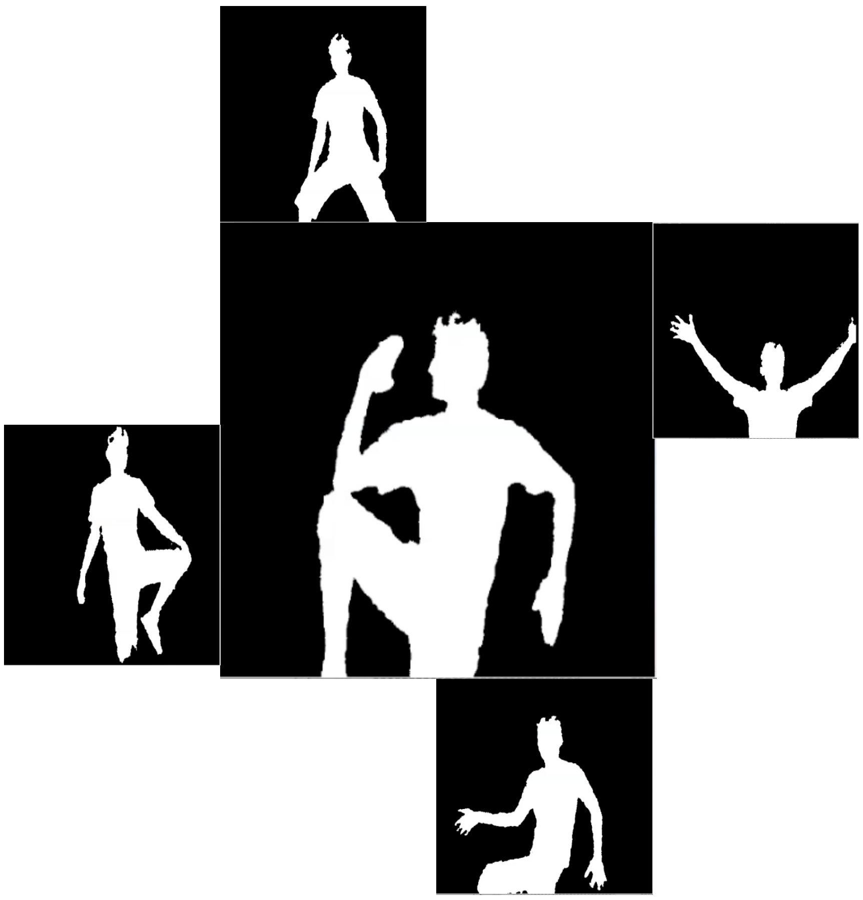
HUMAN DETECTION AND EXTRACTION USING KINECT DEPTH IMAGES



MASTER'S THESIS

SUNDAR NARAYAN KRISHNAMURTHY
i7913967@bournemouth.ac.uk
sunnarine@gmail.com
MSc Computer Animation & Visual Effects

Contents

Chapter 1

Introduction

Human detection has been a prominent area in the field of computer vision and artificial intelligence. The importance of a perfect human detection lies in the fact that human figures are extremely difficult to get detected and tracked. Among many of the reasons, to site a few are large variations in human postures, and human sizes, lighting in the environment, human clothing, varied activities etc. During a short period of involvement in a motion capture project, I got a chance to practically learn human pose capture with the use of sensors tied up all over the body. It is the optical motion-capture system that is being mentioned. The beauty of the human joint positions being tracked perfectly in real-time made a strong impact that led to the basement of this project. As the techniques behind human pose estimation and detection is extremely vast, an interest in implementing a tiny drop of this enormous ocean, led to the development of this project on human detection and extraction.

In the meantime, the state-of-the-art technology behind Microsoft's Kinect Xbox for 360, had started gaining a prominent interest among the computer vision and artificial intelligence geeks, in making use of the motion-sensing device in varied applications. Kinect's human motion-capture without any sensors or markers on the human body was really a breakthrough and welcome boon in computer vision. After going through numerous tweaks and works on the Kinect, it was widely learnt that Kinect can perform more than what it has been applied in nowadays – computer games. One such applied work was the use of Kinect in detecting human in a scene and extracting the human contour out of it. The paper presented in the Workshop on Human Activity Understanding from 3D Data in conjunction with CVPR, by Xia,L., et.al [2011] uses a novel idea of human detection using the depth information provided by the Kinect. Xia,L., et.al. employ a 2-D head contour model and a 3-D head surface model in extracting the human figures off the scene. This project is another implementation of this paper and all the steps provided in this project report are based on the guidance provided in the paper.

This report outlines the step-by-step procedure involved in Human detection and extraction from the Kinect data, discussing further the results obtained and analysing them.

1.1. Thesis Overview:

Chapter 2: Previous Work: An overview of the previous works that have been done in Human detection

Chapter 3: Process Overview: An overview of the procedures involved in the implementation. A simple description of how the project is broken into stages

Chapter 4: Dependencies: A detailed description on the hardware and the software used in the project coursework.

Chapter 5: Pre-processing: An elaborate explanation of the stages involved in pre-processing the data received from the Kinect. Each stage in this phase is explained in the sub-sections

Chapter 6: 2D-Chamfer Matching: A comprehensive account on the chamfer matching technique and a proposal on modified distance transform algorithm used.

Chapter 7: 3D Virtual Model Fitting: A complete coverage on the math and the logic behind the human head detection using the 3D – head surface model

Chapter 8: Extracting: A brief description on the extraction phase of the detected human figure

Chapter 9: Results & Discussion: A detailed analysis of the results obtained and discussing the optimal performance

Chapter 10: Improvements, Future Work & Conclusion: A brief account on what was done, what needs to be done.

Chapter 2

Previous Work

Though numerous works have been done on human detection technique, most of them use the visible light information in the scene for the detection process [Xia,L., et.al, 2011]. The detection algorithms are a simulation of the performance of human eye and brain.

Introduced few years back, Histogram of Oriented Gradients [Dalal,N., and Triggs,B., 2005] describes the technique of object detection using gradient orientation in localized points in an image. The algorithm was primarily described for the use of pedestrian detection in still images. Later the algorithm was tested for human detection in still images and videos. Scale-Invariant Feature Transforms [Lowe,D., 1999], or simply SIFT, have been extensively used in gesture recognition, where key points of objects extracted from a set of reference images are individually compared against each feature of the search image. Using local Edge Orientation Histograms [Levi.K. and Weiss,Y. 2004] as features has greatly improved the learning of frontal faces and thereby improving the real-time systems for learning profile faces. However, these methods employ the visible light to get the information from the scene. According to Xia,L.[2011], the major disadvantage of using visible light in the feature detection techniques, is that their accuracy gets considerably decreased when the scene gets complex and cluttered. Moreover, the precision fails when the features get occluded thereby making the detection a difficult task.

Owing to the disadvantages in the methods described above, emphasis was given to the usage of range images in object recognition. Sabata,B et.al [1993], describe the usage of 3-D range images in segmenting a scene into homogeneous surface patches using pyramidal data structures. In recent years, application of depth information captured from stereo cameras [Yang,HD. And Lee.S, 2007] and time-of-flight cameras [Ganapthi,V. et.al, 2010] have been used to determine the pose estimation and detection. More recently, using the Relational Depth Similarity Features [RDSF] based on the depth information obtained from time-of-flight camera, human detection algorithm has been proposed [Ikemura,S., Fujiyoshi,H., 2010].

Chapter 3

Process Overview:

As stated by Xia.L et.al[2011], the entire project can be divided primarily into four stages as follows:

1. Pre-processing
2. 2D Chamfer Matching
3. Virtual 3D Model Fitting
4. Extraction

Each stage is further portioned into several sub-stages that would simplify the understanding of the flow of the entire algorithm. Pre-processing deals with the preparation of the data from Kinect. While the 2D chamfer matching performs the initial human-head matching with a pre-defined template, the 3D model fitting actually eliminates the errors in the 2D stage, thus making the detection robust and accurate. As the name indicates, the fourth stage, extracts the entire human contour based on the detection results. Detailed description along with the algorithm has been given for each stage in the following chapters. Fig. 3.1 gives a clear picture of the process overview.

Though the paper [Xia,L et.al, 2011] deals with an additional final stage of exploring the Tracking algorithm, it is beyond the scope of this project. Tracking stage has been omitted due to the time constraints.

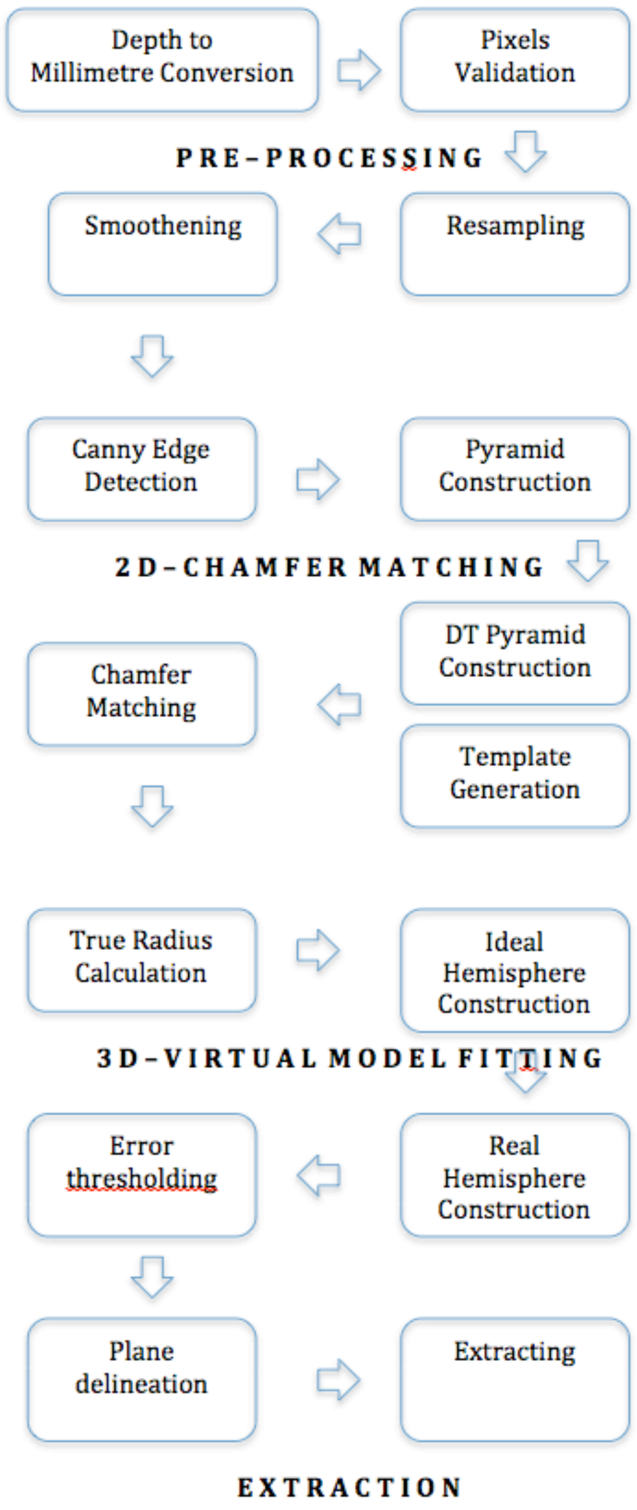


Fig. 3.1: Flow of the process

Chapter 4

Dependencies

4.1. Hardware Dependency:

Fig. 4.1: The Kinect sensor for Xbox 360 [Amazon, 2011]

In order to gather depth information from the scene, Kinect sensor for Microsoft Xbox 360 plays a major role in this project. The Kinect for Xbox 360, in Fig. 4.1, is an innovative motion-sensing device that has a pair of 3D depth cameras and a RGB camera. The RGB camera, like any other ordinary camera, captures the real-time video. However, the function of the 3D depth camera is to perform a variant of image based 3D-reconstruction[Ten, S. 2011]. According to PrimeSense, the company behind the Kinect technology, 3D depth cameras emit infra-red lights into the 3D space and unlike the usual time-of –flight method employed in most of the gesture recognition systems, the deformation of the information originally sent from the device is used for deciphering the image [Schramm, M. 2010]. The deciphered information thus received from the device is just an array of numbers, each number for a pixel. The details of processing this array will be discussed in Chapter 5.

4.2. Software Dependencies:

Being a Microsoft's hardware, it is obvious that only developers using Windows operating system can read data from the Kinect, through the SDK and the drivers released by Microsoft. However, an open-source version of these drivers and libraries would be a bounty, as the cost of the project gets limited to the cost of the hardware alone. Of course, the bounty was showered through the OpenKinect community. This community provides free, open-source libraries that enable the Kinect to be used not only with Windows but also with Linux and Mac. Perfectly named as libfreenect, these libraries have been employed in this project to make use of Kinect other than Windows.

4.3. USB Communication:

The interface between the development interface and the Kinect sensor is achieved through the usual USB communication. As the Kinect sensor comes with a lengthy USB cable, the physical communication is straightforward. However, the Kinect sensor gets recognised in operating systems other than Microsoft Windows, through another set of free, open-source libraries called libusb.

The libraries libusb and libfreenect make a fantastic pair in making Kinect available in all platforms.

Thus the physical cost of this project was limited to 100 GBP, owing to the Kinect for Xbox 360 sensor.

4.4. The SDK:

Nokia's Qt creator has been chosen as the application framework with C++ and OpenGL as the programming language and graphics API respectively. The choice of Qt along with C++ and OpenGL was arrived because of their efficiency, wide usage, flexibility and not the least, their availability at no cost.

Chapter 5

Pre-processing

Not diving instantly into the actual human detection process, this stage deals with the preparation of the data received from the Kinect. The data is processed in a manner that makes it meaningful, continuous and easier for calculations performed in further stages.

5.1. The Kinect Data

As discussed in the Chapter 4, Kinect reproduces the 3D scene captured by it through stereo triangulation. The captured frame is represented as an array of values with each pixel assigned a value. This array can be accessed in various ranges and formats that depend on the libraries being used to talk to the Kinect sensor. Libfreenect, the open-source library used in this project, provides ways of accessing these depth frame states in 10-bit, 11-bit, 10-bit packed, 11-bit packed formats etc. Thus, for example, if the 11-bit enumeration is chosen, as used in this project, then it is going to be 11-bit depth information in one unsigned integer per pixel. Further, it becomes much clearer that the size of this array is going to be the total number of pixels in the frame. Though not all available resolutions are available in all video modes received from Kinect, the infrared frame mode that gives the depth information, supports a medium resolution of 640 X 480. Evidently, the depth frame state array has 307,200 values each of 11-bit in size. This depth frame state array will be mentioned as simply depth array in further discussions. Moreover, all the information that is required to process the scene is contained in this sequence of integers. In a nutshell, whatever information needed to process a frame is sufficiently contained in this array.

Since the depth information is limited from 0 to 2047 (2^{11} precision), as a linear representation, if visualized, they provide only a plain meaningless black and white image. To make them visually meaningful, the 11-bit data is chunked into 8 regions, each with a range 0 to 255[Crock,N. 2011]. Consequently, each pixel gets a 3-Channel data of Red, Green and Blue values. In this way, the depth information can be simulated as a collection of colour gradients.

As a result, we are provided with two different arrays of two different ranges. While one has 307,200 elements each assigned an 11-bit unsigned integer value, the other has 921,000 elements (307,200 X 3) each assigned an 8-bit ($2^8 = 256$) unsigned integer value. The entire human detection and extraction process relies heavily and solely on these two ranges.

5.2. Depth to Millimetre

As discussed in the previous section, the data from the Kinect is just an array of 11-bit integers. However, an intriguing question of what do these values represent in the real world space arises when a close examination of these values is done. It gets interesting to know how this junk of 11-bit unsigned integers is connected to the 3D space being captured. Many experiments have been conducted in the recent months, to calibrate the depth frame data into real world metres. Precisely, attempts in calibrating to the actual meaning of the word 'Depth' itself have been made. After a personal communication, it was advised that the calibration through the equation [Magnenat,S. 2011],

$$\text{Depth (mm)} = 123.6 \times \tan(\text{depth}/2842.5 + 1.1863)$$

(Eqn. 5.1)

gives wrong values in larger range. Herrera,D., et.al[2011] have provided another equation for the calibration. Because of their proven accuracy, practicality and wide usage, it was decided to apply Eqn. 5.2 to calibrate the depth frame data to millimetres.

$$z_d = \frac{1}{\alpha(d - \beta)}$$

(Eqn. 5.2)
[Herrera,D., 2011]

Here, z_d is the calibrated disparity value in metres and d is the original disparity data from Kinect. The values of α and β are -0.00285 and 1091 respectively. Since, these calibrated values play an important role in the third phase of 3D Model fitting, which will be discussed in Chapter 7, the calibration gets an important place in pre-processing.

5.3. The Offset

After visualising the depth in the form of colour gradients, it becomes clear that some pixels do not have a valid depth value i.e. the scene information is missing in these pixels. These pixels appear as black random black specks in the scene. Linearly speaking, some of the elements in the depth array are not valid. If this array is used further, it would result in undesirable calculation issues. Thus, a valid check is done once the depth array is received. Any pixel that does not comply with the 11-bit range is offset to zero. This offset makes sure that all the values in the depth though not visually significant, make them meaningful in the least form from the perspective of linear calculations that are to be done further.

5.4. Nearest-Neighbour Interpolation

Offsetting the invalid pixels to zero does not suffice the depth array's quality to be sent to the next stage of the process. Though offsetting makes sure the validity of the depth array, it does not make the array meaningful and continuous. Xia.L[2011] states this as a kind of noise and to avoid this, it is primarily assumed that the 3D scene from which the data is being deciphered is continuous. Consequently, the data has to be continuous and bind logically within the values of the nearest pixel with non-zero valid value. Thus to avoid interference, the nearest-neighbour interpolation algorithm is applied on this depth array. Nearest-neighbour algorithm is the simplest image-resampling algorithm that resamples the given image to a new resolution. Let us look into the 1-dimensional interpolation, to have a clearer further understanding in the 2-dimensional level.

Interpolation is simply a convolution of a given discrete function g(x) with some continuous interpolation kernel w(x). [Burger,W. and Burge,M., 2008]

$$\hat{g}(x_0) = \sum w(x0-u) \cdot g(u),$$

Where $-\propto < u < \propto$

(Eqn. 5.3)

[Burger,W. and Burge,M., 2008]

The interpolation kernel w (x) for the nearest neighbour is,

$$w(x) = \begin{cases} 1, \text{ for } -0.5 \leq x \leq 0.5 \\ 0, \text{ otherwise} \end{cases}$$

(Eqn. 5.4)

[Burger,W. and Burge,M., 2008]

Visually speaking, the 1-D nearest neighbour interpolation makes the discrete signals continuous as shown in Fig. 5.1.

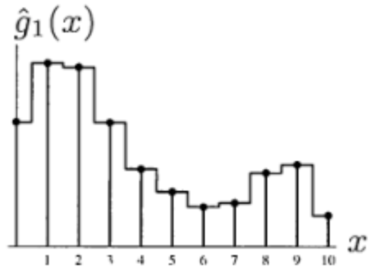


Fig.5.1: 1-D Nearest Neighbour Interpolation.
Image Courtesy: Burger.W, and Burge.M, 2008

Thus, the discontinuous values of the signal g(x) have been transformed to a continuous function ĝ(x) making it meaningful for any given value x. Promoting this to the 2-Dimensional spectrum, makes the noisy interfered image a continuous one. As Burger.W., and Burge,M., [2008] effectively state,

"The pixel closest to the given continuous point (x_0, y_0) is found by rounding the x and y coordinates independently to integral values",

$$\hat{I}\,(x_0, y_0) = I\,(u_0, v_0) \tag{Eqn. 5.5}$$

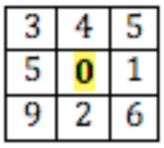
Having an objective to recover the true depth value of the missed pixels from the Kinect, which were offset to zero, the nearest neighbour algorithm provides the simplest and fastest way in achieving this. In this phase, the depth frame from the Kinect data is resampled to a size double the original and then reverted to the original size. Performing the interpolation twice on the pixel data though produces blocking effects that can be resolved in the next stage. This fills all the values in the depth array with continuous and meaningful values.

5.5. Median Filter

Though the Nearest-Neighbour interpolation reduces the interference and makes data continuous, it does produce some noise. The process of image smoothing can effectively do noise reduction. Though several algorithms have been proposed in the digital image-processing domain for noise reduction, median filter has a special attention as it is known for preserving the edge details in the original image. Emphasis for maintaining the edge details is made because of efficiency needed in the next major step of Canny Edge Detection. To better illustrate the median filter process, it is required to introduce the basis of a convolution kernel.

As we know, an image, which is an array of values, can be expressed in terms of a

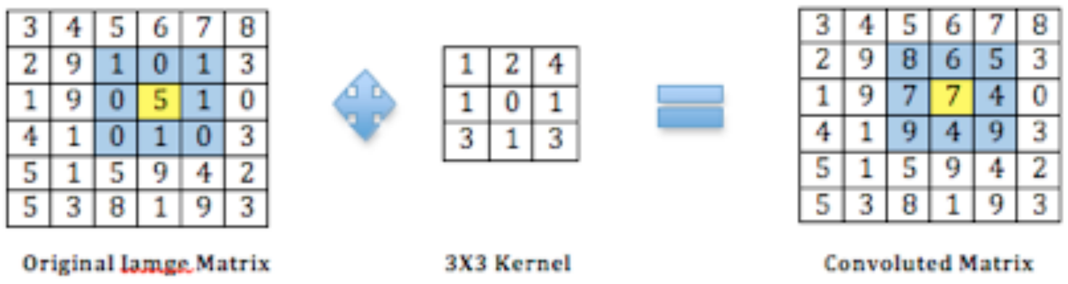
2-Dimensional matrix. The columns and rows of the image matrix is its resolution. Each element in the matrix is the pixel. A convolution kernel can be termed as another 2-Dimensional matrix, however, of lower resolution than the image that is placed on the original image matrix. Every convolution kernel has an element called anchor. Usually, the anchor lies in the centre of the convolution matrix. For example, a 3 X 3 convolution kernel with its anchor in the centre looks as in Fig. 5.2.



3	4	5
5	0	1
9	2	6

Fig. 5.2: A 3 X 3 Kernel with anchor at centre

When placed in an image matrix, summing up multiplied pixel values with the respective kernel value in the convolution matrix and thereby replacing the corresponding pixel value at the current anchor position of the kernel performs actual convolution. Performing convolution on all the pixels thus traverses the kernel over the entire image matrix. As the traversal proceeds from top left to the bottom right corner of the image, the anchor point moves over each pixel, thereby changing every pixel. This process is better explained in Fig. 5.3.



3	4	5	6	7	8
2	9	1	0	1	3
1	9	0	5	1	0
4	1	0	1	0	3
5	1	5	9	4	2
5	3	8	1	9	3

1	2	4
1	0	1
3	1	3

3	4	5	6	7	8
2	9	8	6	5	3
1	9	7	7	4	0
4	1	9	4	9	3
5	1	5	9	4	2
5	3	8	1	9	3

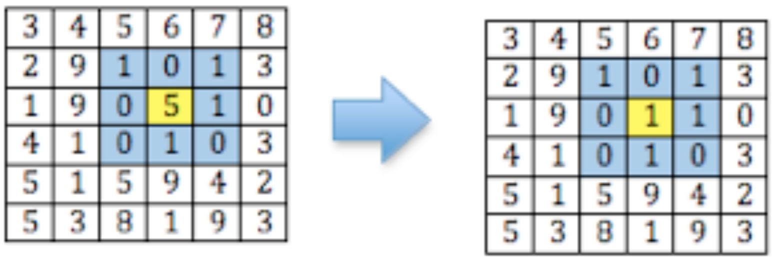
Fig. 5.3: A Convolution example

The highlighted region in the original image matrix is the current position of the 3X3 kernel. And the highlighted pixel value of '5' is the place where the kernel has got anchored in this position. Thus, the result of the convolution aims to change the value of '5' and in this case has changed it to '7'.

In a nutshell, a kernel is a window that is pivoted at the anchor, blocking the entire image matrix except at its resolution.

Though it seems simple, an actual convolution can perform powerful digital image processing calculations that result in derivatives, blurs, edge maps etc. [Sinha, 2010] Thus the major role is in designing the values of the kernel that best suit the operation. Not all convolutions perform the summation of multiplied values, as there are some exceptions. One such exception is the median filter.

Median filter, unlike usual kernels, sorts the values of the pixels and replaces the anchored pixel with the median value. For example, a 3X3 median filter on the same example stated above, results as in Fig. 5.4.



3	4	5	6	7	8
2	9	1	0	1	3
1	9	0	5	1	0
4	1	0	1	0	3
5	1	5	9	4	2
5	3	8	1	9	3

3	4	5	6	7	8
2	9	1	0	1	3
1	9	0	1	1	0
4	1	0	1	0	3
5	1	5	9	4	2
5	3	8	1	9	3

Fig.5.4: Median Filtering

As we can see, the value '5', which is the anchor of the median filter, has been filtered to '1'. The filtered value is achieved by extracting the values visible through the filter window (1,0,1,0,5,1,0,1,0 in this case), sorting them in ascending order (0,0,0,0,1,1,1,1,5 in this case) and then replacing the anchor with the median value of the sorted values. As the filter window traverses over the entire image, the original image gets smoothened effectively. A median filter of 4X4 resolution is used to smoothen the Kinect frame data that has previously been interpolated using Nearest-Neighbour interpolation.

Chapter 6

2D Chamfer Distance Matching

As the depth array has been pre-processed, it is the right time to dive into the human detection procedure. This stage effectively improves the precision of the actual human detection process that is employed in the third phase, thereby reducing the time taken for the actual detection [Xia,L. 2011]. This phase is further broken down into the sub-routines as discussed in this chapter.

6.1. Canny Edge detection

According to the Xia.L,[2011], the first stage in the human detection process is to extract the edges that are embedded in the depth array to determine any trace of human head contour. Canny edge detection has been employed because of its robustness and its flexibility to adapt to all environments [Wikipedia, 2011]. Though Canny Edge Detection is a sub-routine of this phase, the processes involved in this edge detection are broken down further.

Developed by John,F.Canny in 1986, Canny Edge Detection is a multi-stage edge detection algorithm to extract as much edge details as possible from the original image. The stages in the Canny edge detection are as follows [Kuntz,N. 2006]:

1. Applying Gaussian blur to the image
2. Calculating Edge Strength & edge direction at each pixel
3. Tracing edges
4. Non maximum suppression

When performing edge detection, it is necessary to decide the level and accuracy of edge details to be extracted from the image. Moreover, the decision depends on the requirements of the project. As the project deals with human head contour edge detection, through 2D Chamfer Matching, it is not necessary to have all the edges, especially the weak ones, extracted from the scene. As Sinha [2010] mentions,

"Do you want to look at a leaf or the entire tree? If it's a tree, get rid of some detail from the image (like the leaves, twigs, etc.) intentionally"

Blurring an image plays an important role in ignoring the minor details of the image. With its mathematically proven abilities, Gaussian has proved itself as one of the best blurring kernels [Sinha, 2010]. The Gaussian kernel is a 5 X 5 matrix, as shown in Fig 6.1. Once the minor details in the image have been blurred out, the actual edge detection starts by evaluating the gradient strength and its direction at each and every pixel of the image. The gradient strength at a given point (x, y) is given by the following equation

$$G_{(x,y)} = \sqrt{(G_x^2 + G_y^2)}$$

(Eqn. 6.1)

In Eqn. 6.1, G_x and G_y are the X and Y derivatives of the gradient at the point being considered [Sinha, 2010]. As discussed earlier in the previous chapter (Section 5.5), about the basics of convolution, recall that one of the major uses of convolution kernels is to find the derivatives at a given point. The kernels, also called as Masks, used for finding the X and Y derivatives are respectively Sobel X and Sobel Y masks. Sobel is a 3 X 3 convolution matrix and the masks used in this project are shown in Fig. 6.2.

$$(1/159)\ \begin{array}{|c|c|c|c|c|}\hline 2 & 4 & 5 & 4 & 2 \\ \hline 4 & 9 & 12 & 9 & 4 \\ \hline 5 & 12 & 15 & 12 & 5 \\ \hline 4 & 9 & 12 & 9 & 4 \\ \hline 2 & 4 & 5 & 4 & 2 \\ \hline \end{array}$$

Fig. 6.1: The 5X5 Gaussian Kernel [Kuntz, N. 2006]

$$\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

Fig. 6.2: The 3X3 Sobel X & Sobel Y Kernels [Kuntz, N. 2006]

The direction of the gradient at a given point (x, y) is calculated by the formula in Eqn. 6.2 [Sinha, 2010]

$$\Theta = \arctan(G_y / G_x)$$

(Eqn. 6.2)

Before starting the actual edge detection process, the most important point to remember, Sinha[2010] says, is that an edge is always perpendicular to the gradient direction. This makes us clear that gradient intensities change across the edge and not along the edge.

In the edge detection process, let us assume that we are in the white pixel, as shown in Fig. 6.3. There are only four possible edge directions viz. along east-west (yellow pixels), along north-south (green pixels) and along the diagonals (blue pixels and red pixels). Since we have already calculated the gradient orientation of the pixel through Sobel masks, the primary task lies in the segregation of the pixels based on their gradient orientations. This segregation helps in faster tracing along the edges.

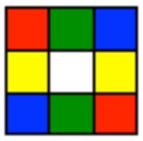


Fig. 6.3: Example Edge gradients

In the example edge gradient stated in Fig. 6.3, if the gradient orientation is in the range from 22.5 to 67.5 degrees it would make sense that the gradient changes from the top left corner pixel to the bottom right pixel and thereby inferring that an the edge flows along the forward diagonal.

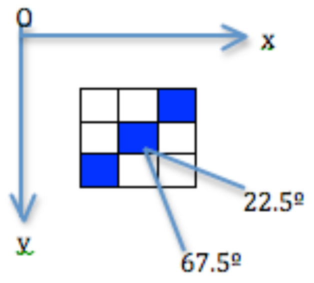


Fig. 6.4: Edge Orientation & gradient direction

Now, the central blue pixel is marked as a part of the edge, if the gradient strength at the pixel's position is greater than a prescribed value. This value is termed as the higher threshold in the Canny edge detection process. Thus the next pixel along this particular edge would be either the top right or the bottom left one. The index is transferred to that pixel and the check is performed once again. Similar are the cases when the gradient directions range between, 67.5 to 112.5, 112.5 to 157.5 and 157.5 to 180 or 0- 22.5, except that the pixels indices are transferred in accordance with each range. This process of segregation and checking, called as Non-maximum suppression, results in thin edges as they might have been broken at some points [Sinha, 2010]. To make them continuous, a process known as Hysteresis is done with another check value 'lower threshold'. The pixels on either side along the edge direction of the edge pixel i.e. in the directions perpendicular to the gradient direction are checked for the same gradient direction. Passing the condition along with their gradient strengths greater than lower threshold makes them the part of the edge. Consequently, the edge gets grown till there is no change in the image.

6.2. Resolution Pyramid Construction

The importance of edge detection in 2D Chamfer Matching becomes more authentic as Borgefors.G., [1988] mentions that matching is the prominent problem in digital image analysis and edges are the most important low level image features. An effective improvement in the conventional Chamfer Matching technique was made in 1988 by Borgefors,G., by providing a hierarchical algorithm that builds image pyramids. The idea behind this hierarchical algorithm is to make several copies of the original edge image in lower resolutions and build them as a pyramid. The original edge image lays in the base of the pyramid, while the lower resolution ones stack the pyramid up.

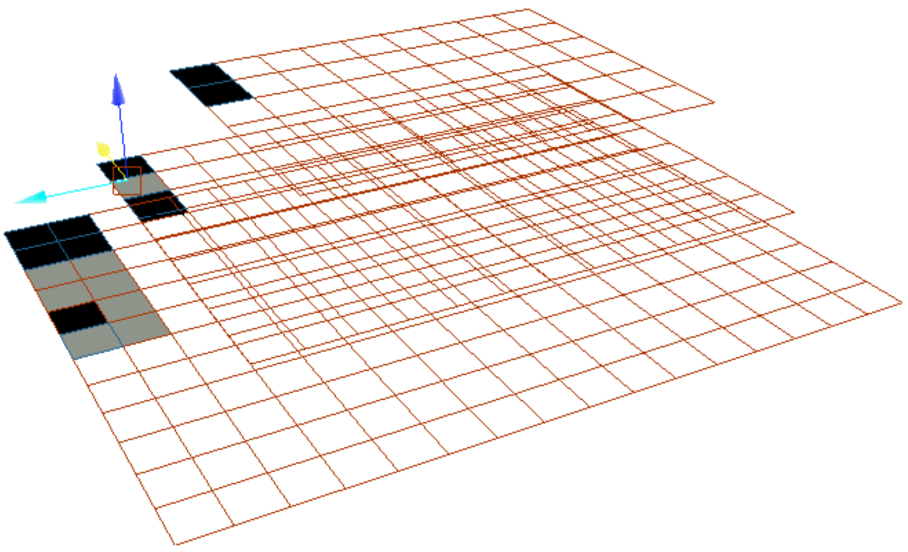


Fig. 6.5: Resolution Pyramid Construction, with the original edge image at level zero. Black pixels represent an edge while the grey ones represent vacant ones

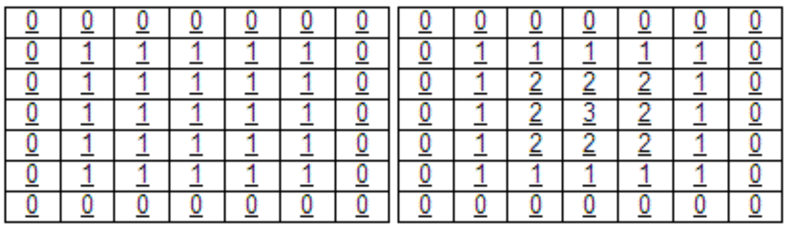
Borgefors,G.,[1988] mentions that the original image is converted into its lower resolution form by a father-son relationship. For every four pixels in the lower level, there is one pixel in the upper level; thereby the father pixel in the upper level has four sons in the level immediately beneath it However, if any one of the sons is 1, then the father becomes one. Otherwise, if none of the sons are 1, the father becomes zero. This pyramid construction is christened as 'OR Pyramid' [Borgefors,G., 1988] The number of levels in the pyramid grows till the number of fathers in the upper level becomes one. This is better illustrated in Fig. 6.5 where the black pixels represent 1 and the grey pixels represent 0. Though the 'OR Pyramid' was initially constructed, a slight deviation has been adapted in this project in the pyramid construction. This was because of the undesirable results that are discussed in the final chapters. The original edge image is resampled to a lower resolution through the Nearest-Neighbour interpolation. Moreover, the user can control the number of levels in the pyramid as the parameter has been exposed in the User Interface. Though larger number of levels provides excellent matching results, it has been restricted to '5' considering the computational time involved.

The major significance in building the resolution pyramid is to detect the edge matches at different scales. The significance is dealt in length in the 'Template Preparation' sub-section.

6.3. Distance Transform

In an edge image, each pixel is given a value that is a measure of its distance from the nearest edge pixel [Borgefors,S., 1988]. Thus, an edge pixel will have a value of zero, while its nearest non-edge pixel will have the value of '1'. Therefore, the distance transform created is another matrix of same resolution as the image, however, with values of each pixel's distance from the nearest edge, rather than the pixel's actual value. The most common ways of calculating the Distance Transform are listed below: [Wikipedia, 2011]

1. Euclidean Distance
2. Manhattan Distance
3. Chessboard Distance



0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	0	0	0	0	0	0

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	2	2	2	1	0
0	1	2	3	2	1	0
0	1	2	2	2	1	0
0	1	1	1	1	1	0
0	0	0	0	0	0	0

Fig. 6.6: The Binary Image and the corresponding distance map on the right [Wikipedia, 2011]

The Euclidean method, computes the distances by applying the Pythagorean formula of finding distance between two points [Wikipedia, 2011]. Though this method provides the accurate results, the computational costs and time involved are extremely high. The Manhattan method, computed the distances by simply finding the absolute differences of the pixels' coordinates [Wikipedia, 2011]. In the Chessboard method, the distance is calculated as the greatest of the coordinate differences between the pixels [Wikipedia 2011].

Borgefors,S., [1988] suggests another novel method called 'Sequential DT' which is termed as 'Chamfer Algorithm' as well [Burger,W. and Burge,M., 2008]. The two local distances in a 3 X 3 neighbourhood are the distance between the horizontal/vertical neighbours and between diagonal neighbours. The algorithm needs two Distance Masks each of 3 X 3 size. While one mask is traversed over the image forward from top-left to bottom-right, the other traverses backwards from bottom-right to the top-left [Burger,W. and Burge,M., 2008]. In order to have the closest accuracy of the Euclidean method, Borgefors,S., suggests the usage of 3-4 DT, where the forward and backward masks have the values as shown in Fig. 6.7.

$$\begin{bmatrix} 4 & 3 & 4 \\ 3 & \times & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix} \qquad \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \times & 3 \\ 4 & 3 & 4 \end{bmatrix}$$

Fig. 6.7: Forward and Backward masks.
[Burger,W. and Burge,M., 2008].

The DT calculation starts with a map of the same resolution as the image. However, the every element in the map is initialized with either infinity or zero. If the pixel at (x, y) of the edge image is an edge, then the corresponding map value at (x, y) is infinite and is zero if the pixel is not an edge. After this initialization, the forward mask is traversed. The anchor point of the mask, denoted as 'x' in the Fig. 6.7 gets the minimal of the convoluted nearby map values corresponding to the mask elements. Once the forward traversal is over, the backward traversal takes place on the altered distance map with the backward mask. Borgefors,S.,[1988] suggests to iterate this traversal as long as the map values in the consecutive forward and backward traversal sets become equal.

6.4. Diagonaled Iterative DT – A Proposal

A slight modification is done in this project by a process called Diagnoled Iterative DT. The usual iterative traversal, suggested by Borgefors,S.,[1988] resulted in an unending infinite loop, thereby worsening the performance of the system. Consequently, the iterations were removed and when the traversal was performed only once, the resulting DT map has relatively larger values than the actual distance. Thus, the iterations were switched on, however with a condition to stop when the maximum value in the distance map becomes lesser than the actual diagonal of the image. For example, in an edge image of 640 X 480 resolution, the maximum possible distance a non-edge pixel can have will not be greater than 800 ($\sqrt{(640^2 + 480^2)} = 800$). Moreover, if the distance value is greater than 800, it suggests further iterations need to be done to reduce the distance map values to a meaningful measure. Though the iterations get limited to 20, at times they go up to 300. However, they are considerably faster than the original method and have provided meaningful distance map values.

6.5. DT Pyramid

The Distance transform map has to be generated for all levels in the image. This is because it is the distance map that is going to be matched and not the edge images. As a result, another pyramid of the same dimensions but with distance maps is generated. This pyramid is termed as the Distance Transform Pyramid or simply 'DT Pyramid'.

6.6. Template Preparation

As the name indicates, Chamfer Matching, is the technique of finding a part of an edge image that matches perfectly with a reference image. This reference image is called 'Template'. In order to detect the human contour from the Kinect depth frame, it would better to start with the head. Thus a binary template of human head till the shoulders would be a good one in starting the matching process. Contour of the entire human body would not be a perfect template, as different templates cannot be loaded for different postures. Even if it is done, it is going to be inefficient, because the postures vary widely. However, a head template, as used by Xia,L. et.al[2011] shown in Fig. 6.8, would be an ideal one. This is because, whatever be the posture of the human body, the human head remains the same in shape and in size. However, the size of the head varies from person to person and also at various distances from the point of capture. Here comes the predominant significance of the resolution pyramids. When the matching is done with a single template on a single edge image of original resolution, the probability of a perfect match is extremely low. However, if the matching process is done on each level of the pyramid, that has different scales of the original image, the probability becomes higher. Consequently, the matching process becomes efficient and faster.

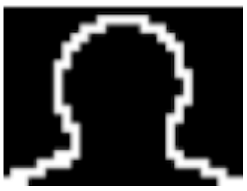
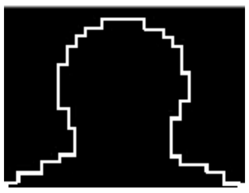


Fig. 6.8: Template used by Xia.L. Et.al [2011]


Fig, 6.9: Template used in the project

The binary template used in the project, is just a replica of the template used by Xia.L et.al[2011]. The template resolution is set as 160 X 120. To help the calculations in the next phase of 3D model fitting, the number of foreground pixels in the template image is calculated.

6.7. Chamfer Matching

The actual matching process gets started in this stage. As discussed earlier, the inputs for the Chamfer Matching are the DT Pyramid and the template image. Though the Chamfer Matching can be performed using the original resolution pyramid, using DT Pyramid provides sufficiently better results because of the localized matching. As a first measure, the size of the template has to be checked for a lower resolution than the DT being matched. This is just to a simple checkpoint, to avoid unnecessary results. The template is traversed over the entire DT map, from top-left to the bottom right. At each position of the template, the sum of the squares of the DT values of the all the elements coinciding with the edge pixels in the template is calculated [Borgefors,S., 1988]. When the number of edge pixels in the template image divides this sum, the root mean square (R.M.S) value of the match is obtained [Borgefors,S., 1988]. This value is also known as the 'Chamfer cost' [Burger,W. and Burge,M., 2008]. For instance, if at a given position of the template, all the edges of the template coincide with the zeros in that region of the distance map, it denotes the perfect match. In other words, an R.M.S value of zero gives the exact match and the position of the template on the DT map gives the position of the feature that is being searched and matched for. This in turn, gives another conclusion that; a position of the template with extremely large R.M.S value has the worst match. As a result, lower the R.M.S value better will be the match. The Chamfer Match gets clearer in the Fig. 6.10.

At the given position (r, s) with respect to the (0,0) of the search DT map, the R.M.S value is calculated by the formula,

$$R.M.S_{(r,s)} = (1/K) \sum D^2(r+i, s+j)$$

For all (i, j) in Template
$0 \leq r < W_s,\ 0 \leq s < H_s$
$0 \leq i < W_t,\ 0 \leq j < H_t$
(Eqn. 6.3)

In the Eqn. 6.3, K is the number of foreground pixels in the template image, (r, s) is the position of the top-left corner of the template with respect to the top-left corner of the DT Map, (i, j) is the local pixel position of the template and D(r+i, s+j) is the distance value in the map at (r+i, s+j).

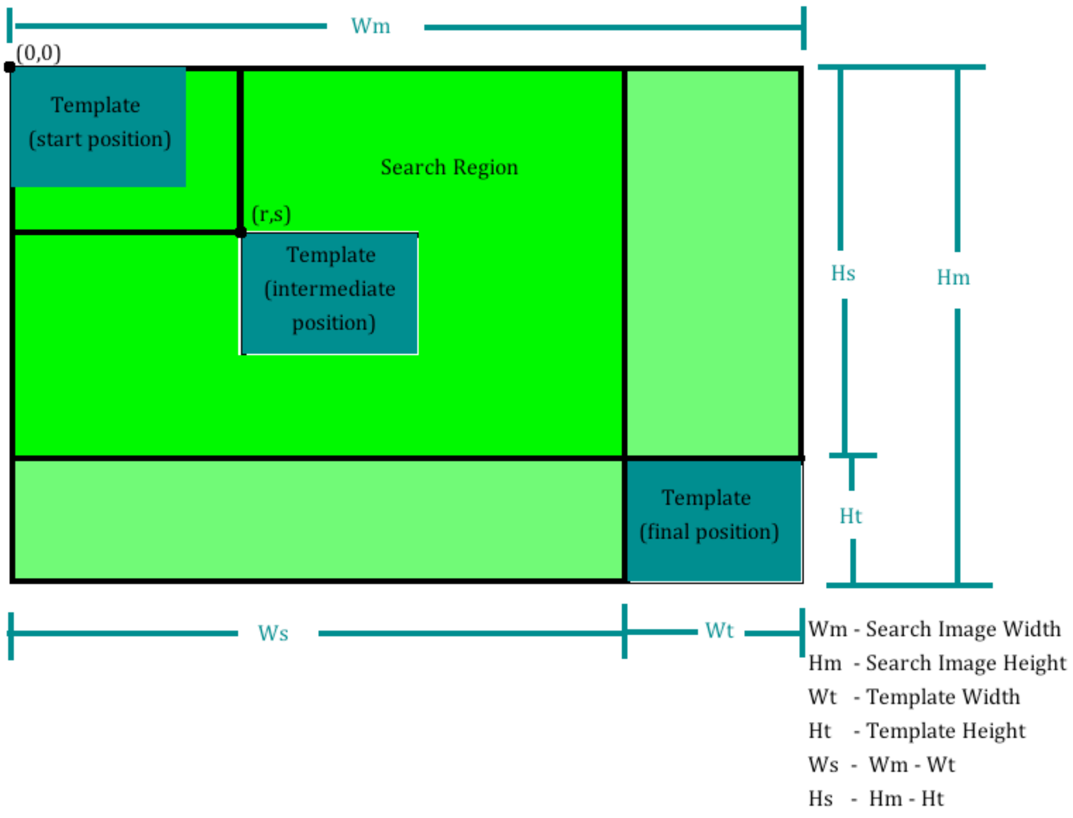


Fig. 6.10: Chamfer Matching : Template with the Search Image (DT Map)

As the Fig. 6.10 indicates, the number of R.M.S values obtained in one complete traversal of the template from its start position to the end position would be

$$\text{Search Area} = (W_s - W_t) \text{ X } (H_s - H_t) \tag{Eqn. 6.4}$$

However, it is not necessary to consider all these values. It is enough, if the first threshold number of minimum values were considered for further processing. Moreover, this is the case for a single level in the pyramid. Therefore, the number of minimum R.M.S values and their corresponding positions would be

$$\text{Number of Minimum R.M.S values} = \text{Threshold} \text{ X } \text{Number of levels in Pyramid} \tag{Eqn. 6.5}$$

The threshold value is exposed in the User Interface for interactive control. The output of the Chamfer Matching process is thus, the threshold number of regions with the minimum R.M.S values. All the processes in the second phase have been summarised in the Fig. 6.11.

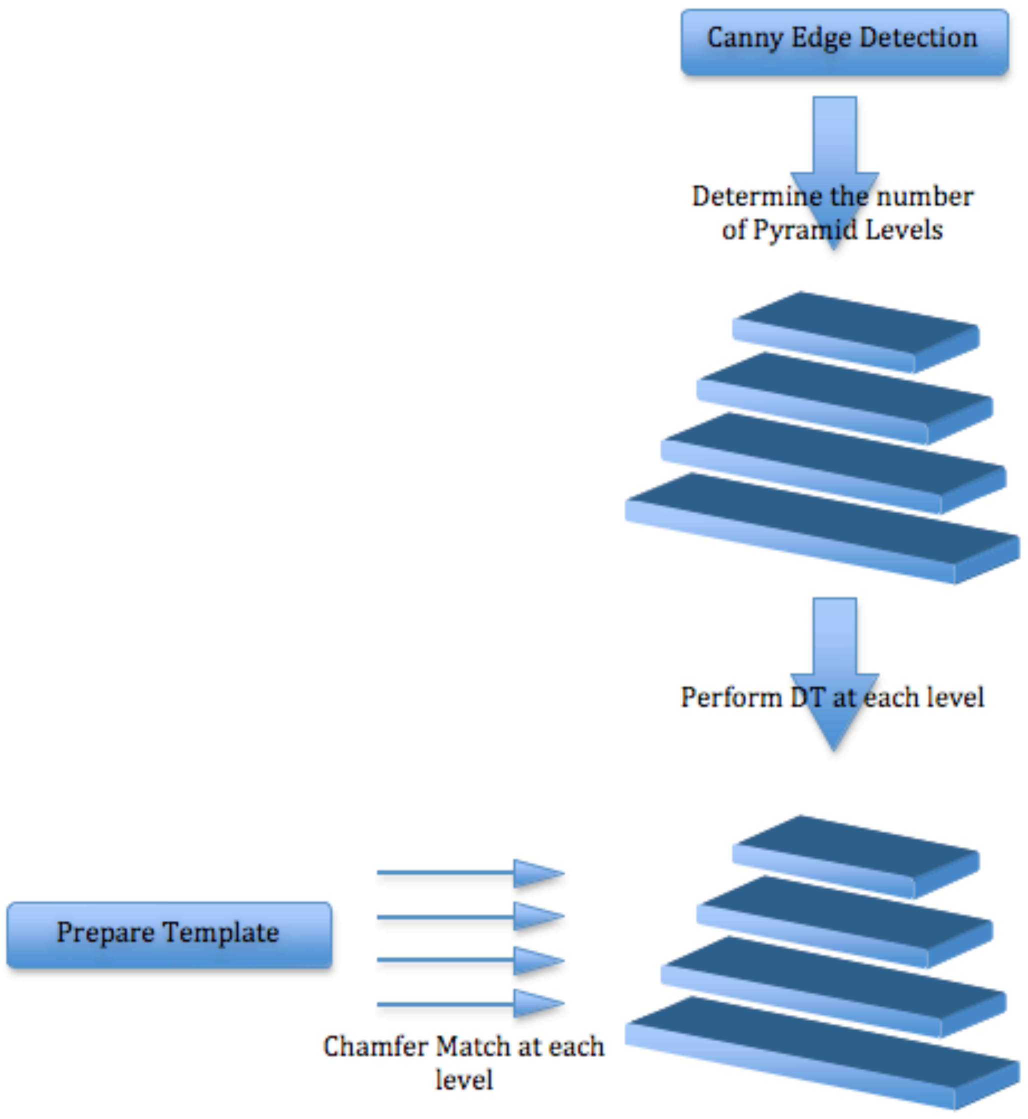

Canny Edge Detection
Determine the number of Pyramid Levels
Perform DT at each level
Prepare Template
Chamfer Match at each level

Fig. 6.11: Phase 2: 2D Chamfer Matching processes

Chapter 7

Virtual 3D Model Fitting

Chamfer Matching technique is just a rough scanning approach. Therefore, the results of Chamfer Matching cannot be authentic enough to spot the person's head. However, the results converge the area of exploration of the image, thereby increasing the computational costs involved in this phase of virtual 3D model fitting. As discussed in the previous chapter, the 2D Chamfer Matching provides a series of regions in the original image where there might be a presence of human "head-like"[Xia,L., et.al 2011] object. It is, however, not necessarily be the actual human head. With this ambiguity and to arrive at exact results Virtual 3D Model Fitting phase is started.

7.1. Calculating Head Parameters

A human head can be roughly estimated as a sphere of radius R. But, the radius is not constant all over. Thus, the human head possesses two parameters viz. the radius, along the horizontal axis and the height, along its vertical axis. In order to generate a Virtual 3D Model of a sphere, the only parameter that is required is the radius of the sphere.

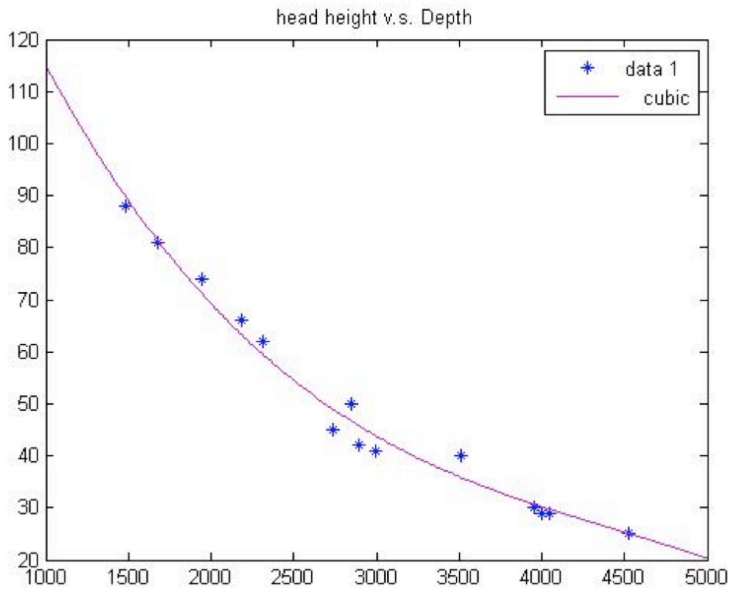


Fig. 7.1: Head Height and Depth relationship
Xia.L., et.al [2011]

Xia,L., et.al [2011] have conducted a regression test and have arrived at a cubic equation that gives the relationship between the depth value and the approximate height of the head, if it were to be present in the position of that depth value. The graphical result of this relationship is shown in Fig. 7.1.

The cubical equation that fits the graphical curve is computed as

$$-y = -1.3835 \times 10^{-9} x^3 + 1.8435 \times 10^{-5} x^2 - 0.091403 x + 189.38$$

(Eqn. 7.1)
[Xia,L., 2011]

In Eqn. 7.1, x represents the depth value and y represents the height of the head at the point where the depth is x. However, both the values are in millimetres. It should be recalled that, in the Pre-processing Chapter, there was a discussion on the conversion of the raw disparity values from Kinect to millimetres. Evidently, it is easier in this phase to retrieve the depth values in millimetres. Substituting the appropriate depth values at the pixels spotted by the Chamfer Matching in the equation, the tentative height of the head at these locations can be calculated. The radius of the head is calculated from the relation,

$$R = 1.33\ h/2$$

(Eqn. 7.2)
[Xia,L., 2011]

7.2. Locating Circular Edges

If there happens to be a head in the chamfer spots, then the probability of a circular edge in the region is high. Since, the radius had already been calculated in the previous step, it becomes easier to trace the circular edge. However, the radius for tracing a circular edge in an edge image should be in pixels and not in millimetres. In order to convert the radius value in pixels, the spatial resolution of the Kinect sensor needs to be known. As Wikipedia clearly states,

"The horizontal field of the Kinect sensor at the minimum viewing distance of ~0.8 m (2.6 ft.) is therefore ~87 cm (34 in), and the vertical field is ~63 cm (25 in), resulting in a resolution of just over 1.3 mm (0.051 in) per pixel"
[Wikipedia, 2011]

This has made the calculations much straightforward as the following relation can convert the radius in millimetres to number of pixels,

$$\text{Radius}_{(\text{pixels})} = (1/1.3) \times \text{Radius}_{(\text{mm})}$$

(Eqn. 7.3)

With the Chamfer spot as the centre and the Radius$_{(pixels)}$ as the range, the presence of edges in the region is performed. Mentioned earlier, as the human need not be a perfect sphere, it becomes quite ideal to search within the strict range of Radius $_{(pixels)}$. A minute relaxation is given as a threshold, to arrive at a circular disc region. The threshold is the number of pixels to reduce the Radius $_{(pixels)}$ and to increase it as well. The search for an approximately circular arc is searched in the region as shown in the Fig. 7.2.

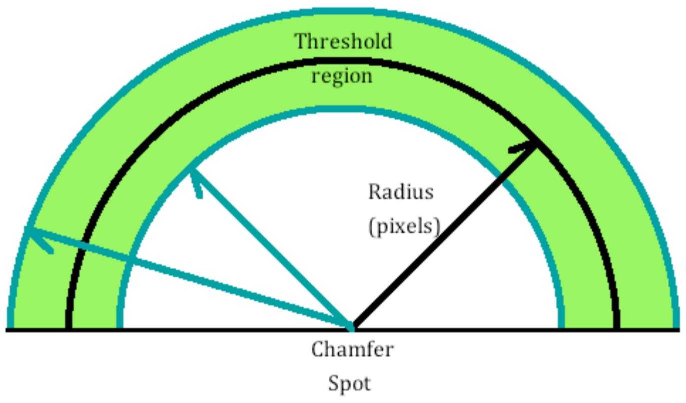


Fig. 7.2: Thresholded circular region where circular edge is searched

It is not necessary to carry all the chamfer spots to the next stage and the erroneous spots can be discarded in this stage. Therefore, only the chamfer spots that gave a certain maximum number of edge count in their thresholded region are passed to the next stage, while the remaining are discarded. This improves the efficiency of the final human head detection.

7.3. Virtual Hemisphere Model

As discussed earlier, the only parameter that is required for a hemisphere model is its radius. To get the exact value of the radius, we make use of the Distance Transform map that was generated in the previous stage. It is clearly mentioned that the Distance Transform Map is the numerical representation of the edge map, with values of each element being the shortest distance from its nearest edge pixel. Thus, if the value of an element in the DT Map is '7', for instance, it means the nearest edge pixel from this pixel is somewhere '7' pixels away. However, the edge pixel could be in its north or east or west or south or in any intermediate directions. Similarly, the DT value of the element at the Chamfer spot gives the exact value of the distance of that corresponding spot from its circular edge. Thus, the true radius, denoted as R $_{(true)}$, of the chamfer spot that has passed the initial threshold is the DT value obtained from the DT Map.

Once the true radius, R $_{(true)}$, is known it is required to build the 2D projection of the 3D hemisphere model. It becomes quite obvious that, the 2D projection of a 3D hemisphere model results in a perfect semi-circular region. As a result, the projection can be visualized as a semi-circular region, with the diameter being twice the value of true radius, R $_{\text{(true)}}$. For example, the projection of a hemisphere with R (true) as 4 pixels looks similar to the one in Fig. For a better visualization, a vertical plane bisects the hemisphere, where the pixels are being projected.

The hemisphere, once built, is considered for depth calculations in two senses viz. the true sense and the ideal sense. Calculation of the ideal depth values starts with an assumption that the hemisphere touches a vertical plane at one end, where the depth is regarded as zero. The point of contact of the hemisphere with the vertical plane is going to be the chamfer spot. This gets more precise, when it is assumed as,

"Imagine you put the hemisphere like a mask in front of you face to face, define the nearest point to you has depth-0, then you can get the depth of any other points in the mask"

[Personal Communication, Xia,L., 2011]

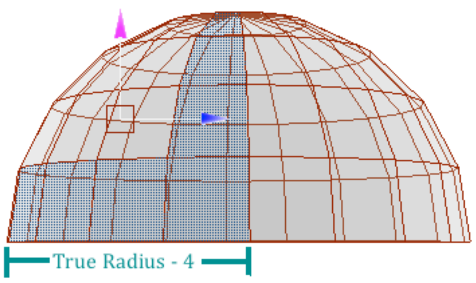


Fig. 7.3: Hemisphere - Radius $_{\text{(true)}}$ 4. Front View

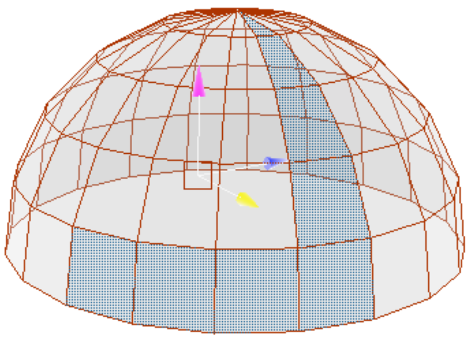
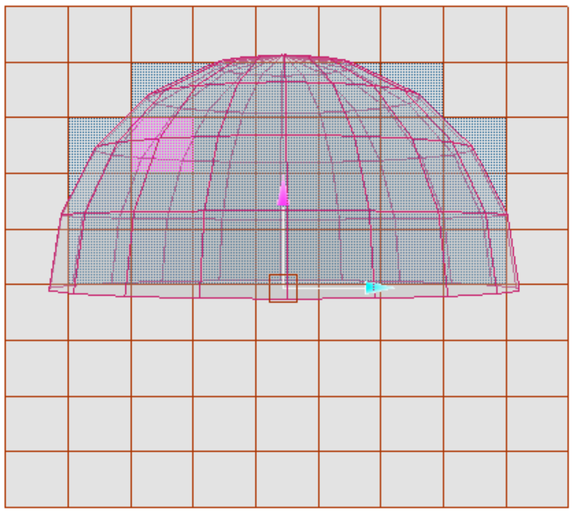


Fig. 7.4: Hemisphere - Radius $_{\text{(true)}}$ 4. Perspective View

With this assumption the ideal depth values can be calculated using the relationship between the spherical coordinates and the Cartesian coordinates. In a spherical coordinate system as shown in Fig. 7.7 any point in a hemisphere is represented by the tuple [r, θ, ф], where r is the radius of the sphere, θ is the azimuth angle in the x-y plane measured from positive x-axis and ф is the polar angle from the positive z-axis.

–



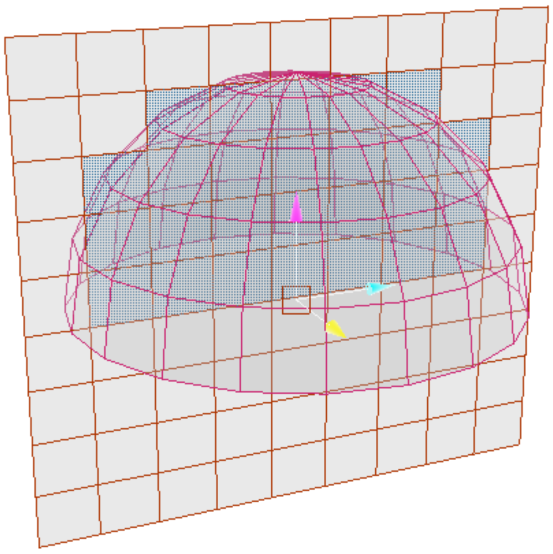


Fig. 7.5: Hemisphere - Radius $_{(true)}$ 4. Planar projection - Front View

Fig. 7.6: Hemisphere - Radius $_{(true)}$ 4. Planar projection - Front View

The Cartesian coordinates x, y and z are calculated based on the following conversion, [Weisstein,E., 2011]

$$\begin{aligned} x &= r\cos\theta\sin\phi \\ y &= r\sin\theta\sin\phi \\ z &= r\cos\phi. \end{aligned} \qquad \text{(Eqn. 7.4)}$$

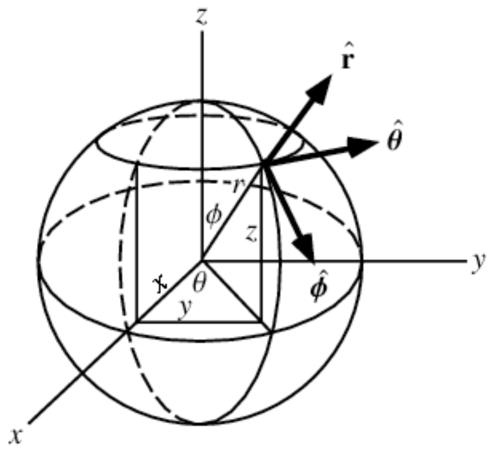


Fig. 7.7: The spherical coordinate system. {Weisstein,E., 2011]

Based on the assumption, as shown in the Fig. 7.8 the black dot is the pixel that touches the vertical plane. The depth values are nothing but the distances of every projected pixel on a plane that bisects the hemisphere, from the plane that touches the hemisphere at the black dot.

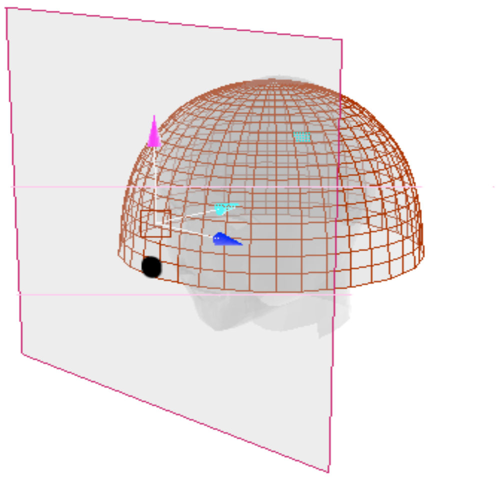


Fig. 7.8: Ideal Depth calculation assumption

Mathematically, it is the absolute difference between the x coordinate value of the Cartesian coordinate system and the true radius, R (true). As we know the spherical coordinate tuple, at every point in the hemisphere model being projected, the x values can very easily be calculated based on the above equations. Thus the ideal depth values can be calculated for the projected pixels.

Since, the actual depth values are already known from the pre-processing stage itself, getting the real depth values for the projected pixels is not a big task. Further, the real depth values are normalized based on the equation,

Depth_Normalized (i, j) = Depth (i, j) – min. Depth in the semi-circular region
For all i, j in the semi-circular region

(Eqn. 7.5)
[Xia,L,et.al 2011]

At the end of this stage we have two sets of values viz. the normalized real depth values and the ideal depth values of the region, where the hemisphere has been placed with its centre as the Chamfer Spot and the true radius, R (true).

7.4. Fitting the Virtual Model

It should be noted that, if this Chamfer Spot were to be a real human head, the normalized real depth values calculated, were not going to get deviated much from the ideal depth values. In other words, the deviation of the real hemisphere model from the ideal hemisphere model is going to be very less. In a nutshell, lesser the deviation more is the chance that the chamfer spot is a head. Based on this logic, the idea behind the fitting stage is as simple as calculating the difference between corresponding values in the normalized real depth and the ideal sets. This is something like, placing the ideal hemisphere on the human head and calculating the difference in depth values of their corresponding pixels. This gets clearer in the Fig.

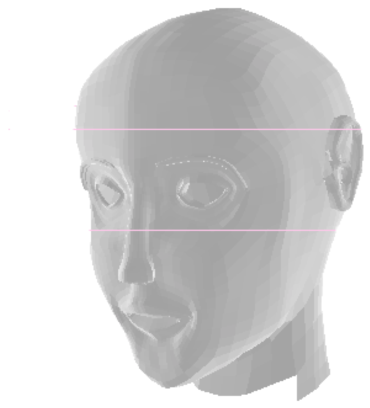


Fig. 7.9: The human head model

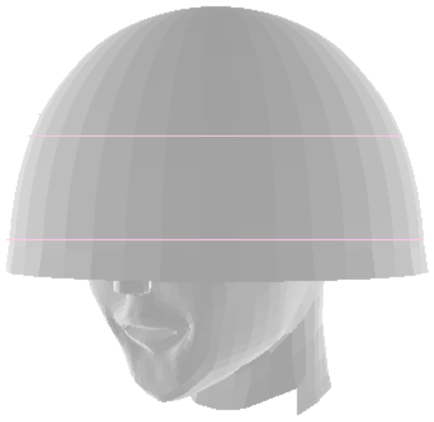


Fig. 7.10: The hemisphere model placed on the head

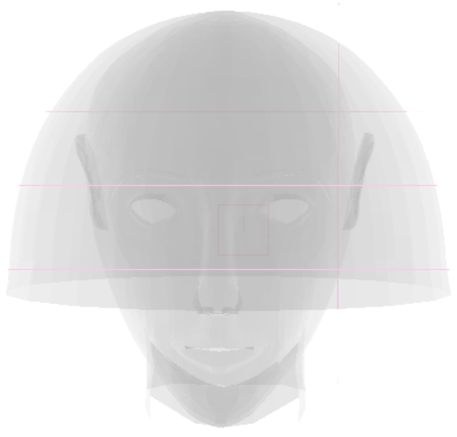


Fig. 7.11: Depth difference calculation. Front View

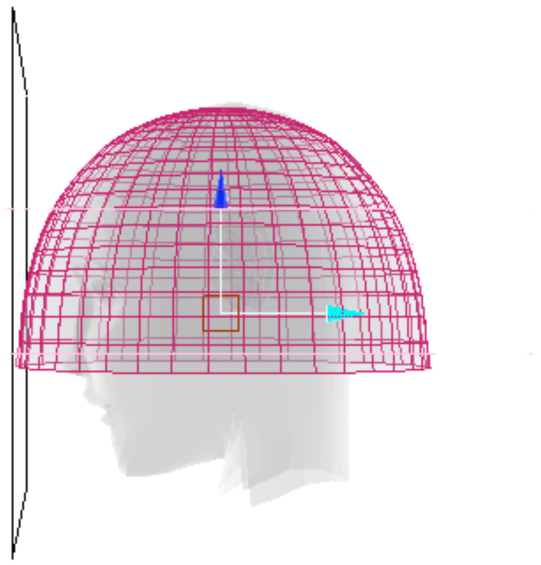


Fig. 7.12: The side view

To increase the accuracy, the square error between the normalized real depth values i.e. the depth of the human head, if it were a head and the real depth values of the hemisphere model is calculated as

$$Err = \sum | normalized\_real\_depth(i, j) - ideal\_depth(i, j) |^2$$

(Eqn. 7.6)
[Xia, L., et.al. 2011]

As we already have threshold number of chamfer spots, we get the same number of error values. The chamfer spot with the lowest error value is confirmed to be the human head. Selecting the first ‘n’ number of chamfer regions that have the minimal error values can increase the number of persons detected.

Chapter 8

Extracting

The final stage in the human detection is based on a simple logic that, if the depth value at the human head is 'd', for instance, the depth value of the entire human body is not going to get deviated from the value 'd' much. As a result, all the regions that have the depth value similar to the depth of the human head are added to the region, with the chamfer spot as the seed. The deviation of the depth values from the seed value is controlled by the threshold value. However, if a person stands on the floor and if it gets covered in the frame, the floor areas have the same depth as that of the human legs. In order to delineate the planar floors from the human leg, and any other parallel surfaces like table, the planar filter F is applied, with its anchor at either the third or the fourth element.

$$F = [1, 1, 1, -1, -1, -1]^T$$

[Xia,L., et.al, 2011]

Applying this filter helps in extracting the edges between the human contour and the planar floor areas. Thus, after applying the filter, Canny Edge detection is applied once again, however only at areas where the planar floors are. Applying the canny edge detector on the entire image produces a lot of noise. In order to resolve this issue, the median filter is once again applied to the extracted edges after applying the filter. The edge image, after smoothing with the median filter, is combined with the initial edge image. Thus the extracting phase has the chamfer spots with the minimal error values and the combined edge image. With the chamfer spot as the seed location, the regions with the depth value similar to the seed value are added to the seed location, thereby growing the region. The locations that are extracted are highlighted on a plain black image, giving a black and white extracted output of the human in the scene. This results in good extraction results. However, planar filter application has certain problems that are discussed in the Chapter 9.

Chapter 9

Results & Discussion

9.1. Technical Specifications:

The following are the technical specifications of the machine on which the project was carried out:

Processor: Intel Core i7
Clock Speed: 2.2 GHz
Number of Processors: 1
Number of Cores: 4
Cache: 6MB
Memory: 4GB

9.2. Test Environments

The project was carried out and tested in two environments. However, both were indoor environments.

9.3. Close Plane Environment

To get lower false positive ad false negative rates, the distance between the Kinect and the farthest plane it could capture was limited to 202 centimetres. This was intentionally done, so that the human figures can interact with the Kinect sensor within the range where Kinect sensor's depth range is effective. The sensor was also placed at a height of approximately 5½ feet from the ground. No importance was given to the lighting conditions as the sensor works purely on infrared lasers. However, to aid the RGB camera, lighting was not ignored completely. Starting off with very simple postures, the test was carried out regressively with extreme human postures. The tests carried out in this environment is split into four categories:

1. One person – without an obstacle
2. One person – with an obstacle
3. Two persons – without an obstacle
4. Two persons – with an obstacle

Also, the environment had very less number of distractions other than the human figures involved. As mentioned earlier, this was done to make sure that most of the detections are positive. Timers were used to calculate the time take to process each frame. Also, the time taken by the Chamfer Matching in each level of the pyramid was also taken. Chamfer matching was the only process that consumed most of the time in the detection process. Simple cardboard boxes were used as obstacles. The main aim of using the obstacles was to provide additional stronger edges to the environment. They were placed as close as possible to the human figures and sometimes supported by hands, to check whether the extraction includes the obstacles as well in the output. While engaging two human figures, tests were carried out with the figures interacting with each other i.e. having some sort of physical contact and also without interaction.

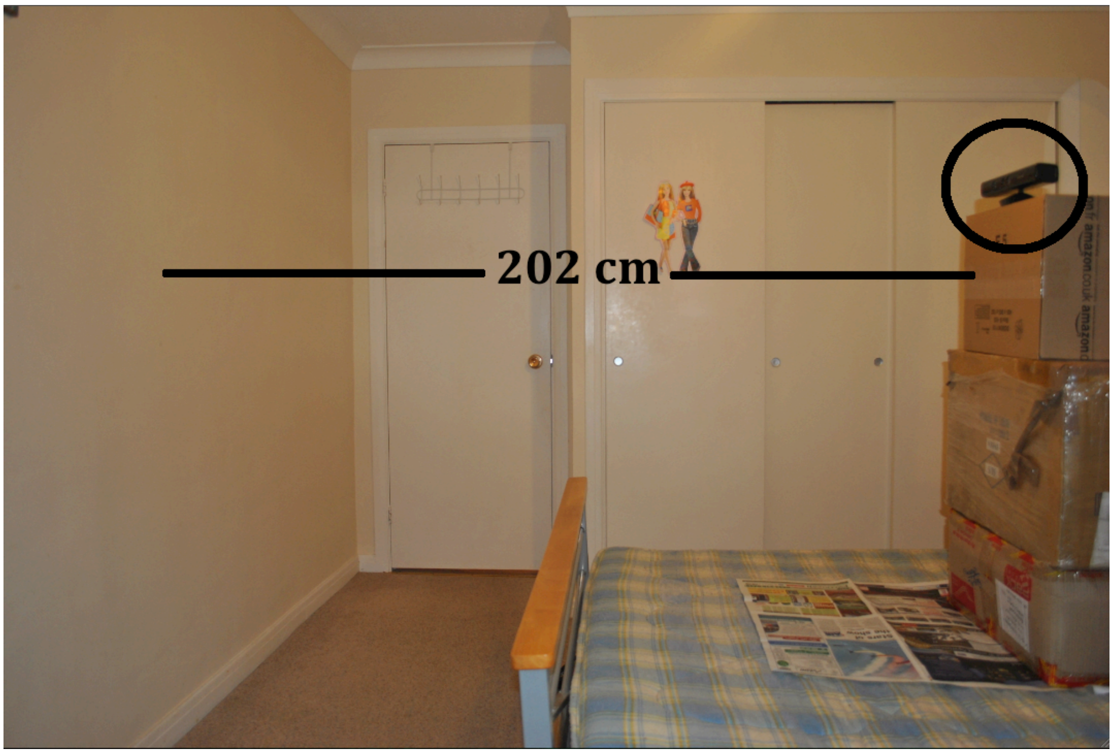


Fig. 9.1: Close Plane Environment: The farthest plane from Kinect is only 202 cm

The Figs 9.2 to 9.6 show the detection and extraction results carried out in the Close plane environment. The results were extremely satisfactory as the contours extracted were precise even in the extreme postures. Though some of the postures failed miserably, the best detection results have been shown here. The poor results will be discussed further. It should however, be mentioned that, the processing speed of a single frame ranged from a minimum of 22 seconds to a maximum of 32 seconds, resulting in an average of 27 seconds per frame. It was also to be noted that, a usual chamfer matching technique takes 60 – 120 seconds, for matching a template of size 257 X 257 over the search image of size 600 X 824 [Experienceopencv, 2011]. Also, the time taken by the chamfer matching in each level of the pyramid was noted and were found to be on an average of 13, 8, 5 and 2 seconds respectively from level zero to level four. The tests were carried out at different threshold levels and at lower levels of Pyramid. However, the best results obtained in this "Close Plane" test environment had four levels of the pyramid, with the depth threshold to be 25. The high and low thresholds for the Canny Edge detection were tried with several combinations, finally resulting in a best combination of 3.0 and 4.0 respectively.

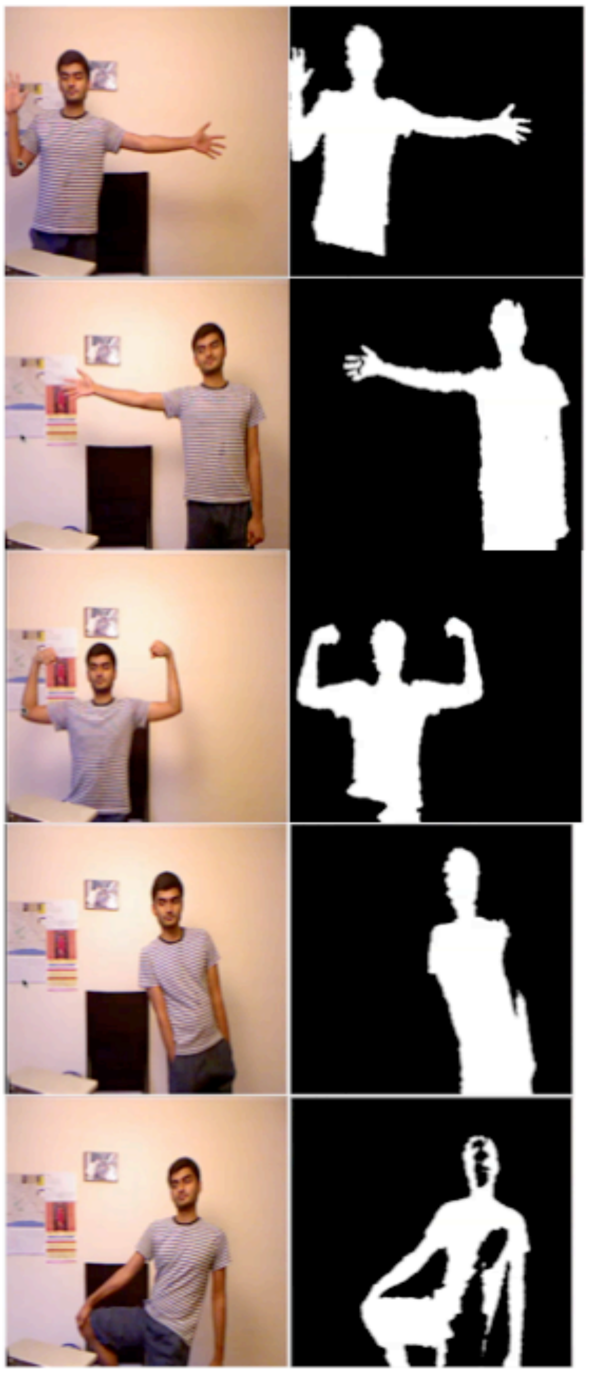


Fig. 9.2: Close Plane Environment with one person and no obstacle. Simple postures

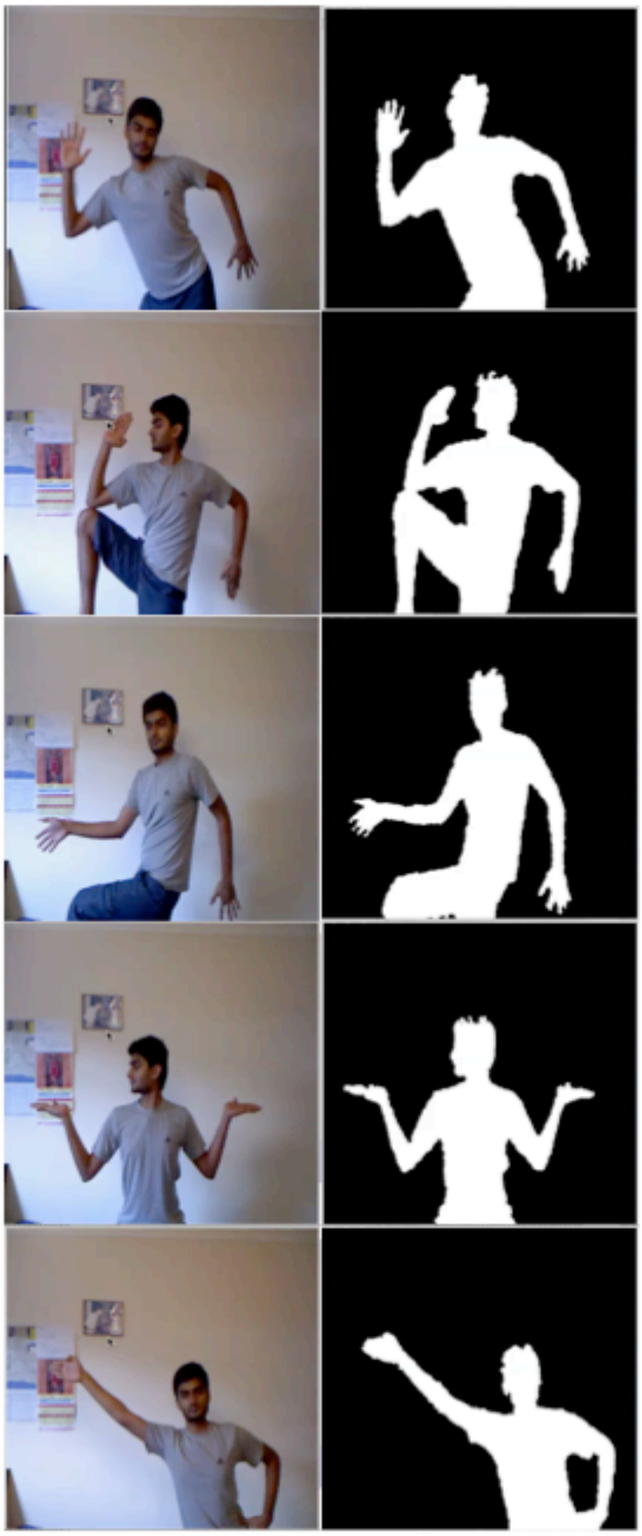


Fig. 9.3: Close Plane environment with one person and no obstacle. Extreme postures

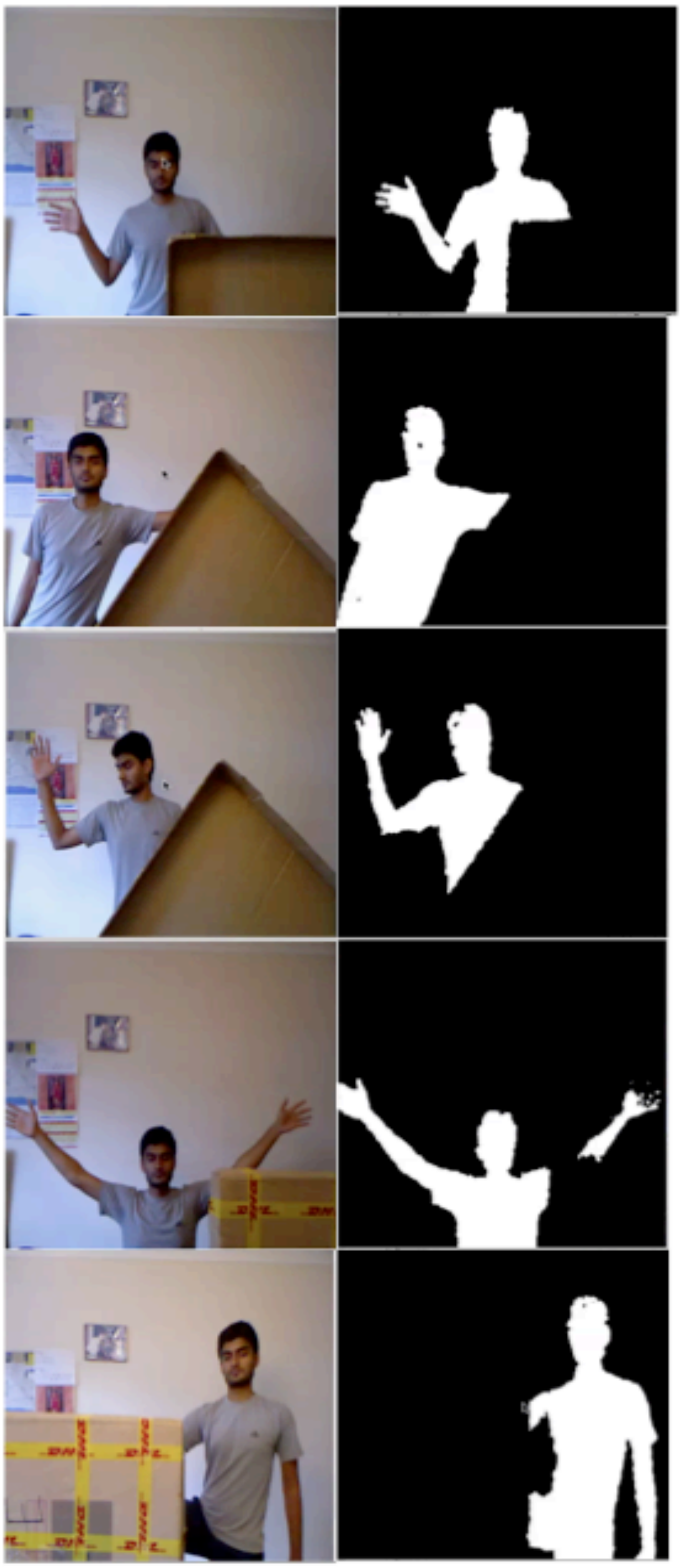


Fig. 9.4: Close Plane environment with one person and with an obstacle

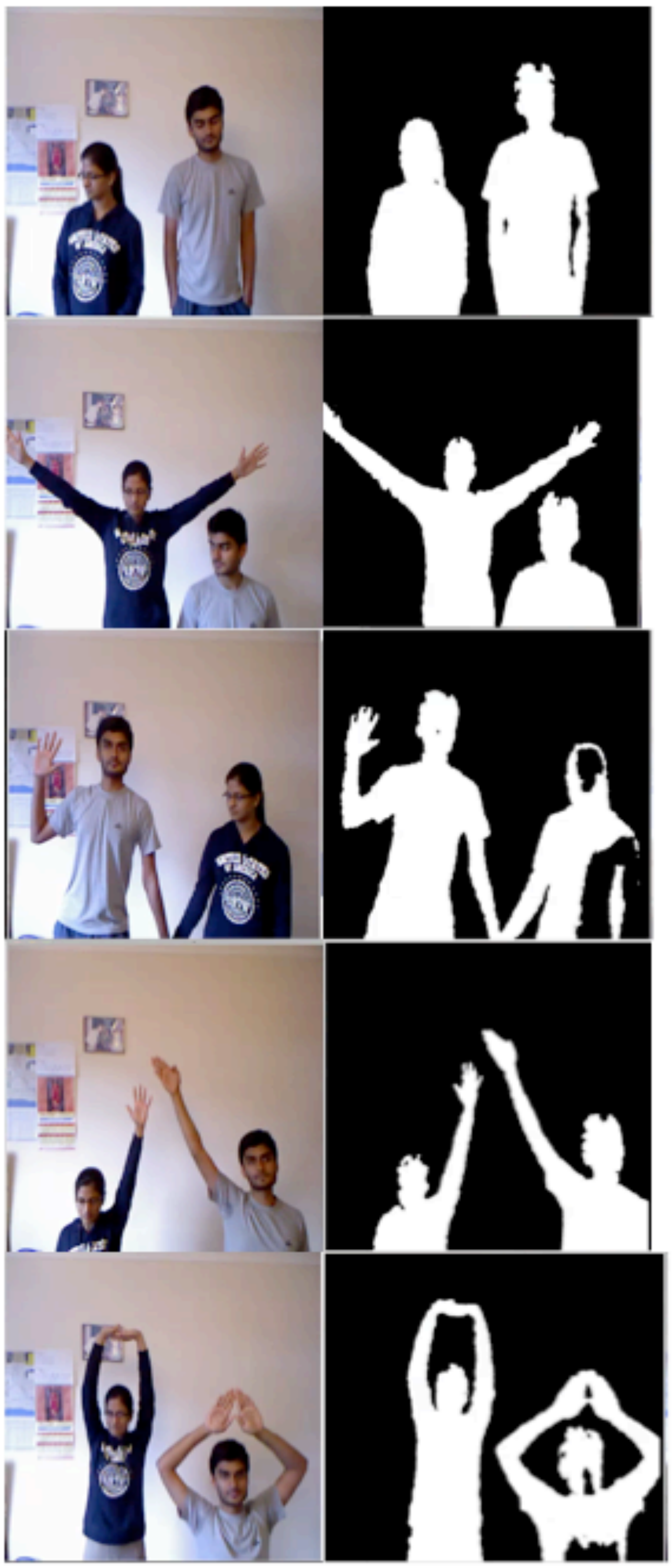


Fig. 9.5: Close Plane environment with two persons and without obstacle

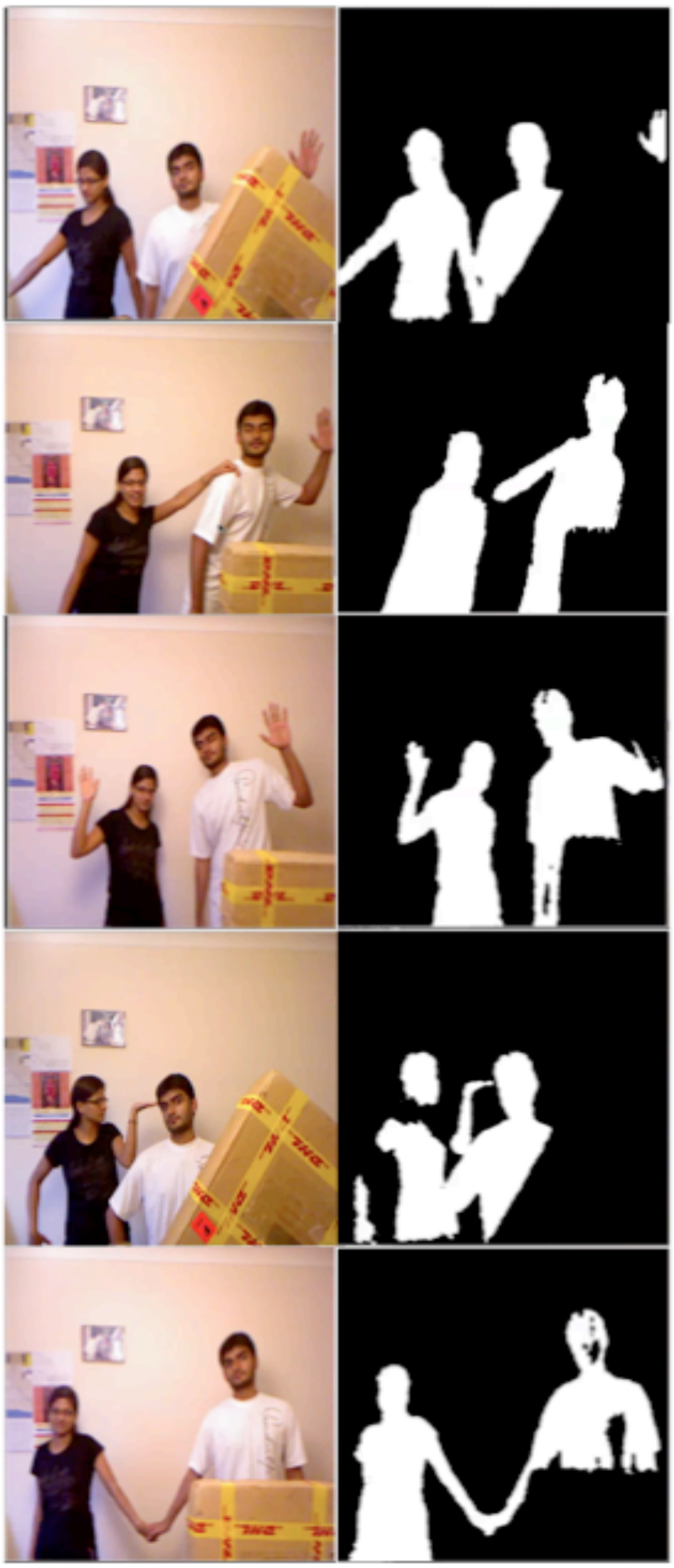


Fig. 9.6: Close Plane environment with two persons and with an obstacle

9.4. Far Plane Environment:

As the results were quite satisfactory in the “Close Plane” environment, the experiment was shifted to test in “Far Plane” environment. The living lounge was chosen as it had enough disturbances in the scene and had the far plane at a distance of 6m from the Kinect. Markers till 3m from the Kinect sensor were placed to mark the end of a metre. The Kinect sensor was placed in the same height, like the “Close Plane” environment, of 5½ feet from the ground plane. The obstacles used in this environment were much similar to those used in the previous environment. A large photo frame and stacked cartons were used as obstacles.

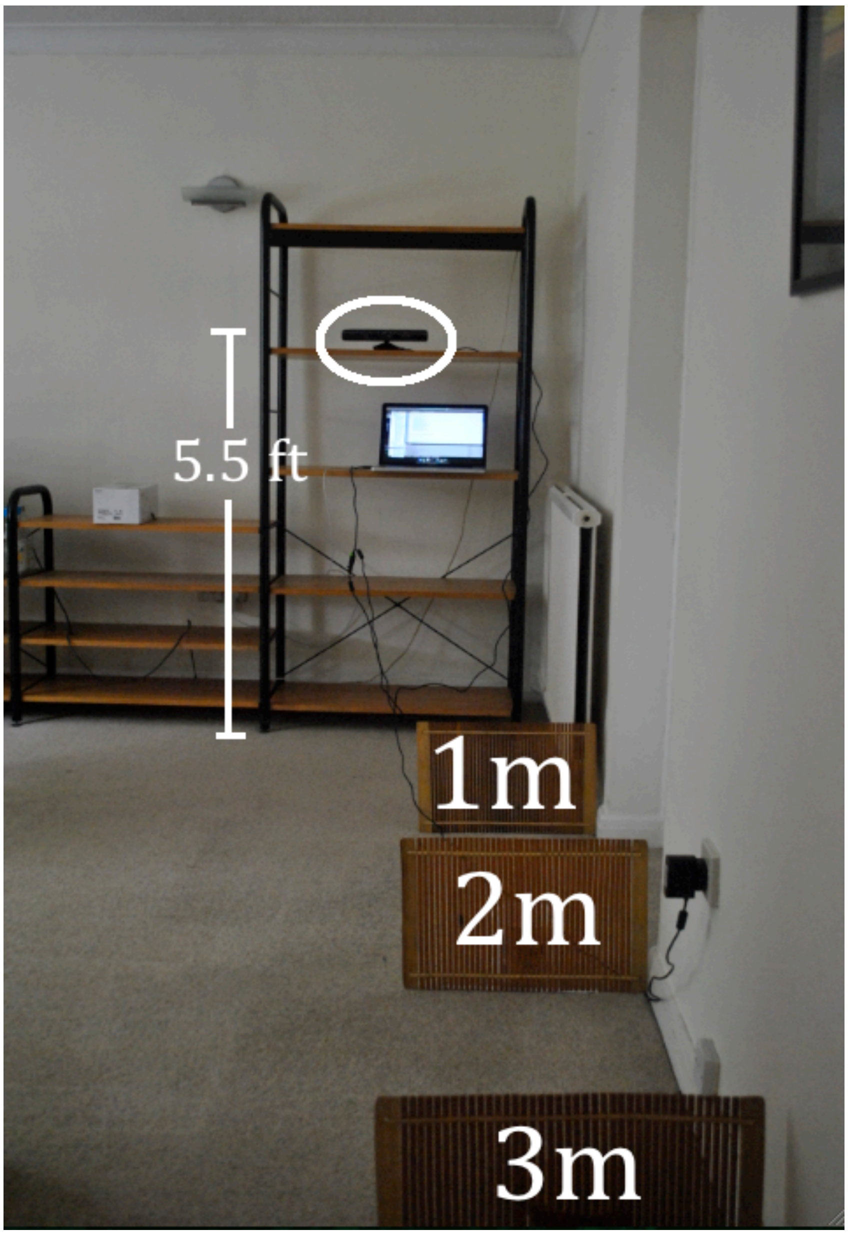


Fig. 9.7: Far Plane Environment. The markers are placed at every metre from Kinect

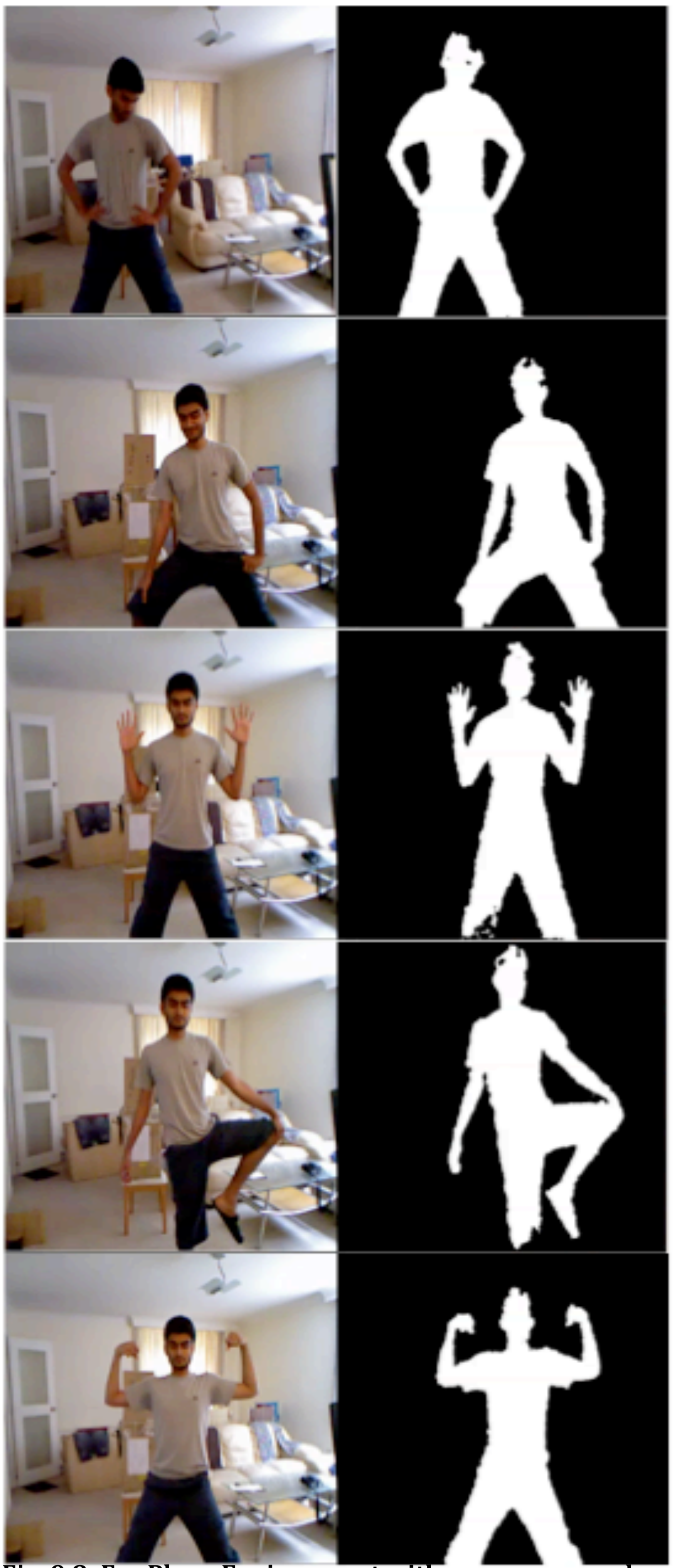


Fig. 9.8: Far Plane Environment with one person and no obstacle

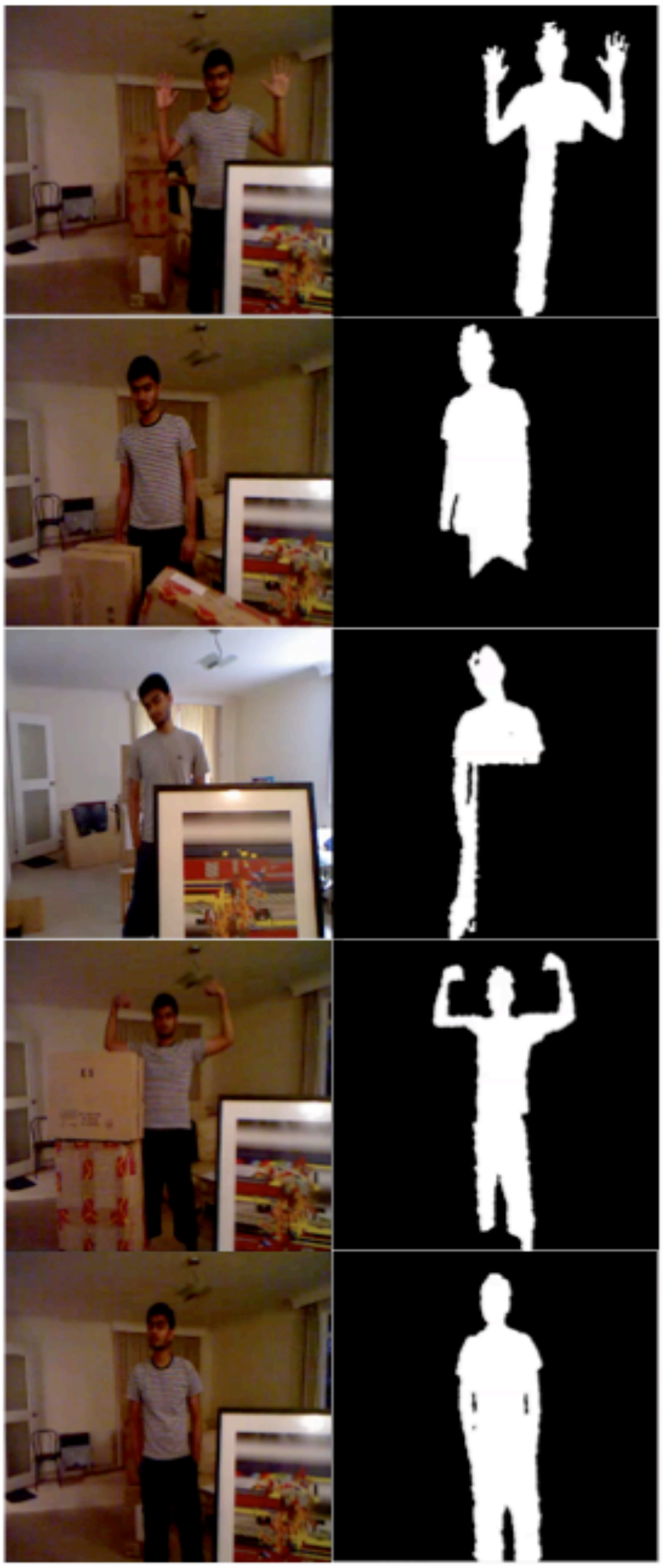


Fig. 9.9: Far Plane Environment with one person and obstacles

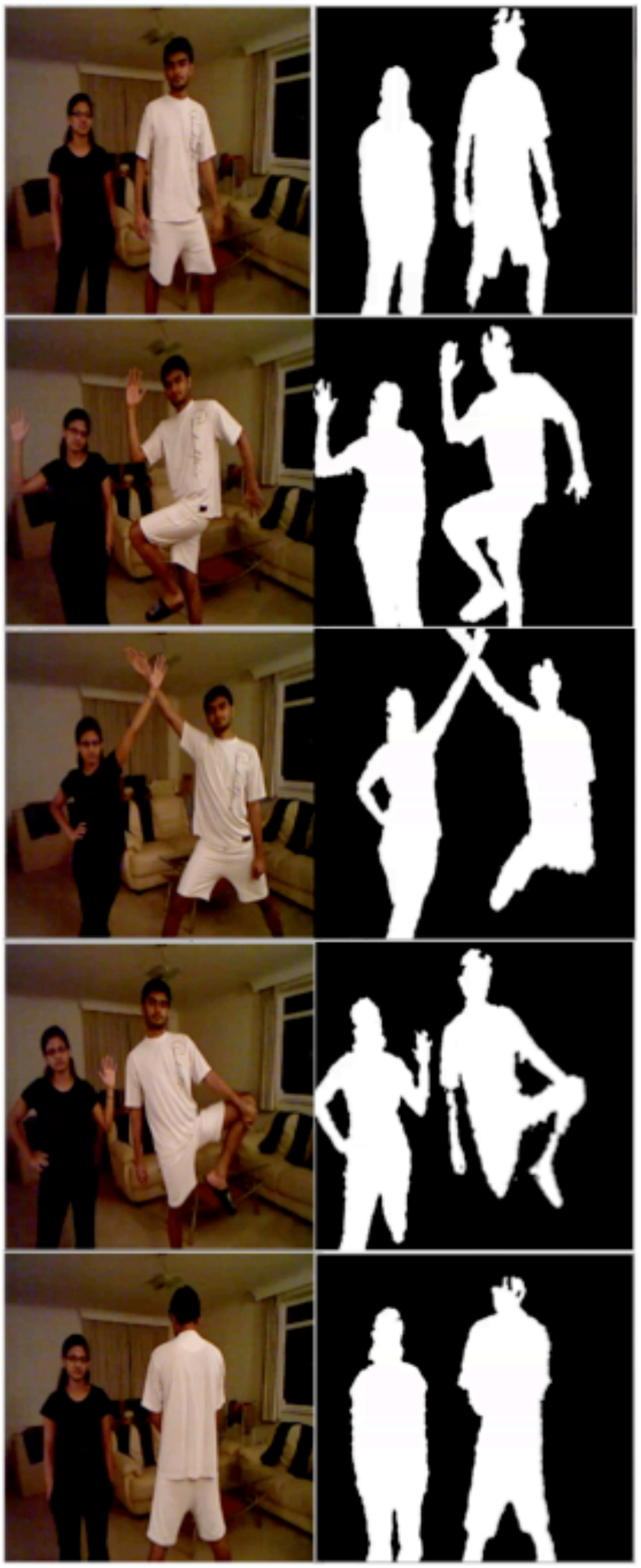


Fig. 9.10: Far Plane environment with two persons and with no obstacles

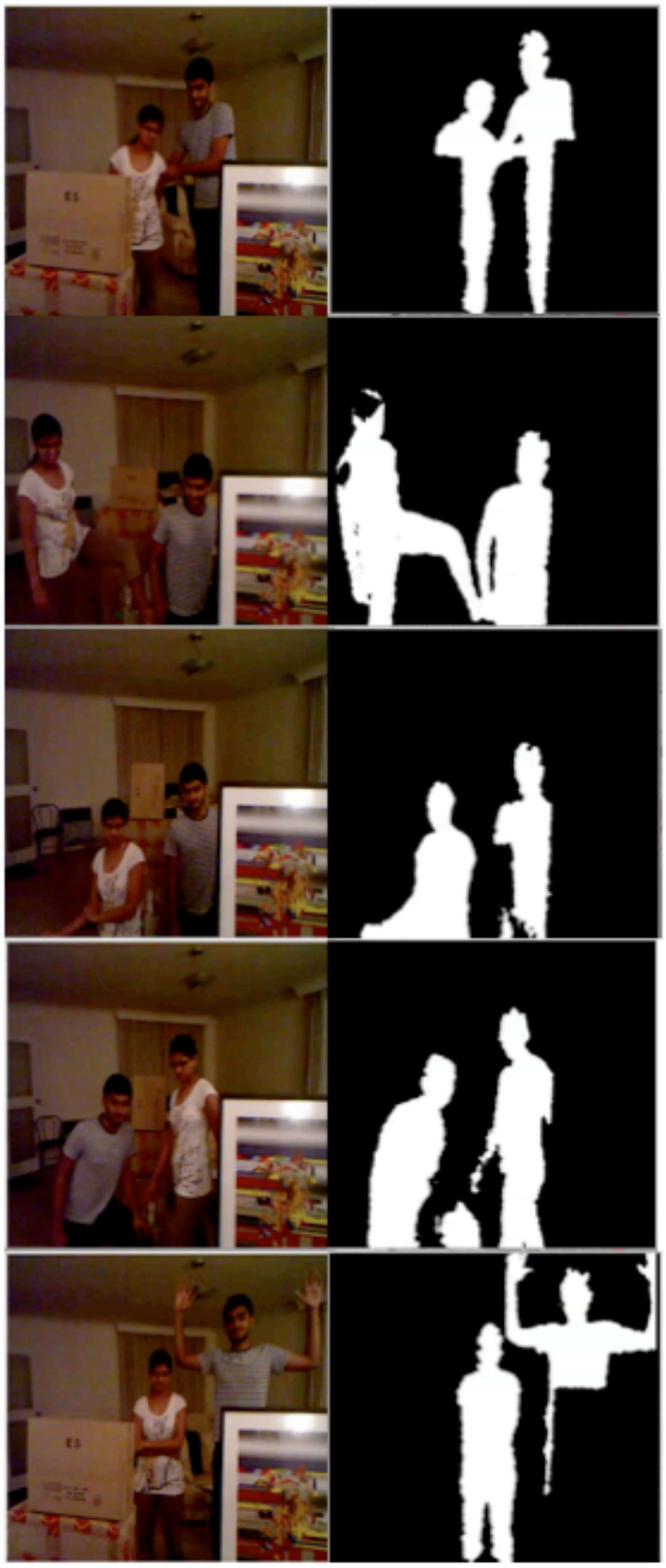


Fig. 9.11: Far Plane Environment with two persons and with obstacles

Figs. 9.8 to 9.11 show the results in Far plane environment. The results obtained were extremely satisfactory, though the processing time had the same average of 27.5 seconds. It was also inferred that, the best range within which the algorithm gives the best results is between 1m and 2m. In the range of 2m to 3m, the results became much noisy and were capable of capturing only one person. In some cases, the results become exactly opposite as shown in Fig. 9.13. Later it was discovered that it was because of the template like regions that had dominated the human head in the scene. In some cases, the Canny threshold values had to be decreased to reduce the weak edges in the disturbed environment. As the human figures, move away from the Kinect, the probability of getting exact detection results becomes extremely low. In case of the scenes involving two persons, the plane of detection of one human figure can differ from zero difference to a maximum of 0.75m. In some cases the planar regions were also getting detected. After analysing all these results, the following inferences have been made.

1. The best range of human detection, in both one person and two persons environment is found to be 1.5m to 2.1m from the Kinect sensor
2. The Canny edge threshold values must be as low as possible in any far plane and highly disturbed environment
3. The lesser the number of levels in the Pyramid, lesser is the accuracy of the detection results.
4. The detection gets poorer as the person moves away from the Kinect, and is not at all detected beyond 3m in some cases
5. The chance of the head template getting matched with the environments gets higher, if the environment is cluttered and filled with many objects
6. Best and consistent results are obtained, in “Close Plane” environment.
7. The consistency of the detection results in the “Close Plane” and the “Far Plane” environments are found to be 72% and 65% respectively

8. Some of the erroneous detection results were found to be
    a. None of the persons getting detected
    b. Areas other than human figure getting extracted
    c. Some parts of the human figures not getting extracted
    d. Exactly opposite detection

9. Detections get improper when the person’s head gets occluded with another person’s head.

The erroneous detections can be seen in Fig.9.12 and 9.13.

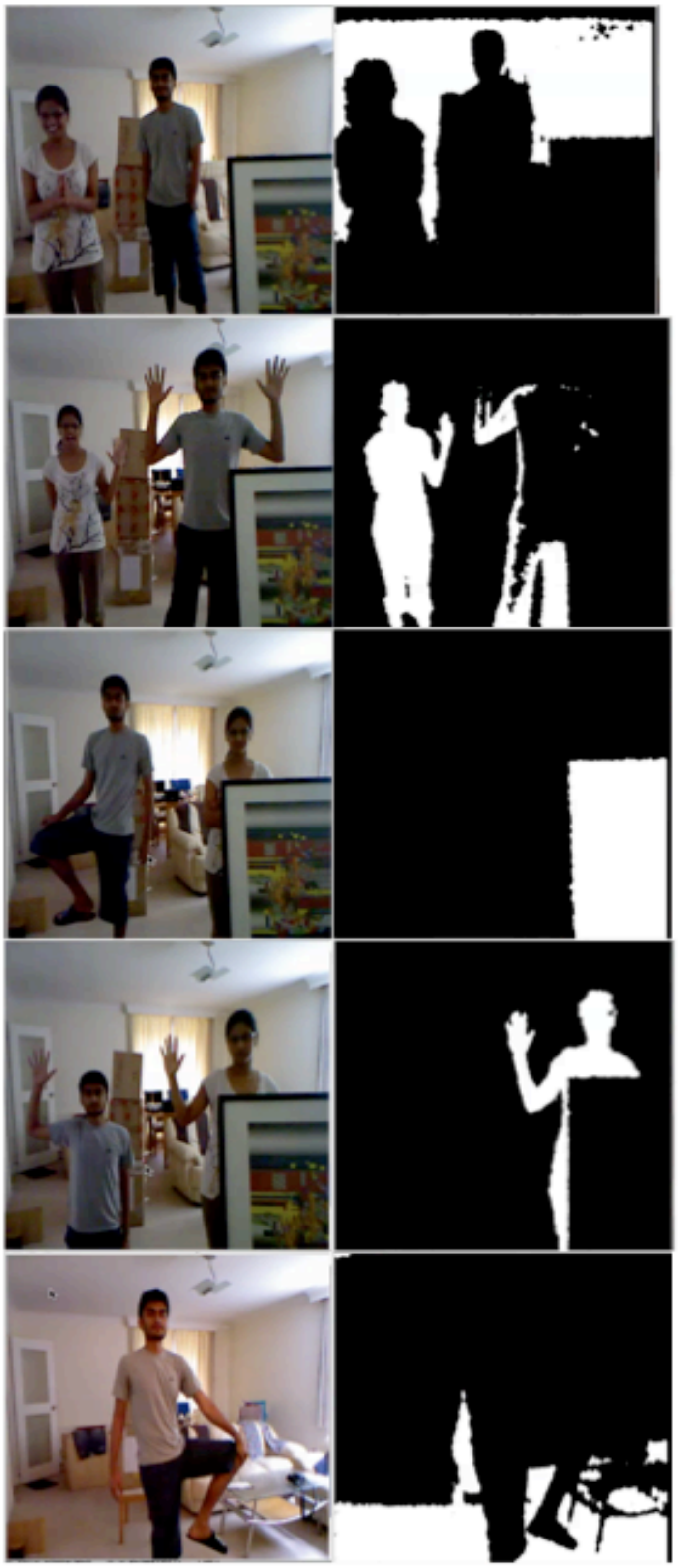


Fig. 9.12: Erroneous Detections

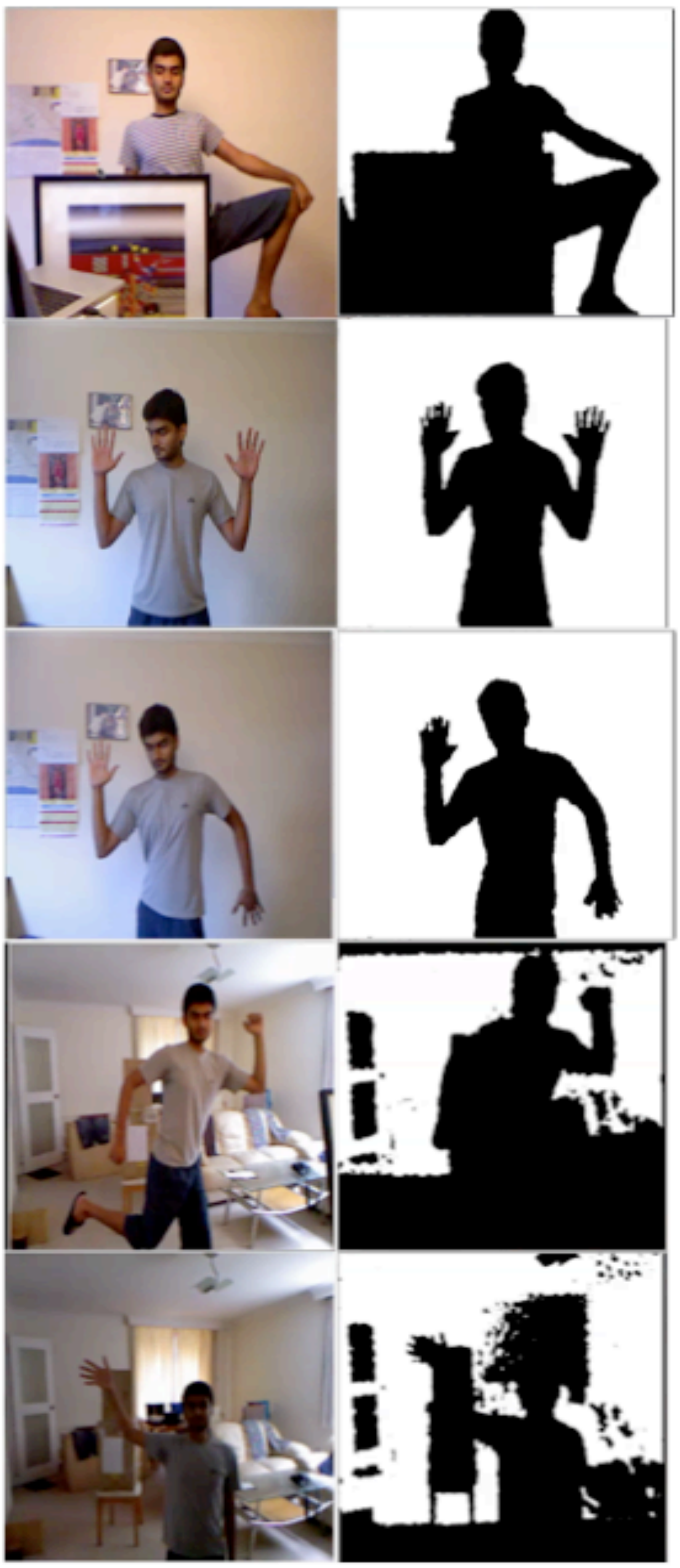


Fig 9.13: Erroneous Detections

Chapter 10

Future Work and Conclusion

The major improvement that needs to be done in the project is increasing the frame-processing rate. Though the number of pyramid levels increase, the processing should be improved and the detection should happen in as low as 0.4 frame/second. This will be achieved by implementing some calculations in GPU. Further, attention needs to be focused in the OpenGL drawing routine because the frames are drawn in the immediate mode. Efforts will be taken to implement the retained mode. Also the consistency of the human detection has to be improved, especially in far plane environment. Performing as many numbers of tests as possible in different locations and thereby achieving the relationship between the detection and the threshold values, the consistency can be improved.

Similar to head detection, other body part detectors have to be implemented, so that even if the person's head is not in the scene, the human contour captured in the scene could be detected. By thoroughly studying the square error values obtained as a result of the 3D Model fitting, the detection of the number of persons in the scene can be automated. In this project, however, only based on the user input, the maximum number of heads to be considered is decided.

As a result, the human detection and extraction from the Kinect depth images have been successfully implemented using C++ and OpenGL. Also, variety of tests has been carried out to study the behaviour of the detection algorithm. A new proposal on the modification of Distance Transform has been made. Further, this algorithm has proved to be faster than the usual iterative ones. The erroneous detections have also been analysed and the reasons for false detections were also found.

Bibliography

Burger,W. and Burge,M., 2008. Digital image processing. An algorithmic introduction using Java. New York: Springer

Crock,N., 2 March 2011. *Kinect Depth vs. Actual Distance*. Mathnathan. [Accessed: 8 August 2011]

Dalal,N., and Trigges,B., 2005. Histograms of oriented gradients for human detection. *CVPR*, 20-26 June 2005, San Diego, USA. Available from: http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf [Accessed: 8 August 2011]

Fisher,M., 2010. *Kinect*. Stanford:Stanford University. Available from: http://graphics.stanford.edu/~mdfisher/Kinect.html [Accessed: 8 August 2011]
Ganapathi,V., Plagemann,C., Koller,D. and Thrun,S., Real-time motion capture using single time-of-flight camera. ,*CVPR.* Available from: http://ai.stanford.edu/~koller/Papers/Ganapathi+al:CVPR10.pdf [Accessed: 8 August 2011]

Herrera,D., Kannala,J., Heikkila,J., 2011. *Accurate and practical calibration of a depth and color camera pair*. Oulu. Available from : http://www.ee.oulu.fi/~dherrera/kinect/2011-depth_calibration.pdf [Accessed: 8 June 2011]

Ikemura,S. and Fujiyoshi,H. 2011. Real-time human detection using relational depth similarity features, *ACCV*. Available from: http://www.vision.cs.chubu.ac.jp/04/pdf/PIA64.pdf [Accessed: 8 August 2011]

Kuntz,N., 2006. *Canny Tutorial*. Available from: http://www.pages.drexel.edu/~nk752/Research/cannyTut2.html [Accessed: 8 June 2011]

Levi,K. and Weiss,Y., 2004. Learning object detection from a small number of examples: the importance of good features*. Computer Vision and Pattern recognition*. Available from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.5.2657&rep=rep1&type=pdf [Accessed: 8 August 2011]

Lowe,DG., 1999. Object recognition from local scale-invariant features. *International Conference on computer vision*,1999. Available from: http://citeseer.ist.psu.edu/viewdoc/download:jsessionid = 8363255D9476F7D11ABA1FAB02667F33?doi=10.1.1.218&rep=rep1&type=pdf [Accessed: 8 August 2011]

Niles,N., 3 May 2011. *3D point cloud from Kinect images*. OpenCV-useres. [Accessed: 8 August 2011]

Sabata,B., Arman,F. and Aggarwal,J.K, 1993. Segmentation of 3D range images using pyramidal data structures, *CVGIP:Image Understangins*. Available from: http://cvrc.ece.utexas.edu/aggarwaljk/Publications/B.%20Sabata,%20F.%20Arman%20Segmentation%20of%203d%20range%20images.pdf [Accessed: 8 August 2011]

Schramm,M., 19 June 2010. *Kinect: The company behind the tech explains how it works*. Virginia:AOL. Available from: http://www.joystiq.com/2010/06/19/kinect-how-it-works-from-the-company-behind-the-tech/ [Accessed: 8 Aug 2011]

Sinha,U., 9 August 2010. *Convolutions*. AI Shack. Available from: sinha.utkarsh1990@gmail.com. [Accessed: 8 August 2011]

Ten,S., 30 January 2010. *How Kinect depth sensor works – stereo triangulation?* Mirror Image Mostly AR and Stuff. [Accessed 15 Aug 2011]

Weisstein,E., 2011. *Spherical Coordinates*. Mathworld – a wolfram web resource. Wolfram Research. Available from : http://mathworld.wolfram.com/SphericalCoordinates.html [Accessed" 8 August 2011]

WIKIPEDIA, 2011a. *Canny Edge detector* [online]. Wikimedia Foundation, Inc. Available from: http://en.wikipedia.org/wiki/Canny_edge_detector. [Accessed: 8 August 2011]

WIKIPEDIA. 2011b. *Taxicab geometry* [online]. Wikimedia Foundation, Inc. Available from: http://en.wikipedia.org/wiki/Taxicab_geometry [Accessed: 8 August 2011]

WIKIPEDIA. 2011c. *Chebyshev distance* [online]. Wikimedia Foundation, Inc. Available from: http://en.wikipedia.org/wiki/Chessboard_distance [Accessed 8 August 2011]

WIKIPEDIA. 2011d. *Distance Transform* [online]. Wikimedia Foundation, Inc. Available from: http://en.wikipedia.org/wiki/Distance_transform [Accessed 8 August 2011]

Xia,L., Chen,C., Aggarwal,J.K., 2011. Human detection using depth information by Kinect. *Workshop of aerial video processing in conjunction with CVPR* , June 2011 Colorado, USA. Available from: http://cvrc.ece.utexas.edu/Publications/HAU3D11_Xia.pdf [Accessed: 8 August 2011]

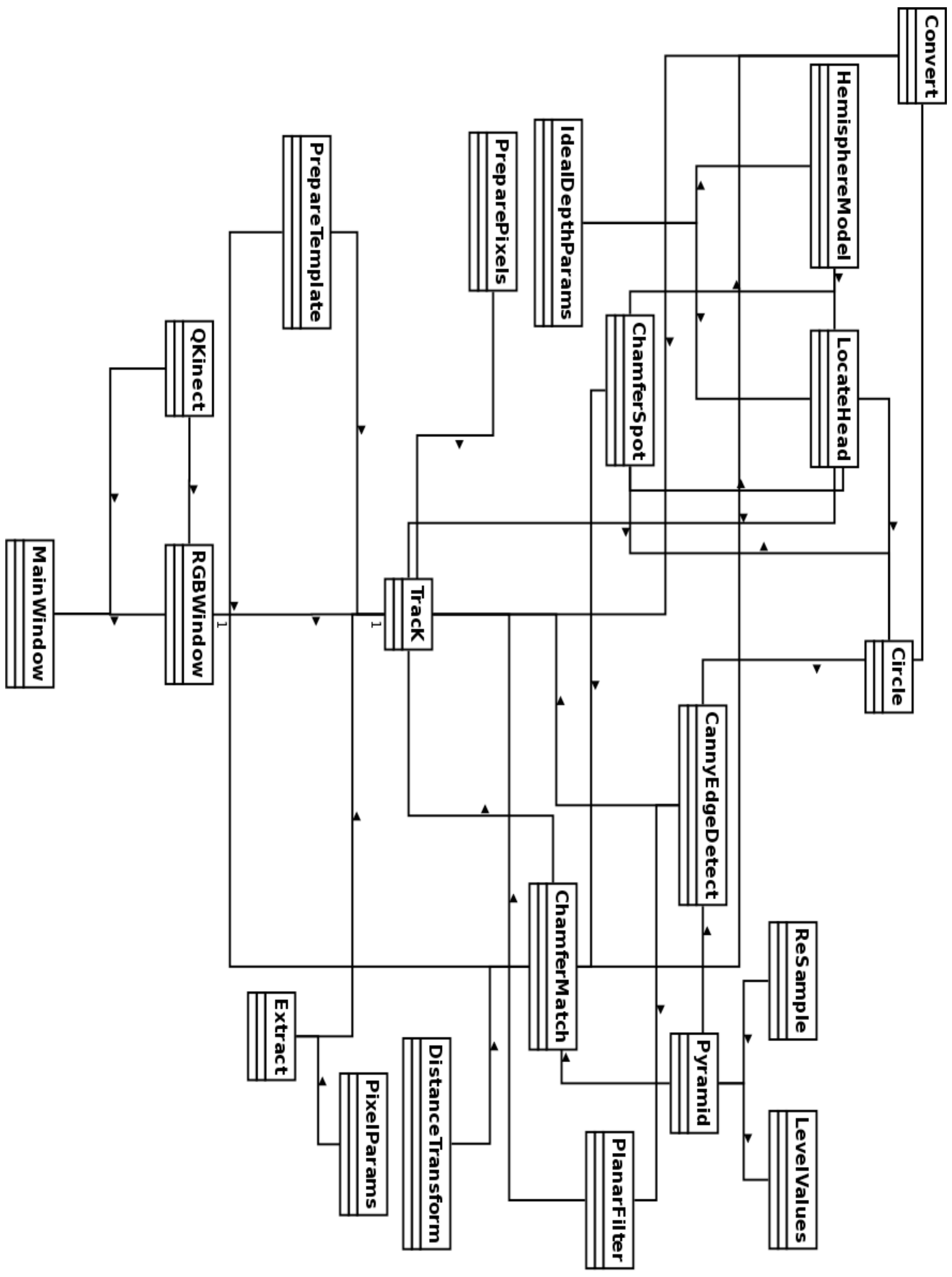
BIBLIOGRAPHY

Yang,HD. And Lee,S.W., 2007. Reconstruction of 3D human body pose from stereo image sequences based on top-down learning, *Pattern Recognition*. Available from: http://image.korea.ac.kr/publication/journal/07_04.pdf [Accessed: 8 August 2011]

Yet Another blogger, 31 December 2010. Chamfer Matching. OpenCV Adventure. [Accessed: 8 August 2011]

Appendix

Class Diagram Overview


MainWindow
RGBWindow
QKinect
PrepareTemplate
PreparePixels
IdealDepthParams
HemisphereModel
Convert
ChamferSpot
LocateHead
Track
Circle
CannyEdgeDetect
ChamferMatch
ReSample
Extract
PixelParams
DistanceTransform
Pyramid
PlanarFilter
LevelValues
1
1