Learning Full-Body Motions from Monocular Vision in Real-Time: Dynamic Imitation in a Humanoid Robot

Jeffrey B. Cole

A dissertation submitted in partial fulfillment
of the requirements for the degree of

Master of Science in Electrical Engineering

University of Washington

2007

Program Authorized to Offer Degree: Electrical Engineering

University of Washington
Graduate School

This is to certify that I have examined this copy of a master's thesis by

Jeffrey B. Cole

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____
Maya Gupta

_____
Rajesh P.N. Rao

Date: _____

In presenting this thesis in partial fulfillment of the requirements for a master's degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this thesis is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Any other reproduction for any purpose or by any means shall not be allowed without my written permission.

Signature_____

Date_____

University of Washington

Abstract

Learning Full-Body Motions from Monocular Vision in Real-Time: Dynamic Imitation in a Humanoid Robot

Jeffrey B. Cole

Chair of the Supervisory Committee:
Professor Maya Gupta
Electrical Engineering

In an effort to ease the burden of programming motor commands for humanoid robots, a computer vision technique is developed for converting a monocular video sequence of human poses into robot motor commands for a humanoid robot. The human teacher wears a multi-colored body suit while performing a desired set of actions. Leveraging the colors of the body suit, the system detects the most probable locations of the different body parts and joints in the image. Then, by exploiting the known dimensions of the body suit, a user specified number of candidate 3D poses are generated for each frame. Using human to robot joint correspondences, the estimated 3D poses for each frame are then mapped to corresponding robot motor commands. A set of kinematically valid motor commands is generated using an approximate best path search through the pose candidates for each frame. Video processing is performed with a distributed algorithm using a cluster of 25 computers allowing for real-time motor command generation. The viability of the approach is demonstrated by presenting results showing a full-body imitation of human actions by a Fujitsu HOAP-2 humanoid robot in real-time.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I would first like to thank my parents for giving me the support and encouragement to know that I can succeed in whatever I set my mind to. Everything that I know about hard work, dedication, critical thinking, and value for education comes directly from my parents. I am truly grateful for everything they continue to do for me throughout my life.

Next, I would like to thank my advisors Maya Gupta and Rajesh Rao for providing direction, guidance, and funding for my work. Their knowledge, patience, and encouragement have made my experiences in graduate school both rewarding and enjoyable.

I would also like to thank several colleagues for countless useful conversations and support. Particularly I am grateful to Scott Phillips for his selfless willingness to work with me on practically every technical problem that I came up against in this thesis. Through daily discussions, Scott has provided me with his critical insight and expertise in signal processing. My work would not have been possible without Scott's help and friendship. I am also grateful to Brandon Smith, Eric Garcia, David Grimes, Josh Storz, Pascal Clark, and Eric Rombokas for their friendship and support throughout my time in graduate school.

DEDICATION

To my parents who have always given me their full support.

Chapter 1
INTRODUCTION

Teaching complex motor behavior to a robot can be extremely tedious and time consuming. Often, a programmer will have to spend days deciding on exact motor control sequences for every joint in the robot for a pose sequence that only lasts a few seconds. A much more intuitive approach would be to teach a robot how to generate its own motor commands for gestures by simply watching an instructor perform the desired task. In other words, the robot should learn to translate the perceived pose of its instructor into appropriate motor commands for itself. This imitation learning paradigm is intuitive because it is exactly how we humans learn to control our bodies [1]. Even at very young ages, we learn to control our bodies and perform tasks by watching others perform those tasks. But the first hurdle in this imitation learning task is one of image processing. The challenge is to develop accurate methods for extracting 3D human poses from monocular image sequences.

The work presented in this thesis is an attempt to address the issues of motor command generation for robotic imitation and to explore the benefits and limitations of using only a monocular video sequence to automatically generate motor commands in real-time for a humanoid robot.

Chapter 2 provides the reader with some useful background information about imitation learning and pose estimation for humanoid robots. It also provides references for similar work in the literature. We then propose a new algorithm for converting monocular video sequences into motor commands for a humanoid robot in chapter 3.

In an effort to achieve real-time frame rates, we then turn our focus towards explor-

ing methods for distributing the algorithm among an arbitrary number of processors. Chapter 4 describes how the algorithm was redesigned so that a cluster of processors could work together to achieve arbitrarily high frame-rate analysis in real-time.

In Chapter 5, we show examples of the types of imitation learning that we were able to achieve using our proposed algorithm. Chapter 6 is used to present some alternative approaches and possible future improvements that were explored during the course of developing the algorithm described in this thesis. Finally, Chapter 7 presents conclusions along with a general discussion of the main contributions of our work to the field of imitation learning for humanoid robots.

Chapter 2
BACKGROUND

Imitation learning in humanoid and other robots has been studied in depth by a wide array of researchers. Early work such as [2, 3] demonstrated the benefit of programming a robot via demonstration. Since then researchers have addressed building large corpora of useful skills [4, 5, 6], handling dynamics [7, 8], studied biological connections [9], or addressed goal-directed imitation [10].

Typically a marker based motion capture system is used to estimate human poses as input for training robots to perform complex motions. This requires a full motion capture rig to extract the exact locations of special markers in a restricted 3D space. An instructor is typically required to wear a special suit with careful marker placement. The motion capture system then records the 3D position of each marker and recovers degree-of-freedom (DOF) estimates relative to a skeletal model using various inverse kinematic techniques. Due to careful calibration of the cameras, highly accurate pose estimates can be extracted using multi-view triangulation techniques.

The biggest downside to using a motion capture rig in our imitation learning scenario is that training can only be performed in a rigid (and expensive) environment. Also, the motion capture system is unsatisfying because it does not allow the robot to behave autonomously. In this thesis we demonstrate initial steps in allowing the robot to use its own vision system to extract the 3D pose of its instructor. This allows us to "close the loop" for the learning process. Using only its own eyes, a robot should be able to watch an instructor, convert what it sees into a sequence of 3D poses, and then translate that sequence into appropriate motor commands.

A large body of work has studied the problem of performing pose estimation

from vision. Early computational approaches [11, 12] to analyzing images and video of people adopted the use of kinematic models such as the kinematic tree model. Since these earliest papers many systems have been proposed for pose estimation and tracking (for examples see [13, 14, 15, 16]), yet none have significantly supplanted marker based motion capture for a broad array of applications.

The biggest limitation of many of these vision-based pose estimation techniques is that they require multiple, distant and often carefully calibrated cameras to be placed in a ring around the instructor. While more portable and less costly than a commercial motion capture rig this is still not desirable for autonomous robotic imitation learning. Thus in this thesis we propose a method which relies solely on the robot's own commodity monocular camera. We note that our work on monocular pose estimation builds on previous techniques for solving the human and limb tracking problem using learned image statistics [17, 18, 19, 20, 21].

Chapter 3

POSE ESTIMATION ALGORITHM

As an alternative to expensive and cumbersome motion capture systems, we have developed a new approach to estimating human poses using only a single, uncalibrated camera and a multi-colored body suit. The method uses a nonparametric probabilistic framework for localizing human body parts and joints in 2D images, converting those joints into possible 3D locations, extracting the most likely 3D pose, and then converting that pose into the equivalent motor commands for a HOAP-2 humanoid robot. The algorithm is highly parallelizable and thus we demonstrate how real-time analysis can be performed using a cluster of processors. The overall flow of the data processing is shown in Figure 3.1.

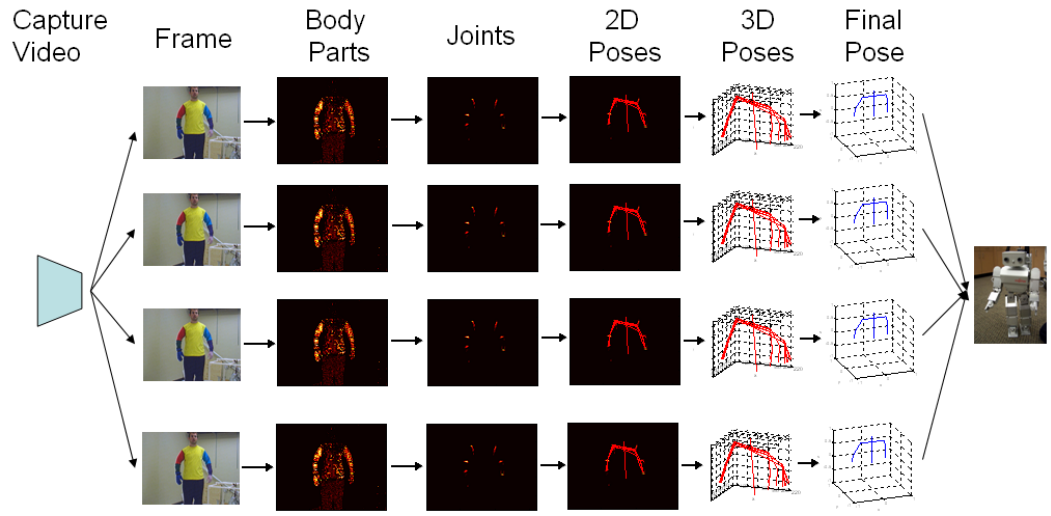

Figure 3.1: General overview of the proposed approach to pose estimation. Each column indicates a step in the proposed algorithm for converting a raw video sequence into a set of motor commands for the humanoid to perform.

3.1 Detecting Body Parts

The first step of the process is to detect where the different body parts are most likely located in each frame of the video sequence. Since we have granted ourselves the concession of using clothing with known colors, body part detection is done by training a classifier in RGB color space.

During the training phase, the user labels example regions from a single frame of the video sequence. The RGB values of the pixels in each region are then fit with Gaussian distributions and the 3D curve fit parameters are saved to a file. An example of hand selected upper body parts and their RGB color clusters are shown in figure 3.2.

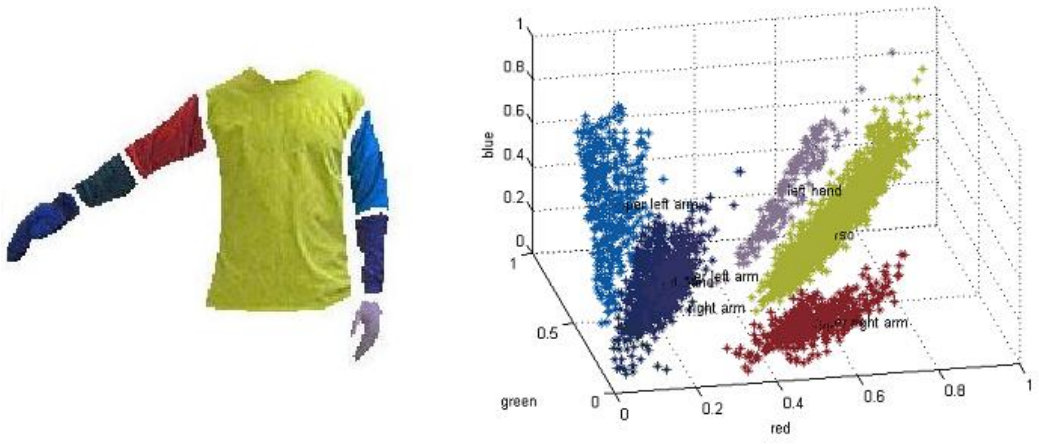

Figure 3.2: RGB training for body part detection. The left image shows hand selected body part regions and the right plot shows each body part's color clusters.

Once the color distributions have been learned for each body part, it is relatively fast and easy to detect the probable body part locations in any other frame from the sequence. For example, figure 3.3 shows the probability of each pixel being a member of a body part class, where intensity of the image encodes the relative likelihood. To increase the speed of this step of the algorithm, we downsample each dimension of the

RGB space from 256 to 50, reducing the number of possible colors from 16,777,216 to 125,000. We then precompute a lookup table for each bin's probability of being a member of each body part class. This allows the part location probability maps to be generated for each body part in each frame of the video sequence with very limited processing time. Processing time is also reduced by taking advantage of the fact that each pixel can be processed in parallel. Since all processors used in this work have dual cores, we were able to distribute body part detection for each frame, doubling the rate at which images are converted to body part probability maps.

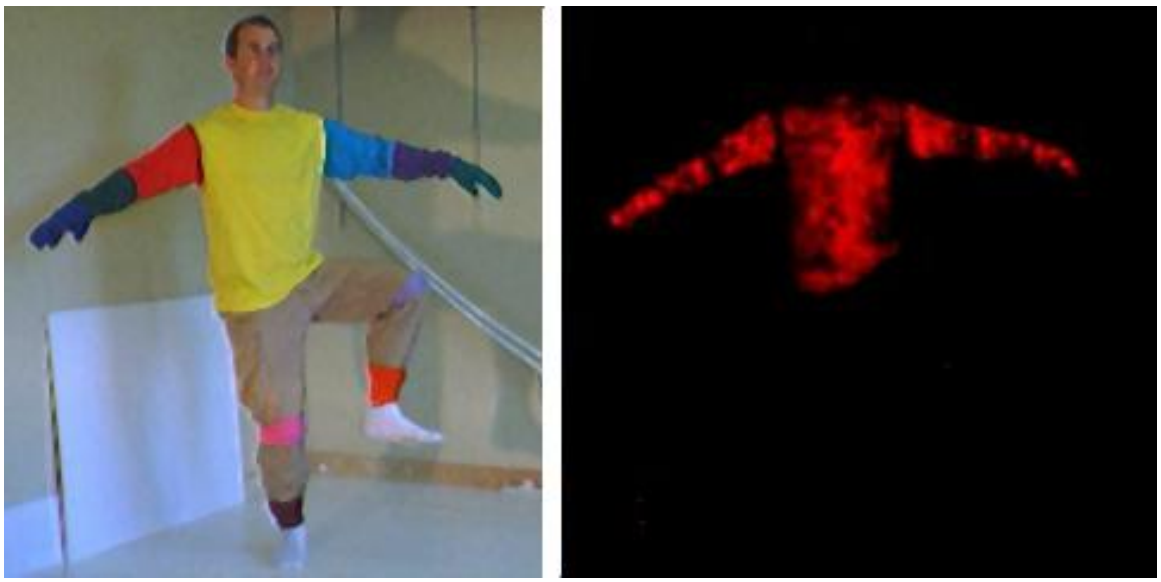

Figure 3.3: Probability map for the location of each upper body part in the given frame. The value assigned to each pixel in the map is found by evaluating the pixel's RGB values using the previously trained Gaussian distributions. Intensity of the image on the right indicates the relative likelihood of a pixel being a body part.

3.2 Converting Body Parts into 2D Joint Location Probability Maps

Once probability maps have been generated for each body part, we need to then use these maps to determine where the shoulders, elbows, wrists, hips, knees, and ankle

joints are located. We expect a body joint to be located in the highest probability overlap region between the two body parts that meet at the joint.

To be more formal, if we have a probability distribution function (PDF), $P_{upper}(x, y)$, for the location of the upper right arm and we have a PDF, $P_{lower}(x, y)$, for the location of the lower right arm, then we will expect the right elbow joint to be located at the pixel coordinates that satisfy the equation:

$$\underset{x,y}{\operatorname{argmax}}(P_{upper}(x, y) \times P_{lower}(x, y)) \tag{3.1}$$

However, when working with the actual body part location maps generated, we find that there are usually no single pixels which exhibit high probability of membership in any two body part classes, and full regions of overlap for body parts are practically non-existent. In other words, most pixels only have negligible probability of membership to a single body part class. In order to generate reasonable overlap regions we first perform a slight spatial blurring of the body part maps. Also, we do not want to generate a single point for each joint in the body as that would not allow us to be robust to occlusions or noise. Instead, we would rather generate a new non-parametric PDF for each joint location that can be sampled to generate discrete candidate human poses.

So, for every pair of body parts that are connected by a joint, the system performs two steps to generate a joint location probability map. First, each body part probability map is spatially blurred with a Gaussian kernel with a variance of 1 pixel. To speed up processing this blurring is performed in the frequency domain using FFTs. Then, for every pair of body parts that are connected by a joint, the spatially blurred body part maps are multiplied together and the resulting map is normalized so it is a valid PDF for the current joint. The resulting maps characterize the most likely locations for each of the instructors joints in the current 2D video frame. An example of a 2D joint location probability map is shown in figure 3.4.

Much like body part detection, the joint location probability map generation step

is highly parallelizable. Utilizing the dual core processors in our system, we were able to halve our processing time by distributing the processing among both cores.

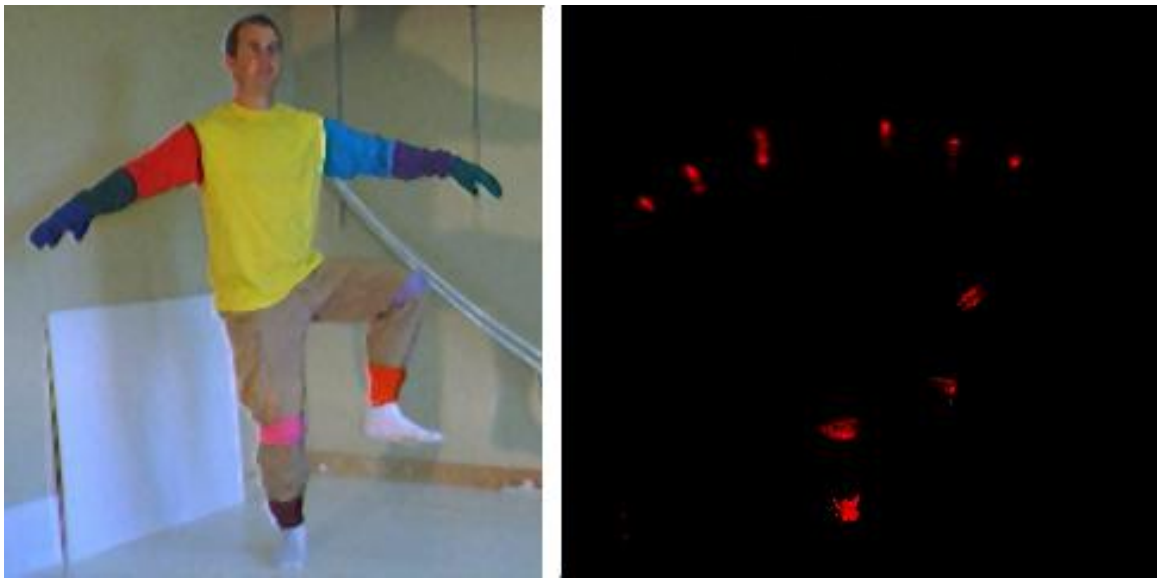

Figure 3.4: Example of a probability map for the 2D locations of each joint for the video frame shown on the left. Joint maps are found by multiplying together blurred versions of each of the body part maps.

For the work described herein, the lower body joint localization was done directly through color detection unlike the upper body where full parts were detected first and then converted into joint locations. The differences in processing of the lower body and upper body are meant to illustrate two varying methods for joint localization. Detecting the joints directly from color is much faster but is more likely to result in joint locations being lost due to self occlusions throughout the video sequence. The technique used on the upper body is more robust to occlusions as there is a larger region of color to detect and the likelihood of full occlusion of a body part is much lower than occlusion of a joint. However the processing time required is considerably higher when body part locations need to be converted into joint locations.

3.3 Sampling 2D Poses From The Joint Maps

The next step the system takes is to randomly sample N different 2D poses from the joint location distributions. The sampling is done with replacement using the non-parametric PDF of each joint to control the sampling. In other words, we use the PDF of each joint to generate N likely $\{X, Y\}$ coordinates for that joint. All the candidate 2D joint locations are grouped together to make N candidate poses. The poses thus generated are a collection of the most likely poses estimated from a single frame. Figure 3.5 shows an example of 50 2D poses sampled from a set of joint distributions.

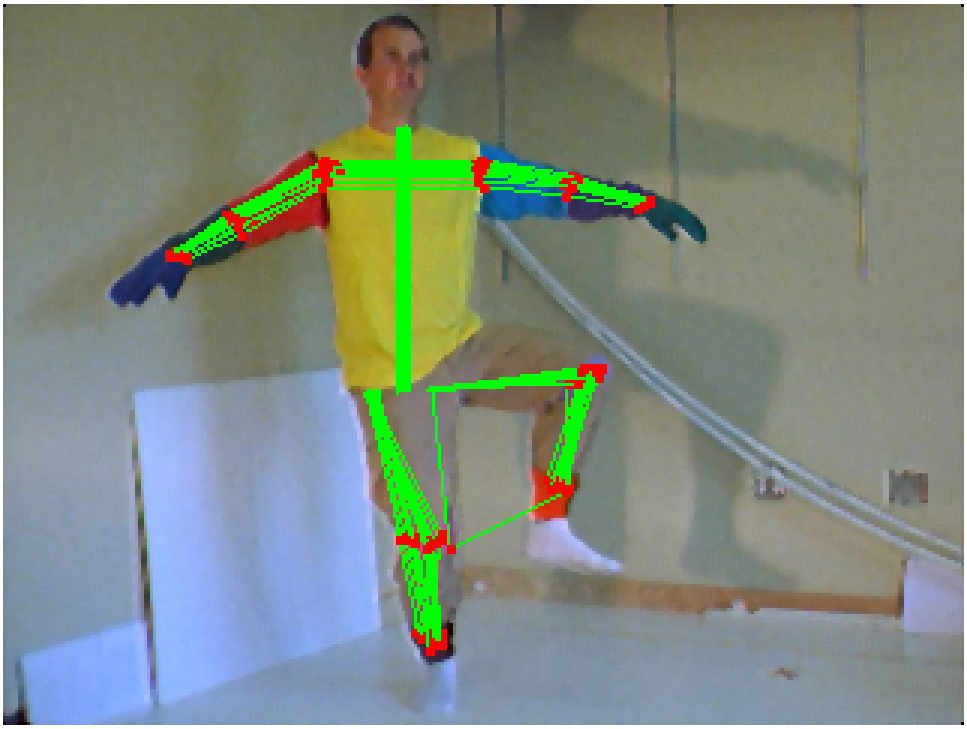

Figure 3.5: Example of 50 2D poses sampled from the joint distribution maps. Red dots indicate sampled joint locations and the green lines show which joints are connected in each sample pose.

3.4 Converting 2D Poses into 3D Poses

Converting the 2D poses into poses in 3D space is done by detecting foreshortening and requires that we exploit the approximate known dimensions of the human body. In this system, all body part lengths are measured with respect to the length of the torso. This helps make the system more robust and allows the trainer to be any distance from the camera. In our course model of the human body, the shoulder line is 0.6 times the length of the torso, the upper arms are 0.4 times the length of the torso, and the lower arms are 0.35 times the length of the torso. However, this model could be extended to the case of multiple human instructors by learning probability distributions over the lengths rather than a single proportional length.

Converting a given 2D pose into 3D is thus a matter of figuring out how far forward or backwards each joint needs to move in order to make each body part the correct length in 3D space. For example, if the upper left arm is measured to be length $D_{measured}$ in the current 2D pose and the upper left arm is supposed to be length D_{true} in 3D space, then the left elbow could either be forward or backwards the distance D_{offset}, where

$$D_{offset} = \pm\sqrt{D_{true}^2 - D_{measured}^2}. \tag{3.2}$$

The limitation of using foreshortening to generate candidate 3D poses is that the user cannot bend forward at the waist during the video sequence or the normalization factor will be thrown off. The user can, however move in any other manner desired. The user can freely move any distance from the camera. Also, if the user is not facing the camera (or even with his back to the camera) the system will detect the foreshortened shoulder width and still be able to generate 3D poses.

As an illustrative example, a top down view of how a single 2D upper body pose can be converted into 64 (2^6) possible 3D poses is shown in Figure 3.6. Figure 3.7 shows a frontal view of the results of 2D to 3D conversion using the above described method.

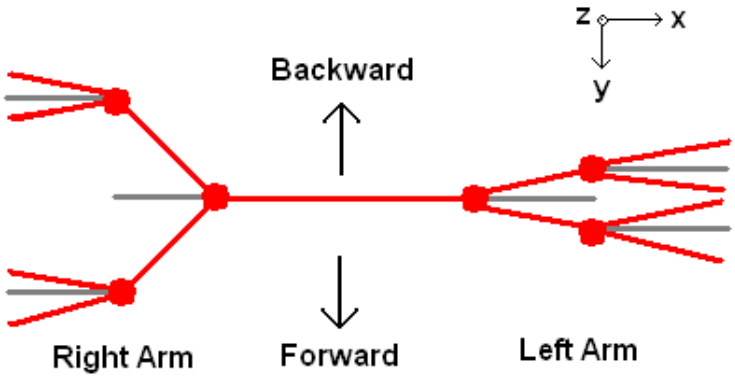

Figure 3.6: This figure shows a top down view of how a single 2D upper body pose would be converted into 64 different possible 3D poses. Grey lines indicate the measured length of the 2D pose body parts and the red lines indicate the possible poses in 3D.

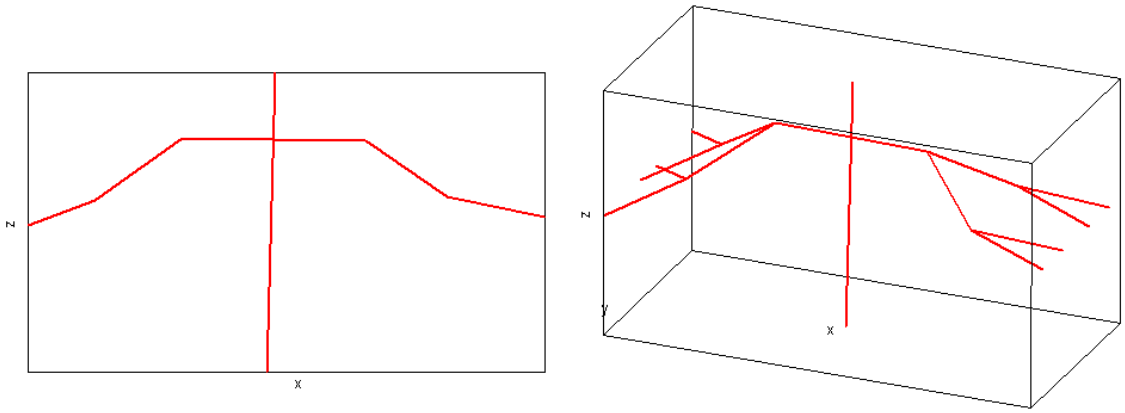

Figure 3.7: This figure shows results for a single 2D pose (left) converted into all possible 3D poses (right).

3.5 Converting 3D Human Poses into Robot Angles

The robot's upper body has 8 degrees of freedom (3 for each shoulder and one for each elbow) and the lower body has 12 degrees of freedom (3 for each hip, 1 for each knee, and 2 for each ankle). Each degree of freedom is controlled by a servo motor. We use position-based control so motor commands are simply joint angles from an initial "rest" state.

Converting each of the 3D poses into the corresponding angles for the robot joints is performed differently for the upper body and lower body.

The upper body angles are found directly using linear transformations. Starting with the upper left arm, the system detects the amount of forward/backward rotation in degrees, saves that angle, and then simulates a rotation of all of the left arm joints about the shoulder using the negative of the found angle. This procedure is carried out for each of the degrees of freedom until all of the joints have been rotated back to their initial state. Thus, after finding all the angles required to get the 3D pose to its zero state, we have all the motor commands the robot needs to perform to get to the current 3D pose.

During this conversion from joint locations to angle space, we can take advantage of our known model of the instructor's dynamic capabilities. Any time that an angle which is outside of a human's natural motion is detected, the entire pose is thrown out and the system immediately moves on to process the next candidate 3D pose. For example, if the system finds that the robot will have to bend its elbow backwards to get its hand to the desired 3D location, it can reject the entire candidate pose. Essentially we have imposed a hard prior on the expected motor commands for a human instructor. This both saves processing time and greatly reduces the number of 3D poses that are generated for the given frame.

Unlike the upper body, the lower body angles are solved using inverse kinematics and an iterative optimization. To find the angles that generate each of the desired 3D

leg positions for a given pose, the degrees of freedom are adjusted iteratively using the Nelder-Mead Simplex Method until the ankle and knee locations converge to the desired 3D points.

The discrepancy between the upper and lower body processing techniques is due to the different motor configurations for arms and legs on the HOAP-2 humanoid. Ambiguities that arise from the motor configurations in the robot hip made it impossible to isolate the hip angles serially as was done with the upper body angles. The direct technique used on the upper body is much faster than the iterative technique used on the lower body.

3.6 Finding the Smoothest Path Through the Frames

After performing all the steps listed above, the system is inevitably left with a fair number of possible poses (motor commands) it could send to the robot for any given frame in the sequence.

Initially, we tried to implement a full tree search to look forward a few frames and decide which complete path would be the smoothest. We define smoothness as the minimum sum of Euclidean distances between the motor commands sent to the robot over an entire sequence of poses. However, an exhaustive search of the space proved to be very computationally intensive as the branching factor of the tree is quite large (usually between 10 and 300 poses per frame). Finding the best path by searching through all possible poses for even a modest 5 future frames for a real-time system would potentially become unmanageable with today's current processor technology.

To bypass this issue we only keep a finite number, M, of the smoothest paths as we search forward in time through the space. (For the results shown in this thesis M was set to 10.) As each frame is processed, we reassess the best M paths through the candidate poses. Any pose that is not included in one of the best paths is thrown out so as to reduce the search size for the next frame. This trimming method is similar to a forward Viterbi search except that our set of possible states in each frame is not

discrete and so we are forced to pick an arbitrary number of best paths.

Almost inevitably, we eventually get to a point in time where all M paths agree on the best pose to use for a given frame in the past. Once this agreement is reached by all M paths, the motor commands for that frame can be sent on to the robot.

It is important to note here that we are not guaranteed to have convergence on a single pose for each frame of a finite video sequence. However, as we step forward in time processing frames, we become less and less likely to have disagreement about the best poses in the past. As a practical measure in this algorithm, we set a limit as to how many frames, L, the robot is allowed to lag the instructor before making a decision as to the preferred pose to assume. So even if the M best paths still have not converged on a single pose for the frame that we received L time steps in the past, we will force the system to just choose the pose that is associated with the best path at that time. For the results presented in this thesis, L was set to 5 frames.

Chapter 4
DISTRIBUTED PROCESSING

Ideally, a robot that is capable of learning through imitation should be able to process visual data in real-time. A long delay in processing times would mean that the robot would either have to use a video frame-rate that is unacceptably slow to capture the nuances of human motion, or the entire gesture would have to be prerecorded and then processed offline before imitation could be performed. Both scenarios are not satisfying because they limit the types of gestures that can be imitated and make real-time feedback in the learning process impossible. If instead, the robot was capable of working at arbitrarily fast frame rates, then real life motions that require high frame rates such as swinging a golf club could be accurately analyzed and interactively imitated.

4.1 Single Processor Limitations

Our initial efforts at implementing the algorithm described in chapter 3 were performed using a single core processor clocked at 1.75 GHz on images with a resolution of 320x240 pixels. At that speed, we were able to achieve a frame rate of 3 frames per second after a considerable number of coding tricks were applied to the algorithm. However, frame rates of at least 15 to 30 Hz would be needed to accurately capture most standard human motions in real-time.

4.2 Redesigning for Multiple Processors

In an effort to achieve real-time frame rates, we turned our focus towards exploring methods for distributing the algorithm among an arbitrary number of processors.

The first step to redesigning any system to exploit multiple processors is to identify which steps in the algorithm could be performed in parallel. In our case, the conversion of a single frame of video to a set of candidate poses does not require any information from other frames of the sequence. The only step that needs to be performed serially in our algorithm is the search for the smoothest set of motor commands through the set of candidate 3D poses (as described in section 3.6). Thus it is possible to let each frame of the video sequence be processed by an independent processor and then use a single processor for the final best-path search.

For the work presented in this thesis, we performed all processing using a Linux cluster of 25 dual-core 1.7 GHz processors. One processor was used as the data distributor (D_{node}), another was used as the candidate poses collector (C_{node}), and the rest of the nodes were used as general worker nodes (G_{nodes}) for converting single frames of video into candidate poses. An overview of the distribution/collection process is presented in figure 4.1. All code was written in C++ and all communications between nodes was made using TCP socket send and receive commands.

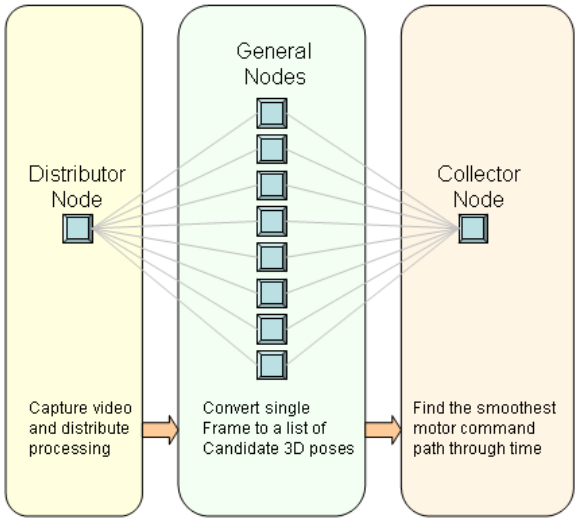

Figure 4.1: Schematic showing how the processing is distributed to a pool of nodes that send their results to a collector node for final analysis.

4.3 Distributor Node

The distributor node, D_{node}, functions primarily as a task manager for the general worker nodes. It is designed to act as both a server (receiving frames from the camera) and a client (sending video frames to individual worker nodes). Initially the distributor node sits idly waiting for the first frame of video to be received. When the first frame is sent from the camera, the D_{node} checks to see which of the G_{nodes} is available for processing and then assigns that video frame to the available G_{node}.

4.4 General Processing Nodes

Like the D_{node}, the general processing nodes, G_{nodes}, function as both a server and a client. Their server task is to wait for and receive a frame of video from the D_{node}. Upon receiving an image, the G_{node} will follow all of the steps described in sections 3.1, 3.2, 3.3, and 3.5 to convert the image into a set of candidate motor commands.

When finished processing a frame, the G_{node} will then act as a client and send the resulting candidate motor commands to the collector node C_{node}. The G_{node} will also send a signal to the D_{node} to let it know that it is now available to process another video frame.

4.5 Collector Node

The collector node, C_{node}, is perhaps the most overworked node in the system. The C_{node} is responsible for receiving candidate poses from all the G_{nodes}, finding the smoothest path through time (as described in section 3.6), and sending final motor commands to the robot. The computational burden on the C_{node} is due to the serial nature of its processing task. For every frame that is received from a G_{node}, the C_{node} has to re-analyze the smoothest path through time and then decide whether to send a final pose to the robot to perform.

Chapter 5

RESULTS

A monocular vision real-time imitation learning system was constructed as described in chapters 3 and 4. The resulting work is a complete closed loop system with a humanoid robot capable of imitating a human instructor by processing monocular video at 15 to 20 frames per second on a distributed framework. As our work resulted in a functioning system, there is no single experimental result to present in this chapter. Instead we will show and discuss images from a number of live demonstrations we recorded upon completion of the system.

5.1 Real-time Demonstration with Instructor, Robot and Simulator

Figure 5.1 shows the setup for a demonstration that was performed in September, 2007. The instructor is shown standing in the center of a room. To his right is a graphical simulation of the robot projected on the wall and to the instructor's left is the real humanoid robot, suspended from the waste by a harness. A monocular camera (not visible in the image) is positioned in front of the instructor to act as the robot's eye. It should be noted that the monocular camera used to watch the instructor was taken directly from the robot's head and positioned in front of both the instructor and the robot only to make the demonstration visually easier to document.

The simulation of the robot was used to show how the robot visualized itself performing the gestures, unconstrained by the physics of self collision and gravity. As we will discuss, the final motor commands actuated by the robot sometimes differ from the desired motor commands generated by the system. Specifically, in cases where performing the gesture would result in self collision, the robot simply ignores

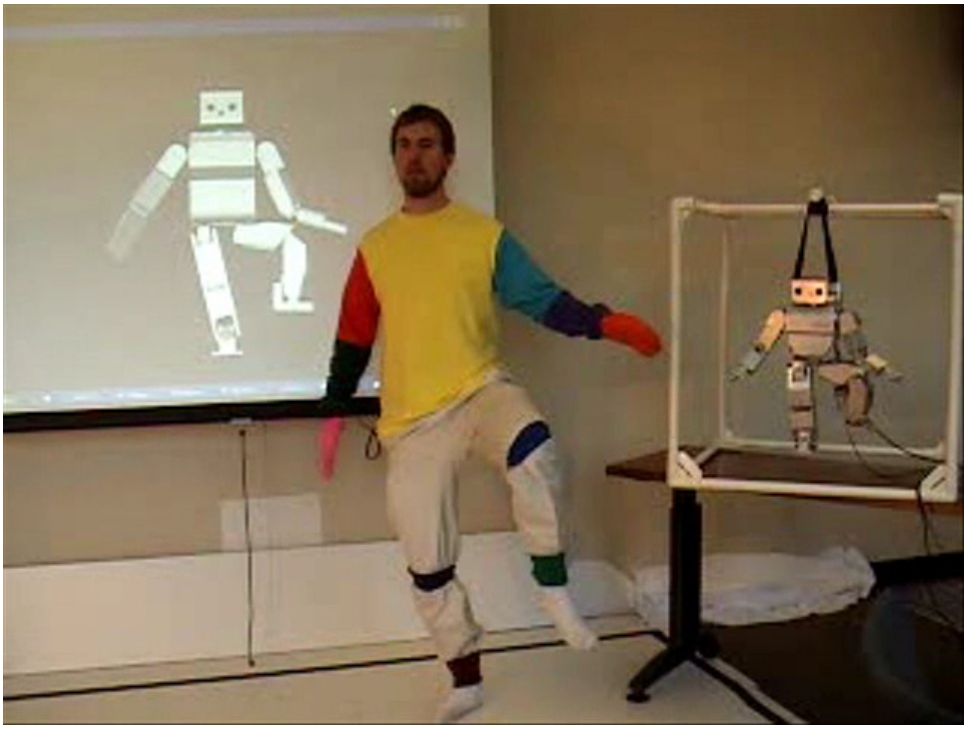

Figure 5.1: Setup for a full system demonstration of real-time imitation using the HOAP-2 humanoid robot.

the impossible gesture and waits for the next physically possible portion of the motor command sequence. The work presented in this thesis does not address issues of self collision or stability and balance.

The demonstration consisted of the instructor performing a number of gestures while the robot imitated the gestures with approximately one second of lag-time. For the first demonstrated gesture the instructor puts his arms out and raises his left leg and balances on one foot. Since the robot is attached to a waist harness, there are no complications from the issues of stability and balance, and after a one second delay, the robot imitates the leg lift in a visually satisfying manner. The second gesture performed is a kicking motion. Similar to the first gesture, this leg kick is imitated by the robot exactly as was planned by the algorithm without any self-collision issues or

balance corrections and results in a visually satisfying imitation of a leg kick motion by the humanoid robot.

For the third gesture in the demonstration, the instructor puts his left arm out to his side and swings his right arm in a whirlwind motion. In this case, the computer simulated version of the robot performs a similar gesture to the instructor. However, the motor commands generated would have caused the robot's right hand to collide with his torso so the gesture is aborted by the real robot.

The rest of the gestures presented in the video demonstrate successful imitation by the real robot and do not suffer from the self-collision issues of the previous arm motion. Gestures performed in the demonstration include a squatting motion, another leg lift, an arm in front of the body motion, and a waving motion of both arms similar to a hula dance.

5.2 Imitation Sequences with Intermediate Results

In a separate video demonstration, the robot imitates two gestures from the instructor and we present two figures that show intermediate results for each stage of the processing algorithm. The first gesture is primarily an upper-body gesture and is shown in figure 5.2. In this sequence, the instructor puts both of his arms in front of his torso and then ends by putting both of his arms above his head and waving. The second gesture performed is primarily a lower-body gesture and is shown in figure 5.3. In this sequence the instructor raises his left leg and balances on his right leg. Both figures show key frames of the visual input to the robot, body part detection results, joint location results, discrete sampled candidate poses, simulated imitation in a visualizer, and the resulting performance by the physical robot.

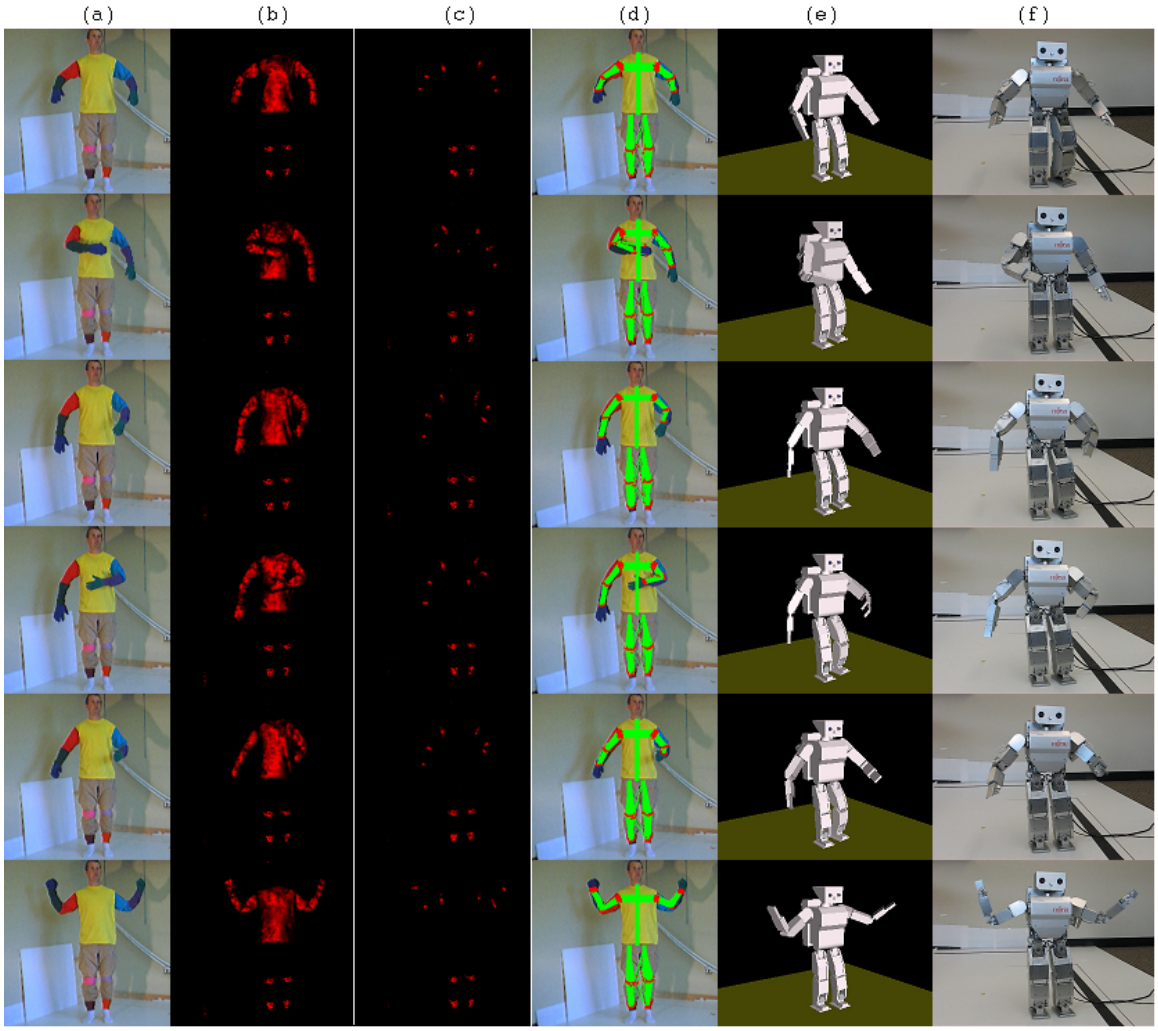

Figure 5.2: Results of pose estimation for a monocular video sequence of a primarily upper-body gesture. Column (a) shows the raw video sequence for a leg lift gesture. Column (b) shows the probability maps for body part locations in each frame of the sequence. Column (c) shows the probability maps for body joint locations. Column (d) shows candidate 2D poses sampled from the joint location maps. Column (e) shows a computer simulation of the robot performing the final motor commands. Column (f) shows the humanoid performing the resultant gesture.

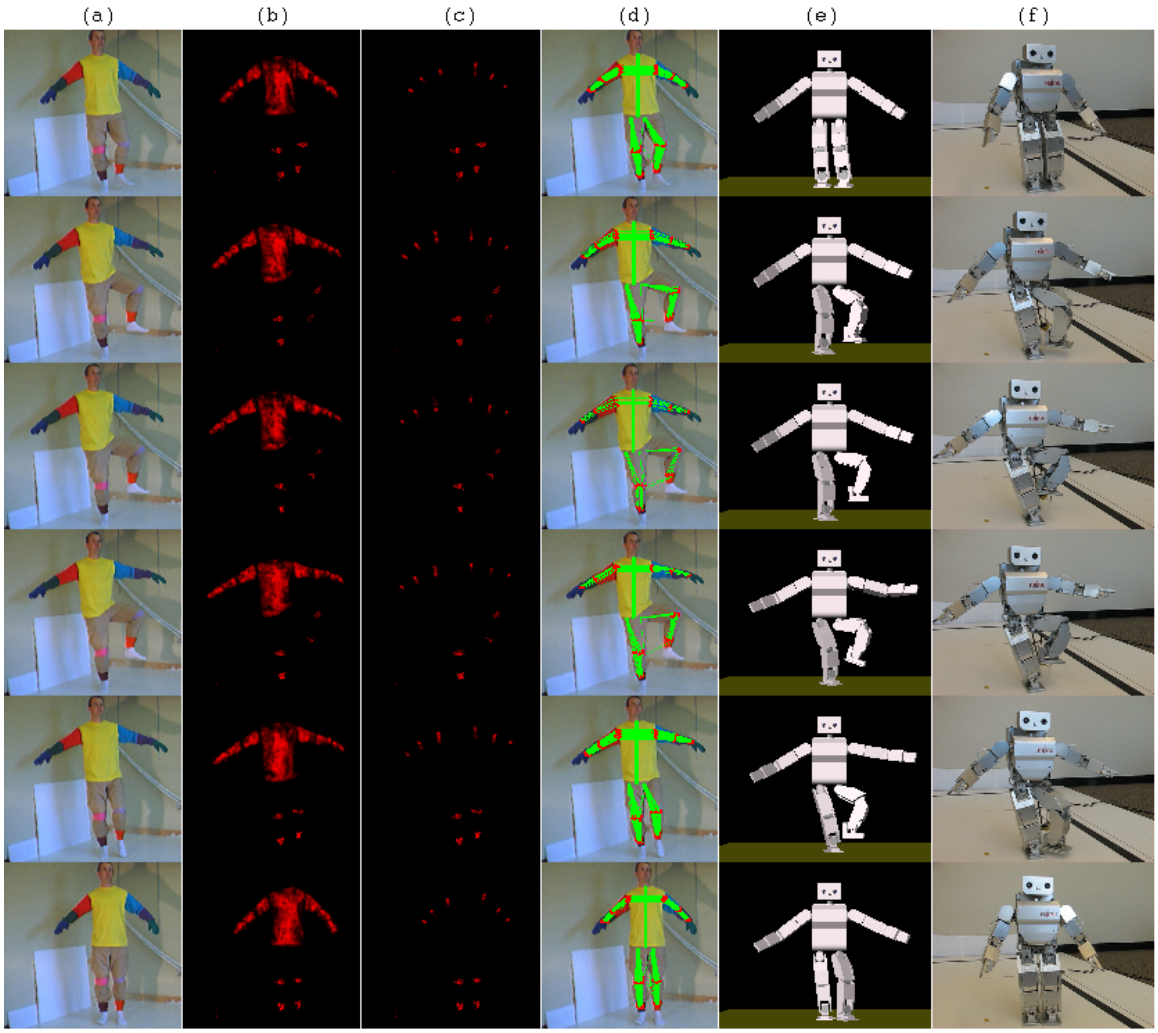

Figure 5.3: Results of pose estimation for a monocular video sequence of a leg lift gesture. Column (a) shows the raw video sequence for a leg lift gesture. Column (b) shows the probability maps for body part locations in each frame of the sequence. Column (c) shows the probability maps for body joint locations. Column (d) shows candidate 2D poses sampled from the joint location maps. Column (e) shows a computer simulation of the robot performing the final motor commands. Column (f) shows the humanoid performing the resultant gesture.

Chapter 6
ALTERNATE APPROACHES AND IMPROVEMENTS

The results presented in the previous chapter were generated using the algorithm as described up to this point. However, there have of course been a number of modifications and alternate approaches that were explored during the course of this work. In this chapter we present some of these alternate approaches and also present some ideas for ways that the algorithm could be improved in future work.

6.1 Alternate Color Spaces and Classification Techniques

Body part localization through color classification is an early and critical step in the imitation algorithm. For all results presented in this thesis, the image data was left in its raw RGB formats and body part training and classification were performed without any color transformations to alternate color spaces. However, when using the finished system, we did find that lighting changes in the room had noticeable deleterious effects on the ability to detect body parts.

An obvious, and relatively simple improvement to the work presented in this thesis, was to perform a transformation of pixel values into a color space, such as HSV or CIELAB, that is less affected by lighting changes in a scene. With this concept in mind we coded the system to be able to work in any 3-dimensional color space and initial testing was performed using both HSV and RGB spaces. However, practically no difference was seen in the ability to detect body parts in the video sequences and so, in an effort to reduce processing time, we decided to simply leave the pixel data in its raw RGB format.

We believe that a more fruitful modification to the body part detection would be

to use an alternate, more flexible, classifier instead of the single Gaussian classifier. As can be seen in figure 3.2, the actual color distributions from the training data are not particularly well fitted by simple mean and covariance values. This is especially true when training data contains color saturation clustering. Instead, it would likely be more reasonable to use a multivariate Gaussian, or a non-parametric method such as nearest-neighbors or support vector machines (SVMs) to better fit the data.

We used single Gaussian distributions because we believed it would require the minimal amount of processing time per pixel. However, as a lookup table of body part probabilities for all 125,000 possible quantized colors is precomputed during the training stage, the amount of time to process a single RGB value is constant for any classifier. The prior generation of a lookup table means that the increased processing time required for a more computationally intensive classifier will not change the processing rate during the real-time imitation, and this frees us to choose the best performing classifier with no concern for computational burden.

6.2 Time Smoothing of Joint Location Maps

One of the biggest limitations of our system is its inability to generate poses when only a limited number of body parts are visible in the monocular image. It is essential for a monocular vision system to be robust to partial occlusion of different joints as it is likely to occur frequently. Ideally, we need to design the algorithm so that even if the instructor puts his hand behind his back, or is partially blocked by other objects in the room, the system can still make a best guess at an appropriate set of motor commands for the robot to perform.

Often, a single joint is only occluded for a brief moment during the course of the instructor's gesture. For example, when performing a kicking motion, the left knee is temporarily blocked from view by the swinging right leg. This occlusion only lasts for a few frames and it is patently clear to a human viewer that the left leg has simply stayed in its previous location during the time period that it was not visible.

However, the final system presented in this thesis takes no measures to gracefully handle temporary occlusions. It instead simply skips any video frame where every joint on the body is not sufficiently well localized.

One solution to the issue of temporary occlusion is to use time smoothing of the joint location PDFs to allow the system to hallucinate joints when they are not visible. We performed some early experiments using the concept of joint location smoothing over time but the extra processing required and the inability to perform joint map generation in parallel left us with a system that could no longer run in real-time. As the final goal was to make a real-time imitation system, we were forced to disable joint smoothing in the final version of the demonstrated system. In the interest of completeness, we will present here this time smoothing portion of the algorithm which was eventually removed from the final system.

For every frame of the video, a joint location PDF is initially generated using the steps described in section 3.2. For time smoothing purposes we can assume that we now have access to the joint location PDFs for N frames in the future and N frames in the past. Thus, to improve the localization of the current joint map, P_t, we can modify it by multiplying it together with spatially blurred versions of past and future joint location PDFs. The blurring operation is a convolution with a Gaussian kernel h where the variance of the kernel σ is proportional to the time difference with the current time t. Thus, using the equation

$$P_t = \prod_{n=t-N}^{t+N} \left(P_n \otimes h_{\sigma \propto n}\right), \tag{6.1}$$

we can refine the PDF of the current time frame to take into account our estimates of likely joint locations in future and past frames and even hallucinate reasonable joint maps for frames with full occlusion. Figure 6.1, shows an example of how a joint location PDF for an occluded joint in the current frame could be successfully hallucinated through time smoothing.

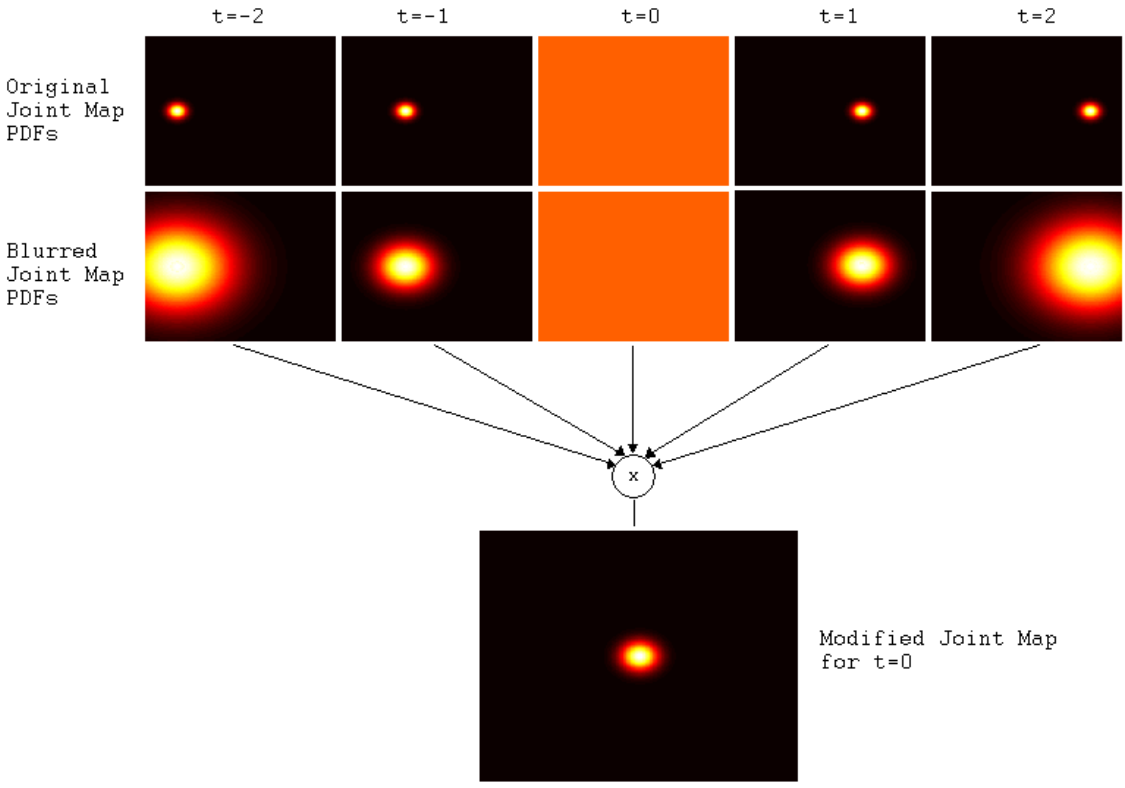

Figure 6.1: Example of time smoothing for joint location PDFs. Blurred versions of the joint maps for future and past times are multiplied together to hallucinate a reasonable joint location PDF for the occluded joint at time $t = 0$.

6.3 Visual Cues for Additional Depth Information

There are many visual cues in images besides length foreshortening that help us determine 3D information about a scene. Examples of these visual cues include shadow analysis and partial occlusion detection. In future improvements to the work in this thesis, it would be reasonable to improve the search for the final motor commands by taking into account these visual cues about the likely 3D locations of different joints.

Notably, if time smoothing of joint locations was implemented as described in the previous section, it would be relatively easy to detect relational occlusion between body parts and use that information to inform the final pose selection. For example, if we find that the smoothed PDF for a joint differs greatly from its original unsmoothed PDF and also has a high amount of overlap with another joint's PDF, then we can guess that the first joint is located behind the second joint. Then, when performing the path search through the possible motor commands in the final stage of the algorithm, weights could be applied to the candidate poses based on these visual cues so that poses that exhibit the expected depth relationship between the two joints are preferred.

Chapter 7
DISCUSSION

In this thesis we have described a fully functioning, closed-loop system for training a humanoid robot to imitate human gestures in real-time. To the best of our knowledge, this is the first time that an uncalibrated monocular vision system has been used in a relatively unconstrained environment to demonstrate real-time gesture imitation in a humanoid robot. Despite the successes of our work, it is important to discuss here the drawbacks that limit the usefulness of our system for training humanoid robots to perform gestures.

Perhaps the most glaring drawback of our work is the requirement that the instructor wear a colored body suit while teaching the robot to move. Admittedly, it would be ideal to remove this restriction, and there is a fair amount of work being done currently by other researchers to extract 3D human poses from video sequences that do not require colored clothing. But most of these algorithms are still in their infancy, and for the specific task of training a robot to perform gestures through watching an instructor, it is not unreasonable or overly burdensome to require that the instructor wear a specific shirt and pants while training the robot. Also, compared to the commonly used marker-based, multi-camera motion capture systems, our clothing requirements are quite manageable, our setup time is minimal and our environmental restrictions are practically non-existent.

A much greater theoretical drawback to our system is perhaps that it is overly dedicated to the concept that imitation of human actions should involve simply positioning a robot's joints at specific locations in space. It is hard to imagine practical situations where the goal of imitation is simply to look like a human. More realisti-

cally, we will be interested in teaching a robot to perform a task with no concern for the visual appearance of the robot's motions. For example, we may want to teach a robot to pick up an object and place it on a shelf. In this situation it would be of no use for the robot to simply perform lifting and placing gestures in an empty imitation of the instructor. Instead we want the robot to be able to infer the trainer's goals through interpretation of his behavior and then generate whatever motor commands are necessary to accomplish that goal.

With this realistic criticism in mind, we think that the usefulness of our system might lie more in using the 3D extracted pose information to classify and recognize different types of gestures. In this way, human gestures could be used to communicate with the robot or to prompt the robot to perform certain previously learned tasks.

BIBLIOGRAPHY

[1] R. P. Rao, A. Shon, and A. Meltzoff, "A Bayesian model of imitation in infants and robots," in *Imitation and Social Learning in Robots, Humans, and Animals*. Cambridge University Press, 2005.

[2] M. Y. Kuniyoshi and H. Inoue, "Learning by watching: Extracting reusable task knowledge from visual observation of human performance," *IEEE Transaction on Robotics and Automation*, vol. 10, no. 6, pp. 799–822, 1994.

[3] C. Atkeson and S. Schaal, "Robot learning from demonstration," pp. 12–20, 1997.

[4] K. Yamane and Y. Nakamura, "Dynamics filter - concept and implementation of on-line motion generator for human figures," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 3, pp. 421–432, 2003.

[5] T. Inamura, I. Toshima, and Y. Nakamura, "Acquiring motion elements for bi-directional computation of motion recognition and generation," in *Siciliano, B., Dario, P., Eds., Experimental Robotics VIII.* Springer, 2003, pp. 372–381.

[6] Y. Takahashi, K. Hikita, and M. Asada, "Incremental purposive behavior acquisition based on self-interpretation of instructions by coach," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003, pp. 686–693.

[7] Y. Katsu and N. Yoshihiko, "Dynamics filter-concept and implementation of online motion generator for human figures," *IEEE Transactions on Robotics and Automation*, vol. 19, pp. 421–432, 2003.

[8] Y. N. M. Okada, K. Tatani, "Polynomial design of the nonlinear dynamics for the brain-like information processing of the whole body motion," in *IEEE International Conference on Robotics and Automation*, 2002, pp. 1410–1415.

[9] A. Billard and M. Mataric, "Learning human arm movements by imitation: Evaluation of a biologically-inspired connectionist architecture," *Robotics and Autonomous Systems*, no. 941, 2001.

[10] S. Calinon, F. Guenter, and A. Billard, "On learning, representing and generalizing a task in a humanoid robot," *IEEE Transactions on Systems, Man and*

Cybernetics, Part B. Special issue on robot learning by observation, demonstration and imitation, vol. 37, no. 2, pp. 286–298, 2007.

[11] D. Hogg, "Model-based vision: A program to see a walking person," *Image Vision Computing*, vol. 5, no. 20, 1983.

[12] K. Rohr, "Towards model-based recognition of human movements in image sequences," *CVGIP: Image Underst.*, vol. 59, no. 1, pp. 94–115, 1994.

[13] D. M. Gavrila and L. S. Davis, "3-d model-based tracking of humans in action: a multi-view approach," in *CVPR '96: Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition (CVPR '96)*. IEEE Computer Society, 1996, p. 73.

[14] I. A. Kakadiaris and D. Metaxas, "Model-based estimation of 3D human motion with occlusion based on active multi-viewpoint selection," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition, CVPR*. Los Alamitos, California, U.S.A.: IEEE Computer Society, 18–20 1996, pp. 81–87. [Online]. Available: citeseer.ist.psu.edu/kakadiaris96modelbased.html

[15] S. X. Ju, M. J. Black, and Y. Yacoob, "Cardboard people: A parameterized model of articulated image motion," in *FG '96: Proceedings of the 2nd International Conference on Automatic Face and Gesture Recognition (FG '96)*. IEEE Computer Society, 1996, p. 38.

[16] L. Sigal, S. Bhatia, S. Roth, M. J. Black, and M. Isard, "Tracking loose-limbed people," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, vol. I, June 2004, pp. 421–428.

[17] D. Ramanan and D. A. Forsyth, "Finding and tracking people from the bottom up," in *Conference on Computer Vision and Pattern Recognition*, vol. II. IEEE Computer Society, June 2003, pp. 467–474. [Online]. Available: http://citeseer.ist.psu.edu/ramanan03finding.html

[18] V. Pavlovic, J. M. Rehg, T.-J. Cham, and K. P. Murphy, "A dynamic bayesian network approach to figure tracking using learned dynamic models," in *ICCV (1)*, 1999, pp. 94–101. [Online]. Available: citeseer.ist.psu.edu/pavlovic99dynamic.html

[19] C. Bregler and J. Malik, "Tracking people with twists and exponential maps," in *Conference on Computer Vision and Pattern Recognition*, vol. I. IEEE Computer Society, June 1998, pp. 8–15. [Online]. Available: http://citeseer.ist.psu.edu/context/2038698/0

[20] J. Lee, J. Chai, P. Reitsma, J. Hodgins, and N. Pollard, "Interactive control of avatars animated with human motion data," 2002. [Online]. Available: citeseer.ist.psu.edu/lee02interactive.html

[21] Y. N. Dongheui Lee, "Mimesis scheme using a monocular vision system on a humanoid," in *Robotics and Automation, 2007 IEEE International Conference on*, 2007, pp. 2162–2168.