Being John Malkovich

Ira Kemelmacher-Shlizerman[1], Aditya Sankar[1], Eli Shechtman[2], and Steven M. Seitz[1,3]

[1] University of Washington
{kemelmi,aditya,seitz}@cs.washington.edu
[2] Adobe Systems
elishe@adobe.com
[3] Google Inc.

Abstract. Given a photo of person A, we seek a photo of person B with similar pose and expression. Solving this problem enables a form of *puppetry*, in which one person appears to control the face of another. When deployed on a webcam-equipped computer, our approach enables a user to control another person's face in real-time. This image-retrieval-inspired approach employs a fully-automated pipeline of face analysis techniques, and is extremely general—we can puppet anyone directly from their photo collection or videos in which they appear. We show several examples using images and videos of celebrities from the Internet.

Key words: Image retrieval, facial expression analysis, puppetry

1 Introduction

> "Ever wanted to be someone else? Now you can."
> —tagline from the film *Being John Malkovich*

In the film Being John Malkovich, a puppeteer (played by John Cusack) discovers a portal that allows him to control the real life movements of John Malkovich (played by himself). While puppeteering real people might seem a bit far fetched, it should be possible to control *digital likenesses* of real people. In particular, we seek to construct a photographic simulation (i.e., avatar) of John Malkovich that you can control by moving your face; when you smile, move your head, or close your eyes, you see John Malkovich doing the same.

One way to attack this puppetry problem might be to create a photo-realistic 3D model of John Malkovich's head, instrumented with several degrees of freedom (e.g., mouth open, head rotate, etc.), and map the user's head motions to the model. Indeed, most prior work on avatars and puppetry has followed a similar approach [1,2,3,4]. However, creating a sufficiently accurate model of a real person is a major challenge, particularly if we don't have access to the actor to pose for a scanning session.

Instead, we recast the puppetry problem as image retrieval: given a query image or video of person A (the user), and a set of images of person B (John

Fig. 1. One person's expressions (top row) are mapped to another person's face (bottom row) by real-time matching to an image database. In this case, the input is a video of Cameron Diaz, and the database is formed from a video (John Malkovich, bottom-left 4 images) or an unstructured set of photographs downloaded from the Internet (George W. Bush, bottom-right 4 images).

Malkovich), find and display the best matching image or image sequence of person B. This approach has a number of key advantages, as follows. First, we avoid the complexity and technical difficulty of creating a 3D model and parameterizing expressions. Second, because the output are real photos, we can capture all the complexities of the face (hair, light scattering, glasses, etc.) that are difficult to simulate. And finally, the approach operates on just about any set of photos or video, and is fully automatic. I.e., it is possible to create an avatar simply by typing an actor's name on an image/video search site and processing the resulting images and/or videos. The approach can also be used to drive one video with another; Fig. 1 shows Cameron Diaz driving John Malkovich and George W. Bush.

The main challenge to making this image retrieval approach work is defining a metric that can reliably match an image of person A to an image of person B with similar pose and expression. Significantly complicating this task is the fact that the facial characteristics, lighting, and sex of the two people may be different, resulting in large appearance variation between person A and person B. The main contribution of this paper, in addition to posing puppetry as image retrieval, is a processing pipeline that yields high-quality real-time facial image retrieval, and that operates reliably on both video and unstructured photo collections. While this pipeline is based on existing pieces from the literature, we argue that it is not at all straightforward to create a real-time system that achieves the results presented here; the contribution is the system and the novel application.

Our approach operates as follows: images of the target face (person B) are processed using a combination of face detection, extracting fiducial features (e.g., eyes, nose, mouth), estimating pose, and aligning/warping to a frontal pose. The user (person A) is tracked to determine head-pose at video rates. After alignment, the user's face is compared to all aligned images of the target face using a fast implementation of Local Binary Patterns (LBP) and chi-square

matching, and the best match is returned. We incorporate additional terms for temporal coherence. We've found the resulting approach to work remarkably well in practice. Fig. 2 shows our interactive web-cam-based system in action for a case of a user driving George Clooney. Video captures can be found on the paper's website: http://grail.cs.washington.edu/malkovich/.

1.1 Related Work

There is quite a large literature on avatars, puppetry, and performance-driven animation. We therefore limit our discussion to methods that specifically involve tracking video of a user's face to drive the appearance of a different person or model. And while tracking mocap markers to drive scanned or hand-crafted animated models is a mainstay of digital special effects (famous examples in movies include Polar Express and Avatar), we focus here on markerless solutions that operate from photos alone.

While there are a number of markerless puppetry techniques, the vast majority of these methods assume the availability of a 3D model of the target face, e.g., [3]. A very recent example is the work of Weise et al. [4], who demonstrate very impressive puppetry results using a real-time structured light scanner to drive a previously captured 3D model. 3D puppetry via face tracking is starting to becoming mainstream—Logitech's webcams now come with software that tracks a users's face and gestures to control an animated on-screen avatar.

Pighin et al. [1] were among the first to demonstrate purely image-based face capture and puppetry. In this work, the model was created by manually specifying a set of correspondences between features on a 3D head model and features in several photos of the person. More recent work in this vein includes Zhang et al. [2] who used video to create the 3D model and simplified the manual work to 5 features in two views.

Although they do not relate to puppetry per se, we are inspired by Kumar et al.'s face search [5] and Bitouk et al.'s swapping [6] work, which operate robustly on large collections of images downloaded from the Internet, and Goldman et al. [7] who enable mouse-based face posing from a database of tracked video frames.

We note however, that no prior work has enabled puppetry with arbitrary, unstructured photo collections. This capability dramatically broadens the applicability of puppetry techniques, to *any* person whose photos are available.

2 Overview

Our system allows fully automatic real time search of similar facial expressions of a target person given image queries from a webcam. The user can be any person; there is no need to train the system for a specific user. The user can make expressions to the camera, as well as change the pose of the head, and get in real time similar expressions and poses of the target face. The target face can be represented by a video or by a set of photos (e.g., photos of a celebrity downloaded from the Internet). In each query frame the face is tracked

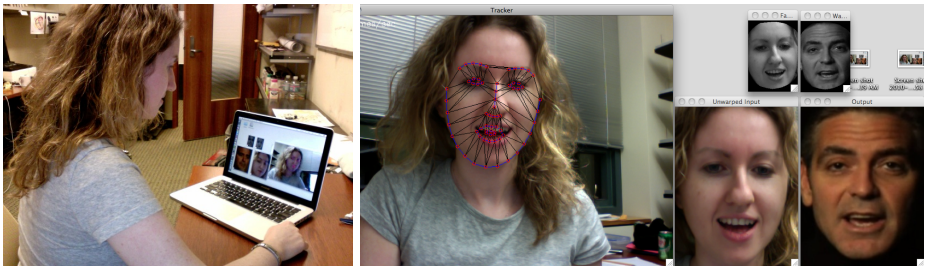


Fig. 2. Our puppeteering system. Left - our setup. Right - screen capture of our system: the face tracker applied on an input webcam video; Top grayscale small pair - the input face after cropping and warping to frontal view and the matching target face found by our method; Bottom large pair - user face and matching target image (raw input).

automatically, the 3D position of the face is recovered, then the detected face region is fitted to a template 3D model of a face and is warped to a frontal pose. In the process, we estimate the location of the eyes and mouth of the user's face. We consider each of these regions independently and for each region compute a Local Binary Patterns (LBP) feature vector. The same is done for each photo (or video frame in case a movie is available) of the target person. We then compute distances between the mouth region of the target face and the mouth region of the user face, and similarly for the eyes regions. These two distances define our appearance distance. In addition, we compute the distance between the 3D pose of the user and the 3D pose of the target in each image, and a temporal continuity distance. These three distances are combined together to find the best match in terms of appearance, pose and continuity. We describe our geometric alignment method in Section 3, and the appearance representation in Section 4. In Section 5 we present our distance function. Results and evaluations of the method are presented in Section 6.

3 Image alignment to canonical pose

In this section we present a framework to align the images of the user and target faces to a canonical (frontal) pose. The input to the method is a live video feed (e.g., webcam) or a video of a person. We first automatically track the face in each frame of the video, using the algorithm of Saragih et al. [8]. The aim here is to estimate the location of the face and its pose in each given frame, and use these to perform warping of each image to a canonical pose.

[8] is based on fitting a parametrized shape model to an image such that its landmarks correspond to consistent locations on the face. In particular, in each frame, predictions regarding locations of the model's landmarks are made by utilizing an ensemble of local feature detectors, and then combined by enforcing a prior over their joint motion. The distribution of the landmark locations is

Fig. 3. Results of our warping procedure (based on 3D pose estimation and using 3D template model). Top row: images aligned in 2D . Bottom row: warped images (3D alignment).

represented non-parametricaly and optimized via subspace constrained mean-shifts.

For the target person, in case we have a video available we estimate the location of the face and landmarks using the same procedure as the user's face. In case the target face is represented by a collection of photos we cannot use tracking. We instead apply a face detector [9] followed by a fiducial points detector [10] that provides the landmarks (the left and right corners of each eye, the two nostrils, the tip of the nose, and the left and right corners of the mouth). Given the landmark positions we recover the 3D position of the face. For this we use a neutral face model from the publicly available spacetime faces dataset [11] as our template model. Given the points on the image and the corresponding pre-labeled points on the 3D template model we first subtract the centroid from each of the point arrays, recover a linear transformation between them and then find rotation and scale using RQ decomposition. The yaw, pitch and roll angles are then estimated from the rotation matrix.

Given the estimated pose we can transform the template model to the orientation of the face in the image, and consequently warp the image to a frontal pose. In Figure 3 we show a few examples of images warped using this procedure.

4 Appearance representation

Once the images have been aligned and warped to a frontal pose, the next step is to compare the appearance of the faces to find similar facial expressions. Since all images are aligned to a template model we can identify the areas in the image that correspond to different face regions. In this paper we concentrate on the regions that correspond to eyes and mouth, however one can consider comparing other regions as well (e.g., position of eyebrows).

To compare appearance of facial regions we have chosen to use the Local Binary Pattern (LBP) histograms [12], which have previously proven effectiveness for face recognition [13] and facial expression recognition. These methods

however were applied on frontal faces captured with similar conditions (lighting, resolution etc.). In our case (especially in the case of unstructured photo collections) these conditions do not often hold. However our alignment step that warps the face to frontal position compensates for pose differences and allows us to effectively use LBP features for comparison of facial regions.

LBP operates by converting each pixel into a code which encodes the relative brightness patterns in a neighborhood around that pixel. In particular, each neighbor is assigned a 1 or 0 if it is brighter or darker than the center pixel. This pattern of 1's and 0's defines a binary code that is represented as an integer. Explicitly the LBP code is defined as:

$$LBP(c) = \sum_{p=0}^{|\mathcal{N}|-1} 2^p H(I_p - I_c), \tag{1}$$

where $H(x) = 1$ if $x > 0$ and 0 otherwise, I_c and I_p are the intensities of the center pixel and neighboring pixel correspondingly, and $\mathcal{N}$ is a set of neighbors of the center pixel c. The histogram of these codes defines the descriptor for each facial region. For example in case the neighborhood around a pixel is chosen to be 3x3 square, there are 8 neighbors, and so there are $2^8 = 256$ labels (or bins in the histogram). Intuitively each code can be considered as a micro pattern, that encodes local edges, homogenous areas and other primitives. The binarization quantization achieves robustness to small lighting changes and robustness to small motions is obtained by forming the histogram. Following [12] for each pixel we use a circular neighborhood around it and bilinearly interpolate values at non-integer pixel coordinates. We further use the extended version of the operator, called uniform code, that reduces the length of the feature vector. A code is called uniform if it contains at most two transitions between 0 to 1 or vice versa. In the computation of the LBP histogram each uniform code has its own label and all non-uniform codes get a single label.

5 Distance metric

Distance between two facial regions is defined by χ^2-distance between the corresponding LBP descriptors. χ^2 is defined as:

$$\chi^2(x, y) = 1/2 \sum_i (x_i - y_i)^2/(x_i + y_i), \tag{2}$$

where in our case x and y are two LBP descriptors. To compare the mouth region we divide it to 3x5 cells and sum up the distances of all cells, each eyes region is divided to 3x2 cells (the whole face is divided to 15x7 cells). Figure 4 shows the masks we use.

Given an input image i we compare it to all target's images j. Our appearance distance function between each frame i and image j is defined as

$$d_{appear}(i, j) \;=\; \alpha^m d^m(i, j) + \alpha^e d^e(i, j) \tag{3}$$

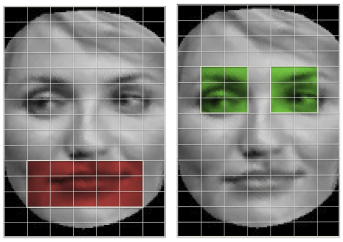


Fig. 4. The regions we use in our appearance distance. The image is aligned to a canonical pose and divided to 15x7 cells. The mouth region (marked in red) is divided to 3x5 cells and each eye region (marked in green) is divided to 3x2 cells.

where $d^{\{m,e\}}$ are the LBP histogram χ^2-distances restricted to the mouth and eyes regions, respectively, and $\alpha^{\{m,e\}}$ are the corresponding weights for these regions. For example, assigning $\alpha^m = 1$ and $\alpha^e = 0$ will result in only the mouth region being considered in the comparison. Prior to the combination of the mouth and eyes distances we normalize each of these by subtracting the minimum distance (over the target images) and dividing by the maximum distance.

Our complete distance function also includes difference in pose and a temporal continuity term. The difference in pose is measured separately for yaw Y, pitch P and roll R, and each of these is normalized using a robust logistic function. The pose term is:

$$d_{pose}(i,j) = L(|Y_i - Y_j|) + L(|P_i - P_j|) + L(|R_i - R_j|) \tag{4}$$

where the logistic function $L(d)$ is defined as

$$L(d) = \frac{1}{1 + e^{-\gamma(d-T)/\sigma}} \tag{5}$$

with $\gamma = \ln(99)$. It normalizes the distances d to the range $[0, 1]$, such that the value $d = T$ is mapped to 0.5 and the values $d = T \pm \sigma$ map to 0.99 and 0.01 respectively. The temporal continuity term computes the appearance distance between the previous input frame $i-1$ and all the target images j. The complete distance function is then:

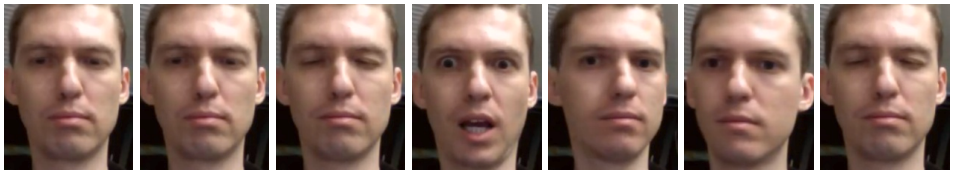
$$D(i,j) \;=\; d_{appear}(i,j) + \alpha^p d_{pose}(i,j) + \alpha^t d_{appear}(i-1,j) \tag{6}$$

where α^p,α^t are the weights of the pose and continuity terms. The best match per input frame is the target image that minimizes $D(i,j)$.

6 Results

In this section we give details on our experimental setup. The performance of the method is much better conveyed through videos; for the video captures and

(a) Full measure



(b) Without mouth similarity

(c) Without eyes similarity

Fig. 5. Puppeteering evaluation. We recorded a video of person A (70 frames) and a video of person B of similar length. (a) 7 frames from person A video (first row); The corresponding frames of person B using the combined measure - mouth+eyes+pose (second row); (b) The corresponding frames *without mouth* measure - only expressions with high correlation between the eyes and mouth (like surprise) have similar mouth expression (third row). (c) Person A and the corresponding matches of B *without eyes* measure - the eyes are flickering across consecutive output frames.

more results please see the paper's website: http://grail.cs.washington.edu/malkovich/. We have experimented with controlled experiments as well as videos

of celebrities downloaded from the Internet[1]. We also used the unstructured collection of photos of George W. Bush from the LFW database [14] as a target dataset. We begin by describing the implementation details of our system and then describe the exprimental results.

6.1 Implementation details

We use the following parameters for all experiments: $\alpha^m = \alpha^e = 1$, $T = 5$, $\sigma = 2$, $\alpha^p = \alpha^y = \alpha^r = 0.2$, $\alpha^t = 0.2$. Before we apply our distance function we ignore from consideration target images that differ in pose from the user image by more than 5^o (for yaw, pitch and roll). The LBP histogram is calculated per image cell using Gaussian weighting as a function of pixel's distance from the center of the cell. The sigma we used is the width of the cell with a margin in the size of half of the cell.

The system runs at 7fps on a 2.26GHz Intel Core 2 Duo Macbook Pro. The images or video used to create the target dataset are processed using the same pipeline as the input video of the user, i.e., tracking the face (or detecting in case of unstructured photo collection), estimating the pose in each frame and calculating the feature vectors. When constructing the target dataset from a video, we sample every 3rd frame of the video. Processing a video of 1000 frames takes approximately 2.5 minutes.

6.2 Controlled experiments

To evaluate performance, we captured videos of two people with different facial characteristics making similar facial expressions and used one person's video to drive the other video. We also evaluated the effect of comparing different regions of the face (eyes only or mouth only) on overall performance. Figure 5 shows the results of this experiment. We can see that the match is remarkably good when both eyes and mouth are used, despite the different facial appearance of these two users (note that one is Asian and has a mustache while the other has neither of these characteristics). When the mouth is omitted in the metric, the pose and eyes are matched, but the expression remains relatively constant, except for the example in column 4, where eyes of the "surprise" expression are well-correlated with the mouth. Similarly, when the eyes are left out of the distance metric, the output sequence exhibits random blinks.

6.3 Videos of celebrities

We downloaded and processed videos and still photos for several celebrities from the Internet. Figure 6 shows an example of puppeteering George Clooney; the user (top row) makes facial expressions and the best matching frame from George

[1] Cameron Diaz–http://www.youtube.com/watch?v=fWHgZz809Pw
George Clooney–http://www.youtube.com/watch?v=iZyw5-Sm0Zk
John Malkovich–http://www.mefeedia.com/watch/23930904

Fig. 6. Puppeteering George Clooney. A few frames of the user captured by a webcam, followed by the corresponding retrieved faces of George Clooney (the target database consists of 1197 video frames).

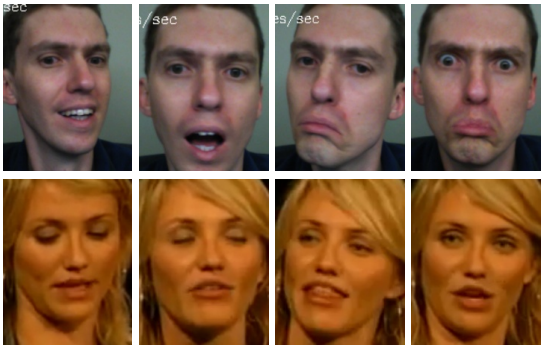
Clooney's video is shown at the bottom. Note how both, the eyes of the user and the eyes of George Clooney close in the 3rd example from the left, and how the mouth changes are quite consistent. Figure 7 shows results of puppeteering Cameron Diaz, (a) shows a sample of the good matches and in (b) we show some failure cases. Most of the failure cases seem to be due to the combination of an expression and pose that do not exist in the video/collection of photos of the target face. Similarly, we show an example where a user is able to drive an unstructured photo collection of George W. Bush obtained from the Internet (Figure 8). We also show an example of driving a video using another video in Figure 9. More examples are shown in Figure 1. In our experiments, we observed that when the user and the target face are of the same gender (woman to woman and man to man) the output sequence is smoother and better captures the expressions, due to similar facial features. However, we observed that the method also works quite well with a man driving a woman and vice versa (as shown in these figures).

7 Conclusions

We presented a real-time puppetry system in which the user can make a celebrity or other person mimic their own facial gestures. As with traditional puppetry, part of the fun is learning how to master the controls. In particular, the user often learns to best drive the celebrity (rather than the other way around); to make John Malkovich smile, the user may have to smile in a similar style to the celebrity.

Unlike most prior work in this area which maps an image to a model, our formulation is photo to photo, using metrics that seek to match facial pose and eye/mouth similarity. The key advantage of this approach is its generality—it operates fully automatically and works on just about any video or photo collection of a person.

(a) A sample of good matches



(b) Some failure cases

Fig. 7. Puppeteering Cameron Diaz. (a) A few frames of the user captured by a webcam, followed by the corresponding retrieved faces of Cameron Diaz (the database is a video of 1240 frames). (b) Some failure cases - most failures are due to a combination of an expression with pose of the user that do not exist in the target database. In this example the proportion of good/bad matches was around 0.7/0.3.

Beyond our puppetry application, this is also a general solution for face image retrieval, i.e., one can search for photos by acting out a particular expression and pose. In addition this allows to use unlabeled datasets and to retrieve facial expressions that are difficult to define with keywords.

There are several aspects of performance that could be improved. While LBP provides some robustness to lighting changes, shadows and other strong effects sometimes bias the match to similar lighting instead of similar expression. Better tracking and modeling of head shape could also increase the operating range, particularly with near-profile views. Finally, we use a first order model for temporal coherence; a more sophisticated model could result in temporaly smoother output.

Fig. 8. Puppeteering an unstructured dataset of George W. Bush. A few frames of the user captured by a webcam, followed by the corresponding retrieved faces of George W. Bush (the target database is a collection of 870 photographs of George W. Bush).

Fig. 9. Puppeteering John Malkovich with a video of Cameron Diaz.

Acknowledgments. The authors gratefully acknowledge Jason Saragih for providing the face tracking software [8]. This work was supported in part by Adobe and the University of Washington Animation Research Labs.

References

1. Pighin, F., Hecker, J., Lischinski, D., Szeliski, R., Salesin, D.H.: Synthesizing realistic facial expressions from photographs. In: SIGGRAPH. (1998) 75–84
2. Zhang, Z., Liu, Z., Adler, D., Cohen, M.F., Hanson, E., Shan, Y.: Robust and rapid generation of animated faces from video images: A model-based modeling approach. Int. J. Comput. Vision **58** (2004) 93–119
3. Vlasic, D., Brand, M., Pfister, H., Popović, J.: Face transfer with multilinear models. ACM Trans. Graph. **24** (2005) 426–433
4. Weise, T., Li, H., Gool, L.V., , Pauly, M.: Face/off: Live facial puppetry. In: Symposium on Computer Animation. (2009)

5. Kumar, N., Belhumeur, P., Nayar, S.: Facetracer: A search engine for large collections of images with faces. In: ECCV. (2008) 340–353
6. Bitouk, D., Kumar, N., Dhillon, S., Belhumeur, P., Nayar, S.K.: Face swapping: automatically replacing faces in photographs. In: SIGGRAPH. (2008) 1–8
7. Goldman, D.B., Gonterman, C., Curless, B., Salesin, D., Seitz, S.M.: Video object annotation, navigation, and composition. In: UIST. (2008) 3–12
8. Saragih, J.M., Lucey, S., Cohn, J.: Face alignment through subspace constrained mean-shifts. ICCV (2009)
9. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: CVPR. (2001) 511–518
10. Everingham, M., Sivic, J., Zisserman, A.: "Hello! My name is... Buffy" – automatic naming of characters in TV video. In: BMVC. (2006)
11. Zhang, L., Snavely, N., Curless, B., Seitz, S.M.: Spacetime faces: high resolution capture for modeling and animation. In: SIGGRAPH. (2004) 548–558
12. Ojala, T., Pietikinen, M., Menp, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. In: IEEE Trans. PAMI. Volume 24. (2002) 971–987
13. Ahonen, T., Hadid, A., Pietikinen, M.: Face description with local binary patterns: Application to face recognition. In: IEEE Trans. PAMI. Volume 28. (2006) 2037–2041
14. Huang, G.B., Ramesh, M., Berg, T., Learned-Miller, E.: Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst (2007)