Near Real Time Human Pose Estimation using Locality Sensitive Hashing

By Michael Betten
2010 Undergraduate Honors Thesis
Advised by Professor Hao Jiang
Computer Science Department, Boston College

Abstract: This thesis aims to develop a data set of poses to be used with locality sensitive hashing in human pose estimation, and then to determine the potential for real or near real time use of locality sensitive hashing in human pose estimation with that data set. Background subtraction is used to obtain a silhouette of the human pose from a single frame of video. The pose is then matched against the data set of poses using locality sensitive hashing. Results indicate that a data set can be developed to be used with locality sensitive hashing to match human poses in video and that near real time results are possible. The application of locality sensitive hashing for human pose estimation in video facilitates further discussion of the uses for pose estimation in video as well as the uses of locality sensitive hashing in fast feature matching.

Keywords: human pose estimation, locality sensitive hashing, fast feature matching, foreground detection

Contents

1. Introduction 3

2. Overview 5

3. Foreground Detection 6

   3.1. Frame Difference 6

   3.2. Approximate Median 7

   3.3. Mixture of Gaussians 8

   3.4. The Algorithm 9

4. Features 12

   4.1. Person Detection 12

   4.2. Shape Context 12

   4.3. Blob Extraction 13

5. Locality Sensitive Hashing 15

   5.1. Definition 15

   5.2. Data Set Development 16

   5.3. Feature Mapping 18

6. Results 20

7. Discussion 23

8. Future Work 25

9. References 26

1. Introduction

Human pose estimation is defined here as the determination of the human pose and extraction of useful information concerning the human pose either from an image or frame of video. This thesis examines the problem of detecting the seemingly infinite number of possible human poses. Human pose estimation from video sequences has various useful applications. Human computer interaction, animation development, and robotics are only a few of many areas that have the potential to benefit from fast and accurate human pose estimation.

Current techniques, however, are limiting. One method, such as that used to develop many motion capture datasets including that in [7], involves the use of markers placed on the body to capture motion data using specialized equipment. Other methods, such as [5], require multiple cameras. Techniques involving high dimension feature matching for pose estimation are limited as well. kd-trees [3], one such data structure popular in nearest neighbor searches, performs little better than linear time at high dimensions in what is known as the “curse of dimensionality”, nearly equivalent to simply comparing every point in the data set [1].

The method for human pose estimation explored here involves extracting a pose feature and then matching that feature to large data set of features using locality sensitive hashing. This technique only requires one video camera placed in a static location. Locality sensitive hashing, developed by Alexandr Andoni and Piotr Indyk in [1, 10], is used to hash features in such a way that the probability for collision is higher for features closer together than those further apart. A query feature can then quickly be matched by retrieving elements stored at the location returned by hashing the feature. With a large enough data set, fast and accurate pose estimation can be achieved.

The goal of this thesis is to determine the viability of using locality sensitive hashing for real or near real time human pose estimation in video. Because the effectiveness of locality sensitive hashing relies in part on the data set stored by locality sensitive hashing, a related goal, consequently, is to develop a data set to be used in human pose estimation with locality sensitive hashing. As will be discussed, a data set can be developed for use with locality sensitive hashing in human pose estimation, and near real time results can be achieved.

2. Overview

This thesis is made up of essentially two components, one being the development of a dataset to be used with locality sensitive hashing and the second being the analysis and testing of locality sensitive hashing with that dataset. While these two tracks have a great deal of overlap, there are some differences that should be noted before proceeding. The figure below gives an overview of these two components developed for this implementation of human pose estimation in video.

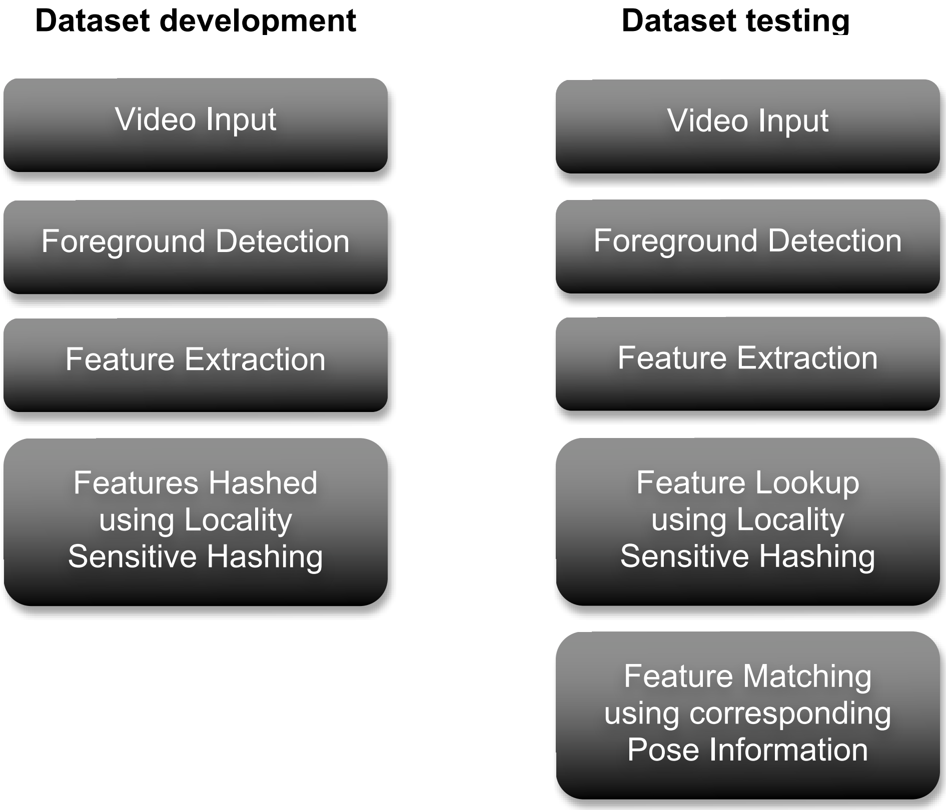

Dataset development
Video Input
Foreground Detection
Feature Extraction
Features Hashed using Locality Sensitive Hashing
Dataset testing
Video Input
Foreground Detection
Feature Extraction
Feature Lookup using Locality Sensitive Hashing
Feature Matching using corresponding Pose Information

Figure 1 – An overview of the two tracks of development for this thesis.

3. Foreground Detection

The first step in extracting a human pose from a single frame of video to be matched against the database of poses is to determine the useful information in that frame. Extracting useful and pertinent information is critical to the development of meaningful features and the success of the database query. As a result, accurate foreground detection, or the separation of information from the background of a scene, is critical. A number of different methods were implemented and analyzed before determining the most beneficial method for the purpose of this thesis.

3.1 Frame Difference

The frame difference technique is perhaps one of the simplest background subtraction techniques. Foreground information is extracted by finding the difference between the current frame and the previous frame [6]. A pixel is marked as foreground if that difference is greater than some predefined threshold.

$$\left| frame_i - frame_{i-1} \right| > threshold$$

But if an object does not move in the time between two frames, foreground information is lost. Just because the object does not move, however, does not mean there is no useful pose information present.

Another option for the frame difference method is to utilize a static background.

$$\left|frame_i - background\right| > threshold$$

Determining the difference between a frame and predefined background model returns much more useful information including interior pixels and information for non-moving objects. A drawback to this method is the need for a background model, often unavailable from most videos in uncontrolled settings.

3.2 Approximate Median

The approximate median method employs the frame difference technique but with a constantly updated background model, alleviating the need for a static background model. Median filtering, a precursor to the approximate median method, uses a background model that is the median of all previous frames [6]. The storage requirements for median filtering are alleviated in the approximate median method by comparing the current frame to a single frame background model and then updating the background model accordingly [11], described in Figure 2.

```
if $pixel_i$ in $frame_n$ > $pixel_i$ in $background$
        $pixel_i$ in $background$ = $pixel_i$ in $background$ + 1
elseif $pixel_i$ in $frame_n$ < $pixel_i$ in $background$
        $pixel_i$ in $background$ = $pixel_i$ in $background$ – 1
```

Figure 2 – A pseudo code algorithm for the approximate median method of foreground detection.

Approximate median develops a better background model as well as more foreground information without the use of a static background model.

There is, however, trailing data left by moving objects before the background model is updated, as well as much to be desired in terms of interior pixel consistency.

3.3 Mixture of Gaussians

The Mixture of Gaussians method develops a parametric background model maintaining a probability density function for each pixel represented by a mixture of Guassian functions. The pixel distribution $f(I_t = u)$ is represented by a mixture of K Gaussian functions:

$$f(I_t = u) = \sum_{i=1}^{K} \omega_{i,t} \cdot \eta(u; \mu_{i,t}, \sigma_{i,t})$$

where $\eta(u; \mu_{i,t}, \sigma_{i,t})$ is the Gaussian component with intensity mean μ and standard deviation σ. $\omega i,t$ represents the i^{th} component. K, the number of Gaussians, ranges from 3 to 5, depending on memory limitations [6, 12, 4].

This method provides a stronger background model, but at the expense of added complexity and memory requirements. Trailing artifacts and object movement remnants as the background model is updated, while reduced, are still present, as seen in [4, 6, 12, 13]. As a result, mixture of Gaussians was not determined to be the best method for the objectives of this thesis.

3.4 The Algorithm

As is usually the case, the simplest method can be the most effective. Using the frame difference method and relying on a static background proved to be the most efficient solution to fast and informative foreground detection. Critical to separating foreground from background is the predetermined threshold value. Because the best threshold value may change depending on the camera used, lighting, and numerous other factors, a GUI was developed to determine the threshold before recording any data.

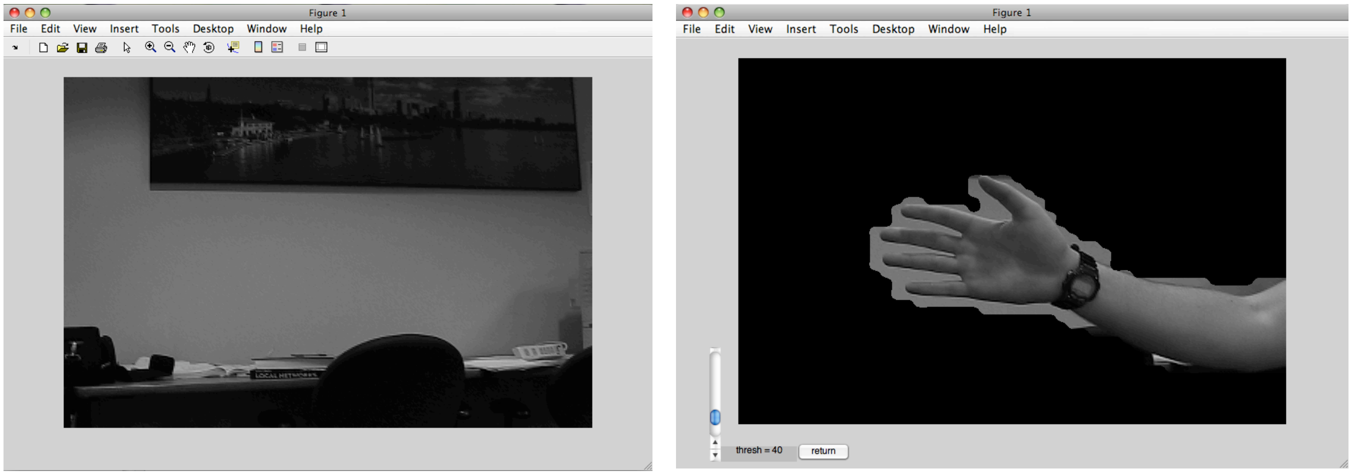


Figure 1
File Edit View Insert Tools Desktop Window Help
Figure 1
File Edit View Insert Tools Desktop Window Help
thresh = 40
return

(a) (b)

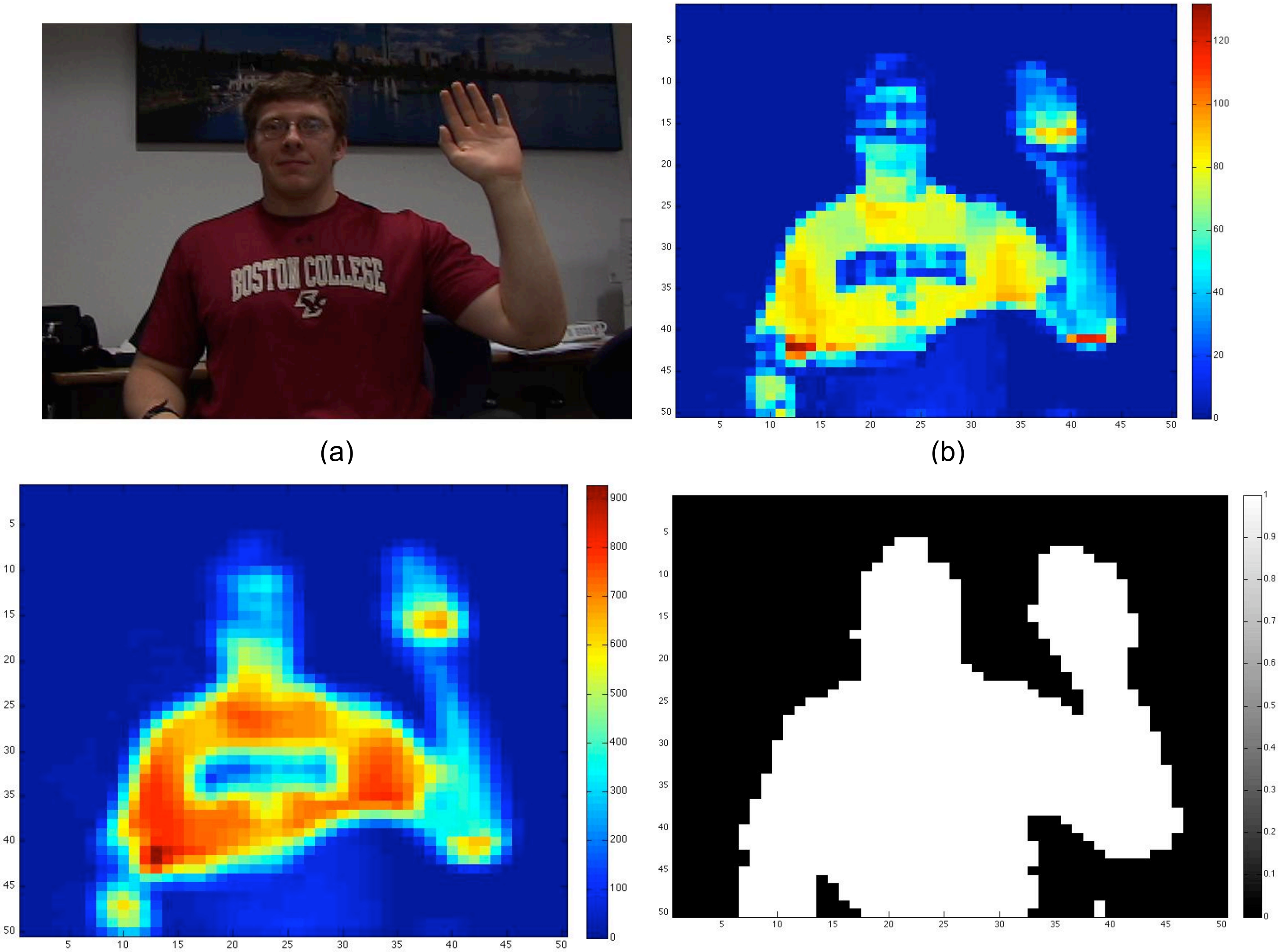
Figure 3 – The GUI developed and used to determine an effective threshold for feature detection. (a) depicts the static background. (b) shows a hand that it is determined to be foreground.

The threshold returned by the GUI can then be passed to the function responsible for video input. Before foreground detection takes place, the static background model is recorded. Foreground detection for a single frame from video is then performed as follows:

```
$frame_i$ = $frame_i$ resized to nxn
$d$ = | $frame_i$ – $background$ |
$d$ = apply filter to $d$
for each $pixel\ x,y$ in $d$
        if $d_{x,y}$ > $threshold$
                $foreground_{i:x,y}$ = 1
        else
                $foreground_{i:x,y}$ = 0
```

Figure 4 – Pseudo code for foreground detection algorithm used.

The frame is first resized before determining the difference between the frame and background. Resizing and then blurring after determining the difference aids in removing artifacts such as clothing textures and design patterns. The resize and blur result in an acceptable loss of detail since we are only concerned with the general shape of the foreground figure representing the human pose. After the resize and blur, each pixel is matched against the threshold, returning a binary mapping of the useful foreground information within the frame.



(a)

(b)

(c)

(d)

Figure 5 – (a) shows an input frame from video, (b) shows the result of background subtraction after resize, (c) shows the application of a blurring filter to the background subtracted image, (d) represents what is finally determined as foreground.

4. Features

Once pertinent foreground information has been retrieved, the next step is determining the best way of storing pose information in the database. A number of features were considered with the aim of minimizing database storage requirements while maintaining and emphasizing significant data.

4.1 Person Detection

Utilizing person and human shape detection algorithms was considered for the extraction of pose information from the foreground of a frame. The object detection system developed in [9] effectively detects a person in an image. The person detection, however, focuses on human body and ignores limb information at times, information that is critical to pose determination. Parameters can be changed to allow for a more generous marking of a person with the hopes of including limbs, but such changes also increase the possibilities for false positives. Speed is also an issue. Person detection, at least using the algorithm in [9], makes near real time pose estimation difficult.

4.2 Shape Context

A shape context feature, as described in [2], takes a distribution of points on a shape return a descriptor for that shape. For a point p_i on the shape, a histogram is computed using log polar coordinates for the coordinates for the remaining points on the shape relative to p_i such that:

$$h_i(k) = \#\{q \neq p_i : (q - p_i) \in bin(k)\}$$

which results in features that similar for corresponding points and dissimilar for non-corresponding points. The points on the shape used in shape context features, however, are sampled from contour points. Performing edge operations on an image takes more time, in addition to time spent forming the feature itself. Shape context features, although highly descriptive, have an unjustifiable computational expense in light of a simpler and faster feature.

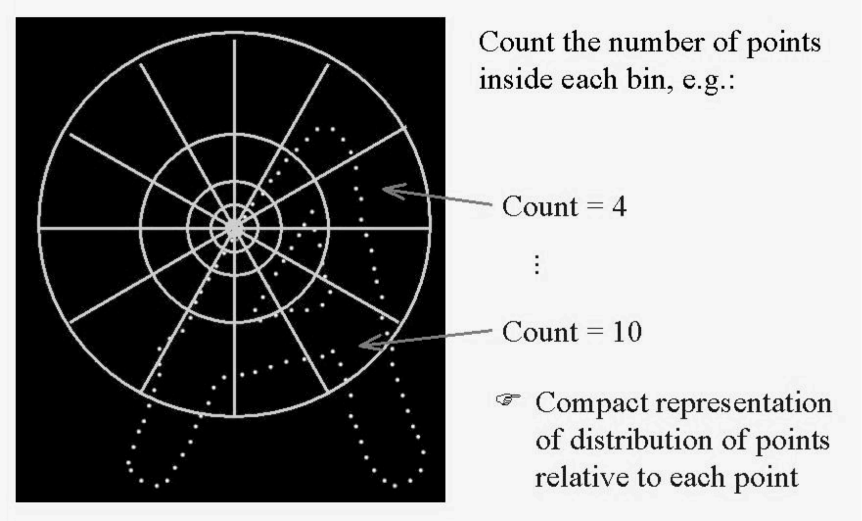
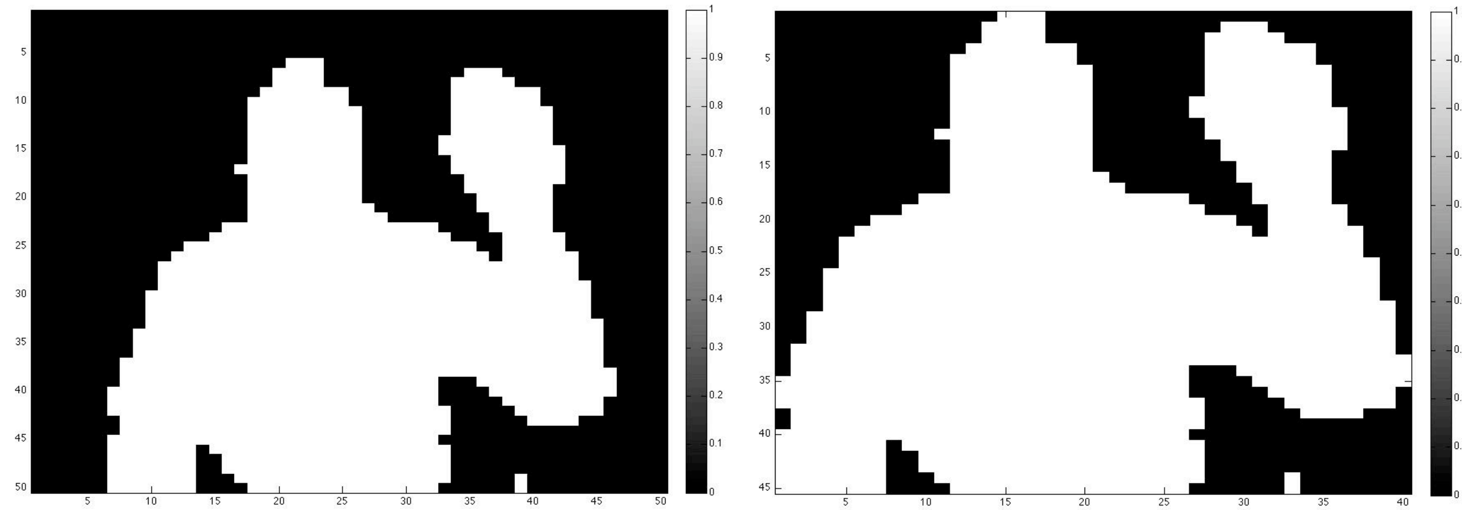

Count the number of points inside each bin, e.g.:
Count = 4
⋮
Count = 10
☞ Compact representation of distribution of points relative to each point

Figure 6 – This image from [2] provides a visual example of the method in which shape context figure are extracted from a shape.

4.3 Blob extraction

The method chosen for the purpose of fast feature development is to rely on effective foreground detection and to use foreground pose information as the feature. The blob returned by the foreground detection is extracted and any non-foreground information is discarded. Features are then uniformly resized, providing data that can be used in locality sensitive hashing.



(a)

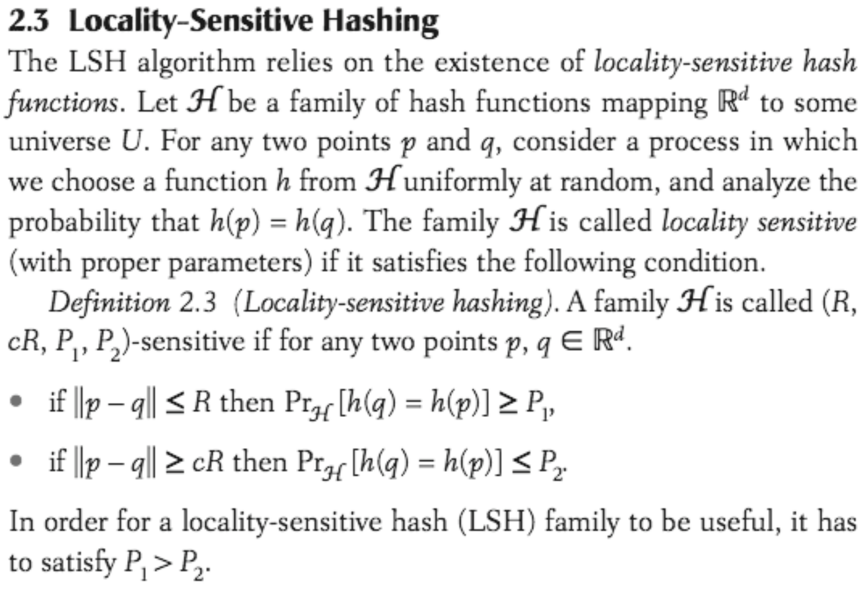
(b)

Figure 7 – (a) represents the foreground of a frame of video, (b) represents a blob-extracted feature in which any non-informational rows or columns of a foreground image have been removed.

5. Locality Sensitive Hashing

5.1 Definition

Locality sensitive hashing is used for the quick nearest neighbor look up of high dimensional data. In the case of this thesis, the high dimensional data are human pose features. The idea behind locality sensitive hashing is to use a number of hash functions to ensure that the chance of collision is higher for elements that are closer together than for those that are farther a part. Nearest neighbor queries then become simply a matter of hashing the query and returning the elements in that bucket [1, 10]. Locality sensitive hashing is best defined in [1] as follows:



2.3 Locality-Sensitive Hashing

The LSH algorithm relies on the existence of *locality-sensitive hash functions*. Let $\mathcal{H}$ be a family of hash functions mapping $\mathbb{R}^d$ to some universe U. For any two points p and q, consider a process in which we choose a function h from $\mathcal{H}$ uniformly at random, and analyze the probability that $h(p) = h(q)$. The family $\mathcal{H}$ is called *locality sensitive* (with proper parameters) if it satisfies the following condition.

Definition 2.3 (Locality-sensitive hashing). A family $\mathcal{H}$ is called (R, cR, P_1, P_2)-sensitive if for any two points $p, q \in \mathbb{R}^d$.

- if $\|p - q\| \leq R$ then $\Pr_{\mathcal{H}}[h(q) = h(p)] \geq P_1$,
- if $\|p - q\| \geq cR$ then $\Pr_{\mathcal{H}}[h(q) = h(p)] \leq P_2$.

In order for a locality-sensitive hash (LSH) family to be useful, it has to satisfy $P_1 > P_2$.

Figure 8 – A definition of locality sensitive hashing from [1].

Locality sensitive hashing allows for the quick lookup and matching of high dimensional data. In terms of this thesis and human pose estimation, locality sensitive hashing allows for the quick lookup of a single pose from a database containing a large number of human poses. The speed of the lookup allows for near real time human pose estimation using video.

5.2 Dataset Development

The success of human pose estimation using locality sensitive hashing is dependent upon the data being hashed and used for matching the query pose. In this case, the data being hashed are the human pose features extracted from the foreground of video frames. Once features have been extracted from each frame of video, each uniformly sized *n x n* feature is resized to a *1 x n*^{2} vector suitable for hashing in high dimensional space. For *m* video frames, *m* features in an *m x n*^{2} matrix are then hashed by columns using locality sensitive hashing. In the process of extracting features from the frames of video for the development of the dataset, the frames of video corresponding to the features is stored as well. Additional pose information, such as pose labeling, can be added to corresponding features once all features have been extracted.

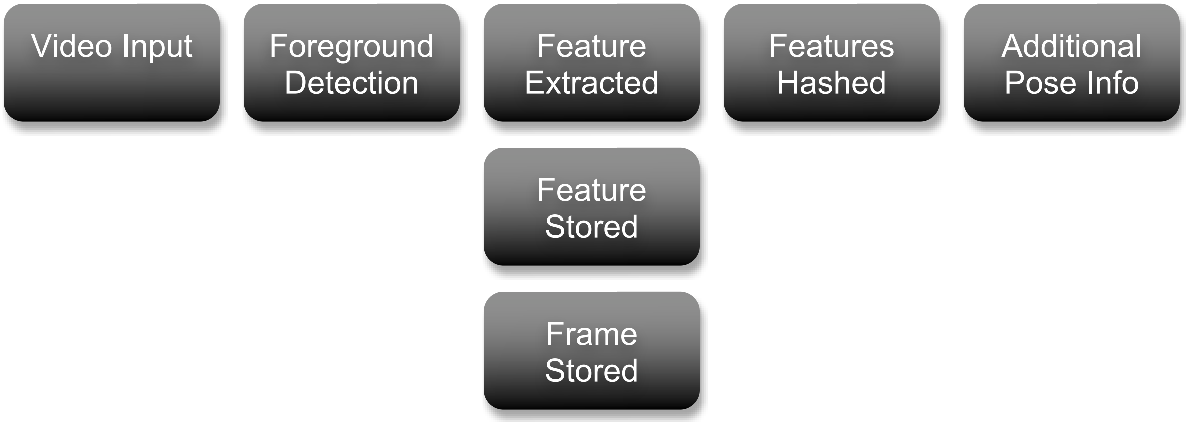


Figure 9 – An overview of the steps involved in dataset development.

For the purposes of this thesis, two datasets have been developed. The first, which will be referred to as the stick figure dataset, is a MATLAB generated dataset of ‘stick figure’ poses, six 3-dimensional rectangular shapes formed in such a way so as to represent a human figure from the waste up. A right upper and right lower arm as well as a left upper and left lower arm is rotated on each side of a head and body about the z-axis to generate arm poses that match potential human movement. 2026 poses were generated for the stick figure dataset. Examples from this dataset are shown in the figure below.

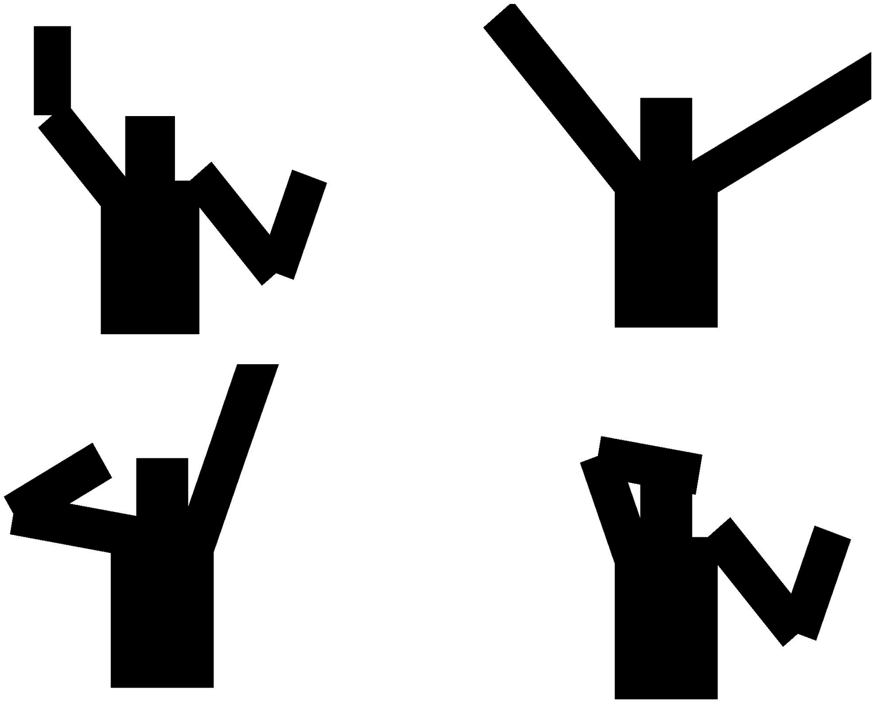


Figure 10 – Four example poses of 2026 from the stick figure dataset developed in MATLAB.

The second dataset consists of poses generate by recorded human movement. A static background image is recorded before a human subject steps in front of the camera. Foreground detection takes place followed by feature extraction to develop a dataset of pose features based on actual human movement. A number of datasets were formed throughout the progression of this thesis, some with as few as 500 features. Examples of the features from a human pose based dataset are shown in the figure below.

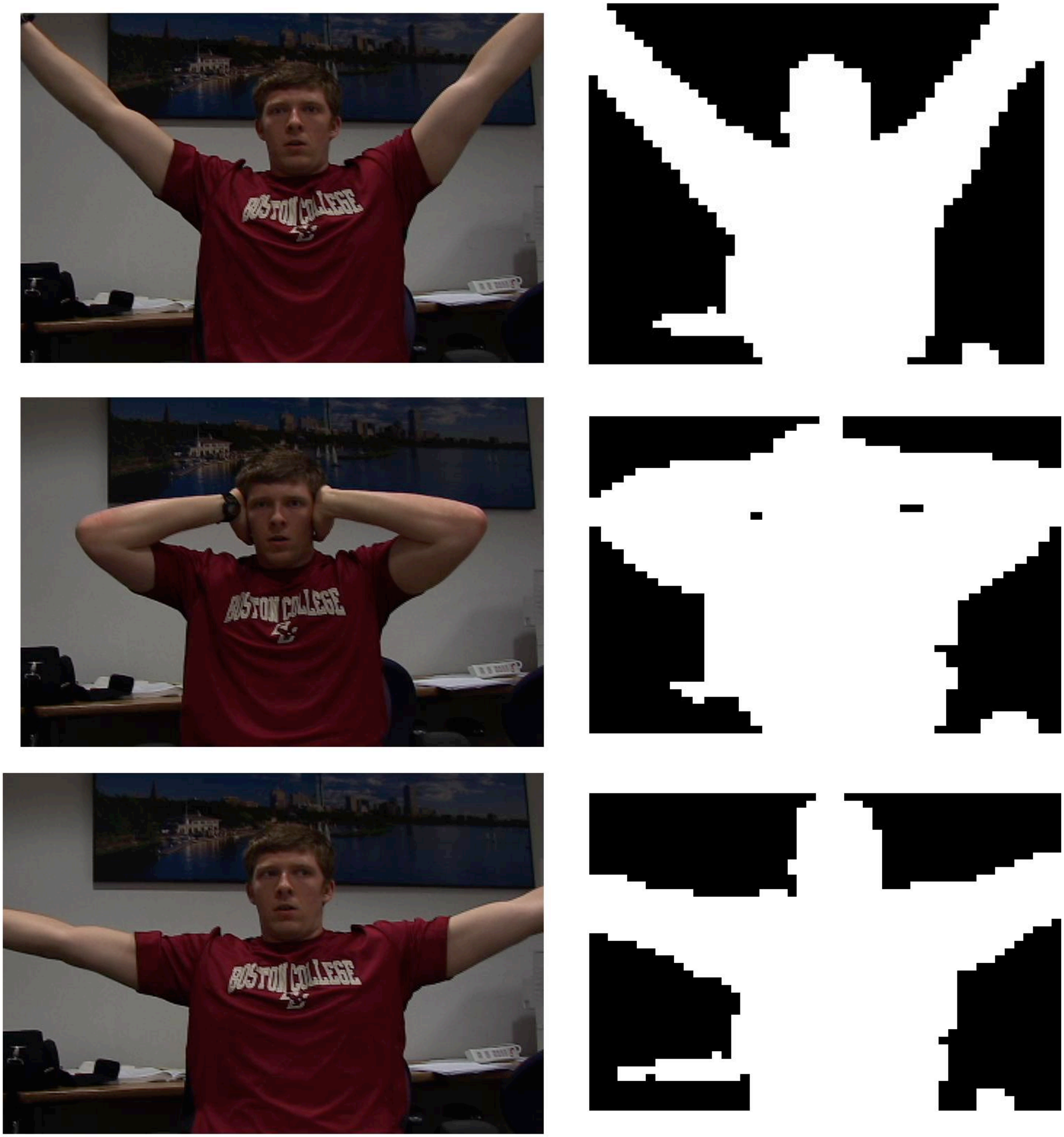


Figure 11 – Examples of features pulled from frames of video for a dataset of human poses.

5.3 Feature Mapping

Feature mapping is the effort of querying the database of poses for a matching input pose. Similar to the initial steps of dataset development, foreground detection and feature extraction are performed on an input frame from video. The feature pulled from the frame of video is then hashed against the set of poses stored using locality sensitive hashing. If there is a collision and a matching feature is found in that bucket, the index of that feature within the *m x n*^{2} matrix of features formed during dataset development is returned by the locality sensitive hashing lookup function. This index can then be used to retrieve the corresponding pose information, such as the matching video for that feature in the dataset or any pose labels, stored during the dataset development stage.

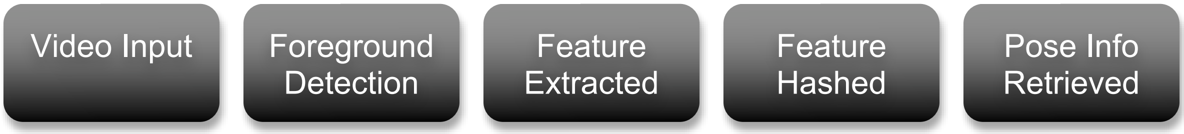


Figure 12 – An overview of the steps involved in feature mapping and locality sensitive hash lookup.

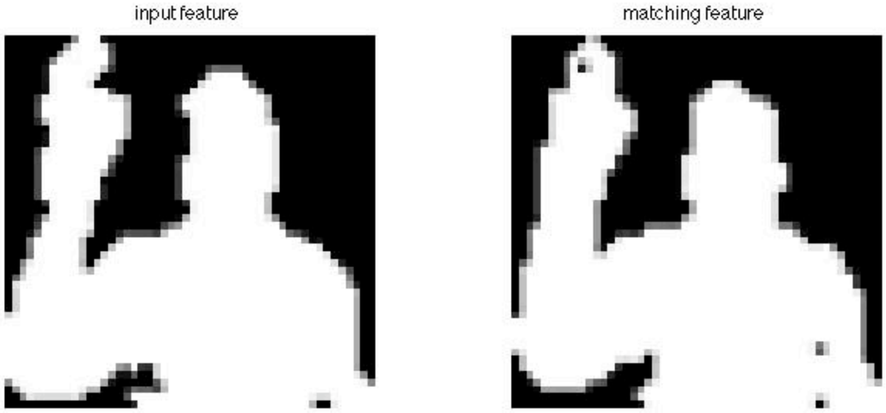
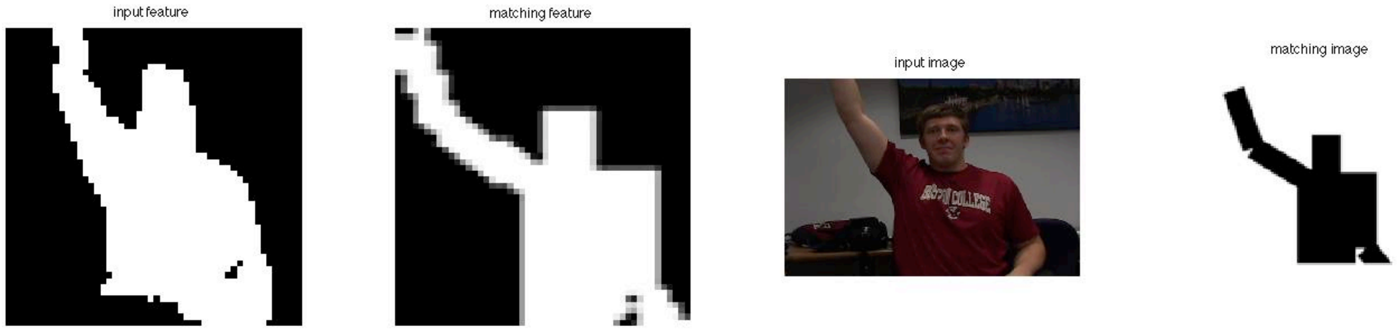
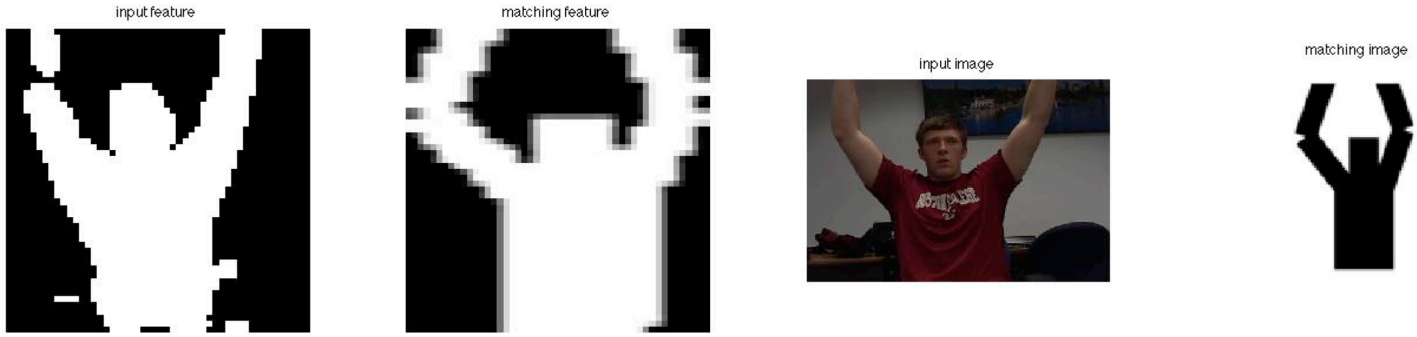


Figure 13 – An example of feature mapping showing an input feature on the left with a matching feature pulled from a locality sensitive hash lookup on the right.

6. Results

The success of human pose estimation in video using locality sensitive hashing is dependent upon the database of poses being used. Being able to extract useful features is important, but without a sufficient database, queries made using locality sensitive hashing cannot return a match. Dataset development, consequently, is critical in determining the validity of locality sensitive hashing as a means of human pose estimation in video. Determining a successful lookup is based on a visual judgment that the input pose matches the pose returned by a database lookup.





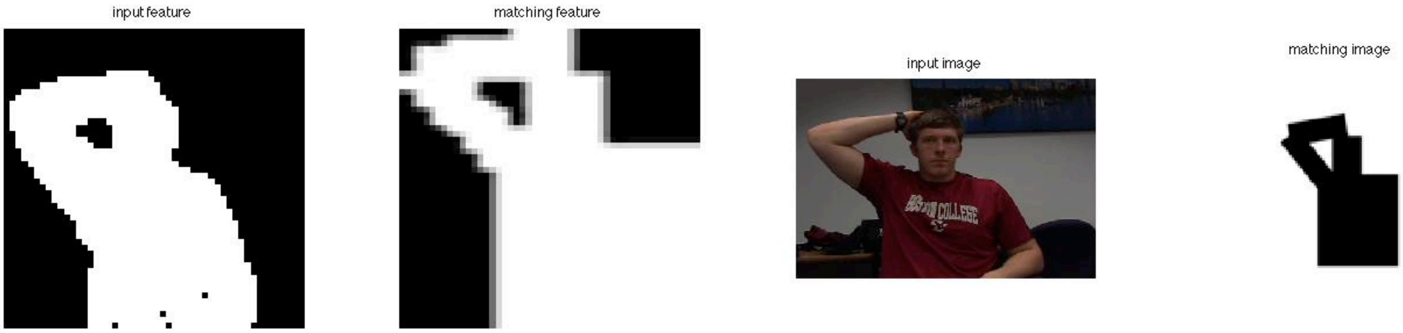


Figure 14 – Example successful matches using the stick figure dataset. From left to right the images represent: an input feature, the matching feature pulled from a locality sensitive hash lookup, the input frame of video, the pose information matching the feature pulled from the database.

Experiments were performed for this thesis using a Sony DCR TRV-950 video camera on a 2.33 GHz Intel Core 2 Duo Mac with 2 GB 667 MHz DDR2 SDRAM in MATLAB for OS X. Foreground detection and feature extraction took 0.01 seconds on average, regardless of the dataset being tested. A locality sensitive hash lookup for the human dataset of 500 poses took 0.0075 seconds on average. A locality sensitive hash lookup for the stick figure dataset of 2026 poses took 0.0073 on average. Total processing for a single frame for the human dataset, including foreground detection, feature extraction, feature lookup, and a display of the matching feature and corresponding pose information, took 0.2333 seconds on average, or at a rate of 4.3 frames per second. Total processing for a single frame for the stick figure dataset took 0.2453 on average. A linear feature lookup took 0.2671 for a dataset of 300 poses, compared to 0.0129 using locality sensitive hashing, indicating that locality sensitive hashing leads to a significant decrease in lookup time.

7. Discussion

Near real time human pose estimation using locality sensitive hashing seems to be possible considering the results of this thesis. The speed of a locality sensitive hash lookup allows for near real time results. The features used also lend themselves well to locality sensitive hashing. A dataset with as few as 500 poses presents a surprisingly large number of accurate matches. A primitive stick figure dataset presented a large number of matches as well. Although the largest dataset tested was only 2026 features due to system memory limitations, further optimizations can be imagined for even faster results with a larger dataset.

Of the two datasets tested, a human dataset of only 500 poses seemed to outperform the stick figure dataset of 2026 poses. Reasons for this apparent illogicality include the fact that a stick figure pose consists of only 4 moving parts, the upper and lower arms, whereas a human pose consists of many more moving parts with some degree of consistency at their points of connection. The stick figure dataset presented sharp angles not present in human poses, which can be imagined to cause some issues in feature matching.

Both datasets did present errors. In the examples from the human dataset below, cases in which the arms are close to or in front of the body produce errors, attributed to the pose feature being used.

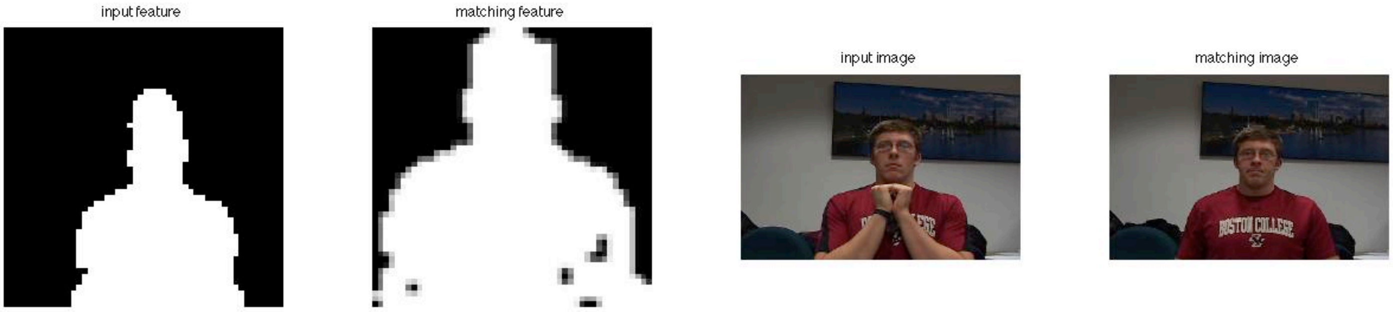


Figure 15 – An example of a false positive feature match. This error is the result of features being incapable of determining poses in which limbs are in front of the body.

The stick figure dataset, made up of only black rectangles, is incapable of displaying poses in which arms are in front the body. And both datasets, as expected, produce errors when queried for poses not in the dataset, as shown below.

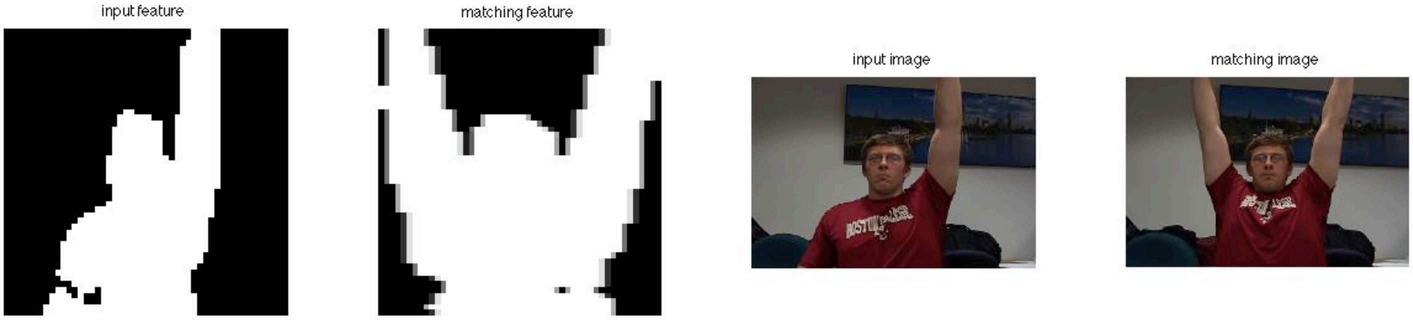


Figure 16 – An example of an error in which the input pose is not in the database of poses.

8. Future Work

Although this thesis demonstrates that near real time human pose estimation is possible using locality sensitive hashing, there is still much potential to be explored. Foreground detection could be improved to remove the restriction of a static background. Motion based poses should logically be possible as well. Features could be improved to resolve poses in which limbs are overlapping or in front of the body. Larger and more comprehensive datasets should present immediate improvement in the quality and consistency in matches. Upper body poses were used in the work performed here only to minimize the number of possible poses, but locality sensitive hashing should be just as effective for full body poses as well as for fast feature matching beyond human pose estimation.

There are a number of potential interesting applications for near real time human pose estimation using locality sensitive hashing as well. Human computer interaction could be changed by allowing humans to interact with computers through no actual physical input but rather by attaching significance to human poses. Computers as well as robots could be taught to respond to human poses. Digital animation could be influenced as well if a dataset of features were attached to corresponding animated figures. While this thesis examines the possibility of human pose estimation using locality sensitive hashing, there is still a great deal of possibilities still to be discovered.

9. References

[1] Andoni, Alexandr and Indyk, Piotr. Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. Communications of the ACM, Vol. 51, No. 1, 117-122. 2008.

[2] Belongie, S. and Malik, J. Matching with shape contexts. IEEE Workshop on Content-based Access of Image and Video Libraries. June 2000.

[3] Bentley, J.L. Multidimensional binary search trees used for associative searching. Comm. ACM 18, 509–517. 1975.

[4] Benton, Seth. Background subtraction, part 1: MATLAB models. DSP Design Line. August 10, 2008.

[5] Chellappa, R. and Sundaresan, A. Multicamera Tracking of Articulated Human Motion Using Shape and Motion Cues. IEEE Transactions on Image Processing, Vol. 18, No. 9. September 2009.

[6] Cheung, S.C. and Kamath, C. Robust techniques for background subtraction in urban traffic video. Video Communications and Image Processing, SPIE Electronic Imaging. 2004.

[7] CMU Graphics Lab Motion Capture Database, mocap.cs.cmu.edu, developed with funding from NSF EIA-0196217

[8] Friedman, N. and Russell, S. Image segmentation in video sequences: A probabilistic approach. Proceedings of the Thirteenth

Annual Conference on Uncertainty in Artificial Intelligence (UAI–97), 175–181. 1997.

[9] Felzenszwalb, P., Girshick, R., McAllester, D., Ramanan, D. Object Detection with Discriminatively Trained Part Based Models. To appear in the IEEE Transactions on Pattern Analysis and Machine Intelligence.

[10] Indyk, P. and Motwani, R. Approximate nearest neighbor: Towards removing the curse of dimensionality. In Proceedings of the Symposium on Theory of Computing. 1998.

[11] McFralane, N. J. B. and Schofield, C.P. Segmentation and tracking of piglets in images. Machine Vision and Applications, Vol. 8, No. 3, 187-193. 2005.

[12] Mcivor, A. M. Background Subtraction Techniques. Proc. of Image and Vision Computing, Auckland, New Zealand. 2000.

[13] Stauffer, C. and Grimson, W. Learning patterns of activity using real-time tracking. IEEE Trans. on Pattern Analysis and Machine Intelligence, 22, 747–57. August 2000.