CSE/EE-576, Final Project

– Torso tracking –

Ke-Yu Chen

Introduction

Human 3D modeling and reconstruction from 2D sequences has been researcher's interests for years. Torso is the main part of the human body and connects to our limbs (through shoulder joints) and legs (through pelvic joints). Finding the accurate position of torso is important in human 3D modeling and reconstruction. In recent years, many products, such as Kinnect [5], provide excellent performance of human 3D modeling and reconstruction by multiple cameras. However, when high-tech devices are not accessible or old videos recorded by a single camera are the only data source, we need to explore this kind of reconstruction process from 2D sequences purely captured by a single camera.

For this purpose, I designed an algorithm to track the torso of a human body in real time from a single general-purpose camera. The algorithm exploits silhouette of the human body and extracts the torso with distance transform [1] and dynamic threshold techniques.

Related work

Silhouette can tells us a lot of things. *Siddiqi* [3] addressed some general principles for partitioning objects from a silhouette and used computational supports for showing that link-based and neck-based parts are invariant, robust and stable and yield hierarchy of parts. *Gorelick* [4] proposed a function to extract many useful properties of a silhouette, such as torso, corners and limbs, by solving Poisson equation. Both these papers presented comprehensive studies based on silhouette.

To set an appropriate scope of the final project, I focused on extracting the human torso in a stationary background. A different approach, `Distance transform` [1], is used to calculate weights of silhouette pixels. The smoothing equation derived in Gorelick's paper [4] is then applied to enhance the result images. The system architecture is shown in figure 1. All steps are described in the following sections.

My Algorithm

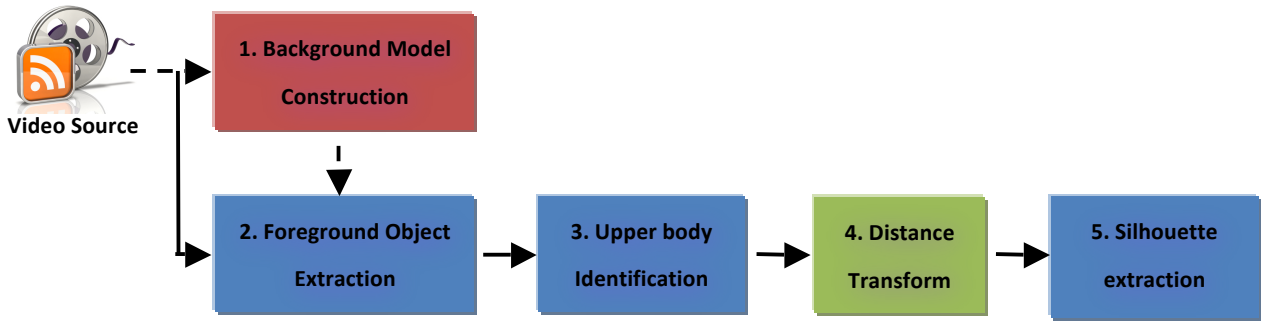


Fig 1. System architecture

1. Background model construction

The first step is to build a background model that will be used in foreground object extraction (step 2). The background model construction includes two sub steps:

- *Drop the first 50 frames*
  When a webcam is turned on to record a video, it requires a few seconds for initialization. In this initialization stage, the illumination of images is not stable and image frames may contain a lot of noise. Therefore, the first 50 frames are "dirty" and being dropped – the system does not use them.
- *Buffer the following 10 frames for building background model*
  The system then buffered 10 frames and take the *average* image of them as the background model:

$$\text{Background model image} = (\text{frame}_{61} + \ldots + \text{frame}_{70}) / 10$$

Fig 2. Background model image

2. Foreground object extraction

The foreground object (i.e. human silhouette in this project) is extracted by *subtracting* the current frame from the background model frame. Image subtraction is a simple and efficient method to obtain the foreground subject in a stationary background, but may generate non-perfect silhouette because of:

- shadow pixels
- internal fragmentation due to non-recognizable foreground pixels
  (when a foreground pixel is similar to a background pixel, we will get a "black" pixel after image subtraction and this foreground pixel will be identified as "background".)

Figure 3 shows an example of a non-perfect silhouette. The left image is the original frame and the right one is the corresponding fragmented foreground image after image subtraction. A fragmented silhouette will deteriorate the performance of distance transform (step 4), which will lead to an incorrect torso identification (step 5). To get a better silhouette, we need extra steps to remove shadow pixels and "fill in" the fragmented parts.

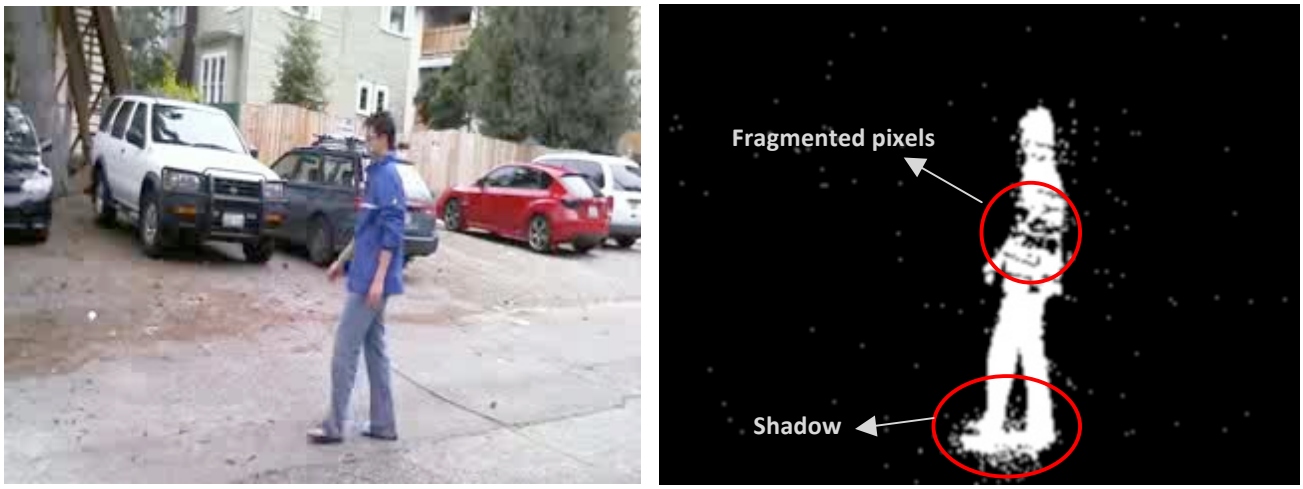


Fig 3. Current frame (left) and the corresponding fragmented foreground image (right)

2.1 Shadow detection and removal

I follow Horprasert's work [6] for shadow detection and removal. The basic idea is that a shadow pixel should have RGB values similar to its corresponding background pixel with lower illumination. In figure 5 (in the next page), the blue dotted line represents the background pixel and the orange line is the shadow pixel. These two pixels have similar RGB values (close to each other in the RGB space) but different illumination (vector length is different).

I threshold at two parameters to identify a shadow pixel: 1) the angle between two vectors, and 2) the difference of vector length. A pixel is identified as a shadow pixel if

```
Angle (radius) < 0.2 && difference of vector length between 0~10
```

Since I used fixed threshold value and the outdoor videos contain a lot of noises, the method failed to detect shadow pixels in an outdoor video. The indoor videos, on the contrary, have a better background model and therefore, the shadow pixels can be detected more easily. Figure 4 shows the shadow detection result of one frame of my indoor videos.

Fig 4. Original image (left) and the shadow detection (right)

The white pixels in the right image represents the detected shadow pixels

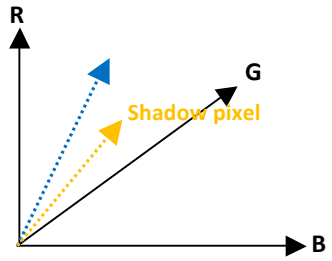


Fig 5. Shadow detection in RGB model

2.2 "Fill in" fragmented parts

I perform `closing reconstruction` [8] on the silhouette to fill the fragmented parts. The difference between `closing reconstruction` and the normal `morphological closing` is that `closing reconstruction` tries to keep the same silhouette after the closing operation. Figure 6 shows my implementation. Then, `Gaussian filter` is applied on the image for smoothing the contour.

```
void ImgOperation::closeRec(IplImage* pImg, IplImage* pCloseRecImg, IplConvKernel* ker)
{
    IplImage* pDilateImg = cvCreateImage(cvSize(WIDTH,HEIGHT),IPL_DEPTH_8U,1);
    int iSteps = 4;                        // Iterations to converge
    cvDilate(pImg, pDilateImg, ker, 1);
    cvMax(pImg, pDilateImg, pCloseRecImg);
    for(int i=0 ; i<iSteps ; i++){
        cvErode(pCloseRecImg, pCloseRecImg, ker, 1);
        cvMax(pCloseRecImg, pImg, pCloseRecImg);}
    cvReleaseImage(&pDilateImg);
}
```

Fig 6. Closing reconstruction

Figure 7 shows the silhouette after exercising the `closing reconstruction`. Compared with figure 3, the fragmented parts disappear and the shape of silhouette keeps the same.

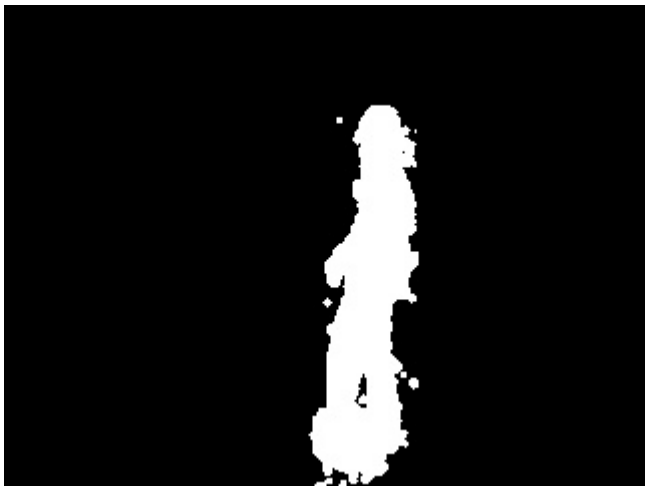


Fig 7. Foreground image after closing reconstruction

3. Upper body identification

If we use the whole body silhouette in the distance transform (step 4), the extracted silhouette will not be accurate and become a strip shape. To get a better result in step 4, I identified the rough upper body part by a simple dynamic thresholding technique, as described in figure 8. In the algorithm, the parameter, `box`, identifies the width and height of the rectangle that contains the whole human silhouette. If the shape of this rectangle is close to a strip (i.e. the person is walking through the camera), we need to cut a shorter part (`dRatio` = 0.5) from the whole silhouette as our upper body. When the shape of this rectangle is close to a square (i.e. the person is walking toward or away from the camera), we need a larger portion (`dRatio` = 0.7).

```
// Set up the thresholding
double dRatio = box->size.width / box->size.height;
if( dRatio < 0.2)
     dRatioHeightROI = 0.5;
else if ( dRatio < 0.4)
     dRatioHeightROI = 0.6;
else
   dRatioHeightROI = 0.7;
```

Fig 8. Upper body identification by dynamic thresholding

Figure 9 shows an example when the `dRatio` = 0.6 is chosen:

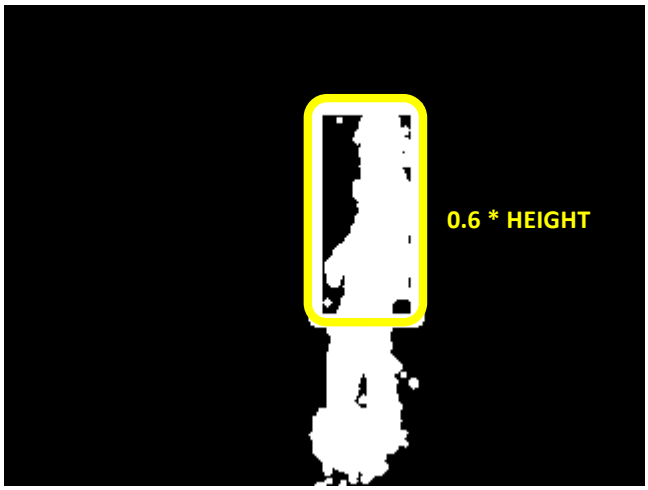


Fig 9. Upper body identification by dynamic thresholding

4. Distance transform

`Distance transform` `(DT)` [1] is performed to calculate the weight of each silhouette pixels. The "weight" means the minimum distance between a pixel and its nearest contour pixel. Pixels closer to the silhouette center get higher weight (score). Luckily, `OpenCV` [2] has the implementation of `distance transform`. Figure 10 shows an example of `distance transform` and the silhouette after `DT`.

1	1	1	1	1
1	2	2	2	1
1	2	3	2	1
1	2	2	2	1
1	1	1	1	1

Figure 10. Example of distance transform (left) and the silhouette after distance transform (right)

As you can see, the contrast of the resulting silhouette is not visually observable and may be difficult to find an appropriate threshold to extract our interested part (torso). To enhance the image, I adopted the Poison Equations [7]:

$$U(x,y) = 1 + \frac{1}{4}\Big(U(x+h,y) + U(x-h,y) + U(x,y+h) + U(x,y-h)\Big)$$

The new weight of a pixel, `U(x,y)`, is the average of four neighboring pixels *plus one*. It enhanced the peak value of the silhouette and enlarged the difference between neighboring pixels. Then, all pixels are normalized into 9 scales ranging from 0 to 255. Figure 11 displays the silhouette after performing the Poisson Equation and pixel scaling. The rectangle marks the `SOI` (`Silhouette of Interest`) that is decided in the dynamic thresholding in step 3. The silhouette is more visually observable now.

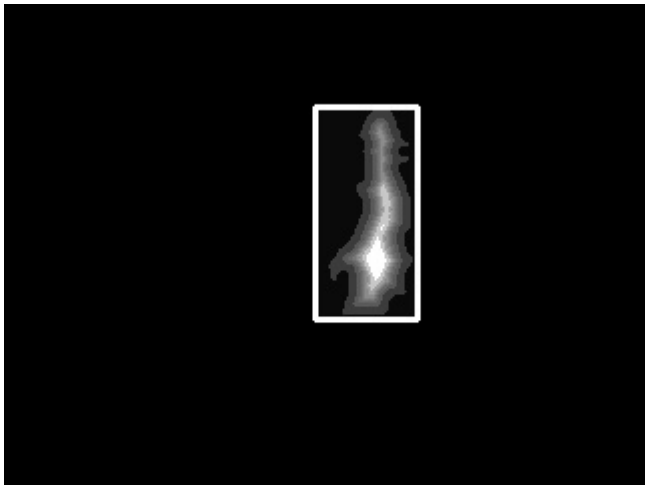


Fig 11. Silhouette after performing Poisson Equation

5. Silhouette extraction

The last step is to extract the torso part from the human silhouette. First, all pixels are normalized to [0, 1]. Pixels with values under the threshold **0.55** are *cut off*, that is, set as 0 (black). Remaining pixels are identified as the torso and set to 255 (white). The result is a binary image, as shown in figure 12 (left). Finally, the torso part is plotted on the original frame with a red rectangle (figure 13).

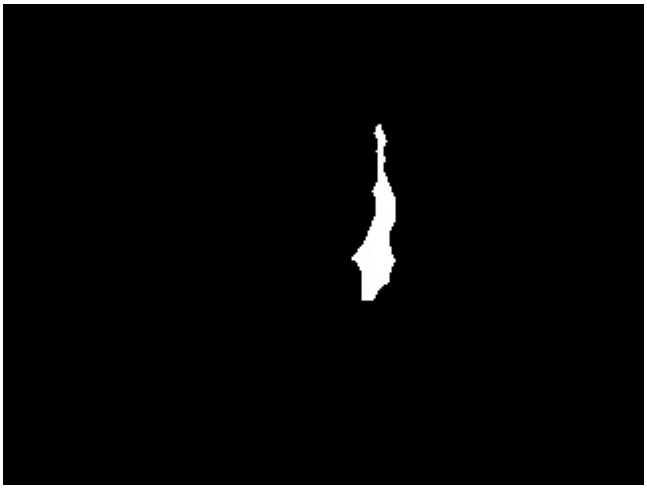


Figure 12. The extracted silhouette after thresholding

Figure 13. The final result (red rectangle represents the extracted torso)

Experiments and results

1. Datasets

I recorded 6 videos for testing, as shown in Table 1. The dataset includes two indoor videos and four outdoor videos so that I can compare the results in different environments. All outdoor videos (ID 1 ~ 4) and the first indoor video (ID 5) contains the same foreground object (i.e. myself). The second indoor video (ID 6) contains a different foreground object (a friend of mine) to show how the algorithm adapts in different size of people.

ID	*Video name*	*Duration (second)*	*Description*
1	**test_outdoor1.avi**	20	Person 1 walks from right to left
2	**test_outdoor2.avi**	22	Person 1 walks from left to right
3	**test_outdoor5.avi**	22	Person 1 walks in a S-shape track
4	**test_outdoor6.avi**	42	Person 1 walks randomly and out of screen once a while
5	**test_indoor1.avi**	18	Person 1 walks in a S-shape track
6	**test_indoor2.avi**	21	Person 2 walks in a S-shape track

Table 1. Testing videos and their descriptions

You can download all videos at **http://emmanuel.cs.washington.edu/CSE576/2011-Spring_DataSet.zip**.

2. Performance evaluation

I aimed to design a real-time system, so `FPS (frame per second)` should be at least 24 - 30. The program ran on a qual-core machine (Intel i5 750) with 6GB memory. The `FPS` is shown in table 2. It shows this algorithm has no problem to be executed in real time.

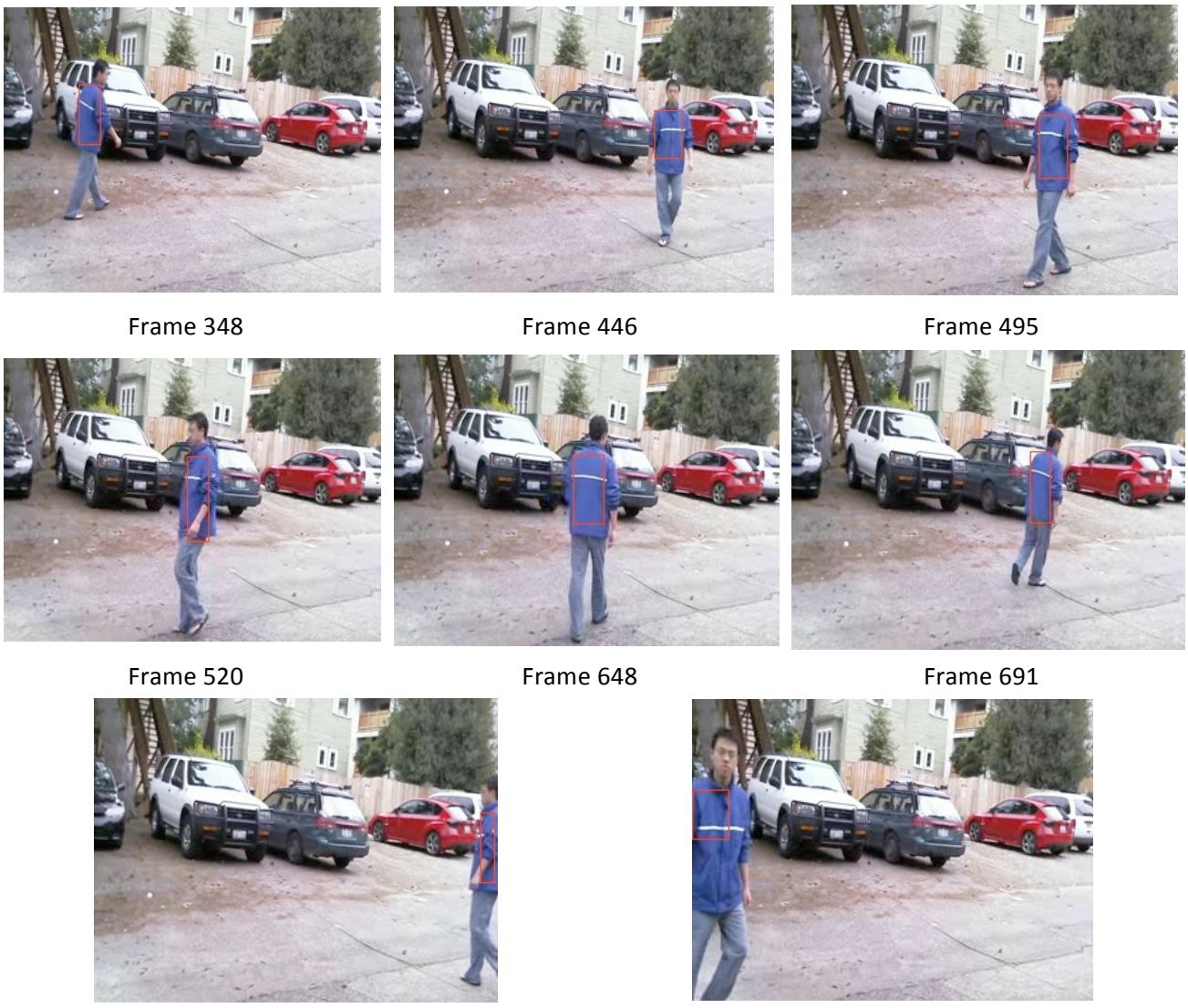
Since I made the videos by myself and there is no ground truth (i.e. the real positions of four corners of the torso), I cannot present the accuracy of my torso tracking algorithm. I listed it in the future work (please see next section).

Video name	*Frame rate (Unit: frames per second)*
test_outdoor1.avi	46.08
test_outdoor2.avi	26.04
test_outdoor5.avi	23.35
test_outdoor6.avi	35.03
test_indoor1.avi	31.69
test_indoor1.avi	40.00

Table 2. Real-time testing (unit: `FPS`)

3. Results and analysis

For the outdoor testing, I pick up `test_outdoor6.avi` to show my results since this video combines all angle views of the human silhouette. In some frames, the person left the screen and came back and the algorithm still can keep track of it. Figure 14 contains some selected frames from different angle views.



Frame 348 Frame 446 Frame 495

Frame 520 Frame 648 Frame 691

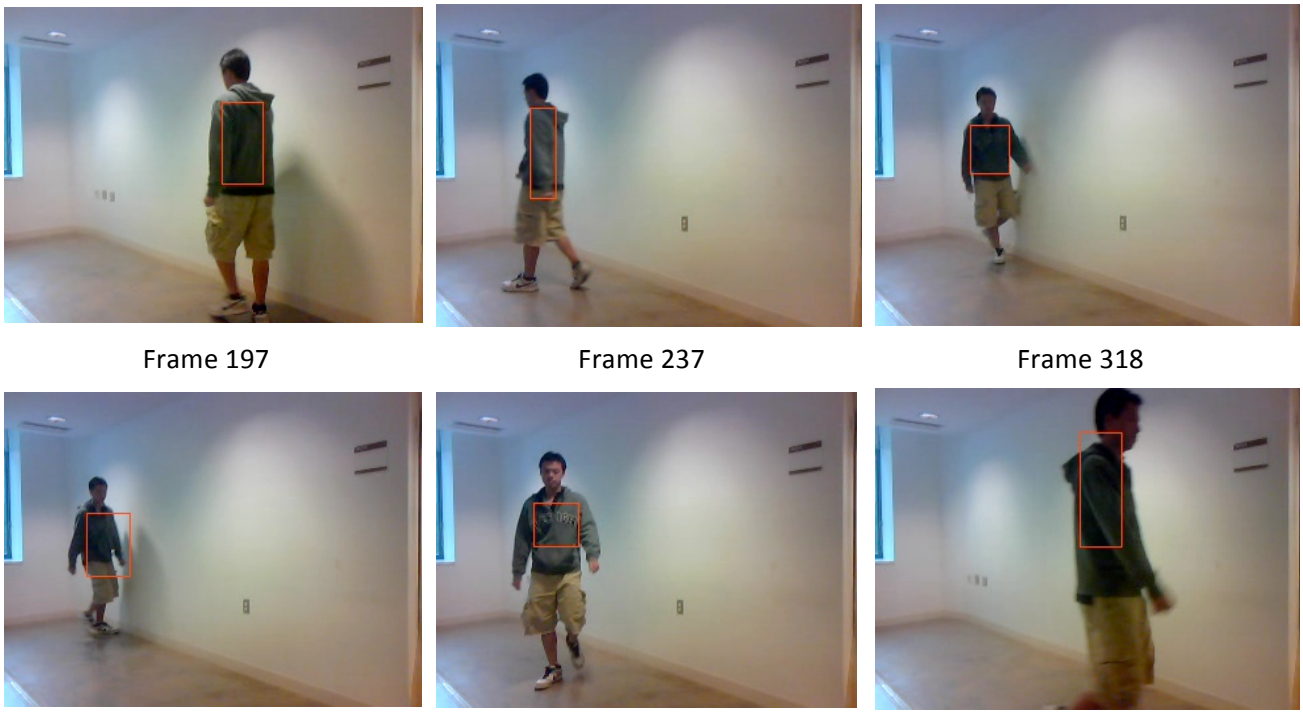
Frame 833 Frame 588

Figure 14. Selected results of the outdoor video (`test_outdoor6.avi`)

The images in figure 14 show that the algorithm can successfully identify the torso in different views. `Frame 833` is the first frame after the person left the screen for a few seconds and walked back. It shows that the algorithm can keep tracking the torso even if the object left and re-joined the video some time later. However, when the object is not completely in the screen (`frame 588`), the algorithm can not get correct torso location sometimes. It may be caused by an inappropriate threshold (in step 3), by which an incorrect upper body is extracted. This incorrect upper body crashed the result of distance transform (in step 4).

I also tested this algorithm in the indoor videos. Figure 15 shows some selected frames from `test_indoor1.avi`. Overall, this algorithm performed worse in the indoor environment. Some frames still work well (`frame 197`, `237` and `318`) but some frames suffered from different issues:

- `Frame 306`: shadow pixels on the wall are not removed so they become part of the silhouette, which make the torso part "shift" to the shadow a bit
- `Frame 349`: Indoor environment has stable background and suffered less fragmented silhouette problem. When performing the closing reconstruction (p.4) on a non-fragmented silhouette, the algorithm enlarges the original silhouette, which gives a smaller torso after the distance transform. The closing reconstruction should be adaptive to different environment (see the future work).
- `Frame 395`: This problem is the same as the one in outdoor videos. When the person is not completely in the screen, the algorithm may not be able to identify correct part of the upper body and crashed the result of distance transform.



Frame 197 Frame 237 Frame 318

Frame 306 Frame 349 Frame 395

Figure 15. Selected results of the outdoor video (`test_indoor1.avi`)

4. Demo videos

All demo videos are uploaded to Youtube and can be viewed through the following link:

- test_outdoor1: http://www.youtube.com/watch?v=62y43EG0wLY
- test_outdoor2: http://www.youtube.com/watch?v=EMZPJcvl0TQ
- test_outdoor5: http://www.youtube.com/watch?v=2i6yIDFw0rc
- test_outdoor6: http://www.youtube.com/watch?v=DWy3ac9HoKY
- test_indoor1: http://www.youtube.com/watch?v=0fcRYeXCcR8
- test_indoor2: http://www.youtube.com/watch?v=ROWod1CBaT8

Future work

- There should be another program to get the ground truth. The user can click the location of torso on each frame and uses the positions (two shoulder and two pelvic joints) as the ground truth.
- The algorithm can only track single object in the video. It could be improved to track multiple objects by using feature descriptors.
- I used static threshold in shadow detection so in some case (shadow on the wall), the detection results are pretty poor. This could be possibly improved by looking at the histogram and choosing dynamic threshold in each frame.
- Closing reconstruction is important in outdoor environment since it successfully fill a fragmented silhouette. However, it hurts the tracking results in an indoor video because it deform (enlarge) the original perfect silhouette. It should be modified so as to be adaptive in different environments.

Summary and conclusion

In this project, I designed and implemented an algorithm that tracks the human torso from 2D sequences. This algorithm can be executed in real time in a stationary background. Image subtraction is used to retrieve the foreground objects. The human torso is then extracted by Distance Transform with an appropriate threshold. The results show the algorithm works much better in an outdoor environment because there is no shadow-on-the-wall problem in an outdoor environment. Since the nature of difference between indoor and outdoor, it would be interesting but challenging to improve this system to be adaptive in both surroundings.

Reference

[1] Distance transform: http://en.wikipedia.org/wiki/Distance_transform

[2] OpenCV: http://opencv.willowgarage.com/wiki/

[3] Siddiqi, K.; Member, S. & Kimia, B. B. Parts of Visual Form: Computational Aspects IEEE Transactions on Pattern Analysis and Machine Intelligence, 1995, 17, 239-251

[4] Gorelick, L.; Galun, M.; Sharon, E.; Basri, R.; Society, I. C.; Society, I. C. & Br, A. Shape representation and classification using the poisson equation In In Proc. of CVPR'04, 2004, 61-67

[5] Kinnect: http://www.xbox.com/en-US/Kinect/healthyfun

[6] Horprasert, T.; Harwood, D. & Davis, L. S. A statistical approach for real-time robust background subtraction and shadow detection 1999, 1-19

[7] Gorelick, L.; Galun, M.; Sharon, E.; Basri, R.; Society, I. C.; Society, I. C. & Br, A. Shape representation and classification using the poisson equation In In Proc. of CVPR'04, 2004, 61-67

[8] Closing by Reconstruction: Closing by Reconstruction: http://artis.imag.fr/Members/Adrien.Bousseau/morphology/morphomath.pdf