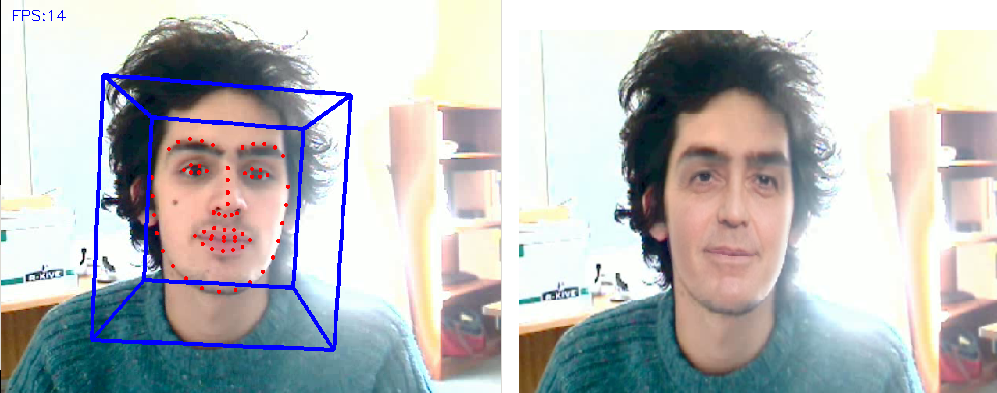
|  |
| --- |
| Puppets Manual (Alpha) |
| 2013 |
| Leonardo Impett |



Contents

[Introduction and Background 1](#_Toc368645053)

[Windows in Puppets 4](#_Toc368645054)

[Window I: Main control panel window 5](#_Toc368645055)

[Window II – Source Face 7](#_Toc368645056)

[Window III – Puppets 7](#_Toc368645057)

[Software and Code 8](#_Toc368645058)

[Key Changes from the original software, CLM-Z with Gavam 8](#_Toc368645059)

[How Puppets works 9](#_Toc368645060)

[Warping in OpenCV and OpenGL 10](#_Toc368645061)

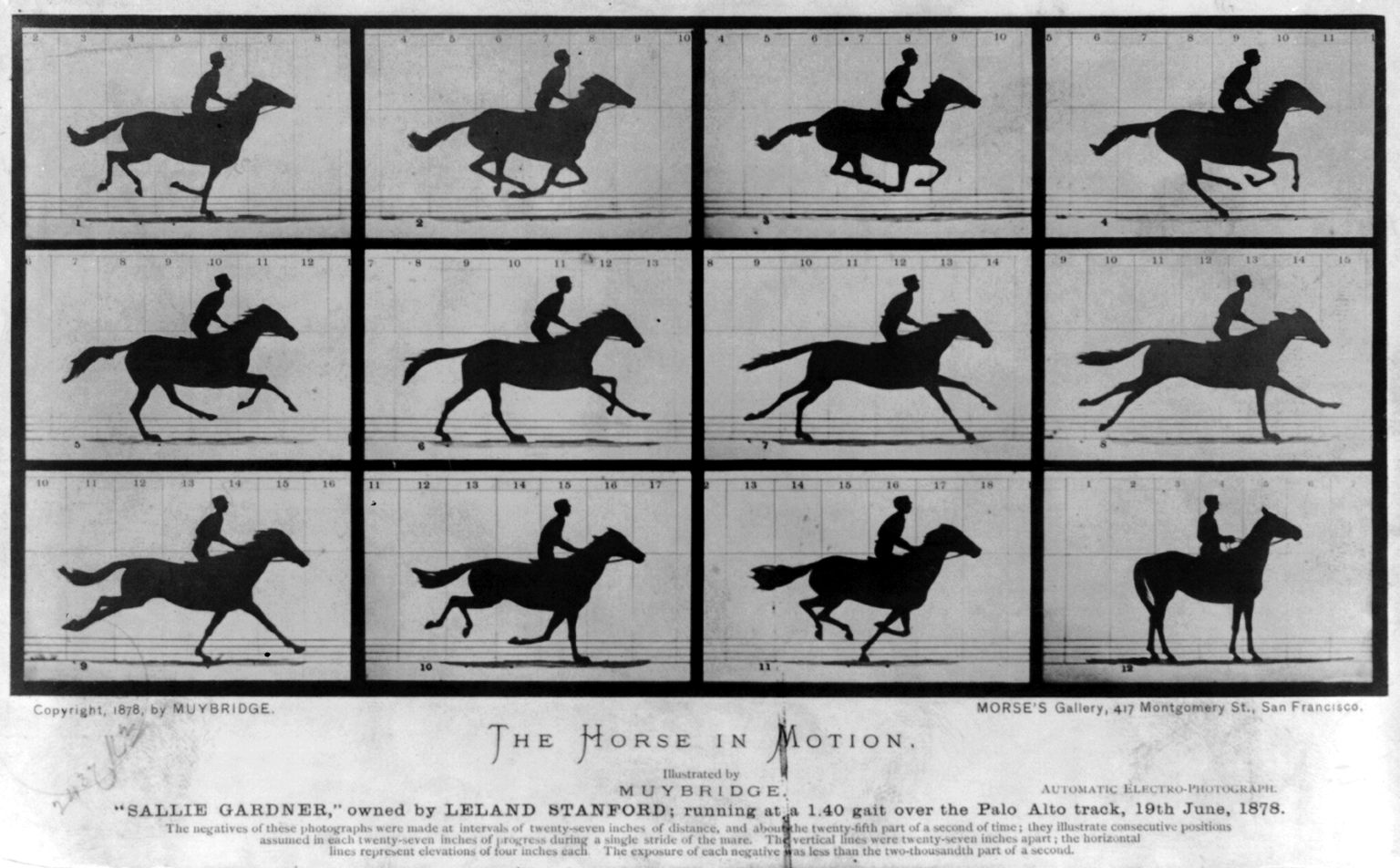
[Colour, brightness and contrast compensation for lighting and skin colour 11](#_Toc368645062)

[Failure Cases 14](#_Toc368645063)

[Occlusion 14](#_Toc368645064)

# Introduction and Background

Perhaps the first example of modern vision-recognition technology was Eadweard Muybridge’s horse, a photographic experiment in June 1878. He produced a set of rapid photographs (shown as a film on a zoopraxiscope) of a galloping horse, funded by Leland Stanford, an industrialist interested in horse gait analysis. It proved that all four legs of a course leave the ground simultaneously during gallop; with a great effect on late-19th century painting.



*Fig.1, above: Muybridge’s horse photographs were of enormous influence*

One of the first prototypes of true capture-and-synthesis technology was Max Fleischer’s rotoscope, a technique still used today (in digital form). In this technique, each frame of a section of film is projected onto an animation table, where it is then traced by hand – allowing for much more realistic animation. This technique was used to great success in Disney’s Snow White and the Seven Dwarves (1937).



*Fig. 2, above: the rotoscope in action*

“Scanimate’s… ‘real-time’ capability was unheard of, and was even incorporated live in the 1978 Grammy Awards presentation.” [SIEG]

The first online facial puppetry system was demonstrated by Brad deGraf and Michael Wahrman in 1988 [ROBERTSON]. Called Mike the Talking Head, it used mechanical sensing to animate in real-time a digital 3D-avatar. This was a great success, and the system was used in the Hollywood film Robocob 2 to represent the digital face of Cain.



*Fig. 3, above, clockwise from top-left: Mike the Talking Head’s mechanical rig, Mike’s rendering, and the modified version used in the film Robocop 2*

Shortly after Mike the Talking Head came Lance William’s “performance-driven facial animation”, which tracked feature points on a face and recreated a 3D model that replicated these expressions.

Marker-based systems for facial and bodily motion capture have predominated since the introduction of digital CGI in film. However, more recently non-marker-based systems have been increasingly popular in the film industry, with films such as Benjamin Button using ultraviolet optical methods. Though rendering was offline, the ability to work without marker-based changed the way in which actors could perform, and was widely praised in the industry.

Markerless systems such as Image Metrics have been used for live performances (though the rendering is offline and recordings were used) in Jeff Wayne’s stage production of The War of the Worlds, as well as wide use in the film and video game industries, in films such as Harry Potter.

The online technique also has applications for use in theatre and interactive artworks. Arturo Castro and Kyle McDonald’s non-photorealistic 2012 work “Faces” is an example of this – an electronic mirror (a display with webcam) is installed which digitally replaces the faces of unsuspecting visitors, experimenting with Masahiro Mori’s concept of the “uncanny valley”.



*Fig. 4, above: a sample frame from Castro-McDonald’s pioneering work in face replacement*

Puppets is a purely optical facial puppetry system, with a Constrained Local Model (CLM) face-tracker which aims to give online (real-time) avatar rendering with realistic results on standard hardware (ie, with a home computer and webcam). Recalling the advantages of the Scanimate, where performances could be visualised in real-time and tweaked to suit, an online face-rendering system could be used for on-set visualisation or “previsualisation”. Then once a satisfactory shot has been taken, the data can be rendered offline using traditional, slower, more sophisticated methods. Unlike state-of-the-art commercial markerless systems such as Faceware, it does not need the user to wear a camera and light on a mount, as it is robust to changes in head pose and lighting.

# Windows in Puppets

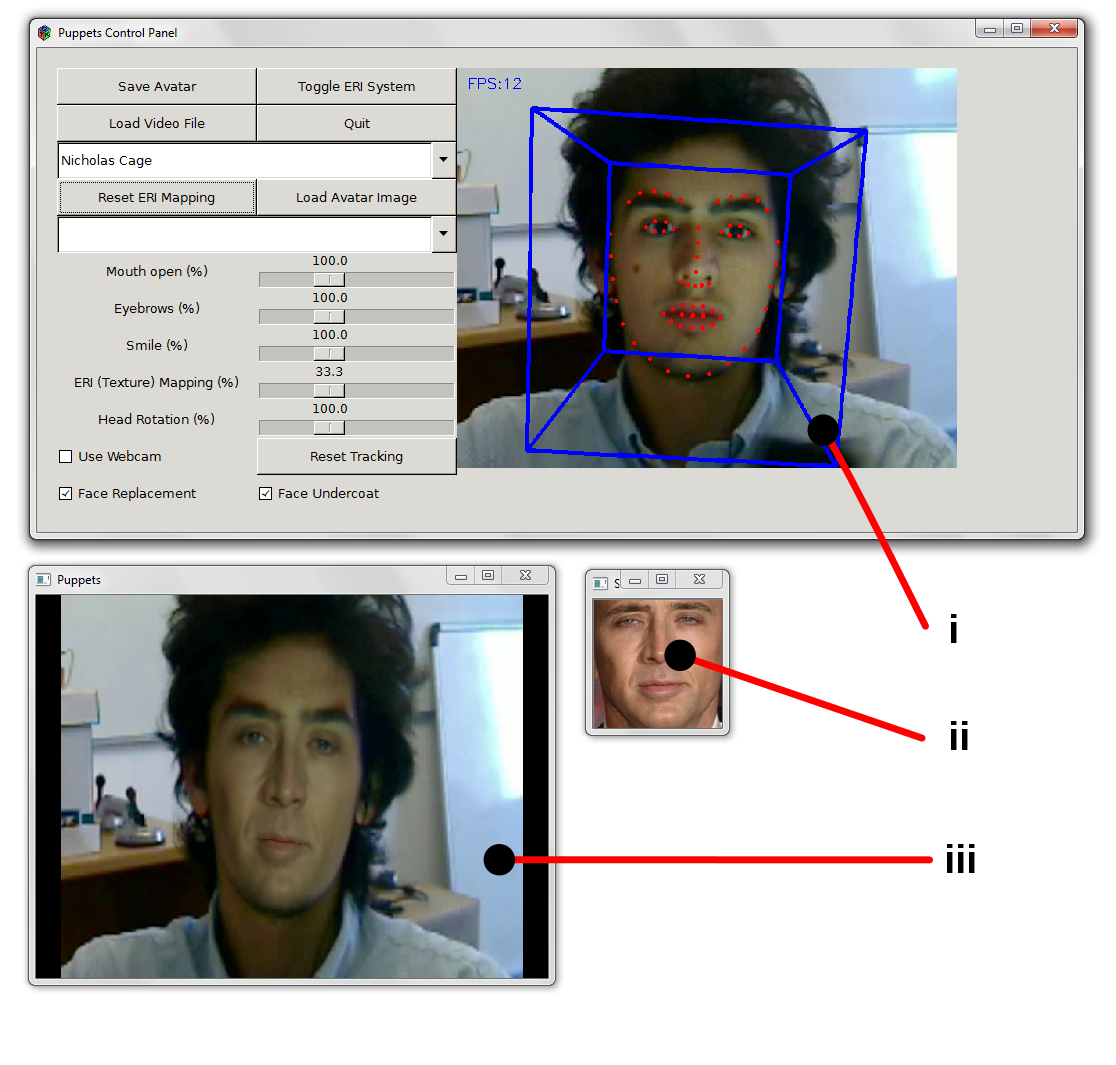
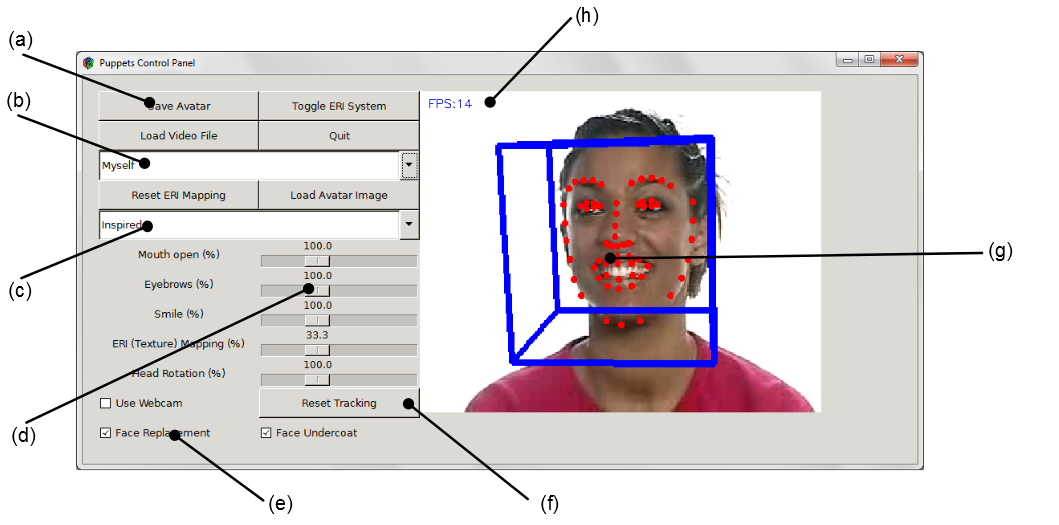


Fig. 5, above: the main windows of the Puppets GUI

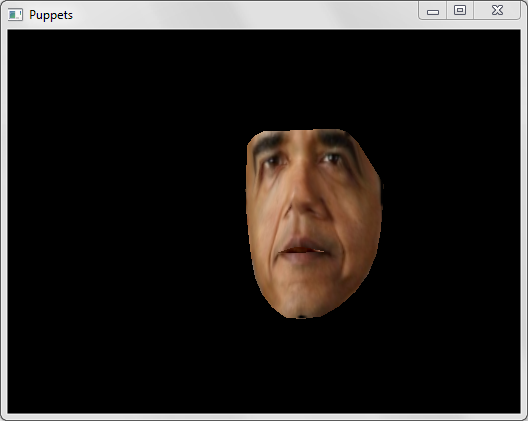
|  |  |  |
| --- | --- | --- |
| **Window** | **Title** | **Function** |
| i | Puppets Control Panel | Allows the user to change parameters, load and save source files, see and reset face tracking and quit the program. |
| ii | Source Face | The face image (or *‘puppet’*) to be transplanted onto the source video. |
| Iii | Puppets | The output video, with the source face overlaid on the video. |

## Window I: Main control panel window



*Fig.6, above: the control panel*

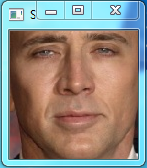
|  |  |  |  |
| --- | --- | --- | --- |
| **Label** | **Feature** | **Function** | |
| (a) | Main panel buttons | *Save Avatar* | Save the current face – from the input video source in *(g)* – to disk. This can then be accessed from the avatar face selector, under “myself”. |
| *Toggle ERI System* | Switches on or off the Expression Ratio Imaging (ERI) expression transfer system. When disabled, the overlaid face will make no texture corrections for expression, shadow or lighting. ERI is partly responsible for lighting adjustment, and is not recommended for use when “Face Replacement” is selected. |
| *Load Video File* | Opens a file selector window to load the input video or image file. |
| *Quit* | Exit the program immediately. |
| *Reset ERI Mapping* | Capture the current expression as the reference “neutral” expression for the ERI system. (See explanation of ERI system). |
| *Load Avatar Image* | Loads a source video or image file and immediately saves their face to disk, as per *Save Avatar* above. |
| (b) | Avatar face selector | Select which face will be applied to the source image as an avatar. | |
| (c) | Source video selector | Select from a stock list of pre-recorded expression video sources. For other video sources use ‘*Load Video File’* above. | |
| (d) | Expression strength and shape sliders | Mouth open, Eyebrows, Smile | Used to change the shape (not texture) of the puppet face. Not recommended for use when “Face Replacement” is selected. |
| ERI (Texture) Mapping | Changes the amplitude of expression transfer to the texture. If it is set to 0%, ERI is effectively disabled |
| Head Rotation | Changes the amount of head rotation in the pose. If set to 0%, the head will still translate, but constantly facing forwards. Not recommended for use when “Face Replacement” is selected. |
| (e) | Control checkboxes | Use Webcam | When checked, the software will try to read source video from the computer’s webcam. When unchecked, it will go back to reading the last valid video source. |
| Face Replacement | When checked (default), the avatar face is colour-compensated and warped-blended onto the source video. When unchecked, the avatar face is instead presented against a black background (allowing shape changes to be made more naturally), with the oral cavity only shown from the source image. See figure 4, below. |
| Face Undercoat | When checked (default), the avatar is not blended directly over the source face, but rather over a blurred image of the source face. Not only does this help preserve skin colour, it also preserves local lighting variation to some extent (eg, strong light from one side). Although it is never directly visible in the software, here is an example visualisation of the blurred source face: |
| (f) | Reset Tracking button | Manually re-initialises the face tracker. This should be used when tracking fails, or it tracks the wrong object | |
| (g) | Input video frame | A visualisation of the input frame, along with the 66 tracked feature points on the face and a 3D bounding cube showing position and orientation. A blue box indicates successful tracking, whereas a red box indicates a tracking failure (when the model will re-initialise locally) | |
| (h) | Frames per second indicator | Shows the current moving-average number of frames processed per second | |

*Fig. 7, above. Left: Face Replacement checked, right: Face Replacement unchecked.*

## Window II – Source Face

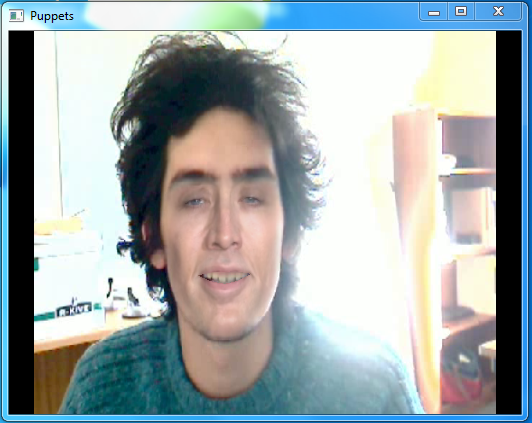
This is where the cropped source face is displayed, selected either from the list of presets or loaded using “Load Avatar Image”.



*Fig. 7, above: the source face window*

## Window III – Puppets

The main video output is displayed in this window, with the face relit and overlaid on the background image.



*Fig. 8, above: the Puppets main output window*

# Software and Code

## Key Changes from the original software, CLM-Z with Gavam

1. New project *SimplePuppets* (from which to run)
2. New project Puppets, including header & source Avatar.cc
3. Within CLM: some slight changes in PAW
4. Within the model: slight modification to ./lib/local/CLM/model/face\_check\_pie.txt
5. New folder ./*images*, which contains the important face images and shape points (stored in XML/YML files)
6. New XML model (specifying the face-model-specific triangle indices that represent the mouth and eyes) under lib/local/Puppets/model/mouthEyesShape.yml
7. **SimplePuppets**

This is the launch part and main loop of the program, and contains the important GTK calls, as well as some high-level face-tracking code from SimpleCLM. It also contains the calls to facial reconstruction in Puppets.

1. **Puppets**

Within here is Avatar.cc, which calls or contains all the important facial reconstruction code. It creates and draws into an OpenGL window (separate from the GTK), and handles the file reading, saving and processing of different facial avatars, image warping, expression transfer, posture detection and so on. For the image warping, some calls are made to PAW within the project CLM.

1. **PAW changes**

WarpToNeutral, unWarpRegion, and so on are added simple functions; they take an image with a series of (x,y) points and warp it to a neutral image, which is then used as a texture by OpenGL (which warps it in 3D to the new series of points).

1. **Model Modification**

The triangle map for the front view of face\_check\_pie is filled in in points around the face (instead of being set to ‘-1’). This is for the warping so that an expression ratio image can be applied to a ‘neutral’ face shape; it’s not used for side-views, where no ERI is applied.

There is another model folder, in the /lib/local/Puppets/model/ directory; an xml (.yml 1.0) file called moutEyesShape.yml. It contains 2 OpenCV matrices, *eyestri* and *mouthtri*, each of which list the appropriate triangles to fill in for the eyes and mouth respectively. This is the only file to change if using a different number of points in the face-tracker.

1. **Images folder**

This folder contains two types of file: images and corresponding XML (actually .yml) files describing the orientation and shape points on that image, as well as the file locations of the other corresponding views (of the same model). The shape points are represented as a 132x1 floating-point vector, representing the 66 points first in x, then y coordinates.

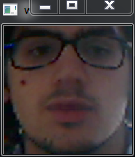
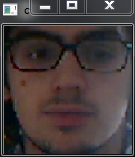
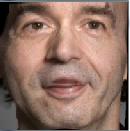
## How Puppets works

1. We get the face image and shape from the source video/webcam, and from the (static) avatar we want to imitate. **(a)**
2. We warp the source face to a neutral expression **(b)**, and convert the neutral face from BGR 8-bit to YUV floating-point. The luminance is normalised and the histograms are equalised (so that the cumulative histograms are linear) to give better robustness against ambient lighting conditions
3. After 10 frames, or when the ‘Reset ERI’ button is pressed, we capture a ‘reference’ (expression-neutral) face. Because the face is warped to a neutral shape, we can capture changes in texture between this and future faces.
4. Blur both the reference face and the expression (to remove very fine noise, especially to do with misalignment of features despite the neutral-warping). Downscale, blur and upscale (for time-efficiency’s sake)
5. Create a new matrix, the luminance Expression Ratio Image (ERI) matrix, of the same size of the faces: each pixel within is the floating-point division of the corresponding pixels in the current source face by the pixel in the ‘neutral’ reference face. I.e, ERI = newFace / referenceFace. This is only done in the forward-facing posture – side-facing images are exempt from these steps (even when warped to frontal-neutral, they produce some very strong artefacts. We thus avoid having to have several reference ‘expressionless’ face images)
6. Create a mask of the size of the ERI which has non-zero elements only where the ERI value is less than 1.0. This means we’re only interested in features which get darker (shadows, wrinkles and frown-lines), as ones which get lighter cause many more artefacts (and are often created by mismatching of the mouth area, causing it to draw ‘teeth’ on the avatar’s lips)
7. The amplitude of this ERI is then attenuated by a user-controllable ‘ERI strength’ factor (initially 0.33, which gave good results by visual inspection). As the ‘neutral’ value of the ERI is 1.0, to scale by this factor in practice we subtract 1.0 from each pixel, multiply by the ERI strength factor, then add 1.0 again to the matrix. **(c)**
8. This ERI matrix is then used as a pixel-by-pixel multiplication factor for the neutrally-warped target avatar face (effectively making wrinkle and shadow areas darker), before it is sent for colour compensation and OpenGL rendering. **(d)**

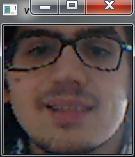
*Fig. 9, below, first row, from left to right:* ***(a)*** *a neutral source expression,* ***(b)*** *the same expression warped to a neutral shape,* ***(c)*** *a fairly featureless ERI, and* ***(d)*** *the resulting avatar face.*

*Second row:* ***(a)*** *a smiling source expression with creases in the skin,* ***(b)*** *warped to a neutral shape,* ***(c)*** *the ERI image showing these creases clearly, and* ***(d)*** *the resulting avatar face with these creases applied.*

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



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



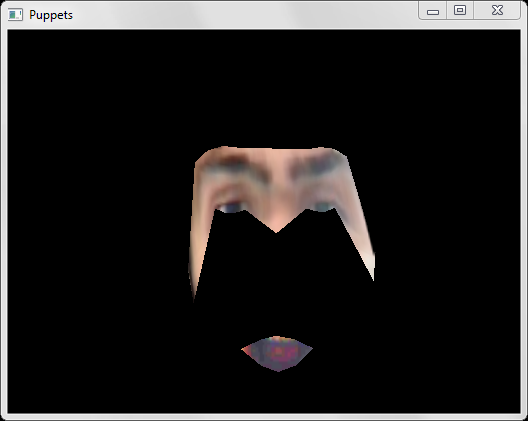
## Warping in OpenCV and OpenGL

To warp the source face image to a neutral expression, so that ERI can be applied, a CPU-based OpenCV warp is performed. This is done in CLM/Paw.cc, in three steps:

1. For each polygon (triangle) on the face, lookup the three vertices.
2. Find the alpha and beta values of the inverse lookup (from the local face model) to compute the affine warp matrix. This will differ according to the posture of the head (whether it looks forwards or to the side)
3. Create a new remapping lookup table that maps the positions of the new image to a lookup (interpolated) value in the old image

This method involves a certain amount of precomputation (the alpha and beta values for each polygon in the desired shape), and is therefore quick for warping to neutral, where the expression can be evaluated with ERI. To re-warp the image to the appropriate destination shape, precomputation is not possible and a faster solution was necessary. The GPU therefore warps the image, by converting the face to a texture and rendering each triangle as a 2D textured polygon, using OpenGL (supported by most graphics cards), along with an alpha channel which enables smooth blending between the original background image and the new face. Thus lighting changes and shadows need to be compensated for in the facial texture before GPU rendering.



*Fig. 10, above. First row: a neutral face, warped to the front and to the side. Note the significant aliasing artefacts in the remapping (this is solved by applying a small Gaussian filter to the lookup matrix). Second row: the avatar face is rendered triangle-by-triangle by the GPU. Although the screen buffer is not refreshed until it has finished all the triangles (a double buffer is used), this is a visualisation of what the buffer contains halfway through a face rendering.*

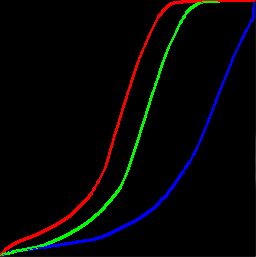
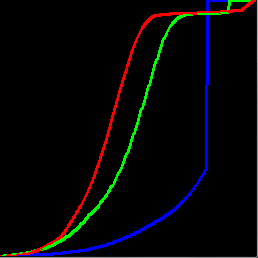
## Colour, brightness and contrast compensation for lighting and skin colour

To compensate for different skin colours and textures, and differing and dynamic lighting conditions, some compensation must be used. Although we don’t notice it ourselves, as our eyes compensate locally for colour shifts due to lighting, even the same person’s face can drastically change colour in different lighting conditions.

*Fig. 11, above: two similar frames from the same video sequence, with the same avatar applied. On the left is the image without colour compensation, whereas on the right is the image with colour compensation. Note the three scales of lighting: universal lighting with colour compensation, local lighting with a blurred undercoat (the right half of the face is lighter than the left half), and detailed shadows with the ERI system.*

If we simply scale or offset the RGB values of each pixel to make the mean colour component the same in 8-bit colour space, we risk saturating a large portion of the pixels in one – or more – components. This is particularly an issue when applying a high-contrast face to a very light or very dark image. An example histogram comparison is shown below:

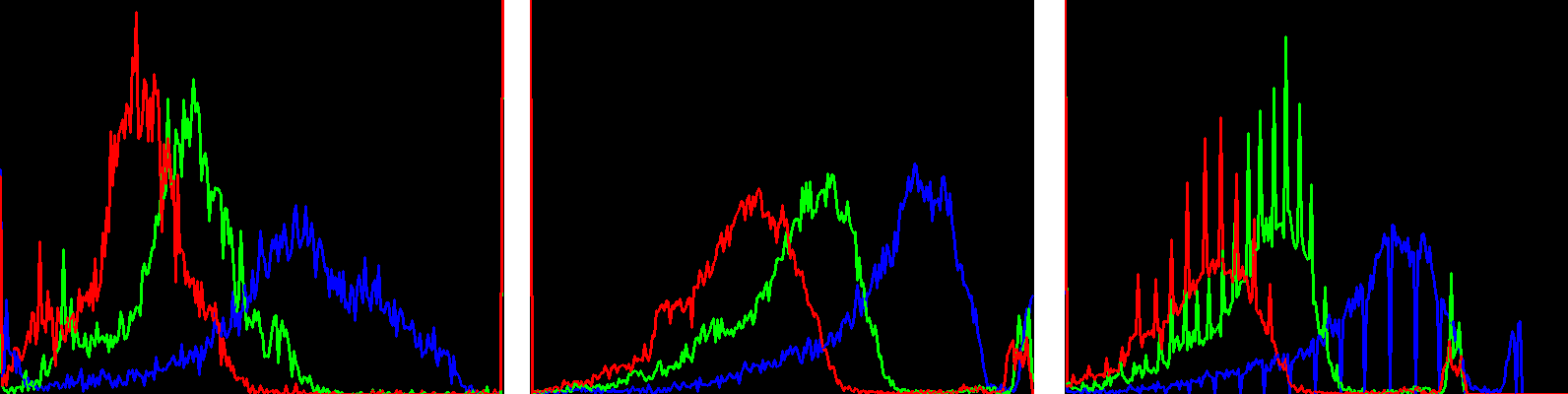
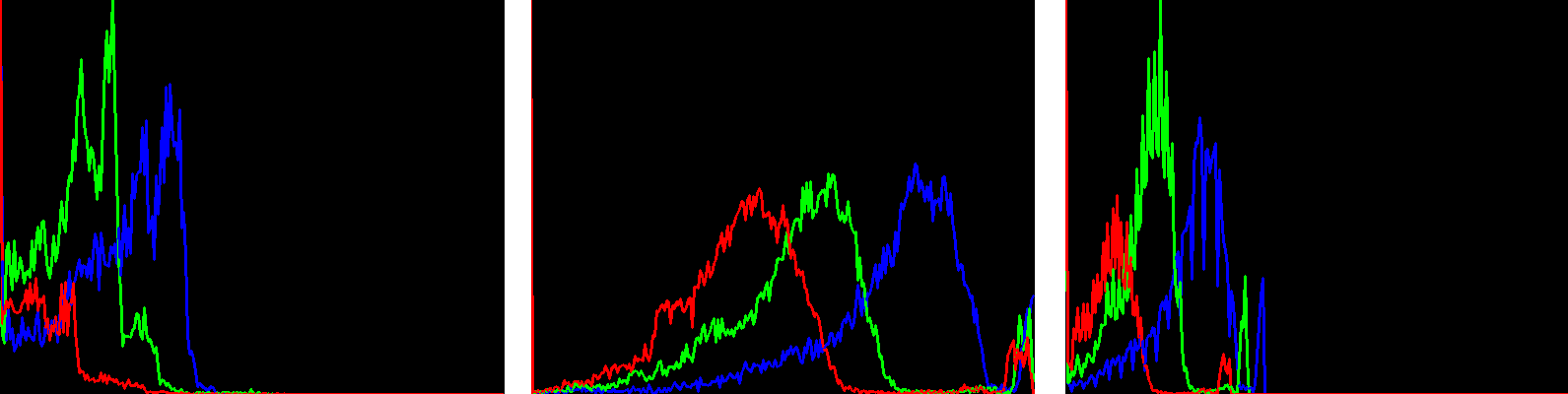
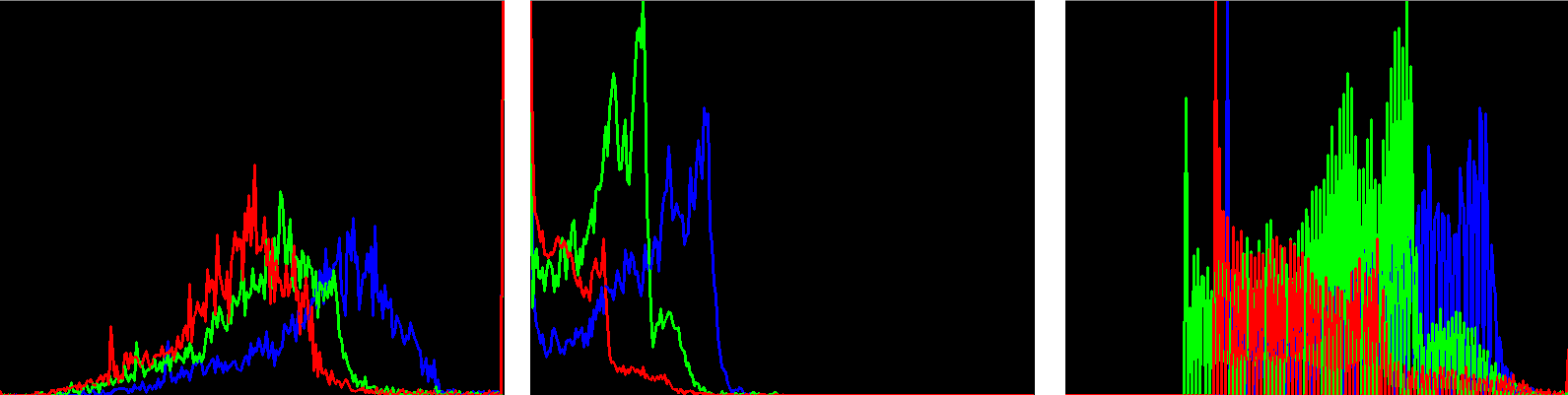
  

*Fig. 12, above: cumulative RGB histograms for the source (original) face, uncompensated avatar and mean-compensated avatar. Note the blue saturation in the final (mean-compensated) histogram*

In general, colour histograms for faces are vaguely bell-shaped; we therefore approximate them as Gaussian, and describe their shape by only two variables: their mean and their standard deviation. We can calculate these two parameters quickly within OpenCV, and scale and offset the source avatar face to compensate. To avoid the effects of background colours in the original or avatar image, we calculate these parameters only within a ‘Mickey Mouse’ mask.



*Fig. 13, above: the ‘Mickey Mouse’ mask for colour compensation. A blurred version is also used as the alpha channel for OpenGL blending*

1. (a) (b) (c)

2. (a) (b) (c)

3. (a) (b) (c)

*Fig. 14, above, histograms for three cases of colour compensation. (a) Original face in source image, (b) original avatar image, (c) compensated avatar image. First row: a dark avatar on a light image, causing cut-off at the lower end of the colour distribution. Second row: a normally-lit avatar on a dark image, causing no artefacts. Third row: a normally-lit avatar on a normally-lit face, also being compensated correctly. The quantisation evident in the compensated avatar image is due to upscaling the low-resolution (8-bit) colour spectrum in the original avatar, and does not in itself constitute a failure case.*

## Failure Cases

### Occlusion

Although the face tracker itself makes some allowance for occlusion, the face replacement software does not correct for them. Although self-occlusion of the face (such as one cheek being obscured by the nose) is dealt with during rendering, occlusions of other objects represent a failure case; the most common being glasses. Hands or other objects placed in front of the face after the start of the video will occasionally appear to shine through due to the ERI mapping (see the hand in Fig. 15). To solve this, it is best to use Puppets in no-background mode in occlusion-dense scenarios, and in extreme cases, to reduce the ERI strength by using the slider.



*Fig. 15, above, clockwise from top-left: occlusion-free image, image with double occlusion (glasses and hand), image with small self-occlusion due to head pose, image with occlusion processed without background (note absence of re-lighting outside oral cavity)*

* We use floating-point matrices to avoid saturating at 0 or 255, then convert them back to 8-bit colour.
* We scale and shift, to keep mean and the standard deviation of each colour the same as in the target image.

Changing means

Colour spaces

Colour histograms

Conserving standard deviation

The floating-point problem

Transparency (alpha) mask

The blurred undercoat

Sieg, Dave. "Simulation in the analog days." *ACM SIGGRAPH Computer Graphics* 32.3 (1998): 58-59

Robertson, Barbara. "Mike the talking head." *Computer graphics world* 11.7 (1988): 57