Landmark Detection and 3D Face Reconstruction using Modern C++

Patrik Huber
Centre for Vision, Speech and Signal Processing
University of Surrey, UK

p.huber@surrey.ac.uk
Landmark detection and 3D face reconstruction using modern C++

• We aim to fill two (or three) gaps:
  • Open & available 3D face models and fitting algorithms
  • Easy-to-use, modern C++ code for above

• Bonus:
  • Reproducible research
3D morphable face models & fitting

- 3D face models have some desirable properties
  - Shape and pose are modelled separately (i.e. camera model, and shape model)
  - Depth information

- Usually harder to train and use than other (e.g. 2D) methods
  - 3D scans instead of images, dense 3D-3D registration, usually much more parameters during fitting (shape, pose, albedo, light)

- Not many models are available, even less freely, and even less with fitting code
  - Most well known: BFM from T. Vetter et al., *A 3D Face Model for Pose and Illumination Invariant Face Recognition*, AVSS 2009
The Surrey 3D Morphable Face Model

• PCA model of shape and colour (albedo)
• Built from 170 3D scans with diverse ethnicity
• 3 different resolution levels
• Metadata (texture coordinates, landmark definitions, ...)

3448 vertices
16759 vertices
29587 vertices
The Surrey 3D Morphable Face Model

- PCA model of shape and colour (albedo)
- Built from 170 3D scans with diverse ethnicity
- 3 different resolution levels
- Metadata (texture coordinates, landmark definitions, ...)

- The low-resolution shape-only model is available directly with the software on GitHub
- Higher resolutions & full model via University licencing
Along with the model there’s a lightweight modern C++ framework to use the model and perform basic shape fitting tasks

https://github.com/patrikhuber eos

(more after the next slide...)
Modern C++?

• There’s not many 3DMM fitting frameworks, even less in C++
  • menpo (Python)

• C++ is a very widely used language in computer vision
  • Speed, cross-platform compatibility, mobile devices, embedded, robots, ...

• C++ has a reputation of being low-level, hard to learn, read, use & maintain, pointers, memory leaks, segmentation faults, ...
  • A lot of the C++ code «out there» looks like that

• ➔ That’s not how it should be written nowadays
  • C++ can be as easy and safe as a language like Matlab, while not losing any of its advantages
How does that look like?

```
MorphableModel morphable_model = morphable_model::load(filename); // loaded using cereal
Mesh mesh = morphable_model.draw_sample(vector<float>{1.0f, 0.0f, -1.0f}, vector<float>());
mesh = morphable_model.draw_sample(/* shape_sigma=1.0f, colour_sigma=1.0f*/);
write_obj(mesh, "out.obj");
```

*All namespaces omitted. Complete example: https://github.com/patrikhuber/eos/blob/master/examples/fit-model.cpp*
Fitting the shape-3DMM

vector<Vec2f> image_points;
vector<Vec4f> model_points;

Mat affine_cam = estimate_affine_camera(image_points, model_points);

vector<float> shape_coeffs = fit_shape_to_landmarks_linear(morphable_model, affine_cam, image_points, vertex_indices);

Mesh mesh = morphable_model.draw_sample(shape_coeffs, vector<float>());
write_obj(mesh, "out.obj");

Mat texture = extract_texture(mesh, affine_cam, image);
supervised descent / cascaded regression

• A generic implementation of SDM
• Learn a series of regressors:

\[ \Delta \theta = A_n f(I, \theta) + b_n \]

• \( \theta \) are traditionally 2D landmark locations: \( \theta = [x_1, \ldots, x_n, y_1, \ldots, y_n] \)
  • E.g. X. Xiong & F. De la Torre, *Supervised Descent Method and Its Applications to Face Alignment*

• But it can be anything, for example 3D model parameters:
  • \( \theta = [R_x, R_y, R_z, t_x, t_y, t_z, \alpha_0, \alpha_1, \ldots] \)
  • Huber et al. 2015, Li et al. 2015
supervised descent / cascaded regression

- A generic implementation of SDM
- Learn a series of regressors:

\[ R_n: \delta \theta = A_n f(I, \theta) + b_n \]

- Projection \( f(\ldots) \) is generic too:
  - Can be HOG feature extraction
  - or involve full 3D model projection: \( \delta \theta = A_n f(I, M, \theta) + b_n \)
Fitting 3D Morphable Models using Local Features

The parameter vector $\theta = [r_x, r_y, r_z, t_x, t_y, t_z, \alpha_0, \alpha_1]$ is updated using the learned regressors.

Initialisation from default parameters

Input image

Model projection using the current parameter estimate $\theta$

Local feature extraction regions

ICIP 2015, P. Huber, Z. Feng, W. Christmas, J. Kittler, M. Rätsch
Landmark-detection in 3 lines of code

definition_model model = load_detection_model("model.bin");
Mat image = cv::imread("image.png");
vector<Landmark<Vec2f>> landmarks = model.detection(image, Rect(50, 50, 80, 80));

*All namespaces omitted for brevity.*
Surrey 3D Morphable Face Model and fitting library

C++11/14, Fully cross-platform (Windows/Linux/Mac/...)

Header-only

Apache licence

External dependencies: OpenCV core, (Eigen)

https://github.com/patrikhuber/eos

https://github.com/patrikhuber/superviseddescent
Demo
Concluding words

• C++ is an important language for computer vision
  • It’s important to learn, use and spread modern best practices
  • ...and have libraries that are easy to use and maintain

• Push research with 3D face models
  • via open & shared code & models

• superviseddescent – generic cascaded regression library

• Low-resolution 3DMM shape model available in the repo
  • Higher resolutions & full model via University licencing

• eos – 3DMM framework & fitting library
Team

• Zhenhua Feng (Uni Surrey)
• Guosheng Hu (previously Uni Surrey)
• Philipp Kopp (Reutlingen Uni)
• Rafael Tena (previously Uni Surrey)
• Pouria Mortazavian (previously Uni Surrey)
• Willem Koppen (Uni Surrey)
• Michael Grupp (Reutlingen Uni)
• Dr. Matthias Rätsch (Reutlingen Uni)
• Dr. William Christmas (Uni Surrey)
• Prof. Josef Kittler (Uni Surrey)
References

• Own & related publications:
  • **Fitting 3D Morphable Models using Local Features**, P. Huber, Z. Feng, W. Christmas, J. Kittler, M. Rätsch, *ICIP 2015*
  • **Supervised Descent Method and Its Applications to Face Alignment**, X. Xiong and F. De la Torre, *CVPR 2013*
Thank you!

Questions?

Suggestions, comments and contributions very welcome! (email me or open a GitHub issue)

p.huber@surrey.ac.uk