International Journal of Computing Academic Research (IJCAR)
ISSN 2305-9184 Volume 2, Number 1 (February 2013), pp. 1-13

http://www.meacse.org/ijcar

Augmented Dressed Body System Controlled By Motion Capture Data

Khaled F. Hussain, Adel A. Sewisy and Islam T. El-Gendy

Computer Science Department, Assuit University, Egypt

Abstract

Augmenting deformable surfaces like cloth and body in real video is a challenging task. This paper presents a system for cloth and body augmentation in a single-view video. The system allows users to change their cloth either by changing the color, the texture, or the whole cloth. It augments the user with virtual clothes. As a result, users can enjoy changing their cloth with any other cloth they want. As a prerequisite, the user needs to wear a special suit and enters through our motion capture system that captures the movements of the user. From the captured data, an animated 3D character model is created, which will serve as the new body. The model is rendered with the new cloth but without the head. We extract the real face of the user and place it on the virtual model. This system can be used in film production and advertisement.

Keywords: Camera registration, Cloth simulation, Color transfer, Matting, Motion capture system, Segmentation, Video editing.

Introduction

Combining computer generated content or virtual objects with real video is used with great interest in many applications such as movies and augmented reality [1]. AR *(Augmented reality)* is the integration of computer generated objects with live video in real time in a way that the viewer cannot tell the difference between the real and augmented world. Film production takes place all over the world in a huge range of economic, social, and political contexts, and using a variety of technologies and cinematic techniques. A very important issue in any film production or advertising is the costumes used by actors or actresses, and the cloth design. Making various and different costumes is essential for making a good film. However, the budget needed for this task usually is very high, and sometimes it is even impossible to change clothes while shooting the scene without compromising the flow of the scene.

We developed an off-line system which can solve this problem, where users need only to wear a single suit in the entire production. The user performs any kind of movement (e.g. dance, fashion walk, … etc.) within the capture volume of our motion capture system, which consists of 24-cameras and records the user's movements. An animated 3D character model is created similar to the user with the new cloth on it but without a head. An outside camera records the user's movements, in which the user's head is extracted and placed on the animated 3D character model with the new clothes, and the augmented body. The combined real head and virtual body can be placed in any background. Our system can work on smaller 8-cameras that is more appropriate in stores because it is cheaper.

The contribution of this work is a complete system to combine real video with a virtual character animated with motion capture data. The possibility to put clothes varying in style and color, and

even change the look of the users to make them for example slim or athletic, make the system useful for applications like virtual try-on and digital film production.

Previous work

Cloth tracking is a difficult and a challenging problem because of its flexibility and ability to self-occlude. In order to perform augmentations on a non-rigid object such as a flexible piece of cloth, a mesh representation of the cloth is obtained [2]. To achieve correct folds and wrinkles in virtual cloth, complex physical simulations are used.

A system that gives users the ability to interactively control a 3D model of themselves at home using a commodity depth camera is made by [3], it augments the model with virtual clothes that can be downloaded. The user in this system needs to pass through or enter a multi-camera setup, which captures him or her in a fraction of a second. A 3D model is created from the captured data. Then, the model is transmitted to the user's home system to serve as a realistic avatar for the virtual try-on application. The drawback is that it can only be used in virtual try-on, and it doesn't look realistic because of the virtual face.

In [4], real-time 2D augmentations on non-rigid objects, such as clothing is presented, in which a technique is included to establish common illumination and render augmentations with correct real world shadows.

Another way to augment cloth involves sparse cloth-tracking in video images using a vision-based marker system with temporal coherence [5], with an image-based method to automatically acquire real world illumination and shadows from the input frame.

A method of 3D structure reconstruction and blending using a single-camera video stream is used to insert or modify models in a real video stream [6], the approach is based on a simplification of camera parameters and the use of projective geometry without camera calibration.

A method for augmenting deformable surfaces, like cloth, in single-view video with realistic geometric deformation and photometric properties without a 3D reconstruction of the surface for augmented reality applications is presented in [7], where retexturing the retrieval of deformation and motion in the image plane is sufficient as the augmented surface is rendered from the same point of view as the original one.

An extended optical flow version is used in [1,8] together with mesh-based models and a specific color model to estimate deformation and photometric parameters simultaneously, that not only accounts for changes in the light intensity, but also in the light color. These methods work only if the cloth has a suitable texture, and place only texture on the cloth not augmenting the whole cloth. External occlusions can be accounted for in an occlusion map, showing and classifying whether a pixel is visible or not, based on local texture patch color distributions and a global occlusion color distribution.

System overview

Our approach consists of two major parts (see Figure 1). First, the virtual part in which the 3D character model is generated with the free MakeHuman (http://www.makehuman.org) software. The animation of the model is taken from the user's movements in a 24-camera system, which records the movements of the reflective markers on the suit. The animated 3D character model can be rendered from any viewpoint. The virtual model is rendered with the same viewpoint of the outside camera with a new virtual cloth and without the head. Section 4 describes the construction and registration part in detail.

The second part is the outside camera and its output, where we segment the user's body, extract the real head of the user. Finally, place that head on the virtual body. Section 5 describes the segmentation part in detail, while section 6 describes the final part of head placement.

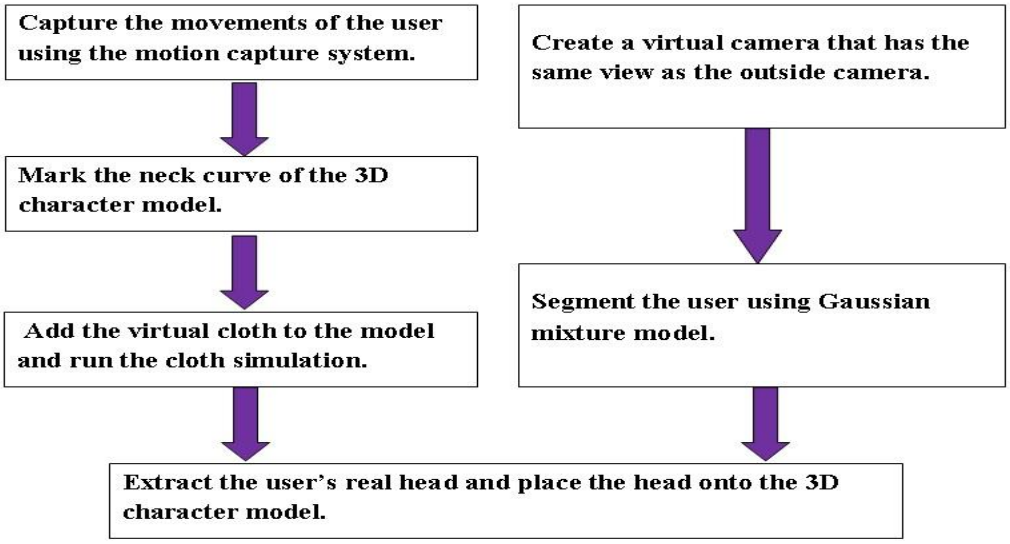


Figure 1: Overview of our pipeline.

Avatar construction and registration

This section describes the construction of the 3D character model, and the steps of scene registration.

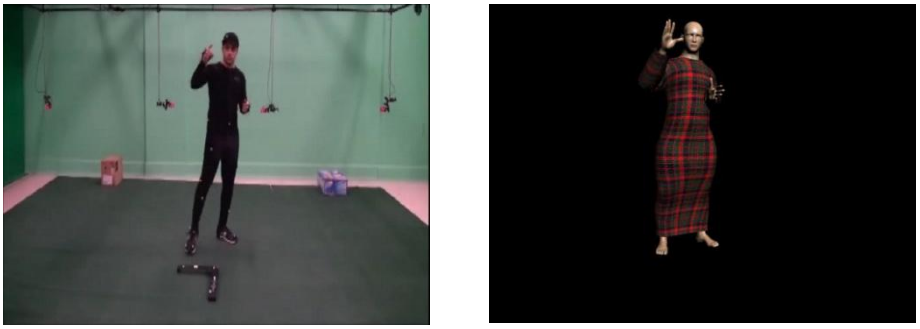
The animated 3D character model. For rendering a video of the user and augmenting him or her with different clothes or deforming his or her body, a virtual avatar is needed. Furthermore, the user's movements need to be captured, so that the avatar has the same movements of the user. We use the Naturalpoint (http://www.naturalpoint.com) motion capture system to capture the user's movements, which consists of a $6 \times 6 \times 3$ meter^3 capture volume where 12 cameras are mounted top, and 12 cameras are mounted below them as shown in figure 2a. All cameras are pointed towards the centre of the capture volume, where the user is allowed to move freely. Every six cameras are connected together in a USB Hub, every USB Hub is connected with the next one with a synchronization cable, and the four hubs are connected to a single PC via USBs. The cameras capture images with a resolution of 640×480 pixels. All cameras are calibrated using ARENA (http://www.naturalpoint.com/optitrack/products/arena) software.

The freely available MakeHuman application allows users to create and modify human avatars by specifying a wide range of parameters, like facial features, gender, body part proportions etc. The 3ds Max (http://usa.autodesk.com/3ds-max) software can import the MakeHuman DAE file format. In order to create the animated 3D character model, the user's movements are captured, and the 2D points are trajectoried into 3D points. The system exports the animation into BVH file format. The 3ds Max software can easily read the BVH file into a rigged body.

Figure 2 shows a frame of an outside camera and a virtual scene where the 3D character model is created and has the same movements as the user. The outside camera is a fixed single camera placed anywhere in the scene. If the user wants to change the camera's viewpoint, he/she must do the calibration process again. The outside camera is used at the same time and in the same place as the tracking system.

We used the 3ds Max cloth modifier to perform the cloth simulation. We design the new virtual cloth which will replace the real one as shown in figure 2b. The cloth itself can be designed as pieces of garments that is sewed together to fit the avatar body, and then simulated for each frame.

Registration. The purpose is to make the world's coordinate system of the outside camera the same as the world's coordinate system of the 24-camera system. The virtual scene is modeled with



(a) (b)

Figure 2: (a) the user wears the suit in side of the motion capture system. (b) the virtual avatar is created with MakeHuman and dressed with purely virtual clothing.

some objects similar to the real scene. We already made the 3D character model with the same movements, but we must create a virtual camera that has the same point of view as the outside camera which is the registration process. The purpose of the *registration* in our system is to align the virtual body with the real body. Without accurate registration, the output will be visually inconsistent.

In order to make a correct registration with the virtual scene, we prepare the lab with calibration square on the floor and two boxes in the back of the lab, so we can reference the 3 markers on the calibration square and the corners of the boxes. The virtual scene contains the positions of the 3 markers (3 small spheres) and the boxes with virtual ones of the same dimensions and the exact distance from the model as in the real scene. We have used a registration method [9] that aligns objects by providing a number of points (r_i, c_i) in image coordinate system whose location is (x_i, y_i, z_i) in the world coordinate system. The camera's intrinsic parameters (r_0, c_0) (principal point coordinates), (f_u, f_v) *(focal length)*, and image distortion coefficients (radial and tangential distortions) have been determined and are fixed; thus simplifying the system of equations needed to be solved. Given the points in the image coordinate system and the corresponding points in the world coordinate system, the transformation (R, T) between the 24-camera world's coordinate system and the outside camera's coordinate system is calculated

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad , \quad T = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \tag{1}$$

where R is the rotation matrix, and T is the translation vector. A constrained optimization technique is used that maintains the orthonormality of rotation matrix R.

Segmentation

The virtual body is now similar to the real body with the same viewpoint. Next, we segment the user from the background, extract the head, and mark the virtual neck curve.

User segmentation. We extract the body of the user from the video using Gaussian mixture models (GMMs) [10]. Simple difference matte key can be used, in which the matte is generated by taking the absolute value of the difference between two images *(video frames)*, one with the item of

interest present and an identical one without the item of interest. The problem is that in the real world, this rarely happens as the lighting conditions may change.

Hence, we used the GMMs in which the Gaussian distribution for *d* dimensions of a vector $a = (a^1, a^2, \dots, a^d)^T$ is defined by:

$$N(a \mid \mu, \sum) = \frac{1}{(2\Pi)^{\frac{d}{2}}\sqrt{|\sum|}} \exp\left(-\frac{1}{2}(a - \mu)^T \sum^{-1} (a - \mu)\right) \quad (2)$$

where μ is the mean, and $\sum$ is the covariance matrix of the Gaussian.

The probability given in a mixture of K Gaussians is:

(a) (b) (c)

Figure 3: (a) Background frame. (b) Source frame. (c) Foreground frame.

$$p(a) = \sum_{j=1}^{K} w_j . N(a \mid \mu, \sum) \quad (3)$$

where w_j is the prior probability (weight) of the *j*th Gaussian. The first few frames represent the background and they are used to initialize the GMMs (figure 3a shows a background frame). The GMMs are updated for each pixel, so even when the lighting conditions change the system can still detect the foreground. Figure 3 shows the background subtraction using the GMMs.

Head extraction. We extract the head using graph cut segmentation and apply Poisson matting to compute the alpha channel.

Skin detection: In order to detect the face region, we use skin detection. First, we convert the *RGB* color values of the image (skin samples) to *YCbCr* color space. The *YCbCr* is an *M-by-3* matrix that contains the *YCbCr* luminance *(Y)* and chrominance (C_b and C_r) color values as columns.

Calculate the mean μ :

$$\mu_b = \frac{\sum_{i=0}^{n} p_{bi(x,y)}}{N} \quad , \quad \mu_r = \frac{\sum_{i=0}^{n} p_{ri(x,y)}}{N} \quad (4)$$

where p_{bi} is the *ith* C_b value, and p_{ri} is the *ith* C_r value.

Calculate the Mahalanobis distance to get the skin regions in the frame:

$$D_M(l) = \sqrt{(l - \mu)^T S^{-1} (l - \mu)} \quad (5)$$

where vector $l = (l_1, l_2, l_3, \dots, l_N)^T$, μ is the mean, and *S* is *2×2* covariance matrix.

Apply a threshold on $D_M(x)$ where 1 is skin, and 0 is background. Then, apply connected component to get the biggest area *(the face)*. Then, we fill in the holes in the skin region *(such as the eyes)* to get the whole face area.

Graph cut segmentation: Our goal is to accurately segment a given human image into "head" and "body" regions. We used the Graph Cut segmentation technique [11,12] because it gives the best balance of boundary and region properties. Supply a *trimap* which partitions the image into three parts: "definitely foreground", "definitely background", and "unknown region". Skin detection provides the foreground region. Also, the hair must be taken into account. From the region properties of the video frames we: 1) Compute the boundary of the face region. 2) Calculate the centroid of the face region. 3) Calculate the orientation, which is the angle in degrees between the x-axis and the major axis of the ellipse that has the same second moments as the face. 4) Extend a line from the centroid of the face region toward the hair region in the direction of the major axis. 5) Compute the intersection of the line with the face boundary. 6) Merge a small sample region around the intersection, representing the hair, with the face region. 7) Consider the merged region (head region) as a definite foreground.

The solution to our segmentation problem is a binary vector $A = (A_1, \ldots, A_r, \ldots, A_R)$, where A_r equals one for head and equals zero for body. To improve efficiency, we deal with regions instead of image pixels. We used watershed algorithm [12, 13] which divides the image into smallregions. We create a graph $G = (V, E)$ with nodes corresponding to regions $r \in R$ of the image. There are two additional nodes: a ”head” terminal (a source S) and a ”body” terminal (a sink T). So, becomes

(a) (b) (b)

Figure 4: (a) Graph cut input background. (b) Graph cut input foreground. (c) Graph cut output foreground.

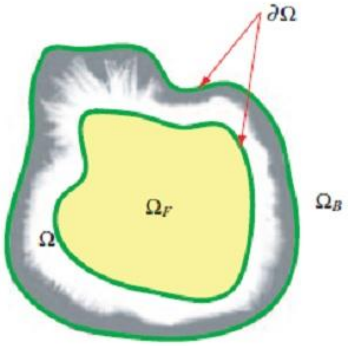


Figure 5: The trimap {ΩF, ΩB, and Ω} is supplied from the algorithm. $\partial\Omega$ is the exterior boundary of unknown region [15].

$$V = R \cup \{S, T\} \tag{6}$$

Each region *r* has two t-links (terminal links) *r, S* and *r, T* connecting it to each terminal. n-links (neighborhood links) connect each pair of neighboring regions *p, q* in N.

Therefore,

$$E = N \cup_{r \in R} \{\{r, S\}, \{r, T\}\} \quad (7)$$

We divide the regions into three types: head, body, and unknown regions. The K-means method is used to divide the head and body regions into clusters. The mean intensity of the head and body clusters are denoted as $\{K_n^C\}$ and $\{K_m^B\}$ respectively. For each region *r*, we compute the minimum distance from its mean intensity *M(r)* to head clusters as $d_r^C = min_n \|M(r) - K_N^B\|$. Table 1 gives weights of edges in E. After defining the graph G, we need to calculate the minimum cost of a cut on the graph G. The cost of the cut is defined as $CH = \sum_{e \in H} W_e$. The minimum cost of the cut CH can be calculated in polynomial time using the maxflow algorithm in [14].

After calculating the cut *H*, the segmentation binary vector $A = (A_1, \ldots, A_r, \ldots, A_R)$, where A_r is calculated as follows:

$$A_r(H) = \begin{cases} 1(Head) & \text{if } \{r, T\} \in H \\ 0(Body) & \text{if } \{r, S\} \in H \end{cases} \quad (8)$$

Figure 4 demonstrates the stages of graph cut segmentation. From left to right, the background frame, after background subtraction using GMMs as described before, the supplied trimap background, the final output after graph cut segmentation.

Poisson matting: In order to place the real head seamlessly on the virtual body, we calculate the alpha channel using the Poisson matting. The gradient of matte is estimated from the image, then we reconstruct the matte by solving the Poisson equations [15]. In our system the intensity change in the foreground and background is smooth, thus we use the global Poisson matting. The *trimap* is computed from the output of the graph cut segmentation. In the unknown region, the matte α can be estimated from the color statistics in the known foreground *F* and background *B* regions.

Take the partial derivatives on both sides of the matting equation:

$$\nabla I = (F - B)\nabla\alpha + \alpha\nabla F + (1 - \alpha)\nabla B \quad (9)$$

where $\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)$ is the gradient operator. $\alpha\nabla F + (1 - \alpha)\nabla B$ is relatively small with respect to $(F - B)\nabla\alpha$, we can get an approximate matte gradient field as follows:

$$\nabla\alpha \approx \frac{1}{(F-B)} \nabla I \quad (10)$$

It means that the matte gradient is proportional to the image gradient.

As shown in figure 5, ΩF, ΩB, and Ω are the "definitely foreground", "definitely background", and "unknown" regions respectively. For each pixel $p = (x, y)$ in the Image I_p is its intensity, F_p and B_p are the foreground and background intensity respectively. Let N_p be the set of its 4 neighbors. $\partial\Omega = \{p \in \Omega_F \cup \Omega_B \mid N_p \cap \Omega \neq 0\}$ is the exterior boundary of Ω.

To recover the matte in the unknown region Ω given an approximate *(F B)* and image gradient ∇I, we minimize the following variational problem:

$$\alpha^* = \arg min_\alpha \iint_{p\in\Omega} \left\| \nabla \alpha_p - \frac{1}{F_p - B_p} \nabla I_p \right\|^2 dp \quad (11)$$

with Dirichlet boundary condition $\alpha|_{\partial\Omega} = \hat{\alpha}|_{\partial\Omega}$. We define:

$$\alpha|_{\partial\Omega} = \begin{cases} 1 & p \in \Omega_F \\ 0 & p \in \Omega_B \end{cases} \quad (12)$$

The associated Poisson equations with the same boundary condition is:

$$\Delta\alpha = div\left(\frac{\nabla I}{(F-B)}\right) \quad (13)$$

where $\Delta = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)$ and *div* are Laplacian and Divergence operators respectively. Then apply iterative optimization on the Poisson matting.

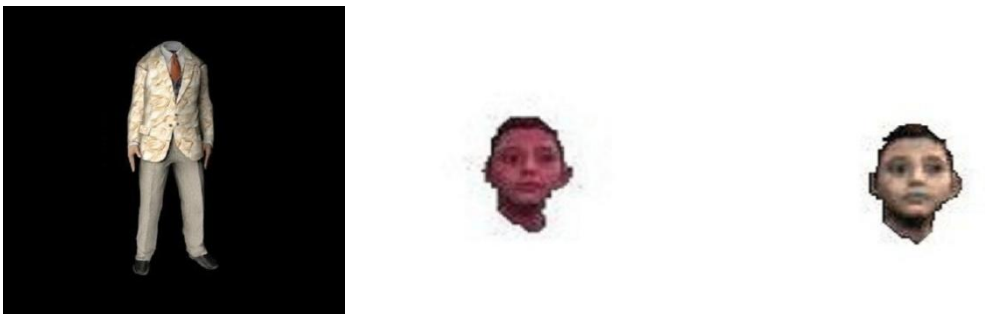
Neck curve marking. The 3D character model's head is replaced with the user's head for more realistic view. In order to replace the virtual head with the user's head, we attach points with the curve of the model's neck and save the position of these points for all frames in a separate file.

We remove the virtual head, and render the virtual model, as shown in figure 6a. However, the points on the neck curve will not be rendered, but for each frame the positions of these points are transformed from 3D into 2D points and saved in a data file. These points will be used for later placing the head of the user on the animated 3D character model's body consistently.

Head replacement

To place the real head accurately onto the virtual body, we use the non-reflective similarity transformation. To calculate this transformation, we need two pairs of points. Two points on the sides of the virtual neck, and two points from the motion capture system which are the two reflective markers on the sides of the real neck. These two points are 3D points in the world coordinate system. Using the intrinsic and extrinsic parameters calculated previously, we convert the 3D points in the world coordinate of the motion capture system into 2D points in the image coordinate of the outside camera. Then, apply non-reflective similarity transformation onto the real head, which includes a rotation, a scaling, and a translation.

The tracking system is not ideal. There can be a small error in tracking the two points on the sides of the real neck. This small error can produce visual inconsistency frame. Hence, we take the average of the distance between the two points on the sides of the real head and the corresponding nearest points on the head boundary. Then, we apply a translation of the head with that distance.



(a) (b) (c)

Figure 6: (a) Source image. (b) Target image. (c) Result image.

Global color transfer. Apply a global color transfer [16] on the head *(target)* and the virtual body *(source)* to have the same lighting conditions to get more realistic appearance. The main idea of the method is to move source image's data points by scaling, rotation and translating to fit data points' cluster of target image in RGB color space. First, calculate the mean of pixel data and the covariance matrix of the three components in color space RGB for both the source and target images.

Then, decompose the covariance matrices using SVD algorithm.

$$Cov = U \,.\, \Lambda \,.\, V^T \quad (14)$$

where U and V are orthogonal matrices and are composed of the eigenvectors of *Cov*, $\Lambda = \mathrm{diag}(\lambda^R, \lambda^G, \lambda^B)$, and λ^R, λ^G, and λ^B are the eigenvalues of *Cov*. Employ U as a rotation matrix to manipulate pixels of the source image.

Apply a transformation as follows:

$$I = T_{src} \,.\, R_{src} \,.\, S_{src} \,.\, S_{tgt} \,.\, R_{tgt} \,.\, R_{tgt} \,.\, I_{tgt} \quad (15)$$

where $I = (R, G, B, 1)^T$ and $I_{tgt} = (R_{tgt}, G_{tgt}, B_{tgt}, 1)^T$ denote the homogeneous coordinates of pixel points in RGB space for the result and target images respectively, and T_{src}, T_{tgt}, R_{src}, R_{tgt}, S_{src}, S_{tgt} denote the matrices of translation, rotation and scaling derived from the source and target images separately. Figure 6 shows the global color transfer output.

Neck inpainting or cutting. Now that we have placed the real head on the virtual body consistently. One of three problems can occur; either there is a gap between the real neck and the virtual body, the neck is larger than the virtual neck area, or a portion of the real neck is larger than the virtual neck and there is a gap in the rest of the virtual neck. In the first case, we use the inpainting technique [17] to fill in the gaps between the head and the virtual body (filling in the neck area). The target region to be filled *(the empty region between the head and the body)* is indicated by Ω, and its contour is denoted $\delta\Omega$. The contour evolves inward as the algorithm progresses. The source region Φ *(portion of the skin)*, which remains fixed throughout the algorithm, provides samples used in the filling process.

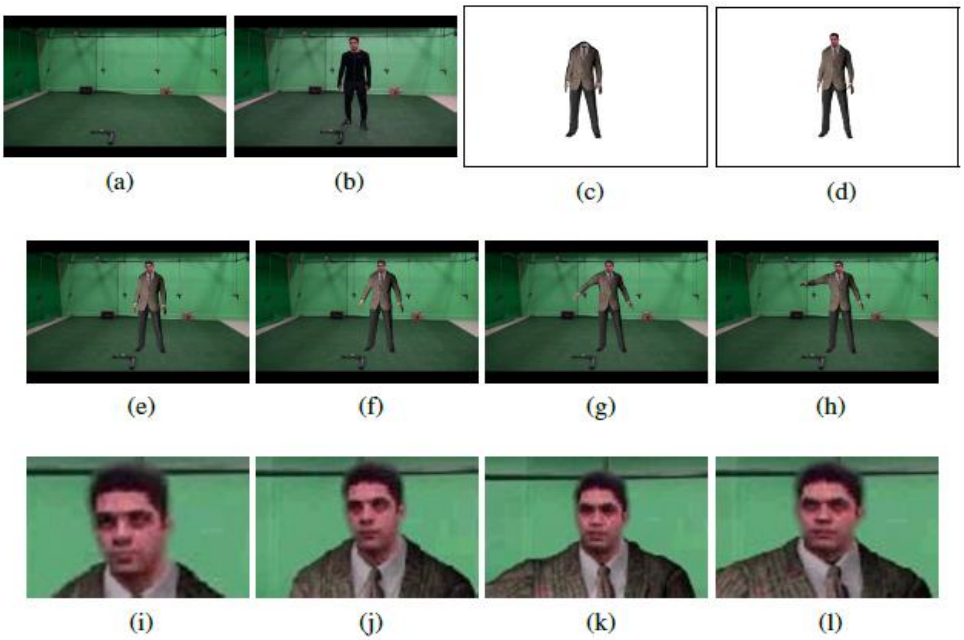


Figure 7: (a) Background frame. (b) Original frame. (c) Headless virtual body with cloth. (d) Real head combined with virtual body. (e,f,g,h) Four frames from the output video of our system. (i,j,k,l) Closer look at the real head.

In the second case, we remove all the pixels below the neck curve points to cut the extra neck region. First, we get the virtual neck boundary $Neck_{mask}$ and go through all pixels of the real neck. Set all pixels below the $Neck_{mask}$ to 0 *(removing them)*.

In the third case, we remove all the pixels below the neck curve points to cut the extra neck portion as explained before, and then we use the inpainting technique to fill in the gaps between the head and the virtual body.

Results

We tested our system on several real video sequences with a resolution of 720 × 480 pixels and a frame rate of 30 fps from the outside camera, and video sequences with a resolution of 720 × 480 pixels and a frame rate of 30 fps from the virtual scene. The actors/actresses would carry out movements that were tracked by the system and imitated by a 3D character model. The system has a very fast response time and that even fast motions are tracked consistently. However, if a user leaves the capture volume, then the Motion Capture software has difficulties in figuring out the correct skeletal state of the user. In such situations when the user re-enters the capture volume, the motion capture system may take some time to adapt to the correct state.

We produced augmented versions of several video sequences which are best evaluated by visual inspection. Figure 7 shows the main steps of our system. Figures 7a - 7d present a background frame from a video sequence, a frame with real actor, a dressed virtual model without head, and a dressed virtual model with real head. Figures 7e - 7h show different frames from the output video of our system that composites between the dressed virtual model with real head and the background frames.

Figure 8a shows an original frame from a video that is recorded by the outside camera while the rest of the frames show the result of our system using virtual bodies with different sizes. Figure 9

demonstrates how we can use different clothes. Furthermore, it demonstrates the results of our system using different background.

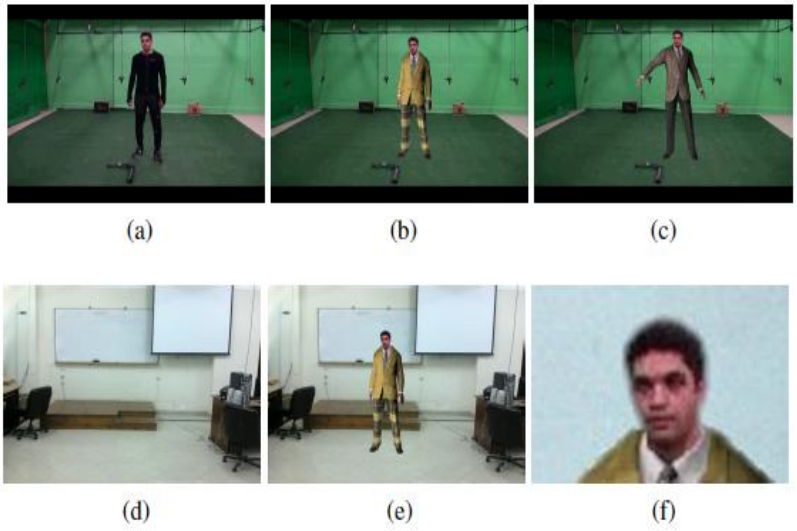


Figure 8: (a) Original frame. (b,c,d) Results of our system using virtual bodies with different sizes.

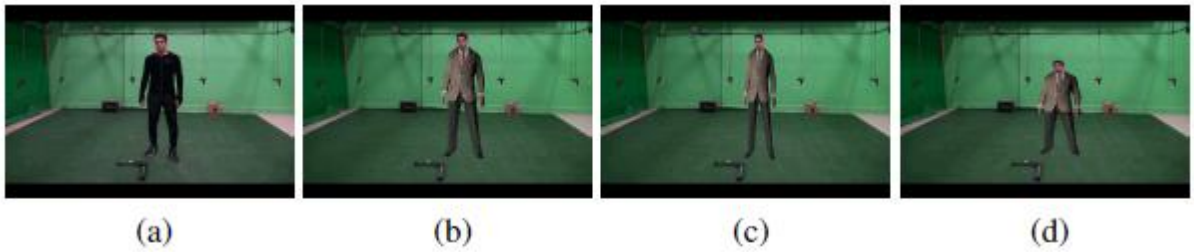


Figure 7: (a) Original frame. (b,c) Results of our system using different clothes. (d) Different background frame. (e) Results of our system using different background. (f) Zoom in.

Conclusions and future work

As shown in figure 7, this paper has presented a realistic and affordable system for cloth and body augmentation in single-view video. The system allows users to change their cloth either by changing the color, the texture, or the whole cloth. The user wears a special suit and enters through our motion capture system that captures the movements of the user. From the captured data, an animated 3D character model is created, which will serve as the new body with the new cloth but without the head. The real head of the user is extracted and placed consistently on the virtual model. The main contributions of this paper are:

- Description of methods and software to create a virtual character animated with motion capture data.

- The application of registration techniques to align the 3D character model to the users' body in the video.
- Segment the user from the background in the video file and extract the head with a skin detection method, graph cut segmentation and image matting.
- Replacing the virtual characters head with the one extracted from the video file including color transfer to match lighting conditions.

Currently, our system is an off-line system. In the future we plan to make a realtime system, that captures the user's movements and augments the cloth and body of the user in real time. We used the cloth simulation and rendering of the 3ds Max, we plan to implement our own cloth simulation and rendering. A real-time system would have many applications such as film production, advertisement, or fashion.

References

[1] Anna Hilsmann, David C. Schneider, and Peter Eisert. Technical section: Realistic cloth augmentation in single view video under occlusions. Computer. Graph., 34(5):567574, October 2010.

[2] R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. In Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, SCA 03, pages 2836, Aire-la-Ville, Switzerland, Switzerland, 2003.

[3] Stefan Hauswiesner, Matthias Straka, and Gerhard Reitmayr. Free viewpoint virtual try-on with commodity depth cameras, 2011.

[4] G. Roth D. Bradley and P. Bose. Augmented clothing. In Graphics Interface, 2005.

[5] D. Bradley, G. Roth, and P. Bose. Augmented reality on cloth with realistic illumination. Machine Vision and Applications, September 2007.

[6] Jong-Seung Park, Mee Sung, and Sung-Ryul Noh. Virtual object placement in video for augmented reality. In Yo-Sung Ho and Hyoung Kim, editors, Advances in Multimedia Information Processing - PCM 2005, volume 3767 of Lecture Notes in Computer Science, pages 1324. Springer Berlin / Heidelberg, 2005.

[7] A. Hilsmann and P. Eisert. Realistic cloth augmentation in single view video. In Proc. of Vision, Modeling, and VisualizationWorkshop, volume 2009, pages 5562, 2009.

[8] A. Hilsmann, and P. Eisert: Joint Estimation of Deformable Motion and Photometric Parameters in Single View Videos. Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on Oct. 2009.

[9] David E. Breen, Eric Rose, and Ross T. Whitaker. Interactive occlusion and collision of real and virtual objects in augmented reality , 1995.

[10] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In Proceedings of the Pattern Recognition, 17th International Conference on (ICPR04) Volume 2 - Volume 02, ICPR 04, pages 2831,Washington, DC, USA, 2004.

[11] Yuri Y. Boykov and M. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images, Proceedings of Eighth IEEE International Conference on Computer Vision (ICCV), Vol. 1, pp. 105-112, 2001.

[12] Yin Li, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum. Lazy snapping. In ACM SIGGRAPH 2004 Papers, SIGGRAPH 04, pages 303308, New York, NY, USA, 2004.

[13] Luc Vincent and Pierre Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. IEEE Trans. Pattern Anal. Mach. Intell., 13(6):583598, June 1991.

[14] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of mincut/max-flow algorithms for energy minimization in vision. IEEE Trans. Pattern Anal. Mach. Intell., 26(9):11241137, September 2004.

[15] Jian Sun, Jiaya Jia, Chi keung Tang, and Heung yeung Shum. Poisson matting. ACM Transactions on Graphics, 23:315321, 2004.

[16] X. Xiao, and L. Ma Color transfer in correlated color space. Proceedings of the 2006 ACM international conference, 2006.

[17] Criminisi Erez Toyama, A. Criminisi, P. Perez, and K. Toyama. Object removal by exemplar-based inpainting. In Computer Vision Pattern Recognition, pages 721728, June 2003.