configurations using a CyberGlove II motion capture system. These configurations span the sign language alphabet, common hand gestures, and random jiggling of the fingers. We define a distance metric between two configurations using the root mean square (RMS) error between the vertex positions of the corresponding skinned hand models.

Given a distance metric $d(q_i, q_j)$, we can use low-dispersion sampling to draw a uniform set of samples $\Lambda$ from our overcomplete collection of finger configurations $\Omega$. The dispersion of a set of samples is defined to be the largest empty sphere that can be packed into the range space (of natural hand poses) after the samples have been chosen. We use an iterative and greedy sampling algorithm to efficiently minimize dispersion at each iteration. That is, given samples $\Lambda_\ell$ at iteration $\ell$, the next sample $i_{\ell+1}$ is selected to be furthest from any of the previous samples.

$$i_{\ell+1} = \arg\max_{i \in \Omega} \min_{j \in \Lambda_\ell} d(q_i, q_j)$$

The selected configurations are rendered at various 3-D orientations using a (synthetic) hand model at a fixed position from the (virtual) camera. The rendered images are cropped and scaled, forming our database of tiny images. The result is a database that efficiently covers the space of natural hand poses.
4.1.2 Image Normalization

To query the database, we convert the camera input image into a tiny image (See Figure 4-3). First we smooth sensor noise and texture from the image using a bilateral filter. Next, we classify each pixel either as background or as one of the ten glove colors using Gaussian mixture models trained from a set of hand-labeled images. We train one three-component Gaussian mixture model per glove color.

After color classification, we are left with an image with glove-pixels and non-glove-pixels. In practice, we use mean-shift with a uniform kernel of variable-bandwidth to crop the glove region. We start at the center of the image with a bandwidth that spans the entire image. After each iteration of mean-shift, we set the bandwidth for the next iteration based on a multiple of the standard deviation of the glove pixels within the current bandwidth. We iterate until convergence, using the final mean and bandwidth to crop the glove region.

4.1.3 Tiny Image Distance

To look up the nearest neighbor, we define a distance metric between two tiny images. We chose a Hausdorff-like distance. For each non-background pixel in one image, we penalize the distance to the closest pixel of the same color in the other image (See Figure 4-4):
\[ d(r', r^j) = \sqrt{\frac{1}{|C_i|} \sum_{(u,v) \in C_i} \min_{(u',v') \in S_{xy}} (u - x)^2 + (v - y)^2} \]

where \( S_{xy} = \{(u,v) | r_{x,y}^i = r_{u,v}^j\} \)

\[ C_i = \{(x,y) | r_{x,y}^i \neq \text{background}\} \]

\[ d(r^i, r^j) = d(r^i, r^j) + d(r^j, r^i) \]

**Figure 4-4:** Hausdorff-like image distance. A database image and a query image are compared by computing the divergence from the database to the query and from the query to the database. We then take the average of the two divergences to generate a symmetric distance.

We found that our Hausdorff-like image distance metric was more robust to alignment problems or minor distortions of the image than the \( L_2 \) distance. For a database of 100,000 poses, our Hausdorff-like metric is able to retrieve nearest neighbors approximately 12% closer than \( L_2 \). Our distance is also more robust to color misclassification than a Hausdorff distance that takes the maximum error across all pixels rather than an average.

The nearest neighbor tiny image can provide an approximate pose of the hand, but cannot account for global position (e.g. distance to the camera) since each query is resized to a tiny image. To address this, we associate 2-D projection constraints with each tiny image for the centroids of every color patch. Thus we can obtain the global hand position by transforming these projection constraints into the coordinate space of the original query image and using inverse kinematics.
Given the database construction and lookup algorithms described above, we can quantitatively evaluate the effect of database size on the accuracy of retrieval. For each database size, we measure the average performance of five hundred test poses sampled randomly from our collection of recorded natural hand configurations. We observe the distance to the nearest neighbor in the database according to the pose distance metric and the image distance metric (See Figure 4-5).

![Distance to nearest neighbor (NN) versus database size](image)

**Figure 4-5:** Database coverage evaluation (log scale). As the database size increases, pose estimation becomes more accurate.

The consistent decrease of the pose nearest-neighbor distance with database size validates the efficiency and coverage of our database sampling. The consistent decrease of the image nearest-neighbor distance validates our tiny images distance metric and indicates that retrieval becomes more accurate with database size.

### 4.1.4 Blending Nearest Neighbors

In addition to selecting the nearest neighbor in our database, we can also use a blend of the $k$-nearest-neighbors. This provides a smoother and more accurate result. We blend a neighborhood $\mathcal{N}$ of the ten nearest tiny images with a Gaussian radial basis...
kernel,

\[ q_p(r) = \frac{\sum_{i \in N} q^i \exp \left( -\frac{d(r^i, r)^2}{\sigma^2} \right)}{\sum_{i \in N} \exp \left( -\frac{d(r^i, r)^2}{\sigma^2} \right)} \] (4.1)

where \( \sigma \) is chosen to be the average distance to the neighbors.

Blending nearest neighbors in this way provides a more accurate result than taking the top nearest neighbor. To choose \( k \), we measured average accuracy improvement on simulated random pose over using only the first nearest neighbor \( k = 1 \). For a database of 100,000 images, we obtain pose accuracy improvements of 27.1\%, 28.3\%, 27.7\% and 26.9\% by blending \( k = 5, 10, 15 \) and 25 neighbors respectively.

### 4.2 Fast Lookup Using Boosting

To achieve satisfactory accuracy, we use a database size of 100,000 entries for the hand model. However, querying a database of this size is computationally expensive. While our simple Hausdorff-like image distance is robust, it is not fast enough to be evaluated millions of times per second. Instead, we follow Torralba and colleagues, compressing each database entry into a short (e.g. 192-bit) binary code for faster retrieval [66, 3]. The codes are designed so that nearby poses are encoded as bit strings with a small Hamming distance. Since these codes are short and the Hamming distance is fast, we can significantly accelerate database search with this approach.

For our database of images \( \{r^i\} \) and distance metric \( d(r^i, r^j) \), we seek a bitvector representation for each image \( b^i = [h_1(r^i)h_2(r^i) \ldots h_B(r^i)] \) such that the hamming distance \( d_H(r^i, r^j) = \sum_{n=1}^B |h_n(r^i) - h_n(r^j)| \) preserves the neighborhood relationships of our original distance metric. Our task is to learn the functions \( h_n \). Once the functions \( h_n, n = 1 \ldots B \), have been selected, we can encode any query as a bitvector \( b \) and use the faster Hamming distance to determine its nearest neighbors in the database.

In the learning phase, we construct a training set composed of example pairs \( (r^i, r^j) \). Positive examples are pairs that are nearest-neighbors. Negative examples
are pairs that are not. We follow the similarity sensitive coding [50] formulation of Torralba and colleagues [66], defining \( h_n \) to be of the form \( h_n(r') = e_n^T \text{vec}(D(r')) > T_n \), where \( e_n \) and \( T_n \) are learned parameters and \( D(r') \) is the feature matrix representing the tiny image \( r' \). We use the GentleBoost algorithm to select the parameters of \( h_n \): \( e_k \) is a unit vector, so that \( e_k^T x \) selects the \( k \)th component of a feature vector \( x \); and \( T_n \) is a scalar threshold. Our choice of features resembles the Hausdorff-like image distance metric that we seek to approximate. Each component of the feature matrix \( D(r) \in \mathbb{R}^{40 \times 40 \times 10} \), \( D_{xyz}(r) \), expresses the shortest distance to a pixel with color \( z \) from the location \((x, y)\):

\[
D_{xyz}(r) = \min_{(u,v) \in S_z} (u - x)^2 + (v - y)^2
\]

where \( S_z = \{(u,v) | r_{x,y} = z\} \)

We sampled 10,000 pairs of training examples from our database and experimented with bitvectors of various lengths. We measured the retrieval accuracy of our codes by computing the rank of the true nearest neighbor according to the learned Hamming distance approximation. We found that the rank decreases quickly with the length of the code (See Figure 4-6).

For our real-time system, we used 192-bit codes, yielding an average rank of 16 for the true nearest neighbor and a standard deviation of 78. To robustly obtain the true nearest neighbor, we re-rank the top 300 approximate nearest-neighbors with the original image distance. Overall, we achieve results approximately 50 times faster than the brute-force nearest-neighbor search described in the previous section.

### 4.3 Inverse Kinematics

We improve upon our nearest-neighbor pose estimate by using inverse kinematics to penalize differences between the rasterization of the pose estimate and the original image. However, rasterization is too slow to perform at every iteration of inverse
4.3. INVERSE KINEMATICS

Figure 4-6: The rank of the true nearest neighbor (log scale) according to the Hamming distance approximation decreases quickly with longer (more descriptive) codes.

kinematics, and the image Jacobian is difficult to evaluate [15]. Instead, we establish correspondences between points on the rasterized pose and the original image. We compute the centroids of each of the visible colored patches in the rasterized pose and identify the closest vertex to each centroid. We then constrain these vertices to project to the centroids of the corresponding patches in the original image (see Figure 4-7). Note that our correspondences are still valid for poses with self-occlusion because the nearest-neighbor result is usually self-occluded in the same way as the image query.

Figure 4-7: Correspondences for inverse kinematics. We compute centroids of each patch in the query image and the nearest neighbor pose. We establish correspondences between the two sets of points and use IK to penalize differences between the two.

We use inverse kinematics to minimize the difference between each projected ver-
tex from our kinematic model and its corresponding centroid point in the original image. We regularize by using the blended nearest neighbor \( q_p \) (See Equation 4.1) as a prior:

\[
q^* = \arg\min_{q} \| f(q) - c \|^2_R + \| q - q_p \|^2_Q
\]  

(4.2)

where \( f \) is a nonlinear function that projects the corresponded points of our kinematic model into image space; \( R \) and \( Q \) are the covariances of the constraints \( c \) and the blended nearest neighbor \( q_p \) respectively; and \( \| \cdot \|_X \) is the Mahalanobis distance with respect to covariance \( X \).

We learn the covariance parameters \( R \) and \( Q \) on a training sequence as follows. We define a local average of the estimated pose \( \bar{q}_p^i = \frac{1}{|S|} \sum_{j \in S} q_p^{i+j} \) over each consecutive five-frame window \( S = \{-2, \ldots, 2\} \), and compute covariances of \( c \) and \( q_p \) about these local averages,

\[
R = \frac{1}{N} \sum_{i=1}^{N} \left( c^i - f(q_p^i) \right) \left( c^i - f(q_p^i) \right)^T
\]

\[
Q = \frac{1}{N} \sum_{i=1}^{N} \left( q_p^i - q_p^i \right) \left( q_p^i - q_p^i \right)^T
\]

We use Gauss-Newton iteration to solve Equation 4.2, with an update of

\[
\Delta q = (J^T R^{-1} J + Q^{-1})^{-1} (J^T R^{-1} \Delta c + Q^{-1} \Delta q_p)
\]

where \( \Delta q_p = q_p - q \), \( \Delta c = c - f(q) \) and \( J \) is the Jacobian matrix \( Df(q) \).

### 4.3.1 Temporal Smoothness

We can add an additional term to our inverse kinematics optimization to smooth our results temporally:
4.4. DISCUSSION

\[
q^* = \arg\min_q \| f(q) - c \|_R^2 + \| q - q_p \|_Q^2 + \| q - q_h \|_P^2
\]

where \( P \) is the covariance of the pose in the last frame \( q_h \), and is learned on a training sequence similarly to \( Q \) and \( R \). This yields a Gauss-Newton update of the form:

\[
\Delta q = (J^T R^{-1} J + Q^{-1} + P^{-1})^{-1} (J^T R^{-1} \Delta c + Q^{-1} \Delta q_p + P^{-1} \Delta q_h)
\]

where \( \Delta q_h = q_h - q \).

4.4 Discussion

In this chapter, we have described a method for database-driven pose estimation and demonstrated its performance on databases of colored garments. We have shown that with a reasonably sized database of 100,000 poses, we can accurately determine the pose of a colored glove using a combination of tiny images, our Hausdorff-like distance metric, and weighted blending. One advantage of the database-driven technique is that we can improve accuracy simply by adding more entries to the database. That is, our technique will naturally improve as memory capacity grows and computers become faster.

We have also described a method for accelerating database queries by compressing the keys to 192-bit binary codes. These keys are designed so that their hamming distances preserve neighbor relationships. Our acceleration data structure is able to increase retrieval speed by a factor of 50, without sacrificing retrieval accuracy. This reduces database lookup times to under 20 ms, allowing our system to be incorporated in real-time applications.

Finally, we have shown that we can refine the database estimate using an inverse
kinematics optimization. The optimization operates on constraints determined from the centroids of colored patches. In addition to improving pose estimation accuracy, our optimization framework also allows us to incorporate regularization and temporal smoothing.

Our database-driven approach successfully leverages low-frequency features (large patches) for accurate and efficient retrieval, but the same features are not ideal for inverse kinematics. Computing the centroids of the color patches is our method of obtaining higher frequency features (points) from lower frequency ones (patches), but this method has its drawbacks. When a patch is partially occluded, its centroid estimate will be inaccurate. In general, the further the nearest neighbor estimate is from the actual pose, the more likely our algorithm could make a mistake matching the centroids between the nearest neighbor and the image. For future work, we would like to explore using inverse kinematics on actual point features of the glove, such as the intersection point of several patches. Alternatively, a set of point features could be encoded on top of the color patches.
Robust Color and Pose Estimation

The pose estimation method described in Chapters 4 and 4.3 depend on an accurate segmentation of the garment from the background. However this becomes a challenging task in itself in a cluttered environment. In this chapter, we leverage the extraordinary colorfulness of our garments to locate them.

Once the garment has been located, the next step is to perform color classification and estimate the 3-D pose. Once again, in real-world environments, the former is particularly challenging due to mixed or changing illumination. For instance, a yellow patch may appear bright orange in one frame and dark brown in another (Figure 5-1). In this chapter, we also describe a continuous color classification process that adapts to changing lighting and variations in shading.

5.1 Histogram-Based Localization

Our garment is distinctive from real-world objects in that it is particularly colorful, and we take advantage of this property to locate it. Our procedure is robust enough to cope with a dynamic background and inaccurate white balance. It is discriminative enough to start from scratch at each frame, thereby avoiding any assumption of temporal coherence.

To locate the garment, we analyze the local distribution of chrominance values in the image. We define the chrominance of an \((r, g, b)\) pixel as its normalized counterpart
Figure 5-1: The measured values of the color patches can shift considerably from frame to frame. Each row shows the measured value of two identically colored patches in two frames from the same capture session.

\( (r, g, b)/(r + g + b) \). We define \( h(x, y, s) \) as the normalized chrominance histogram of the \( s \times s \) region centered at \( (x, y) \). In practice, we sample histograms with 100 bins. Colorful regions likely to contain our garment correspond to more uniform histograms whereas other areas tend to be dominated by only a few colors, which produces peaky histograms (Figure 5-2 and 5-3). We estimate the uniformity of a histogram by summing its bins while limiting the contribution of the peaks. That is, we compute \( u(h) = \sum \min(h_i, \tau) \), setting \( \tau = 0.1 \). With this metric, a single-peak histogram has \( u \approx \tau \) and a uniform one \( u \approx 1 \). Other metrics such as histogram entropy perform similarly. The colorful garment region registers a particularly high value of \( u \). However, choosing the pixel and scale \( (x', y', s') \) corresponding to the maximum uniformity \( u_{\text{max}} = \max u(x, y, s) \) proved to be unreliable. Instead, we use a weighted average favoring the largest values:

\[
(x', y', s') = \frac{1}{\sum_{x,y,s} w(x, y, s) \sum_{x,y,s} (x, y, s) w(x, y, s)} \sum_{x,y,s} (x, y, s) w(x, y, s)
\]

where \( w(x, y, s) = \exp\left(-\frac{(u(h(x,y,s))-u_{\text{max}})^2}{u_{\sigma}^2}\right) \) and \( u_{\sigma} = \frac{1}{10} u_{\text{max}} \).

The garment usually occupies a significant portion of the screen, and we do not require precise localization. This allows us to sample histograms at every sixth pixel and search over six discrete scales. We build an integral histogram to accelerate
5.2. COLOR MODEL AND INITIALIZATION

While the histogram search process does not require background subtraction, it can be accelerated by additionally restricting the processing to a roughly segmented foreground region. In practice, we use background subtraction [57] for both the histogram search and to suppress background pixels in the color classification (§ 5).

**Figure 5-2:** The colorful shirt has a more uniform chromaticity (2-D normalized RGB) histogram (b) with many non-zero entries whereas most other regions have a peakier histogram (a) dominated by one or two colors. We visualize our uniformity measure \( u(h(x, y, s)) = \sum_{i} \min(h_i, 0.1) \) with scale \( s = 80 \) as a heat map.

**5.2 Color Model and Initialization**

Once the garment has been located, we need to perform color classification to generate the tiny image input for our pose estimation algorithm. This can be a challenging task in real-world environments due to mixed lighting, and we describe our method for coping with changing illumination and shading in the following sections. First, we describe the offline process of modeling the garment colors. The online component estimates an approximate color classification and 3-D pose before refining both to obtain the final pose. In addition to the final pose, we compute an estimate of
the current illumination as a white balance matrix and maintain a list of reference illuminations that we use to recover from abrupt lighting changes. Most of the examples in this chapter will show upper-body tracking, although no aspect of this technique is specific to tracking the upper body. We demonstrate robust color and pose estimation results for both hand-tracking and upper-body tracking.

We model each of the $k = 10$ distinct shirt colors as a Gaussian $N(\mu_k, \Sigma_k)$ in RGB space. We build this model ahead of time by manually labeling five white-balanced images of our shirt (Figure 5-4).

At the beginning of a capture session, we estimate the illumination of the scene by solving for a white balance transformation that maps colors in the current scene to the colors captured by our Gaussian color model. We propose two methods to achieve this. The first is a simple technique requiring manual labeling of the shirt colors in an image. The second is an automatic technique well-suited to interactive applications.

For the manual process, we estimate the illumination by taking an image of the shirt in the current environment and asking the user to coarsely scribble ground truth labels on each of the visible color patches of the shirt. We solve for a $3 \times 3$ white balance matrix $W$ that maps the mean patch colors $\mu'_k$ from the reference image to the mean colors $\mu_k$ of the Gaussian color model, that is, $W = \arg\min_W \sum_k \|W\mu_k - \mu'_k\|^2$. 

**Figure 5-3:** We show several frames of a hand tracking sequence with their uniformity heat maps. Notice that the system is not confused by the colorful mousepad. Only a Rubik's cube comes close to approaching the color uniformity of the glove.
The white balance matrix $W$ is used to bootstrap our color and pose tracking, and we also use it to initialize our list of reference illuminations.

Figure 5-4: Ahead of time, we build a color model of our shirt by scribbling on 5 white-balanced images. We model each color with a Gaussian distribution in RGB space.

For scenarios such as gaming where the user cannot manually initialize the white balance estimate, we have developed a simple method for automatic white balance initialization. The method requires that initially the user stands in the middle of the camera’s field-of-view facing the camera so that the shirt is centered in the image. From this image, we seek to identify the colors corresponding to the patches and align them to the colors of our model. We proceed in two steps. In the first step, we look at the peaks of the chrominance (normalized RGB or NRGB) histogram of the shirt region. We expect that a subset of these peaks correspond to the color patches from the shirt. By finding a mapping of these peaks to the peaks of our color model, we can identify the colors in the image corresponding to the patches. Once the identity of the colors are computed we solve the same white balance alignment problem as before, mapping identified patch colors from the image to their corresponding mean colors of the Gaussian color model, $W = \arg\min_W \sum_k ||W\mu_k - \mu'_k||^2$.

To identify the colors corresponding to the shirt patches in the image, we compute
a 100 $\times$ 100 normalized RGB histogram of the shirt region and filter this histogram with a $\sigma = 1$ Gaussian kernel. We suppress non-local-maxima, which yields a distinctive set of peaks $\{q_i\}_i$, a subset of which corresponds to each of the color patches. We correspond a subset of these peaks to the expected peaks $\{p_i\}_i$ from our color model, generated by projecting the center of each color Gaussian into normalized RGB $p_i = NRGB(\mu_i)$. The correspondence process is brute force. For each subset of three peaks from the image, we match three peaks from our model and solve for a $2 \times 3$ affine transformation matrix $A$ mapping the two sets of 2-D peaks $p_i = Aq_i, i \in \{1, 2, 3\}$. We then test this alignment matrix $A$ on all of the peaks, selecting the matrix with the minimal distance to each model peak.

$$d(A) = \Sigma_j \min_i ||Aq_i - p_j||^2$$

Given a correspondence between the peaks, we can determine the colors in the image that match the selected peaks. Thus we have reduced the problem to finding the best white balance matrix as before. We solve for the $3 \times 3$ matrix $W$ that best maps the identified patch colors to their corresponding Gaussians. This entire initialization procedure takes approximately ten seconds (See Figure 5-5).

### 5.3 Online Analysis

The online analysis takes as input the colorful cropped region corresponding to the shirt (Chapter 5.1). We roughly classify the colors of this region using our color model and white balance estimate. The classified result is used to estimate the pose from a database. Next, we use the pose estimate to refine our color classification, which is used in turn to refine the pose. Lastly, we update our current estimate of the white balance of the image (Figure 5-6).
5.3. ONLINE ANALYSIS

Figure 5-5: We crop and compute the normalized RGB histogram of the shirt region. The peaks of the NRGB histogram are distinctive and easy to extract. We brute force align the observed peaks (squares) with the NRGB coordinates of the color patches of our Gaussian color model (circles). This gives us a coarse initial classification of the image, which we use to bootstrap our white balance estimate to produce the final classification.

5.3.1 Step 1: Color classification

We white balance the image pixels $I_{xy}$ using a $3 \times 3$ matrix $W$. In general, $W$ is estimated from the previous frame, which we will explain in Step 5. For the first frame in a sequence, we use the user-labeled initialization (§ 5.2). After white balancing, we classify the colors according to the Gaussian color models $\{(\mu_k, \Sigma_k)\}_k$. We produce an id map $r_{xy}$ defined by:

$$
 r_{xy} = \begin{cases} 
 \arg\min_k \|WI_{xy} - \mu_k\|_{\Sigma_k} & \text{if } \|WI_{xy} - \mu_k\|_{\Sigma_k} < T \\
 \text{background} & \text{otherwise}
\end{cases}
$$

where $\|\cdot\|_\Sigma$ is the Mahalanobis distance with covariance $\Sigma$, that is: $\|X\|_\Sigma = \sqrt{X \Sigma^{-1} X}$, and $T$ is a threshold that controls the tolerance of the classifier. We found that $T = 3$ performs well in practice, that is, we consider that a pixel belongs to a Gaussian if it
Figure 5-6: After cropping the image to the shirt region, we white balance and classify the image colors. The classified image is used to estimate the upper-body pose by querying a precomputed pose database. We take the pose estimate to be a weighted blend of these nearest neighbors in the database. The estimated pose can be used to refine our color classification, which is converted into a set of patch centroids. These centroids drive the inverse kinematics (IK) process to refine our pose. Lastly, the final pose is used to estimate the white balance matrix for the next frame.

is closer than three standard deviations to its mean. In addition we use a background subtraction mask to suppress false-positives in the classification.

Most of the time, the above white balance and classification approach suffices. However, during a sudden change of illumination our white balance estimate from the previous frame may no longer be valid. We detect this case when less than 40% of the supposedly foreground pixels are classified. To overcome these situations, we maintain a list of previously encountered reference illuminations expressed as a set of white balance matrices $W \in \mathcal{W}$. When we detect a poor classification, we search among these reference matrices $\mathcal{W}$ for the one that best matches the current illumination. That is, we re-classify the image with each matrix and keep the one that classifies the most foreground pixels.

5.3.2 Step 2: Pose estimation

Once the colors have been correctly identified as color ids, we can estimate the pose with a data-driven approach as described in Chapter 4.

For upper-body tracking, we precompute a database of 80,000 poses that are selected by uniformly sampling a large database spanning a variety of upper-body configurations and 3-D orientations. We rasterize each pose as a tiny id map $r^i$. At
run time, we search our database for the ten nearest neighbors \( \mathbf{f}^i \) of our classified shirt region, resized as a tiny 40 × 40 id map. We take our pose estimate to be a weighted blend of the poses corresponding to these neighbors \( \mathbf{q}^i \) and rasterize the blended pose \( \mathbf{q}^b \) to obtain an id map \( \mathbf{r}^b \). This id map is used in Step 3 to refine the classification and Step 4 to compute inverse kinematics (IK) constraints.

### 5.3.3 Step 3: Color classification refinement

Our initial color classification (Step 1) relies on a global white balance. We further improve this classification by leveraging the rasterized pose estimate \( \mathbf{r}^b \) computed in Step 2. This makes our approach robust to local variations of illumination.

We use the id map of the blended pose \( \mathbf{r}^b \) as a prior in our classification. We analyze the image pixels by taking into account their measured color \( I_{xy} \) as before and also the id predicted by the rasterized 3-D pose \( \mathbf{r}^b \). To express this new prior, we introduce \( d_{xy}(\mathbf{r}, k) \), the minimum distance between \( (x, y) \) and a pixel \( (u, v) \) of the rasterized predicted prior with color id \( k \):

\[
d_{xy}(\mathbf{r}, k) = \min_{(u, v) \in S_k} \| (u, v) - (x, y) \|
\]

with \( S_k = \{(u, v) | \mathbf{r}_{uv} = k\} \)

With this distance, we define the refined id map \( \mathbf{r} \):

\[
\hat{r}_{xy} = \begin{cases} 
\arg\min_k \| WI_{xy} - \mu_k \|_{\Sigma_k} + C d(\mathbf{r}^b, k) \\
\text{if } \| WI_{xy} - \mu_k \|_{\Sigma_k} + C d(\mathbf{r}^b, k) < T \\
\text{background otherwise}
\end{cases}
\]

We set the influence of the prior term \( C \) to 6/s where \( s \) is the scale of the cropped shirt region. The classifier threshold \( T \) is set to five.

We compared the strength of our pose-assisted color classifier with the Gaussian color classifier by varying the classification thresholds and plotting correct classifica-
tion versus incorrect classifications (Figure 5-7). This additional information significantly improves the accuracy of our classification by removing impossible or highly improbable color classification given the pose estimate.

![Comparing classifier accuracy](image)

**Figure 5-7:** Our pose-assisted classifier classifies more correct pixels at a lower false-positive rate than the baseline Gaussian classifier discussed in Step 1.

### 5.3.4 Step 4: Pose refinement with inverse kinematics

We extract point constraints from the newly computed id map \( \hat{r} \) to refine our initial pose estimate \( q^b \) using inverse kinematics (IK) as described in Chapter 4.3.

For each camera \( i \), we compute the centroids \( c^{ki} \) of each patch \( k \) in our color-classified id map \( \hat{r}^i \). We also render the pose estimate \( q^b \) as an id map and establish correspondences between the rendered centroids of our estimate and the image centroids. We seek a new pose \( q^* \) such that the centroids \( c^{*i} \) of its id map \( r^{*i} \) coincide with the image centroids \( c^{ki} \) (See 5-8). We also want \( q^* \) to be close to our initial
5.3. ONLINE ANALYSIS

Figure 5-8: For each camera, we compute the centroids of the color-classified id map $r^i$ and correspond them to centroids of the blended nearest neighbor to establish inverse kinematics constraints.

guess $q^b$ and to the previous pose $q^h$. We formulate these goals as an energy:

$$q^* = \arg\min_q \sum_{i,k} ||c^i(q) - c^{ki}||_{\Sigma_c}^2 + ||q - q^b||_{\Sigma_b}^2 + ||q - q^h||_{\Sigma_h}^2$$

where the covariances matrices $\Sigma_c$, $\Sigma_b$, and $\Sigma_h$ are trained off-line on ground-truth data similarly to Wang and Popović [68]. That is, for each term in the above equation, we replace $q$ by the ground-truth pose and $q^h$ by the ground-truth pose at the previous frame, and compute the covariance of each term over the ground-truth sequence.

5.3.5 Step 5: Estimating the white balance for the next frame

As a last step, we refine our current estimate of the white balance matrix $W$ and optionally cache it for later use in case of a sudden illumination change (Step 1). We create an id map from our final pose $q^*$ and compute a refined $W^*$ matrix using the same technique as in Section 5.2. We use $W^*$ as initial guess for the next frame. We also add $W^*$ to the set $\mathcal{W}$ of reference illuminations if the minimum difference to each existing transformation in the set is greater then 0.5, that is, if: $\min_{W \in \mathcal{W}} ||W^* - W||_F > 0.5$ where $\| \cdot \|_F$ is the Frobenius norm.
5.4 Discussion

In this chapter we have described a system capable of color garment tracking in real-world environments. Specifically, we introduced a robust method for locating the color garment using histogram search and presented an adaptive color classification model. We discussed the initialization of our Gaussian color model, adapting the model to new environments, coping with mixed lighting using a pose prior, and adjusting to changes in local lighting with adaptive white balance. These features allow our system to function in a variety of environments as demonstrated in the next chapter.

Even with the proposed techniques of this chapter, tracking in colorful environments can still fail when the background is very similar to the shirt colors. This is a fundamental limitation of working in the visible spectrum, and there will always be environments where background colors prohibit usage of our technique. In general, this problem can be ameliorated by further narrowing the foreground color model. Our current system adapts an existing color model to a new environment, which assumes that a white balance matrix is sufficient to map one illumination environment to another. A more sophisticated system would dynamically build a new color model while tracking in the new environment. A color model built in the new environment inherently captures more subtle color variations due to illumination and is likely to be more discriminative than a remapped color model.

In future work, we would also like to exploit the known colors of the shirt to calibrate the lighting environment. In theory, we should be able to acquire the colors and positions of the light sources in the scene based on the perceived colors of the shirt. Recovering the colors and positions of the illuminants would allow us to generate an even more accurate color classification model.
In this chapter we describe the experimental setup and validations of our tracking system. Our primary mechanism for validation was by tracking recorded sequences of representative motions in various environments. In the case of our hand tracking system, we could not easily obtain ground truth data on interesting sequences, and thus computed the match quality of our projected hand estimates and the image data. For the upper-body tracking system, we instrumented our color shirt with retroreflective markers for a Vicon motion capture system capable of millimeter precision tracking. The markers did not significantly occlude the color patches of our shirt, and we were able to compare our technique against the Vicon tracking on the same motion.

Our hand tracking sequences primarily validate the database-driven pose estimation algorithms described in Chapter 4. We tested various hand motions such as jiggling of the fingers and sign language, but limited our setting to a relatively monochrome office environment with a single light source. For our upper body tracking system, we validated our work on robust color and pose estimation described in Chapter 5. We focussed on testing various environments where our color shirt received mixed or changing illumination such as a basketball court, a squash court, and outdoors.
6.1 Experimental Setup

We begin by describing the experimental setup for the hand tracking and upper body tracking systems.

6.1.1 Hand Tracking

We use a single Point Grey Research Dragonfly camera with a 4 mm lens that captures $640 \times 480$ video at $30$ Hz. The camera is supported by a small tripod placed on a desk and provides an effective capture volume of approximately $60 \times 50 \times 30$ cm (See Figure 1-1).

We use the Camera Calibration Toolbox for Matlab\(^1\) to determine the intrinsic parameters of the camera. Color calibration is performed either by scribbling ground truth labels on a single image or by moving the hand to the center of the camera view and performing automatic color initialization (See Section 5.2).

![Figure 6-1: In addition to color calibration, we calibrate the desk plane by asking the user to click on four points of an 8.5 \times 11 inch sheet of letter paper. We also calibrate the scale of the hand by asking the user to hold his hand flat against the known desk plane.](image)

In addition to calibrating the camera, we also determine the desk plane by asking the user to click on the four corners of a sheet of paper of known size. We solve a four point pose estimation problem to determine the transformation of these points with respect to the camera (See Figure 6-1).

\(^1\)http://www.vision.caltech.edu/bouguetj/calib_doc
Hand Shape Variation

We use a 26 degree of freedom (DOF) 3-D hand model: six DOFs for the global transformation and four DOFs per finger. We also calibrate the global scale of the hand model to match the length of the user's hand. This can be done automatically once the desk plane has been determined, by asking the user to hold his hand flat against the desk. We iteratively try twelve hand sizes of between 170 cm and 290 cm long. For each hand size, we estimate the height of the virtual hand using our IK solver and choose the size that best estimates the hand to be 1 cm (or resting) above the desk.

![Figure 6-2: We tested several subjects performing the same sequence of motions. The image error of the tracked sequences varied by less than 5% across the subjects.](image)

We also tested three subjects with thinner versus pudgier hands without noticing any differences in tracking accuracy. When asked to perform the same sequence of motions involving both rigid movements and jiggling of fingers, the image error of the tracked sequences varied by less than 5% (See Figure 6-2).

More substantive testing of different hand sizes and shapes would be useful, and if necessary, we can allow a new user to explicitly select from several prototypical hand shapes.
6. RESULTS

6.1.2 Upper Body Tracking

Our body tracking stereo system is also designed with low cost and fast setup in mind. We used two Logitech C905 cameras that generate $640 \times 480$ frames at 30 Hz. The cameras are set atop of two tripods placed roughly two meters apart (See Figure 1-2). We geometrically calibrated the cameras with a $60 \text{ cm} \times 60 \text{ cm}$ ($2 \text{ ft} \times 2 \text{ ft}$) checkerboard using a standard computer vision technique [75]. We color calibrated the cameras by scribbling ground truth color labels on a single frame for each camera or by prompting the user to stand in the center of the camera view and performing automatic color initialization (See Section 5.2). The entire setup time typically takes less than five minutes.

While the commodity USB webcams we use are plugged into the same machine, they are not synchronized—they have no such function—and the frames we receive may be off by as much as 30 ms. Nonetheless, our algorithm is robust enough to produce the results demonstrated in the video. We disabled the webcams built-in auto-gain and auto-white-balance settings.

Human Torso Variation

We have tested our system on four subjects (three male and one female) with different torso shapes. We asked each subject to exercise the degrees of freedom of his torso and did not notice a qualitative difference in quality. The subjects ranged in height from 158 cm to 187 cm and in mass from 52 kg to 76 kg. We used a single database for all of the subjects with a 3-D mesh taken directly from the “Simon” model from the Poser 7 software. Only at the pose refinement step did we substitute a scaled mesh (parameterized by arm length and shoulder width) that better fits the particular subject. For subjects with entirely different torso shape, the database may be resynthesized in less than two minutes using graphics hardware. A production system could ship with several databases corresponding to different torso shapes.
6.2 Hand Tracking Results

We evaluate the tracking results of our system on several ten second test sequences composed of different types of hand motions (See Figures 6-3, 6-4, 6-5 and 6-6). We show that our algorithm can robustly identify challenging hand configurations with fast motion and significant self-occlusion. Lacking ground truth data, we evaluated accuracy by measuring reprojection error and jitter by computing the deviation from the local average $d(q^i, \frac{1}{|S|} \sum_{j \in S} q^{i+j})$ of a five-frame window $S = \{-2, \ldots, 2\}$ (See Figure 6-7).

![Figure 6-3](image)

**Figure 6-3:** Representative frames from the rigid hand tracking sequence. Our prediction is superimposed on the input image.

![Figure 6-4](image)

**Figure 6-4:** Representative frames from the free jiggling hand tracking sequence.

As expected, inverse kinematics reduces reprojection error by enforcing a set of corresponded projection constraints. However, there is still significant jitter along the optical axis. By imposing the temporal smoothness term, this jitter is heavily penalized and the tracked motion is much smoother.
While temporal smoothing reduces jitter, it does not eliminate systematic errors along the optical axis. To visualize these errors, we placed a second camera with its optical axis perpendicular to the first camera. We recorded a sequence from both cameras, using the first camera for pose estimation, and the second camera for validation. In our validation video sequence, we commonly observed global translation errors of between five to ten centimeters (See Figure 6-8). When the hand is distant from the camera, the error can be as high as fifteen centimeters. We attempt to compensate for this systematic error in our applications by providing visual feedback to the user with a virtual hand. While our system was designed to be low cost, the addition of a second camera significantly reduces depth ambiguity and may be a good trade-off for applications that require higher accuracy.
6.3 Body-Tracking Vicon Validation

We evaluated our two-camera system in several real-world indoor and outdoor environments for a variety of activities and lighting conditions. We captured footage in a dimly lit indoor basketball court, through a glass panel of a squash court, at a typical office setting, and outdoors (Figure 6-9). In each case, we were able to setup our system within minutes and capture without the use of additional lights or equipment.

To stress test our white balance and color classification process, we captured a sequence in the presence of a mixture of several fluorescent ceiling lights and a tungsten floor lamp. As the subject walked around this scene, the color and intensity...
Figure 6-8: Even when our estimate closely matches the input frame (left), monocular depth ambiguities remain a problem, as shown from a different camera perspective (right).

of the incident lighting on the shirt varied significantly depending on his proximity to the floor lamp. Despite this, our system robustly classifies the patches of the shirt, even in the event when the tungsten lamp is suddenly turned off. Unlike other garment tracking techniques [49, 70, 68], our method dynamically white-balances the images, which makes it robust to these lighting variations (See Figure 6-10). We show in the companion video that this procedure is critical to the success of our approach.

We evaluated the accuracy of our system by simultaneously capturing a sequence containing a variety of movements with a 16 camera Vicon motion capture system and our two-camera system. We applied a standard correction step to the Vicon data to fill gaps, smooth trajectories, and manually correct marker mislabelings due to occlusions. We also compared our results to a version of our method which lacks the pose prior and adaptive white balancing steps of our approach. The data from both methods are left unprocessed to avoid any bias.

On simple sequences without occlusions, both methods perform well. However on faster motions and in presence of occlusions, our algorithm can be twice as accurate (Figure 6-12). On average, our method has RMS error of 4.0 cm with is 20% better than without the pose prior and adaptive white balancing (5.0 cm). Because RMS is often an insufficient measure of visual quality [2], we provide the plot of the shoulder joint angle that confirms that our method is closer to the ground-truth data.
Figure 6-9: We demonstrate motion capture in a basketball court, inside and outside of a squash court, at the office, outdoors and while using another (Vicon) motion capture system. The skeleton overlay is more visible in the accompanying video.

(Figure 6-13), as well as a video of the corresponding captured motions. A visual comparison shows that our approach faithfully reproduces the ground truth motion whereas disabling the pose prior and adaptive white balancing steps yields significant jittering. We also compared the two methods on a jumping jacks sequence in which the arms are moving quickly. Our method correctly handles this sequence, whereas disabling the pose prior can result in losing track of the arms (Figure 6-14 and companion video).

Our system runs at interactive rates for both the hand-tracking and upper-body tracking data sets. On an Intel 2.4 GHz Core 2 Quad core processor, our current implementation processes each frame, consisting of two camera views, in 120 ms, split roughly evenly between histogram search, pose estimation, color classification, and IK. The histogram search, color classification and database lookup steps are computed in parallel on separate threads for the two images. The resulting blended nearest neighbor estimates and constraints are then combined in the inverse kinematics optimization which is computed on a single thread. We achieve approximately 9 Hz with a latency of 130 ms for the two camera upper-body tracking sequences using
CHAPTER 6. RESULTS

Color Classifications

Pose Estimates

Our Method  No White Balance  No Prior  No Prior, No White Balance

Figure 6-10: The pose prior and white balancing described in Chapter 5 have a significant effect in natural scenes with mixed lighting, like this basketball court. Here, we show the color classification mistakes that arise when we drop the pose prior, the white balance, and both.

two threads of our quad core processor.

For the single camera sequences used in hand-tracking, single-threaded performance of our system is approximately 100 ms per frame. We have also experimented with pipelining on our multi-core processor to increase throughput (frame rate). We can achieve 10 Hz (and 15 Hz on two threads) with a latency of 110 ms for single camera tracking.

Our current implementation is written almost entirely in Java. Certain pieces of the image processing and pose estimation use the Intel Matrix Kernel Library for performance. However, we expect that a fully-optimized C++ implementation may perform considerably better than our research prototype.

6.4 Discussion

We have shown above that our color-based tracking can accurately capture a variety of hand and upper body motions in real-world environments. However, our system is not perfect, and we outline the most common sources of tracking error in this section.

The most common source of error is patch misidentification, the incorrect identification or localization of the color garment patches in an input image. This is usually the result of mistakes in color classification. While the pose prior and white balancing steps (See Chapter 5) significantly reduce color classification mistakes, mistakes still
Figure 6-11: We show one frame of the validation sequence, where we have ground truth from a Vicon motion capture system. The pose prior and white balance steps produce a better color classification and pose estimate.

Figure 6-12: We compare the accuracy (RMS of all mesh vertices) between a simple system without pose prior nor adaptive white balance akin to [68] and our approach. In absence of occlusion, both methods perform equivalently but on more complex sequences with faster motion and occlusions, our approach can be nearly twice more precise. On average, our method performs 20% better.

occur and are ultimately unavoidable in colorful environments. In challenging conditions with motion blur, mixed lighting, or a dynamic background, these mistakes lead to spurious constraints for the inverse kinematics algorithm.

Another source of color classification mistakes is motion in the foreground. Because the face and legs of the tracked character are both in the foreground and similar enough in color to the orange patch on the shirt, they can survive background subtraction and be misclassified. One solution to this problem is to add the face and legs to a “blacklist” color histogram. Any colors falling within this narrowly defined
Figure 6-13: We plot the angle of a shoulder joint for ground-truth data captured with a Vicon system, our method, and our method without the pose prior or white balance steps (akin to [68]). Our results are globally closer to the ground truth and less jittery than those of the simple method. This is better seen in the companion video.

Figure 6-14: Despite the significant motion blur in this frame, we are still able to estimate the patch centroids and the upper-body pose.

histogram can be automatically discarded from classification. This histogram can be built automatically from the calibration frame as the relative locations of the face and legs with respect to the shirt are highly predictable.

A third source of patch misidentification errors comes from ambiguity of the patch given a correctly classified color. In both our glove and shirt pattern designs, each of the ten colors is shared by two patches. While the color assignment procedure (see Section 3.3) minimizes the chance of color collision between nearby patches, collisions can still occur. When they do, the identity of the patch corresponding to the classified color is ambiguous.
We believe that patch identification can be improved with a more sophisticated pose model. Presently, we make hard decisions about the identity and location of each patch. Incorrect decisions lead to jerky or jittery motions similar to those exhibited when we skip the pose prior and white balancing steps. A more sophisticated approach that maintains multiple hypotheses and integrates past motion information [61] could yield better results at the cost of additional computation.

Another type of pose estimation error occurs when a relevant body part is occluded. For instance, when an arm passes behind the body in a profile view, we can no longer recover the location of the arm. Because our pose estimation algorithm does not use information from future frames, the error of the arm can accumulate for as long as the arm is occluded. Similarly, fingers are often occluded behind the palm when the hand makes a fist. While the temporal smoothness term (See §4.3.1) of our IK optimization prevents the pose from rapidly diverging, errors can still become significant for long periods of occlusion.

We evaluated our motion captured sequence where the user picks up a box, moves it across the scene, and stretches. In 5.0% of the frames, there is a color misclassification error that causes a pose estimation error. The majority of these cases were due to classifying something in the background as one of the shirt colors. In 0.6% of the frames, there is a patch identification problem due to the ambiguity of the correctly classified color. In 5.2% of the frames, there is a pose estimation mistake due to occlusion. Occlusion errors had the most impact on the accuracy of the system.

Ultimately, the solution to both the patch misidentification and occlusion problems is to use more cameras. Misidentified patches due to misclassified background pixels can be culled through information from stereo constraints from several cameras. Occlusion is also less likely to occur given several viewpoints. Modern motion capture systems typically use between eight and sixteen high-speed cameras to accurately track a set of retro-reflective markers. Our system may still be an attractive alternative if we can provide good results with only three or four cameras.

Despite the shortcomings and limitations of our approach, we demonstrate several applications compatible with the accuracy of our current prototype in the next
chapter. These applications range from controlling an animated character using hand tracking to squash analysis (in a squash court) using upper body tracking.
Chapter 7

Applications

7.1 Applications for Hand Tracking

We demonstrate our system as a hand tracking user-interface with several applications. First, we show a simple character animation system using inverse-kinematics. Next, we use the hand in a physically-based simulation to manipulate and assemble a set of rigid bodies. We also demonstrate pose recognition performance with a sign language alphabet transcription task. For bimanual tracking, we demonstrate control of a flight simulator using a virtual yoke and a trackball controller for rotating and scaling a 3-D model.

7.1.1 Driving an Animated Character

Our system enables real-time control of a character's walking motion. We map the tips of the index and middle fingers to the feet of the character with kinematic constraints (See Figure 7-1) [60]. When the fingers move, the character’s legs are driven by inverse kinematics. A more sophisticated approach could learn a hand-to-character mapping given training data [18, 32]. Alternatively, Barnes and colleagues [5] demonstrate using cutout paper puppets to drive animated characters and facilitate story telling. We hope to enable more intuitive and precise character animation as we reduce the jitter and improve the accuracy of our technique.
7.1.2 Manipulating Rigid Bodies with Physics

We demonstrate an application where the user can pick up and release building blocks to virtually assemble 3-D structures (See Figure 1-1). When we detect that a block is within the grasp of the hand, it is connected to the hand with weak, critically damped springs. While this interaction model is unrealistic, it does suffice for the input of basic actions such as picking up, re-orienting, stacking and releasing blocks. We expect that data-driven [34] or physically-based [44, 30] interaction models would provide a richer experience.

We encountered several user-interface challenges in our rigid body manipulation task. The depth ambiguities caused by our monocular setup (See Figure 6-8) result in significant distortions in the global translation of the hand. The user must adjust his hand motion to compensate for these distortions, compromising the one-to-one mapping between the actions of the real and virtual hand. Moreover, it was difficult for the user to judge relative 3-D positions in the virtual scene on a 2-D display. These factors made the task of grabbing and stacking boxes a challenging one. We found that rendering a virtual hand to provide feedback to the user was indispensable. Shortening this feedback cycle by lowering latency was important to improving the

---

Figure 7-1: We demonstrate animation control (a) and a sign language finger spelling application with our hand tracking interface.
usability of our system. We also experimented with multiple orthographic views, but found this to complicate the interface. Instead, we settled on using cast shadows to provide depth cues [29].

7.1.3 Sign Language Finger Spelling

To demonstrate the pose recognition capabilities of our system, we also implemented an American Sign Language finger spelling application [55]. We perform alignment and nearest-neighbor matching on a library of labeled hand poses (one pose per letter) to transcribe signed messages one letter at a time. A letter is registered when the pose associated with it is recognized for the majority of ten frames. This nearest-neighbor approach effectively distinguishes the letters of the alphabet, with the exception of letters that require motion (J, Z) or precise estimation of the thumb (E, M, N, S, T). Our results can be improved with context-sensitive pose recognition and a mechanism for error correction.

7.1.4 Flight Simulator

We have also built a rudimentary flight simulator and used our hand tracking system to control a virtual yoke. While our virtual yoke control does not deliver usability improvements for flight control, it is more engaging than a mouse and keyboard and is more portable than a special purpose flight simulator joystick. We envision that our system can be used to replace specialized controllers like flight joysticks or steering wheels for portable gaming (See Figure 7-2).

7.1.5 Trackball Controller

We also built a simple trackball controller that uses both hands to rotate, translate and scale a 3-D model (See Figure 7-3). We experimented with several gesture mappings for this task. One issue we encountered was detecting the start and finish of an operation, such as a rotation. Initially, we used a closed fist to signify the start of a rotation, and mapped the rotation and translation of the model directly to the trans-
We demonstrate a virtual yoke controller for a flight simulator. Opening the fist marked the end of the operation. However, we found that it was easy to inadvertently rotate the hand when opening or closing quickly. Our tracking is also less accurate when none of the fingers are visible, as is the case for the fist.

We eventually settled on a pinching gesture consisting of pressing the index finger and thumb together, forming a loop. Pinching is both easier to perform for the user and more accurate to track for our algorithm because most of the fingers are visible. We use pressing and releasing the pinch to signify the start and end of an operation.

Our pinching gestures are inspired by prior work using Fakespace’s Pinch Gloves [40] to detect discrete contacts between the thumb and fingers. Pinch Gloves have been combined with Polhemus 6 DOF sensors and head mounted displays to facilitate bimanual manipulation [13] and navigation [10] for virtual environments.

Pinch detection has also been used in the context of bare-hand tracking for signaling the beginning and end of operations. Wilson uses background subtraction and connected component analysis to detect the distinct oval-shaped component formed by pinching [72]. Pinch detection is simpler in our case because the index finger and thumb tips are clearly marked with colored patches. We use the distance between the two patches to detect pinching.

In our system, pinching with both hands and rotating them about a virtual axis rotates the model about that axis. Using the virtual rotation described by both
7.1. APPLICATIONS FOR HAND TRACKING

Figure 7-3: We provide a trackball-like interface for the user to rotate and scale a 3-D model.

Hands afford us more precision than the rotation of a single hand. Pinching with both hands and moving them closer together or further apart results in a scaling operation. We restrict the user to only one operation (rotation or scaling) at a time, as it can be difficult to perform a rotation without scaling otherwise. Pinching with only one hand and moving that hand translates the model in 3-D (See Figure 7-4).

Figure 7-4: We combine motion with clicking to define rotation, scaling, and translation.
7.2 Upper-Body Tracking for Sports Analysis and Gaming

We demonstrate possible uses of upper body tracking on two sample applications. The "squash analysis" software tracks a squash player; it enables replay from arbitrary viewpoints and provides statistics on the player's motion such as the speed and acceleration of the arm (Figure 7-6). The "goalkeeper" game sends balls at the player who has to block them. This game is interactive and players move according to what they see on the control screen (Figure 7-5). These two proof-of-concept applications demonstrate that our approach is usable for a variety of tasks, it is sufficiently accurate to provide useful statistics to an athlete and is effective as a virtual reality input device.

Figure 7-5: We show two frames of a simple goalkeeper game where the player controls an avatar to block balls. We show the view from each camera alongside the game visualization as seen by the player.
Figure 7-6: We demonstrate a motion analytics tool for squash that tracks the speed and acceleration of the right wrist while showing the player from two perspectives different from those captured by the cameras.
Conclusion

8.1 Contributions

In this thesis, we have described a general technique for using color garments to assist pose estimation using one or two commodity webcams. Our technique is low cost and easy to setup. It can be deployed in a variety of environments in a matter of minutes. We have applied our technique in the context of hand tracking and upper body tracking using one or two cameras.

We have shown that custom color garments can lead to robust and efficient algorithms to difficult computer vision problems. The colorfulness of the garment is exploited in our histogram-based search algorithm. The distinctiveness of the garment as seen from different poses leads to our data-driven pose estimation technique. The known pattern of colors simplifies the white balance problem to allow operation in difficult illumination.

In the context of hand tracking, we have introduced a user-input device composed of a single camera and a cloth glove. We demonstrated this device for several canonical 3-D manipulation and pose recognition tasks. We have shown that our technique facilitates useful input for several types of interactive applications.

In the context of upper body tracking, we have demonstrated a lightweight practical motion capture system consisting of one or more cameras and a color shirt. The system is portable enough to be carried in a gym bag and typically takes less than five
minutes to setup. Our robust color and pose estimation algorithm allows our system to be used in a variety of natural lighting environments such as an indoor basketball court and an outdoor courtyard. While we use background subtraction, we do not rely on it and can handle cluttered or dynamic backgrounds.

For both the hand and upper body tracking applications, our system runs at interactive rates, making it suitable for use in virtual or augmented reality applications. We hope that our low-cost and portable system will spur the development of novel interactive motion-based interfaces and provide an inexpensive motion capture solution for the masses.

8.2 Future Directions

There are many possible extensions to our system. Our system can be combined with multi-touch input devices to facilitate precise 2-D touch input. We can introduce physical props such as a clicker or trigger to ease selection tasks.

We can also improve the accuracy of our system by improving the pattern on our garment. When designing our garment, we restricted our pattern to consist of large patch features due to the problems with robustly detecting small patch features in the presence of occlusion and motion blur. However, small features would offer more precise constraints for our inverse kinematics stage. Ideally we would be able to overlay a set of high-frequency (small) features on top of our low-frequency (large) features. If the small features do not interfere with our detection of the large ones, we could use our current method to bootstrap a more accurate detection step to search for the smaller features.

8.2.1 Bare-Hand Tracking

While we have limited our discussion in this thesis to the topic of color-based garment tracking, the proposed tiny image features and nearest neighbor pose estimation system may have applications in bare-hand tracking as well. While a single tiny image of a markerless bare hand may not be descriptive enough to determine hand pose,
several tiny images of the same hand taken from different perspectives may be sufficient (See Figure 8-1). Ambiguities in pose can also be eliminated by restricting the set of poses used by the application. We believe that a wide baseline stereo system combined with accurate background subtraction could forgo the glove altogether. Such a system would work well alongside the keyboard and mouse, as the user would not need to switch into 3-D manipulation by putting on a glove. However, we also anticipate that certain tasks such as fingertip localization and pinch detection will become more difficult again without the clearly marked fingertips on our custom glove [24, 35].

8.2.2 Usability Optimization

Adoption of our 3-D input system ultimately depends on the usability of the device. A number of studies have pointed out the difficulty of accurately selecting or moving objects with 3-D free space input [59]. Bérard and colleagues showed that users of 3-D input devices are comparably less efficient and more stressed at a 3-D translation task than users of a 2-D mouse [8]. Teather and colleagues demonstrate that latency and jitter, two problems common to many 3-D input devices, significantly degrade performance for selection tasks [63]. To address these issues, additional work needs to be done to optimize the usability of 3-D input. This may include ensuring that the arms are supported to minimize jitter, encouraging small motions to reduce fatigue, and optimizing the tracking system to lower latency (See Figure 8-1).

We can envision applications in computer animation and 3-D modeling, new desktop user-interfaces and more intuitive computer games. We can leverage existing design methods for hand interactions [60] to apply hand tracking to applications such as virtual surgery to virtual assembly.

8.2.3 Computer Aided Design

We believe that one particularly suitable application is Computer Aided Design (CAD) of mechanical assemblies. Designing mechanical assemblies can be tedious
due to the large amount of 3-D placement of parts. The parts are typically combined or "mated" by selecting and establishing relationships between surfaces on two parts. For instance, the surface of one part can be constrained to be coincident to the surface of another part. Presently, mechanical parts are placed and assembled almost entirely based on the relationships defined between them. These mating relationships can be inferred automatically if we had an input device that could allow the user to accurately place the parts in 3-D space. This would provide a user experience that is more similar to assembling a set of physical parts. We hope to demonstrate improvements to the efficiency and usability of CAD assembly through our 3-D input device (See Figure 8-1).

Figure 8-1: An early prototype of a system that incorporates several ideas for future direction of our work. We use a wide baseline stereo camera system to obtain two views of each hand. The working area of this system is relatively small and the arms are supported at all times. We intend to apply this system to 3-D manipulation tasks for CAD.

Exploring 3-D CAD assemblies is another application that is ripe for a new type of interface. In the absence of exploded view diagrams, a user typically pages through the hierarchy of parts in an assembly and toggles the visibility of each branch. This allows the user to isolate sub-assemblies and cull occluding parts. We can allow the user to accomplish this same goal by virtually disassembling the CAD model. The user can move enclosing and occluding parts out of the way just as he would when exploring a physical model. We believe that this disassembly process allows the user to better internalize the relationships and layout of the parts.

Finally, CAD presentation may also benefit from 3-D input. Presently, there is no
efficient mechanism for a user to present a 3-D model. The mouse is not an ideal tool for navigating or demonstrating 3-D CAD assemblies in a presentation setting. A typical user has difficulty manipulating 3-D entities, such as the camera or the parts of an assembly, with a 2-D mouse during a presentation. Currently, engineers will take screenshots of particular views of a model or prerecord transitions or animations ahead of time. Screenshots can be inflexible and unclear to the audience and animations are time consuming to create. We would like to explore manipulation of a 3-D virtual model using hand gestures similar to how one would manipulate a physical model.

The primary contribution of this thesis is a robust and low-cost user-input device. We have provided a software development kit and inexpensive gloves so that other researchers may develop imaginative applications of their own.
Bibliography


