Estimating 3D Shape and Texture Using Pixel Intensity, Edges, Specular Highlights, Texture Constraints and a Prior

Sami Romdhani Thomas Vetter
University of Basel, Computer Science Department,
Bernoullistrasse 16, CH - 4056 Basel, Switzerland
{Sami.Romdhani, Thomas.Vetter}@unibas.ch

Abstract

We present a novel algorithm aiming to estimate the 3D shape, the texture of a human face, along with the 3D pose and the light direction from a single photograph by recovering the parameters of a 3D Morphable Model. Generally, the algorithms tackling the problem of 3D shape estimation from image data use only the pixels intensity as input to drive the estimation process. This was previously achieved using either a simple model, such as the Lambertian reflectance model, leading to a linear fitting algorithm. Alternatively, this problem was addressed using a more precise model and minimizing a non-convex cost function with many local minima. One way to reduce the local minima problem is to use a stochastic optimization algorithm. However, the convergence properties (such as the radius of convergence) of such algorithms, are limited. Here, as well as the pixel intensity, we use various image features such as the edges or the location of the specular highlights. The 3D shape, texture and imaging parameters are then estimated by maximizing the posterior of the parameters given these image features. The overall cost function obtained is smoother and, hence, a stochastic optimization algorithm is not needed to avoid the local minima problem. This leads to the Multi-Features Fitting algorithm that has a wider radius of convergence and a higher level of precision. This is shown on some example photographs, and on a recognition experiment performed on the CMU-PIE image database.

1. Introduction

Estimating the 3D shape of an object from 2D image data is a central problem of Computer Vision. In this paper, we address the task of recovering the 3D shape of a human face from a single photograph without information about its texture (albedo), its pose and the illumination environment. As a pixel value depends on these attributes, accurately estimating the 3D shape requires recovering its texture and the imaging conditions as well. Separating the contributions of light and albedo of the pixel intensity from a single image of the object is an ill-posed problem and requires a model of variations of the 3D shape, texture, and reflectance and a prior on its parameters. Hence, we use a 3D Morphable Model (3DMM) [4] that represents the 3D shapes and textures as a linear combination of shapes and textures principal components. These linear combinations are performed in a vector space of shapes and textures in which all 3D shapes and RGB textures vertices are in correspondence with a reference. The correspondence between instances of a class of object is a fundamental principle of accurate modeling of 3D Linear Object Class [15]. Thus, in order to recover the 3D shape and texture from an image, it is necessary to also estimate the correspondences between the input image and the model. The illumination is modeled by a Phong reflection model and one light source that can take into account as well as the diffuse component, the specular highlights and the attached and cast shadows.

The Stochastic Newton Optimization [4] algorithm fits the 3DMM to a single facial image thereby estimating the 3D shape, the texture and the imaging conditions. A drawback of this fitting algorithm is that it estimates the correspondences, the 3D shape, the texture and the imaging conditions from pixel intensity only. As a result, the energy function, or cost function, that it minimizes is highly non-convex and presents many local minima. In this paper, we present a new fitting algorithm, the Multi-Features Fitting (MFF) algorithm, that uses not only the pixel intensity but also other image cues such as the edges and the specular highlights (Figure 1). The resulting cost function is smoother and easier to minimize, making the system more robust and reliable. A question raised by this problem is how to fuse the different image cues to form the optimal parameter estimate. We chose a Bayesian framework and maximize the posterior probability of the parameters given the image and its features.

The problem of estimating the 3D shape of an object from 2D images is also addressed in the context of shape-from-shading, in which, usually, a Lambertian reflectance model is employed, ignoring both attached and cast shad-

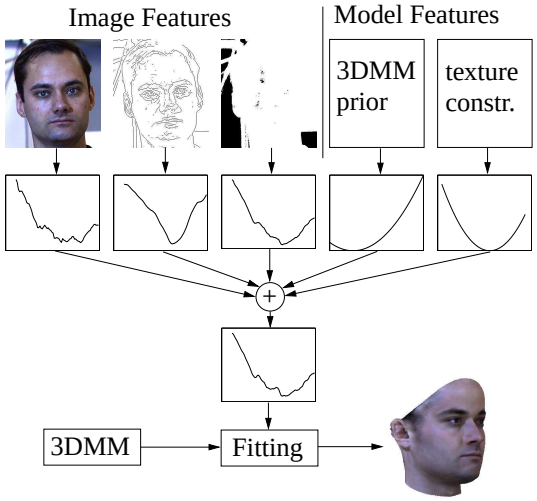

Figure 1. 3D Shape, texture and imaging parameters are estimated using the pixel intensity, the edges and the specular highlights detected in the input image, shown on the top row. Additionally, two model-based features are used: the 3D Morphable Model (3DMM) prior and texture constraints. Second row: plots of each feature cost function along the azimuth angle. These cost functions are combined yielding a smoother function that the one based on pixel intensity alone, which is, then, minimized.

ows as well as specular highlights. When the light direction and the texture is not known, several images are required to estimate the shape. If the pose is fixed and the light direction varies, three images are necessary, when ignoring the attached and cast shadows [13]. When the attached shadows are considered, Belhumeur and Kriegman [1] showed that the set of all N pixels images of an object at constant pose lies in a N-dimensional subspace, however, seven images were used to estimate the shape of a human face in order to obtain good recognition results. Common drawbacks of these photometric approaches are that 1) they use a simple reflectance model treating the cast and attached shadowed pixels, which do convey substantial information, as outlier and 2) they require multiple images of the same object to estimate the shape owing to the simple image model that they rely on. Here, the image model used is more complex and includes a prior on the shape, on the texture and a more general reflectance model, enabling the estimation from a single image.

Recently, Xiao *et al.* [16] presented an algorithm estimating the 3D shape of a face from a single image by fitting a 2D Active Appearance Model (AAM) and constraining the resulting shape to be a valid 2D projection of a 3D shape modeled by a 3DMM. They demonstrated that a 2D AAM is capable of instantiating the projection of any 3D shape in the span of a 3DMM, at the expense of using up to six times more parameters. They showed good fitting results obtained on the same individual used to train the 2D AAM. However, the accuracy of the recovered depth is questionable as the lighting is not estimated and hence the shading, which conveys most of the depth information in an image, is not explicitly employed.

The next section of this paper briefly reviews the main characteristics of the 3DMM. The third section motivates the use of multiple features to fit the 3DMM and presents the formula that combines the different features to maximize the posterior of the parameters. Then, the five features used by the Multi-Feature Fitting (MFF) algorithm are detailed in Section 4 before we conclude in Section 5.

2. 3D Morphable Model

The aim of this paper is to present a method able to estimate the 3D shape and texture of a face by registering a single image to the 3D Morphable Model (3DMM). The 3DMM is a correspondence-based model introduced in [3] and detailed in [4]. Its main properties follow: Firstly, the 3D shape and the texture of the faces of all individuals are represented on the reference frame (u, v). Hence a vertex of the reference frame represents the same facial feature across all heads instantiated by the model. The shape and texture are each modeled by PCA space. The pose and illumination variations are addressed using popular Computer Graphics techniques, such as the Phong reflectance model. Using a dense sampling of the reference frame, photo-realistic renderings are produced from the model parameters, θ, by warping an illuminated texture, $\mathbf{t}^C$, to the image frame using the inverse shape projection that maps the image frame to the reference frame [11]:

$$I^m(x, y; \theta) = \mathbf{t}^C(u, v; \theta) \circ \mathbf{p}^{-1}(x, y; \theta) \qquad (1)$$

3. Using multiple features

To date, the fitting algorithm that estimate the facial 3D shape and texture from a single image the most accurately, is the Stochastic Newton Optimization (SNO) algorithm [4]. Often it converges to a minimum close enough to the global minimum, such as the recovered shape and texture approximate well those of the photographed individual. However, in many cases, it converges to a local minimum far from the global one, yielding an unrealistic face, as seen on Figure 2. The second column on this figure shows the fitting results obtained with the SNO algorithm and a profile view synthesized using the recovered shape and the texture extracted from the input image. It is apparent on this rendering that the eyebrows are not properly recovered due to a lack of correspondence between the input image and the model. It is also clear that the contour is not accurately fitted as some of the background is visible on the extracted texture (bottom row). These artifacts are not present on the

face estimated using the Multi-Feature Fitting (MFF) algorithm presented in this paper (third column).

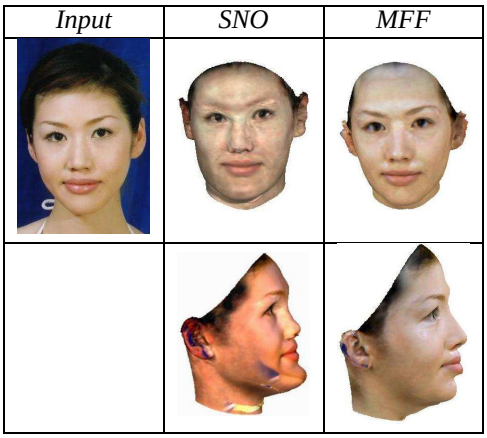

Figure 2. Example of poor fitting yielded by the Stochastic Newton Optimisation algorithm. Top row: fitting results with the SNO algorithm and with the algorithm presented in this paper using as input the photograph on the left. Bottom row: Novel view of the fitting result of the top row.

The contribution of this paper is a fitting algorithm that is more robust to the local minima problem. It is inspired from the field of pattern classification in which stronger classifiers are constructed from multiple weaker classifiers.

A fitting algorithm using multiple features aims at maximising the posterior probability of the model parameters given not only the input image, as it is done in SNO, but also different features of it. To make the notations more readable, we derive the posterior probability using two features, f^1 and f^2, and making their dependence on the input image I implicit. Using the Bayes rule and denoting by θ the ensemble of model parameters, we obtaine the following, assuming that the features are independent:

$$p(\theta|f^1, f^2) = \frac{p(f^1|\theta) \cdot p(f^2|\theta) \cdot p(\theta)}{p(f^1) \cdot p(f^2)} \qquad (2)$$

The aim of the MFF algorithm is to maximise the above equation with respect to the model parameters, θ. If we use deterministic algorithms for the feature extraction, then this equation is simplified to:

$$p(\theta|f^1, f^2) = p(f^1|\theta) \cdot p(f^2|\theta) \cdot p(\theta) \qquad (3)$$

Maximising this equation is equivalent to minimising the negative of its logarithm:

$$-\ln p(\theta|f^1, f^2) = -\ln p(f^1|\theta) - \ln p(f^2|\theta) - \ln p(\theta) \quad (4)$$

Hence, in the rest of this paper, the negative of the logarithm of a probability function is called a *cost function*, as it is to be minimized. As a result, this equation says that if the features are independent and deterministic, maximizing their joint posterior probability is equivalent to minimizing the sum of their cost functions.

The use of combined classifiers, in the pattern classification field, stems from the fact that each classifier has different strengths and their combination maximises the overall performances. For example, an optimal cascaded classifier is constructed with a chain of classifiers such as the classifier i achieves its maximum discriminations on the patterns that are poorly discriminated by the first $i-1$ classifiers [12]. Similarly, for the fitting problem, optimally, the local minima of the features cost functions should be located in different areas of the parameter space. Then, the combination of the multiple cost functions would have fewer local minima and a smoother behaviour on the parameter space as it is shown in Figure 3.

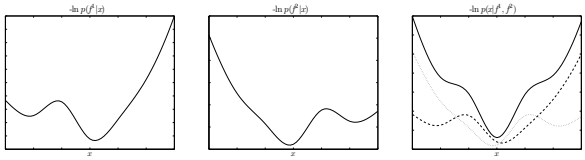

Figure 3. Two single feature cost functions with, each, two minima. The last plot shows the multiple feature cost function (plain line) with one global minimum yielded by the addition of the two weak cost functions (dashed lines).

If there is a local minimum that is persistent across all features, then it would persist in the combined cost function. If the features are independent, this is not likely to happen. To make this even less likely to happen, a good strategy is to use as many features as possible.

4. Multi-Features Fitting algorithm

In this section, we motivate and derive a cost function for each feature used in this work: the pixel intensity, the edge, the specular highlight, the prior, and the texture constraints features.

4.1. pixel intensity feature

Fitting the pixels intensity aims to recover the correspondence, the 3D shape, the texture, and the imaging parameters, by matching the color of the model image to the color of the input face image. Hence, the feature used to fit the pixel color, f^c, is simply the input image:

$$f^c(I(x,y)) = I(x,y) \qquad (5)$$

If the pixels are assumed independent and identically distributed with a Gaussian probability, the pixel intensity cost

function is given by:

$$-\ln p(I(x,y)|\theta) \propto \frac{1}{2}\sum_i \left(I(x_i,y_i) - I^m(x_i,y_i)\right)^2 \quad (6)$$

The cost function, C^c, can then be written as the inner product of the error vector $\mathbf{e^c}$:

$$C^c = \frac{1}{2}\mathbf{e^{c\intercal}} \cdot \mathbf{e^c}, \quad (7)$$

$$\text{with} \quad \mathbf{e_i^c} = I(x_i,y_i) - \mathbf{t}^C(u,v;\theta) \circ \mathbf{p}^{-1}(x_i,y_i;\theta) \quad (8)$$

Composing this last equation on its right with the forward shape mapping $\mathbf{p}(u,v;\theta)$, detailed in [11], using the shape composition axiom [11], and sampling the reference frame instead of the image frame, yields a cost function that depends on the forward shape mapping:

$$\mathbf{e_i^c} = I(x_i,y_i) \circ \mathbf{p}(u_i,v_i;\theta) - \mathbf{t}^C(u_i,v_i;\theta) \quad (9)$$

The SNO algorithm minimizes the sum of this cost function and a cost function that depends on the prior (Section 4.4). Unfortunately, this former cost function is perturbed by many local minima, as it is shown on Figure 4 that plots values of the pixel intensity cost function obtained by varying the azimuth angle around its global optimum and using average values of the other model parameters.

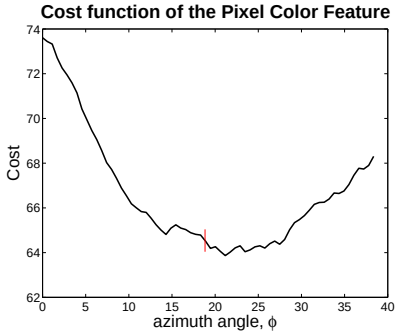

Figure 4. Plot of the Pixel intensity cost function along variations of the azimuth angle of $\pm 20°$ around the optimum.

Representing a shape in the 3DMM requires putting it in correspondence with the reference frame. Its depth is estimated using the shading recovered from the pixel intensity by estimation of 1) the texture (using the prior model) and 2) the lighting environment. In the SNO algorithm, the pixel intensity feature is used to recover both the correspondence and the shading. As the shading depends directly on the pixel intensity, it is reasonable to estimate it from the pixel intensity feature. However, the correspondences are implicit in the cost function of Equation 4, this feature is, then, not optimal to evaluate them.

4.2. Edge feature

The image edges provide informations about the 2D shape independently of the texture and of the illumination. Hence, the cost function used to fit the edge features provides a more direct constraint on the shape and pose parameters. This is why its plot across azimuth direction (made on the same image and with the same parameters as plot of Figure 4, but on a wider range) is much smoother. The edge feature is useful to recover the correspondences of specific facial characteristics (eyes, eyebrows, mouth, nose). On the other hand, it does not carry much depth information. So, it is beneficial to use the edge and intensity features in combination.

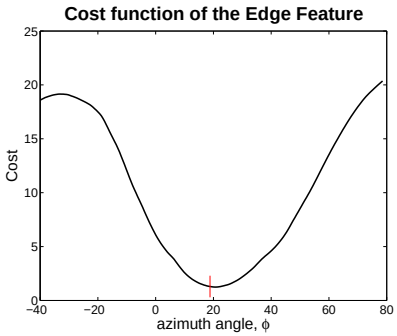

Figure 5. Plot of the edge cost function along variations of the azimuth angle of $\pm 60°$ around the optimum.

The edge feature is defined as the binary edge map given by a canny edge detector [6]: $f^e(I(x,y)) = \mathsf{canny}(I(x,y))$. If the image edge pixels are independent and identically normally distributed over the image, then the edge feature cost function is given by the following equation:

$$-\ln p(f^e(I)|\theta) \propto C^e = \mathbf{e^{e\intercal}} \cdot \mathbf{e^e}, \quad (10)$$

$$\text{with} \quad \mathbf{e_i^e}(\theta) = \|\mathbf{q}^{\mathbf{e}}_{k(i)} - \mathbf{p}_i(\theta)\| \quad (11)$$

where $\mathbf{q}^{\mathbf{e}}_{k(i)}$ is the 2D position of an image edge pixel in correspondence with the model edge point whose index is i and that is projected in the image frame to $\mathbf{p}_i$, with $i = 1,\ldots,N_e$, where N_e is the number of model edge points. The mapping between the image and the model edge points is denoted by $k(i)$. To estimate it, we use the same rule as the Iterative Closest Point (ICP) algorithm [2]: The model edge point i is set in correspondence with the input image edge point, $\mathbf{q}^{\mathbf{e}}_j$, closest to it:

$$k(i) = \arg \min_{j=1,\ldots,J} \|\mathbf{q}^{\mathbf{e}}_j - \mathbf{p}_i\| \quad (12)$$

where J is the number of image edge points. Similar to the ICP algorithm, fitting the edge involves, at each iteration, the following two steps: First, the correspondence

mapping, $k(i)$, is computed. Then, given this mapping, the model parameters are updated to reduce the cost function of Equation (10). However, performing these two steps separately is not optimal, as modifying $k(i)$ alters the minimum. Hence, it is desirable, to update $k(i)$ along with the parameters θ. Levenberg-Marquardt ICP (LM-ICP) is an algorithm, proposed by Fitzgibbon [8], addressing this problem. The difference between the Fitzgibbon algorithm and this one, is that in [8], only the rigid parameters were estimated. Here, not only the rigid parameters are estimated but also the non-rigid parameters.

The trick is to use the Chamfer Distance Transform (CDT) [5]. It is defined as the mapping $D(\mathbf{x})$ that associates a point of the image space, $\mathbf{x}$, with the distance to the closest edge point:

$$D(\mathbf{x}) = \min_{j=1,\dots,J} \|\mathbf{q}_j^{\mathbf{e}} - \mathbf{x}\| \tag{13}$$

The CDT at the image point, $\mathbf{x}$, in essence, incorporates two pieces of information: The closest edge point to $\mathbf{x}$ (which, according to the ICP algorithm, is set in correspondence with $\mathbf{x}$); and the distance between these two points. Hence, it replaces both the mapping $k(i)$ and the distance $\|\mathbf{q}_{k(i)}^{\mathbf{e}} - \mathbf{p}_i\|$. As a result, using the CDT, the edge cost function is transformed to:

$$\mathbf{e}_i^{\mathbf{e}}(\theta) = D(\mathbf{p}_i(\theta)) \tag{14}$$

The edge cost function is computed simply by sampling the CDT at the position where the edge model points are projected under the shape and projection parameters.

Note that the CDT depends only on the input image, not on the model parameters. Hence it can be computed only once, at the beginning of the fitting algorithm. An efficient implementation of the computation of $D(\mathbf{x})$ is proposed in [7] , whose complexity is $O(2wh)$, where w and h are the width and height of the input image.

Differentiating the edge cost function of Equation (14), with respect to a model parameter θ_j, is straightforward and yields:

$$\frac{\partial \mathbf{e}_i^{\mathbf{e}}}{\theta_j} = \frac{\partial D}{\partial x}(\mathbf{p}_i) \cdot \frac{\partial \mathbf{p}_i^x}{\partial \theta_j} + \frac{\partial D}{\partial y}(\mathbf{p}_i) \cdot \frac{\partial \mathbf{p}_i^y}{\partial \theta_j} \tag{15}$$

The advantage of the LM-ICP algorithm is that the derivatives of the CDT encode both the derivative of the mapping function $k(i)$ and the one of the distance between corresponding points. As opposed to the ICP algorithm, in which the derivatives do not depend on the mapping and hence regard it as being constant for each update of the parameters. As a result, in the LM-ICP algorithm, used here, the parameter update takes into account the variation of the mapping function. This is the reason why this scheme is called 'soft correspondence'.

Model Edges The cost function of Equation 10 depends on the model edge points $\mathbf{p}_i$, with $i = 1, \dots, N_e$, which form a subset of the model vertices. We distinguish between two types of model edges: The textured edges result from a texture change in the inner part of the face and delimit the facial features such as the lips, eyes and eyebrows. The contour edges, on the other hand, are defined by the points that are on the border between the face area and the non-face area in the image plane.

The location of the textured edges is constant on the reference frame. As the reference frame is densely sampled by 3D vertices, the textured edges form a constant subset of model vertices. The subset of model vertices chosen as textured edges is shown in white on the texture map of the average head on Figure 6.

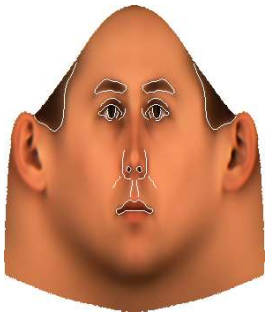

Figure 6. The textured edges on the face surface are shown on this figure in white on the texture map of the average head.

As opposed to the textured edges, the contour edges do depend on the rigid parameters. The fitting of contour edges proceeds in two steps: First, the model vertices on the contour for the current rigid parameters are selected, then these vertices are fitted to the input edges in the same way as for textured edges, i.e. using the CDT. The selection of model contour edges is performed by the following algorithm that includes four step whose results, on an example image, are shown in Figure 7.

1. *Vertex map*: First the vertex map is constructed. A vertex map is a rendering of the face, but instead of setting the R, G, B color at one pixel, a vertex index is set. The vertex whose index is set at one pixel, is the one that is projected nearest to this pixel.

2. *Binary vertex map*: The vertex map is converted to a binary format by thresholding the vertex map to one. The binary vertex map has, hence, a value of one for any pixel in the face area of the image, and zero, for any pixel outside the face area.

3. *Erosion and subtraction*: The morphological operation of erosion is applied to the binary vertex map, using as structuring element a disk of radius of one pixel. Then this eroded binary image is subtracted to the binary vertex map obtained at step 2. The resulting binary image is the contour map. Each pixel on this contour map

set to one is on the contour of the face. Sampling the vertex map (obtained at step 1) on these pixels yields the indexes of the model vertexes on the contour.

4. *Contour vertex indexes*: The 3DMM does not model the entire skull. It only models the points ranging from one ear to the other and from the top of the forehead to the neck. As a result, some parts of its contour, on the lower part of the neck and on the top of the forehead, are artificial. These artificial contour points are removed at this step. The good news is that this artificial contour is present always in the same area of the face. Hence, it is constant. Thus, a list of the vertex indexes on the artificial contour can be made, and all the contour vertex indexes yielded by step 3 that are on the artificial contour list are removed. This step yields the indexes of the contour vertex used for fitting.

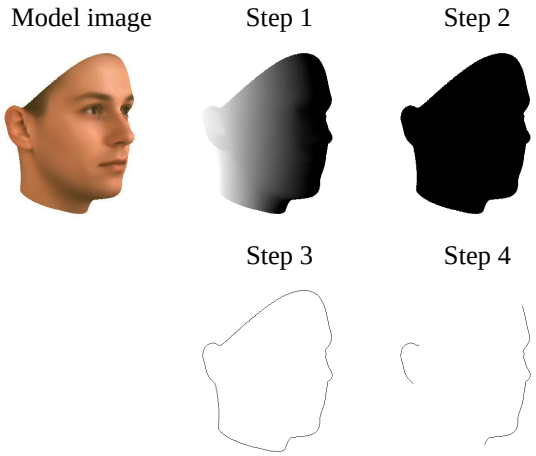

Figure 7. Rendering of the images yielded by the four steps of the algorithm computing the model contour, when applied to the 'Model image' shown in the upper left part of this figure.

4.3. Specular highlight feature

The specular highlights are easy to detect: The pixels with a specular highlight saturate. Additionally, they give a direct relationship between the 3D geometry of the surface at these points, the camera direction, and the light direction: A point on a specular highlight has a normal that has the direction of the bisector of the angle formed by the light source direction and the camera direction (see Figure 8).

The benefit of the specular highlight feature is shown on an example on Figure 9. Image (a) presents an input image (at a frontal pose) with a specular highlight on the left of the tip of the nose. Two fittings of this photograph are performed. The first one without the specular highlight feature (second column) and the second one using this feature (third column). Renderings of these two fittings at a frontal pose

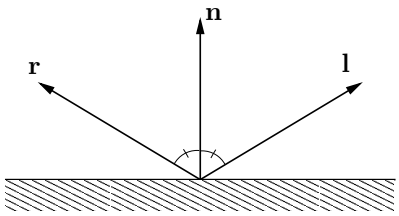

Figure 8. The light vector, $\mathbf{l}$, is reflected along the reflection vector, $\mathbf{r}$.

(top row) appear to be similar. Renderings of the contour at a profile view (using the fitted parameters of the top row) are shown on the bottom row. It is clearly apparent that the tip of the nose is much closer to the ground truth on the fitting result obtained with the specular highlight feature. To sum up, the specular highlight feature provide additional depth information improving the 3D shape estimation.

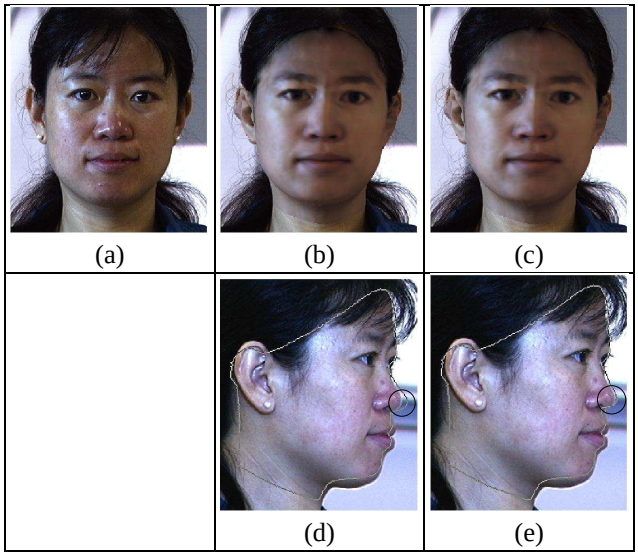

Figure 9. Comparison of fitting results with and without the specular highlight feature. *(a)* **Input photograph.** *(b)* **Fitting result of the photograph (a) without specular highlight feature.** *(c)* **Fitting result of the photograph (a) using the specular highlight feature.** *(d)* **Contour of the fitting result (b) at a profile view rendered on top of a profile photograph of the same person.** *(e)* **Contour of the fitting result (c) at a profile view.**

A point on a specular highlight as a normal equal to:

$$\mathbf{n}(\theta) = \frac{\mathbf{r}+\mathbf{l}}{\|\mathbf{r}+\mathbf{l}\|} = \frac{\mathbf{v}(\theta)+\mathbf{l}}{\|\mathbf{v}(\theta)+\mathbf{l}\|} \tag{16}$$

where $\mathbf{r}$ is the reflection vector, which, for a point on the specular highlight, has the same direction as the viewing vector $\mathbf{v}(\theta)$, that connects the point to the camera. Hence, we define the specular highlight cost function as follows:

$$C^s = \mathbf{e}^{\mathbf{s}\mathrm{T}} \cdot \mathbf{e}^{\mathbf{s}} \text{ , with } \quad \mathbf{e}_i^{\mathbf{s}} = \mathbf{n}_i(\theta) - (\mathbf{v}_i(\theta) + \mathbf{l}) \tag{17}$$

where $\mathbf{e}_i^{\mathbf{s}}$ is a three elements sub-vector of $\mathbf{e}^{\mathbf{s}}$; $\mathbf{n}_i$ and $\mathbf{v}_i$ are, respectively, the normal and the viewing vector of the vertex i that is projected on a pixel on a specular highlight.

4.4. Gaussian Prior feature

The 3DMM is based on a Gaussian probability model for both the 3D shape and the RGB texture. In the model construction, a PCA is applied to diagonalize the shape and texture covariance matrices. Hence the cost function of the prior feature is as follows:

$$C^p = \sum_i \frac{\alpha_i^2}{\sigma_{S,i}^2} + \sum_i \frac{\beta_i^2}{\sigma_{T,i}^2} \tag{18}$$

where α_i and β_i are the shape and texture PCA coefficients (i.e. they are elements of the vector θ); $\sigma_{S,i}$ and $\sigma_{T,i}$ are, respectively, their standard deviations. This prior term is also used in the SNO fitting algorithm [4].

4.5. Texture constraint feature

In order to accurately estimate the 3D shape, it is required to recover the texture, the light direction and its intensity. To separate the contribution of the texture from light in a pixel intensity value, a texture prior model is used (see previous section). However, it appears that this prior model is not restrictive enough and is able to instantiate invalid textures. To constraint the texture model and to improve the separation of light from albedo, we introduce a feature that constraints the range of valid albedo values.

An example is presented in Figure 10. The second column shows a poor fitting where the light intensity is underestimated and the texture over-estimated. As these effects compensate each other, the fitting result rendered in image (b) look plausible. However when the texture from the image is extracted and the illumination inverted, the result is affected by some blueish regions that look unnatural (image (d)). The process of texture extraction and inverse illumination, which is required to perform a re-rendering of a photograph in novel conditions, is detailed in [10].

The valid intensity range of a pixel is $[0, 255]$. This constraint is not enforced by the Gaussian prior texture model. Imposing a valid range over all the texture points make the texture less likely to be over-estimated, as in the second column of Figure 10, and hence, make the light intensity less likely to be under-estimated. The results shown in the third column of the same figure are obtained by imposing, during fitting, the model color to be in the range $[5, 250]$. This is achieved by using the following cost function:

$$C^t = \frac{1}{2}\mathbf{e}^{\mathbf{t}\mathbf{T}}\cdot\mathbf{e}^{\mathbf{t}}, \text{ with } \quad \mathbf{e}_i^{\mathbf{t}} = \begin{cases} t_i(\theta) - l & \text{if } t_i(\theta) < l \\ 0 & \text{if } l \leq t_i(\theta) \leq u \\ t_i(\theta) - u & \text{if } t_i(\theta) > u \end{cases} \tag{19}$$

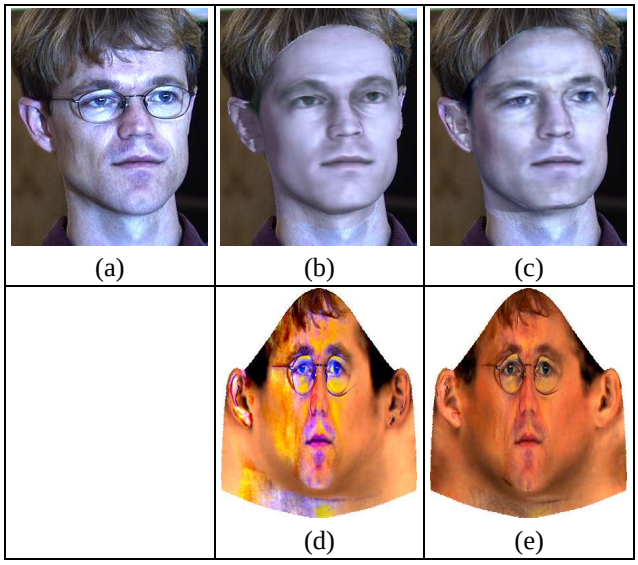

Figure 10. Comparison of fitting results with and without the texture constraint feature. ***(a)*****: Input photograph.** ***(b)*** **and** ***(c)*****: Rendering of the fitting result obtained without and with, respectively, the texture constraint feature.** ***(d)*** **and** ***(e)*****: Inverse illuminated extracted texture obtained from the fitting results shown in images (b) and (c), respectively.**

where $t_i(\theta)$ is the color intensity of a channel of a model vertex used for fitting, and l and u are the lower and upper bounds of the valid intensity range. Note that this feature is not a hard constraint, but is rather a soft constraint, whose influence on the resulting estimate is relative to the other features.

4.6. Multi-Features Fitting algorithm

Equation (4) says that when the features are independent and deterministic, maximizing the posterior of the model parameter is equivalent to minimizing the sum of the negative likelihoods. As the negative likelihood of a feature is proportional to the feature cost function, the overall cost function is a linear combination of the features cost functions:

$$\min_\theta \tau^c C^c + \tau^e C^e + \tau^s C^s + \tau^p C^p + \tau^t C^t \tag{20}$$

In this function, the τ's are the feature weighting factors. It is minimized using a Levenberg-Marquardt optimization algorithm [9]. Similarly to the SNO algorithm, the fitting is initialized using a set of five to seven (depending on the pose) manually set anchor points located at the eyes corners, nose tip, mouth corners and on the contour. Standard initial values of the shape, texture, illumination and rigid parameters are used (average shape and texture, frontal pose and one frontal directed light).

A Matlab implementation of the MFF Fitting algorithm requires, on average, 70 seconds to reach convergence on a 3.0 GHz Intel Pentium IV computer. On the other hand, the SNO algorithm requires 4.5 minute on a 2.0 GHz computer.

Identification The MFF algorithm was experimented using the `light` portion of the CMU-PIE facial image database [14] that includes 22 different illumination conditions at three poses (front, side and profile) for 68 individuals. Several identification experiments, using one image per individual at a side view as the gallery, and a cosine-based distance measure between shape and texture model parameters, were performed. Averaging all these experiments over the probe sets (one probe set includes photographs taken at one of the 3 poses with one of the 22 flash lights), yielded 94.6% correct identification with the MFF algorithm and 94.2% with the SNO algorithm [4, Table 2]. More information about these experiments is available in [4]. To sum up, the identification performances of both algorithms are similar, however, the efficiency of MFF is substantially improved, owing to the fact that the cost function is smoother and a stochastic optimization is not required.

5. Discussion and Conclusion

The image analysis problem tackled in this paper is the estimation of the 3D shape, the texture of a face and the imaging parameters from a single input facial image. The only clue available in a single image about depth is contained in the shadings and the shadows. However using this information requires the illumination environment and the texture of the face to be known. As these are not known, the problem is ill-posed. One way out of this dilemma is by using prior knowledge. Therefore, texture and shape PCA models are employed to constraint the set of possible solution. A lighting model is also used, the Phong reflectance model, which has a few parameters when only one light source is handled. However, even applying this prior models is not enough to obtain an accurate estimate of the 3D shape when just a few manually set anchor points are used as input. This is because the cost function to be minimised is highly non-convex and exhibits many local minima. In fact, the shape model requires the correspondence between the input image and the reference frame to be found. Using only facial color information to recover the correspondence is not optimal and may be trapped in regions that present similar intensity variations (eyes/eyebrows, for instance). Other features could be used to obtain a more accurate estimate of the correspondence and, as a result, of the 3D shape. One example of such feature is the edges. Other features that improve the shape and texture estimate are the specular highlights and the texture constraints. The specular highlight feature uses the specular highlight location, detected on the input image, to refine the normals and, thereby, the 3D shape of the vertices affected (see Figure 9). The texture constraint enforces the estimated texture to lye within a specific range, which improves the illumination estimate (see Figure 10).

To avoid getting trapped in local minima, the SNO algorithm used a stochastic optimization that introduced a random error on the derivative of the cost function. This is performed by sampling the cost function at a very small number of points (40 pixels). This random error enables the algorithm to escape from local minima. On the other hand, the MFF algorithm, presented in this paper, has a smoother cost function owing to the various sources of information used. Hence, it is not required to use a stochastic optimization algorithm and the pixel intensity feature may be safely sampled at many more points providing a more stable parameter update. Hence, much fewer iterations are needed to reach convergence (a few tens instead of several thousands in the case of SNO). As a result, as well as being accurate, MFF is also more efficient than SNO.

References

[1] P. Belhumeur and D. Kriegman. What is the set of images of an object under all possible illumination conditions. *IJCV*, 28(3):245–260, 1998.

[2] P. J. Besl and N. D. McKay. A method for registration of 3d shapes. *PAMI*, 14(2):239–256, 1992.

[3] V. Blanz and T. Vetter. A morphable model for the synthesis of 3D-faces. In *SIGGRAPH 99*, 1999.

[4] V. Blanz and T. Vetter. Face recognition based on fi tting a 3d morphable model. *PAMI*, 2003.

[5] G. Borgefors. Hierarchical chamfer matching: A parametric edge matching algorithm. *PAMI*, 10(6):849–865, 1988.

[6] J. Canny. A computational approach to edge detection. *PAMI*, 8(6), 1986.

[7] P. Felzenszwalb and D. Huttenlocher. Distance transforms of sampled functions. Technical Report TR2004-1963, Cornell Computing and Information Science, 2004.

[8] A. Fitzgibbon. Robust registration of 2d and 3d point sets. In *BMVC*, volume 2, pages 411–420, 2001.

[9] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes in C : the art of scientific computing*. Cambridge University Press, 1992.

[10] S. Romdhani. *Face Image Analysis using a Multiple Feature Fitting Strategy*. PhD thesis, Univeristy of Basel, Jan. 2005.

[11] S. Romdhani and T. Vetter. Effi cient, robust and accurate fi tting of a 3d morphable model. In *Proceedings of the International Conference on Computer Vision*, 2003.

[12] J. Schürmann. *Pattern classification: a unified view of statistical and neural approaches*. J. Wiley & Sons, Inc., 1996.

[13] A. Shashua. On photometric issues in 3d visual recognition from a single 2d image. *IJCV*, 21:99–122, 1997.

[14] T. Sim, S. Baker, and M. Bsat. The cmu pose, illumination and expression (pie) database of human faces. Technical report, CMU, 2000.

[15] T. Vetter and T. Poggio. Linear object classes and image synthesis from a single example image. *PAMI*, 1997.

[16] J. Xiao, S. Baker, I. Matthews, R. Gross, and T. Kanade. Real-time combined 2d+3d active appearance model. In *CVPR*, pages 535–542, 2004.