

## ABSTRACT

Title of thesis: REAL-TIME POSE BASED HUMAN DETECTION  
AND RE-IDENTIFICATION WITH A SINGLE  
CAMERA FOR ROBOT PERSON FOLLOWING

John Bradford Welsh, Master of Science, 2017

Thesis directed by: Professor Gilmer Blankenship  
Department of Electrical Engineering

In this work we address the challenge of following a person with a mobile robot, with a focus on the image processing aspect. We overview different historical approaches for person following and outline the advantages and disadvantages of each. We then show that recent convolutional neural networks trained for human pose detection are suitable for person detection as it relates to the robot following problem. We extend one such pose detection network to spatially embed the identity of individuals in the image, utilizing the pose features already computed. The proposed identity embedding allows the system to robustly track individuals in consecutive frames even in long term occlusion or absence. The final system provides a robust person tracking scheme which is suitable for person following.

REAL-TIME POSE BASED HUMAN DETECTION  
AND RE-IDENTIFICATION WITH A SINGLE CAMERA  
FOR ROBOT PERSON FOLLOWING

by

John Bradford Welsh

Thesis submitted to the Faculty of the Graduate School of the  
University of Maryland, College Park in partial fulfillment  
of the requirements for the degree of  
Master of Science  
2017

Advisory Committee:  
Professor Gilmer Blankenship, Chair/Advisor  
Professor Rama Chellappa  
Professor Romel Gomez

## Preface

This thesis presents a real-time system for detecting and re-identifying multiple people using a single camera. The main contribution is a method for re-identifying multiple individuals in real-time on moderate hardware. Example results are available at <https://youtu.be/IOne0U7tEBI>.

# Table of Contents

|         |  |    |
|---------|--|----|
| 1       | Introduction                                   | 1  |
| 2       | Person Following With Mobile Robots            | 4  |
| 2.1     | Stereo Camera                                  | 6  |
| 2.2     | Thermal Imaging                                | 8  |
| 2.3     | LIDAR  | 9  |
| 2.4     | Structured Light Camera                        | 10 |
| 2.5     | Single Camera                                  | 11 |
| 3       | Efficient Object Tracking                      | 13 |
| 3.1     | Method Overview                                | 13 |
| 3.2     | Correlation Filtering                          | 14 |
| 3.3     | MOSSE Filter                                   | 15 |
| 3.4     | Kernelized Correlation Filter                  | 17 |
| 4       | Methods for Detecting People in Images         | 18 |
| 4.1     | Histogram of Oriented Gradients                | 18 |
| 4.2     | Deformable Part Model (DPM)                    | 20 |
| 4.3     | Fastest Pedestrian Detector in the West (FPDW) | 23 |
| 5       | Pose Detection Neural Network                  | 24 |
| 5.1     | Datasets                                       | 25 |
| 5.2     | Recent Work                                    | 26 |
| 5.3     | Pose Detection using Part Affinity Fields      | 27 |
| 6       | Reidentification Neural Network                | 31 |
| 6.1     | Datasets                                       | 32 |
| 6.2     | Recent Work                                    | 34 |
| 6.3     | Proposed Identification Architecture           | 36 |
| 6.3.1   | Network Structure                              | 37 |
| 6.3.1.1 | Pose Detection Sub-Network                     | 38 |
| 6.3.1.2 | Low-Level Sub-Network                          | 39 |
| 6.3.1.3 | Mixing Sub-Network                             | 41 |

|                                  |    |
|----------------------------------|----|
| 6.3.2 Identity Parsing . . . . . | 43 |
| 6.4 Implementation . . . . .     | 45 |
| 6.5 Training . . . . .           | 46 |
| 6.6 Accuracy . . . . .           | 49 |
| 7 Example Application            | 54 |
| 8 Conclusions and Future Work    | 57 |
| Bibliography                     | 59 |

## Chapter 1: Introduction

This thesis is concerned with the ability of a mobile robot to follow an individual. This is not a novel task and there are many existing techniques that address this problem. However, most of these techniques make strong assumptions about available sensors, computing power, environmental conditions, or other infrastructure. For example, approaches relying on depth images recovered from structured light sensors like the Microsoft Kinect can work relatively well, but are limited to indoor use within a specific range. Approaches that incorporate LIDAR sensors are simply too expensive for many applications. The few approaches that do not have heavy hardware dependencies often constrain the person in some way, or simply do not work well. While few techniques for person following are easily implemented, recent advances in computer vision using deep learning techniques suggest that an effective person following method could be implemented using only a single camera and a computer equipped with a modern, low cost, low power GPU. While the use of a GPU may have been prohibitive in certain applications for reasons such as power consumption, weight, cost, and programming complexity; new low cost embedded platforms such as the Nvidia TX1 make their use practical even on small unmanned aerial vehicles. The primary contribution of this thesis is a person tracking system

design that operates under the aforementioned constraints.

While modeling and control of the mobile robot is a crucial aspect of the person following task, it is not the focus of this thesis. This thesis is focused on the signal processing side of detecting, identifying, and tracking individuals in images. This is a problem that extends beyond the scope of mobile robotics into fields such as surveillance and video annotation, but we focus on person following scenario. In fact, simplicity, functionality, speed, and extensibility as they relate to mobile robotics inform the design as strongly as raw benchmark performance statistics that drive most modern image processing algorithms.

This thesis is intended to serve as a reference to those interested in investigating the problem of person following, and to application developers wishing to integrate the algorithms and methodologies into their work. As such, we include an overview of historical methods and building blocks that may be useful to those designing their own system, and attempt to cover the specifics of our system in as much detail as necessary. The remainder of this thesis proceeds as follows. In chapter 2 we discuss existing systems that directly address the robot person following problem. For each we note the hardware assumptions and performance. In the remaining chapters we constrain ourselves to the scenario where only a single camera is used for the identification and tracking problem. In chapter 3 we overview efficient object tracking algorithms that could be directly applied to the person following problem, or integrated into some larger system to address performance issues. These object tracking methods run at very high frame rates and are suitable for most CPUs. In chapter 4 we review selected historical methods for detecting people in images

and make suggestions for how the algorithms could be used in the robot following problem. In chapter 5 we see that human pose detection is possible using convolutional neural networks trained on large image datasets for the pose recognition task. We overview many competing methods for the task and select one for use in the final system. Given the reliability of the methods, and the intuition that knowledge of the human pose could be useful for action and identity recognition, we proceed through the remainder of the thesis assuming a specific pose detection algorithm is used for detecting people. In chapters 5 and 6 we assume that a modern GPU is available which can efficiently perform convolutional neural network computations. In chapter 6 we extend the pose detection algorithm to spatially encode the identity of people in the image. This method is strongly related to the task of human re-identification, for which many large datasets are available. We review and draw connections between competitive algorithms for the human re-identification task to reinforce the viability of our method. While we do not focus on obtaining competitive results for the human re-identification benchmark (since most of the algorithms are intended for off-line use), we believe that our (online processing) system provides an efficient, simple, extendable, and sufficiently accurate tool for the robot following problem that is possibly useful in other domains.



## Chapter 2: Person Following With Mobile Robots

The challenge of following a person with a mobile robot has been addressed in many papers including [3] [4] [5] [6] [7] [8] [10] [11] [13] [14] [15]. In [1] a survey is provided which summarizes a few methods that address the problem of person following. This survey is focused on works that directly address the person following problem, and not of the human detection problem in general. By no means is the person following task new and there are a variety of methods for addressing the problem, but most make very specific hardware assumptions.

D. Calisi [3], Z. Chen [4], H. Koyasu [9], J. Satake [13], and T. Yoshimi [15] all use stereo vision for the task. Stereo vision has the advantage of simple background subtraction via depth estimation, the ability to determine the distance to the target, and the ability to detect obstacles in the field of view. However, stereo imposes an additional computational burden in computing depth, the range is often limited, the cameras are often more expensive, and most segmentation methods relying on depth segmentation are confused by adjacent objects. In addition stereo vision approaches do not apply to many existing infrastructures such as surveillance camera systems. Nevertheless, stereo-vision based approaches are appealing when compared to methods like that of [5] which depends on an expensive thermal imaging sensor

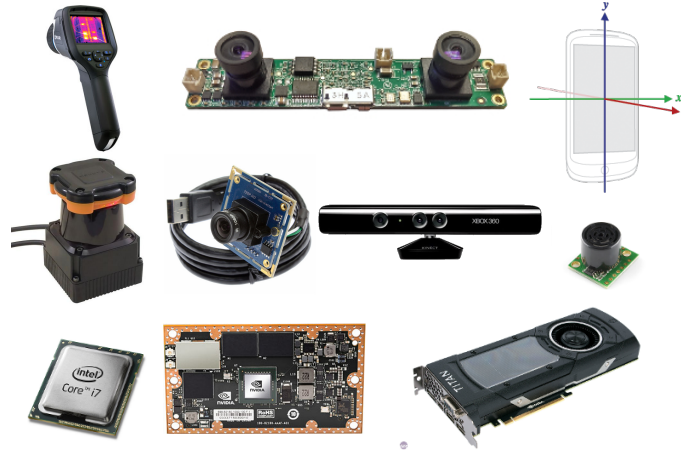


Figure 2.1: Useful hardware used for person following. From top left to bottom right; a thermal imaging sensor, stereo camera, inertial sensing mobile device, 2D LIDAR, standard USB camera, Microsoft Kinect (RGBD sensor), ultrasonic range finder, standard CPU, low power GPU module, high performance GPU

for detecting people. Thermal vision makes the process of segmenting people in the image from the background a simple one, but it provides little information as to the identity of individuals in the image. Methods using structured light sensors like the Microsoft Kinect provide a method for segmenting people from the background similar to that of stereo-vision based approaches, but typically at higher resolution and without the computational burden of computing disparity. These types of sensors can even be used to determine the pose of the person at high speeds [16]. Unfortunately, such sensors do not work well in the presence of heavy infrared light like that outdoors. B. Ilias attempts to address the poor operation of the Kinect outdoors by filtering the image, but he provides little concrete evidence as to the method's success [7]. Like stereo camera methods, structured light sensors are limited in sensing range and not applicable to many existing domains. Other approaches like

that of S. Shaker [14] rely solely on a 2D LIDAR sensor to detect people in the scene. Such approaches typically rely on heuristics that may be prone to detection errors. M. Kobilarov [8] presents a method for person following that uses both an omnidirectional camera and LIDAR. Methods like this are highly dependent on the robot configuration and difficult to extend. The LIDAR is used in their approach to address the loss of tracking that occurs when using only an omnidirectional camera with a feature based tracker.

In the following sections we will cover in more detail some of the methods summarized above. We show that there is a variety of ways to approach the person following problem, but most existing approaches work well only under specific constraints that are suitable usually only for demonstration and not practical long-term use. In addition, the specificity of the approaches makes them difficult to extend in future work or other approaches. Our work is intended to provide both a foundation of knowledge and a low-dependency baseline system that could be extended and applied on a variety of platforms.

## 2.1 Stereo Camera

In [13] J. Satake and J. Miura present a stereo base person detection and tracking scheme focused on the robot person following problem. Their tracking method begins by computing a depth image from the stereo pairs. Regions proposed for the person's head are then detected by template matching regions of the depth image against three binary templates using a sum of squared error loss metric. The tem-

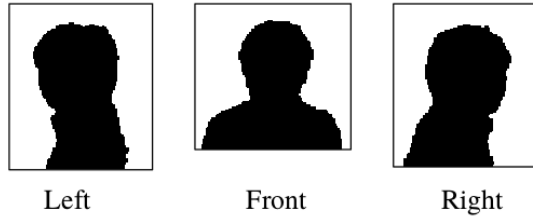


Figure 2.2: Depth templates from [13]

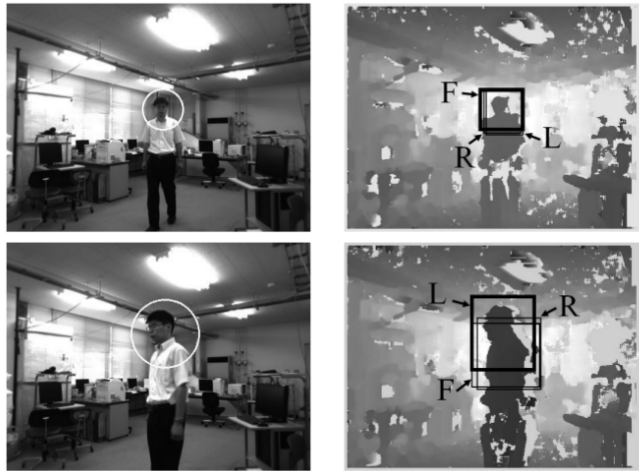


Figure 2.3: Depth template matching results from [13]

plate corresponding to the lowest error is selected as the proposed detection region. The depth matching procedure is crude and results in many false positives so the authors train an SVM on the intensity images of the proposal regions to classify and remove false positives. This results in more robust person detection. The distance to the proposed person is then estimated using the depth. People are tracked in consecutive frames using an Extended Kalman Filter (EKF) for each person, along with a simple data association process to determine which measurements belong to which track. Each EKF has as a state the 3D position and velocity of the person. Data association is done by matching tracked states with their closest 3D position

detection from the template matching phase. The algorithm prefers missed detections over false positives because missed detections are handled by simply skipping the update step in the EKF.

This tracking algorithm has the advantage that it is relatively simple, is not dependent on feature tracking, and is able to track multiple people. However, a major limitation is that it cannot maintain individuals' identities during long term occlusion, people in very close proximity, or people that leave the scene. These drawbacks make the method unusable in many practical robot following scenarios.

## 2.2 Thermal Imaging

In the paper [5] the authors present a method for detecting people in thermal images for person following. The method involves thresholding the magnitude of a thermal image, computing various shape statistics for regions in the thresholded image, and then training an SVM classifier on these shape statistics to determine if the thresholded region is a person or not. An issue with the approach is that it provides no method for distinguishing multiple people. A specific individual could not be tracked if there are multiple people in the scene.

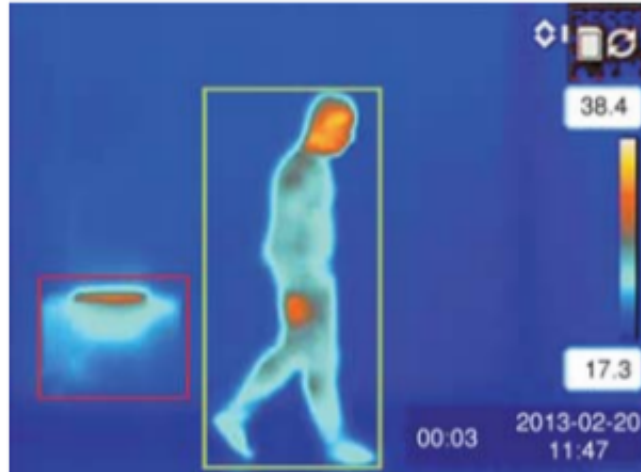


Figure 2.4: Thermal based person detection from [5]

## 2.3 LIDAR

The paper [14] reports a method for following a person with a mobile robot using LIDAR. The algorithm for detecting the person is a logical ruleset that includes thresholding the length of contiguous laser reading segments. It is difficult to assess the performance of this method with no concrete statistics, but it is possible to imagine many scenarios where such a ruleset could fail. Moreover, the method does not seem able to distinguish individuals. In the paper [8] they utilize both an omni-directional camera and a LIDAR for tracking. The camera is used primarily to compute color features that can be used to probabilistically match the target. The lidar in this scenario is used to determine candidate regions for detecting the target. Methods utilizing LIDAR are useful because they can accurately determine the distance to the target, which is useful for controlling the robot, but there is little evidence that such tracking methods work for an extended period of time.

Such methods are also expensive and highly dependent on the sensor configuration.

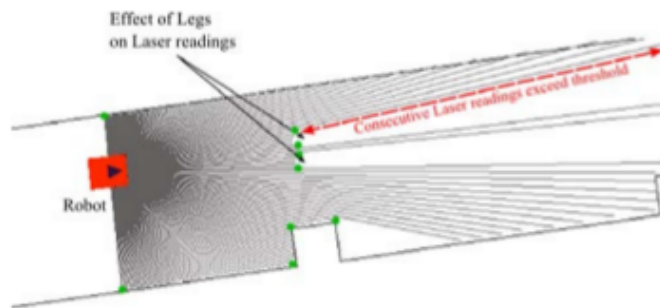


Figure 2.5: LIDAR visualization from [14]. Heuristic decisions based on the segmented LIDAR scan are used for detection and tracking.

## 2.4 Structured Light Camera

Structured light sensors like the Microsoft Kinect are capable of estimating the depth of pixels in an image. Utilizing algorithms like that of [16] it is possible to determine the pose of multiple individuals in real-time using only the depth images. This provides rich information about people in the image that may be used for robot person following. In the work [7] they use this method for person detection to enable a robot to follow a nurse indoors. Their approach uses the standard Kinect skeleton tracking software to perform tracking. A simple control strategy is used that involves turning left or right if the tracked person crosses a threshold region and going straight otherwise. This is depicted in Figure 2.6.

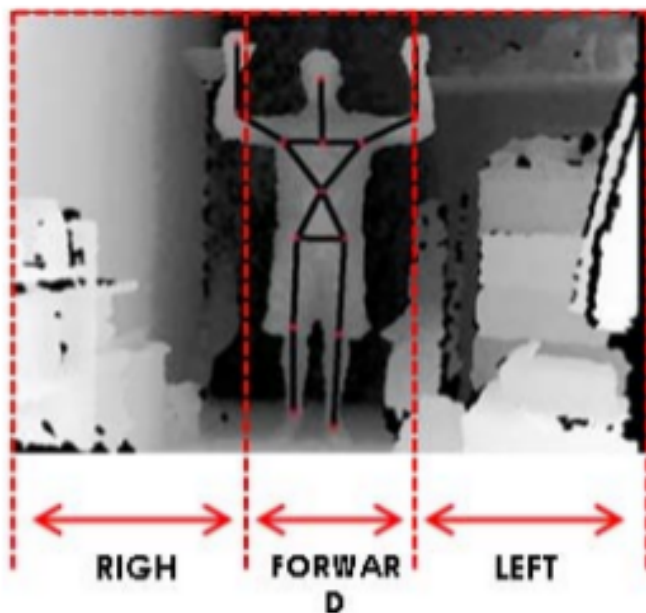


Figure 2.6: Visualization of detection and control strategy used in [7].

A major issue with this approach is that it does not work well outdoors due to ambient infrared light. The authors place an IR filter on the Kinect to help address this issue, but provide little information about the performance. In addition, there is little indication that a method like this is capable of distinguishing between multiple people, especially in the case of long-term absence or occlusion. Structured light sensors are also typically limited to a few meters in range. This limits their applicability for longer range person following.

## 2.5 Single Camera

In this thesis we are concerned with developing a real-time person tracking system that uses only a single camera. There are relatively few works directly addressing the person following problem that use only a single camera. The work [?]



begins by using a single omnidirectional camera for tracking the target, but they ultimately add a LIDAR sensor to improve reliability. In the paper [?] they train a convolutional neural network to detect people in real-time for person aware robot navigation. This work however does not address the person following problem, but how to navigate when there are people in the scene. The method does not distinguish individuals to detect a tracking target. There is however a large body of work related to detecting and identifying individuals in images, but it is not directly aimed at the person following problem. In the remaining chapters we will go into detail on methods that could be used for person following using only a single camera.

## Chapter 3: Efficient Object Tracking

In this chapter we discuss methods for generic object tracking. These methods are initialized by selecting a bounding box in an initial image frame and then tracking the 'primary' object in the bounding box through consecutive frames. Most of these methods discriminate the object appearance given a single image, and update the model as the object is tracked from frame to frame. A major drawback is that the learned model drifts from the true object over time. Thus, the following methods either need to be manually re-intialized after they lose track, or combined with a slower, more accurate detection method. These methods may also be useful where computational resources are limited.

### 3.1 Method Overview

As mentioned, this section deals with methods for tracking an object across multiple frames. They do not detect a specific object class, but learn the object appearance from consecutive frames. I avoided extensive investigation of methods using point features because the deformability of the human body and clothing was not well suited to the rigid requirements that such methods work best under. Instead, I investigated filter based approaches which are arguably better at discrim-

ination using the entire object appearance rather than localized descriptors.

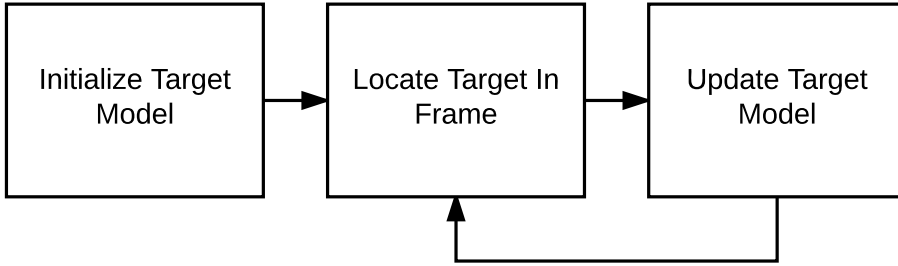


Figure 3.1: Generic object tracking process

### 3.2 Correlation Filtering

Correlating a template image with a query image is a simple, yet sometimes effective way of detecting the template image within the target image. Correlation filtering begins by selecting a target window  $h_k \in R^{m \times n}$ . This is done by selecting the bounding box of the target object to be tracked. Once the target is defined, tracking is performed by iteratively selecting the peak coordinates of the correlation of the target window with the query image  $f_k \in R^{M \times N}$ . Noting that correlation in the spatial domain is element-wise multiplication in the frequency domain, the correlation filtering process is thus

$$(x_k, y_k) = \arg \max \mathcal{F}^{-1}(H_k \odot F_k)$$

where  $H_k$  and  $F_k$  are the Fourier transforms of  $h_k$  and  $f_k$ , and  $\odot$  represents element-wise multiplication. The computational complexity of the method is  $P \log(P)$  with  $P$  the number of pixels in the query image. For tracking problems, the target window  $h_k$  may be updated at the new location  $(x_k, y_k)$ . If detection is desired, the target

window may remain constant.

While correlation filtering is very fast and simple there are many drawbacks. First, while correlation typically produces strong peaks at the target location, it can also produce strong peaks at locations that are not the target center. Second, as tracking continues it is likely that the target window learned in the first frame will not accurately represent the target in later frames. The target, especially in the context of human tracking from a mobile robot, is likely to change scale, pose, illumination, etc. Methods covered in the next two sections, called Minimum Output Sum of Squared Error (MOSSE) filtering and Kernelized Correlation Filtering (KCF) address these two major problems while maintaining the speed and relative simplicity of correlation filtering.

### 3.3 MOSSE Filter

In 2010 D. Bolme introduced an adaptive correlation filter capable of tracking a target window at 669 frames per second. The tracking filter, namely Minimum Output Sum of Squared Error (MOSSE) learns an optimal correlation filter for each frame using the detected location found via correlation with the previous frame's filter. The filter learning process helps ensure that correlation results in peaks only at the target center and not at other points in the image that may have otherwise caused peaks using simple correlation filtering.

Like correlation filtering, the MOSSE filtering process begins by selecting a window around the target. With the initialized target window position set, pre-

processing is performed on the target window. This involves taking the log of the target window to improve low-contrast performance. The target window is then normalized to have a mean value of 0.0 and a norm of 1.0. Next, the window is multiplied by a cosine window, which puts emphasis on the center of the target. Once the target window is preprocessed, an optimal filter is learned which minimizes the mean sum of squared error between the *actual* output of correlation and the *desired* output of correlation. The closed form solution to this problem is simple and can be found in [?]. An example of the output of the learned MOSSE correlation filter is shown in Figure 3.2



Figure 3.2: The MOSSE Tracking process. The left thumbnail shows the input template image, the middle thumbnail shows the learned MOSSE filter, and the right thumbnail shows the output of correlating the learned filter with the new image. The solid red box shows the new target location. [42]

### 3.4 Kernelized Correlation Filter

In 2014 J. Henriques presented a method for high speed tracking using Kernelized Correlation Filters (KCF). In their work they show that real-time tracking can be accomplished by applying the kernel 'trick' over cyclic shifts of an input patch to learn a Gaussian peak regression target. They show that this can be accomplished in the same time complexity of the MOSSE filter ( $n \log(n)$ ). Like the MOSSE filter, they then detect the target peak and update the model in each consecutive frame. An advantage that the kernelized correlation filter has over the MOSSE filter is that it can operate on images with an arbitrary number of channels. Using histogram of oriented gradients (HOG) feature images they achieved state-of-the-art tracking results at 292 frames per second. The method could even be extended with other channels, including color channels, added to the feature image. It is worth noting that all of the person detection algorithms detailed in the next section also utilize HOG features, so augmenting one of these person detection methods with a KCF for tracking could possibly be done at little extra computational cost.

## Chapter 4: Methods for Detecting People in Images

In the previous chapter I discussed methods for generic object tracking that could be directly applied to the robot following problem. The issue with these algorithms is that they often fail if tracking is lost at any point. A different approach to following a person would be to detect all people in consecutive images and distinguish which person is the target. This is the methodology we use in our final tracking system. In this chapter we overview a few historical methods that could be used for the detection phase of the system. In our final system we did not use any of these methods, but they are commonly used algorithms for human detection. In fact, much of the work related to person reidentification (which we detail in chapter 6) assumes the use of the person detection algorithms described in this chapter [32] [33].

### 4.1 Histogram of Oriented Gradients

The method of Histograms of Oriented Gradients (HOG) for human detection was presented by N. Dalal and B. Triggs in 2005 [37], and is one of the most popular methods for human detection in images. Their work essentially describes a well-tuned feature descriptor that they prove to work well for the task of human detection

when combined with a linear support vector machine (SVM). The concepts presented in their work are fundamental to more complicated methods for human detection like that of P. Felzenszwalb’s Deformable Parts Model (DPM) [38]. In any work addressing the problem of human detection, it is worth drawing comparisons to HOG. The pipeline for human detection using HOG is depicted in Figure 4.1.

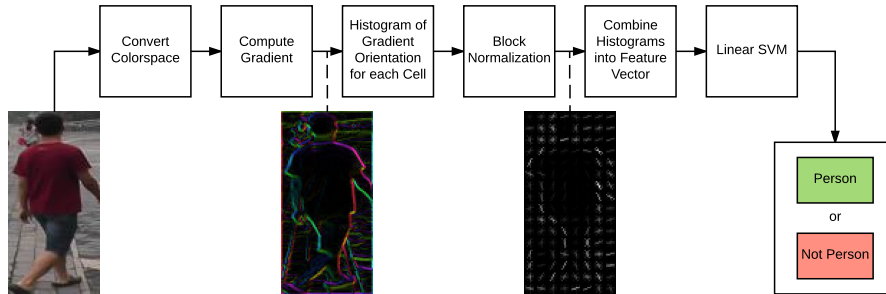


Figure 4.1: The HOG detection pipeline.

The HOG pipeline begins with colorspace selection. The choice between RGB or LAB colorspace has a negligible impact on the detector performance, but the choice between color and grayscale does have a significant impact [37]. After the colorspace has been selected, the gradient of the image is computed. This is done by convolving the image with two filters, one for each direction of the gradient. The authors attempt multiple filters, but the filters  $[-1, 0, 1]$  and  $[-1, 0, 1]^T$  were found to work best for the task at hand. Once the gradient image is computed the binning process begins. The entire image is divided into *cells*, which are  $8 \times 8$  pixel regions<sup>1</sup>. Within each cell, a histogram of gradient orientations is computed by having each gradient pixel within the cell vote for orientation bins with a weight dependent on

<sup>1</sup>Cells are not required to be  $8 \times 8$  squares, I am referencing the cell size of the primary detector the authors use.



the gradient magnitude. The cell histograms are then normalized via comparison to nearby cell histograms. This normalization reduces the impact of local illumination variation. After normalization, a final feature vector is constructed by collecting all of the histogram bin values in a  $64 \times 128$  pixel window ( $8 \times 16$  cells). This feature vector is the input to a linear SVM which decides whether the  $64 \times 128$  pixel windows is a person or not [37].

HOG works well for human detection because it effectively captures the human shape with local descriptors. The nature of the descriptor makes it robust to illumination variation, slight translation, and slight angular rotation. These qualities make the HOG feature very useful in practice.

In their paper, Dalal and Triggs are primarily focused on the HOG feature vector itself rather than the classification process, and this is arguably the more important contribution of the work. HOG features appear in more sophisticated person detection methods like that of Felzenszwalb and are crucial to the performance of the Kernelized Correlation Filter (KCF) object tracking method discussed in the previous chapter. Knowledge of HOG, even if unused in the final system, was a crucial consideration in designing the real-time tracking system.

## 4.2 Deformable Part Model (DPM)

In 2008, P. Felzenszwalb presented a discriminatively trained deformable part model that achieved state of the art results in the 2006 and 2007 PASCAL person detection challenges. The model surpasses the performance of the plain HOG

detector by incorporating separate part detectors that may be shifted relative to a root detector. The part based model is thus better able to handle the non-rigid human structure. Unfortunately, deformable part models are more computationally complex than HOG and are not well suited for real-time detection. Nevertheless, they are one of the best performing person detectors and are worth reviewing. The parallel nature of many of the computations could be well suited for a parallel processor like a GPU, and so they remain an important option for designing a real-time system even though most current implementations do not run in real time. An overview of the system pipeline is depicted in Figure ??.

Felzenszwalb’s deformable part model is highly dependent on the HOG feature of Dalal and Triggs. The deformable part model detection procedure begins by computing HOG features over an image at multiple scales. A root filter, which is equivalent to that of Dalal and Triggs, detects the object center. Multiple part filters detect parts relative to the root filter origin. Each part is scored similar to the root filter, but operates on higher spatial resolution HOG cells. Associated with each part filter is also a deformation cost model. This model assigns a spatial placement cost for each part relative to the root filter. Intuitively this model captures the essence of human detection better than that of Dalal and Triggs; the human body shape can change drastically depending on the persons pose, but the individual part’s shape does not change as much. In addition, we expect certain parts will only exist in certain placements (ie: it is unlikely that a person’s head would exist below the root origin). In summary, classification performance is improved over rigid models by using both part appearance and placement

As mentioned earlier, the DPM is not immediately suited for real-time operation. Felzenszwalb reports a detection rate of once every 2 seconds. To enable real-time feedback control in the context of robot person following, updates are required several times a second. Using the DPM in this context would require algorithm tuning to meet the requirements of real-time person following. Another possible solution would be to combine the slow DPM with a higher rate tracking algorithm such as a Kernelized Correlation Filter (KCF). The DPM could be used to detect the human bounding box once every two seconds, and the KCF could track the person in between DPM updates at the frame-rate of the camera. This combination could alleviate the major issues that the DPM and KCF have when used alone. The issue with the DPM is clearly speed, and the major issue with the KCF is tracking drift, which will be covered in the KCF discussion. Such an implementation may be suitable for a standard CPU.

Though I did not use the DPM in my final solution, the performance improvements of DPMs over rigid HOG methods certainly inspired the decision to investigate pose based solutions for the purpose of detection and identification. Pose is one of the largest variables in human appearance, and incorporating pose information into the identification process could make the system more robust to variation of this type.

### 4.3 Fastest Pedestrian Detector in the West (FPDW)

DPM and HOG are arguably the most popular methods for human detection, but several other methods and variants have been produced to address the issues of performance and speed. A notable algorithm which addresses the issue of speed is P. Dollár's *Fastest Pedestrian Detector in The West* (FPDW). The speed increases are achieved by observing that the primary computational bottleneck in typical pedestrian detectors is the construction of the image feature pyramid. Typically, such as in a standard HOG detector, construction of the image pyramid requires feature computation at several (more than 10) scales. P. Dollár shows that many global features computed at a given scale can be approximated using the features computed at a different scale. This approximation results in significant computational savings by removing the need to compute features at each scale. While the approximation breaks down for large scale differences, P. Dollár shows that computing features only at each octave results in good detection rates. Detections scales within the octave are achieved by creating a pyramid classifier similar to Viola and Jones. This results in detection accuracy similar to traditional image pyramid methods, with speeds approaching the lower-accuracy method of Viola and Jones. For real-time pedestrian detection on typical hardware using HOG based approaches, optimizations like that in FPDW are necessary.

## Chapter 5: Pose Detection Neural Network

In the previous chapter we outlined a few historically popular methods for detecting people in images. These methods are all dependent on hand-crafted HOG features, which have been shown to work well for person detection. Recently, large datasets have emerged that directly address the problem of detecting the pose of people in images. The pose, in this context, is the location of major joints or keypoints on the human body. Competing methods that use these datasets show that human pose can be recovered reliably with high accuracy. One such method which maintains state-of-the-art performance in the MSCOCO keypoint detection benchmark is able to determine the pose of all individuals in an image in real-time.

Human pose detection provides information that is useful beyond just detecting a person's location. The pose information could be used for gesture recognition, extracting the face region for face detection, or determining a person's facing direction, which may be useful for robot control. In chapter 6 we utilize the pose information to aid in re-identifying individuals in different image frames. In the remainder of this chapter we will overview different datasets related to human pose detection, discuss a few competing methods that address the problem, and cover in detail the state-of-the-art algorithm we use in our final system.

## 5.1 Datasets

We looked at two popular datasets for human pose detection, the *MPII Human Pose Dataset (MPII)* and the *Microsoft Common Objects in Context (MSCOCO)* dataset. Both datasets contain a large number of annotated individuals (28,821 for MPII, over 100,000 for MSCOCO), but vary slightly in the image sources and annotation style.

The MPII dataset contains pose annotations of 28,821 people. The images for the dataset were extracted from YouTube videos of people performing different activities. These activities provide large variation in pose and environmental context. The pose information provided includes the positions of body joints, full 3D torso and head orientation, occlusion labels for joints and body parts, and activity labels [18].

The MSCOCO dataset includes pose annotations for more than 100,000 people, but also includes many other annotations. In the dataset they include the segmentation masks of 91 object types, including people in the images. The images cover a wide variety of scenarios, and are intended to be natural images showing objects in their typical context. The pose annotations in the dataset do not include the 3D annotations that the MPII dataset includes, but do include the location of visible eyes, ears, and nose associated with each individual. The state-of-the-art pose detector we used was trained on this dataset, and we prefer it for its widely varied context and rich annotation beyond just human pose. □

## 5.2 Recent Work

In 2014 A. Toshev introduced a system, *DeepPose*, to detect human pose that transitioned the methodology from classical detection approaches to deep neural networks [21] [20]. In their work they trained a multi-stage neural network to directly learn the  $(x, y)$  coordinates of part locations in the image. They trained their method on the smaller Frames Labeled In Cinema (FLIC) and Leeds Sports Dataset. Their method achieved state-of-the-art performance at the time. Since the introduction of DeepPose several methods using convolutional neural networks have emerged. A popular methodology for using convolutional neural networks to detect human pose is to learn a network that generates 'heatmap' images corresponding to the body part locations. These heatmap images are then parsed to determine the  $(x, y)$  body part locations. A typical heatmap includes a single feature map channel per body part detected.

One algorithm that used heatmap regression achieved state-of-the-art results in 2016 on the MPII benchmark. This algorithm by A. Newell [20] used a stacked hourglass network structure to perform heatmap regression. The hourglass design is intended to capture information at multiple scales. Below is an example of the learned heatmap regression.



Figure 5.1: Heatmap regression performed by [20]. A heatmap image is learned for each body part.

The more recent work by X. Chu [?] which achieved state-of-the-art results in the 2017 MPII human pose benchmark also uses stacked hourglass networks for similar regression. A major issue with the above approaches arises when attempting to determine the pose of multiple individuals in the same image. With multiple people in the image it is not clear which body parts in the heatmap are associated with each other. The work by Z. Cao [17] solves this issue in a way that is intuitive, simple, and directly integrates into the heatmap regression network. In their approach they learn not only heatmaps corresponding to the body part locations, but also vector fields that point between associated body parts. These vector fields, called *part affinity fields* are easily parsed to determine the likelihood that two body parts are associated with each other. This is the multi-person pose detection method that we use in this thesis.

### 5.3 Pose Detection using Part Affinity Fields

In 2016 Z. Cao introduced a method for human pose detection capable of running in real-time [17]. The method trains a neural network to learn body part confidence maps and part affinity fields that can be parsed to determine the body



part locations and associations for multiple people in an image.

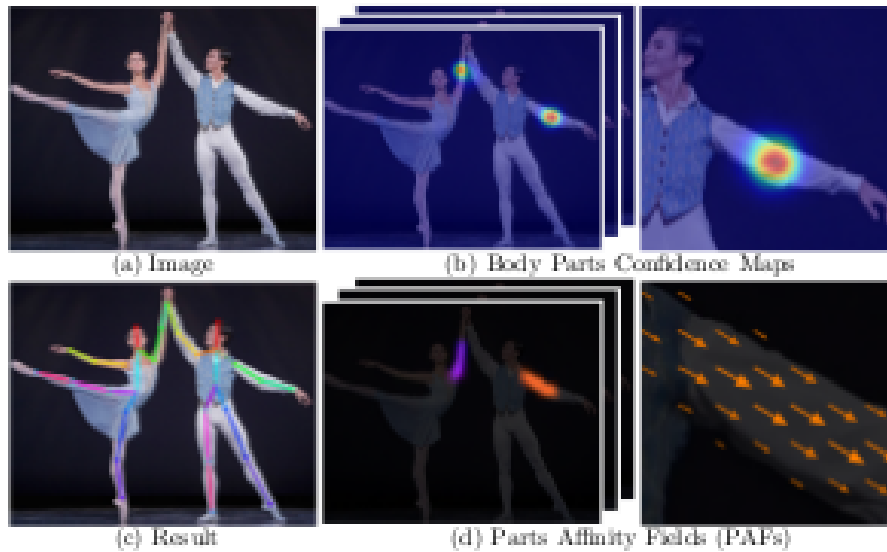


Figure 5.2: Pose detection via Part Affinity Fields from [17]. The input to the neural network is a color image, the output includes body part confidence maps (b) and part affinity fields (d). The confidence maps and part affinity fields are parsed to determine the pose of multiple people in the image.

The neural network structure is quite simple and employs a multi-stage architecture similar to that of [20].

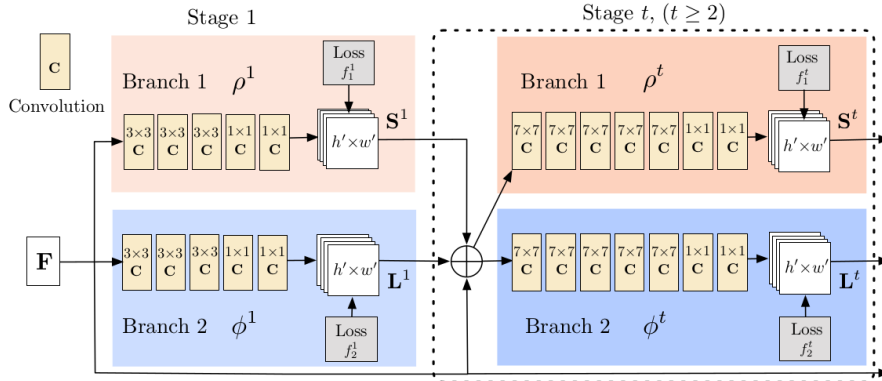


Figure 5.3: The network architecture from [17].  $F$  is a feature map computed from the first 10 layers of VGG-19.  $S, L$  are the part confidence maps and part affinity fields at predicted at each stage. The loss function is applied to all stage predictions.

The first stage in their network calculates a feature map  $F$  using the first 10 layers from the VGG-19 network [?]. This feature map is used as the input to a multi-stage confidence map and part affinity field regression network. In each stage  $t$  the feature map  $F$ , the predicted confidence map  $S^{t-1}$  and the predicted part affinity field map  $L^{t-1}$  are concatenated from the previous stage and used as inputs to the current stage. Each stage is attempting to learn the confidence maps and part affinity fields. Each successive stage improves the accuracy. The loss function is the sum of the  $L2$  distance between the ground truth confidence maps and part affinity fields and those predicted at each stage. To trade runtime for performance it is possible to run fewer stages. We use the second stage in this work. Each feature map has a spatial resolution  $1/8$  the size of the original image. There are a total of 18 parts detected and 19 part connections. Each part connection is represented by two channels in the part affinity field; one for the  $x$  direction and one for the  $y$

direction. This results in 18 channels for the part confidence map and 38 channels for the part affinity field.

The method for parsing people in the image is simple. First, we detect the peak locations of body parts in the image. Then, for each part association we construct a graph containing the pairwise likelihood that two body parts are associated with each other (and thus the same person). This likelihood is determined by taking the dot product of the vector pointing from one body part to the other and the vector predicted in the part affinity field (sampled at  $n$  positions between the two parts). In our implementation of the parsing, we begin with the neck body part and select the most likely shoulder, then elbow, then wrist, etc. for each person. We do not detect the individual if the neck is not visible. We can then determine the bounding box by selecting the extrema coordinates over all body parts corresponding to an individual.

While parsing is necessary to detect multiple people in the image, we show in the next chapter that we can spatially embed the identity of individuals *before* parsing. By directly using the predicted part confidence maps and part affinity fields as features to a mixing network, we can learn to associate low level features with body parts. This spatial feature map provides a simple, fast, and reliable method for detecting the identity of all individuals in an image.

## Chapter 6: Reidentification Neural Network

The focus of this chapter is on re-identifying individuals in consecutive frames. The methods for person following described in chapter 2 address this problem in different ways. Methods like that of J. Satake address the problem by iteratively associating measurements that best agree with a predicted tracking state. These methods rely on the dynamics of the individual to perform tracking. The issue with this type of approach is that it cannot recover from failure. If tracking is lost at any point, it is unlikely to be recovered correctly without any additional person association metric. Other approaches use some form of color feature for matching people between frames, but often employ many heuristics that are not statistically reinforced. We instead hope to leverage the recent availability of several large datasets available for the task of re-identifying individuals in different camera frames. Provided the datasets extend a variety of environmental conditions and person orientations, a method trained on the dataset would provide a more statistically sound way of identifying people in consecutive images.

There are many competing methods that address the problem of person re-identification, but the methods are primarily focused on obtaining good re-identification rates on the benchmarks. Considerations like speed, simplicity, and extendability are

of less importance. For us however, these considerations are of great performance. Our re-identification rates should be as good as possible but we must maintain adequate speed for real-time person following. Because the person following problem often involves distinguishing only a handful of individuals, the re-identification rate requirements are less stringent than that of the benchmark scenario where matching is made against large datasets of individuals. Nevertheless, we will cover several competing methods that address the benchmark datasets to draw insight into what is important in obtaining good re-identification rates. However, we apply these insights in a way that we would work with the pose detector selected in the previous chapter. The final result is something that we believe provides useful and reliable information for the person following task.

## 6.1 Datasets

There exist several datasets that directly address the problem of person re-identification. The datasets vary primarily in the environment conditions, number of individuals, number of images, and number of cameras used. The paper [31] provides a comprehensive list of datasets addressing the task, from which we select datasets that would be most useful for our goal.

The *Market1501* dataset includes bounding box images of 1501 different people. The images are taken from six different cameras; five high-resolution cameras and one low-resolution camera. The bounding boxes are acquired by both hand-labeling and use of a DPM detector. Each bounding box has  $64 \times 128$  pixels and is

associated with a single person’s identity. This dataset is commonly used to assess competing methods for the human-reidentification, and comprehensive benchmark results are provided. Unfortunately, the full images are not available so we are dependent on the labeled bounding box instead of the bounding box determined by the pose detector. Thus, the dataset does not allow us to evaluate the system from end-to-end. Nevertheless, the dataset is simple and an excellent option for validating the identification architecture.

The *Person Re-identification in the Wild* (PRW) dataset utilizes the images from Market1501 but includes the full images and bounding boxes for 932 different people. The advantage of this dataset over the Market1501 dataset is that it allows us to evaluate the performance of our system when there are multiple people in the same image. This is crucial for our implementation because we never explicitly crop-out people in the image before determining their identity. Instead, we embed the identity of individuals spatially in a feature map and then parse out the identities after detection. If we were to use the Market1501 dataset where there is only one individual per image, we could not guarantee that the identity was being embedded around the location of the individual. This would cause issues if we then used the system with multiple people in the scene.

Like the PRW dataset the *Large Scale Person Search* (LSPS) dataset includes the full image frames and bounding box annotations. The LSPS dataset contains 18,184 images, 8,432 people, and 99,809 annotated bounding boxes. The advantage of the LSPS dataset over the others is that images are acquired from a wide variety of cameras in a larger variety of scenes. This variability is desirable since it increases

the likelihood that the trained method would extend to different cameras and environments. Utilizing the LSPS dataset is the subject of future work; the dataset is only available at request of the authors.

We used the PRW dataset in this thesis since it was readily available and contained multiple individuals per image.

## 6.2 Recent Work

Recently, L. Zheng released a comprehensive report of state-of-the-art methods competing in the Market1501 benchmark [32]. The top competing algorithms, including [28] [29] [23] [22] [30], address the problem using convolutional neural networks operating on each bounding box image. While these methods are not suitable for real-time operation they provide valuable insight into what is important in re-identifying individuals in images.

One of the major challenges in the Market1501 competition is addressing the limited size of the dataset. The dataset contains only 1501 identities from a limited number of cameras which is relatively small compared to the popular Labeled Faces in the Wild (LFW) dataset used for face recognition which contains 5,749 identities [23]. Most of the methods are less focused on the specifics of the network architecture than they are on methods for making the results generalize well. The work by M. Geng [23] uses a standard GoogleNet base network, but focuses on ways to make knowledge from larger datasets like ImageNet transfer to the human re-identification problem. They achieve a state-of-the-art rank 1 detection accuracy of 83.7% across

751 identities in the Market1501 benchmark. In the work by Z. Zheng [30] they address the small dataset size by learning a generative adversarial network (GAN) to generate additional unlabeled samples. With this method they achieve a competitive rank-1 accuracy of 78.06%.

The work by L. Zheng [29] directly addresses the problem that misaligned detection and pose variation have on re-identification. In their paper they apply a state-of-the-art CNN pose detector to the bounding box image to detect key joints of the person in the image. Using the joint locations they construct a *PoseBox* feature, which is a compound image containing the person’s torso, arms, and legs aligned in a specific position and orientation.

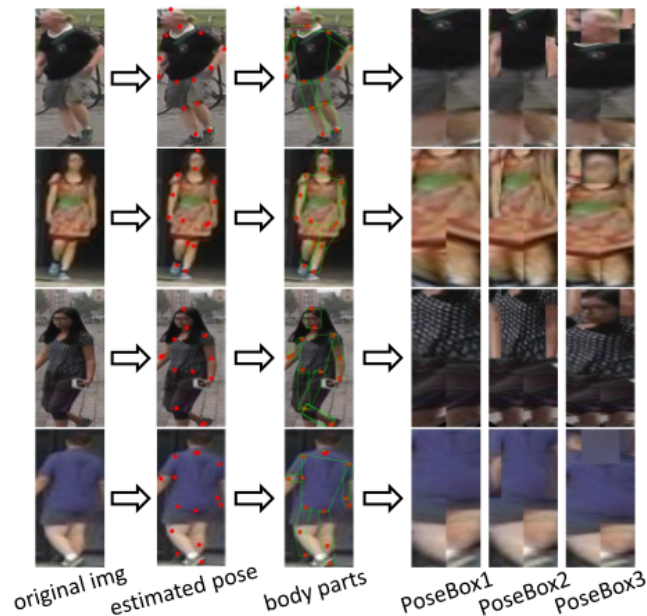


Figure 6.1: The PoseBox feature. The body parts are detected by a CNN pose detector; affine transformations are used to create the PoseBox features on the right. [29]



This feature image, shown in Figure 6.2, is used as the input to a network that learns a discriminative identity embedding. A clear issue with the PoseBox feature is that misdetections of the joint locations could dramatically effect the matching performance. To alleviate this, they input the original bounding box image and a vector containing the confidence of each joint detection as features to their embedding network. Using this method they achieved a competitive rank-1 accuracy of 79.33%.

### 6.3 Proposed Identification Architecture

The work by L. Zheng in [29] presents a pose invariant feature that achieves competitive results in the person re-identification task. The feature even performs well without the use of deep learning. This suggests that pose plays an important role in the identification of individuals. Because the person detection algorithm we selected uses the individual’s pose, we may include the pose information at little extra cost. Unfortunately, the method of [29] requires us to parse the pose for each person to generate a feature vector. Use of this method would cause the computational complexity to increase depending on the number of people in the image. The performance would also be highly dependent on the pose parsing performance. In addition, we could not utilize features used in pose detection that may provide important information for the identification task. We instead suggest augmenting the pose detection network with an identification network that can spatially embed the identity of individuals. This fixes the computational complexity, which is important

for real-time operation, and allows the network to utilize the pose confidence maps and part affinity field maps that were already calculated in the pose detection phase. By doing this we are able to incorporate knowledge of pose into the identification procedure without the need for parsing.

### 6.3.1 Network Structure

The proposed network architecture is an extension of the convolutional neural network by Z.Cao [17]. As covered in the previous chapter on pose detection, the network by Z. Cao takes as input a color image, and outputs confidence maps for the part locations as well as a set of part affinity fields which are used to associate the parts in the image. We use a pre-trained version of this pose detection sub-network, and fix the parameters when learning the spatial identity embedding. Our proposed extension includes a *low-level* network that runs parallel with the pose detection network and a *mixing* network that spatially embeds the identity using the low-level and pose features. The intuition behind the design is that the low-level network detects features like textures and colors and the mixing network associates these low-level features with body parts to create a feature vector representing the individual's identity.

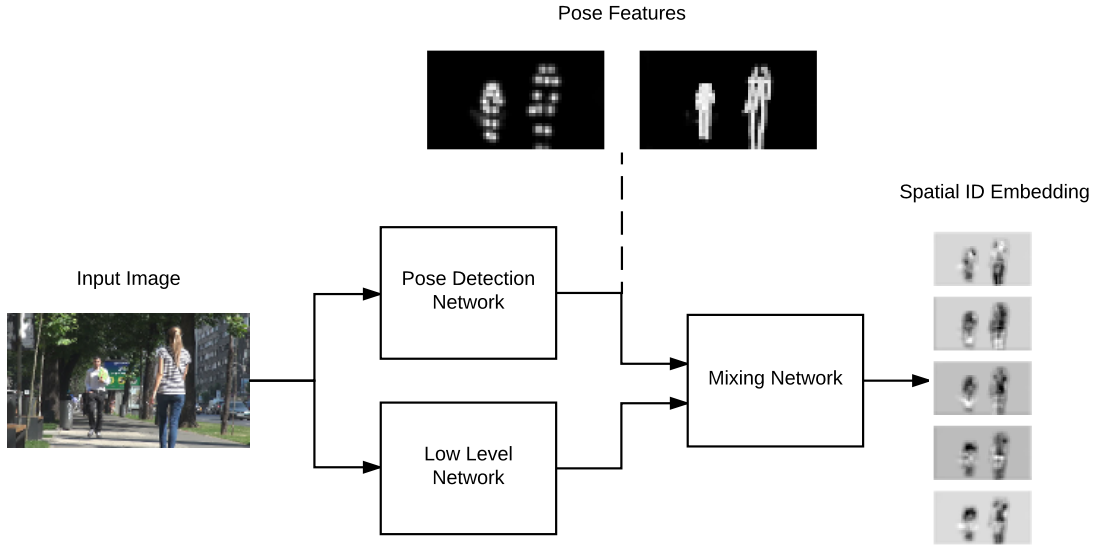


Figure 6.2: The proposed spatial identity embedding network.

### 6.3.1.1 Pose Detection Sub-Network

The pose detection sub-network,  $f_{pose}$  is identical to the network by Z. Cao [17]. The pose detection network is made of multiple stages, each of which improves the detection accuracy. In the pre-trained version we used there were six total stages. Instead of utilizing all stages for part detection, we only utilize the second stage. This results in quicker detection rates, albeit at lower accuracy. However, we found the accuracy sufficient for our task. The network takes the preprocessed color image of size  $M, N$  as input and provides the part confidence maps and the part affinity fields of spatial size  $m, n = M/8, N/8$  as outputs. In the original work these maps are used for associating parts with each other to determine the pose of multiple people in the same image. In our work we also utilize the part maps and part affinity fields in the mixing network to provide context to lower level features. Details on

this sub-network are covered in the previous chapter on pose detection.

$$f_{pose} : \mathbb{R}^{M \times N \times 3} \rightarrow (\mathbb{R}^{m \times n \times 18}, \mathbb{R}^{m \times n \times 38})$$

$$I_{cmap}, I_{paf} = f_{pose}(I)$$

### 6.3.1.2 Low-Level Sub-Network

The low-level sub-network,  $f_{low-level}$ , runs independent of the pose detection sub-network. The network contains residual units similar to those introduced in [35], but excludes batch normalization because we did not perform training in batches due to computational constraints. The network structure is depicted in figure 6.3.

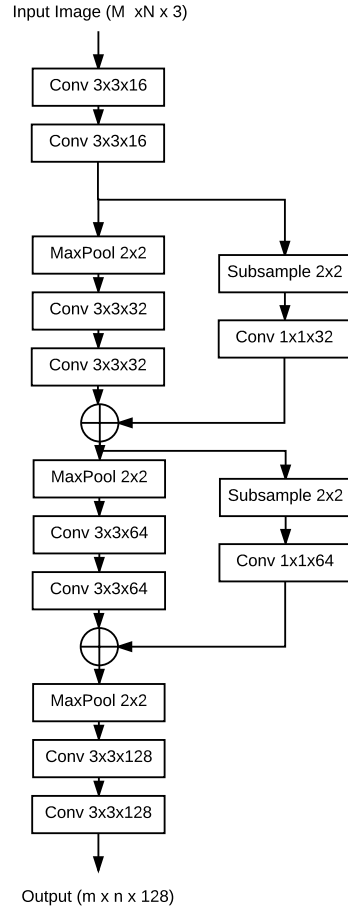


Figure 6.3: The low-level convolutional neural network

The network takes as input the image  $I \in \mathbb{R}^{M \times N \times 3}$  and performs two sequential convolutional layers with a filter size of  $3 \times 3$  and 16 channels. Max pooling of stride  $2 \times 2$  is performed after these convolutional layers, halving the spatial dimension of the feature maps. Another two  $3 \times 3$  convolutional layers are added but with double the channels. The output of these convolutional layers is summed with a skip connection from the output of the first two convolutional layers. This skip connection involves subsampling and identity activation  $1 \times 1$  convolution to make the dimensions match. This process is repeated until an output dimension of

$m \times n \times 128$  is reached. Note that all activations in this network are rectified linear units (ReLU), aside from the skip connection convolutions which have identity activations. In summary,

$$f_{low-level} : \mathbb{R}^{M \times N \times 3} \rightarrow \mathbb{R}^{m \times n \times 128}$$

$$I_{low-level} = f_{low-level}(I)$$

### 6.3.1.3 Mixing Sub-Network

The mixing network is responsible for generating the spatial identity embeddings. In the first stage of the mixing network we approximate the binary mask of individuals in the image using the part confidence maps and part affinity fields. This binary mask is multiplied by the low-level feature network output to reduce the impact of the surrounding environment on the identity embedding. We found this improved generalization and training convergence. To compute the mask we first compute the masks for the confidence maps and part affinity fields by taking the max of the thresholded magnitude along each channel.

$$I_{cmap-mask}[i, j] = \max(I_{cmap}[i, j]) > \text{threshold}$$

$$I_{paf-mask}[i, j] = \max(\|I_{paf}[i, j]\|) > \text{threshold}$$

where the norm,  $\|I_{paf}[i, j]\|$  is applied for each part affinity field channel corresponding to the same part association. The *max* operation is applied channel-wise, thus

preserving the spatial dimensions. The combined mask is then

$$I_{mask} = I_{cmap-mask} \vee I_{paf-mask}$$

where  $\vee$  represents the logical OR operation. We then multiply each channel of the low-level network output by this mask.

$$I_{low-level-masked} = I_{mask} \odot I_{low-level}$$

where  $\odot$  represents element-wise multiplication which is applied to each channel. After masking the low-level features we combine them with the pose features via concatenation along the channel dimension.

$$I_{features-mixed} = [I_{low-level-masked} | I_{cmap} | I_{paf}] \in \mathbb{R}^{m \times n \times 184}$$

we then perform two  $3 \times 3 \times 64$  convolutions with ReLU activations on this mixed feature channel followed by one  $1 \times 1 \times 64$  convolution with identity activation, which results in the final spatial identity embedding,  $I_{idmap} \in \mathbb{R}^{m \times n \times 64}$ . In summary,

$$f_{mixing} : (\mathbb{R}^{m \times n \times 128}, \mathbb{R}^{m \times n \times 18}, \mathbb{R}^{m \times n \times 38}) \rightarrow \mathbb{R}^{m \times n \times 64}$$

$$I_{idmap} = f_{mixing}(I_{low-level}, I_{cmap}, I_{paf})$$

where  $f_{mixing}$  includes the operations described in this section. The functionality of the combined network can then be summarized,

$$I_{cmap}, I_{paf} = f_{pose}(I)$$

$$I_{idmap} = f_{mixing}(f_{low-level}(I), I_{cmap}, I_{paf})$$

### 6.3.2 Identity Parsing

In the previous section we presented the network structure for constructing the spatial identity embedding, but this spatial embedding has little meaning until parsed into the final identity embedding vectors for all the people detected in the image. The parsing stage is simple and requires only taking the maximum along the spatial dimensions within the bounding box associated with the person in question. During the training phase we used the annotated bounding boxes provided by the dataset, but during real-time operation we used the bounding boxes calculated from the boundary parts parsed by the pose detector. The identity parsing scheme is outlined in figure [6.4](#).



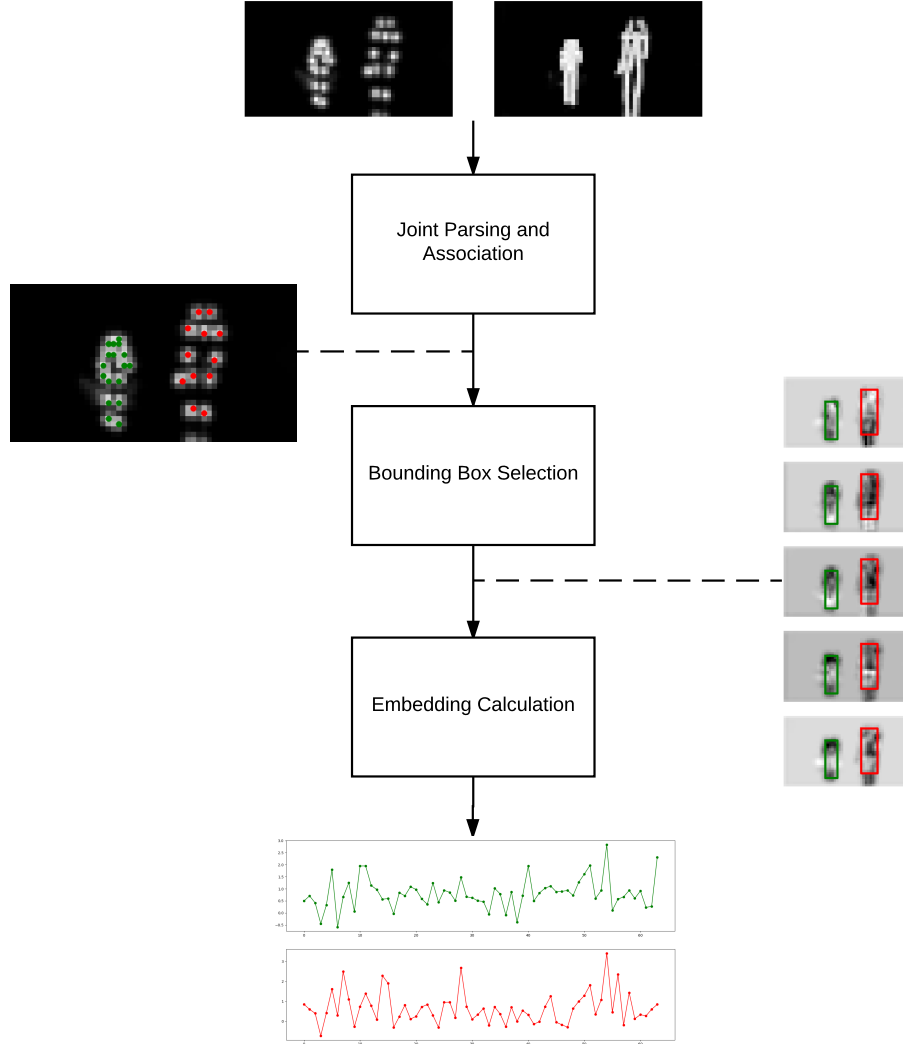


Figure 6.4: Parsing scheme for the proposed identity embedding network.

First, we perform part detection and association as described in the previous chapter on pose detection. This yields the list of part locations and visibilities for all  $N$  people detected in the image.

$$\{\{p_{ij}\}_{j=1:18}\}_{i=1:N}$$

where  $p_{ij} = (v_{ij}, x_{ij}, y_{ij})$  contains the visibility  $v$  and coordinates  $(x, y)$  for part  $j$  belonging to person  $i$ . To determine the bounding boxes we select the minimum and maximum spatial coordinates over each visible part for each person  $i$ .

$$x_i^- = \min(\{x_{ij}\}_{j|v_j=1})$$

$$y_i^- = \min(\{y_{ij}\}_{j|v_j=1})$$

$$x_i^+ = \max(\{x_{ij}\}_{j|v_j=1})$$

$$y_i^+ = \max(\{y_{ij}\}_{j|v_j=1})$$

$$box_i = (x_i^-, y_i^-, x_i^+, y_i^+)$$

the identity embedding vector  $z_i \in \mathbb{R}^{64}$  for person  $i$  is then calculated,

$$z_i[k] = \max(I_{idmap}[x_i^- : x_i^+, y_i^- : y_i^+, k])$$

$$z_i = [z_i[1], z_i[2], \dots, z_i[64]]$$

this identity embedding vector is trained so that the Euclidean distance to identity embedding vectors belonging to the same person is relatively small compared to the distance to embeddings of other individuals. This type of training is accomplished by minimizing a *triplet loss* which we overview in the following section on training. Details of how to use the embedding in a tracking scenario are covered in the next chapter.

## 6.4 Implementation

The above network was implemented and tested using the Tensorflow machine learning library. We converted the pose detector network, which was originally de-

veloped using the Caffe deep learning framework, into a fixed tensorflow graph which we augmented with our low-level and mixing networks. For the entirety of training and testing the weights of the pose detection network were fixed. Preprocessing and parsing were accomplished using Python and the OpenCV libraries. Z. Cao released an entire open-sourced version of his pose detection network and parsing software, but we found it easier and more portable to implement our own version of pose parsing in Python. We did however use the pre-trained weights from his open-sourced software. Training and testing were both done on a Dell Inspiron 7559 laptop equipped with a solid state drive, Intel i5 processor, and Nvidia 960m GPU. The Nvidia 960m GPU is capable of 1.8 TFLOPS, which is comparable to the embedded Nvidia Jetson TX2, which could be used in a portable implementation.

## 6.5 Training

The network was trained using only the PRW dataset, with some slight modifications. First, we removed any of the annotated bounding boxes that were smaller than a certain area. These boxes were essentially indistinguishable at the resolution we were using. Second, we limited the number of training samples per individual to 10. Some of the individuals were overrepresented in the dataset so this helped prevent bias towards any individual. Third, we used a larger portion of training identities than the benchmark. We used 749 identities during training and 134 during evaluation. All of the images were resized to a width of 270 pixels and a height of 480 pixels. For most of the images this maintained the aspect ratio, but resulted

in a slight skew for the others. We found this skew acceptable and even desirable as it added variety to the data. To add additional variety we randomly rotated and scaled the images. The rotations were done uniformly within 15 degrees and the scaling within 30% of the original image size. The scaling maintained the aspect ratio. We found that these random operations aided in generalization.

An important step in training the network was selecting a loss function that could be used for the identification task. Following the method of FaceNet, which achieved state-of-the-art recognition accuracy on the Labeled Faces in the Wild (LFW) dataset, we selected a triplet loss to train our network [25]. To compute the triplet loss, we first select an *anchor* sample, *positive* sample, and *negative* sample. The anchor sample is a specific image of the identity in question, the positive sample is a different image of the same identity, and the negative sample is of a different identity. The triplet loss is the relative pairwise distance between the anchor and the negative and positive samples up to a margin  $\alpha$ ;

$$d_{positive} = \|z_{anchor} - z_{positive}\|_2^2$$

$$d_{negative} = \|z_{anchor} - z_{negative}\|_2^2$$

$$L_{triplet}(z_{anchor}, z_{positive}, z_{negative}) = \max([0, \alpha + d_{positive} - d_{negative}])$$

In an ideal scenario we would expect that after training there will be a margin of  $\alpha$  between  $d_{positive}$  and  $d_{negative}$ . It is important to note that this loss function does not constrain the samples corresponding to one identity to lie within an  $L2$  sphere.

Instead, the loss function acts on the relative distance between positive and negative samples so that when comparing any two images to a query image we could select which image is more similar to the image in question. The advantage of this is that it allows samples belonging to the same identity to lie freely on a manifold while maintaining discriminability from samples of negative identity [25]. Unfortunately, the triplet loss has a disadvantage in the tracking scenario; we cannot *absolutely* decide whether an individual is of the same identity or not. That is, using the distance metric we can only determine who is closest in identity to the target we are tracking. Thus, for tracking we require additional logic for absolute reidentification, which could involve maintaining a database of identity embeddings. Nevertheless, maintaining a database of identity vectors is suitable for many applications including the example in the next chapter. In the future, exploring identity embedding loss functions that do constrain embeddings of the same identity to lie within an  $L2$  sphere might improve the generality of the system.

When training the embedding it was important to select triplets that most violated the margin constraint above. This process is called *hard-negative mining* and is critical during training [25]. If we were to select training samples randomly without hard negative mining we would likely select many triplets that do not violate the constraint and have zero loss. These samples do not contribute at all to training. Unfortunately, the process of determining hard negative triplets is not trivial. To determine the hardest negative triplet we must first compute the identity embeddings of all samples in the training set and then compute the pairwise distance between all of the samples. At first we found this process prohibitively

slow for experimentation given our hardware constraints, but after some computation optimizations we reduced the negative mining process to 2 minutes per epoch on our hardware. For comparison, training through an entire epoch took about 12 minutes. The primary optimization that improved training time was pre-computing the pose features and storing them on disk. This was acceptable because the pose sub-network weights were fixed. Note that rather than using the hardest negative we randomly selected from the worst 3 negative samples. This helped prevent the network from focusing only on highly occluded or incorrectly labeled samples that may be present in the dataset.

Before training we initialized the network weights by random sampling from a normal distribution with a mean of 0.0 and standard deviation of 0.05. The biases in the network were all initialized as 0. We used the Adam optimization method introduced in [36] with parameters  $\alpha$ ,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.999$ , and  $\epsilon = 10e^{-8}$  applied to the triplet loss and optimizing over the low-level and mixing network weights and biases. Training was performed for each triplet individually (we did not use batches due to memory constraints). We repeated the training process for over 20 epochs.

## 6.6 Accuracy

At the end of training we evaluated the rank-1 and rank-5 lookup accuracy against the training and test sets. The rank-1 accuracy corresponds to the percentage of images correctly matched with an image of the same identity. The rank-5 accuracy is the percentage of images that contain a correct match in the top-5 closest

queries. We used the first 500 samples from each set ordered by identity. This corresponded to 75 different training identities and 74 different identities from the test set. There were about 7 images per identity on average. The results are summarized in figure 6.6.

| Rank | Training Accuracy | Test Accuracy |
|------|-------------------|---------------|
| 1    | 71.2%             | 65.8%         |
| 5    | 87.4%             | 82.4%         |
| 10   | 92.4%             | 88.8%         |

Figure 6.5: Train and test accuracy using 500 images of 75 identities (train set) or 74 identities (test set).

The above figure is used primarily as a comparison between the train accuracy and test accuracy, as it contains the same number images and similar number of identities. Below are the accuracies on the full training and test sets. Note that the training accuracies are lower because the training set is much larger.

| Rank | Training Accuracy |
|------|-------------------|
| 1    | 46.3%             |
| 5    | 70.5%             |
| 10   | 79.4%             |

Figure 6.6: Training accuracy over entire training set of 5204 samples and 749 identities.

| Rank | Test Accuracy |
|------|---------------|
| 1    | 60.83%        |
| 5    | 80.7%         |
| 10   | 86.6%         |

Figure 6.7: Test accuracy over entire test set of 909 samples and 134 identities.

It is difficult to draw a direct comparison between our results and previous work for several reasons. Previous methods that directly address the real-time robot person following problem often do not provide concrete statistics on the performance. Thus any comparison to these previous results would have to be speculative and qualitative. The Market1501 dataset has arguably the most competing algorithms, but these algorithms operate only on the bounding box images and are not configured for real-time use, especially when there are multiple people in the same image. The benchmark results for the PRW dataset utilize data similar to ours, but are dependent on using a DPM for human detection and evaluating the identity of each bounding box separately. This too is not suited for real-time use. In addition, they utilize the full resolution of the training images. It is important to note that the recall accuracy of method reported in PRW includes the detection phase. In our method we utilize the labeled bounding boxes. Nevertheless, we hope to show that our results are comparable to competing methods for human re-identification while running in real-time (including the pose detection).



| Method              | Realtime? | Dataset    | No. Ids | R1     | R5     | R10    | R20    |
|---------------------|-----------|------------|---------|--------|--------|--------|--------|
| Z. Zheng (2016)     | No        | Market1501 | 751     | 79.51% | 90.91% | 94.09% | 96.23% |
| L. Zheng (2017)     | No        | Market1501 | 751     | 79.33% | 90.76% | 94.41% | 96.52% |
| J. Liu (2016)       | No        | Market1501 | 751     | 45.1%  | 70.1%  | 78.4%  | -      |
| Best of [33] (2016) | No        | PRW        | 450     | 48.3%  | -      | -      | 78.8%  |
| H. Bouma (2013)     | Yes       | Own        | 77      | 20%    | 40%    | 53%    | 73%    |
| Ours (2017)         | Yes       | PRW*       | 74      | 65.8%  | 82.4%  | 88.8%  | -      |
| Ours (2017)         | Yes       | PRW*       | 134     | 60.83% | 80.7%  | 86.6%  | -      |

One of the closest functional comparisons to our system that operates in real-time is that of H. Bouma. They evaluated their system on a similar sized dataset to our 74 identity dataset, though our system achieves much higher recognition rates. A few state-of-the-art methods competing in the Market1501 benchmark achieve very high recognition rates (79.51% rank 1 over 751 identities) but operate on fixed resolution bounding boxes cropped from higher resolution images. Distinguishing multiple people in the image with these methods would require accurately detecting the bounding boxes, which using a DPM could take several seconds on our hardware. These methods also require that we run the network once for every person in the image. Our live system takes approximately 100-110ms per 240x320 frame, which includes image retrieval, preprocessing, detection, parsing, and identification for all individuals in the image. We will detail this full real-time detection system in the next chapter.

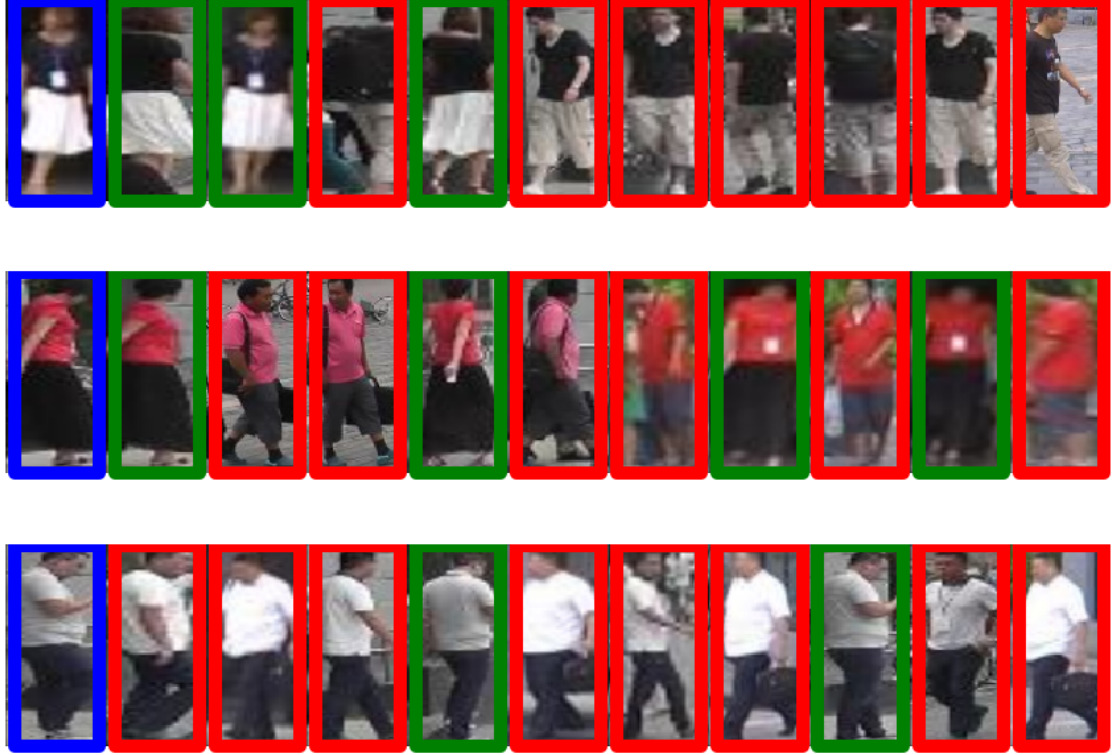


Figure 6.8: Some example top 10 query results against the test database of 74 identities and 500 images. The query images were selected arbitrarily as indices 100, 200 and 300 to avoid bias. The images highlighted in blue are the query images. The remaining images are highlighted in green if they are from the same identity or red if they are not. They are sorted from left to right based on their embedding distance to the query image; the leftmost image being the closest.

## Chapter 7: Example Application

In the previous chapter we detailed our network architecture for computing the identity embedding vectors of individuals in the images. In this chapter, we describe a real-time application that utilizes these embeddings to track multiple people from a live camera feed. The application was developed entirely in Python using OpenCV for the graphical user interface and TensorFlow to run our detection and identification network. The user interface depicted below is simply a live image feed of the detected bounding boxes and parsed joint locations for each person in the image. To identify individuals we ask them to stand in front of the camera and we press a number between 1 and  $N$  corresponding to their identity. The number of individuals could be arbitrarily large but we limited ourselves to 5 for this example application so that each individual had a distinguishable color in the display. When the number corresponding to an individual was pressed, we recorded their identity embedding at that instant and stored it in a database. When the user hit the character 't' we trained an SVM using the identities in the database. Note that using this method we could collect multiple identity embeddings per person, which we typically found improved the performance. After the SVM was trained, the system would make live predictions of the individuals identities and display them

by color coding the bounding boxes.

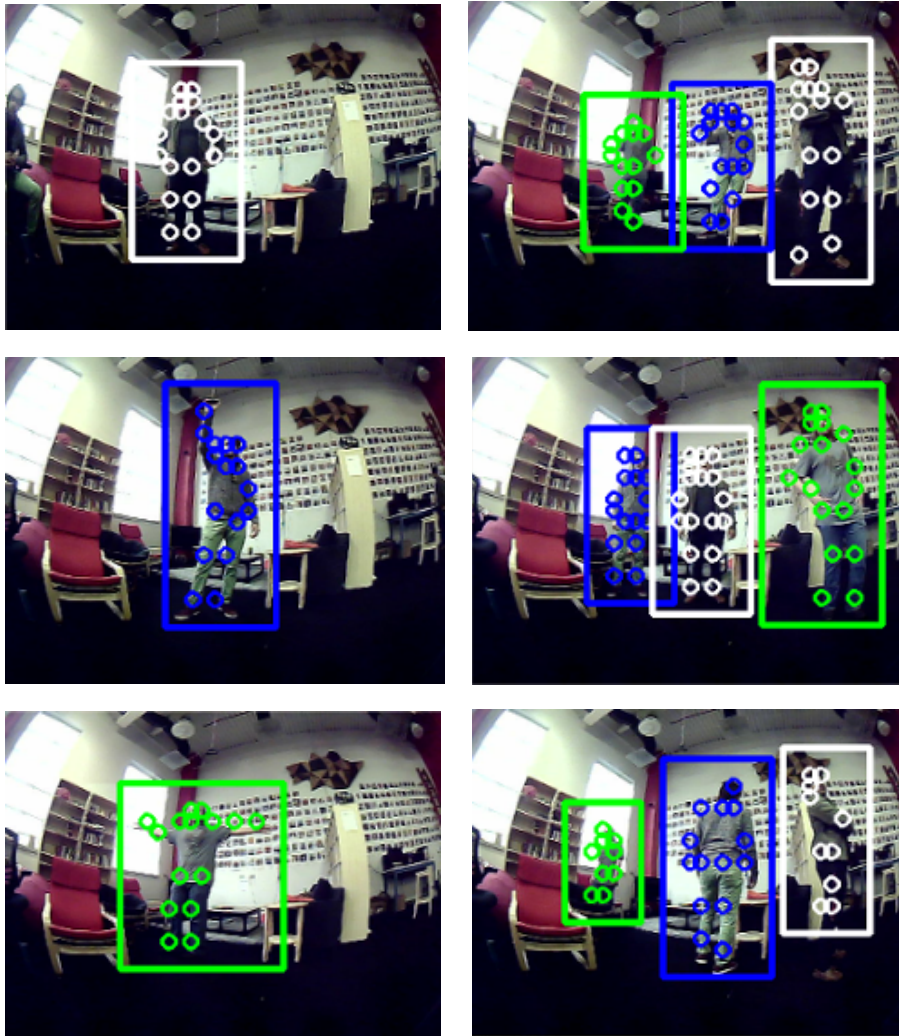


Figure 7.1: The live identification and pose detection system operating on 3 individuals. On the left are example images from the identity learning phase. On the right are the detection results. 7 identity embeddings were recorded per individual in this example. We apologize for the difficult visibility caused by the annotations. Note in this example we used the inexpensive fisheye lens camera we intend to use for person following which had largely different lighting and distortion characteristics than the training images. The system was still able to work reliably.

The example application presented in chapter is for demonstration. A more user-friendly and practical application would be to create the identity embedding database in response to a user 'clicking' himself on a video feed that we stream to their phone, or some other intuitive method. The pose provided by the system offers the possibility to train the system in response to gestures as well. For instance, we could record the identity embedding by detecting the individual in the image raising their right hand and giving the voice command 'follow me'. We consider the flexibility provided by the pose and identity embeddings as an advantage of the system.

## Chapter 8: Conclusions and Future Work

In this work we presented a method capable of detecting multiple people, their pose and identities in real-time. The system is capable of running at approximately 10Hz on moderate hardware (Nvidia 960m GPU, Intel i5 CPU). We believe this system provides a useful set of information to enable reliable person following. Our system uses only a single camera and we found that it generalizes well to cameras of varying field of view and distortion. This makes our approach less hardware specific than most prior approaches for tracking people in the context of person following.

The generality of our system and the rich set of information that it provides makes it a useful tool for other tasks related to human machine interaction. The system could be easily extended to respond to body gesture commands given by specific individuals. Moreover, the pose detection system we use also detects the locations of the eyes, ears and nose for people in the image. We expect that in the future this could be used to learn facial identity features that persist even when the user changes clothes. This would be useful for initializing the tracking system without the need for human input. One of the main issues that we found in our real-time approach is that the identity embedding fails if people are partially overlapping in the image. In the future we would like to re-train the spatial identity

embedding method so that the embedding is stored in the joint locations rather than in the general bounding box location. We would also like to learn some measure of how likely two individuals are a match. This would allow us to identify individuals even when they are partially overlapping in the image.

We hope that this work provides information related to the person following problem and believe that our system could be easily extended into an infrastructure for long-term human robot interaction.

## Bibliography

- [1] M. Bakar, M. Amran, "A Study on Techniques of Person Following Robot," *International Journal of Computer Applications*, vol. 125, no. 13, 2015
- [2] D. Beymer, K. Konolige, "Real-Time Tracking of Multiple People Using Continuous Detection," *IEEE Frame Rate Workshop*, 1999
- [3] D. Calisi, L. Iocchi, R. Leone, "Person Following through Appearance Models and Stereo Vision using a Mobile Robot," *VISAPP (Workshop on Robot Vision)*, 2007
- [4] Z. Chen, S.T. Birchfield, "Person Following with a Mobile Robot Using Binocular Feature-Based Tracking," *Intelligent Robots and Systems, 2007. IROS, 2007.*, 2007
- [5] I. Ćirić *et al*, "Intelligent Optimal Control of Thermal Vision Based Person-Following Robot Platform," *Thermal Science*, vol. 18, no. 3, 2014
- [6] R. Gockley, J. Forlizzi, R. Simmons, "Natural Person-Following Behavior for Social Robots," *Proceedings of the ACM/IEEE international conference on Human-robot interaction. ACM.*, 2007
- [7] B. Ilias *et al*, "A Nurse Following Robot With High Speed Kinect Sensor," *ARPJ Journal of Engineering and Applied Sciences*, vol. 9, no.12, 2014
- [8] M. Kobilarov *et al*, "People tracking and following with a mobile robot using an omnidirectional camera and a laser," *Robotics and Automation*, 2006
- [9] H. Koyasu, J. Miura, Y. Shirai, "Realtime Omnidirectional Stereo for Obstacle Detection and Tracking in Dynamic Environments," *Intelligent Robots and Systems*, 2001



- [10] H. Kwon *et al*, "Person Tracking with a Mobile Robot using Two Uncalibrated Independently Moving Cameras," *Robotics and Automation*, 2005
- [11] K. Nishant, "Using Smartphone Sensors for Robotic Person Following," *ProQuest Dissertations Publishing*, 2016
- [12] D. Ribeiro *et al*, "A Real-Time Pedestrian Detector using Deep Learning for Human-Aware Navigation," *arXiv preprint arXiv:1607.04441*, 2016
- [13] J. Satake, J. Miura, "Robust Stereo-Based Person Detection and Tracking for a Person Following Robot," *ICRA Workshop on People Detection and Tracking*, 2009
- [14] S. Shaker, J.J. Saade, D. Asmar, "Fuzzy Inference-Based Person-Following Robot," *International Journal of Systems Applications, Engineering & Development*, vol. 2, no. 1, 2008
- [15] T. Yoshimi *et al*, "Development of a Person Following Robot with Vision Based Target Detection", *Intelligent Robots and Systems*, 2006
- [16] J. Shotton *et al*, "Efficient Pose Estimation From Single Depth Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no.12, 2013
- [17] Z. Cao *et al*, "Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields," *arXiv preprint arXiv:1611.08050*, 2016
- [18] M. Andriluka *et al*, "2D Human Pose Estimation: New Benchmark and State of the Art Analysis," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014
- [19] T. Lin *et al*, "Microsoft COCO: Common Objects in Context," *European Conference on Computer Vision. Springer International Publishing*, 2015
- [20] A. Newell, K. Yang, J. Deng, "Stacked Hourglass Networks for Human Pose Estimation," *European Conference on Computer Vision. Springer International Publishing*, 2016
- [21] A. Toshev, C. Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014
- [22] I. Barbosa *et al*, "Looking Beyond Appearances: Synthetic Training Data for Deep CNNs in Re-identification," *arXiv preprint arXiv:1701.03153*, 2017

- [23] M. Geng *et al*, "Deep Transfer Learning for Person Re-identification," *arXiv preprint arXiv:1611.05244*, 2016
- [24] J. Long *et al*, "Fully Convolutional Networks for Semantic Segmentation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015
- [25] F. Schroff *et al*, "FaceNet: A Unified Embedding for Face Recognition and Clustering," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015
- [26] C. Su *et al*, "Deep Attributes Driven Multi-Camera Person Re-identification," *European Conference on Computer Vision. Springer International Publishing*, 2016
- [27] R. Varior, M. Haloi, G. Wang, "Gated Siamese Convolutional Neural Network Architecture for Human Re-Identification," *European Conference on Computer Vision. Springer International Publishing*, 2016
- [28] Z. Zheng, L. Zheng, Y. Yang, "A Discriminatively Learned CNN Embedding for Person Re-identification," *arXiv preprint arXiv:1611.05666*, 2017
- [29] L. Zheng *et al*, "Pose Invariant Embedding for Deep Person Re-identification," *arXiv preprint arXiv:1701.07732*, 2017
- [30] Z. Zheng, L. Zheng, Y. Yang, "Unlabeled Samples Generated by GAN Improve the Person Re-identification Baseline *in vitro*," *arXiv preprint arXiv:1701.07717*, 2017
- [31] Gou. M, "Person Re-Identification Datasets," *Roboustsystems.coe.neu.edu*, retrieved 2017
- [32] L. Zheng *et al*, "Scalable Person Re-identification: A Benchmark," *Proceedings of the IEEE International Conference on Computer Vision*, 2015
- [33] L. Zheng *et al*, "Person Re-identification in the Wild," *arXiv preprint arXiv:1604.02531*, 2016
- [34] T. Xiao *et al*, "End-to-End Deep Learning for Person Search," *arXiv preprint arXiv:1604.01850*, 2016
- [35] K. He *et al*, "Deep Residual Learning For Image Recognition," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016

- [36] D. Kingma, "Adam: A Method For Stochastic Optimization," *arXiv preprint arXiv:1412.6980*, 2014
- [37] N. Dalal, B. Triggs, "Histograms of Oriented Gradients for Human Detection," *Computer Vision and Pattern Recognition*, 2005
- [38] P. Felzenswalb, D. McAllester, D. Ramanan, "A Discriminatively Trained, Multiscale, Deformable Part Model," *Computer Vision and Pattern Recognition*, 2008
- [39] P. Dollár *et al.*, "Pedestrian Detection: An Evaluation of the State of the Art," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol 34., no 4., 2012
- [40] P. Dollár, S. Belongie, P. Perona, "The Fastest Pedestrian Detector in the West," *BMVC*, vol.2, no. 3, 2010
- [41] A. Angelova *et al.*, "Real-Time Pedestrian Detection With Deep Network Cascades," *BMVC*, 2015
- [42] D. Bolme *et al.*, "Visual Object Tracking using Adaptive Correlation Filters," *Computer Vision and Pattern Recognition (CVPR)*, 2010
- [43] J. Henriques *et al.*, "High-Speed Tracking with Kernelized Correlation Filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, 2015