Magic Mirror: A Virtual Handbag Shopping System

Lu Wang, Ryan Villamil, Supun Samarasekera, and Rakesh Kumar
SRI International Sarnoff
201 Washington Road, Princeton 08536, USA
lu.wang@sri.com

Abstract

We present an augmented reality system based on Kinect for on-line handbag shopping. The users can virtually try on different handbags on a TV screen at home. They can interact with the virtual handbags naturally, such as sliding a handbag to different positions on their arms and rotating a handbag to see it from different angles. The users can also see how the handbags fit them in different virtual environments other than the current real background . We describe the technical details for building such a system and demonstrate the experimental results.

1. Introduction

With the advance in high speed Internet, network security and privacy, on-line e-commerce market is growing rapidly in the last decade, including Internet fashion shopping. Over a third of consumers in US bought some of their apparel, footwear and accessories over the Internet in 2011. For customers, they usually would like to try on the products before placing an order; for retailers, giving the customers a way to try their products before shipping is important to minimize the return rate and reduce cost. Therefore, the technologies, often called virtual fitting room, that allow customers virtually try on different fashion products in front of a TV screen or a computer monitor become very important and have attracted intensive interests from researchers.

These technologies can be divided into two kinds: virtual reality [6, 10, 11] and augmented reality [7]. In the first kind, 3D body models of users are created beforehand and are loaded into computer memory as rigid 3D objects. Virtual garments and accessories are displayed on top of the 3D body models, and the users can see the wearing effects from different angles by rotating the 3D models with a mouse or a keyboard. In some approaches, the 3D body models are created from a 3D mannequin adjusted to fit the body measurements of the users. More advanced techniques, such as structured light and 3D laser scanners, can be used to construct more accurate 3D body models, but they are usually too expensive to be used at home.

Compared to virtual reality, the approaches based on augmented reality in which virtual garments are directly displayed on the body of the users in real videos are more realistic and convenient. However, they require accurate real-time 3D body tracking and pose estimation, which is typically the bottleneck of these augmented reality systems.

The Kinect sensor [5] introduced by Microsoft in 2010 is evidently a breakthrough that brings low-cost 3D depth sensing and human body pose estimation technologies to the general public. Since its launch date, it has been exploited in all kinds of computer gaming and augmented reality applications including those for virtual fitting room. Bodymetrics [1] created a system in which a 3D avatar is generated for each user whose body shape and size are measured with Kinect. The user can control the 3D avatar through Kinect and see how the virtual clothes look on the avatar in different poses. Fitnect [2] also provides a system for users to try on different clothes, but instead of using an avatar, the virtual clothes are directly displayed on the user's body in augmented videos. Facecake [3] gives a similar augmented reality system called Swivel in which users can virtually try on a variety of fashion products including handbags, clothes, necklace and sunglasses.

We present an augmented reality system for on-line handbag shopping by using Kinect with the OpenNI API [4]. We call it Magic Mirror. Compared to the Swivel system of Facecake, with our system the users can not only hold the virtual handbags in their hands but also can move the handbags to different positions on their arms and see how the handbags look on their elbows or shoulders. To implement this, we need to explicitly handle the physical interaction between the human body and the 3D models of virtual handbags.

In addition, our system has a teleportation function in which the background of the video can be switched from the real room to a virtual environment, such as a runway or a hall, so that the users can see how the handbags fit them in different situations. To do this, an important issue is to obtain the accurate segmentation of human body from the

978-1-4673-1612-5/12/$31.00 ©2012 IEEE

real image in order to insert it into a virtual background image. Although OpenNI gives a segmentation of human body from the background, the boundary is usually not accurate due to several possible reasons such as the limitation of the quality of the depth map and the slight delay between the generation of the depth map and the color image. We present a novel approach designed for Kinect applications that can refine the coarse segmentation to obtain more accurate boundaries.

The rest of the paper is organized as follows: Section 2 gives the details on implementing the handbag sliding control. Section 3 describes how to compute the pose of a virtual handbag given the current kinect readings so that its rendering looks real. The segmentation algorithm to obtain accurate human body boundaries for teleportation is presented in Section 4. Some experimental results are demonstrated in Section 5, and the paper is concluded in Section 6.

2. Handbag sliding control

Our system requires a Kinect sensor. To start, a user stands in the initial calibration pose defined in OpenNI [4] for a couple of seconds. Once the user is detected, he (or she) can select virtual handbags on a screen with gestures. The selected virtual handbag will initially be displayed in one of his hands. When the user raises his arm, the virtual handbag will slide to his elbow or shoulder depending on the slope of his forearm and upper arm (the part between shoulder and elbow); and when the user lows his arm, the handbag can slide back to his hand.

There are 3 stable positions of the handbag: hand, elbow and shoulder. Figure 1 shows an example of the stick figure of an arm. The angles from the upper arm AB and the forearm BC to the gravity direction are α and β, and A, B and C represent the shoulder, elbow and hand respectively. The position of the handbag is determined based on the following rules:

1. If the handbag is at C, then it will start sliding towards B with a fixed velocity if $\beta > 150$ degree; otherwise it will stay at C.

2. If the handbag is at B, then it will start sliding towards C is $\beta < 30$ and sliding towards A if $\alpha > 150$; otherwise it stays at B.

3. If the handbag is at A, then it will start sliding towards B if $\alpha < 30$; otherwise it stays at A.

4. If the handbag is between BC, then it will slide towards B if $\beta > 90$; otherwise slide towards C.

5. If the handbag is between AB, then it will slide towards A if $\alpha > 90$; otherwise slide towards B.

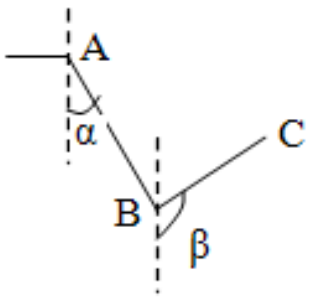
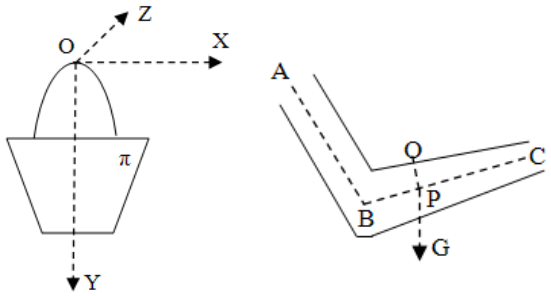


Figure 1. Stick figure of an arm.

3. Handbag pose computation

Based on the rules in Section 2, we can compute the 3D position of the handbag on the stick figure of the arm from the 3D positions of the hand, elbow and shoulder given by OpenNI. We call it the stick position of the handbag. To render the virtual handbag, its 6-DOF pose should be calculated in the world frame. To simplify the problem, each handbag is treated as a rigid object. As shown in Figure 2a, we can define the origin of the body frame of a handbag to be the tip O of its handle, and the Y axis as its vertical symmetry axis and the X axis as the direction vertical to Y on its symmetry plane π.



(a) (b)

Figure 2. (a)The body frame of a virtual handbag. (b) O is the handbag position on the arm surface if P is its stick position.

We assume the tip O is always on the surface of the user's arm. Given the stick position P of the handbag, the corresponding position of O is calculated in the way illustrated in Figure 2b. PO is on the surface formed by the gravity vector G and BC, and is perpendicular to BC. The length $|PO|$ is determined from the shape of the arm. For simplification, we assume the upper arm is a cylinder while the forearm is a cone with the elbow as the base. The radius of the upper arm and the forearm are estimated from the calibration pose of the user during the system initialization.

Therefore the position O is computed with:

$$O = P + |PO|(G \times \overrightarrow{BC} \times \overrightarrow{BC}), \quad (1)$$

where G is the unit gravity vector and $\overrightarrow{BC}$ is the unit vector of direction BC. At any moment, gravity forces the Y axis of the handbag frame to be parallel to G, and the XY plane is perpendicular to the surface formed by G and the forearm stick (or upper arm depending on the current stick position). The rotation R from the handbag frame to the world frame is hence fully determined:

$$R = \left[\overrightarrow{BC} \times G, G, \overrightarrow{BC} \times G \times G\right]. \quad (2)$$

However, with the above pose, the virtual handbag may have intersection with the human body. Therefore, we need a collision detection algorithm and an approach to adjust the handbag away from collision. In our system, we simply apply two additional rotations to the handbag frame:R_x around its X axis and R_y around its Y axis, and search the angles of the two rotations with adaptive step size to find a pose with which the handbag has not intersection with the human body. To check if there is an intersection, we sample the 3D handbag model with a set of 3D points and project them onto the image plane of the Kinect camera. If the depth values of the 3D points are all smaller than the corresponding values in the depth map generated by Kinect, the handbag has not intersection with the human body. The final rotation from the handbag frame to the world will be:

$$R = \left[\overrightarrow{BC} \times G, G, \overrightarrow{BC} \times G \times G\right] R_y R_x. \quad (3)$$

In practice, the occlusion relationship between the handbag, the trunk of the human body and the arms can be very complicated so that sometimes the above approach cannot find reasonable R_x and R_y to avoid the intersection judged by the simple Z buffer checking. This happens often when the handbag is at the shoulder position where the handbag is occluded by arms and the trunk of the body is occluded by the handbag. In our system, if reasonable R_x and R_y cannot be found, they will be set to the identity matrix.

4. Image segmentation for teleportation

In our system, the users can change the background of the augmented video from its real environment to a virtual background such as a beautiful hall or a red carpet runway, etc. We call this function teleportation that lets the users see how the handbags fit them in different backgrounds and circumstances.

To implement it, we need to segment the user's body from the real image and paste it into a virtual background to obtain a synthesized image. OpenNI gives a coarse segmentation of the user's body based on depth map but the boundary is not accurate. An example is shown in Figure 3 where the green contour is the original output from Kinect. In order to get better visual effects, we need an image segmentation approach to refine the coarse boundary. We propose a novel approach based on background modeling and graph cut that will be described in the following sections.

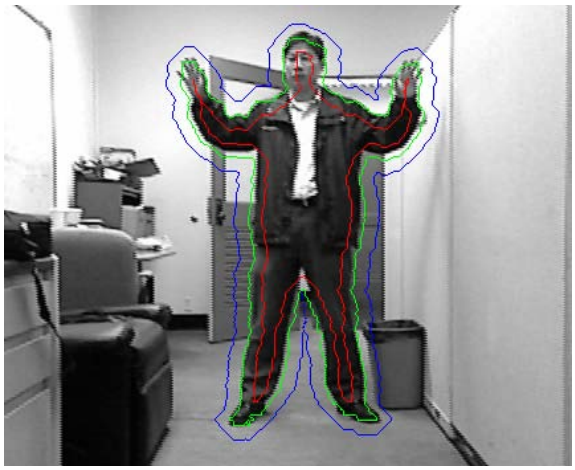


Figure 3. The green boundary is the initial output of Kinect. The pixels on the red and blue contours belong to the foreground and the background respectively.

4.1. Edge-based background modeling

In the application, the Kinect sensor is usually fixed so background modeling can help the segmentation of moving objects. Background modeling has been extensively studied in computer vision. The most popular approach estimates a GMM color model for each pixel [15]. However, due to the illumination change induced by moving objects, misclassification of pixels on the boundary of moving objects is quite often. A lot of approaches based on edges have also been proposed since edges are usually less sensitive to illumination change than color values or gradient magnitude. Some approaches compare individual edge pixels [12, 9], which is prone to noise and small variation of edge pixel locations. Some approaches [8, 13] detect moving objects by matching edge segments in the current image to those in the background image.

Our approach also exploits edge segments, but there are several major differences to the approaches in [8, 13]. First, the segment matching in [8, 13] does not fully enforce connectivity so that a connected segment in the current image can be matched to a set of disconnected edge pixels in the background image, which increases the chance of false matching. Second, in order to reduce false alarm, the edge pixels can be labeled as rather than only foreground or background. Third, the foreground and background edge segments will be given confidence values, and the final binary labeling will be calculated with graph cut by integrat-

ing them with other cues.

With moving objects out of the view, several images of the background are captured with the Kinect camera and their average image is computed. The edge map of the average image is obtained by using non-maximal suppression on its gradient map calculated with a 3x3 Sobel operator. Instead of using the hysteresis thresholding of Canny detector as in [8, 13], we only set a very low threshold (30 in our experiments) on gradient magnitude to remove the edges that are most likely caused by noise. Given a new image with moving objects, its edge map is computed in the same way.

Ideally, an edge pixel p in the current image will be classified as a background pixel if there is an edge pixel p' in the background edge map at the corresponding position whose gradient orientation is similar to that of p; otherwise p is a foreground pixel. However, in practice, the location of background edge pixels may have small deviation due to illumination change and noise. Therefore, if there is no corresponding edge pixel in the background edge map at the exact location of p but there is one in its 8-connected neighborhood, we are not sure if this edge pixel is new or a background edge pixel after a small drift. In this case, instead of making a possibly wrong classification, we prefer to label it as unknown and leave it for the graph cut to make the final decision. To put this formally, the label α of an edge pixel p in the current image is determined with:

$$\alpha = \begin{cases} background & \text{if } \exists p' \mid |p'-p| = 0 \text{ and } (p,p') < \theta; \\ unknown & \text{if } \exists p' \mid p' \in N(p) \text{ and } (p,p') < \theta; \\ foreground & \text{otherwise,} \end{cases} \tag{4}$$

where p' is an edge pixel in the background edge map, $N(p)$ is the 8-connected neighborhood of p, and (p, p') is the angle between the gradient orientation of p and p'. The threshold $\theta = 45$ degree in our experiments.

In addition, the confidence value of being foreground or background should not be determined based on individual pixels. The labeling will be much more confident if a set of edge pixels with the same label can be linked into a long curve segment compared to the case in which they are disconnected. Therefore, the edge pixels with the same foreground/background labels are linked into segments with an edge tracing algorithm. To reduce noise, we remove the segments shorter than 3 pixels. Figure 4 displays the foreground segments (red) and the background segments (blue) of Figure 3. The green pixels are labeled unknown, and we can see some of them are on the background objects such as those on the trash can whereas some are on the foreground object such as those on the left arm of the user.

The confidence value C_f of a pixel p being a foreground pixel is calculated with:

$$C_f(p) = \begin{cases} \sqrt{|g|L} & \text{if } p \in \text{foreground segment;} \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where L is the length of the segment and $|g|$ is the gradient magnitude of this edge pixel.

Similarly, the confidence value C_g of a pixel p being a background pixel is computed with:

$$C_g(p) = \begin{cases} \sqrt{min(|g|,|g'|)L} & \text{if } p \in \text{background segment;} \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

where L is the length of the segment, and $|g|$ and $|g'|$ are the gradient magnitude of the pixel and its corresponding pixel in the background edge map.

Therefore, based on the above two equations we obtain two confidence maps. In practice, in order to avoid the situation in which the following graph cut optimization can choose a boundary right next to some background segments due to their high color contrast, we dilate the background confidence map with the following approach. For each pixel p with $C_g(p) = 0$,

$$C_g(p) = \max_{q \in N(p)} C_g(q), \tag{7}$$

where $N(p)$ is the 8-connected neighborhood of p.

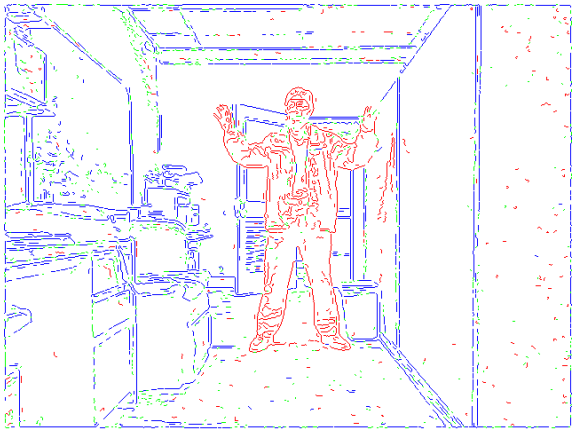


Figure 4. The blue and red segments are classified as background and foreground segments respectively. The green pixels are unknown pixels.

4.2. Graph cut segmentation

We use graph cut [14] to refine the coarse object boundary output by Kinect. As shown in Figure 3, the green contour is the initial boundary, and the pixels on the blue and red contours are assumed to be from background and foreground respectively. The pixels in the band between the blue contour and the red contour are the pixels that need to be classified as background or foreground. In our experiments the width of the band is 10 pixels on each side of the initial boundary. The optimization based on graph cut is to assign foreground/background labels to the pixels in

the band area so that the following energy function is minimized:

$$E(\alpha) = U(\alpha) + V(\alpha), \tag{8}$$

Where α represents the label assignment ($\alpha = 0$ for background and $\alpha = 1$ for foreground), $U(\alpha)$ is a data term that measures the likelihood of the pixels coming from background or foreground, and $V(\alpha)$ is a smoothness term that enforces smoothness of the labeling between neighboring pixels .

In our approach $U(\alpha)$ is computed with the following equation:

$$U(\alpha) = w_1 \sum_n B(\alpha_n) + w_2 \sum_n F(\alpha_n) + w_3 \sum_n D(\alpha_n) \tag{9}$$

The first item $B(\alpha_n)$ is the foreground/background confidence value based on the background modeling in the previous section, and for each pixel p_n:

$$B(\alpha_n) = \begin{cases} C_f(n) & \text{if } \alpha_n = 0; \\ C_g(n) & \text{if } \alpha_n = 1 \end{cases} \tag{10}$$

where C_f and C_g are computed with Equation 5 and Equation 6 respectively. Note that the optimization is to minimize the energy function.

The item $F(\alpha_n)$ is used to keep the final boundary close to the initial boundary given by Kinect. This is because although the initial boundary is not accurate, it is not too far from the truth. This constraint brings the information of the Kinect depth map to help the situation when the color contrast between the foreground and the background is very weak. The $F(\alpha_n)$ for a pixel p_n is computed with:

$$F(\alpha_n) = \begin{cases} d_n & \text{if } \alpha = 0 \text{ and } p_n \in I; \\ d_n & \text{if } \alpha = 1 \text{ and } p_n \in O; \\ 0 & \text{otherwise,} \end{cases} \tag{11}$$

where d_n is the distance of p_n to the initial boundary (the green contour in Figure 3), I is the region between the initial boundary and the inner boundary (red), and O is the region between the initial boundary and the outer boundary (blue) in Figure 3. It is easy to see that if the final boundary equals the initial boundary $F(\alpha)$ will be 0.

The last item $D(\alpha_n)$ is calculated in the same way as the data term in [14], which measures the color similarity to the foreground and background color models. In our case, the foreground GMM model is estimated from the pixels of the initial segmentation given by Kinect except those between the green boundary and red boundary in Figure 3. The background color model is estimated for each pixel during the background modeling phase.

The weights w_1, w_2 and w_3 (w_1 = 0.11, w_2 = 0.53 and w_3 = 0.36) are learnt with a logistic regression classifier from a set of training samples. The training samples are obtained by using interactive image segment [14] to obtain the ground-truth boundaries of moving objects. Finally, the smoothness term $V(\alpha)$ in Equation 8 is also the same as the smoothness term in [14] which reflects the color contrast between neighboring pixels. Please refer to [14] for details.

5. Experiments

In this section we present some experimental results of our system. Figure 5 shows several screen shots of an augmented video in which a virtual handbag is inserted at different positions including hand, elbow, shoulder and forearm. Figure 6 demonstrates the teleportation function in which the user is inserted into a virtual background in real-time. Figure 6a is the result with the original segmentation generated by Kinect. Figure 6b shows significant improvement after the boundary refinement with our approach, especially in the hair and shoe areas.

The rendering all looks realistic and gives the user the feeling of carrying a real handbag. However, we can also notice that the position of the handbag handle on the arm surface is not perfect. This is because the stick figure output by Kinect is not perfect, and the estimation of the arm width (used in Equation 1) from the calibration pose during initialization is also not very accurate.

In Figure 7, we compare the image segmentation result of different approaches. For Figure 7a-b, the initial boundary given by Kinect is shown as the green contour in Figure 3. Figure 7a gives the result if the data term $U(\alpha)$ of the graph cut in Equation 8 only includes $D(\alpha)$ that measures the color similarity. Figure 7b is the result with our approach that is significantly better than Figure 7a.

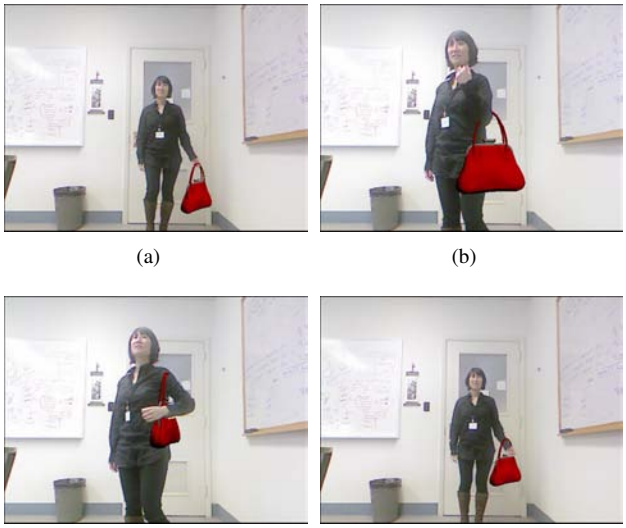
Figure 7c-d demonstrate the effect of adding $F(\alpha)$ in the data term. Figure 7c is the output without $F(\alpha)$, and we can see that the boundary along the left shoulder and arm is off a lot due to the low color contrast between the background and the foreground. By adding $F(\alpha)$ that incorporates the information of the depth map given by Kinect, the final boundary in Figure 7d is much more accurate.

6. Conclusion

In this paper, we described the implementation of an augmented reality system that allows the users virtually try on different handbags at home in front of a TV screen. The users can naturally slide virtual handbags to different positions on their arms and see the effects of carrying the handbags in different poses. We also presented a novel image segmentation algorithm based on the initial moving object boundaries giving by Kinect for the teleportation function in which the users can try handbags in different virtual background. The algorithm is designed based on the properties of the Kinect sensor.

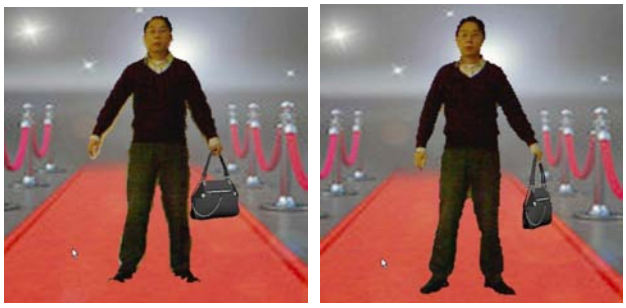
In future work, we plan to integrate the depth map gen-

erated by Kinect with its color image and a general human body model to increase the accuracy of the estimation of the handbag position on the arm surface. In addition, we also want to use the temporal constraints between video frames to reduce jittering of the handbag rendering and further improve the image segmentation.



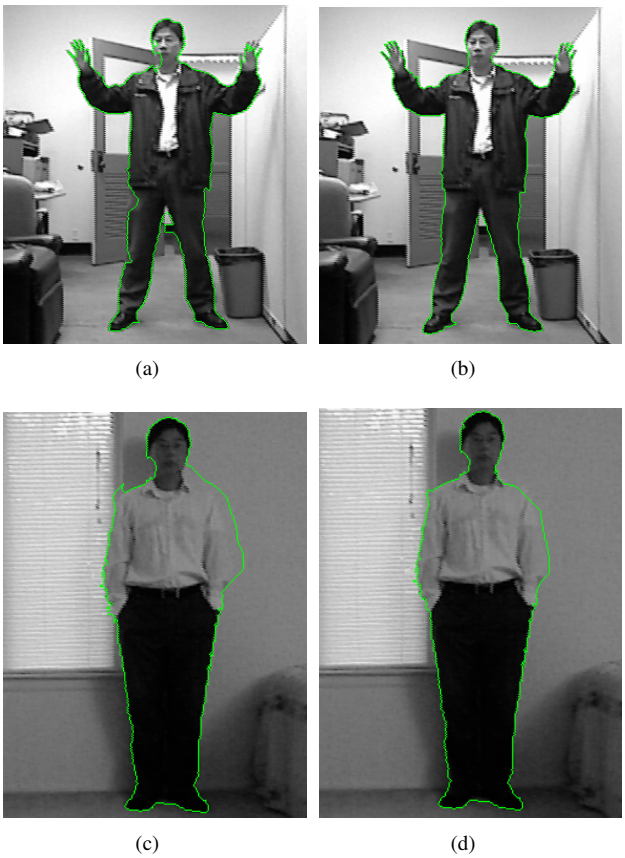
(a) (b) (c) (d)

Figure 5. Screen shots of an augmented video with a virtual handbag inserted at different positions.



(a) (b)

Figure 6. Teleportation with the original body boundary (a) and with the refined boundary with our approach(b).



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

Figure 7. (a)The result with only $D(\alpha)$ as the data term in the graph cut segmentation; (b)The result of our approach on the same input image of (a); (c) The result without $F(\alpha)$ in the data term of the graph cut segmentation; (d) The result of our approach on the same input image of (c).

References

[1] http://www.bodymetrics.com/. 1

[2] http://www.fitnect.com/. 1

[3] http://www.facecake.com/swivel/. 1

[4] http://openni.org. 1, 2

[5] Xbox kinect. http://www.xbox.com/kinect. 1

[6] F. Cordier, H. Seo, and N. Thalmann. Made-to-measure technologies for online clothing storer, 2003. 1

[7] P. Eisert, P. Fechteler, and J. Rurainsky. 3d tracking of shoes for virtual mirror applications, 2008. 1

[8] M. J. Hossain, M. A. A. Dewan, and O. Chae. Moving object detection for real time video surveillance: an edge based approach, 2007. 3, 4

[9] C. Kim and J. Hwang. Fast and automatic viddeo object segmentation and tracking for content-based applications, 2002. 3

[10] R. Li, K. Zhou, X. Xu, Y. Li, and Z. Li. Research of interactive 3d virtual fitting room on web environment, 20011. 1

[11] X. Liu, C. Jiang, K. SZE, and C. Wang. Online cloth virtual fitting room based on a local cluster, 2009. 1

[12] A. Makarov, J. Vesin, and M. Kunt. Intrusion detection using extraction of moving edges, 1994. 3

[13] M. Murshed, A. Ramirez, and O. Chae. Statistical background modeling: An edge segment based moving object detection approach, 2010. 3, 4

[14] C. Rother, V. Kolmogorov, and A. Blake. Grabcut-interactive foreground extraction using iterated graph cuts, 2004. 4, 5

[15] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking, 1999. 3