MARKERLESS HUMAN TRACKING FOR INDUSTRIAL ENVIRONMENTS

MARKERLESS HUMAN TRACKING FOR INDUSTRIAL ENVIRONMENTS

By

Mohanad Elshafie, B.Eng.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree

Master of Applied Science

McMaster University

MASTER OF APPLIED SCEINCE (2008) McMaster University

Mechanical Engineering Hamilton, Ontario

TITLE: Markerless Human Tracking for Industrial Environments

AUTHOR: Mohanad Elshafie, B.Eng. (McMaster University)

SUPERVISOR: Professor Gary Bone

NUMBER OF PAGES: xii, 153

ABSTRACT

This thesis presents a markerless multiple-camera vision-based 3D human tracking method for industrial environments. The method can track humans in the vicinity of moving robots without using skin color cues or articulated human models. It is robust to self-occlusions and to partial occlusions caused by the robot. Foreground pixels corresponding to humans are found by background subtraction. A convex polyhedron enclosing the human(s) is generated online by bounding the foreground pixels in 3D space. Experimental results are included for a single person and multiple persons walking near a moving PUMA robot in a cluttered environment. Reliable tracking at 11.2Hz is demonstrated using four cameras and a Pentium 4 PC. The tracking data may be used for online robot collision avoidance.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my research supervisor, Professor Gary M. Bone, for his valuable guidance and support. I also thank my friends and colleagues for their advice and encouragement. Finally, I would like to specially show appreciation to my family for their encouragement and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Introduction

1.1 Preface

The popular view of robotics consisted of humanlike machines that could walk, talk and perform actions as well as humans. Despite this popular view, industrial robots form the largest percentage of robotics sales and research. There are more than one million industrial robots worldwide [1]. According to the Robotic Industries Association, an industrial robot is an automatically controlled, reprogrammable, multipurpose manipulator programmable in three or more axes that may be either fixed in place or mobile for use in industrial automation applications [2].

Industrial robots can cause injury or death. The first victim of a robot related accident in North America was Harry Allen in 1984. Harry Allen worked for Diecast Corp. in Jackson Michigan as die-cast operator. He entered a restricted area where the robot was operating. He was presumably trying to clean scrap metal from the floor near the robot. While there, he got in the way of the robot and was pinned between a factory pole and the industrial robot. He died in the hospital five days later [3].

Effective robot safeguarding starts with understanding the patterns of injuries that have taken place in robot operations. United Auto Workers (UAW), a labour union which represents workers in the United States, Canada, and Puerto Rico, has collected information about accidents related to robot in North America. In their review "Robot Related Injuries" they describe the following accident:

A robot was being used to sand off a part of a car body. Every 20 cars the robot would return to a station and the sanding paper would be replaced automatically. The sanding paper changer was not functioning properly. An electrician entered the cell through an interlocked gate. A fellow skilled worker reset and started the line. The electrician was looking over the sanding paper changer and did not realize that it was the 20th car. The robot moved quickly to the station, striking the employee on the temple and forced his head against the fixture. He suffered a fractured skull. The other employee hit the emergency stop switch to stop the robot from causing further injury. [4]

Worldwide, there are around 22,000 robot-related accidents occurring annually [1]. These accidents often happen when an operator is inside the work area, called a *workcell*, of a robot. The operator enters the workcell to perform a routine task while unaware that the robot is operating. Then, accidentally, the robot hits the person while trying to perform its task. Accidents usually happen due to negligence by the operator.

Current safety standards in robotics environment, *ANSI/RIA/ISO 10218-1-1999 Robots for Industrial Environment - Safety Requirements Part 1 – Robot*, don't allow humans to operate in close proximity to the robot. To prevent accidents, robots are isolated using safety methods such as: Fences, sensor mats, light curtains and laser scanners. They merely monitor two dimensional planes. To guarantee secure protection, various sensors must be coordinated and aligned, and a whole series of cables are laid. This produces a complex structure.

There is also the problem of workcell flexibility. With the current safety methods, the robot operator is limited to the physical layout and space available at hand. If the operator needs a new workcell arrangement they will have to manually reconfigure all of the objects in the workspace and acquire extra structures that are not already available.

Furthermore, these safety methods occupy a large floor space. Floor space in factories can be financially quantified. The cost of the floor space is dependent on the industry. One of the major obstacles for small and medium-size companies is the big footprint robots require. This is due to installing the robot and requiring fences around the robot. For example, a robot with a two metre long arm will need a safety zone that's at least sixteen metre squared. That is because arm of the robot will rotate in a circle defining its workspace. This workspace will then be protected by **4m x 4m** fence.

In addition, the operation of these safety sensors does not register the type of the violation, so their response behaviour is limited. When the sensor is triggered by a person entering the detection zone, the emergency stop is usually switched on immediately. There is no evaluation of the potential risk to the personnel. This can cause a large downtime in factories. Subsequent stations within the production line may also suffer. Thus the whole production is effected directly or indirectly. This issue is a concern expressed by professionals in the industry. Mike Callardo, vice-president, product management with ABB Inc enquired:

"Does the robot have to stop in the presence of a human? What about slowing down when people are near, or moving only in a restricted path?" [5]

The next generation of robots should allow robot-human collaborations. In industry humans need to operate in close proximity to the robot to achieve tasks that are hard to achieve by the robot or the human on their own. For applications where no physical interaction is required, the robot should move out of the person's way so the same factory floor space can be safely shared. There are still many jobs that humans do better than robots. This collaboration will usher in fields where humans work closely to robots such as assembly, spot welding and tasks in unstructured environments. In the automotive assembly, for example, the robot can provide the power to lift heavy components, while the human guides them into place.

In robot-human collaboration systems the human operator will move inside the workcell and perform actions alongside the robot. The system must understand actions performed by the human and execute controlled actions to avoid injuring the person. A safety system will be required to follow the human actions inside the workcell. This safety system will have to provide safety in real-time, be inexpensive, easy to install, require the least amount of floor space and has the least amount of downtime.

1.2 Computer Vision Systems

Vision is a natural cue for humans. It is an important sensor to assimilate the surrounding and perform actions based on the information. That is why research has aimed at developing systems that can perform tasks similar to the tasks performed by the human vision. This will enable computer vision systems to carry out tasks such as detecting, locating, and tracking, among other tasks. The first vision system in an industrial environment was Consight installed by General Motors at their foundry in St. Catherines, Ontario, Canada in 1970.

Human tracking is interesting from the point of view of research. This is due to the combination of the high number of potential applications and the task's inherent complexity. Human tracking includes a number of hard problems such as inferring the motion of a highly articulated and self-occluding non-rigid 3D human subject from images. In addition, the presence of multiple people, changes in human posture, gestures and human interactions add to the complexity of human tracking. This complexity makes the research area challenging from a purely academic point of view. From an application perspective, computer vision sometimes provides the only non-contact solution making it very attractive.

Cameras can be used as sensors to monitor an area. Compared to other non-contact sensors (e.g. time of flight sensors) they provide a wealth of information about the scene. Some of the difficulties associated with using cameras for tracking people are discussed below.

Occlusion: It means closure or blockage. The camera system must see the person before it starts to track. Occlusion can happen in two different ways. An object can be placed between the camera and the subject occluding the camera view. The second way is self occlusion, where part of the subject is occluded by another body part. This becomes problematic when the vision system tries to locate the different body parts.

Shadows: The appearance of shadows changes the property of the scene. Shadows produce a silhouette of a person that can be mistaken for a person by the camera. More details on the problem of shadows are presented in section 2.2.2.

Loss of the third dimension: When a 3D scene is projected onto a 2D image, all information about the depth dimension is lost. Depth information can be retrieved from multiple cameras using visual perception.

Motion blurs and fast motions: The camera has a shutter with a certain shutter speed. The shutter allows light to pass to the sensor for a specific period of time. During that period of time, the subject must remain static; otherwise the image will be blurred. Fast moving objects require cameras with fast shutter speeds. Fast shutter speeds will allow less light to pass to the camera sensor, hence producing darker images.

Clothing: Systems that are designed to track limbs of subjects face the problem of occlusion for people wearing certain clothing. If the subject is wearing a dress, then the legs will not be visible to the system. In this case, the tracking system will fail.

Similarity between limbs and other structures: The camera system might incorrectly distinguish between body parts and other structures in the scene. If a person is sitting on a chair, the system will need to distinguish between the legs of the person and the legs of the chair.

Diverse Human Bodies: Although people have different body shapes, length of limbs and different weights, they all share the same skeleton structure. The camera system should be dynamically able to fit a skeleton structure to people with different body shapes and sizes.

Different Poses: The human body is very dynamic. It can be oriented in many different poses such as standing, walking or lying down, etc. The camera system should be able to track people in all these different poses.

Camera Resolution: The size of the field of view needs to be constrained such that a change in the scene, for example the movement of the person, registers as a significant pixel change.

1.3 Objective and thesis structure

The objective of the thesis is to develop a markerless vision-based human tracking system for industrial environments. The goal of the system is to define the 3D space occupied by the human(s). The system will utilize multiple cameras placed inside the robot workcell. Each camera will automatically identify moving humans within the field of view of the camera. A pyramid is then constructed using the world position of the camera and the detected human(s). A convex polyhedron will be constructed from the intersection of the pyramids given by the different cameras. This polyhedron defines the space occupied by the human. This system can be used for safety applications, robot-human collision avoidance and surveillance. Furthermore, the system will allow advancement in robot-human collaborations.

The system requires the following assumptions:

1. The robot has a distinct colour from other objects in the area.

2. The human is not wearing a colour similar the robot's colour.

The first assumption is usually the situation or can be satisfied by painting the robot. The second assumption applies to all vision-based tracking systems.

This thesis includes seven chapters. In chapter 2 literature related to this research are reviewed. Related topics in the area include human motion understanding, vision systems, robot-human collision avoidance and robot safety. Detailed design of the vision system for human tracking in a robot workcell is presented in chapter 3. This will include choosing the cameras, placement of the cameras, the lenses, the colour format, and camera calibration process. Chapter 4 explains the process of building the background model used to find the human(s) in the scene. The background subtraction method is employed for detecting the human(s). A novel background variance algorithm is used to classify pixels that belong to the human(s) in the scene. These pixels are grouped using active contours. These active contours are grouped in Chapter 5 to construct a 3D polyhedron that encloses the human(s). The contours are grouped using either minimum bounding rectangle or convex hulls method. Next, half spaces are constructed to surround the groups of contours. A convex polyhedron is then constructed from the intersection of half spaces given by the multiple cameras. Occlusion handling methods are utilized to find the correct polyhedron. Each polyhedron will enclose the space occupied by each human. The system and algorithms are set up and tested in the laboratory. Experimental results of locating and tracking a person in the workcell are presented in chapter 6.

Finally, conclusions, limitations and suggestions for future work are described in chapter 7.

2. Literature review

In this chapter the research literature related to the field of human tracking in industrial environments for safety applications is reviewed. Related topics in the area include human motion understanding, vision systems, robot-human collision avoidance and robot safety.

Understanding human motion using vision systems has been an active research area for the last decade due to the increase in computer processing power. The applications are generally in the areas of human-machine interactions, surveillance, entertainment, and video analysis. The aim of this research field is to: locate, track, determine the human's pose and recognize actions performed by humans in a cluttered environment. In addition, this has to be achieved in real time and without the use of markers. This is essentially having a machine capable of performing, or out performing, similar tasks achieved by the human eye and brain. Currently, to my knowledge, there is no existing machine or system with such capabilities.

The structure of this research can be divided into four sub-problems to be solved: *initialization, tracking, pose estimation, and recognition.* Answering these sub-problems will lead to a human tracking system for industrial environments. Initialization encompasses processing the scene and modeling the subject. Next, tracking involves locating and determining the area occupied by the subject. After that, the pose of the subject need to be estimated by matching the model to the subject. Next, actions performed by the subject may need to be recognized by the system. Finally, for the robot-human collision avoidance system the proximity between the robot and the human can be utilized in controlling the actions performed by the robot.

2.1 Assumptions Used in Prior Publications

Papers published in the field of human tracking have contributed to solving the sub-problems previously mentioned. To tackle these sub-problems separately and make systems manageable, assumptions have been made. Moeslund and Granum [6] divided these assumptions into two classes as shown in Table 2.1. These assumptions are in the order of their frequency in the human motion literature. The first set of assumptions is

Table 2.1: The typical assumptions made by motion capture systems [6]

Assumptions related to movements	Assumptions related to appearance
1. The subject remains inside the workspace	Environment
2. None or constant camera motion	1. Constant lighting
3. Only one person in the workspace at the time	2. Static background
	3. Uniform background
4. The subject faces the camera at all time	4. Known camera parameters
5. Movements parallel to the camera-plane	5. Special hardware
6. No occlusions	Subject
7. Slow and continuous movements	1. Known start pose
8. Only move one or few limbs	2. Known subject
9. The motion pattern of the subject is known	3. Markers placed on the subject
10. Subject moves on a flat plane	4. Specially coloured clothes

based on the subjects' movements and the second set of assumptions is based on the environment or the subjects' appearance.

The first assumption related to movements is obvious for tracking. The subject must be seen by the camera to be tracked. The second assumption is based on the camera calibration. The camera's location needs to be known beforehand to relate images of scene to the camera's location. These two assumptions are used in most systems. The third assumption is used to avoid the challenge of tracking multiple people and concentrate on tracking a single person. The fourth assumption is usually exploited for mobile robots to track the person's body pose or recognize facial expressions. The next assumptions are based on the camera's lack of viewing information about the third dimension (e.g. depth); this is due to the projection of 3D scenes onto 2D images. The sixth assumption is based on the subject being occluded. Information about the subject cannot be acquired, if the subject cannot be seen. The next assumption is based on the speed of system, fast movements of the subject will cause a slow system to lose track of the person. The eighth and ninth assumptions allow the system to track the subject's trajectory and reduce the solution space. The final assumption allows for the projection of the object onto the ground plane.

The first environment assumption requires the system to be placed indoors. The second and third assumption simplifies the segmentation of the subject. The next assumption is based on knowing the parameters of the cameras (e.g. camera location in the world) for obtaining absolute measures of the subject. The fifth assumption refers to the use of special hardware such as infrared cameras to locate the person.

The first subject assumption is used to simplify the initialization problem. The second assumption is required to fit a model to the subject. The next assumption simplifies locating the subject in the scene. The final two assumptions reduce the problem of image segmentation by making the subject easier to be detected.

Assumptions are typically chosen based on the intended application. These assumptions allow the research to concentrate on certain aspects of the human motion capture problem. A system's robustness to the human capture problem can be measured based on the number of assumptions used. The fewer the assumptions used, the more robust is the system.

2.2 Initialization

Initialization is a pre-processing step executed by the system to interpret the scene. It is usually simplified by relying on the assumptions. The initialization process is usually performed offline to prepare the system for tracking. For instance, camera parameters and location can be found offline by calibrating the camera. A passive offline calibration method is introduced by Tsai [7]. This method relies on relating predefined 3D world coordinate frame locations to their image location. This is utilized to calculate the intrinsic camera parameters (e.g. radial distortion, focal length and image principal point) and extrinsic camera parameters (e.g. camera world location, rotational and translational matrices). Technical information about camera calibration is presented in section 3.6.

2.2.1 Background subtraction

Another process that can be performed offline during initialization is background learning. *Background learning* is a pre-processing step to *background subtraction*. It relies on taking a video of an empty scene to build a background model. This background model is used to segment objects in a process called Background subtraction. *Background subtraction* relies on using the learnt background model of the scene to segment foreground objects by subtracting the current image from the background image. This method is illustrated for a human inside a robotic workcell in Figure 2.1. It is usually applied in the initial steps to segment the object of interest and reduce the search area for the object. Therefore, it is a crucial process for tracking and pose estimation. A similar method to *background subtraction* method is evident in weather forecasting television stations in a technique called *chroma-keying*. This is when a person stands in front of a mono-colour screen, usually a green screen, wearing non-green clothing. The person is then detected by thresholding non-green pixels. Another approach for background subtraction is using infrared cameras to segment (through thresholding) warm objects representing the person (foreground) from the cold objects representing the background.

Background subtraction has been implemented for human tracking by several research groups [8] [9] [10] [11] [12] [13] [14] [15]. Statistical approaches have been developed to extract the subject from the background. These approaches rely on the difference in colour, luminance, and edges of the subject to be extracted.

A particularly interesting background subtraction method was presented in [16]. This method used a model of background variation that is a bimodal distribution constructed from statistics of background values during a training period. It was

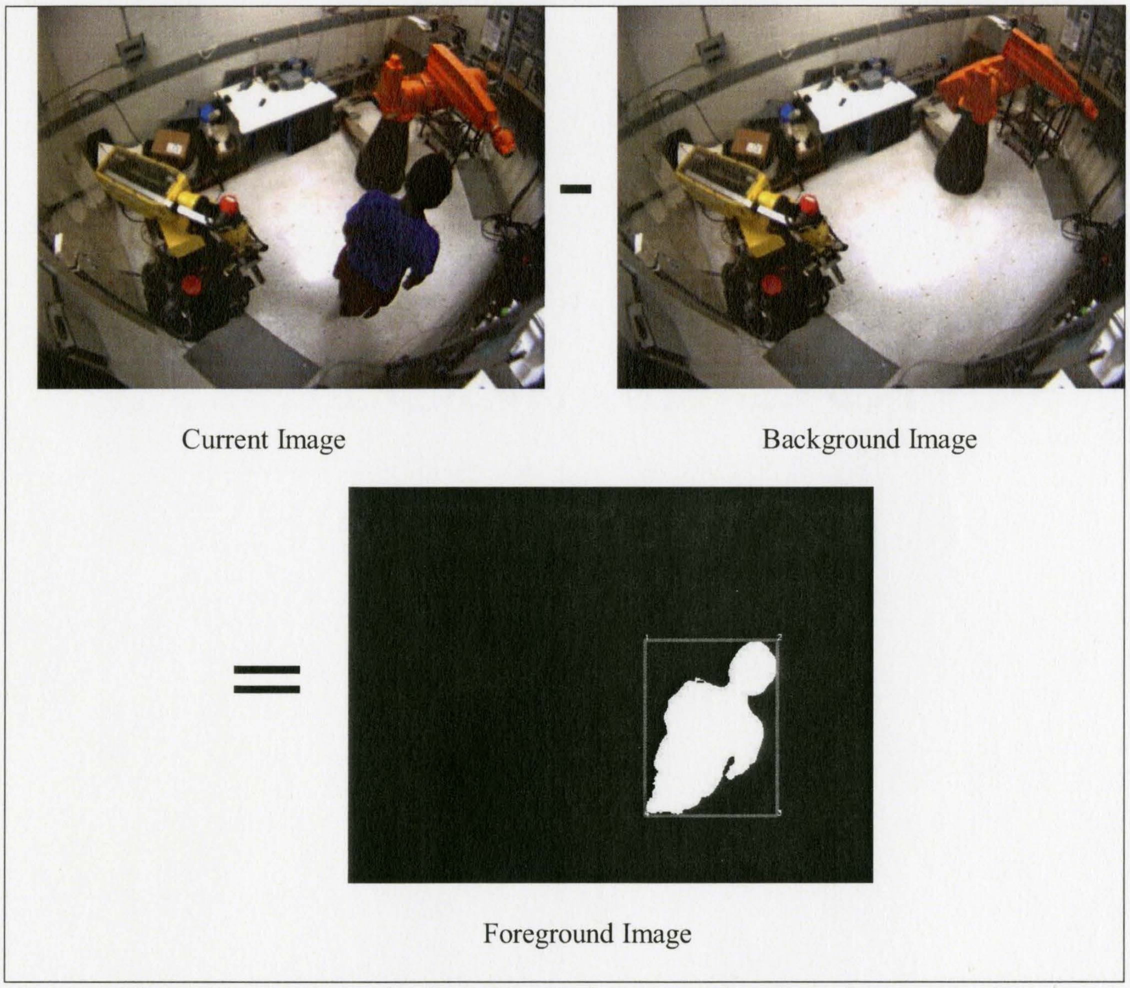


Figure 2.1: Background subtraction of a human standing in a robotic workcell

implemented using a single grey-scale camera operating with a resolution of 320x240 at a frame rate of 28Hz. The first processing stage consisted of applying a pixel wise median filter for several second (20-40 seconds) to distinguish moving objects (foreground) from stationary pixels (background). In this stage, the system learned the minimum and maximum value of each pixel as well as their standard deviations. In the second stage, it used an object based update method to update the background model and adapt to physical changes in the background. During tracking, the system dynamically constructed a change map to determine whether a pixel-based or an object-based update method best applied. The foreground objects were then segmented from the scene in each frame of the video sequence. Due to lighting and changes in the environment, noise was introduced in the foreground image. The foreground image was then processed using thresholding, morphological filtering and object detection. Foreground pixels were grouped together in a contour to be tracked using a Kalman filter and a kinematics model of human bodies. The method was used for real time surveillance and tracking multiple people in an outdoor environment.

A problem associated with background subtraction is that details of the subject might be missing due to having the subject wearing a colour similar to the background. This is a problem inherent in every vision system since the subject is camouflaged.

2.2.2 Shadows and ghosts

Another problem that accompanies background subtraction is shadows and ghosts. Shadows cast by a light source change the properties of the scene and therefore appear as foreground objects in the background subtraction process. To illustrate the problem of shadows Figure 2.2 shows an overhead photograph taken by George Steinmetz of camels crossing the desert. The shadows in the photograph appear similar to the camel. The vision system will have to distinguish between the shadow and the actual camel. To reduce the appearance of shadows in the scene, a second light source must be applied on the other side of the first light source. But this second applied light source will cast an additional shadow of the person, which is the initial problem that we are trying to solve. This brings us to a recursive loop of adding light sources till the point of having an extremely bright scene that saturates the cameras. Therefore, a light source is the cause and solution to shadows.

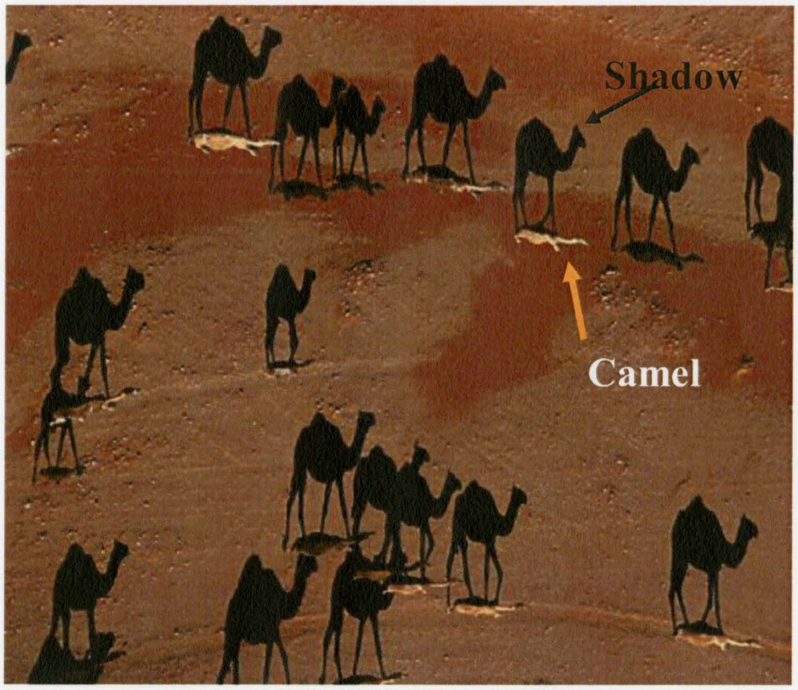


Figure 2.2: Shadows in an image [17]

Removing shadows using image processing techniques is also difficult. What distinguishes a shadow in the image? Dark regions are not always shadows, since a foreground object can be dark. Shadows cast onto the foreground object should be segmented as foreground objects. The assumption that shadows are cast onto the ground plane is not valid for general scenes.

To solve the problem of shadows, Joshi et al. [15] used four different cues for foreground/shadow detection. The method employed background subtraction to segment foreground objects along with shadows. In the second step, the method computed the following cues: edge magnitude, edge gradient error, colour and intensity. The first two cues were used due to shadows lack of significantly modifying the edge gradient direction of any pixel. The second measure relied on shadows only changing the intensity of the object. In the next step, the system used blob level reasoning as a step towards further classifying regions. This method was not able to detect objects with reflective surfaces such as a car's windshield. Another problem associated with the method was the blob level reasoning algorithm. It made mistakes in classifying objects and some parts of the objects were cut-off.

In [13], a different shadow detection method was introduced. The colour space was transformed from Red-Green-Blue (RGB) to Hue-Saturation-Value (HSV). The reason is that the HSV colour space explicitly separates colour information from luminosity. Shadows can then be detected by comparing the current image's intensity value to the background image. If the light intensity value has been reduced without change in the colour information, this signals the detection of a shadow, since shadows only change the light intensity of the scene but not the colour. This method was able to detect static shadows and moving shadows. However, shadows created near the subject were incorrectly classified as foreground objects, thus substantially affecting the subject's shape.

Another problem that arises is the appearance of "ghosts". Ghosts are defined as

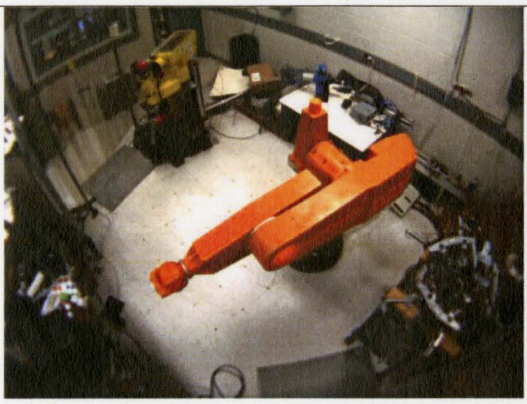
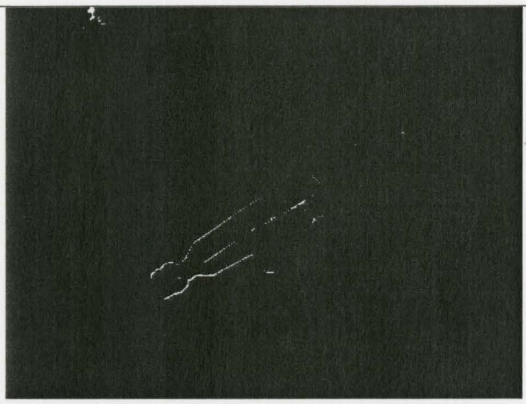
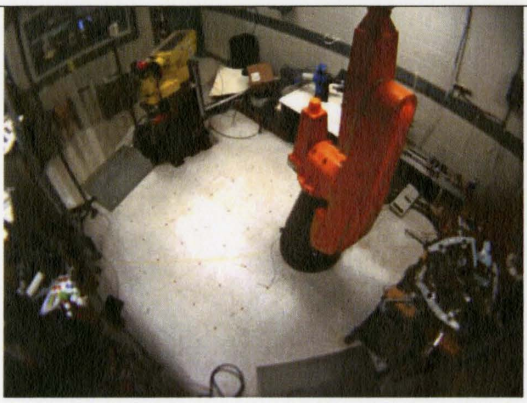
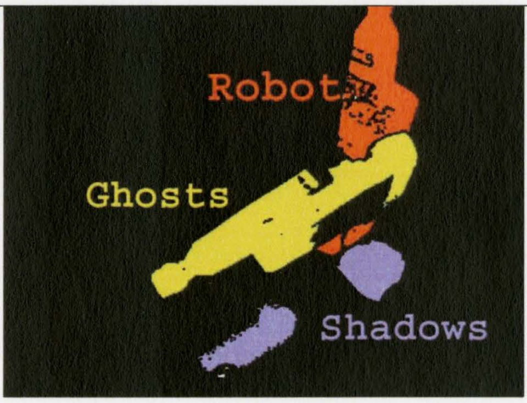
	
a) Background Model	b) Foreground Image
	
c) Current Image	d) Ghosts and Shadows in the Foreground Image

Figure 2.3: Image of Shadows and Ghosts.

objects in the scene that have been recorded in the background learning phase but were removed from the scene. These objects will be reported as foreground objects but do not correspond to any real object. In other words the system detects a false positive. Figure 2.3 (a, b) show an image of the background model and the foreground image, respectively. In Figure 2.3c the robot moves to another location. Figure 2.3d shows the detected foreground robot along with the detected shadows and ghosts. To solve the problem of ghosts, the background image needs to be updated. Adaptive algorithms have been introduced by several research groups [13], [16], [18]. These algorithms update static foreground objects that remain stationary for some period of time. The challenge comes in defining this time period properly.

2.3 Tracking

Tracking is defined as finding and locating the object in the scene on each image frame supplied by the camera. This is the second step after initialization. Tracking starts by representing the information of the object using methods such as blobs and active contours. Blobs model the object by combining pixels with similar colour and spatial properties to a number of blobs. Active contours were introduced in [19]. This method defines the object by finding the edge segments and fitting closed curves to them. The deformations of the contour(s) are controlled by external and internal energy functions. The former fits curves to the image edges, while the latter adjusts to the smoothness of the curve.

2.3.1 Markers and Skin colour

Tracking can also be achieved by using markers. These markers shine brightly when are viewed using a camera. In [20] a person was segmented and tracked easily in the image by thresholding and then fitting the person's kinematics model to the markers. Markers are usually applied in human-robot interactions and in the entertainment industry. Although a powerful method due to its robustness to the ambiguity between the humans' limbs, it requires people to wear markers. Another system that combined cameras and multiple passive infrared sensors was proposed in [14]. This system covered a wide area by using wireless sensor network. The system utilized a background subtraction method to detect the person in the image.

Skin colour can be used instead of markers for tracking. In [9] a tracking method that depends on the human's skin colour has been tested. The system used a statistical model of colour and shape to obtain 2D representations of the head and hands. A limitation of the system is that it expects one person to be in the scene. Multiple people in the scene cause a problem in the gesture recognition algorithm.

In [21], a method is introduced that improves skin colour segmentation by finding the face using global skin colour. It then updated the skin colour based on an area around the face to withstand changes in lighting conditions. This approach achieved a frame rate of 3Hz due to the use of face detection algorithm. This method is very slow for real time applications such as robotics.

Another method introduced in [22] also used skin colour to track head and hands for a humanoid robot system. The system combined edge and region cues in a likelihood function for human motion capture. It then tracked the human movements using particle filters. To avoid the problem associated with ambiguity of tracking multiple people in the scene, the system required the person to have a solid coloured shirt. The system achieved a speed of 15 Hz using two cameras running at 25 Hz and an image size of 320 x 240 pixels.

Using skin colour as a cue to find a human can be problematic, since self-occlusion of the limbs (*e.g.* a hand is hidden behind the torso) requires the system to do extensive calculations to be robust in finding the limbs' locations. Another problem associated with skin colour segmentation is the variety of the human skin colours. Objects in the scene that have a colour close to the human skin colour will also be segmented. A further problem is that workers may be wearing protective headgear and clothing that covers their skins (e.g. during welding).

2.3.2 Multiple cameras

One of the problems associated with tracking is occlusions. A solution to the occlusion problem is to have multiple cameras installed. This will give the system multiple views of the object as well as depth information, at the cost of more computations.

A system for human tracking using multiple cameras was proposed in [10] . The system measured human body position in a world coordinates using a model based template. It was then able to track moving people in an experimental environment at speed of 15Hz using three vision cameras. A state transition map was then used to pass information between the cameras for continuous tracking. This method is dependent on the environment. It requires prior information about the environment as well as template for all possible positions of a human. It also does not provide 3D information about the human's position.

In [12], another system using multiple cameras to track people in a cluttered environment was described. This system took the midpoints of matched segments from different camera views and projected the objects' locations onto the ground plane. Next it used kernel estimation techniques to calculate the probability for the presence of an object on the 2D ground plane. This system does not provide any information about the tracked object's third dimension.

Tracking can also be achieved using a Kalman filter. A Kalman filter predicts the state of the subject in the following frames based on the subject's estimated current location; and sensor and model uncertainties. These predictions can be used in finding the region-of-interest in subsequent frames. The drawback of using Kalman filter is that human motions are very uncertain. This is because humans undergo shape deformation, move with unpredictability and sudden changes in their main direction, and are likely to be occluded by objects or other people. This results in high uncertainties for robust tracking. Furthermore, tracking a region-of-interest might cause the cameras to be oblivious to new subjects entering the area. Therefore Kalman filters are not well suited to human motion tracking.

The drawback of using multiple cameras is the increase in computations of each camera's image. This leads to slowing down the system. Another problem associated with tracking multiple people using multiple cameras is relating the different people in each camera's view. This is referred to in the literature as the correspondence problem.

2.4 Pose estimation

Pose estimation is the process of recovering information about the location and the configuration of limbs of a human body. This information is usually used for human-computer interface and 3D animation. Pose estimation can range from knowing the space occupied by the subject's body, which could be used for surveillance, to finding precise position, orientation, width, etc. of each limb.

Pose estimation can be achieved by fitting an articulated human model to the subject in the image. The human is modeled as segments connected by joints in a stick-figure. The human flesh is represented in various ways depending on the application. This model represents the structural information of the human skeleton. It is fitted to data obtained from different sensors to recover the subject's pose.

Pose estimation can be achieved using a single camera. In [23] the proposed single camera system concentrated on deriving as much information as possible from the monocular image sequences, and reducing the size of the search space by applying restrictions to possible movements. A single camera is not sufficient to determine accurate 3D information, based on principle of depth triangulation. Furthermore, one third of the degrees of freedom of the human model are nearly unobservable due to motion ambiguities and self occlusion. This system requires training to model the human motion.

A 3D tracking system of human body movements using a 3D body model is proposed in [24]. The system incorporated data from different sensors (e.g. time of flight camera and stereo camera) to build a human model by matching the data set using the iterative closest points algorithm. The subject is modeled as cylinders connected by elastic bands. This system used the face and hands as cues to be tracked. Nevertheless, the system cannot model humans of different heights or the addition/deletion of body parts except after changing the model. It also required the necessary input data information to be able to track the person. Furthermore, this system ran at 10 Hz.

Sigal et al. [25] designed a model of the body as a collection of loosely-connected limbs. Motion capture training data was employed to relate the 3D pose of the connected limbs. Their system then solved the human pose problem with a non-parametric belief propagation using a variation of particle filtering. It also utilized limb and head detectors to provide information for the model. This system required training data and used detectors to track people. The authors then mention a system by Duetscher et al. [26] that has outperformed their system.

Ogawara et al. [27] used a human model with 29 degrees-of-freedom. Their method relied on finding the skin model of the person using multiple cameras. It then fitted a deformable articulated model to the skin model. Their method used a Kd-tree search algorithm to track the motion and then used hierarchical estimation to enable accurate estimation. However, the subject tracked is in a controlled environment. Also, the human model is not dynamic for people with different body shapes. This system was tested on a stored sequence of images. Furthermore, this method suffers from initial pose estimation.

One of the major difficulties in recovering a pose from images is the high number of degrees-of-freedom (DOF) in movement that is to be recovered. Any rigid object requires 6 DOF to fully describe its pose. A human body contains more than 29 DOFs. This makes these systems too complex and slow for most real time applications. In addition, the human body is very diverse, so the system must be able to scale to people with different heights or even the absence of body parts. Human model methods also face a problem with occlusion of body parts due to loosely fitting clothing, e.g. a person wearing a dress.

2.4.1 Visual Hulls

Visual hull construction is an approach of building a 3D object that encompasses the subject from a set of multiple views. This constructed object consists of flat faces and straight edges and is called a polyhedron. The algorithm to construct visual halls was first introduced by Laurentini in [28]. It gained interest in recent research due to its simplicity. It requires finding the foreground representation of an object in an image. Then building a 3D cone of the foreground object, where the camera's focal point position is the apex of the cone. The intersection of the different cones from multiple views constructs a polyhedron that encompasses the person.

Using 17 cameras centered on a 1m x 1m x 2m scene, Meozzi [8] described a system to construct visual hulls offline. Foreground objects were segmented from a binary image using background subtraction techniques. A visual hull of the subject was then constructed from the intersection of the multiple cameras.

Borovikov [11] developed a distributed system for constructing volumetric image. The system utilized 14 cameras to cover a 24'x24'x'10' room. For foreground segmentation, the system utilized the method developed by Haritauglu et al. [16]. A volume representation of the scene was constructed using the intersection of the visual hulls. The system then used an algorithm to render the visual hull based on pre-computed boxes of the volume and storing the boxes in lookup tables. The system was able to achieve a speed of 10Hz using specialized hardware.

An approach to find the human pose from visual hulls is described in [29]. This system used a support vector machine for describing the shape of the 3D subject. It utilized 12 trained poses to estimate the person's pose in the constructed visual hull. The global shape descriptors utilized were not affected by rotation, scaling or translation. However, using support vector machine is dependent on the selected trained poses. The system has been tested on a stored video sequence and does not have a solution to multiple people in the area.

Visual hull has several interesting properties. First, it constructs a polyhedron that crudely represents the subject's pose. Second, although it is only an approximation of the subject's pose, it is guaranteed to enclose the subject. This is important for tracking human(s) in a robot workcell. Third, the size of the visual hull decreases with the number of cameras used. Finally, its accuracy is dependent on the foreground object obtained from background subtraction method and the number of segments that construct the person. It is a fast method for finding general information about the subject which makes it application dependant.

2.5 Recognition

Recognition is based on identifying the type of action performed by the subject. These actions can be simple such as walking, running, sitting, etc. or complex and performed over extended periods, e.g. dancing. Recognition is usually performed after estimating the human pose using articulated deformable models. It matches the segmented actions of the human to a library of data base of performed actions, to determine to which class does the action belongs to.

In [30] [31], Wang et al. presented an approach for recognizing human actions based on dynamics and setting of human body parts. The approach was based on extracting contours and constructing a mean contour to represent information about the scene. Then a rigid human model composed of 14 body parts was fitted onto the contour. Particle filtering was next used to compute the likelihood of a pose given by the subject.

In an attempt to understand human actions, Robertson and Reid [32] have designed a hierarchical system that is based on Hidden Markov Models. Human actions were modeled as stochastic sequence of actions. These actions were described by features such as position and velocity. The system was tested by giving commentary of actions preformed by tennis players; the system was able to output information such as walking - left-to-right - at the backcourt. In [33], Hu et al. recognized the posture of a person using frequency and phase information from 2D images. The frequency information is then compared against a data base to estimate the person's pose.

A problem associated with recognition is the transition actions. Transition actions are actions that connect two main actions. For instance, if a system was trained to recognize actions such as lying down and walking, the system needs to recognize actions in the following order: lying down, getting up, sitting down, standing, and finally walking. All the actions between lying down and walking are transition actions that are going to be misclassified by the system. An attempt to solve this problem is introduced in [34]. In this system the person's pose is estimated by a set of features, encoding the angular relationships between body parts. It then used a Hidden Markov Model for spotting and recognizing actions. Transition actions can be reached from any other action state. This is basically specifying any uncertainly recognized action as a transition action.

Actions preformed by the human are hard to be recognized due to the dynamic human body. Another problem associated with recognizing actions is that it depends on the accuracy of the trained data base.

2.6 Vision based robot human collision avoidance

Few attempts have been made towards robot human collision avoidance using vision systems. Ebert and Henrich [35] presented an approach to directly conclude from the scene's images the state of the entire configuration space. The idea was to construct an obstacle image for each camera view. This was achieved by comparing the current scene images to the current robot images. Intersection images were then constructed by intersecting the obstacle images with the images containing a trained configuration of the robot. The intersection images reflect possible collisions and if the number exceeds a certain threshold, the tested configuration was considered occupied. The system was able to decide whether a requested configuration of the robot was free or occupied by an obstacle. However, the system sometimes failed by reporting a configuration was free when there was an object. The system also required images of all possible configuration of the robot model.

Another system is described in [36]. Their method tracked a person in an industrial environment by tracking the worker's hard hat using a camera system. This information was then combined with information given by multiple passive infrared sensors in a data fusion algorithm. A concept called "protective cell" was introduced. This cell represents a cell that is between the subject and the robot created to protect the subject given sensor uncertainty. The system was able to report the human's location within an occupancy grid in a probabilistic manner and protect the subject in the presence of uncertainties using protective cells. However, the system achieved a speed of only

1.7Hz, which is too slow for real time applications. Furthermore, the camera system required the subject to wear a hard hat.

In [37], Kuhn and Henrich presented a method of detecting the distance between known objects and unknown objects. The known objects are expanded in 3D and projected onto the camera image. The camera is placed to view the objects from overhead. This system then determined the minimum distance between multiple known and unknown objects by expanding the known objects till the distance to the unknown objects was found. However, the system required a model for the known objects. This method can be used for surveillance and safety applications.

Pilz Automation Safety Inc. has recently begun selling a safety system named SafetyEYE™ [38]. SafetyEYE™ is a system that uses three colour cameras arranged in a triangle and placed over the robot. This system is intended used, instead of fences, to avoid robot human collisions. Unfortunately, technical information about the system is not available from the source.

2.7 Conclusion

Robot human collision avoidance has is an important research topic due to the injuries associated with the increasing in the number of industrial robots. The research has concentrated on using vision systems to solve this problem. Vision systems provide a lot of information about the scene but also have their drawbacks. Many research methods have used background subtraction as an initialization phase. Background subtraction has robustness to noise and self-occlusions, making it attractive for estimating the human model in an industrial environment. Furthermore, the semi-static nature of the industrial environment allows background learning to achieve reasonable results. Background subtraction's accuracy is dependent on a combination of factors including: the background model, classification method and model updating method. There is a trade-off between the system's speed and accuracy. For collision avoidance safety applications, the system's speed is considered to be of great importance. One of the problems associated with background subtraction is the appearance of shadows and ghosts. An incorrect classification could result in human injury. Therefore, the image processing techniques need to distinguish between shadows, ghosts and the subject.

Tracking subjects in an image is the step of finding the location of the human(s). Research methods have used various techniques to locate the human(s) in the image. Some of the recent techniques have used markers. Markers' robustness to the pose of the human is attractive to applications where markers can be placed on the subject. Unfortunately for industrial applications, making the subjects wear markers is not a feasible option. Tracking can also be done using skin colour. This is also not a feasible option due to the large skin colour variations and objects in the scene with a colour similar to the skin colour. Furthermore, occlusions of the skin colour will be problematic in some applications. To solve these problems, multiple cameras can be used to track people from different views. This arrangement will help in finding the person in the case of an occluded camera. On the other hand, more cameras will increase the complexity and will slow down the system.

To represent the human, the system needs to estimate the pose of the subject. Estimation methods differ by giving various accuracies. Some research methods have relied on using articulated human models. These human models are fitted to subjects in the scene. There are several problems associated with this method. First, the model needs to be able to fit on people with different sizes. Secondly, the person needs to be wearing tight clothing to be able to locate their limps. Third, it is too complex and slow for real time applications. Therefore, it is not suitable for real time collision avoidance. Other estimation methods rely on building visual hulls. Visual hulls can be found quickly but give only a crude geometric representation that encapsulates the subject.

For safety applications, it's sufficient to know the space occupied by the person but not the action. Therefore, visual hulls are a suitable approach. Knowing the space will provide us with enough knowledge about the proximity of the person to the robot. Recognizing the gestures is an unnecessary step.

In this thesis, a markerless human tracking system will be established using multiple cameras. The system will utilize background subtraction as an initialization step. Foreground segmentation algorithms will be used to discard the background/shadows/ghosts from the scene. Visual hull(s) of the human(s) will be constructed from the multiple camera views. These hull(s) will provide information to avoid robot-human collisions for systems such as the one proposed in [39].

3. Vision System Design

This chapter will describe the design of the vision system for human tracking in a robot workcell at the Robotics and Manufacturing Automation Laboratory (RMAL) at McMaster University. This workcell is similar to those found in industrial environments. The design of the vision system includes choosing the cameras, placement of the cameras, the lenses, the colour format, and camera calibration process.

3.1 Workcell

In manufacturing environments, a workcell is an arrangement of resources to improve the quality, speed and cost of the manufacturing process. Workcells are designed to improve these aspects by improving the process flow and eliminating waste. The workcell in the RMAL has one active orange robot PUMA 762 as shown in Figure 3.1. The second yellow Fanuc robot is not currently active. Some of the space inside the workcell is occupied by tables and other objects. The vision system will be designed to monitor the area around the robot.

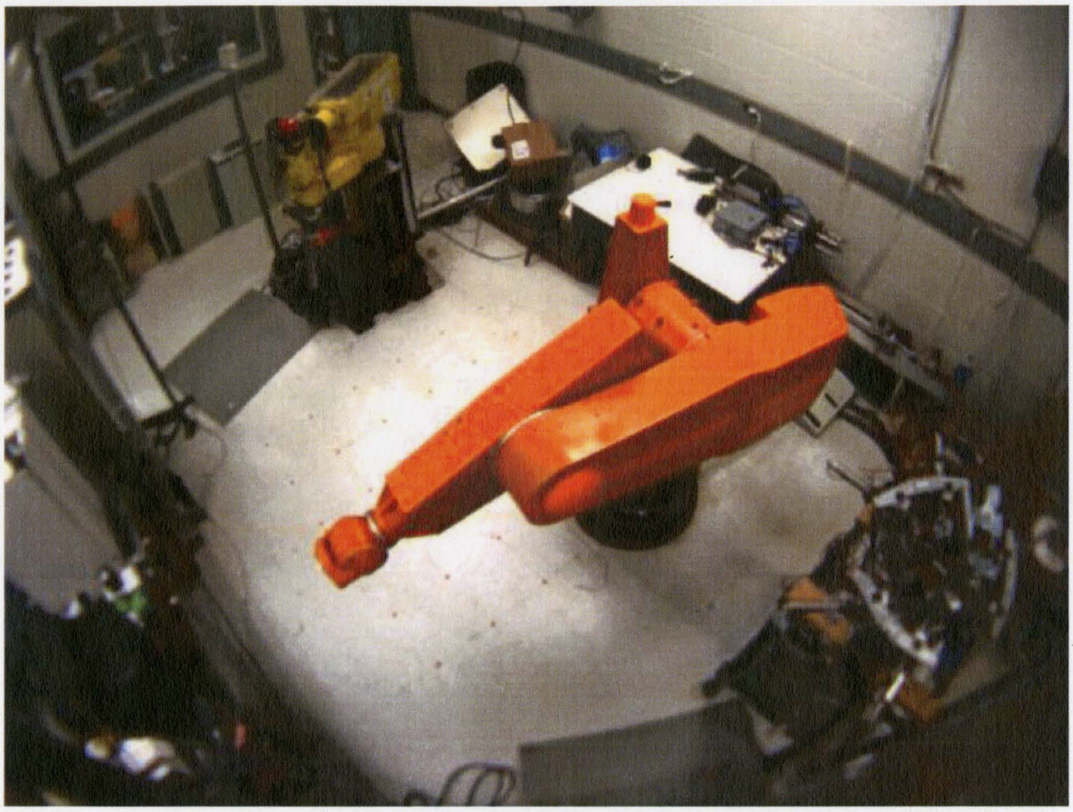


Figure 3.1: The RMAL workcell

The PUMA 762 robot (Figure 3.2) is a six axis articulated robot, capable of carrying a 20 kg payload. This gives the robot a configuration similar to an enlarged human arm. The robot can be used for arc welding, sealant dispensing, material handling, machine loading, inspection, testing, joining and assembly. The workspace of the robot is a spherical workspace as shown in Figure 3.3.

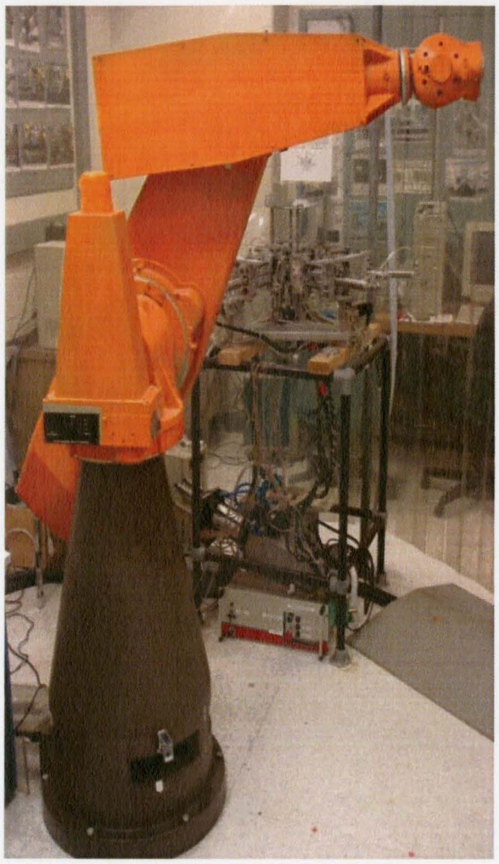
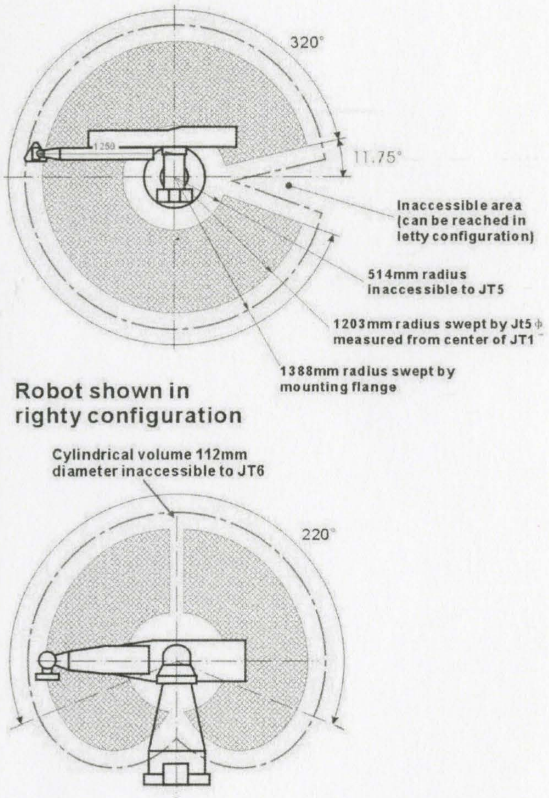


Figure 3.2: Puma 762 Robot


320°
1250
11.75°
Inaccessible area (can be reached in letty configuration)
514mm radius inaccessible to JT5
1203mm radius swept by Jt5 ϕ measured from center of JT1
1388mm radius swept by mounting flange
Robot shown in righty configuration
Cylindrical volume 112mm diameter inaccessible to JT6
220°
Figure 3.3: The workspace of a PUMA 762 robot [40]

3.2 Camera placement

Reconstruction of a scene requires multiple images from multiple cameras. This is because a single image is a 2D projection of the 3D world, for which depth information is lost. The 3D location of every point in the scene can be determined uniquely when multiple cameras are available from different viewpoints.

In this thesis, five different camera locations have been selected. These locations are shown from the top view in Figure 3.4 and from the perspective view in Figure 3.5. Four of the camera locations (C1, C2, C3 and C4) are on the same plane at height of 2.6m above the ground. The fifth camera location (C5) is attached to the ceiling at a height of 2.9 m to give an overhead perspective view.

The first series of experiments, as described in section 6.3, will use the camera locations C1-C4. This gives the camera system a full view of objects around the robot. The second series of experiments in section 6.4 will use the camera locations C1, C2, C3 and C5.

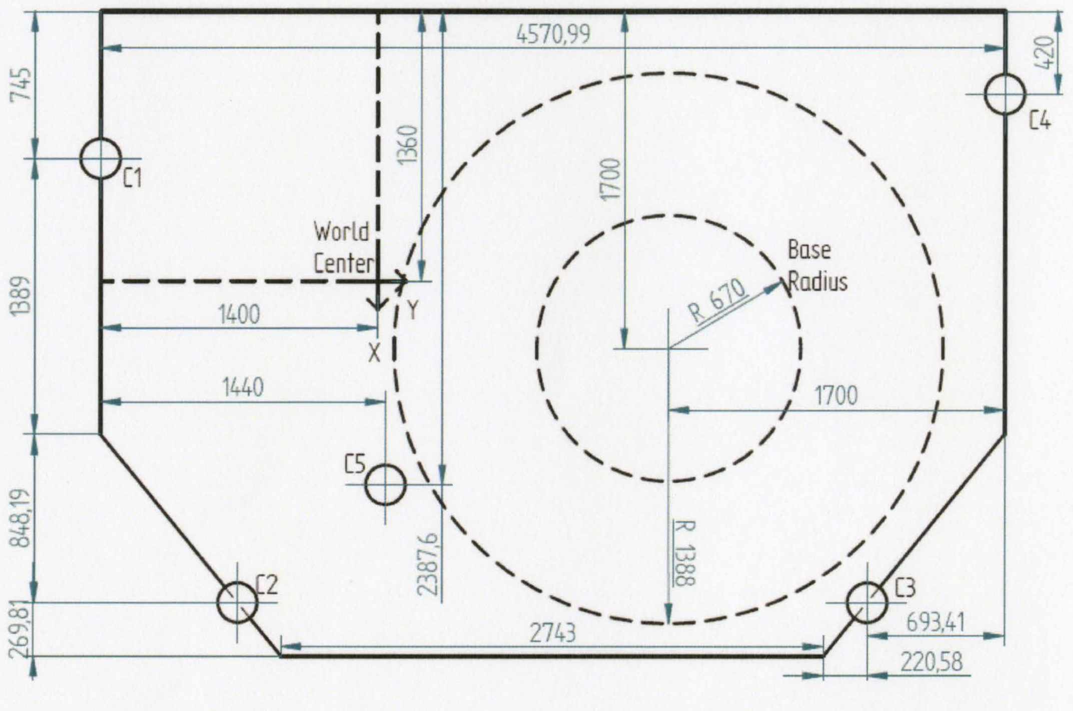


Figure 3.4: Top view of the camera locations. The dimentions are in mm.

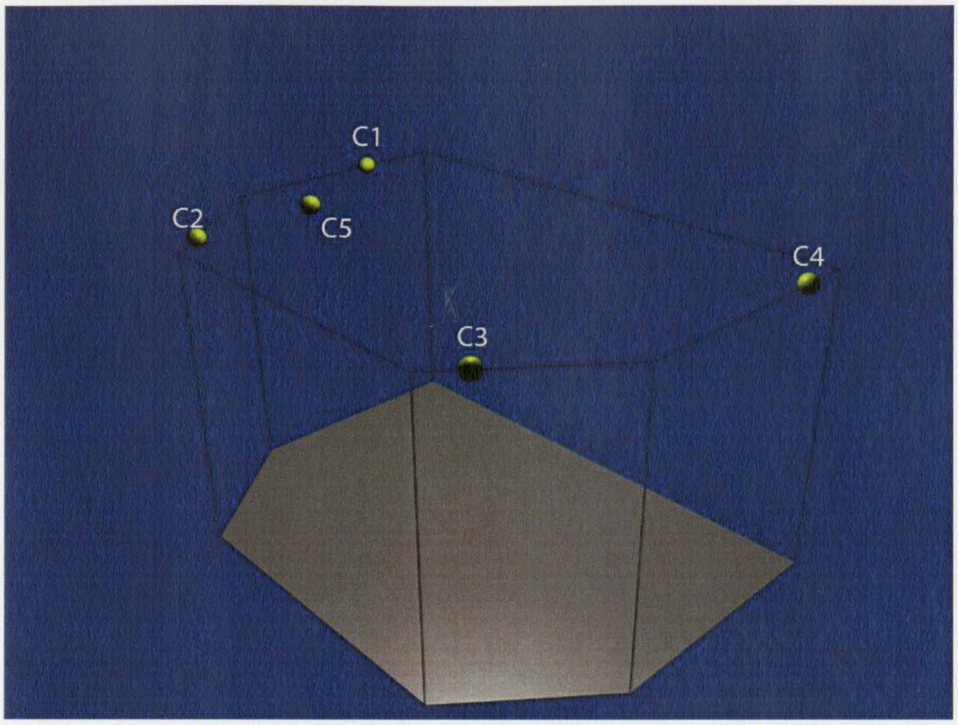


Figure 3.5: Presepective view of the camera locations

3.3 Camera System

Choosing the correct camera for an application is the first step for a vision system. The integration of components such as the sensor technology, the image output form and the connection type, will result in an optimum system performance. Here are some of the parameters that need to be considered for a vision system.

3.3.1 CCD vs. CMOS

Charge Coupled Device (CCD) and Complementary Metal Oxide Semiconductor (CMOS) are two different sensor technologies for converting light into electronic signals. In a CCD sensor, the charge of each pixel is converted to a voltage, buffered, and transferred through a single node as an analog signal. In a CMOS sensor, the charge-to-voltage conversion is done at the pixel level, yielding a less uniform output. The non-uniformity is evident in low light environments, such as applications at high speed or high magnifications [41]. Table 3.1 shows a comparison between CCD and CMOS technologies. CMOS Sensors could be incorporated into applications requiring low power consumption or into space-constrained applications, whereas CCD sensors are recommended for applications requiring superior image quality.

Table 3.1: A comparison between CCD and CMOS technologies [41]

	CCD	**CMOS**
Pixel Signal	Electron Packet	Voltage
Chip Signal	Analog	Digital
Fill Factor	High	Moderate-High
Responsivity	Moderate	Moderate-High
Noise Level	Low	Moderate-High
Dynamic Range	High	Moderate
Uniform	High	Low
Resolution	Low-High	Low-High
Speed	Moderate-High	High
Power Consumption	Low	Moderate-High
Complexity	Low	Moderate
Cost	Moderate	Moderate

3.3.2 Analog vs. Digital

The CCD silicon chip is an analog component. This means that the pixel values are collected by means of sampling (interlaced or progressive). The signal processor converts this information into an analog signal, which can be transferred to a monitor. In digital cameras, digitizing occurs as the signal is collected from the chip. Once digitized, processing and image enhancements can be done with little loss to the signal. Table 3.2 shows a compression between digital and analog cameras.

Table 3.2: A comparison between digital and analog Cameras [42]

Digital	**Analog**
• Typically large cameras. • Vertical resolution is not limited, so digital cameras can offer higher resolution. • With no bandwidth limit, these can offer higher numbers of pixels and larger CCD sensors, resulting in greater resolution. • Computer and capture board required to display signal. • Signal can be compressed so user can transmit in lower bandwidth with no loss. • Usually have square pixels. • The output signal is digital, therefore little signal loss occurs during signal processing.	• Size is typically smaller. • Vertical resolution is limited by the bandwidth of the analog signal. • Sensors are usually standard size formats. • Computers/capture boards can be used for digitizing but are not necessary for display. • Analog printing and recording can be easily incorporated into the system. • Usually have rectangular pixels • Analog signals are susceptible to noise and interference which cause signal loss.

3.3.3 Interlaced vs. Progressive

In interlaced scan, each full image frame is split up into two sets made up of odd and even lines. A picture needs to be scanned twice to produce a full frame. This method of capturing image produces blurred images of fast moving objects. This is because by the time the camera scans the second set of pixels, the object would have moved causing the lines to be out of order, and therefore blurred.

However, a better way to deliver video is to display each frame as a whole. This avoids the frame mismatch that can occur with an interlaced scan. This method of video delivery is called progressive. Progressive scan captures an image in a single exposure. It holds the image till it is fully scanned. This method is ideal for high speed applications but it is more expensive.

3.3.4 FireWire vs. USB connection

Universal Serial Bus (USB 2.0) is a serial interface standard for devices. This interface allows connecting and disconnecting devices without having to reboot the computer. It also provides power to low-consumption devices and operates at a speed of 480Mbits/second.

IEEE 1394 interface (also known as FireWire) is also a serial interface standard for devices. It has similar capabilities to the USB but with a speed of 400Mbits/second. It mostly used with video and audio devices. This is because USB communicates using software which requires a large overhead. This reduces the actual speed of communication to around 240Mbits/second. USB was intended for low cost devices. On the other hand, FireWire communicates using hardware which is fast and more reliable for time sensitive applications such as video streaming.

3.3.5 Colour vs. greyscale cameras

Typically colour cameras use red, green and blue (RGB) pixels in a Bayer filter to output coloured images. The raw Bayer images are converted to RGB using colour interpolation algorithms. This filter degrades the image quality. Greyscale cameras output only greyscale images. They have an increased sensitivity to low-light and have a better signal-to-noise ratio compared to colour cameras. However, for the human tracking application colour cameras will be used since they are better at distinguishing between foreground and background objects.

3.3.6 Chosen cameras

The system hardware consists of four Point Grey Research Dragonfly2 colour cameras equipped with wide angle lenses. Two cameras acquire images of size 640 x 480 pixels at 60 frames per second (FPS). These cameras use a Sony ICX424 1/3" CCD image sensor. The other two cameras acquire images of size 512 x 384 pixels at 60 FPS and they use a Sony ICX204 1/3" CCD image sensor. Dimensional drawing of the camera is shown in Figure 3.6. The first sensor has a pixel size of 7.4 micrometer square pixel and the second sensor has a pixel size of 4.65 micrometer square pixel. The Dragonfly2 features on-camera colour processing and auto white balance. The colour processor can output pixels in various colour formats including YUV411, YUV422 and

RGB. It also allows access to the raw Bayer pattern. This allows reduction in bus bandwidth and results in faster transferring speeds.

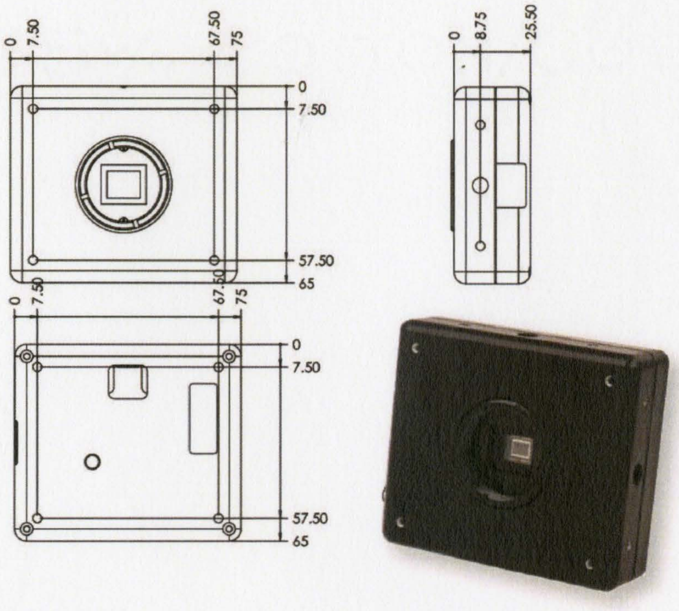


Figure 3.6: Dimensional drawing of a Point Grey Research Dragonfly2 camera [43]

3.4 Lens

An imaging system should create sufficient image quality to allow extraction of desired information about the object from the image. Note that what may be adequate image quality for one application may prove inadequate in another. There are a variety of factors that contribute to the overall image quality, including focal length, working distance, depth of field, field of view, vignetting, and lens distortion.

Focal Length: Rays from infinitely distant objects are condensed internally in the lens at a common point on the image sensor. The point at which the image sensor is positioned is called a focal point. Lenses have two principal points, a primary principal point and a secondary principal point. The distance between the secondary principal point and the focal point on the image sensor determines the focal length f of the lens. A lens with a shorter focal length has a wider field of view. Figure 3.7 shows the pinhole camera model. The distance between image plane and the camera center O is the focal length f. A point p in the 3-D world is projected on the image plane as the point p'. In practice, it is more convenient to use a mathematical equivalent model by moving the image plane to the front side of the camera center as shown in Figure 3.8.

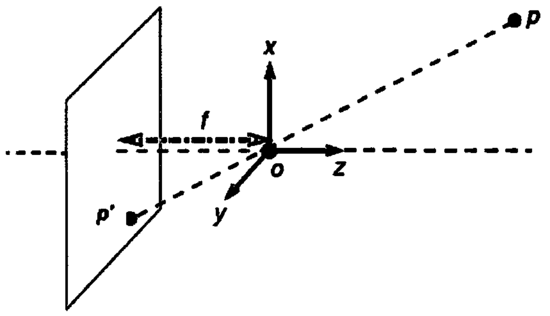


Figure 3.7: Pinhole camera model [44]

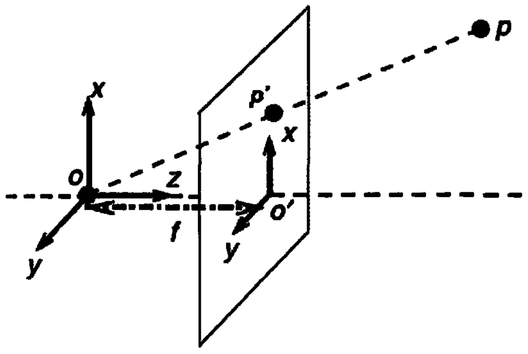


Figure 3.8: Frontal pinhole camera model [44]

Working Distance: The distance from the front of the lens to the object is called the working distance. Each camera lens is designed to work in a certain region. This parameter is important to choose the right lens for the job. In this project, the working distance is defined by the workcell dimensions.

Depth of Field: Defined as the allowable amount of object movement (in and out of best focus) while maintaining a desired amount of focus. It can also be defined as the maximum object depth that can be maintained entirely in focus. Lenses with a smaller aperture give a larger depth of field. However, this permits less light to reach the image sensor which produces darker images.

Field of view: Defines how much the camera can see. Each lens is designed to see a certain limited range called the field of view. This field of view describes how much of an object is detected by the camera system and is dependant of the lens and the sensor size. The angular field of view is the full range of illumination that is collected by the camera system. In general, there is a trade-off between the field of view of a lens and accuracy. Wider angle lenses tend to be less accurate.

Vignetting. It is defined as the darkening in the corners of the image. It is an effect inherent to lenses to some degree. The effect is strongest when the aperture of the lens is wide open. Reducing the aperture size allows less light to pass through to the camera sensor.

Camera distortion: Straight lines in the scene appear curved in the image. This is due to the magnification of the lens. There are two common types of distortion. The first type is barrel distortion. Barrel distortion occurs when using a wide angle lens to project hemispherical scenes onto a flat surface (image sensor). This effect is apparent in fisheye lenses. Lines that do not pass through the center of an image curve outwards. The second type of distortion is pincushion distortion. This effect occurs in telephoto lenses. Lines that are not passing through the center of the image curve inwards towards the center. This effect is less common due to advances in lens design. The two types of distortion can be corrected by warping the distorted image. However, this solution will cause tear areas in images that have to be interpolated. Figure 3.9 shows a barrel distorted image by a fisheye lens. The lines in the chessboard can be observed to curve outwards. Figure 3.10 shows the corrected image by warping. It can be observed that the lines in this corrected image are straight. To fix the tearing problem, the image is processed to view a smaller field of view. This can be observed by noticing the location of the right-top corner in Figure 3.9 and Figure 3.10.

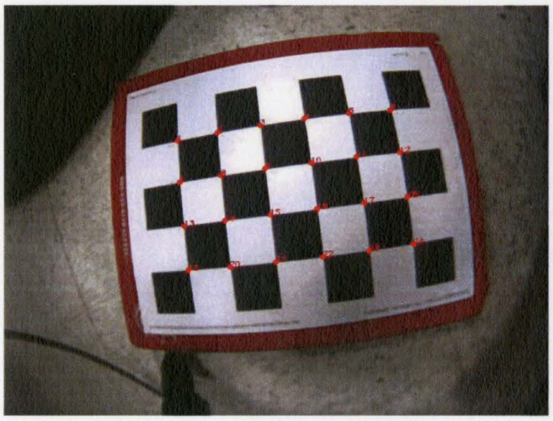


Figure 3.9: Distorted Image

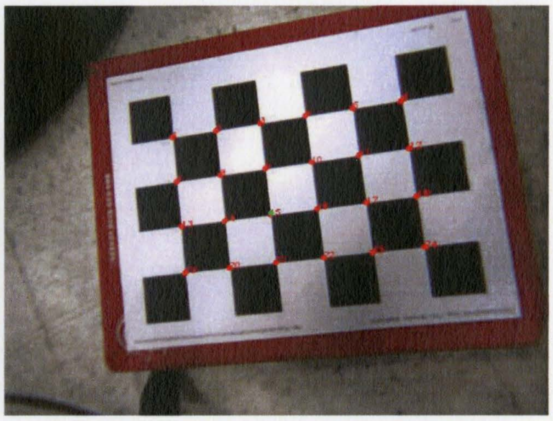


Figure 3.10: Image corrected by warping

Various lenses have been tested for the project. An auto iris vari-focal length lens from Pentax has been selected. The lens model is a C70223HK with a variable focal length of 2.8 - 12mm. This lens was attached to the Dragonfly2 using a CS mount. Given the 1/3" image sensor, the angle of view using this lenses various from 22.9° to 96.7°. This lens has a minimum working distance of 0.3m. Effects such as vignetting were not observable. However, barrel distortion was evident when using a short focal length.

3.5 Colour format

There are two common colour formats used in robotics, RGB and YUV. RGB colour format consists of a combination of the three colours: red, green and blue. Mixing these colours with different intensities gives the colour space shown in Figure 3.11. RGB colour format requires 24 bits to represent the colour of each pixel (8 bits per colour). The RGB colour format is not very stable with regard to alterations in the illumination, because it does not contain separation between the illumination and the colour parts.

The YUV colour format consists of three components, Y which is the luminance (or brightness), U and V which are the chrominance (or colour). Mixing the U and V gives the colour space shown in Figure 3.12. The Y component represents the brightness of the pixel. Having the Y component also helps with identifying shadows as shown in section 2.2.2. This colour format is a better choice for vision systems to be robust against alternations in illumination. The conversion between RGB and YUV is achieved by the following linear transformation:

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.514 & -0.101 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{3.1}$$

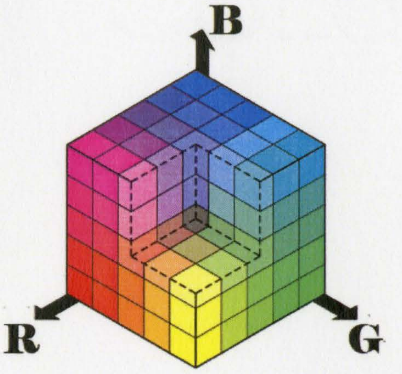


Figure 3.11: RGB colour space [45]

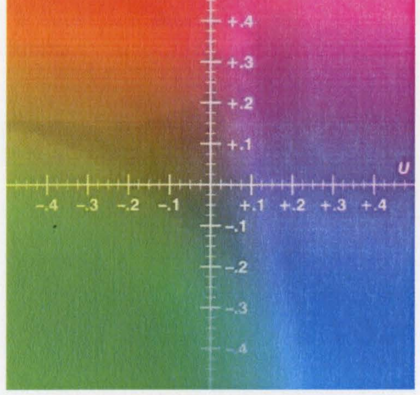


Figure 3.12: U-V colour space in YUV [46]

To show the robustness of the YUV colour space with respect to changes in illumination, the constant *c* will be added to the RGB colour parts. Positive *c* is a brighter colour while negative *c* is a darker colour. The constant *c* affects only the brightness of Y and not the colour parts of U and V. This can be shown in equations (3.2) to (3.4). It can be noted from the transformation the sum of the second and third rows in the transformation matrix from RGB to YUV is equal to zero.

$$Y(R+c, G+c, B+c) = Y(R, G, B) + c \tag{3.2}$$

$$U(R+c, G+c, B+c) = U(R, G, B) \tag{3.3}$$

$$V(R+c, G+c, B+c) = V(R, G, B) \tag{3.4}$$

Images in YUV411 colour format are represented in sets of 6 bytes (6x8 = 48 bits), arranged in the following order U, Y1, Y2, V, Y3, Y4, where all 4 Y pixels share the same U and V within the set. For instance, to represent 4 pixels in RGB it requires (4 pixels x 3 colour layers x 8 (bits/pixel/layer) = 96 bits. On the other hand, it requires 48 bits to represent the same 4 pixels in YUV format. Therefore half the amount of memory is required to represent images in YUV over RGB. This makes images faster to transmit and process. The disadvantage of using YUV is loss in the image quality. The YUV411 format will be used in the human tracking system since fast tracking is desired.

3.6 Camera calibration

Camera calibration is a fundamental part of 3D computer vision. Using calibrated cameras the original 3D structure of a scene can be reconstructed. In reality, there are at least two major differences between the camera model and the real camera. First, the focal length f of the real camera is not 1. Second, the origin of the image coordinate frame is chosen at the top-left corner of the image instead of at the center of the image. Therefore, the real image coordinate requires to be mapped to the homogenous representation.

Camera calibration is the process of finding the internal physical characteristics of the camera (intrinsic parameters) and its 3D position and orientation with respect to the world coordinate system (extrinsic parameters). The extrinsic parameters of the camera are the rotational matrix R_c^w and transitional vector t_c^w. These parameters are used to orientate and translate from the camera coordinate system to the world coordinate system. The intrinsic parameters are the image center and the camera's focal length f, which are used along with the extrinsic parameters to relate the world coordinate system to the image coordinate system. The output of the camera calibration is the establishments of a mathematical relationship between the 3D coordinate of a point in the scene and the 2D coordinate of its projection onto an image as shown in equation (3.5).

$$\begin{bmatrix} x_{im} \\ y_{im} \\ 1 \end{bmatrix} \cong \begin{bmatrix} -f_x & 0 & x_0 \\ 0 & -f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} [(R_c^w)^T - (R_c^w)^T \cdot t_c^w] \begin{bmatrix} x^w \\ y^w \\ z^w \\ 1 \end{bmatrix} \tag{3.5}$$

where x^w, y^w, z^w, are the 3D world point position, R_c^w=$R(\alpha \quad \beta \quad \gamma)$ is the rotational matrix and t_c^w=$[x^w \quad y^w \quad z^w]$ is the transitional vector, $[f_x \quad f_y]$ is the focal lengths, $[x_0 \quad y_0]$ is the image center and$[x_{im} \quad y_{im}]$ is the location of a pixel in the image.

3.6.1 Calibration Grid

The cameras are located in fixed locations inside the workcell with fixed dimensions. This fact can be utilized by calibrating the camera in a pre-processing step to identify the intrinsic and extrinsic parameters of each camera. The cameras are calibrated using Tsai's calibration method [7].

Tsai's calibration method consists of solving two problems: First, solving for the rotational matrix and the transitional vector. Second, solving for and optimizing the focal length and the radial distortion. This method uses images of objects for which the accurate geometric properties are known, and reconstructs the relationship between such properties and the recorded images. From these relations all camera parameters can be estimated. For example, Tsai's method can calibrate using the location of pre-measured 3D world points and their projection onto the 2D image.

Tsai's calibration method requires a minimum of 7 coplanar points or 4 non-coplanar points to solve the intrinsic and extrinsic parameters. To calibrate the cameras, a grid of points is defined in the workcell. This grid defines the world coordinate system with the origin located at point 1. Two grid structures have been used: The first gird consisted of 30 coplanar points as shown in Figure 3.13. The red dots represent the 2D Image location of the predefined world coordinate. This grid produces problems when viewed from an overhead camera due to the lack of depth information. A second grid consists of 120 non-coplanar points is used for calibrating the overhead camera as shown in Figure 3.14. This non-coplanar grid is based on the planar grid but with the ground grid points raised by 3 height increments of 300mm (i.e. heights 0, 300, 600 and 900mm). The red dots represent the ground level points, the black dots represent the first level, the green dots represent the second level and the blue dots represent the fourth and top level. It can be observed that the gird spacing appears larger as the points get closer to the camera. This prospective phenomenon, called parallax, is inherent in most lenses. The human eye uses this phenomenon to interpret the 3D world.

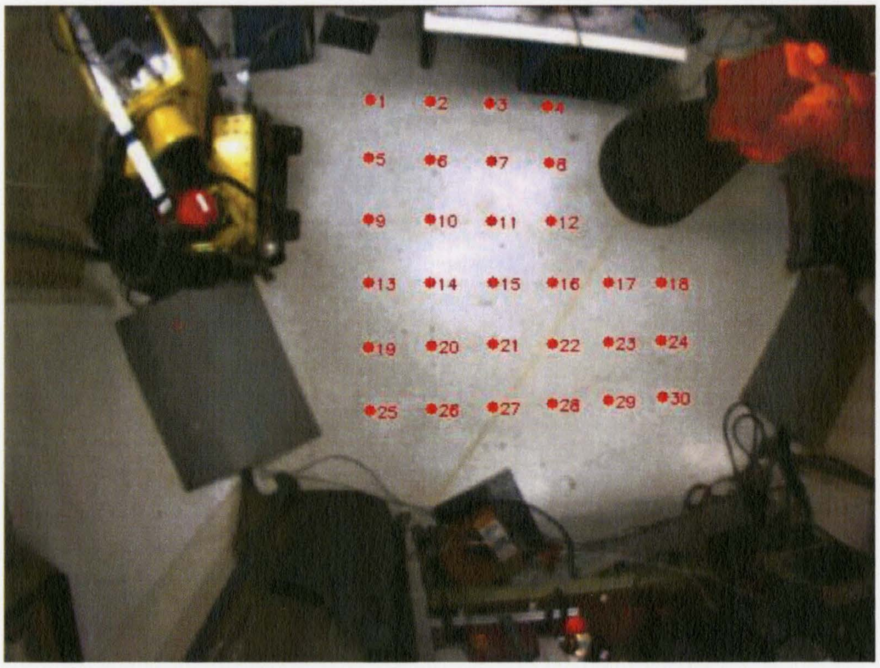


Figure 3.13: Coplanar calibration grid points

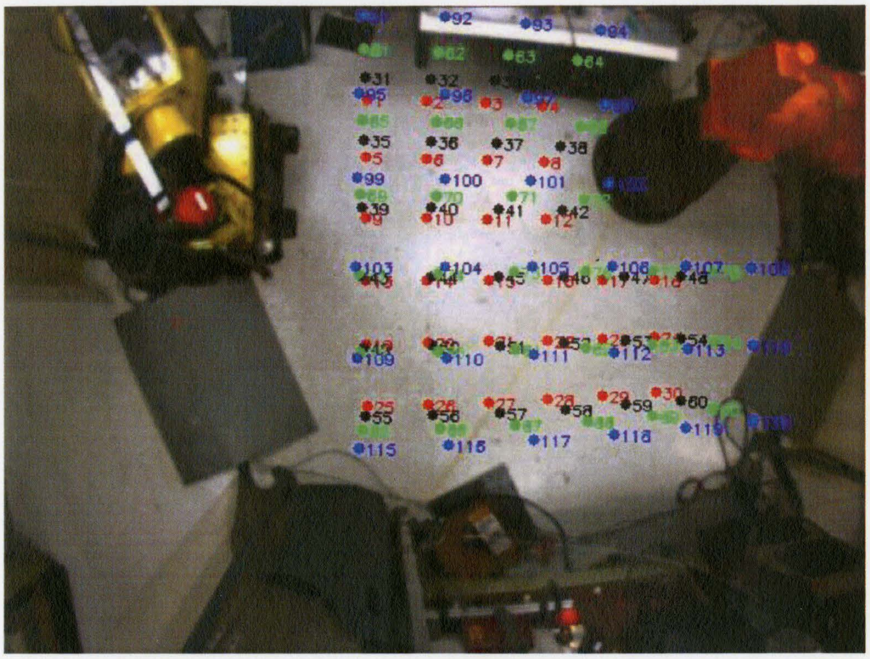


Figure 3.14: Non-coplanar calibration grid points

3.7 Conclusion

A camera system has been designed to cover the workcell at the RMAL. This camera system monitors the area around an active PUMA 762 robot. Given the workcell, two camera setups and placements have been selected. The first setup consists of using four cameras placed on the same plane at the top of the workcell. The second setup moves one of the cameras from the corner of the workcell to a location on the ceiling and facing directly downwards towards the ground plane.

Choosing the camera for the application requires understanding the difference between the various elements of the camera system. The Dragonfly2 colour camera from Point Grey Research Inc. has been selected. The colour information will help the human tracking system to distinguish between foreground and background objects. This camera has a 1/3" CCD image sensor that captures images progressively. The camera outputs data digitally on a Firewire connection. This Firewire connection along with the YUV411 colour format will improve the speed of the image transmission and processing.

A vari-focal lens from Pentax has been selected for the application. This lens has a varying focal length form 2.8-12mm. with angle of view that varies from 22.9° to 96.7°. This lens is able to view most of the workcell when placed on the cameras.

Camera calibration was used to calculate the intrinsic camera parameters such as the cameras focal length, and extrinsic parameters such as the camera position and orientation in the world. Two different calibration grids have been utilized. The first calibration grid consists of 30 coplanar points on the ground plane. The second calibration gird is used to calibrate the overhead camera and solve the problem produced by the loss of depth information. This calibration grid is an elevation of the first calibration grid to three levels. The levels are a 300mm apart in the Z direction to give 120 non-coplanar points. The cameras were then calibrated using Tsai's method.

4. Background and Foreground Processing

This chapter concentrates on the method for constructing the background model. It explains the process of building the background model used to find the foreground objects which represent the human(s) in the scene.

First, the robot is described by its colour. Then, a foreground/background subtraction method is applied to the three colour channels Y, U and V. A pixel-wise classification algorithm is next utilized to detect the human. Pixels in the image that define the robot are excluded from the scene. Image processing methods are employed to identify the objects representing the human(s) while removing noise. The output of the methods in this chapter is objects that represent the human(s).

4.1 Determining the pixels corresponding to the robot

The human and the robot are both moving objects in the scene. The background subtraction method will need to distinguish between the robot and the human. In industry robots are painted in a single solid colour. This robot colour will be exploited to distinguish the robot. The pixels that correspond to the robot are predefined using a fast classification method introduced by Bruce *et al* [47]. This method relies on classifying pixels using a Look-Up-Table (LUT). This method will now be explained in detail.

Digital cameras quantize colours into a digital form to be displayed by the device. In the YUV411 colour space, the Y channel has a range of 256 pixel values. These values can be grouped into a number of levels. For example if, 8 levels are chosen, each level will have a range of $\frac{256}{8}$ or 32 pixel values. This implies that level 4, for instance, will include pixel values that range from 128 - 159. These levels will be applied to the U and V colour channels as well.

To filter a certain colour, the colour must be given a unique binary rank. Each colour can occupy a rank of 2^n, where n is the rank. On a 32 binary number we can assign 32 different colours. For example, in an 8 bit binary number, the colour *orange* can be given the binary value [00000001] while the colour *blue* can be given the binary value [00000010].

After assigning the ranks to each colour, the colour ranges can be put in a LUT. The *orange* and *blue* colour occupies ranges in the three channels as such:

*Orange*YRange [] = {0,1,1,1,1,1,1,1}	*Blue*YRange [] = {0,1,1,1,1,1,1,1}
*Orange*URange[] = {0,0,0,1,1,1,0,0}	*Blue*URange[] = {1,1,1,0,0,0,0,0}
*Orange*VRange[] = {0,0,0,0,0,0,1,1}	*Blue*VRange[] = {0,0,0,1,1,1,0,0}

This implies that *Orange* colour can occupy a luminance pixel value from 32 – 255, a U pixel value from 96 – 191, and a V pixel value from 192-255. And the *Blue* colour can occupy a luminance pixel value from 32 – 255, a U pixel value from 0 – 95, and a V pixel value from 96-191.

We then combine the two colours into a single LUT by multiplying their ranges and the corresponding colour rank. The *orange* colour ranges are multiplied by 2^0 and the

Blue colour ranges are multiplied by 2^1. Therefore, we construct a LUT by OR'ing the *Orange* and *Blue* colour ranges. This LUT, referred to as S_r, will be used to define the colour(s) of the robots of interest. The S_r from the example will be defined as:

$$S_r \begin{cases} \text{YRange} = \{00,11,11,11,11,11,11,11\} = \mathit{Orange}\text{YRange OR }\mathit{Blue}\text{YRange} \\ \text{URange} = \{10,10,10,01,01,01,00,00\} = \mathit{Orange}\text{URange OR }\mathit{Blue}\text{URange} \\ \text{VRange} = \{00,00,00,10,10,10,01,01\} = \mathit{Orange}\text{VRange OR }\mathit{Blue}\text{VRange} \end{cases} \quad (4.1)$$

where the most significant bit in each element represents *Blue* colour and the least significant bit represents the *Orange* colour. This is in accordance with the rank of each colour.

After constructing S_r, pixel classification is performed. First the image of the current scene is grabbed in the YUV411 colour format. The current pixel colour values in Y, U and V are quantized to find their respective location in S_r. The results from the different colours are AND'ed together. If the final binary value is a 1, then a robot colour pixel has been found. Otherwise, the pixel is not a robot colour pixel. Given the previous example, there are three possible outcomes. If the final binary value was 00, then the pixel is neither *Orange* nor *Blue*. A value of 01 means the pixel is *Orange* and value of 10 which means the pixel is *Blue*. Figure 4.1 shows the scene as captured by the camera and Figure 4.2 shows the detected robot pixels. The algorithm has also detected pixels with the robot colour in the surrounding background. These pixels will be treated as noise.

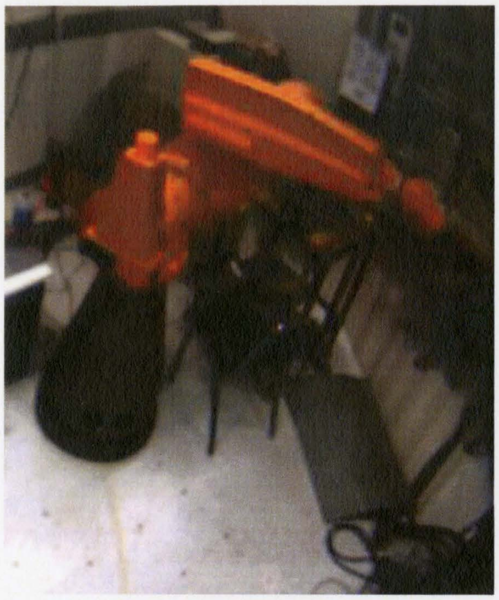


Figure 4.1: The scene from Camera 2

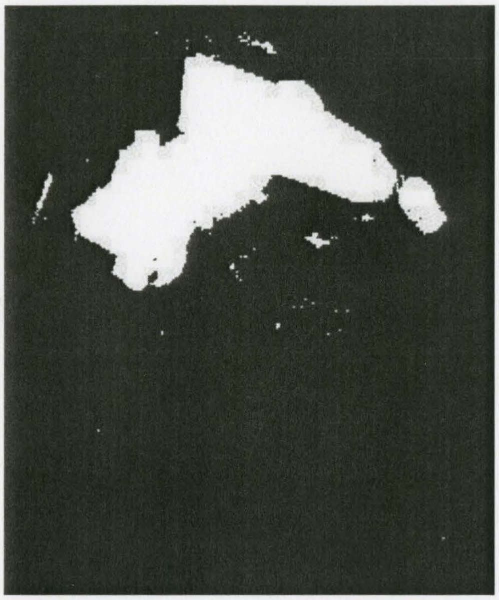


Figure 4.2: Result of the robot segmentation

4.2 Learning the background model

A simple background/foreground representation method involves subtracting each new image from a model of the background scene. Then the method thresholds the difference image to determine foreground pixels. However, this method has a drawback when it attempts to model scenes with dynamic moving objects such as a robot. The robot in these scenes will appear as a foreground object. Therefore, a solution is needed to account for the robot movements. This solution must be minimally affected by the appearance of shadows and ghosts.

To account for the problem of shadows, the background learning will be applied in the YUV411 colour format. The Y channel in this format holds luminance information about the scene. This can be beneficial in detecting shadows, since shadows only change the luminance value of a pixel to a darker region but do not affect the colour of the pixel.

Figure 4.3 shows the background learning flowchart. The robot is first moved to different locations to view as much of the background as possible. The background model is initialized by setting the minimum pixel values to white and the maximum pixel values to black. A sequence of images is recorded of the scene at a rate of 30Hz while the robot is moving. This sequence must have all the possible views of the background from the view point of the camera.

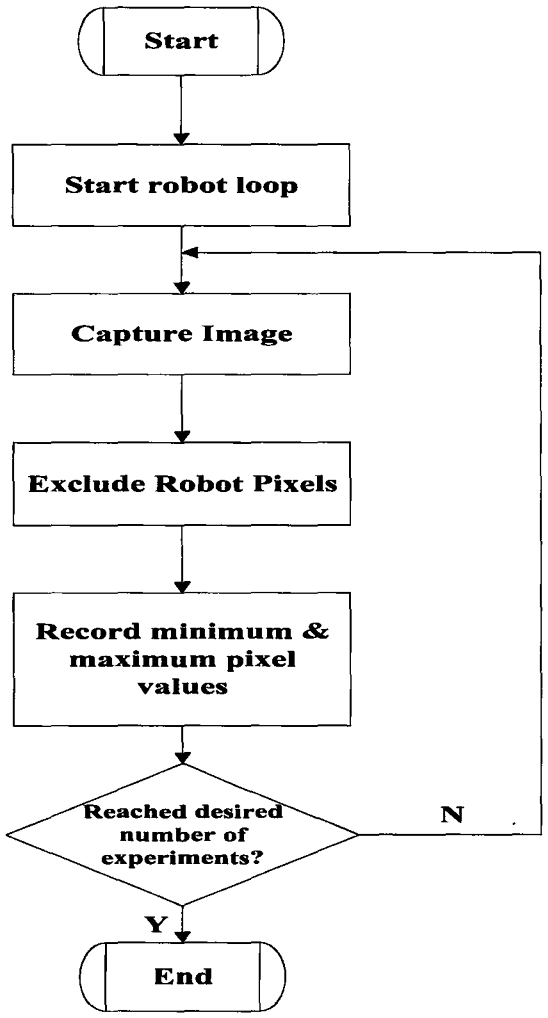

Start
Start robot loop
Capture Image
Exclude Robot Pixels
Record minimum & maximum pixel values
Reached desired number of experiments?
N
Y
End

Figure 4.3: Background learning flowchart

Let *N* be an array of consecutive images of the scene. These images capture a moving robot without any humans in the scene. And let $P = [Y, U, V]$ represent the YUV411 colour channels. The background model of each colour channel is obtained as follows:

$$\text{for } k = 1,2,\dots,N$$
$$[{}^{P}m_{ij}, {}^{P}n_{ij}] = \begin{cases} [\min({}^{k}P_{ij}), \max({}^{k}P_{ij})] & \text{if } {}^{k}P_{ij} \notin S_r \\ [{}^{P}m_{ij}, {}^{P}n_{ij}] & \text{otherwise} \end{cases} \tag{4.2}$$

Where ${}^{P}m_{ij}$ and ${}^{P}n_{ij}$ are the minimum and the maximum pixel values in the current background model, respectively; *i* and *j* are the row and column pixel positions, respectively; ${}^{k}P_{ij}$ is the value of the pixel in the k^{th} image of *N*; and S_r is the LUT of predefined colour values of the robot. Figure 4.4 and Figure 4.5 show the minimum and maximum pixel values of the process in YUV411 colour format. White pixels in Figure 4.4 and black pixels in Figure 4.5 are pixels that have always been occupied by the PUMA robot in its movements. It should be noted that the second robot has colours similar to the PUMA robot colours. This is because yellow and orange are similar in the YUV colour space. This can be seen in Figure 3.12. The inclusion of the stationary robot pixels does not affect the results of the pixel classification process.

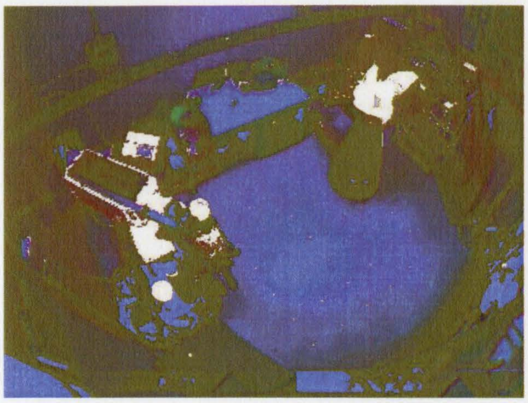


Figure 4.4: Image of the minimum pixel values

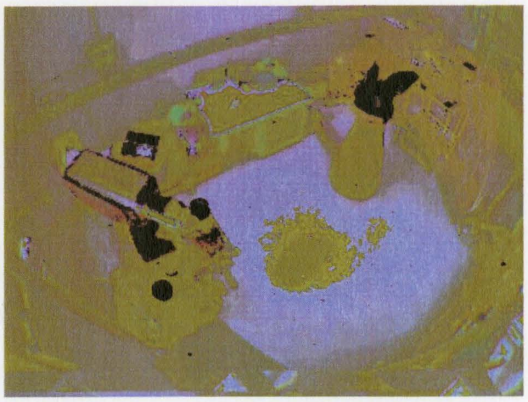


Figure 4.5: Image of the maximum pixel values

4.3 Pixel classification

To detect foreground objects using the background subtraction method, the current frame from the camera is compared to the background model. This method is affected by shadows appearing as foreground objects. To reduce the detection of shadows as foreground objects, a deterministic approach, adapted from [13] and [14], has been implemented to adjust the background model in the luminance (*Y*) colour channel. This approach is based on the theory that shadows have a smaller luminance pixel value of the background chromaticity.

Let $Q = [U, V]$, let Y_{ShPct} be the shadow tolerance percentage, let $Y_{ShOffset}$ be the shadow offset in the *Y* channel, and let Q_{Er} be the colour error percentage in the *Q* colour channels. Every pixel in the background model is then classified by the Pixel classification algorithm shown in Table 4.1.

Table 4.1: Pixel classification algorithm

1. For pixel P_{ij} , compute the following:

$$^{Y}m_{ij} = \left(^{Y}m_{ij} \cdot (1 - Y_{ShPct})\right) - Y_{Shoffset} \tag{4.3}$$

$$^{Y}n_{ij} = {}^{Y}n_{ij} \cdot (1 + Y_{ShPct}) \tag{4.4}$$

$$^{Q}m_{ij} = {}^{Q}m_{ij} \cdot (1 - Q_{Br}) \tag{4.5}$$

$$^{Q}n_{ij} = {}^{Q}n_{ij} \cdot (1 + Q_{Br}) \tag{4.6}$$

2. Apply the regions shown in Figure 4.6 to classify the pixel as foreground or background/ shadow. A pixel is classified as a foreground pixel if it has been detected as foreground in any of the colour channels (*Y*, *U*, or *V*).
3. Repeat steps 1 and 2 for all pixels in the image.

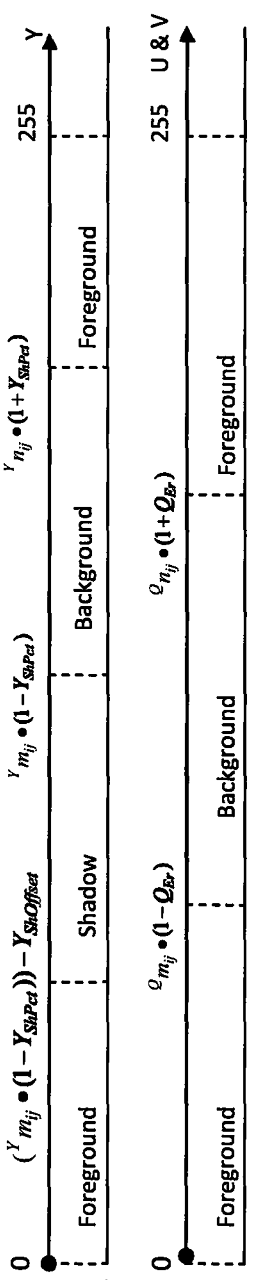

0
$({}^{Y}m_{ij} \bullet (1 - Y_{ShPct})) - Y_{ShOffset}$
${}^{Y}m_{ij} \bullet (1 - Y_{ShPct})$
${}^{Y}n_{ij} \bullet (1 + Y_{ShPct})$
255
Y
Foreground
Shadow
Background
Foreground
0
${}^{Q}m_{ij} \bullet (1 - Q_{Er})$
${}^{Q}n_{ij} \bullet (1 + Q_{Er})$
255
U & V
Foreground
Background
Foreground

Figure 4.6: Luminance and chrominance detection regions

4.4 Robot exclusion

In our application, we would like to detect the space occupied by the human while ignoring the robots' movements and location. Ignoring the robot's location during construction of the background model will cause occlusion of some parts of the background scene. The occluded areas will later incorrectly appear as foreground regions when the robot moves. This problem, as explained in section 2.2.2, is referred to as *ghosts*.

In the industry, robots are usually distinctly coloured relative to their backgrounds. We exploit this in our tracking method. The colour of the robot will be used to detect the robot's location and exclude it from the image. Hence, a human cannot wear a colour similar to the colour of the robot, as stated in the assumptions in section 1.3. The robot exclusion is done by the foreground segmentation algorithm presented in Table 4.2.

Table 4.2: Foreground Segmentation Algorithm

```
Step 1:  A coloured image $I$ is acquired from the camera
Step 2:  A binary foreground image $F$ is set to black
Step 3:  If the pixel ${}^{I}P_{ij}$ of the image $I$ is not in $S_r$, then
             If the background model is still at the initial state, then
                 Set the pixel $F_{ij}$ to one.
             Else
                 If the luminance $Y_{ij}$ in image $I$ is less than ${}^{Y}m_{ij}$ and greater than
                 the value ${}^{Y}n_{ij}$, then
                         Set the pixel $F_{ij}$ to one.
                 End If
                 If the $Q_{ij}$ in image $I$ is less than ${}^{Q}m_{ij}$ OR greater than ${}^{Q}n_{ij}$, then
                         Set the pixel $F_{ij}$ to one.
                 End If
             End If
         End If
```

4.5 Image processing

Image processing is any form of signal processing where the input is an image and the output is an image or a set of characteristics or parameters related to the image. For digital images, simple and complex image processing algorithms have been written. These algorithms are used to acquire information from the images for further analysis.

In this project, we would like to locate the pixels that are occupied by the human(s). The background subtraction method produces a foreground image containing the human(s). This foreground image, which results from the background model pixel classification, usually includes noise and other objects that do not represent a person or their limbs. To remove the noise, two image processing algorithms have been utilized to find and represent the human(s) within the foreground image.

4.5.1 Eroding

Eroding is used to remove noise pixels that look like grains of salt in the images. Figure 4.7 shows the foreground image after pixel classification. It can be noted that the image has white noise pixels. The noise pixels are the result of various elements such as the fluctuations of the light source, reflections of light from the subject to the surroundings, etc.

Eroding is achieved by increasing the number of black pixels by changing white pixels to black if the white pixel was neighbouring a black pixel. Figure 4.8 shows the classification image after eroding.

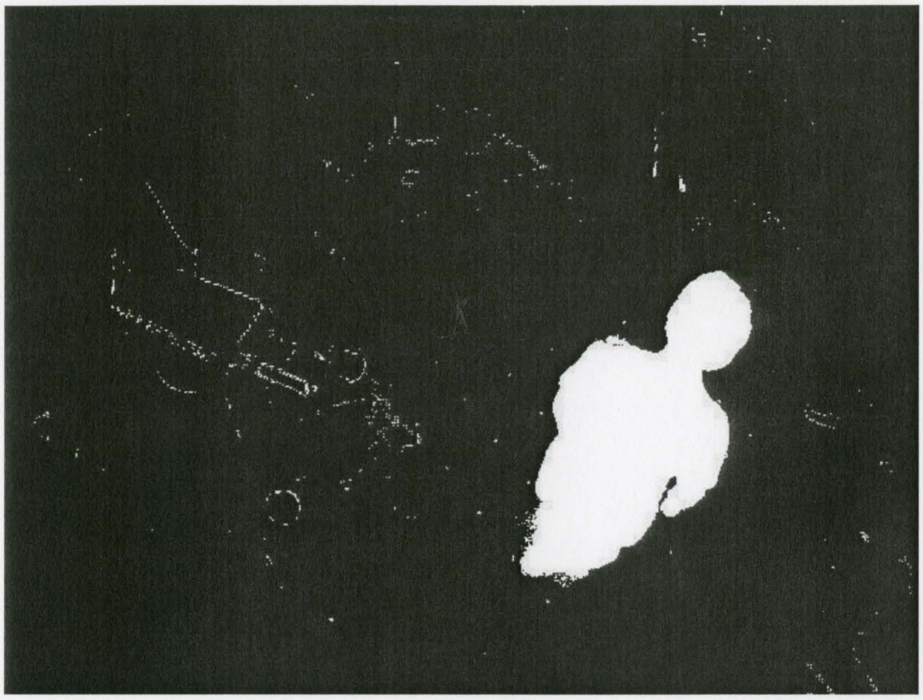


Figure 4.7: Foreground image

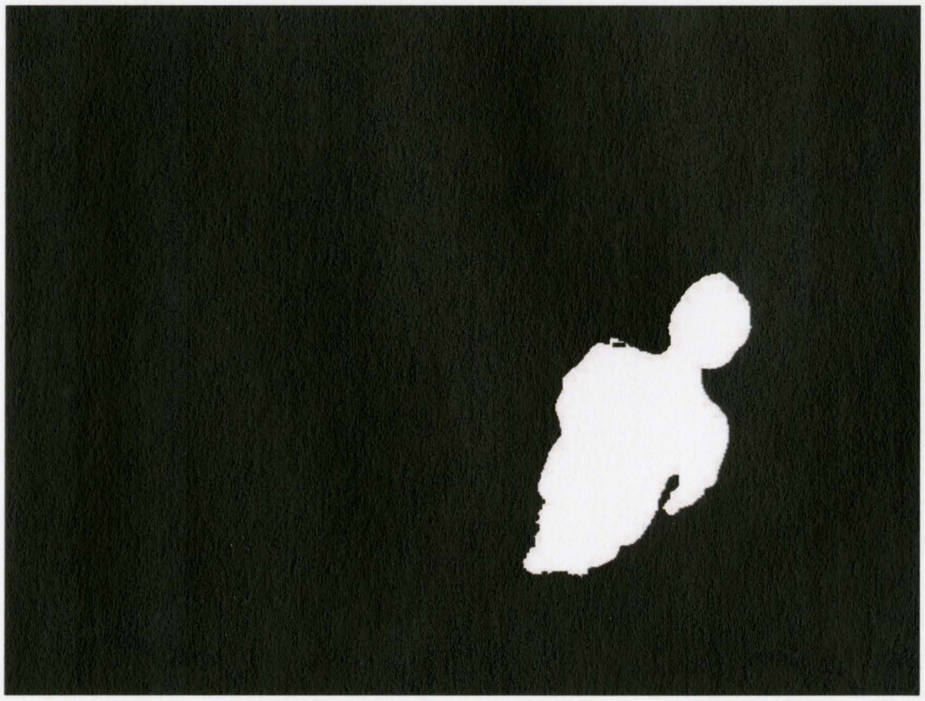


Figure 4.8: Foreground Image after eroding

4.5.2 Active contours

Active contours are active shapes that respond to the pixel values within an image. This method was first introduced by Kass et al [19]. It can find the boundary of a set of pixels that have the same pixel value. Active contours are controlled by external and internal energy functions. The former fits curves to the pixel edges, while the latter adjusts to the smoothness of the curve.

In this project, active contours are used to find the outlines of the pixels that are the result of pixel classification and eroding. A limit on the smallest area specified by the contour is applied to further remove noise and other objects that cannot represent a person. Figure 4.9 shows the boundary pixels given by the active contour algorithm. This is clearly the outline of the person.

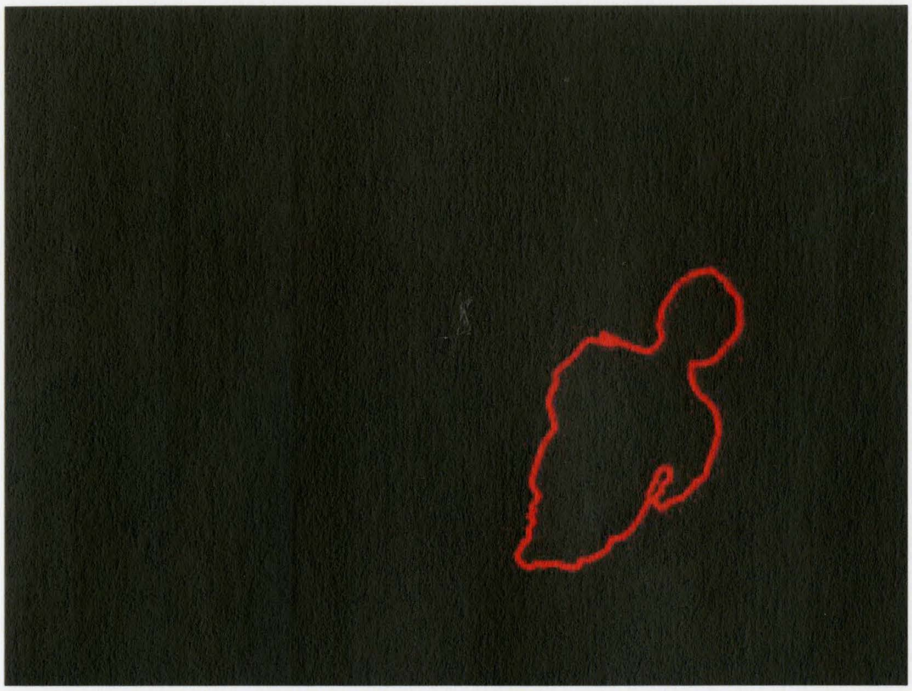


Figure 4.9: Active contour boundaries

4.6 Conclusion

An object that represents the human(s) in a scene is represented in this thesis using active contours. This object is acquired by a background/foreground subtraction method. This method relies on building a background model and subtracting successive images from the background model to produce an image of the foreground objects. A drawback of this method is that the robot movements will also appear as a foreground object. A robot exclusion algorithm is applied to remove the robot from the scene. This algorithm relies on the robot having a distinctive single colour. Therefore, the human cannot wear a colour similar to the robot colour.

Human movements in the scene cause shadows to appear as foreground objects. To reduce the effect of shadows, the images are acquired in the YUV411 colour space.

This colour space is represented using luminance in the Y channel and chrominance in the U and V colour channels. Shadows are removed using detection regions. A fast pixel classification method is then applied to classify pixels as foreground or background/robot/shadow. This method relies on building a LUT of the pixels that represent the robot.

Erosion is then applied to foreground image to remove noise. The remaining foreground pixels are grouped into active contours. A minimum size limit is applied to remove contours not due to human(s). Finally, the human(s) can be represented by the active contours.

5. Polyhedron Construction

The objective of this chapter is to construct a three dimensional polyhedron that encloses each human using the data provided by the multiple cameras. Each camera provides a set of active contours that represent the human(s) from the view point of the camera as described in the previous chapter.

The active contours will be grouped using either *minimum bounding rectangles* or *convex hulls*. For the case of multiple people inside the workcell, pattern recognition algorithms are utilized during the grouping process to detect the contours that belong to each human. Next, half spaces are constructed to surround the groups of contours. A convex polyhedron is then constructed from the intersection of half spaces given by the multiple cameras. Occlusion handling methods are utilized to find the correct polyhedron. Each polyhedron will enclose the space occupied by each human.

5.1 Grouping of contours

The results of the background modeling are active contours that represent the human(s) in the scene. These active contours can appear as separated curves due to partial occlusions or camouflage. For example, in Figure 5.1 a person has entered the workcell. Figure 5.2 shows the resulting five contours of the various detected parts of that person. The similarity in colour between the person and the background (i.e. the robot base) caused the system to categorize parts of the person as background. The contours formed in Figure 5.2 need to be grouped to outline the person. Two methods of contours grouping have been applied: *minimum bounding rectangle* and *convex hull.*

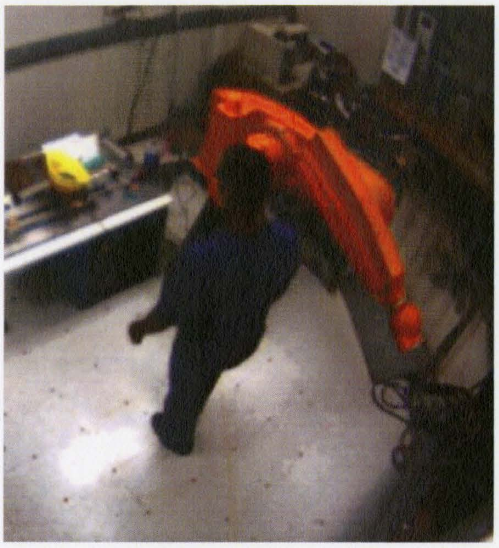


Figure 5.1: Image of a person entering the workcell

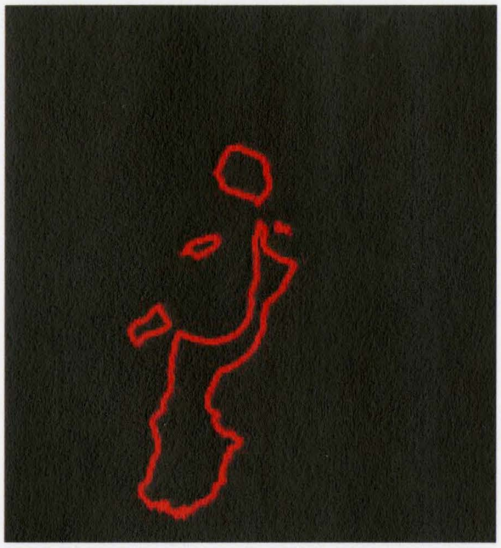


Figure 5.2: Various detected contours of the person

5.1.1 Minimum bounding rectangle

With this method the contours from the background modeling are bounded using the smallest possible rectangle. This rectangle is defined as the *minimum bounding rectangle* (*MBR*). The purple rectangle in Figure 5.3 shows the result given by the *MBR*. This algorithm is fast in grouping the contours together. It only requires finding the location of the extreme top/left/right/bottom pixels. On the other hand, it encloses extra space that is not occupied by the person.

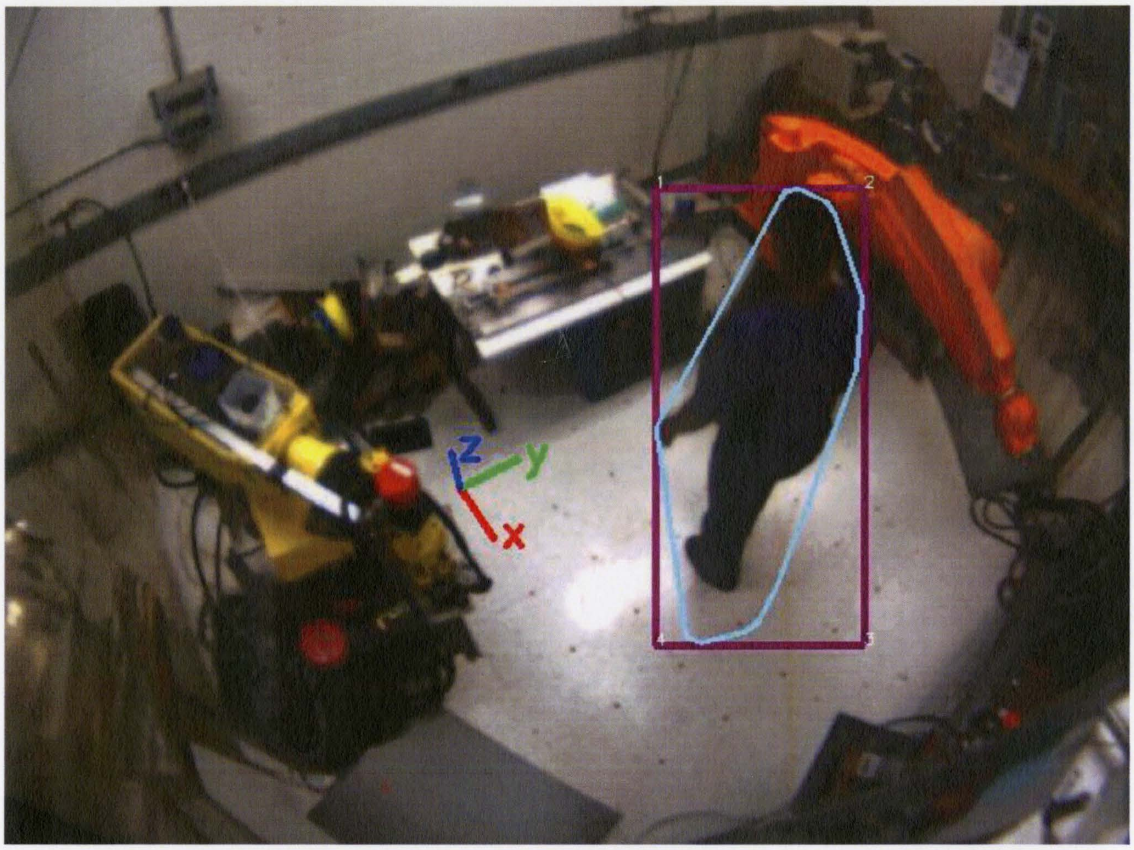


Figure 5.3: The Minimum Bounding Rectangle (purple) and Convex Hull (cyan)

5.1.2 Convex hull

Convex hulls will also be used to group the active contours. Given a set of points *X*, the convex hull is the minimum convex set containing *X*. For coplanar points, the convex hull is a simple closed polygonal chain. This can easily visualized as an elastic band stretched open to encompass the points or object. Figure 5.4 shows an example of a convex hull (green) for a set of points (red).

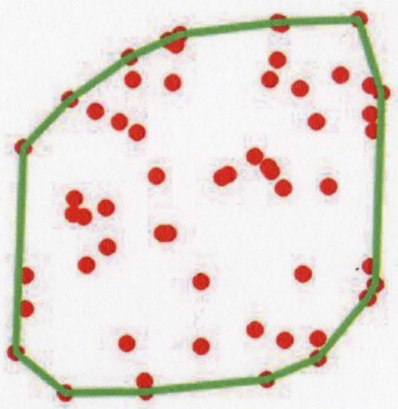


Figure 5.4: The convex hull (green) for a set of a points (red)

The active contours include a set of 2D points that represent the person. Therefore, they can be contained in a convex hull. The fast convex hull algorithm given by A.M. Andrew [48] will be implemented in this research. This algorithm, presented in Table 5.1, has an order of O(*n log n*). The algorithm first sorts the points with respect to the top-leftmost point. Then it finds the minimum and maximum points in the x and then y coordinates. The points are then formed into two chains, each of which is then subject to an iterative point-elimination process so that one of the chains comes to represent the left side of the hull and the other the right side. The intermediate points are allocated to one of the chains according to the side on which they fall of a line joining the lowest point to the highest point. Finally points are added to a stack that represents the convex hull of the points when the algorithm terminates.

Although the convex hull defines a smaller space than the MBR, it requires more computations and a longer time to produce an answer. The output of the convex hull is a set of two dimensional points that outline the subject. The cyan lines in Figure 5.3 represent the convex hull of the person originally shown in Figure 5.1.

Table 5.1: Convex hull algorithm [48]

```
Input: a set $S = P_{ij}$ of $N$ points
Sort $S$ by increasing $x$- and then $y$-coordinate.
Let $P$[] be the sorted array of N points.
Get the points with 1st $x$ min or max and 2nd $y$ min or max
    minmin = index of $P$ with min $x$ first and min $y$ second
    minmax = index of $P$ with min $x$ first and max $y$ second
    maxmin = index of $P$ with max $x$ first and min $y$ second
    maxmax = index of $P$ with max $x$ first and max $y$ second
Compute the lower hull stack as follows:
(1) Let L_min be the lower line joining $P$[minmin] with $P$[maxmin].
(2) Push $P$[minmin] onto the stack.
(3) for i = minmax+1 to maxmin-1  (between the min and max)
      {
        if ($P$[i] is above or on L_min)
            Ignore it and continue.
        while (there are at least 2 points on the stack)
        {
          Let $P_{T1}$ = the top point on the stack.
          Let $P_{T2}$ = the second point on the stack.
          if ($P$[i] is strictly left of the line from $P_{T2}$ to $P_{T1}$)
              break out of this while loop.
          Pop the top point $P_{T1}$ off the stack.
        }
        Push $P$[i] onto the stack.
      }
(4) Push $P$[maxmin] onto the stack.
Similarly, compute the upper hull stack.
Let $W$ = the join of the lower and upper hulls.
Output: $W$ = the convex hull of $S$.
```

5.2 Pattern recognition

Pattern recognition aims at classifying data based on a prior knowledge about the data. The patterns to be classified are usually groups of observations defining points in a multidimensional space. There are two common types of pattern recognition: *supervised learning* and *unsupervised learning*.

Supervised learning is based on the availability of a set of data points that have already been classified. This set of patterns is termed the training data set. It consists of corresponding groups of input and output data points. The task of the supervised learner is to predict the value of the function for any valid input object after having seen a number of training examples. To achieve this, the learner has to generalize from the presented data to unseen situations in a reasonable way. This is commonly used in designing artificial neural networks.

Unsupervised learning aims to achieve the same task as supervised learning but without the use of a training data set. This technique is used when collecting and labelling a set of training patterns is time consuming or infeasible. This method can group data with similar traits into different groups called *clusters*.

Having multiple people in the workcell and detecting various limbs of the different people using contours, makes relating the limbs to the different people challenging. Figure 5.5 shows an image of two people and Figure 5.6 shows the foreground image of the various detected parts of the two people.

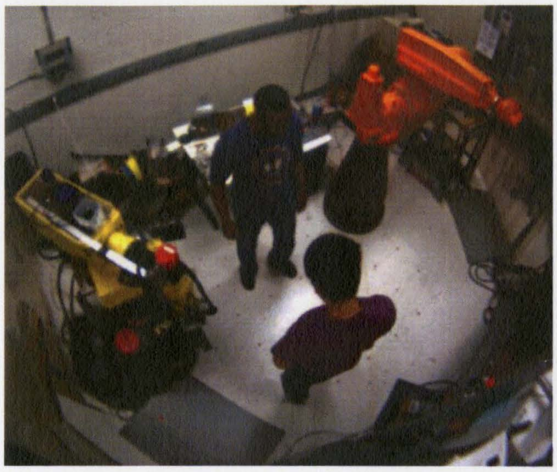


Figure 5.5: Image of two people

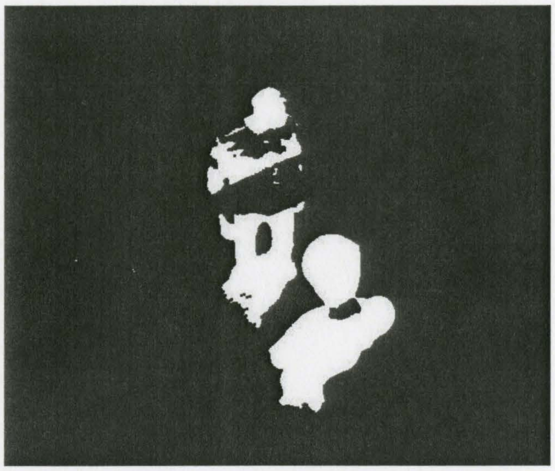


Figure 5.6: Foreground Image of two people

Two unsupervised pattern recognition methods have been applied in this project to solve the problem of correctly relating the limbs to the humans. These methods are the *k-nearest neighbour clustering* and *k-means clustering*. They relate the limbs based on the Euclidian distance between the various limbs that are originating from each person.

5.2.1 k-Nearest Neighbour Clustering

Given a set of points, the nearest neighbour algorithm is based on finding the shortest distance between each pair of adjacent points and assigning them to a cluster. It terminates when the distance between nearest points in the different clusters exceeds an arbitrary threshold value. The algorithm is allowed to continue until all of the points are linked. The result of the algorithm is a *minimal spanning tree.* The nearest neighbour algorithm can be terminated by defining the number of clusters *k*. This is termed the k-nearest neighbour algorithm.

The k-nearest neighbour algorithm can be used to relate the limbs of the different people. It will start by finding the center of each contour. It can then apply the nearest neighbour algorithm to relate the centers of the contours and assign them to clusters. These clusters represent the different people. The nearest neighbour algorithm was easy to implement and could execute quickly. Nevertheless, it usually misrelated the contours. For instance, if the body of the person was occluded by the robot, the algorithm failed in relating the head of the person to their feet. This is because the chosen distance threshold value was not large enough to include the large distance between the contour of the head and the contour of the feet. Choosing a larger threshold solves this problem but can then cause the contours belonging to two people to be incorrectly grouped together.

5.2.2 k-Means Clustering

The k-means algorithm is based on assigning each point in a finite set of points to the cluster whose centroid is nearest. The centroid of the cluster is the average of all the points in the cluster. That is, it is the arithmetic mean of the coordinate of the points. Given an a priori knowledge about the number of clusters k and a finite set contour centers, the algorithm can group the centre of each contour to *k-th* centroid. This associates each contour to a cluster.

This algorithm will be utilized to relate the contours of each person together. Table 5.2 shows the steps of the k-means clustering algorithm. Where λ_r is the centre of the *r-th* contour. The output of the algorithm is clusters ($\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_k$) of the classified centers of contours. If k is equal to 2, then the centers of the contours will be assigned to one of two clusters. Figure 5.7 shows the result of two people in the workcell. The red lines represent the boundaries of the active contours, the cyan dot represents the mean of all the contours $P_{AllMean}$, and the dark blue dots represent the centroids, μ_1 and μ_2, of the clusters $\mathcal{B}_1$ and $\mathcal{B}_2$, respectively. The centres of the contours λ that belong to cluster $\mathcal{B}_1$ and $\mathcal{B}_2$ are represented by the gold and green dots, respectively.

Table 5.2: k-means clustering algorithm

```
Input: Centres of the contours $\lambda$
Steps: Computer the mean $P_{AllMean}$ of all $\lambda$ values
        Split $P_{AllMean}$ into $\mu_k$ centroids of clusters by moving them by a distance $d$ in
            opposite directions away from $P_{AllMean}$
        Do (classify $\lambda_r$ means according to the nearest $\mu_k$)
            Re-compute $\mu_k$
        Until no change in $\mu_k$
Output: $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_k$
```

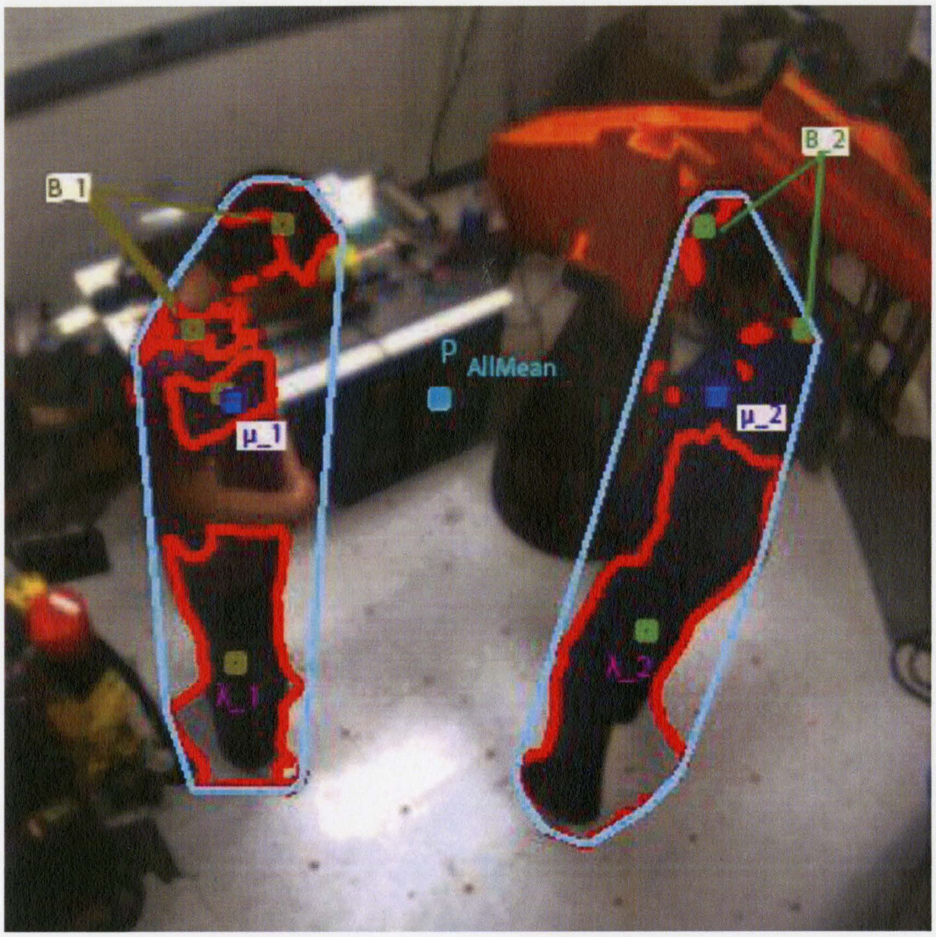


Figure 5.7: The result of k-means clustering for two people

The main advantages of this algorithm are its simplicity and speed which allows it to run on large sets of contours. Although it minimizes intra-cluster variance, it does not ensure that the result has the global minimum variance. This algorithm will be used in the human tracking system.

5.3 Half spaces construction

A half space is one of the two parts of space into which a plane divides three dimensional (3D) space. The normal of the plane defines the positive side of the half space. For instance, the space above the floor of the workcell can be defined as a half space with the normal pointing upwards from the floor. In Figure 5.8 the space is divided by the red plane, which defines the blue half space. A half space in 3D space can be defined as:

$$a_h x + b_h y + c_h z + d_h \leq 0 \tag{5.1}$$

where $[a_h\ b_h\ c_h]$ is an outwardly pointing normal vector; $d_h$is the plane offset; and h is the plane number.

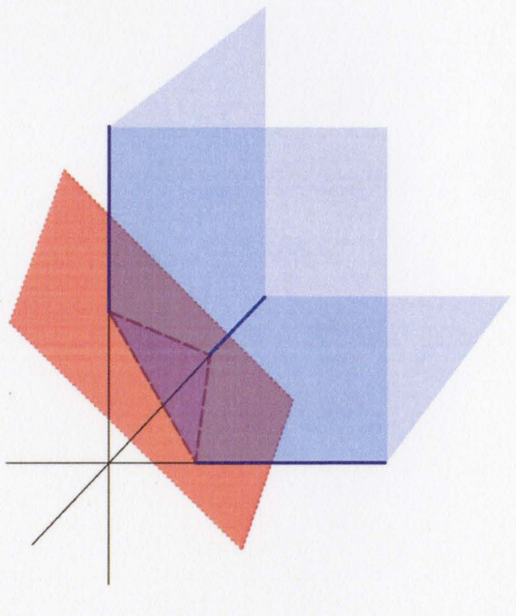


Figure 5.8: The half space (blue) defined by the half plane (red) [49]

Defining a plane in 3D space requires three or more coplanar points. Therefore, the camera origin location and two points on the image will produce a plane that divides the space into two half spaces. Using the points from the grouping of the contours, either by the *MBR* or the *convex hull*, will result in the production of multiple half spaces.

5.3.1 First Case: Minimum Bounding Rectangle

The *MBR* defines the four corner points of the rectangle that encloses the person in the view of each camera C_l. The camera's origin and each corner point of the *MBR* are mapped to the world coordinates to create four vectors in 3D space. Each pair of vectors, corresponding to sequential *MBR* corner points, is used to create a half space whose normal vector points outside of the *MBR*. Therefore, each camera that has foreground pixels in the foreground image *F* will produce four half spaces. These half spaces along with the floor half space will be used to construct a convex polyhedron in the shape of a pyramid, where the location of the camera is the apex of that pyramid. Figure 5.9 illustrates the process of constructing the half spaces. The camera starts by using the floor half space. Then it constructs the right, bottom, left and top half spaces sequentially.

5.3.2 Second Case: Convex Hull

The points given by the convex hull are processed in similar manner to the *MBR*. The convex hull defines multiple points that enclose the person in the view of each camera. The camera's origin and each two sequential points produce two vectors in 3D space. These pairs of vectors create half spaces with the normals pointing outside of the convex hull. Therefore, each camera that has foreground pixels in the foreground image *F* will produce multiple half spaces. These half spaces along with the floor half space will be used to construct a convex polyhedron.

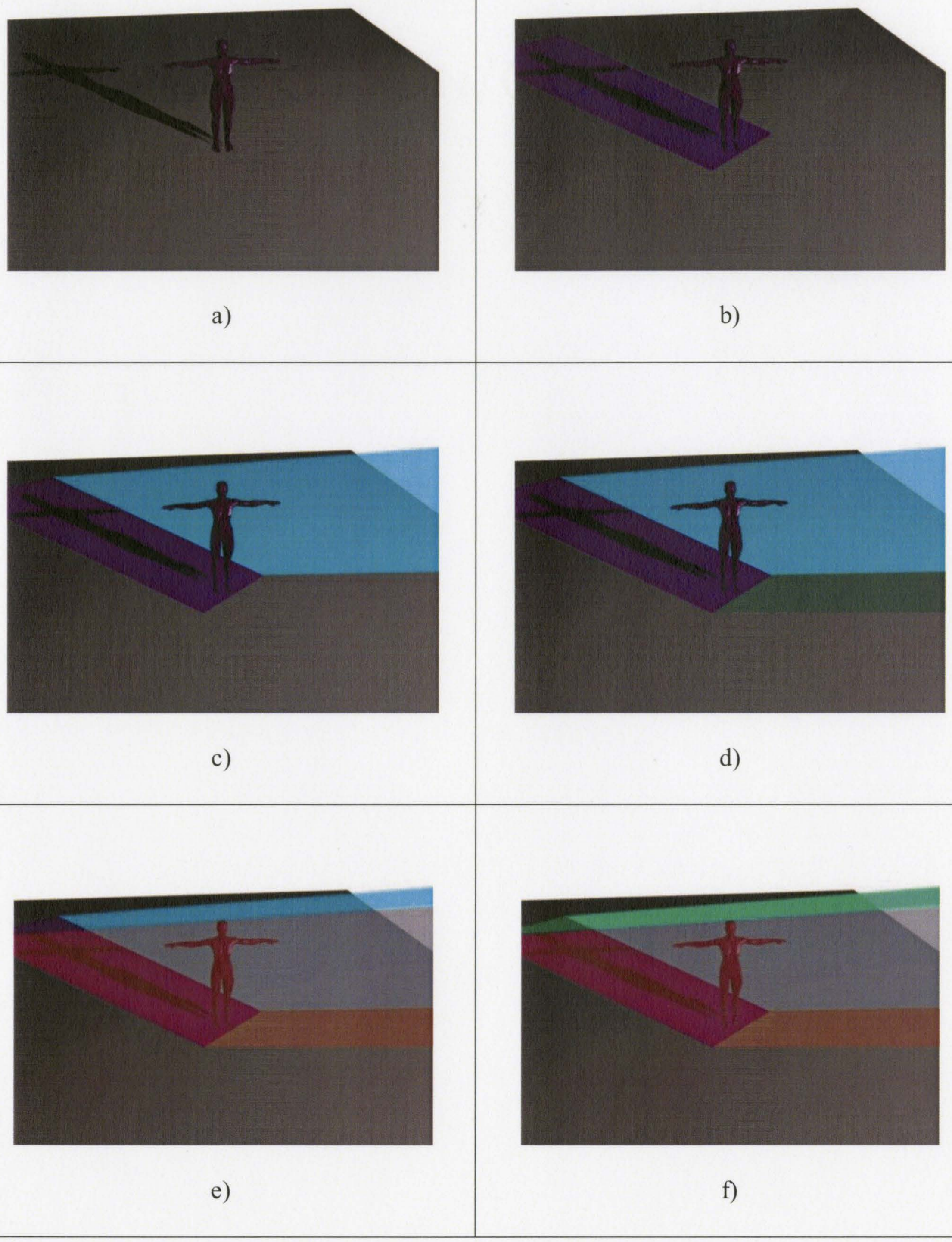


Figure 5.9: Process of constructing the half spaces

5.4 Enclosing polyhedron

Constructing the 3D polyhedron requires: the half-spaces given by the cameras, the floor's half-space and an interior point to the polyhedron. The half-spaces given by the set of cameras $[C_l, \dots, C_L]$, where L is the number of cameras, will intersect in a convex polyhedron that encloses the human(s). To find this region, we first need to find a point that lies inside all of the half-spaces. This inner point is found using the linear programming interior-point search method from [50] and [51]. The linear program is formulated as follows:

$$\begin{aligned} & \textit{minimize}\ x_5 \\ & \textit{subject to}\ a_k x_1 + b_k x_2 + c_k x_3 + d_k x_4 - x_5 \leq 0 \qquad \forall k = 1,2,\dots,N \\ & x_4 > 0 \text{ and} \\ & x_5 > 0 \end{aligned} \tag{5.2}$$

Where N is the number of half-spaces ($N \geq 4L$). Then, an inner point of all half-spaces equals $\left(\frac{x_1}{x_4}, \frac{x_2}{x_4}, \frac{x_3}{x_4}\right)$

Next, the smallest convex set of vertices representing the intersection of the half-spaces and hence the polyhedron is found. The intersection algorithm is given in [51]. Figure 5.10[a-f] illustrates the process of constructing the polyhedron from the intersection of the half-spaces given by the four cameras.

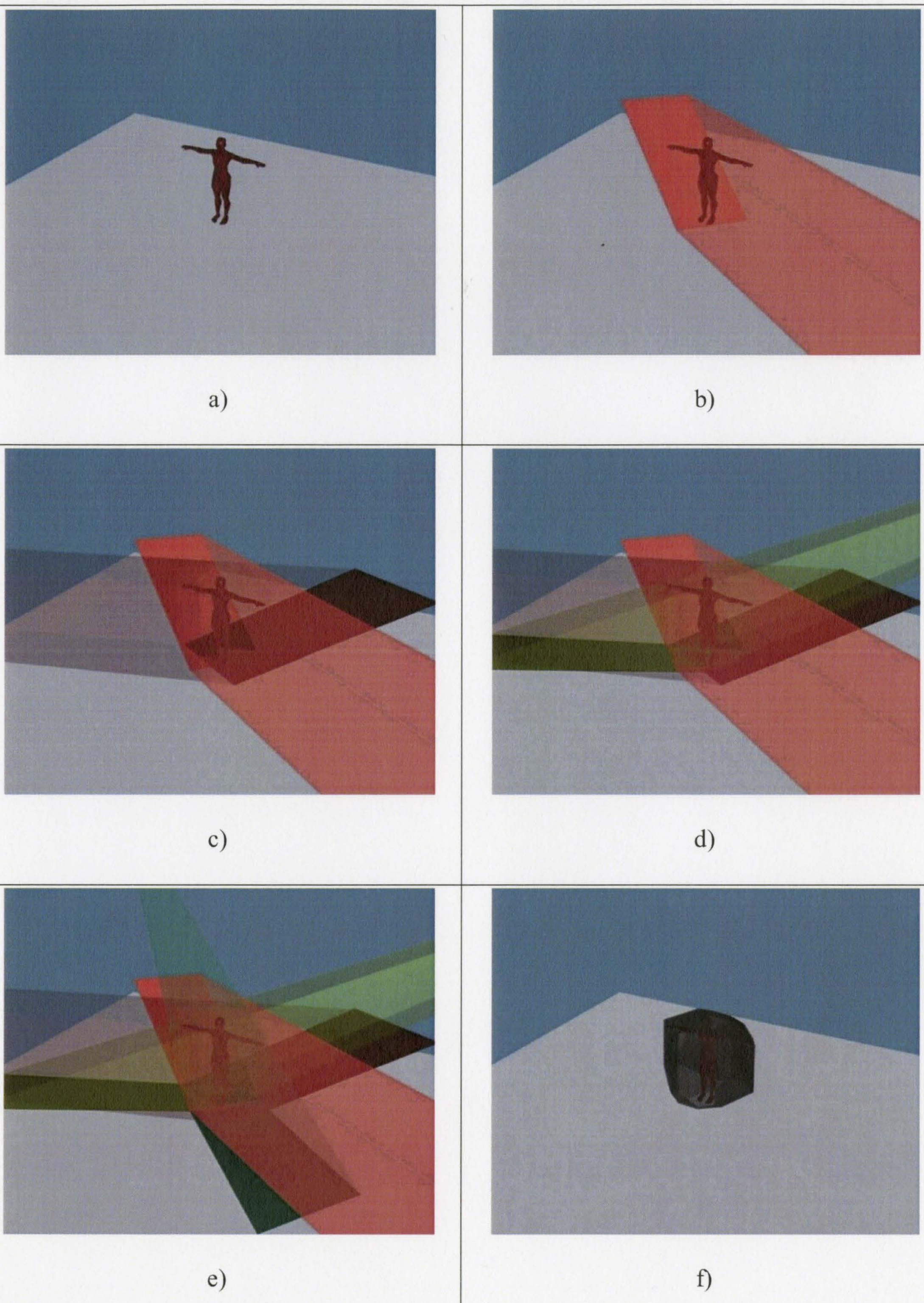


Figure 5.10: Process of constructing a convex polyhedron that encloses a person

The constructed polyhedron will include a volume not occupied by the human. This volume overestimation will provide a clearance that increases the human's safety in collision avoidance applications.

5.5 Occlusion handling

Occlusions occur when the person is blocked from the view of the camera by the robot or the human body parts. A robust camera system should be able to detect occlusions and act correctly in response to the occlusion. There are three cases of occlusions that can exist with a multiple camera system. The first case is full occlusion. If the human is fully occluded in the view of one camera, then the camera will be oblivious to the presence of the human. In this case the remaining unoccluded cameras are used to generate the polyhedron of the person.

The second case occurs when the top part of the person is occluded by the robot. For example, the entire body of a person can be seen by three cameras, but the top half of the person is occluded by the robot in the view of the fourth camera. Since a polyhedron is constructed from the intersection of half spaces from multiple cameras, the half-spaces from the fourth camera will truncate the polyhedron to roughly half its size. This will cause the meaningful information given by the three unoccluded cameras to be lost. To solve this problem, the system detects the camera or cameras with polyhedron height results outside of a reasonable lower bound for the human height (*i.e.*, 1.5-1.8m [52]).

Note that a polyhedron can be built from the grouping of contours from a pair of the L cameras. The system begins by pairing each camera with the other $\boldsymbol{L-1}$ cameras to produce $L-1$ polyhedrons. If the maximum height of all $L-1$ polyhedrons is less than the lower bound for the human height, then that camera is eliminated when building the final polyhedron. If elimination of cameras continues until there are only two cameras left. Then occlusion is ruled out as the cause (*e.g.* the person is not standing up) and all cameras are used to construct the polyhedron.

The third case arises when the robot occludes the bottom part of the person. For instance, if this occurs in the view of one camera, it will cause the polyhedron to float above the floor level. This is an invalid result since the person must be touching the floor. To handle this case, the constructed polyhedron is extended to the floor (*i.e.* projected onto the Z=0 plane) when its lowest vertex is found to be above the floor level. Interior vertices created by this extension procedure are removed by computing the convex hull.

It should be noted that a special problem may occur when the cameras, robots and humans are located in the work area such that the half-spaces constructed by the cameras do not intersect to form a single polyhedron. Normally the cameras can be arranged to eliminate the occurrence of multiple full occlusions and avoid this problem. Furthermore, adding an overhead camera assists in detecting multiple people in the area as will be seen in section 6.4.

5.6 Conclusion

A convex polyhedron that represents the human(s) in the work cell is constructed from the view of multiple cameras. Active contours given by the multiple cameras are grouped to represent the human(s). Two grouping methods are utilized: *minimum* *bounding rectangle* and *convex hull.* These two methods form a boundary around the active contours.

The k-means clustering pattern recognition algorithm is employed to group the active contours belonging to each person. This algorithm separates the multiple contours based on the Euclidian distance between the contours. It requires a priori information about the number of the people in the area. The result is a group of contours for each individual.

Half spaces are constructed to define the space occupied by each group of contours. This space is specified by the location of the camera origin in the world coordinate system and the edges of the group of contours in the view of each camera. The space was in the form of a pyramid with the person located somewhere inside that pyramid. The intersection of the pyramids from the multiple cameras produces a convex polyhedron. Occlusion handling methods are used to correct the problems introduced by occlusions of body parts. Each polyhedron defines the space occupied by each person inside the workcell. A robot can later be programmed to be restricted from the space occupied by these the polyhedra.

6. Experimental Results

6.1 Introduction

This chapter presents an overview of the system hardware and software; and the human tracking experimental results.

The software states has been divided into three processing modules: pre-processing module, image processing module and polyhedron construction module. The pre-processing module includes the steps that are executed prior to entering the tracking loop. The image processing module includes the image acquisition and processing algorithms. Sample outputs of the image processing module are included to demonstrate its functionality. The final output of this module is a set of active contours that represent the human(s). The polyhedron construction block includes grouping of the contours, the half-space construction, occlusion handling and construction of the polyhedron(s) that encloses the human(s).

The results from two series of experiments are included for two different arrangements of the cameras. In the first arrangement, the camera locations were C1 -C4 as shown in Figure 3.4. These cameras were calibrated using a coplanar grid. In the second arrangement, the selected camera locations were C1, C2, C3 and C5. These cameras were calibrated using non-coplanar grid.

The experiments are divided into single person tracking and multiple people tracking. The vision tracking system starts as soon as the human(s) enter the workcell. The polyhedron construction and timing response results are shown for human tracking using the *MBR* and the convex hull methods. The system's robustness to different configurations of the human body and different occlusions are also presented.

6.2 System Overview

The system hardware consists of four colour Point Grey Research Dragonfly2 cameras equipped with wide angle lenses. Two cameras acquire images of size 640 x 480 pixels and the other two cameras acquiring images of size 512 x 384 pixels. These cameras operate at 60 frames/s or 60Hz. The computer is a Pentium 4 PC with a 2.0 GHz dual core processor.

The system software programs were written in the C++ language and compiled using Microsoft Visual Studio 2005. The image acquisition is achieved using Point Grey Research Inc. software library FlyCapture v1.6 Release 23 [53]. The image processing is preformed using Intel® Open Source Computer Vision Library (OpenCV 1.0) [54]. Parallel processing is achieved using Open Multi-Processing (OpenMP 2.0).

The software has been divided into three modules: *Pre-processing module*, *Image processing module*, and *Polyhedron construction module*.

6.2.1 Pre-processing Software Module

Figure 6.1 shows the flowchart of the pre-processing module. First, the module is initialized to set the camera parameters such as: image size, the colour of the robot, camera shutter speed, brightness, exposure, etc. Next, the cameras are calibrated using Tsai's Algorithm [7] as described in section 3.6. After that, the background model of the area is learnt with an operational robot moving continuously as described in section 4.2.

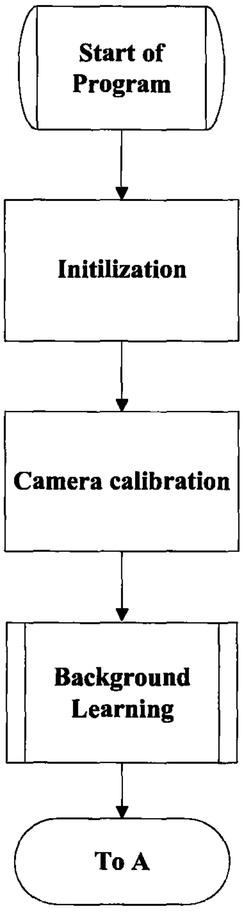


Figure 6.1: Flowchart of the Preprocessing module

6.2.2 Image Processing Software Module

After finishing the pre-processing block, the system enters the human tracking phase. This phase consists of a continuous execution loop of the image processing module followed by the polyhedron construction module. Figure 6.2 shows a flowchart of the image processing module. To increase the execution speed of the system, the computer processes the cameras in parallel. The image data from each camera is processed by a separate processing thread using OpenMP 2.0. The execution speed of the system without OpenMP is 3.6Hz. Using OpenMP increases the execution speed of the system to 10.5 Hz.

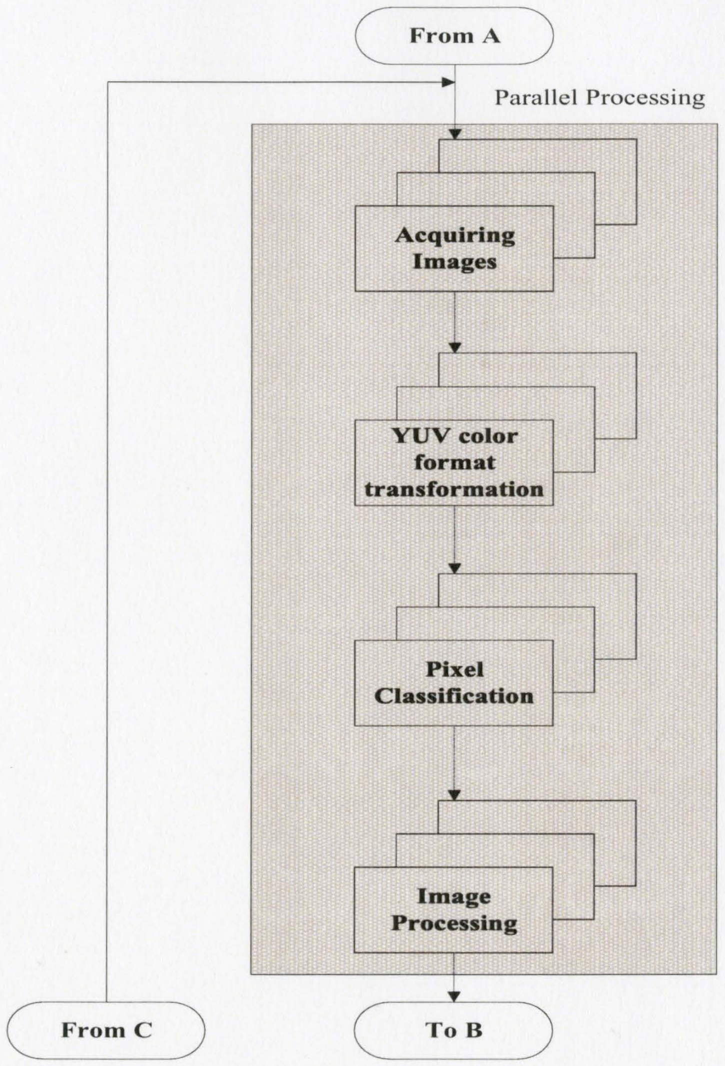


Figure 6.2: Flowchart of the image processing module

The image processing block starts by acquiring the images. The images are acquired using Bayer Raw colour format to maximize the data transfer rate from the camera to the PC. Then the images are transformed to YUV411 colour format to improve the speed of processing the images. Figure 6.3 shows the results of an image given by the second camera in YUV411 colour format. The Y channel is shown as red, the U channel as green and the V channel as blue. The color of the robots appears bright in YUV411 color format. After that, the foreground pixels are classified using the pixel classification algorithm described in sections 4.3 and 4.4. Next, the image processing methods, eroding and active contours, described in section 4.5 are applied to the images. The output of this module is the set of active contour(s) of the human(s).

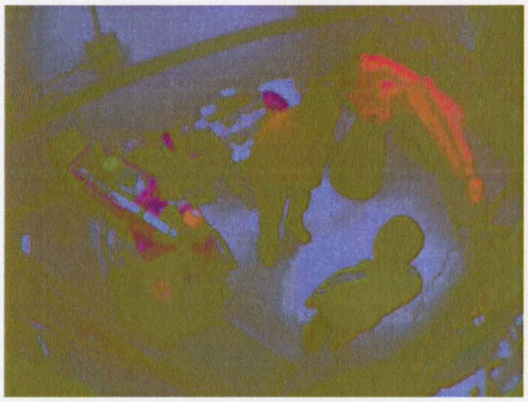
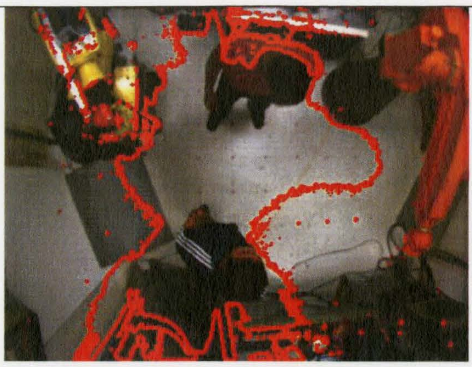


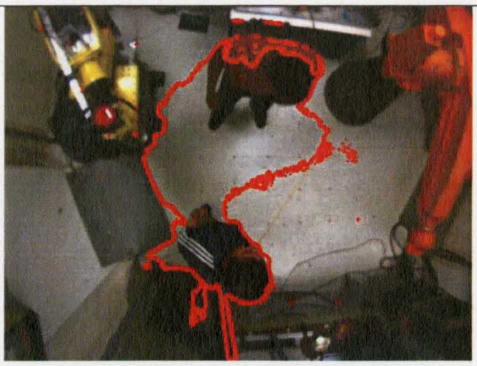
Figure 6.3: YUV411 image from the second camera

6.2.2.1 Sample Outputs of the Image Processing Module

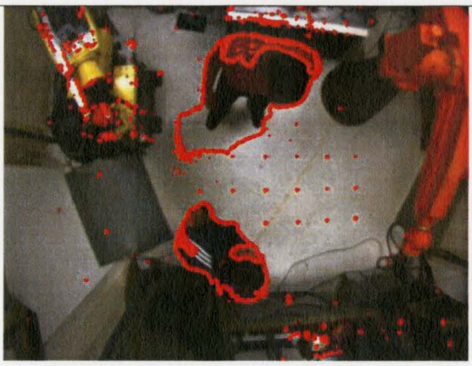
The image processing module extracts the features that are processed to construct the polyhedrons. Each polyhedron constructed by the human tracking system is only as good as the features extracted from the images. Pixels in the image are classified after acquiring the images and transforming them to YUV411 color format. The pixel classification is based on the pixel classification algorithm. The effects of the parameters introduced in the pixel classification algorithm (section 4.3) are illustrated in Figure 6.4. These parameters are used to compensate for fluctuations in lighting and noise. Figure 6.4(a) shows an image from the third (overhead) camera without applying the pixel classification algorithm. The red lines show the contours of the regions detected as foreground. This software has detected all the changes that have been introduced in the image by the two people. The people walking into the room caused reflections of light and shadows to fall onto objects in the workcell. Figure 6.4(b) shows the effect of activating the color error percentage Q_{Er} on the detected foreground object. Some of the noise points have been removed from this image, but the shadow between the two people was still incorrectly detected. Figure 6.4(c) shows the effect of activating the shadow tolerance percentage Y_{ShPct} and the shadow offset $Y_{ShOffset}$. This images shows that the shadows between the two people have been removed. However, noise points still exist in the image. Figure 6.4(d) shows the effect of the eroding the image. The size of the detected contours has become smaller, but shadows still exist in the image. Figure 6.4(e) shows the effect of activating the minimum size of the contour for noise removal. The noise points have been removed from the image without altering the size of the contour, but the shadows are still detected. Activating all of these parameters leads to active contours correctly enclosing each person separately as shown in Figure 6.4(f).



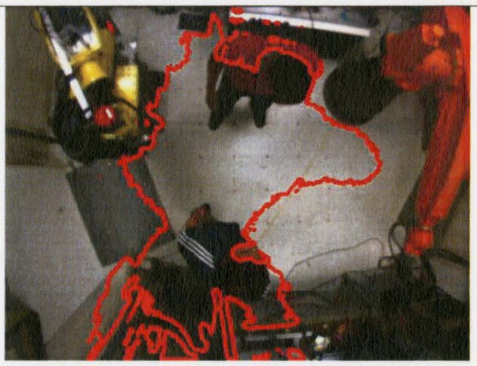
(a)



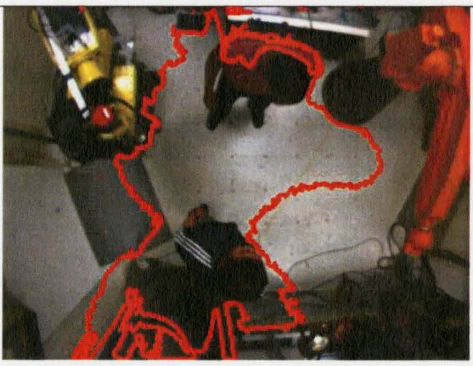
(b)



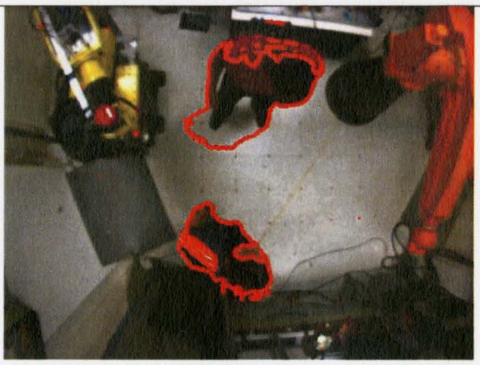
(c)



(d)



(e)



(f)

Figure 6.4: Effect of the parameters of the image processing module on its output

(a)No parameters active,(b) Colour percentage active, (c) Shadow percentage and offset active

(d)Eroding active, (e)Minimum contour size active, and (f) All parameters active

It should be noted that large changes in lighting conditions of the environment will cause the tracking system to fail. For example, switching the lights off will cause the system to conclude that the whole image is a foreground object. This will cause the system to fail.

6.2.3 Polyhedron Construction Software Module

The active contour(s) obtained from the image processing module enter the polyhedron construction module. Figure 6.5 shows a flowchart of this module. These contours are grouped using one of the two methods described in section 5.1: *MBR* or convex hull.

The next step depends on the method chosen for the grouping of contours. If the contours were grouped using the *MBR*, then the half spaces are constructed as described in section 5.3.1. On the other hand, if the contours were grouped using the convex hull, the program groups the different people using the k-means clusters pattern recognition algorithm (section 5.2.2). This method groups the people based on the predefined number of clusters k. Subsequently, half spaces are constructed for each cluster as described in section 5.3.2.

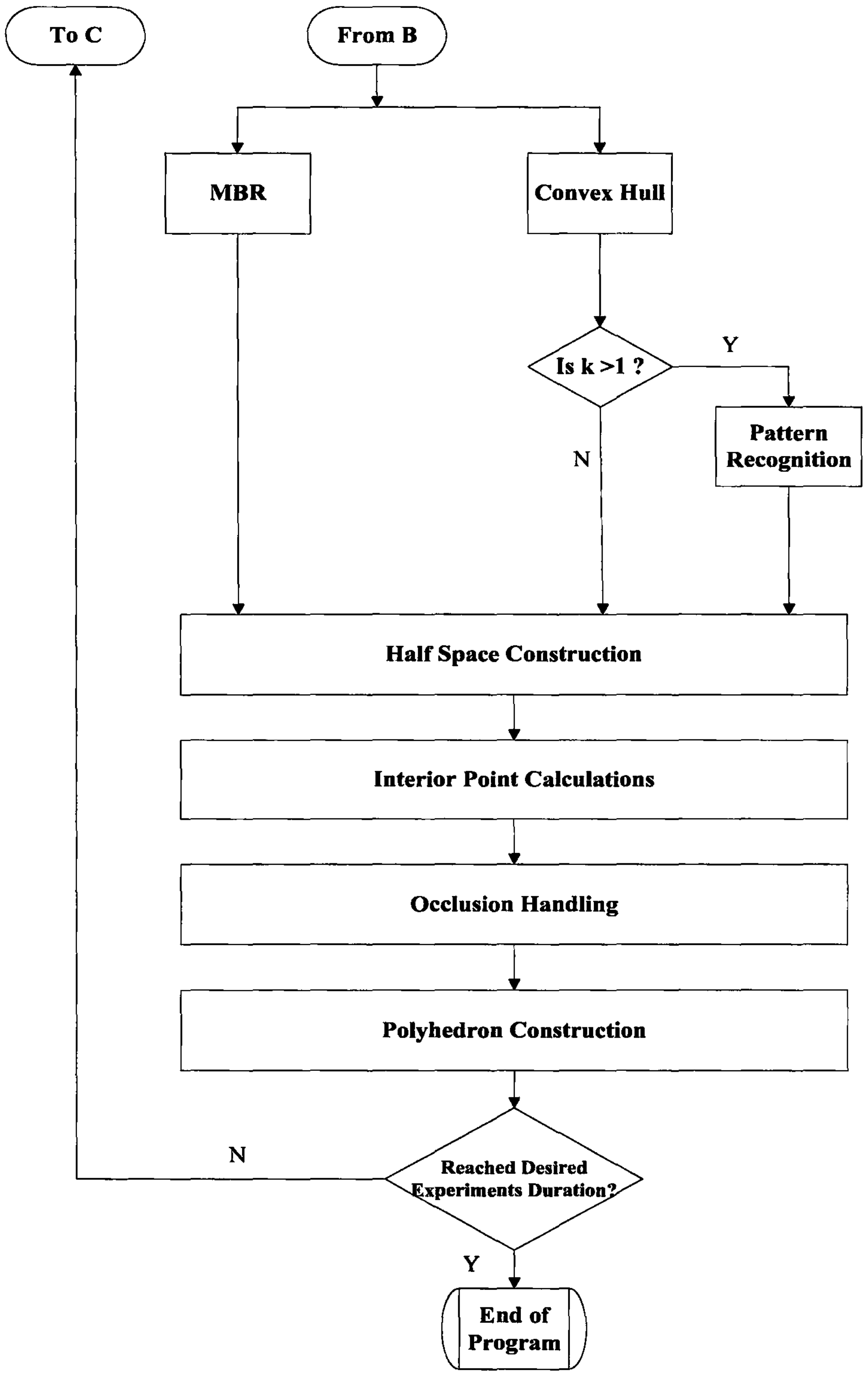


Figure 6.5: Flowchart of the polyhedron construction module

The interior point that lies inside the polyhedron is constructed from all the half spaces as described in section 5.4. Then, the system enters the occlusion handling method explained in section 5.5. This method finds the correct polyhedron if an occlusion has occurred. Finally, a polyhedron enclosing the person is constructed. The image processing and the polyhedron construction modules are executed sequentially until the desired experiment duration is reached.

6.3 Experiment 1

In this experiment four cameras are placed on the same plane at the top of the workcell. The cameras are located at C1-C4 as described in section 3.2. These cameras face towards the inside of the workcell. First, the system is initialized in the pre-processing module. The cameras are calibrated using a coplanar grid and Tsai's [7] calibration algorithm. The maximum calibration error was 19mm. Then, a 400 frame video sequence from the view of each camera is recorded. The length of this video sequence is sufficient enough to view as much of the background as possible while the robot is moving. The robot is moved in circular path to reveal the areas normally occluded by the robot. Next, the system executed the image processing and polyhedron construction modules untill the desired number of frames is reached.

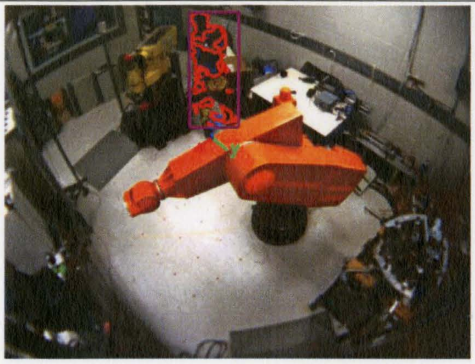
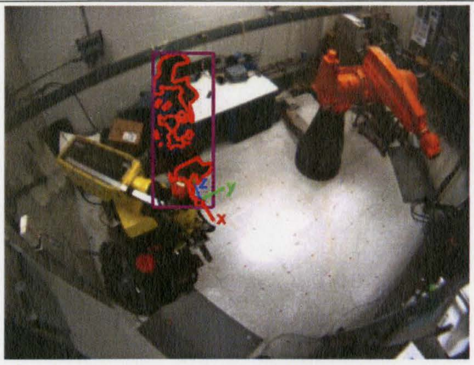
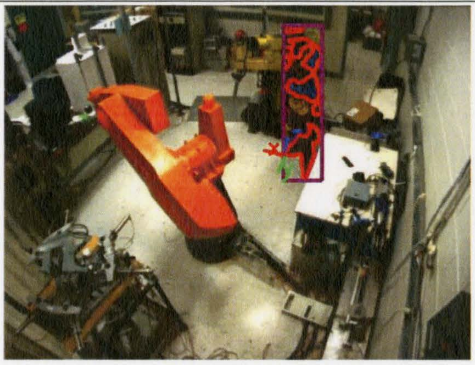
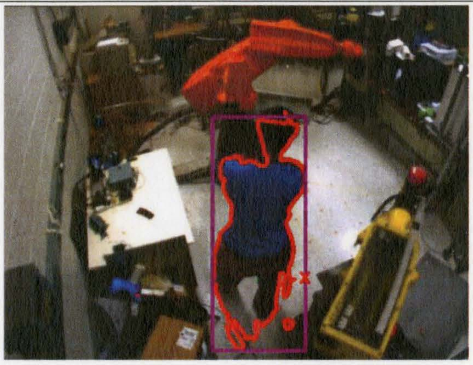
The system has been tested on a single person and then multiple people. Results of the system were obtained for the case of tracking using the *MBR* and the Convex Hull methods. The desired number of frames to be taken was set arbitrarily to a sequence of 250 frames. The system stopped when the desired number of images was reached.

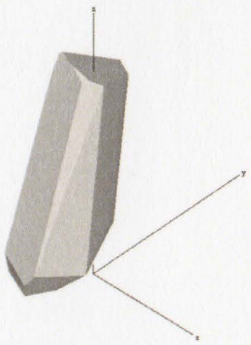
6.3.1 Single person tracking experiment

Two methods have been tested for tracking a single person inside the workcell: *MBR* and *Convex hull.*

6.3.1.1 Minimum Bounding Rectangle

This experiment focuses on building an enclosing polyhedron for a single person moving in the work area using the *MBR*. When the person walks into the area, the system starts detecting them and builds an enclosing polyhedron. Figure 6.6 shows an example set of images taken by all four cameras and the detection results. The red lines represent the detected foreground contours and the purple rectangle represents the *MBR* of the foreground objects. The moving robot was excluded by the system and therefore was not detected as a foreground object. Figure 6.6(e) shows the constructed polyhedron. Note that the top of the person's head was misclassified as shadow so that polyhedron is slightly smaller than it should be.

 (a)	 (b)
 (c)	 (d)



(e)

Figure 6.6: Experimental results for Experiment 1, Single Person, and MBR

(a) Camera 1, (b) Camera 2, (c) Camera 3, (d) Camera 4, and (e) Enclosing polyhedron

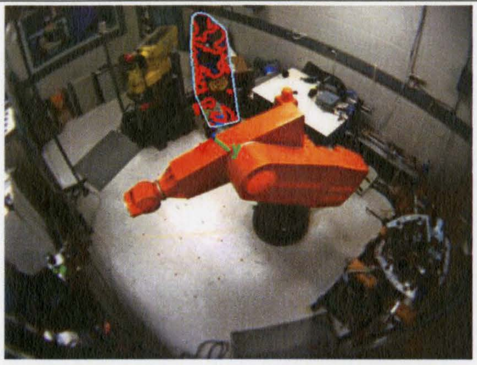
This experiment ran online at an average frame rate of 11.2 Hz. Table 6.1 lists a breakdown of the processing time in milliseconds. It should be noted that the minimum time for the contour building and polyhedron construction is zero when there is no human detected in the image.

Table 6.1: Processing Time for Experiment 1, Single Person, and MBR

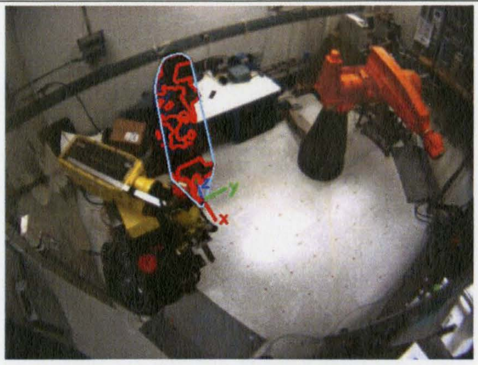
Procedure	**Time (ms)**		
	Minimum	**Maximum**	**Average**
Image acquisition, processing and YUV411 colour transformation	29	36	32
Pixel classification	39	47	42
Contour building	<1	1	1
Polyhedron construction	<1	16	14
Total			**89**

6.3.1.2 Convex Hull

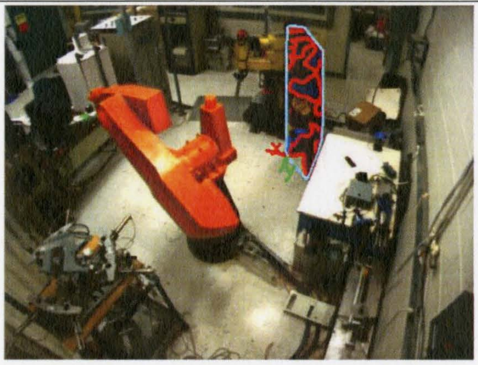
This experiment focuses on building an enclosing polyhedron for a single person moving in the work area using the convex hull method. The number of clusters k is set to 1 for all the cameras, since we are only tracking a single person. Figure 6.7 shows one example set of images taken by all four cameras and the detection results for the same frame as shown in Figure 6.6. The red lines represent the detected foreground contours and the cyan lines represent the convex hull of the foreground objects. Figure 6.7(e) shows the constructed polyhedron. This polyhedron encloses a smaller volume and is smoother than the polyhedron constructed by the *MBR*.



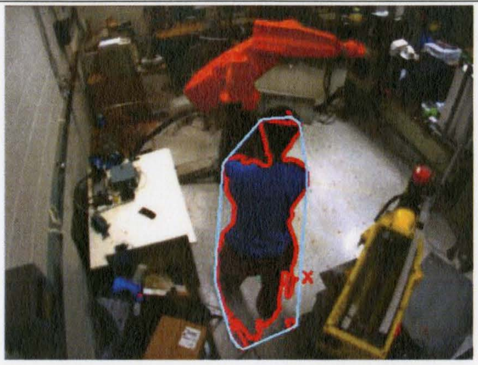
(a)



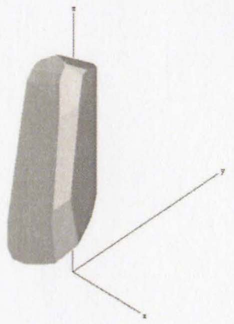
(b)



(c)



(d)



(e)

Figure 6.7: Experimental results for Experiment 1, Single Person, and Convex hull

(a) Camera 1, (b) Camera 2, (c) Camera 3, (d) Camera 4, and (e) Enclosing polyhedron

This experiment ran online at an average frame rate of 10.5Hz. Table 6.2 lists a breakdown of the processing time in milliseconds. Building the convex hulls requires more points and therefore more calculations time was needed.

Table 6.2: Processing Time for Experiment 1, Single Person, and Convex hull

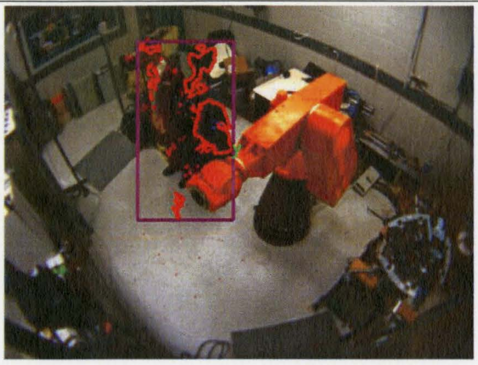
Procedure	**Time (ms)**		
	Minimum	**Maximum**	**Average**
Image acquisition, processing and YUV411 colour transformation	29	36	32
Pixel classification	39	47	42
Contour building	<1	4	2
Polyhedron construction	<1	23	19
Total			**95**

6.3.2 Multiple people tracking experiment

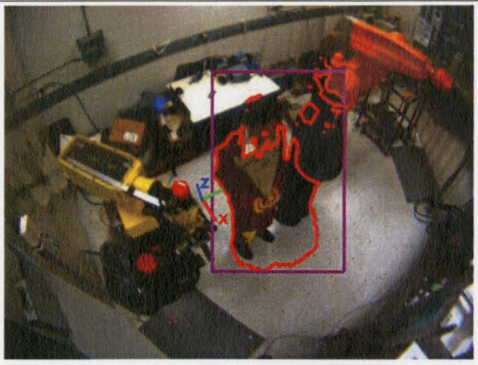
Two methods have been tested for tracking two people inside the workcell: *MBR* and *Convex hull.*

6.3.2.1 Minimum Bounding Rectangle

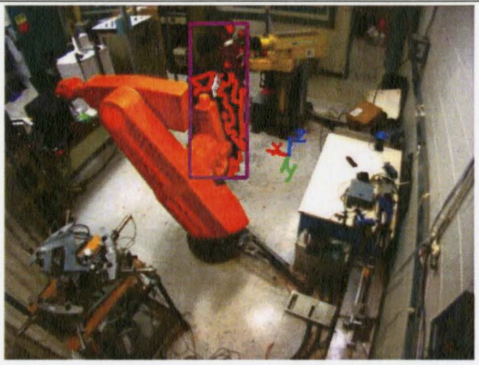
In this experiment, the system's performance is tested with multiple people in the work area. Two people walk in the area adjacent to the moving robot. It can be observed that the system is robust to different clothing and hidden limbs. One person has a piece of cloth obscuring his legs and the other person is hiding his arms inside a maroon jump suit and tracking remains unaffected. Figure 6.8 shows the images of the person taken by all four cameras and the detection results. The red lines represent the detected foreground contours and the cyan lines represent the convex hull of the foreground objects. Figure 6.8(e) shows the constructed polyhedron. This polyhedron encloses the multiple humans which results in a large polyhedron volume. This is not ideal but it is acceptable for some collision avoidance applications to have an overestimation of the volume to provide a clearance that increases the human's safety.



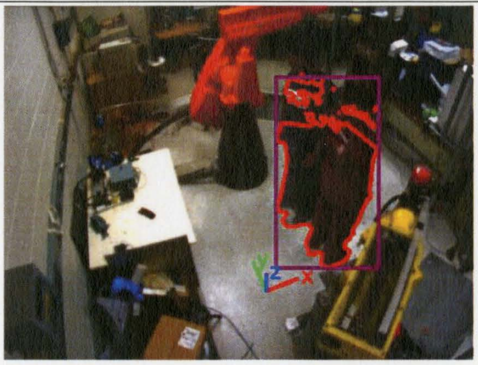
(a)



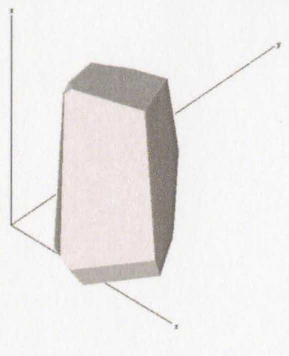
(b)



(c)



(d)



(e)

Figure 6.8: Experimental results for Experiment 1, Multiple people, and MBR

(a) Camera 1, (b) Camera 2, (c) Camera 3, (d) Camera 4, and (e) Enclosing polyhedron

This experiment ran online at an average frame rate of 11.2 Hz. Table 6.3 lists a breakdown of the processing time in milliseconds.

Table 6.3: Processing Time for Experiment 1, Multiple people, and MBR

Procedure	**Time (ms)**		
	Minimum	**Maximum**	**Average**
Image acquisition, processing and YUV411 colour transformation	29	36	32
Pixel classification	39	47	42
Contour building	<1	1	1
Polyhedron construction	<1	16	14
Total			**89**

6.3.2.2 Convex Hull

This experiment focuses on building an enclosing polyhedron for multiple people moving in the work area using the convex hull method. The number of clusters k is set to 1 for all the cameras. This is because having multiple clusters from each camera was found to cause two problems.

The first problem was that grouping of the contours with k>1 resulted in false grouping of the contours with camera locations C1 to C4. This is because the k-means clustering algorithm has no inherent knowledge about the shape of a person as explained in section 5.2.2.

The second problem is the correspondence problem with $k>1$. Let p equal the number of people in the area and let C be the number of cameras in the area. For example, given two cameras and two people in the area each camera will produce two clusters. This gives a combination of p^C or four possible polyhedrons. Two of the four polyhedrons are the correct polyhedrons that correspond to the two people. The remaining polyhedrons must be ruled out as polyhedrons corresponding to a person. But since polyhedrons are constructed from the intersection of two pyramids given by the cameras, there might be a case where two pyramids intersect to construct a "pseudo-polyhedron" that does not contain a person. The pseudo-polyhedron case is illustrated from the top view of the workcell in Figure 6.9. C1 and C2 are the camera locations; the blue and orange lines represent the pyramid given by the first camera and the yellow and green lines represent the pyramids given by the second camera; the cyan shaded areas represent the constructed polyhedron enclosing the human; the read area is the pseudo-polyhedron. Four cameras (C1 to C4) and two people in the area will result in 16 polyhedrons. Two of these will correspond to the two humans and the remainders will be pseudo-polyhedrons. This could not be solved with camera locations C1 to C4.

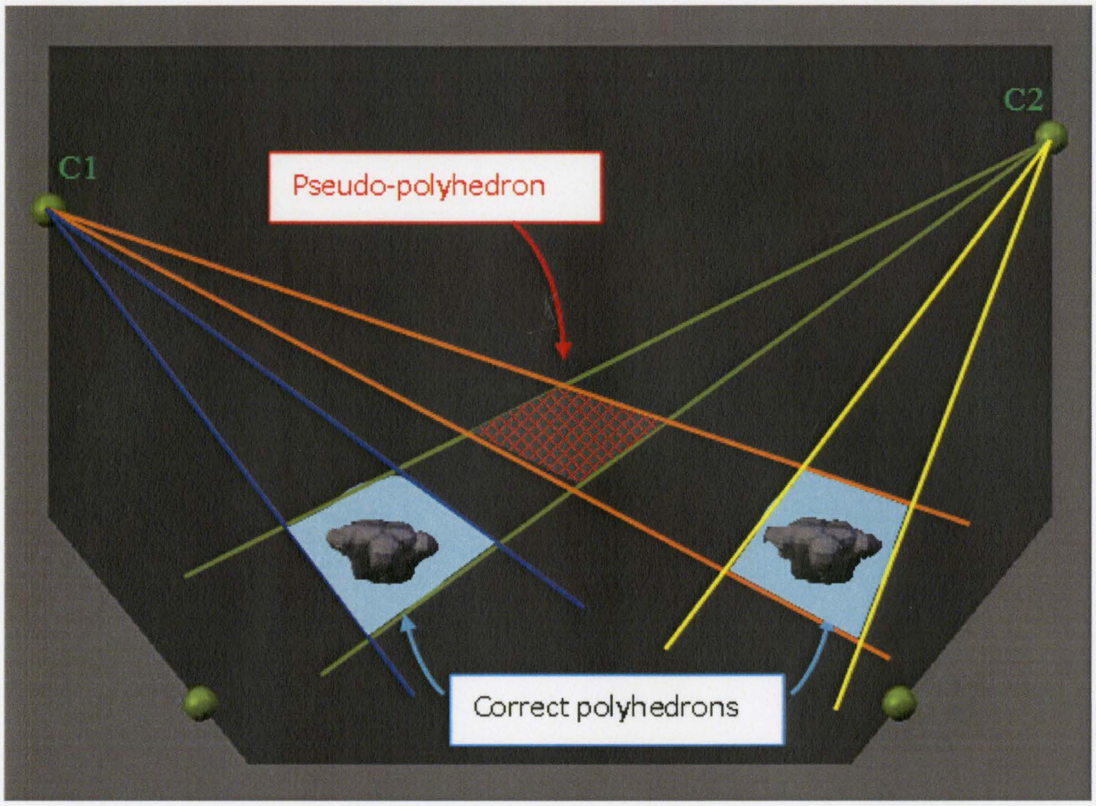
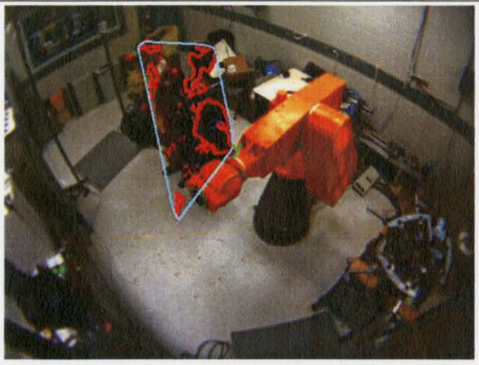
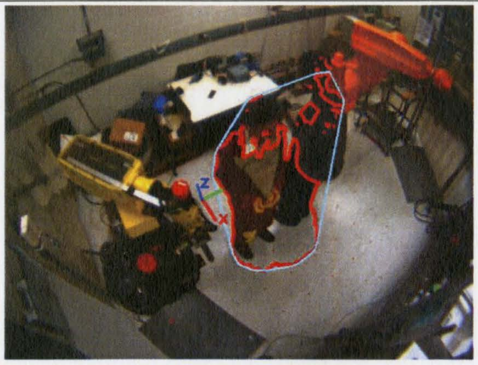
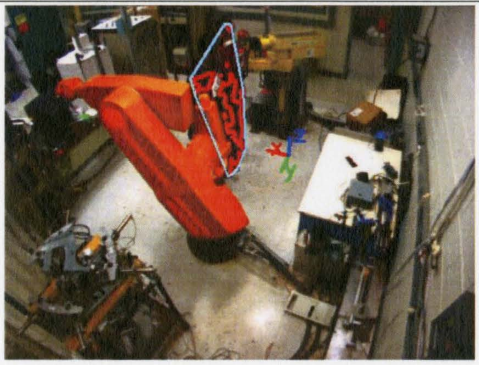


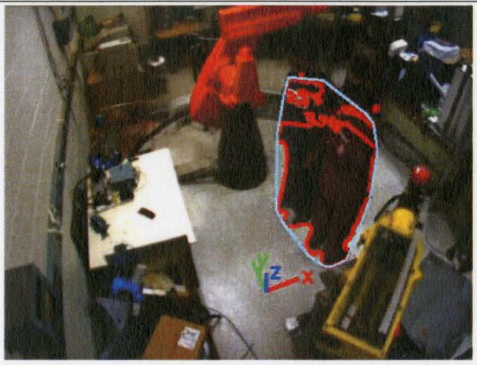
Figure 6.9: Top view example of a Pseudo-polyhedron

Figure 6.10 shows the images of the person taken by all four cameras and the detection results. The red lines represent the detected foreground contours and the cyan lines represent the convex hull of the foreground objects. Figure 6.10(e) shows the constructed polyhedron.


(a)


(b)


(c)


(d)

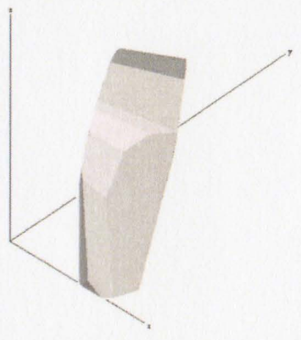

(e)

Figure 6.10: Experimental results for Experiment 1, Multiple people, and Convex hull

(a) Camera 1, (b) Camera 2, (c) Camera 3, (d) Camera 4, and (e) Enclosing polyhedron

This experiment ran online at an average frame rate of 10.5Hz. Table 6.4 lists a breakdown of the processing time in milliseconds.

Table 6.4: Processing Time: Experiment 1, Multiple people, and Convex hull

Procedure	**Time (ms)**		
	Minimum	**Maximum**	**Average**
Image acquisition, processing and YUV411 colour transformation	29	36	32
Pixel classification	39	47	42
Contour building	<1	4	2
Polyhedron construction	<1	23	19
Total			**95**

6.4 Experiment 2: Overhead camera

In this experiment four cameras are placed around the workcell. The cameras are located at C1, C2, C3, and C5 as described in section 3.2. These cameras face towards the inside of the workcell with the camera located as C5 facing directly downwards towards the ground plane. Location C5 was deliberately chosen as a way to solve the grouping and correspondence problems that occurred previously with multiple people tracking. This will be explained further in section 6.4.2.2. As before, the system is first initialized in the pre-processing module. The cameras are calibrated using a non-coplanar grid and Tsai's [7] calibration algorithm. The maximum calibration error was 11mm. This non-coplanar grid is required to calibrate the overhead camera due to the lack of depth information as explained in section 3.6.1. After calibration the software was run in the same manner as Experiment 1.

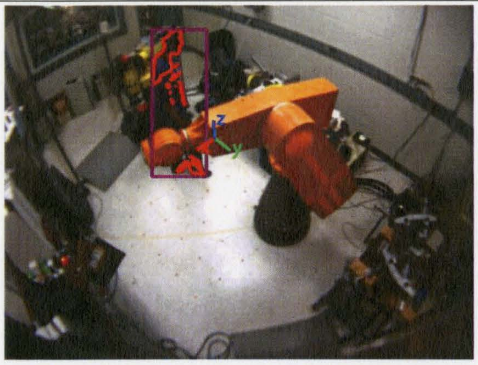
The system has been tested on a single person and then multiple people. Results were obtained for the case of tracking using the *MBR* and the Convex Hull methods. The desired number of frames to be taken was again set to 250 frames.

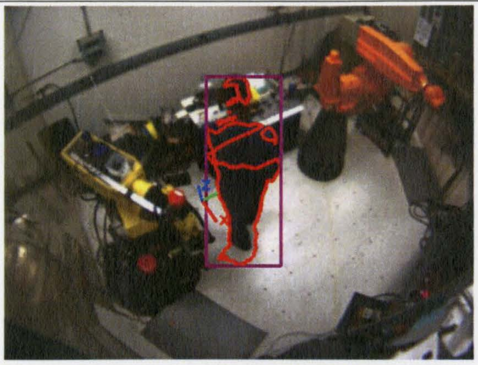
6.4.1 Single person tracking experiment

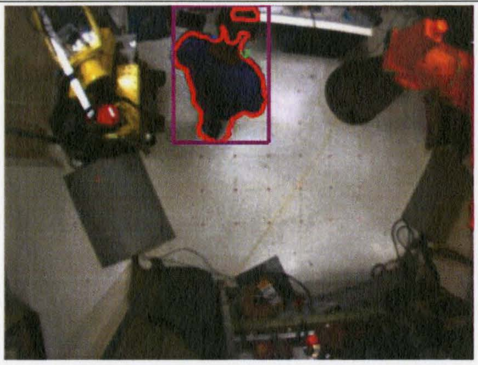
Two methods have been tested for tracking a single person inside the workcell: *MBR* and *Convex hull.*

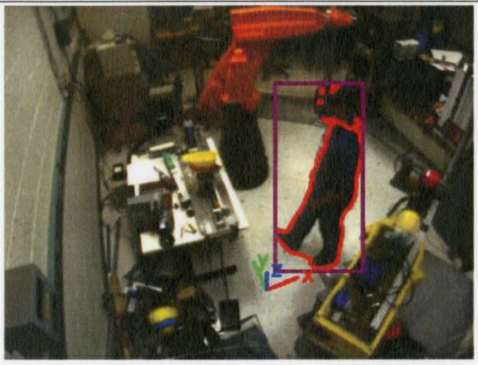
6.4.1.1 Minimum Bounding Rectangle

This experiment focuses on building an enclosing polyhedron for a single person moving in the work area. When the person walks into the area, the system starts detecting them and builds an enclosing polyhedron. Figure 6.11 shows an example set of images taken by all four cameras and the detection results. The red lines represent the detected foreground contours and the purple rectangle represents the *MBR* of the foreground objects. The system also deals well with partial occlusion, as can be seen in Figure 6.11(a) where the person's foot was sufficient to build an *MBR* that includes the occluded region. The moving robot was excluded by the system and therefore was not detected as a foreground object. Figure 6.11(e) shows the constructed polyhedron.


(a)


(b)


(c)


(d)

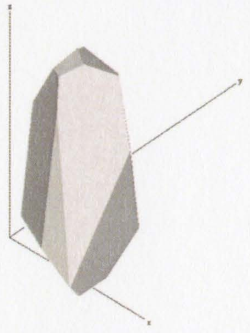

(e)

Figure 6.11: Experimental results for Experiment 2, Single Person, and MBR

(a) Camera 1, (b) Camera 2, (c) Camera 3, (d) Camera 4, and (e) Enclosing polyhedron

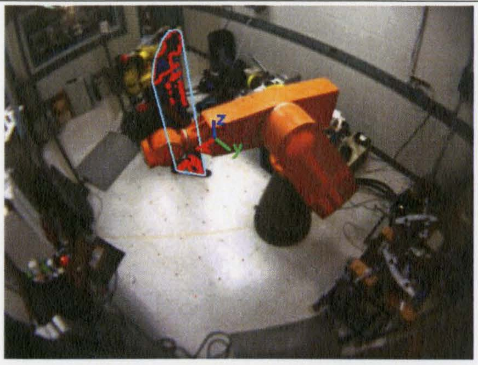
This experiment ran online at an average frame rate of 11.2Hz. Table 6.5 lists a breakdown of the processing time in milliseconds.

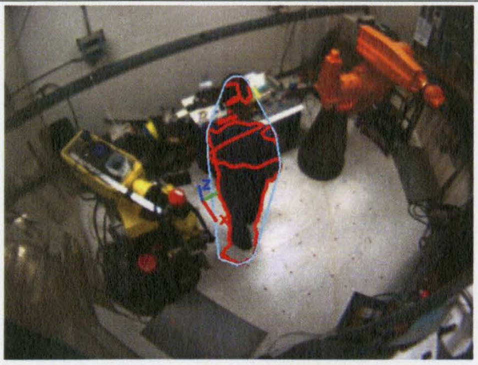
Table 6.5: Processing Time for Experiment 2, Single Person, and MBR

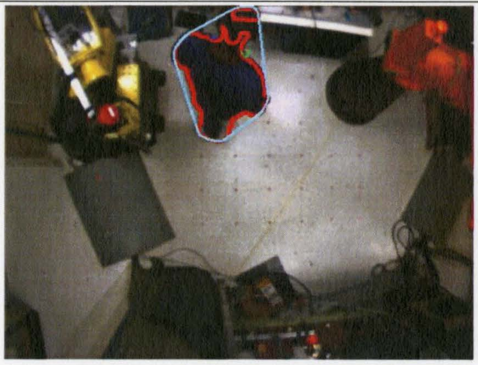
Procedure	**Time (ms)**		
	Minimum	**Maximum**	**Average**
Image acquisition, processing and YUV411 colour transformation	29	36	32
Pixel classification	39	47	42
Contour building	<1	1	1
Polyhedron construction	<1	16	14
Total			**89**

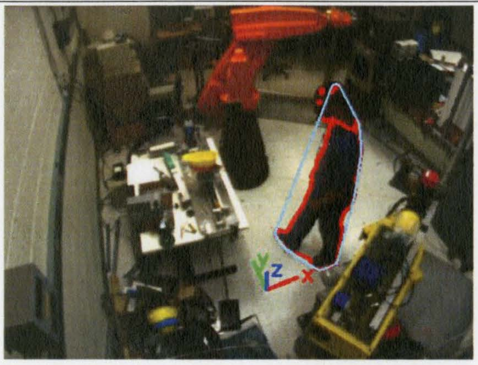
6.4.1.2 Convex Hull

This experiment focuses on building an enclosing polyhedron for a single person moving in the work area using the convex hull method. The number of clusters *k* is set to 1 for all the cameras, since we are only tracking a single person. Figure 6.12 shows an example set of images taken by all four cameras and the detection results for the same frame as in Figure 6.11. The red lines represent the detected foreground contours and the cyan lines represent the convex hull of the foreground objects. Figure 6.12 (e) shows the constructed polyhedron. As before, this polyhedron is a much closer approximation to the person's shape than are the obtained using the *MBR*.


(a)


(b)


(c)


(d)

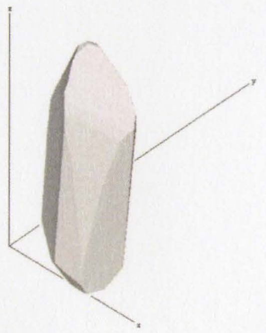

(e)

Figure 6.12: Experimental results for Experiment 2, Single person, and Convex hull

(a) Camera 1, (b) Camera 2, (c) Camera 3, (d) Camera 4, and (e) Enclosing polyhedron

This experiment ran online at an average frame rate of 10.5Hz. Table 6.6 lists a breakdown of the processing time in milliseconds.

Table 6.6: Processing Time for Experiment 2, Single person, and Convex hull

Procedure	**Time (ms)**		
	Minimum	**Maximum**	**Average**
Image acquisition, processing and YUV411 colour transformation	29	36	32
Pixel classification	39	47	42
Contour building	<1	4	2
Polyhedron construction	<1	23	19
Total			**95**

6.4.2 Multiple people tracking experiment

Two methods have been tested for tracking two people inside the workcell: *MBR* and *Convex hull.*

6.4.2.1 Minimum Bounding Rectangle

In this experiment, the system's performance is tested with multiple people in the work area. Two people walk in the area adjacent to the moving robot. Figure 6.13 shows an example set of images taken by all four cameras and the detection results. The red lines represent the detected foreground contours and the purple lines represent the *MBR* of the foreground objects. Figure 6.13(e) shows the constructed polyhedron. This polyhedron encloses the multiple humans which results in a large polyhedron volume. As before, this is not ideal but it is acceptable for some collision avoidance applications to have an overestimation of the volume to provide a clearance that increases the human's safety.

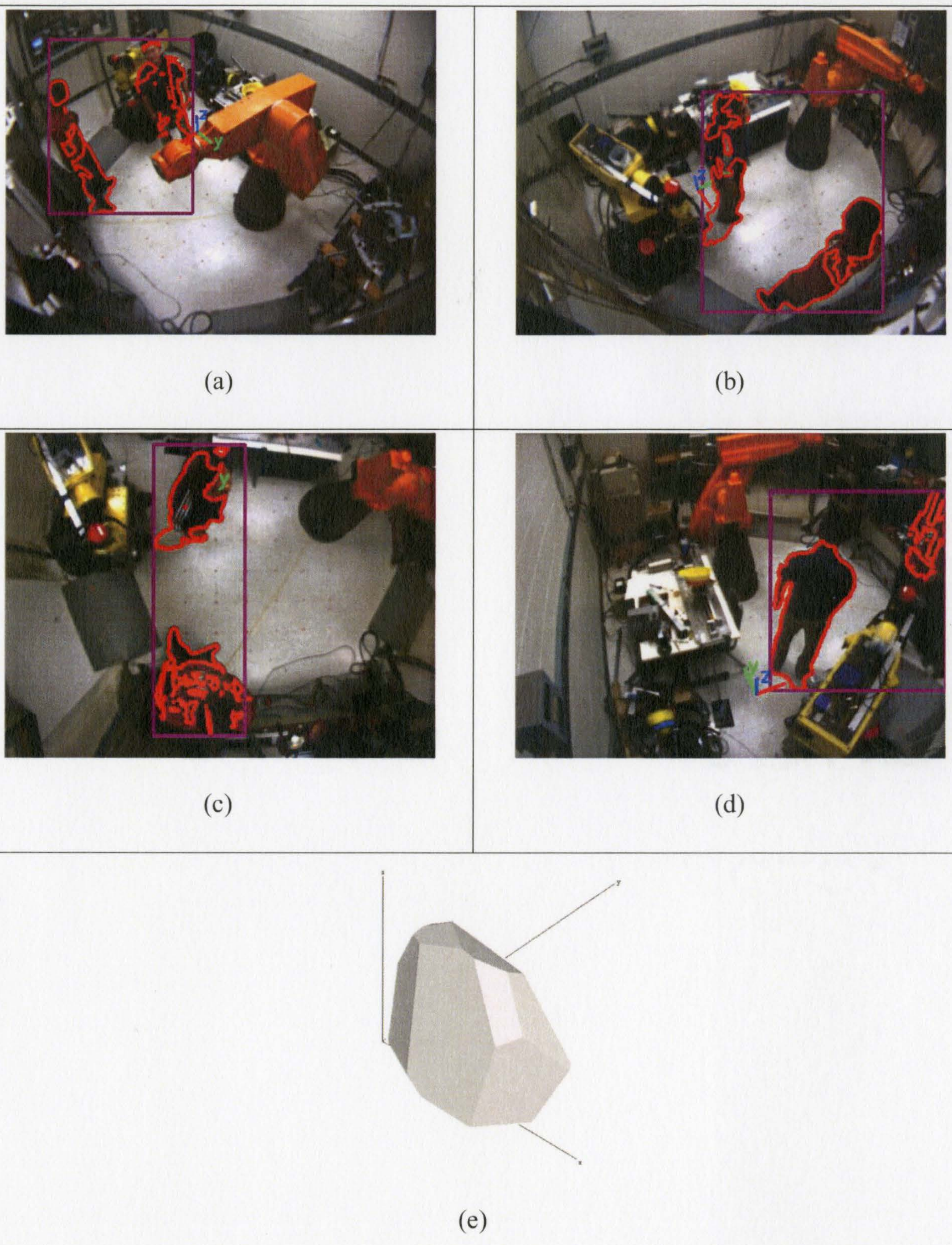


Figure 6.13: Experimental results for Experiment 2, Multiple People, and MBR

(a) Camera 1, (b) Camera 2, (c) Camera 3, (d) Camera 4, and (e) Enclosing polyhedron

This experiment ran online at an average frame rate of 11.2 Hz. Table 6.7 lists a breakdown of the processing time in milliseconds.

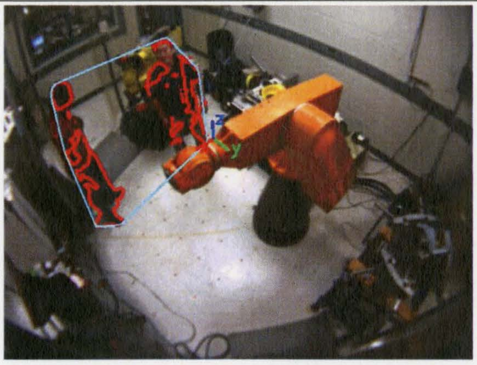
Table 6.7: Processing Time for Experiment 2, Single Person, and MBR

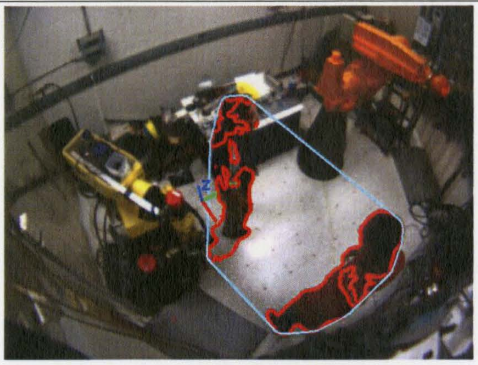
Procedure	**Time (ms)**		
	Minimum	**Maximum**	**Average**
Image acquisition, processing and YUV411 colour transformation	29	36	32
Pixel classification	39	47	42
Contour building	<1	1	1
Polyhedron construction	<1	16	14
Total			**89**

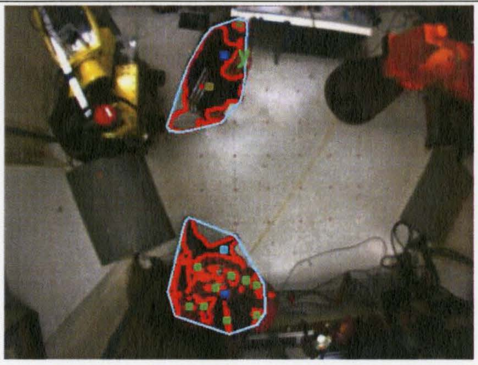
6.4.2.2 Convex Hull

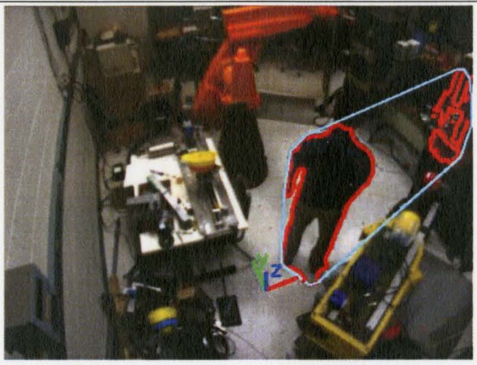
This experiment focuses on building a polyhedron for each human moving in the workcell using the convex hull method. To avoid the correspondence problem described in section 0, the number of clusters *k* for the overhead camera will be set equal to two. The number of clusters *k* for all other cameras will be set to one. This arrangement will allow the over head camera to specify the region occupied by each person. The intersection of the pyramids from all the cameras will then result in individual polyhedrons for each person in the workcell.

Figure 6.14 shows the images of the people taken by all four cameras and the detection results. The red lines represent the detected foreground contours and the cyan lines represent the convex hull of the foreground objects. Figure 6.14(c) shows the convex hull of each person detected by the overhead camera (C5). Figure 6.14(e) shows the constructed polyhedrons for each person.


(a)


(b)


(c)


(d)

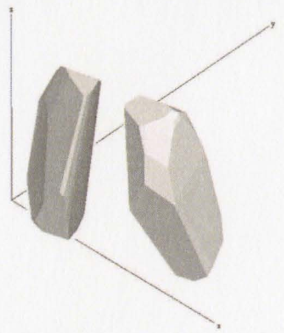

(e)

Figure 6.14: Experimental results for Experiment 2, Multiple People, and Convex hull

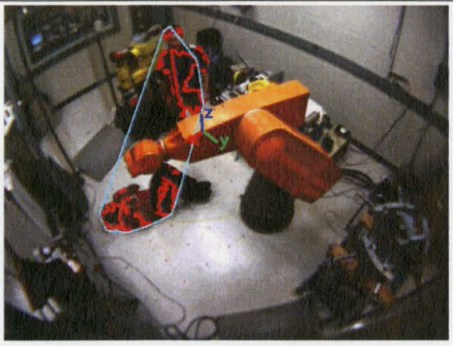
(a) Camera 1, (b) Camera 2, (c) Camera 3, (d) Camera 4, and (e) Enclosing polyhedrons

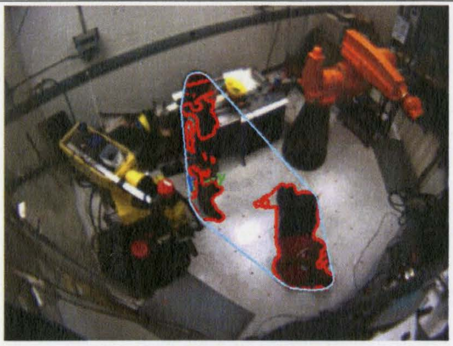
This experiment ran online at an average frame rate of 10.2 Hz. Table 6.8 lists a breakdown of the processing time in milliseconds. The system required a longer time to process two polyhedrons.

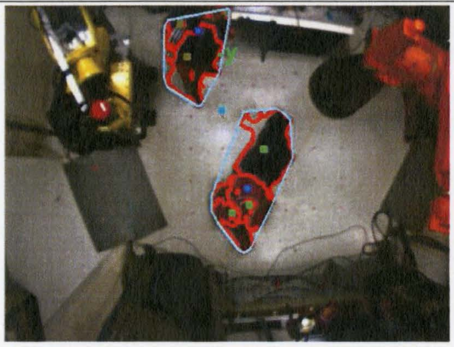
Table 6.8: Processing Time for Experiment 2, Multiple People, and Convex hull

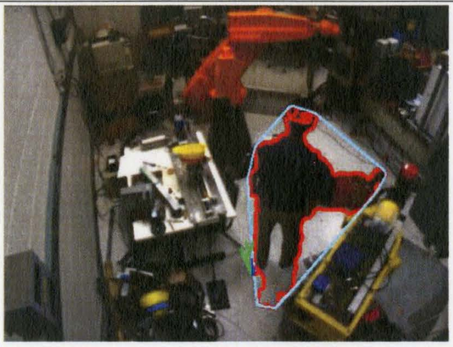
Procedure	**Time (ms)**		
	Minimum	**Maximum**	**Average**
Image acquisition, processing and YUV411 colour transformation	29	36	32
Pixel classification	39	47	42
Contour building	<1	4	2
Polyhedron construction	<1	26	22
Total			**98**

The drawback of using this method is that the height of constructed polyhedrons is dependent on the height of the tallest person in the image. This is because only the overhead camera separates the polyhedrons while the remaining cameras construct a large polyhedron that encloses all the people. For example, if two people enter the workcell, one person was standing and the other person was lying down on the floor, then the constructed polyhedrons will appear as if the two people are standing. This situation is shown in Figure 6.15. The constructed polyhedrons are shown in Figure 6.15(e), (f). The overestimation of the human height is acceptable for most collision avoidance systems.


(a)


(b)


(c)


(d)

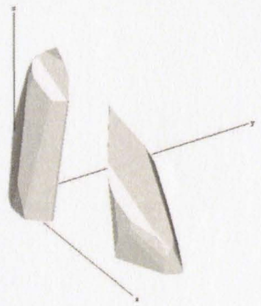

(e)

Figure 6.15: Experimental results for Experiment 2, Multiple People (Standing and Laying on the floor), and Convex hull

(a) Camera 1, (b) Camera 2, (c) Camera 3, (d) Camera 4, and (e) Enclosing polyhedrons

Another problem that occurs is misrelating the contours. In Figure 6.16 the system assigns the legs and feet of the person on the left to belong to the person on the right. The torso of the human on the left has similar color to the background behind it. Therefore the pixel classification algorithm classified this section of the torso to be part of the background. The k-means clustering algorithm has no inherent knowledge about the shape of a person. Thus, it cannot correlate the contours of the person to the correct human dynamically. This problem would not have occurred if the sensitivity of the cameras to colors was higher. Dark colours do not reflect enough light to be detected by the camera. Therefore, it is difficult for the camera to distinguish the difference in colour between dark foregrounds and dark backgrounds. This is true for all of the experiments conducted.

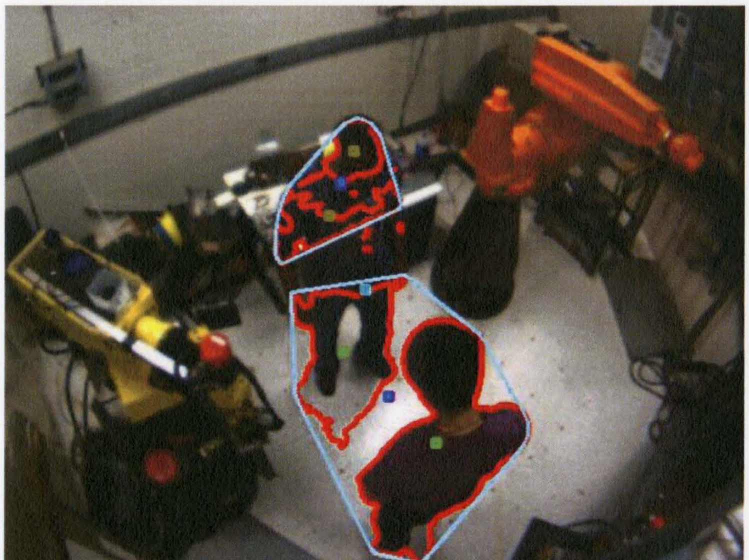
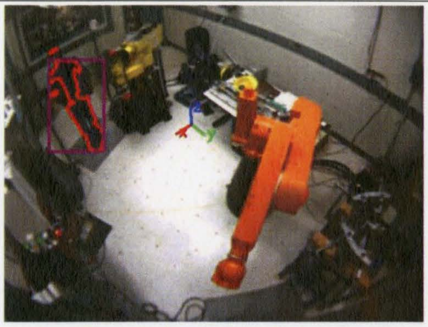


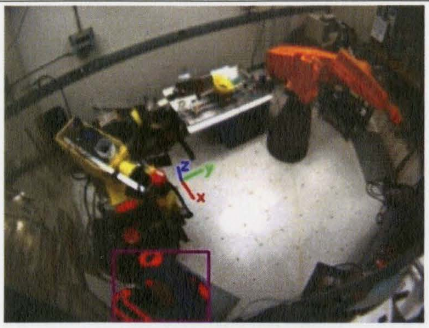
Figure 6.16: Misrelating the contours using k-means clustering

6.5 Occlusion handling algorithm

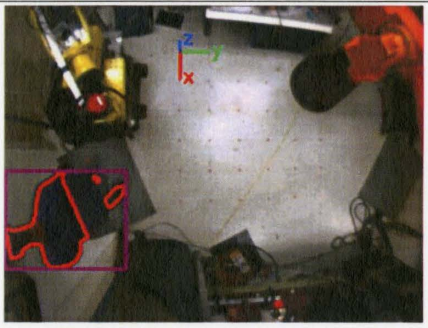
The constructed polyhedrons are based on the intersection of the pyramids from each camera. There are some instances where the human is partially occluded in the view of the camera. Figure 6.17 demonstrates the effect of the occlusion handling algorithm when the top half of the person is occluded. This occurred when the human was about to exit the workcell. This caused him to be outside the view of the fourth camera as can be seen in Figure 6.17(d). The other cameras were still able to see the human. The fourth camera was only able to detect the feet of the human. The red lines represent the detected foreground contours and the purple lines represent the *MBR*. Figure 6.17(e) shows the constructed polyhedron if the occlusion handling algorithm was not implemented. This polyhedron is the intersection of the four pyramids from the four cameras. Valuable information given by three cameras has been lost due to the inclusion of the fourth camera. The occlusion handling algorithm was able to ignore the results given by the fourth camera and construct the polyhedron shown in Figure 6.17(f). The elimination of the fourth camera was based on the comparison of the height of polyhedron given by the combination of the four cameras. It was determined that including the results from the fourth camera will cause the constructed polyhedron to be of a height less than the normal human height which was set to 1.2m (average human height is 1.5-1.8m [52]).



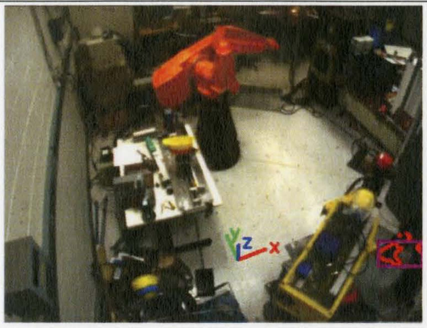
(a)



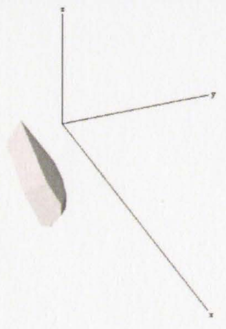
(b)



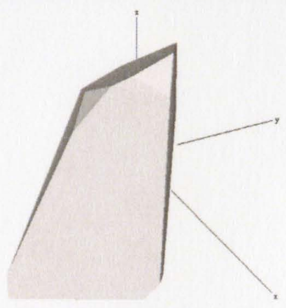
(c)



(d)



(e)



(f)

Figure 6.17: Results of the occlusion handling algorithm for occluded tops

(a) Camera 1, (b) Camera 2, (c) Camera 3, (d) Camera 4,

(e) Constructed polyhedron without occlusion handling algorithm, and

(f) Constructed polyhedron with occlusion handling algorithm

The second type of occlusion is the occlusion of the bottom half of the person. This results in a polyhedron floating in the air. This type of occlusion is shown in Figure 6.18. The red lines represent the detected foreground objects and the cyan lines represent the convex hull of the detected area. The human was occluded by the moving robot in the image from the first camera as can be seen in Figure 6.18(a). This causes the system to construct a polyhedron that is floating as shown in Figure 6.18(e). The occlusion handling algorithm handles this situation by extending the polyhedron to the floor level. This is achieved by projecting all points to the floor level and finding the convex hull of the original and projected points. The constructed polyhedron found using the occlusion handling is shown in Figure 6.18(f).

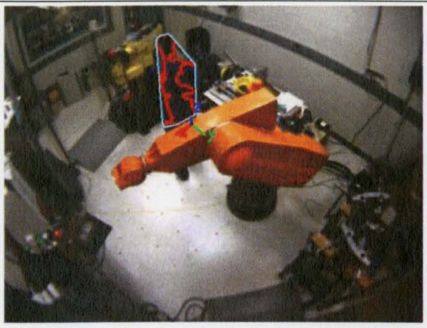
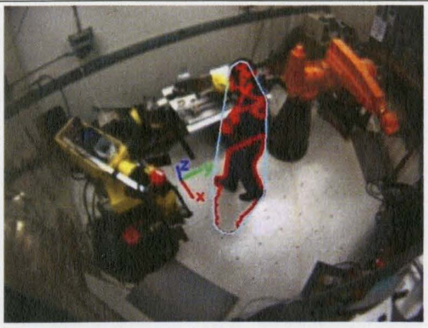
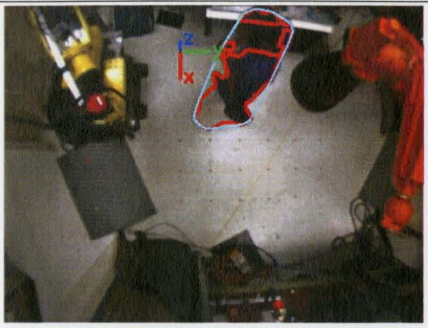
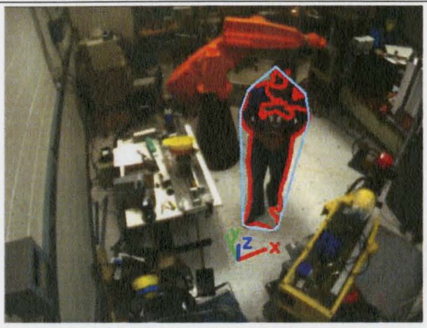
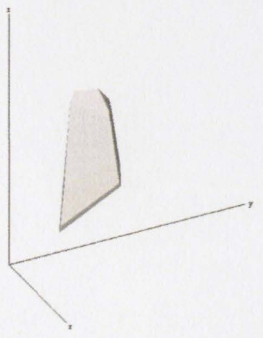
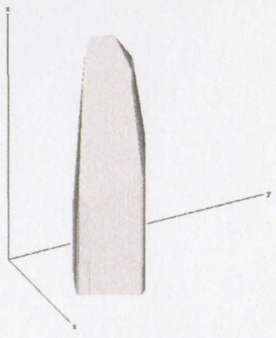
 (a)	 (b)
 (c)	 (d)
 (e)	 (f)

Figure 6.18: Results of the occlusion handling algorithm for occluded bottoms

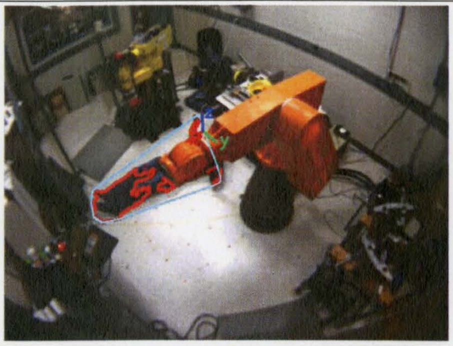
(a) Camera 1, (b) Camera 2, (c) Camera 3, (d) Camera 4,

(e) Constructed polyhedron without extending the polyhedron, and

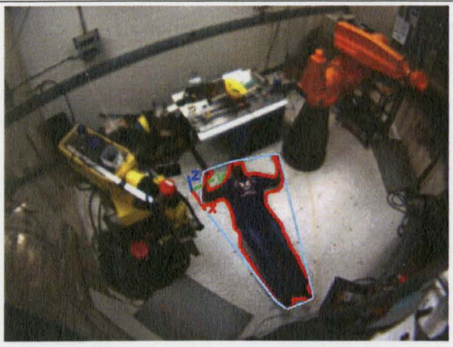
(f) Constructed polyhedron with extending the polyhedron

6.6 Different human poses

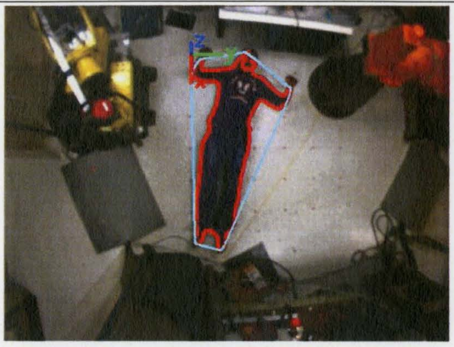
To show the system's robustness to different human postures, a person walks into the work area and lies down on the floor as shown in Figure 6.19(a). The occlusion handling algorithm correctly rules out occlusion as the cause for the low heights of the polyhedrons from the combination of the cameras, and generates the final enclosing polyhedron using the *MBR*s of all four cameras as shown Figure 6.19(e).



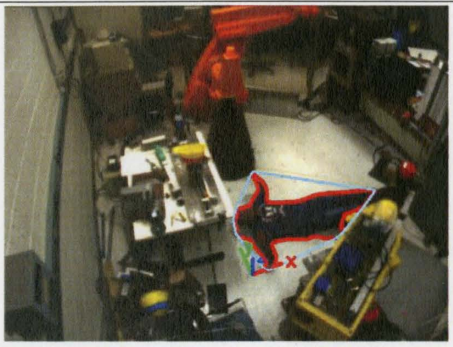
(a)



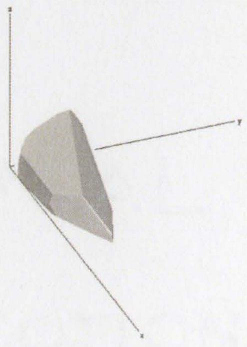
(b)



(c)



(d)



(e)

Figure 6.19: An example of a person lying down in the workcell

(a) Camera 1, (b) Camera 2, (c) Camera 3, (d) Camera 4, and (e) Constructed polyhedron

6.7 Conclusions

Two series of experiments were shown for two different arrangements of the cameras. The experiments were divided into single person tracking and multiple people tracking using the MBR and convex hull tracking methods.

In the experiments using the MBR for single and multiple people tracking, the system was able to reliably construct an enclosing polyhedron of all the human(s) in the workcell. This method performed at an average frame rate of 11.2Hz. This method builds a large polyhedron that encloses all the people in the workcell.

In the experiments using the convex hull with the number of clusters k set to 1, the system was able to reliably construct a polyhedron that encloses all the humans(s) in the workcell. This polyhedron encloses a smaller volume and is smoother than the polyhedron constructed by the *MBR*. Nevertheless, the constructed polyhedron was enclosing all the people in the workcell which is not ideal. This method performed at a frame rate of 10.4Hz due to the extra calculations for the convex hull.

To construct individual polyhedrons for each person in the workcell, the convex hull method was tested with the number of clusters k set to the number of people. These experiments were performed using two humans in the workcell. The case of tracking multiple people using the convex hull method in Experiment 1 results in the correspondence problem. This causes the construction of pseudo-polyhedrons. To solve this problem, one camera was moved to have an overhead view of the workcell in thee second series of experiments (Experiment 2). The number of clusters k in the overhead camera was set to two. The number of clusters *k* for all other cameras was set to one. This procedure allowed the overhead camera to specify the region occupied by each person. The intersection of the pyramids from all the cameras resulted in individual polyhedrons for each person in the workcell. The height of the constructed polyhedrons is dependent on the height of the tallest person in the workcell. This method performed at a frame rate of 10.2Hz. The speed reduction is due to the multiple constructed polyhedrons.

Most of the processing time of the system was spent in the pixel classification. This is due to the large number of pixels that need to be classified in the image from each camera. Smaller image size will result in a faster system but will produce a less accurate polyhedron. Increasing the number of cameras in the system will reduce the speed of the system but will produce a more accurate polyhedron.

Results of the occlusion handling algorithm were presented. The algorithm was able to detect and correct partial occlusions. The system's robustness to the different human poses was also presented.

7. Conclusions and Recommendations

7.1 Summary

In this thesis a markerless multiple-camera vision-based 3D human tracking method for industrial environments has been presented. The system represents the 3D space occupied by the human(s) using one or more convex polyhedron. The system started by calibrating the cameras and learning the background. Images are then acquired and processed to find foreground pixels that represent the human(s) using pixel classification. Pixel classification was performed using a novel algorithm. The foreground segmentation algorithm removes pixels that are related to the robot from the image. The effects of the parameters used in the pixel classification were demonstrated. The foreground pixels produced by the pixel classification are bounded using active contours. These active contours represent the area occupied by the human(s). The contours were grouped using one of two methods: minimum bounding rectangle and convex hull. Each camera built half-spaces of the active contours. These half spaces were combined to construct a polyhedron enclosing the human(s). Occlusion handling algorithms were employed to detect occlusions and correct the polyhedron. The system was able to track multiple humans in a workcell around a moving robot.

7.2 Achievements

The system provides a markerless tracking system for industrial environments. The system is comprised of: capturing a video stream using multiple cameras; processing the acquired video to automatically detect the human(s) in a workcell; and defining one or more polyhedron enclosing the human(s).

The system can track humans in the vicinity of moving robots without using skin color cues or articulated human models. It is robust to self-occlusions and to partial occlusions caused by the robot. The system requires at least two cameras for reliable tracking.

A novel pixel classification algorithm has been implemented to classify pixels as foreground pixels. These pixels represent the detected human(s).

The system can obtain the 3D space occupied by the person at a frame rate of 11.2Hz using four cameras, two cameras with an image size of 640 x 480 pixels and the other two cameras with an image size of 512 x 384 pixels.

7.3 Limitations

Although the multiple camera system can obtain the space occupied by the human(s) in the workcell, it still has some significant limitations. First, a special problem may occur when the cameras, robots and humans are located in the work area such that the half-spaces constructed by the cameras do not intersect to form a single polyhedron. Second, large changes in lighting conditions of the environment will cause the system to fail in detecting the location of the human(s). Third, prior knowledge of the number of people in the workcell is required for tracking multiple people individually.

7.4 Recommendations for future work

These are as follows:

1. The pixel classification algorithm is the bottleneck of the system. This algorithm should be made more computationally efficient.
2. A method to find the number of people entering the workcell needs to be developed. This will allow building individual polyhedrons of the people in the workcell without manually specifying *k*.
3. The system should be combined with a collision avoidance algorithm [39], such as to control the movements of the robot and avoid collisions.
4. The use of infrared cameras should improve the detection of the human(s) by thresholding warm objects in the scene.

8. Bibliography

[1] Litzenberger, G., World Robotics 2007, *International Federation of Robotics - Statistical Department*, 2007.

[2] G. Bekey, J. Yuh., "The status of robotics, report on the WTEC International Study: part II" *IEEE Robotics & Automation Magazine*. Pp 80-86, March 31, 2008.

[3] Keefe, B., "Risks using robots in industry", *ACM Committee on Computers and Public Policy*, Forbes, 1985.

[4] Howe, J., Review of Robot Injuries: One of the Best Kept Secrets, *National Robot Safety Conference RIA*, 2001.

[5] Bury, S., Rules of Collabration, *Manufacturing Automation Magazine*, 2007.

[6] TB. Moeslund, E. Granum., "A Survey of Computer Vision-Based Human Motion Capture." *Computer Vision and Image Understanding*, Acadeic Press, Vol. 81, pp. 231-268, 2001.

[7] Tsai, R. Y., "A versatile camera calibration technique for high accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses." *IEEE Journal of Robotics and Automation,* Vols. RA-3, pp. 323-344, 1987.

[8] S. Moezzi, A. Katkere, D.Y. Kuramura, R. Jain., "Reality Modeling and Visualization from Multiple Video Sequences." *IEEE Computer Graphics and Applications*. pp. 58-63, 1996.

[9] C. R. Wren, A. Azarbayejani, T. Darrell and A. P. Pentland., "Pfinder: Real-Time Tracking of the Human Body." *IEEE Trans. on Pattern Analysis and Machine Intelligence*. Vol. 19, pp. 780-785, 1997.

[10] A. Nakazawa, H. Kato and S. Inokuchi., "Human Tracking Using Distributed Vision Systems." *IEEE Int. Conf. on Pattern Recognition.* Vol. 14, pp. 593-596, 1998.

[11] E. Borovikov, L. Davis., "A distributed system for real-time volume reconstruction." *Fifth IEEE International Workshop on Computer Architectures for Machine Perception*. pp. 183-189, 2000.

[12] A.Davis, L. Mittal., "Unified multi-camera detection and tracking using region-matching." *Proc. of IEEE workshop on Multi-Object Tracking*. pp. 3-10, 2001.

[13] R. Cucchiara, M. Piccardi, A. Prati., "Detecting Moving Objects, Ghosts, and shadows in Video Streams." *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol. 25, pp. 1337-1342, 2003.

[14] R. Cucchiara, A. Prati, R. Vezzanni, L. Benini, E. Farella, P. Zappi., "Using a Wireless Sensor Network to Enhance Video Surveillance." *Journal of Ubiquitous Computing and Intelligence*, pp. 1-9, 2005.

[15] A. J. Joshi, S. Atev, O. Masoud, N. Papanikolopoulos., "Moving Shadow Detection with Low- and Mid-Level Reasoning." *IEEE International Conference on Robotics and Automation*. pp. 4827-4832, Rome, Italy, 2007.

[16] I. Haritaoglu, D. Harwood, and L.S. Davis., "W4: real-time surveillance of people and their activities." *IEEE Trans. on Pattern Analysis and Machine Intelligence*. Vol. 22, pp. 809-830, 2000.

[17] Steinmetz, G., "Empty Quarter." *National Geographic Magazine,* National Geographic Society, 2005.

[18] M. Seki, H. Fujiwara, K. Sumi., "A Robust Background Subtraction Method for Chainging Background." *IEEE workshop Applications of Computer Vision*. pp. 207-213, 2000.

[19] M.Kass, A. Witkin , D. Terzopoulos., "Snakes: Active Contour Models." *International Journal of Computer Vision*, pp. 321-331, 1988.

[20] L. Goncalves, E. Di Bernardo, P. Perona., "Reach out and touch space (motion learning)." *Third IEEE International Conference on Automatic Face and Gesture Recognition,* 1998.

[21] J. Fritsch, S. Lang , A. Kleinehagenbrock ,GA. Fink, G. Sagerer., "Improving adaptive skin color segmentation by incorporating results from face detection." *11th IEEE International Workshop on Robot and Human Interactive Communication*. pp. 337-343, 2002.

[22] P. Azad, A. Ude, T. Asfour and R. Dillmann., "Stereo-based Markerless Human Motion Capture for Humanoid Robot Systems." *IEEE Int. Conf. on Robotics and Automation*. pp. 3951-3956, 2007.

[23] Sidenbladh, H., "Probabilistic Tracking and Reconstruction of 3D Human Motion in Monocular Video Sequences." *Doctoral Dissertaion.* Royal Institute of Technology, Stockholm, Sweden, 2001.

[24] S. Knoop, S. Vacek , R. Dillmann., "Sensor fusion for 3D human body tracking with an articulated 3D body model." *Proc.IEEE Intl.Conf.Robotics and Automation (ICRA),* 2006.

[25] L. Sigal, S. Bhatia, S. Roth , MJ Black, M. Isard., "Tracking loose-limbed people." *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. pp. 421-428, 2004.

[26] J. Duetscher, A. Blake, I. Reid., "Articulated Body Motion Capture by Annealed Particle Filtering." *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Vol. 2, p. 2126, 2000.

[27] K. Ogawara, X. Li , K. Ikeuchi., "Marker-less Human Motion Estimation using Articulated Deformable model.". *IEEE Int. Conf. on Robotics and Automation*. pp. 46-51, 2007.

[28] Laurentini, A., "The Visual Hull Concept for Silhouette-Based Image Understanding."*IEEE Computer Society, IEEE Transaction on Pattern Analysis and Machine Intelligence*, pp. 150-162. Washington DC, USA, 1994.

[29] I. Cohen, H. Li., "Inference of Human Postures by Classification of 3D Human Body." *IEEE International Workshop on Analysis and modeling ofFaces and Gestures*, 2003.

[30] L. Wang, H. Ning, T. Tan, W. Hu., "Fusion of Static and Dynamic body biometrics of gait recognition." *International Conference on Computer Vision*, pp. 1449 – 1454, 2003.

[31] L. Wang, T. Tan, H. Ning, W. Hu., "Silhouette analysis-based gait recognition for human identification." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1505 – 1518, 2003.

[32] N. Robertson, I. Reid., "Behaviour understanding in video: a combined method." *IEEE International Conference on Computer Vision,* pp. 17-21, 2005.

[33] J-S. Hu, T-M. Su, P-C. Lin., "3-D Human Posture Recognition System using 2-D Shape Features." *IEEE International Conference on Robotics and Automation,* pp. 3933-3938, 2007.

[34] H.-D. Yang, A.-Y. Park, S.-W. Lee., "Gesture Spotting and Recognition for Human-Robot Interaction." *IEEE Transactions on Robotics, IEEE Robotics and Automation Society*, Issue 2, Vol. 23, pp. 256-270, 2007.

[35] DM Ebert, DD Henrich., "Safe human-robot-cooperation: image-based collision detection for industrial robots." *International Conference on Intelligent Robots and System*, pp. 1826- 1831, 2002.

[36] Y. Lu, L. Zeng, G.M. Bone., "Multisensor System for Safer Human-Robot Interaction." *IEEE International Conference on Robotics and Automation*, pp. 1767-1772, Barcelona, Spain, 2005.

[37] S. Kuhn, D. Henrich., "Fast Vision-Based Minimum Distance Determination Between Known and Unkown Objects." *IEEE International Conference on Intelligent Robots and Systems,* pp. 2186-2191, 2007.

[38] KG, Pilz GmbH & Co., Safety camera system SafetyEye. [Online] [Cited: May 13, 2008.] http://www.pilz.de/products/sensors/camera/f/safetyeye/index.en.jsp.

[39] L. Balan, G.M. Bone., "Real-time 3D Collision Avoidance Method for Safe Human and Robot Coexistence." *IEEE International Conference on Intelligent Robots and Systems*, pp. 276-282, 2006.

[40] Unimation., Specifications of PUMA 700 series Robot. [Online] [Cited: May 25, 2008.] http://mama.indstate.edu/users/thotas/spec700.htm.

[41] Litwiller, Dave., CMOS vs CCD: Maturing Technologies, Maturing Markets, *Photonics Spectra - Laurin Publishing*, 2005.

[42] Edmund Optics ., Edmund Optics Technical Support. [Online] Edmund Optics. [Cited:July 04, 2008.], http://www.edmundoptics.com/techSupport/DisplayArticle.cfm?articleid=290.

[43] Point Grey Research., *Dragonfly2 Specifications.* 2007.

[44] K. Huang, Y. Ma., T.R. Kurfees, "A Survey of Geometric Vision.". *Robotics and Automation Handbook.* New York : CRC Press, 2005.

[45] Horvath, M., Image:RGBCube b.svg. *Wikipedia, the free encyclopedia.* [Online] October 26, 2006. [Cited: May 28, 2008.] http://en.wikipedia.org/wiki/Image:RGBCube_b.svg.

[46] Denelson83., Image:YUV UV plane.png . *Wikipedia, the free encyclopedia.* [Online] August 25, 2006. [Cited: May 28, 2008.] http://en.wikipedia.org/wiki/Image:YUV_UV_plane.png.

[47] J. Bruce, T. Balch,M. Veloso,., "Fast and inexpensive color image segmentation for interactive robots." *International Conference on Intelligent Robots and Systems*, Vol. 3, pp. 2061-2066, Takamatsu, Japan, 2000.

[48] Andrew, A. M., "Another Efficient Algorithm for Convex Hulls in Two Dimensions,." *North-Holland Publshing Company - Information Processing Letters*, Issue 5, Vol. 9, pp. 216-219, 1979.

[49] Wikipedia, *Wikipedia, the free encyclopedia.* [Online] April 13, 2006. [Cited: June 14, 2008.] http://en.wikipedia.org/wiki/Image:Half_Space.svg.

[50] S. Boyd, L. Vandenberghe., Convex Optimization, *Cambridge University Press*, pp. 27,561-566, Cambridge, UK, 2004.

[51] F. P. Preparata, M. I. Shamos., Computational Geometry: An Introduction. *Springer-Verlag*, pp. 262,307-312, New York, 1985..

[52] Beer, Hans de., "Observations on the history of Dutch physical stature from the late-Middle Ages to the present." *Economics & Human Biology*, Elsevier, Issue 1, Vol. 2, pp. 45-55, March 2004.

[53] Point Grey Research Inc., Product Support - Downloads. *Point Grey Research Inc.* [Online] April 30, 2007. [Cited: June 30, 2008.] http://www.ptgrey.com/support/downloads.

[54] Intel(R) Open Source Computer Vision Library., Open Computer Vision Library. SourceForge.net. [Online] March 15, 2001. [Cited: June 30, 2008.] http://sourceforge.net/projects/opencvlibrary/.

[55] Canadian Heritage Information Network., Digital vs analogue cameras. *Canadian Heritage Information Network.* [Online] 08 01, 2001. [Cited: July 4, 2008.] http://www.chin.gc.ca/English/Digital_Content/Small_Museum/sub9.html.