Turning two dimensional photographs of clothing into three dimensional models

Capita Selecta assignment by:

Jeroen van Vroonhoven
Universiteit Twente
j.c.j.vanvroonhoven@student.utwente.nl

Introduction:

In most current web shops that sell clothing, the user can see photographs of the clothing on different models, or he/she can view pictures of different views of the garment itself. For a better view of the piece of clothing that the user wishes to buy, it would be a good idea to display a three dimensional view of the clothing that the user can rotate at all angles. This could even be expanded to let the user see a 3d model of a person wearing the selected clothing in order to see how well certain pieces match together.

This research assignment will focus on ways to realize this kind of clothing visualization for web shops. The challenge here is to differentiate between different garments and the fact that the photographs of the available clothing might differ. Since the photographs will originate from the web shops, they are not always taken with the same parameters.

The assignment was split into three different phases. The first step is to find an existing 3d model of a garment that would be easy to manipulate and finding a way to place a texture onto it. Phase two will focus on editing the model so that it contains a limited number of parameters that can be altered. These parameters can be used to change the general shape of the clothing without having to create an entirely new 3d model. The final phase was originally planned to be research into background subtraction for clothing, but since that does not really fit with the rest of this assignment, phase 3 will focus on researching the possibilities of automating the parameters from phase 2. It will also look at possible problems that could occur when creating 3d models of other garments. Before going into details of the separate phases, a short overview will be presented of related work that forms the basis of different choices made during the course of the assignment.

Related work:

There are rather few papers that specifically look at creating three dimensional models of clothing, but numerous authors have found different ways of forming a rough representation based on one or more photographs of random objects or landscapes.

The MIRALab at the University of Geneva performs a lot of research into Computer Graphics related topics, including the creating of 3d clothing. In one of their papers, they create realistic cloth models by overlaying a front and back view of the photograph over an existing virtual human with standard measurements. The front view is placed in front of the human, and the back view is placed behind it,
the edges of the two photographs are linked together and the garment is completed by using cloth simulation to link the edges together, see Figure 1. [1]

Hassner and Basri have attempted 3d reconstruction based on a single photograph, together with some metadata about the photograph, namely the class of object represented in it, such as a human body or a fish. Their algorithm requires an object database containing a number of 3d models in the same class as the photograph. For every area of the photograph the program would look through the depth representations of the models in the database and compare it with the same area. The accuracy of the depth information is averaged between all database objects and this creates a depth representation of our new object. [2]

**Phase 1:**
The goal of the first phase was to obtain a three dimensional model of any type of clothing together with matching photographs of the same type of clothing. Then the images of the garment would be put as a texture onto the three dimensional model to get a realistic looking model that can be rotated at will in a web interface.

After finding a decent-looking 3d model of a pair of pants, the program Blender was used to edit the model a bit. Initially, the model contained a lot of extra components that were not needed for the assignment, such as additional pockets and a belt. After removing these parts, the model consisted of
over 7000 vertices, which is too detailed for a model that should be shown in a web interface. In order to improve the rendering speed of the final model, the number of vertices will have to be reduced. Blender is equipped with a modifier called ‘Decimate’ that will reduce the number of vertices while still retaining the general shape of the original model [3]. This was used with a ratio of 0.5, which means that the model will contain 50% of the original number of vertices. Blender shows a preview of the model if the modifier should be applied and lowering the ratio more than 0.5 seemed to radically reduce the quality of the model.

The next step is to place a texture on the model. There are a number of ways to do this in Blender by using different so-called UV unwraps. The most common techniques for doing this are by letting Blender figure out an unwrapping by itself, this is usually quite suboptimal. It is also possible to mark a number of edges as a seam by which the model is ‘cut open’ and then unwrapped. This method is the most intuitive one for a large amount of applications, but in this case it is known that the textures will be photographs of different angles of the clothing. Most web shops provide photographs of the front and back of every piece of clothing that can be considered to be orthogonal since the camera will be relatively far from the garment in order to take a clear picture. This is why the method that was used in the end was ‘Project From View’, which basically means that the wrapping is based on the front and back view of the pants [4], see Figure 2 for a simple example texture.

![Figure 2: Texture example and its resulting 3d model in Blender](image)

This model of pants that was used turned out to be quite wide, as can be seen in the example pictures. Attempts to correct this will be made in the second phase of the assignment, but for now a texture of army pants seemed to fit this type of pants the best. After locating a suitable texture and applying it, attempts were made to implement the Blender model into an actual web interface using WebGL. A number of suitable libraries for easier use of WebGL exist, but one of the most-used and highest acclaimed libraries was Three.js [5], which comes with a Blender exporter module that exports the model to JSON format (JavaScript Object Notation) and this format can be imported by the Three.js framework. After exporting the Blender model to JSON and loading this into some basic WebGL with the texture of army pants, the result looked rather promising, as can be seen in Figure 3.
Phase 2:
The goal for this second phase was to create a parameterized model of the original piece of clothing, meaning that the user is able to modify a number of parameters through the web interface which will drastically alter the model. For example, there will be a slider to decrease the length of the trousers and to decrease the width of the cuffs. The reasoning behind implementing these parameters is to allow the user to change the model so that it matches the photograph(s) of the garment. Additionally, there will be a bit of research towards automating these parameters to immediately match the photo(s) and display a similar-looking three-dimensional model.

Creating parameters:
Blender is full of useful features regarding animating models. The default way of creating an animation is to create a so-called Armature that consists of a number of Bones. These Bones can then be linked to different parts of the model and changing the position or rotation of these Bones will alter the vertices belonging to these Bones accordingly. This did not seem like a very intuitive way of controlling different parameters of a pair of pants, but luckily there is another feature in Blender called Shape Keys that is often used for facial animations [6]. Since real faces consist of a large number of small muscles that work together to form a certain expression, it is extremely tedious work to model each of these muscles in a Bone.

Shape Keys create a way to control a number of vertices linearly by means of a slider that goes from 0 to 1 by default. This basically works as follows: The model in its current state is called $m_{\text{base}}$, and a vertex $v$ in this base model will be called $m_{\text{base}}(v)$. A Shape Key is in fact a translation on vertices. We will denote the translation of a vertex $v$ using Shape Key $s$ by $t_s(v)$. If the Shape Key were to be fully enabled, then:

$$\forall v: m_s(v) = m_{\text{base}} + t_s(v)$$

However, the user is able to specify the strength of a Shape Key by using the aforementioned slider. By default, this slider is a value between 0 and 1, but these limits can be edited at will. Taking the strength of a Shape Key into account, and calling it $p_s$, we generate the following formula for vertex positions using Shape Keys, using $k$ as the number of Shape Keys defined on the model:

$$\forall v: m(v) = m_{\text{base}} + \sum_{s=0}^{k} p_s t_s(v)$$
For example: Create a Shape Key called ‘Waist’. After selecting the Shape Key, the model is altered so the waist is rather small, see Figure 4a. This will be the shape of the model when ‘Waist’ is equal to 1. Changing the value for ‘Waist’ to -1 will increase the size of the waist, since this is the opposite of setting the value to 1, resulting in Figure 4b.

For the model of a pair of trousers there are a limited number of parameters that would allow the user to make any kind of pants. These are: ‘Overall Width’, ‘Waist’, ‘Trouser Length’, ‘Rise’, ‘Hips’, ‘Thighs’, ‘Cuffs’ and ‘Trouser Angle’. These parameters are mostly based on a web shop allowing customers to design custom made pants [7]. Figure 5 shows an overview of most of the previously stated parameters.

**Automating parameters (part 1):**

While this seems like a very potent and efficient way to define a set of parameters to alter the general model of a piece of clothing, Shape Keys introduce a number of problems as well. A practical problem for implementing this into a web interface is the face that the exporter to JSON does not include the Shape Keys. It simply takes the model after applying the Shape Keys and exports this to JSON. WebGL and Three.js do have support for Shape Keys (also known as Morph Targets) [8, 9], but implementing this requires programming work which is not suitable for this research assignment.
Another problem is the fact that the texture is applied before altering the vertices of the model. This directly conflicts with the goal of this phase, since the goal is to alter the model according to the photograph and then use this photograph as a texture on the model. Simply editing the model without changing the texture results in a compressed texture, see Figure 6. A possible solution to this problem is to perform the following steps:

1. Create a texture image with front and back photographs
2. Apply all Shape Keys according to the photographs
3. Create a new UV unwrap for the front and back side
4. Place the UV mappings on their matching photographs on the texture

As with the implementation of the Shape Keys in the web interface, creating a working prototype of the previously mentioned steps would require too much practical work to be feasible for this research assignment.

**Manually setting parameters:**
Manually performing these steps produces quite a pleasing result, as will be shown here. The example we will use is a pair of red shorts from Wehkamp [10]; See Figure 7:
The parameters of the model inside Blender were set to match the approximate shape of the pants according to these photographs, and afterwards the original photos were used to create the following texture, stretching the rise a little bit, and the length of the trousers considerably; See Figure 8a. This texture was placed on the altered Blender model, producing the results that can be seen in Figure 8b and 8c.

**Problems:**

There are a number of small flaws however. As a result of stretching the individual trousers of the red shorts, a number of discrepancies can be seen in the texture, especially with the folds around the crotch of the front view. These may be fixed by using better photo editing software. A larger issue however is the fact that there is a clearly visible line separating the front and back views, which is even more visible in the case of these red pants, because the lighting conditions of the two photographs are not the same. An example of this can be seen in Figure 9:
A number of possible solutions can be found for this larger problem, but they all have a number of pros and cons. It is very dependent on the wishes of the end-user which solution will work best. The ‘prettiest’ option would be to create a photo editing tool that is able to efficiently blend the two sides together. Producing such a tool is a lot of work, however, and therefore expensive. The ‘easy’ solution would be to let the user rotate the model to a certain degree, so that this line is never clearly visible, but this might seem like an arbitrary solution to the user and it severely limits the user in determining the look of this piece of clothing. With some additional tweaking to the model it might also be possible to make sure that the split between the two sides is a straight line from top to bottom. While it is still visible, this is something the user might understand and he/she might be able to ignore this little flaw.

A couple of problems could occur later on as well, when the steps for automating the setting of parameters are implemented. All textures are now manually created and matched onto a fixed UV mapping. If the UV mapping is created dynamically from the photographic images, it is possible that some areas of the mapping are not covered by a texture, resulting in a black area on the 3d model. However, it is also possible that the texture always completely covers the mapping, but a significant portion of the mapping does not cover the texture. This produces a 3d model with a stretched texture with some parts of the original photograph not showing on the model. The final system will need to find some kind of balance between these two issues.

If additional clothing pictures are used to create the texture, some areas of the model can be covered by multiple photographs. In that case, the system will need to efficiently merge the colors of these photographs to create a realistic result.

**Phase 3:**
The planning for this part of the assignment stated that some research should be done regarding background extraction for different garments, together with an investigation in merging multiple photographs from different angles onto the 3d model. After consultation it was agreed that this kind of research does not fit with the rest of this assignment and this phase would consist of modeling a different garment using the same techniques as described in previous phases, where different issues might be encountered. Some research into automating the steps described in phase 2 was also done, which will be elaborated first.

**Automating parameters (part 2):**
In order to implement a system that automatically transforms a given set of photographs to a 3d model, some additional information will be required, together with orthographical pictures of the garment. First of all, the system will need to know what kind of clothing is represented on the photographs, so the correct base model can be chosen. Since these pictures will be obtained from different web shops, also obtaining the type of clothing should be relatively easy. Creating base models for every clothing type may be rather time-consuming, but necessary nonetheless.

The next step will be to perform some Image Processing operations on the pictures to determine values for different parameters which can be applied to the model. Applying these parameters in practice can be done in a number of ways, depending on the wishes of the developer of the web interface that will implement this system. These parameters will always be calculated beforehand on
the server, to reduce the response time when retrieving a 3d model from the web-interface. In the next paragraphs two different methods will be discussed for applying the Shape Keys, a client-side implementation and a server-side implementation.

If the system is going to alter the model’s mesh client-side, it will send the base model as a JSON object to the client, together with all the different Shape Keys and the values that every parameter should have to define the final look of the model. The system will then need to be able to transform the model using its Shape Keys within JavaScript, which will then be shown to the user. The major downside of this approach is that there are rather few libraries available that handle Shape Keys efficiently in WebGL. While it is possible to use this method, it will take a significant time to implement.

The other approach is to transform the model server-side, before sending the model to the web interface where the user can view it. Blender has a large list of available command-line options, including an option to run a Python script. This Python script is able to alter values of the Shape Keys (which are already present on the model) and can also be used to create a JSON object after setting the required parameters. This approach is likely to take less time to implement than the client-side system, and it reduces the load of the client machine, since it will not have to calculate the positions of the Shape Keys.

Making a different model:
The next piece of clothing that was modeled was a shirt/sweater/pullover. The idea is the same as with the pair of trousers; a generic model is able to assume a variety of different shapes to look like different kinds of clothing. The used model can be seen in Figure 10 and it is an altered model of the pullover model by ‘dalenryder’ from the BlendSwap website [11]:

![Figure 10: Basic pullover model](image)

The alterations that were made are quite similar to the pants model. First, all unnecessary components were removed; in this case the pullover had an inner and an outer layer. The inner layer is useless for the system and it greatly increases the number of vertices in the model. Afterwards, the model still consisted of approximately 3000 vertices, and the Decimate modifier was applied to reduce this number by 50%. For clarification, Figure 10 is the model after applying these modifications.
In order to apply the texture, it is easier to change the pose of the model to the same pose that is usually used in photographs of pullovers, such as Wehkamp’s Twinlife T-shirt [12] in Figure 11a. The model will then look like Figure 11b.

Using the same methods as before, the UV mapping for the texture was created with Blender’s ‘Project from View’, resulting in the base texture of Figure 12.
As an example, the Twinlife T-shirt will be used that was shown before, resulting in the texture in Figure 13. Using this texture on the pullover model produces the result that can be seen in Figure 14.

The next task is to parameterize this pullover model to accommodate multiple garments. Since the model in the web interface does not need to match too precisely, only a limited amount of parameters were created in this case. The Shape Keys that were added are called: ‘V-neck’ and ‘Breasts’, which should be self-explanatory. For additional detail in the mesh, a ‘Collar’, ‘Hips’, ‘Shoulder’ and ‘Breast Width’ could be added.

Adding a Shape Key for ‘Arm Length’ seemed to be very hard, because the arms are not straight along an axis as with the trousers, which makes it difficult to scale the length of the arms correctly. In order to create a model representing a short sleeved shirt, the current model was copied and the sleeves were simply removed from this copied model. After a bit of editing and configuring the UV mapping of this short-sleeve model, the results look as can be seen in Figure 15 and 16 [13]:

![Figure 13: Twinlife pullover texture](image1.png)

![Figure 14: Resulting front and backview of the pullover in Blender](image2.png)
Problems:
As stated before, changing the length of something using Shape Keys is only really feasible if the Key is aligned to any kind of axis, so that scaling works correctly. But as can be seen in Figure 16c on the left arm, simply removing parts of the sleeve also produces some artifacts. These are fixable, but this is a bit tricky.

Another issue that is very noticeable in the last set of pictures is the texture and lighting inside the garment. In a real piece of clothing, the inside usually has a different color than the outside and it should also be quite a bit darker. This is not modeled at all in the current scenario, but it might be something to consider for future work.

Conclusion & Future Work
Creating a 3D model based on 2D photographs is very doable, as long as there is a decent base model to start from with a number of Shape Keys that allow for different parameters to be tweaked. The models that were used in this assignment were obtained from the Internet and edited to be more
suitable for this task. This will need to be done for every type of clothing, however, and this will consume a lot of time, especially when there is no model available for a specific garment.

Some 3d models will be easier to produce than others. Clothing with varying cutouts will most likely pose some problems, for example sandals or fancy dresses. The texture will need to have some sort of transparency layer in order to simulate the non-existence of cloth in certain areas of those garments.

It might also be necessary to create different base models for pieces of clothing designed for males or females. The pullover model that was used here had a Shape Key for altering the breast size, but real clothing designed for women will also have some other minor differences. The implementer of the final system will need to decide if these differences are important enough to dedicate resources into creating different models for male/female clothing.

The next big challenge is to build a system that is automatically able to detect the correct settings that the Shape Keys should have, applying those settings, morphing the original photograph to fit the texture that will be placed on the model, and then showing the result in a web interface.
Bibliography


