3D Face Tracking and Reconstruction using Modern C++

Patrik Huber

Centre for Vision, Speech and Signal Processing

University of Surrey, UK

patrikhuber@gmail.com | www.patrikhuber.ch

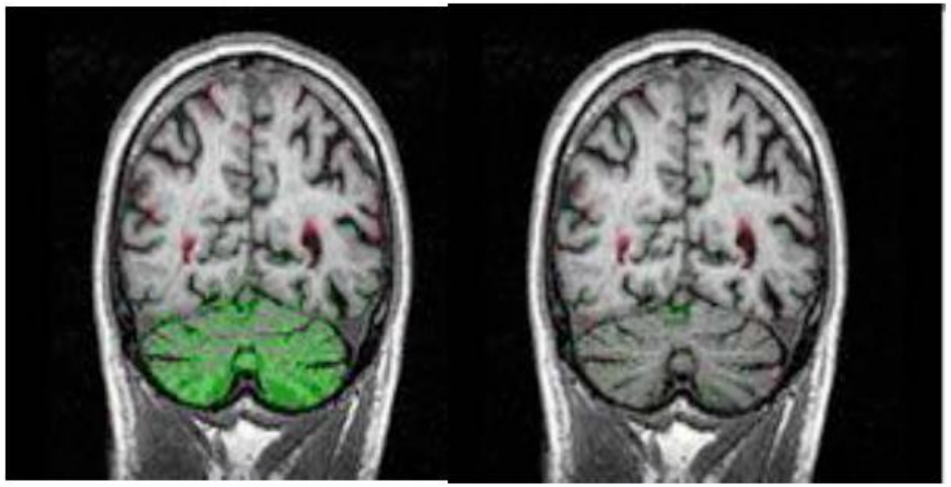
cppcon the c++ conference

UNIVERSITY OF SURREY

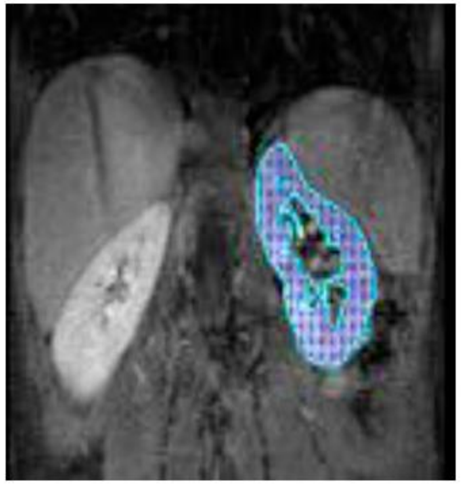
The Centre for Vision, Speech and Signal Processing

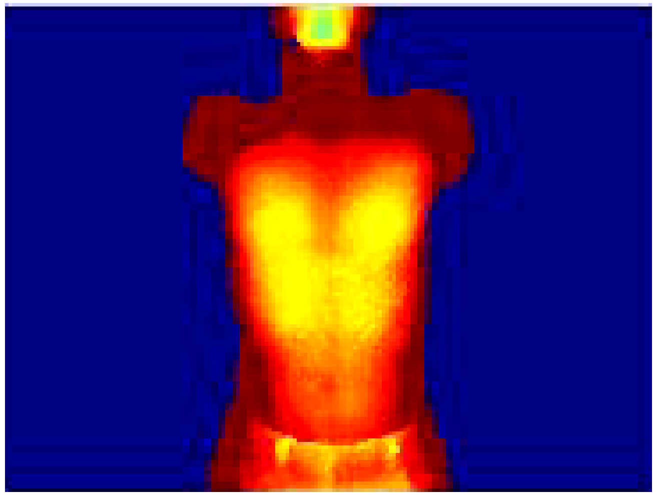
- University of Surrey, Guildford, UK
- One of the largest UK computer vision, audio-visual signal processing and multimedia communication groups
- Founded 1986
- 120 people (70 PhDs)
- Grant portfolio £12M

The Centre for Vision, Speech and Signal Processing

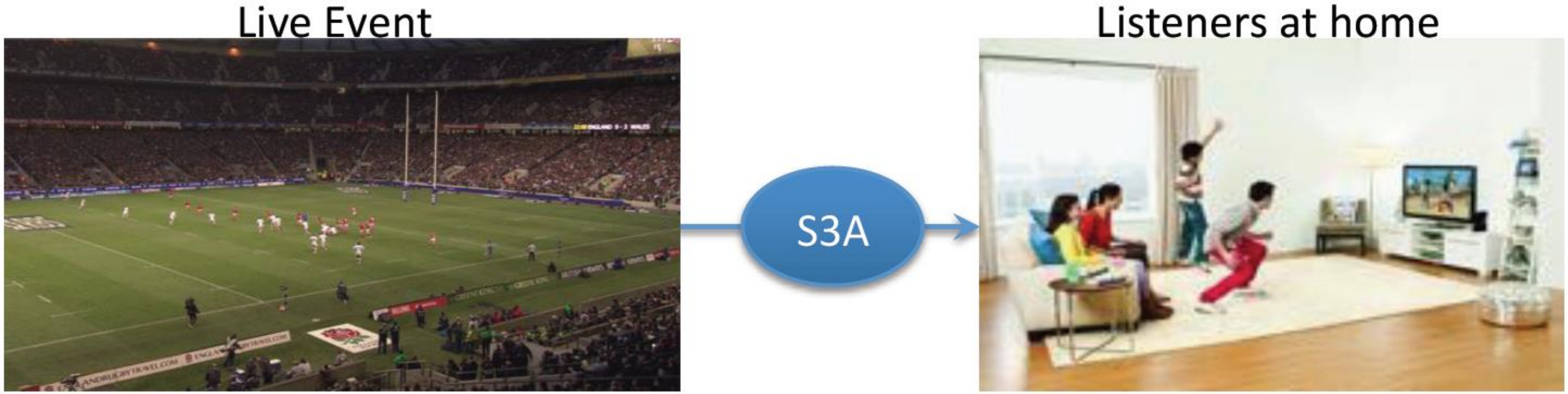


Abnormalities due to Alzheimer’s





The Centre for Vision, Speech and Signal Processing

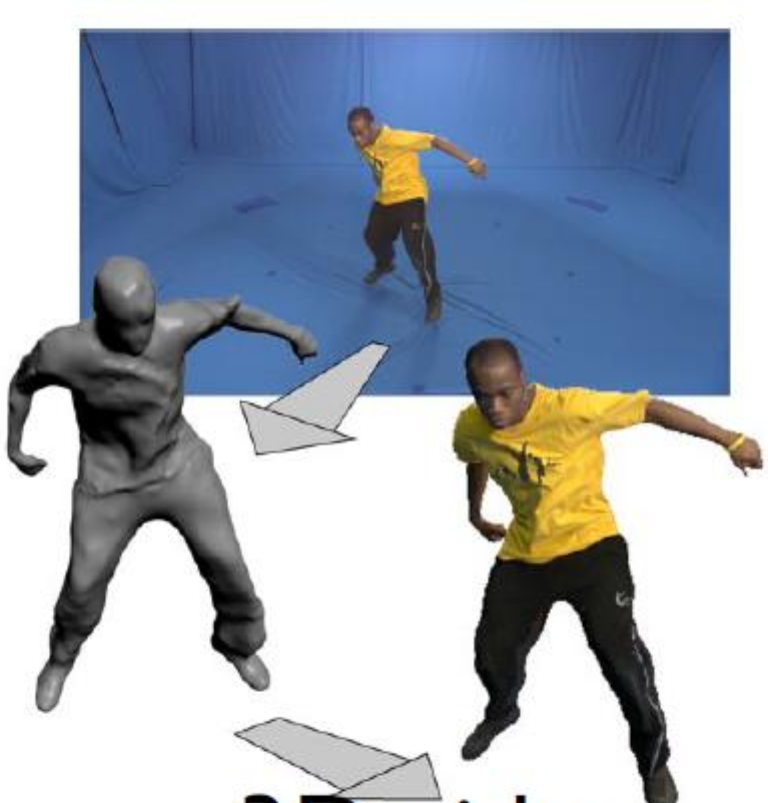


Live Event

S3A

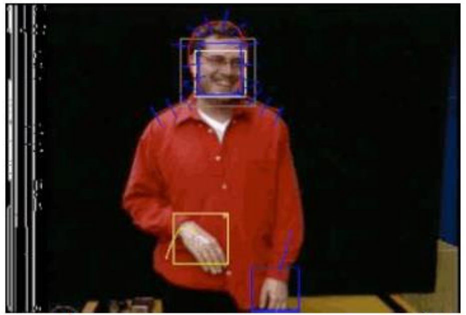
Listeners at home

The Centre for Vision, Speech and Signal Processing



3D video

http://www.bbc.co.uk/taster/projects/dancer



sign-language recognition

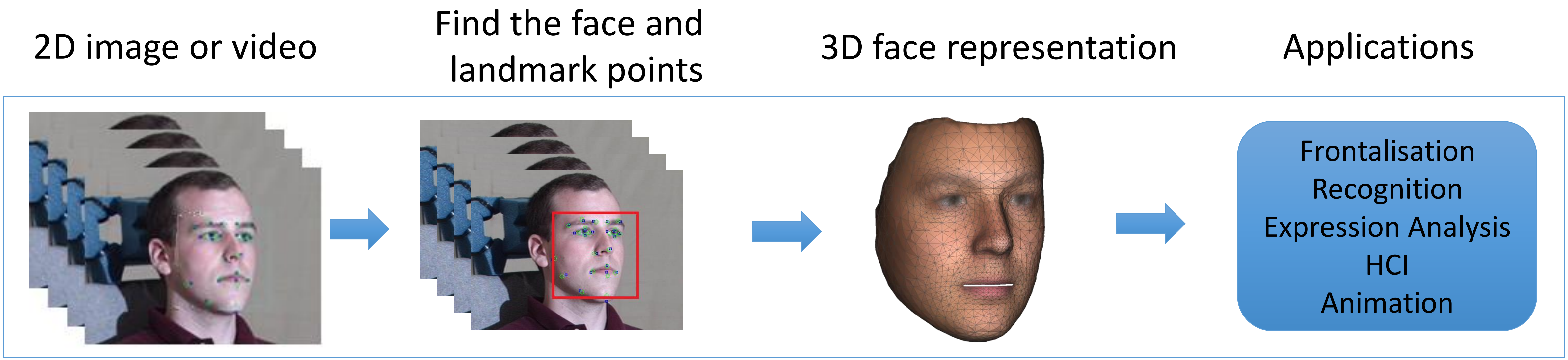
face tracking

people detection

A few words about myself

- PhD student in computer vision
- Working on faces, 3D face models from 2D images
- Very interested in modern C++
- Open source software:
  - infrequent bug reports, recently started to contribute to OpenCV
  - recently released quite a few bits of our research on GitHub

Face tracking and 3D face reconstruction



2D image or video

Find the face and landmark points

3D face representation

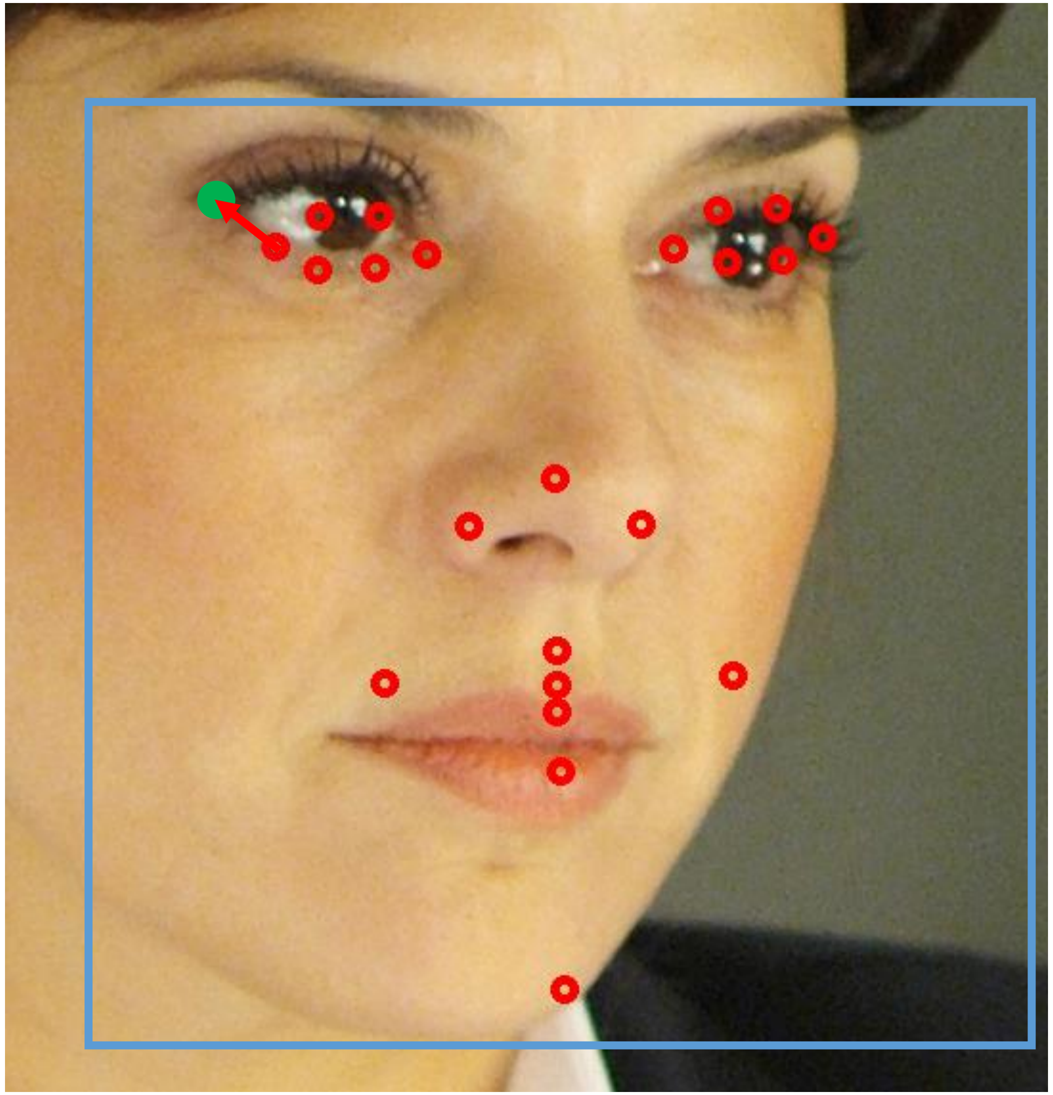
Applications

Frontalisation
Recognition
Expression Analysis
HCI
Animation

- Landmark detection & generic library
  - Algorithm, C++ library, interesting C++ bits
- 3D face reconstruction

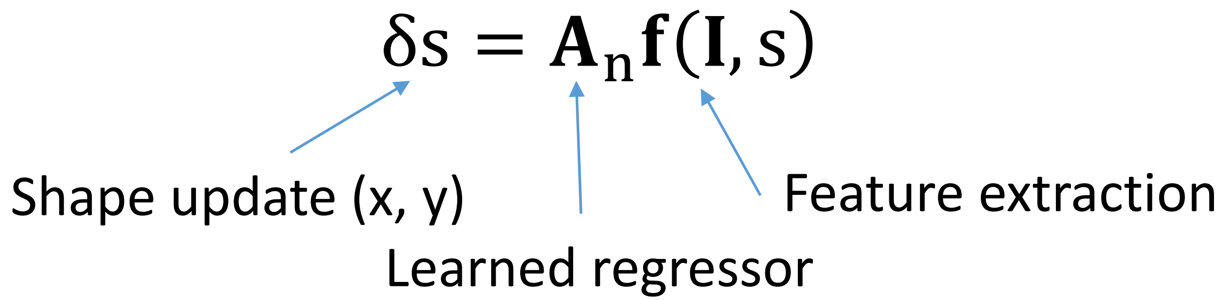
Main goal for us:
Make it easier to work with 3D models

Regression-based landmark detection



Regression-based landmark detection

- Learn «shape-update» to ground-truth location

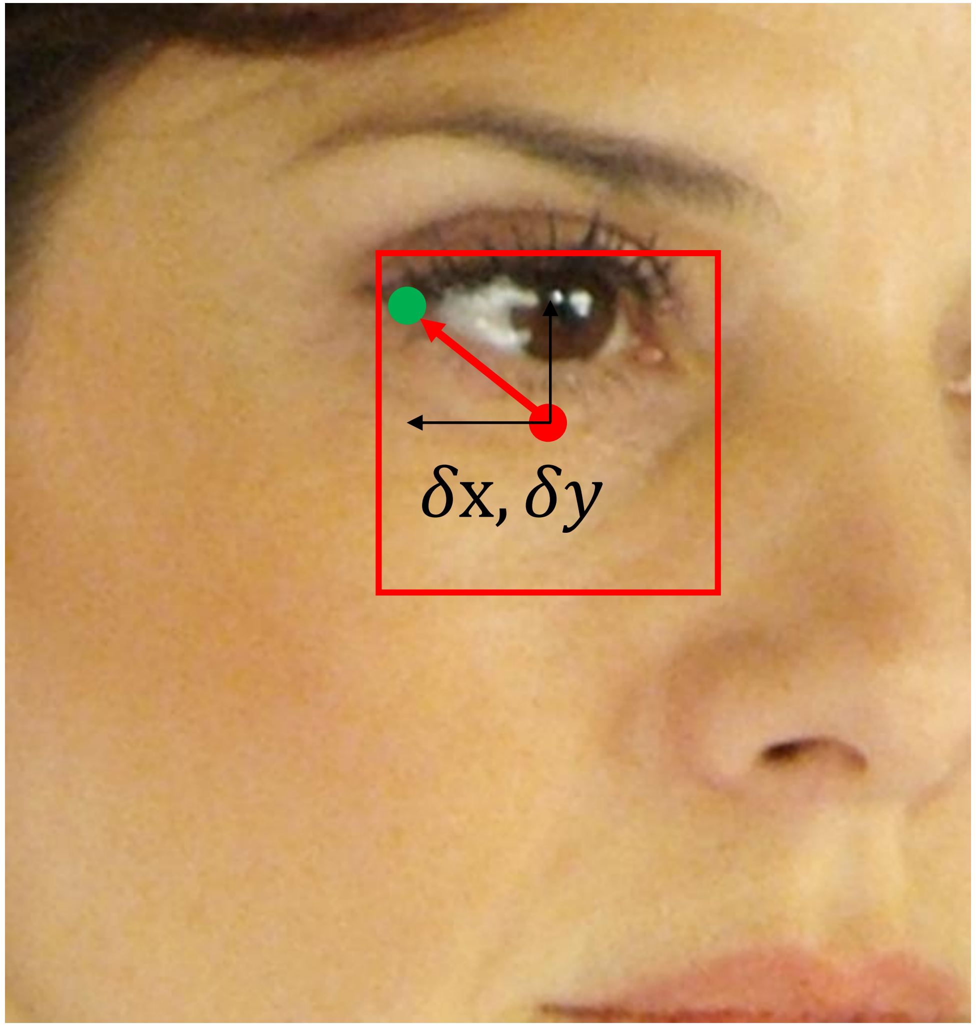


$$\delta s = \mathbf{A}_n \mathbf{f}(\mathbf{I}, s)$$

Shape update (x, y)

Learned regressor

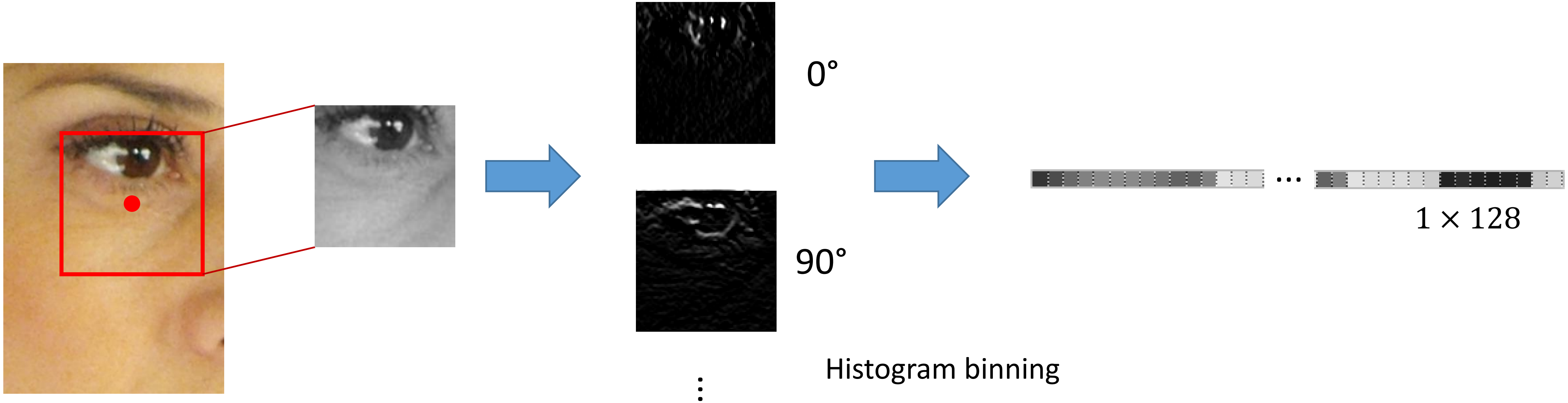
Feature extraction



Supervised Descent Method and Its Applications to Face Alignment, X. Xiong and F. De la Torre, CVPR 2013

A better representation: Local image features

- Feature extraction: Histogram of Oriented Gradients (HOG)



(simplified diagram)

Learning the model

- Learn using a bunch of training data (given images and landmarks)

$$\left[\quad \delta s \quad \right] = \left[\quad \mathbf{A}_{\mathrm{n}} \quad \right] \left[\quad \mathbf{f}(\mathbf{I}, s) \quad \right]$$

Generic supervised descent

- Generic formulation

$$\underbrace{\delta\boldsymbol{\theta}}_{\text{Parameter update}} = \underbrace{\mathbf{A}_{\mathrm{n}}}_{\text{Learned regressor}}\underbrace{\mathbf{f}(\mathbf{I}, \boldsymbol{\theta}, \dots)}_{\text{Feature extraction}}$$

We might also estimate
3D head pose parameters
$\boldsymbol{\theta} = [\mathrm{R_x}, \mathrm{R_y}, \mathrm{R_z}]$

- In previous case: $\boldsymbol{\theta} = [x_1, \dots, x_n, y_1, \dots, y_n]$

Based on: X. Xiong and F. De la Torre, “Supervised Descent Method for Solving Nonlinear Least Squares Problems in Computer Vision”, in submission to TPAMI

Modeling f(...)

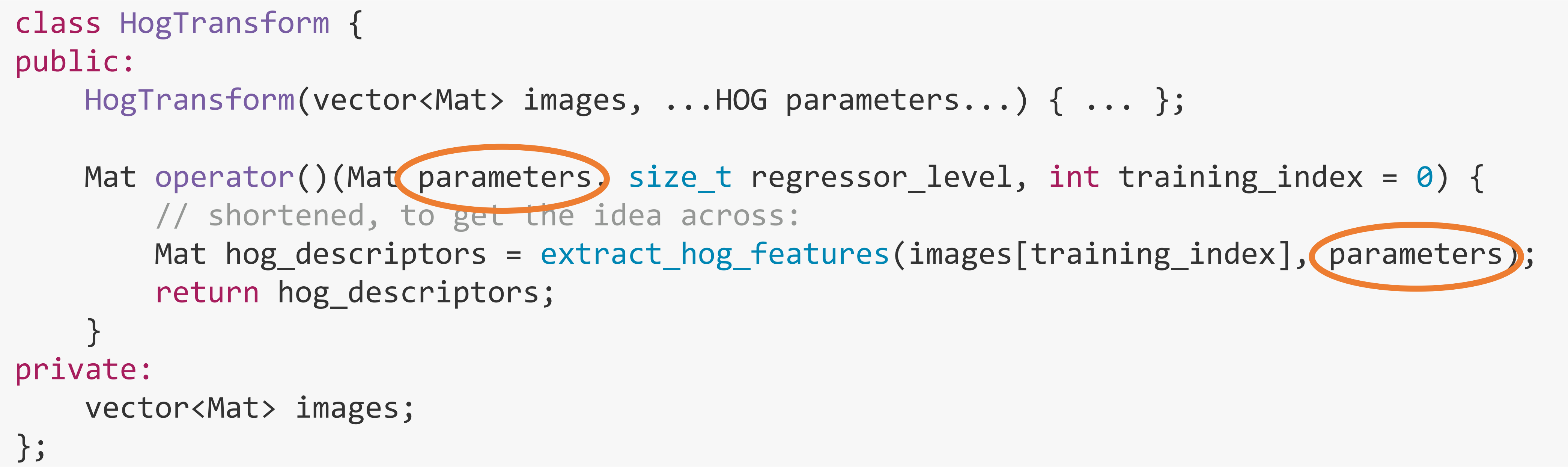
- Model projection `f(...)` as a function in the code

- Can be a lambda in trivial case

- Or function object with state or additional data in more complex scenarios

Hello world example

- $\mathbf{f}(\mathbf{I}, x)$ for 2D landmark detection:



```
class HogTransform {
public:
    HogTransform(vector<Mat> images, ...HOG parameters...) { ... };

    Mat operator()(Mat parameters, size_t regressor_level, int training_index = 0) {
        // shortened, to get the idea across:
        Mat hog_descriptors = extract_hog_features(images[training_index], parameters);
        return hog_descriptors;
    }
private:
    vector<Mat> images;
};
```

Interlude – OpenCV matrix class

- `cv::Mat`
  - Reference-counted
  - «Header» contains: rows, cols, flags
  - Images = matrices

CV_8UC4
CV_32SC1
CV_64FC3

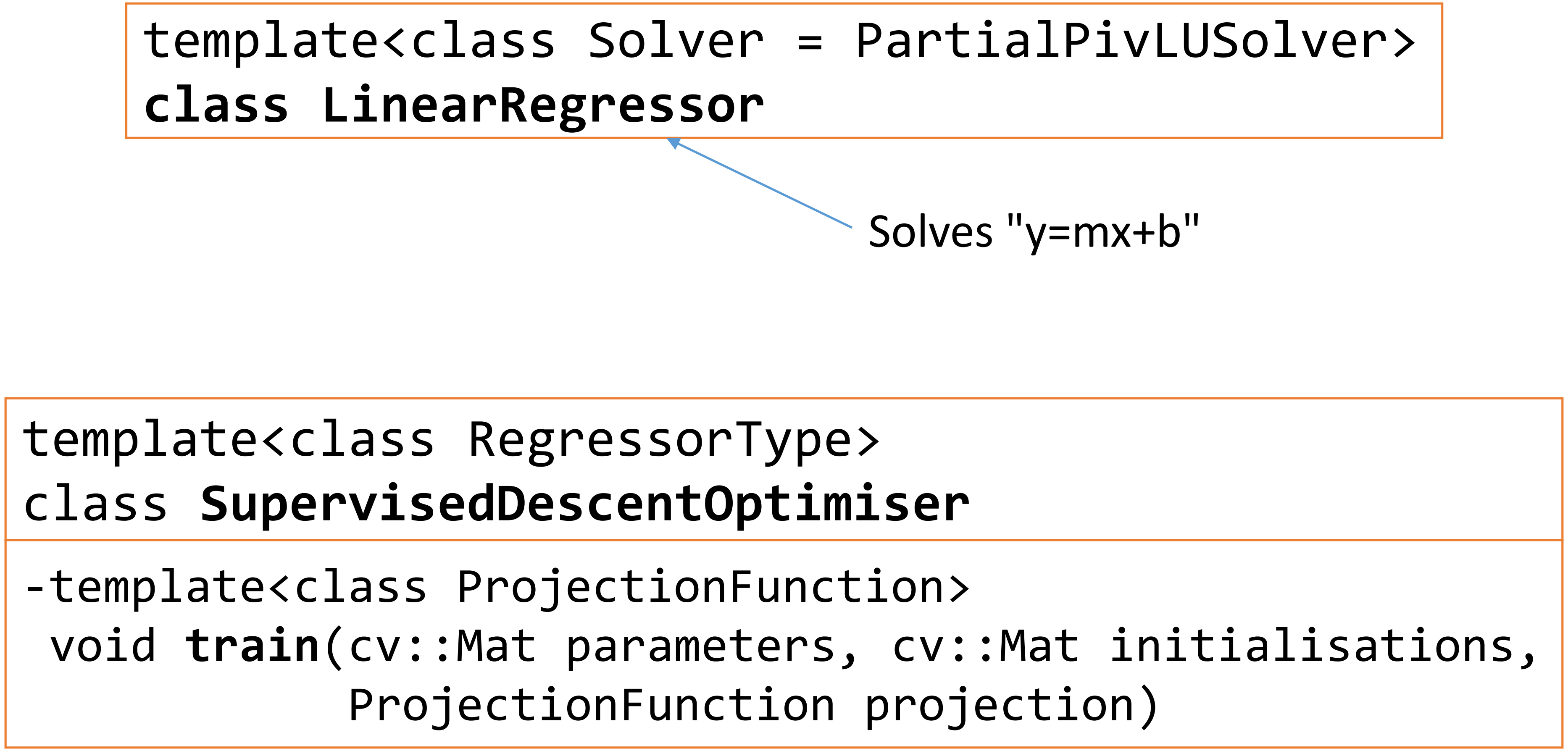
```
cv::Mat m = cv::Mat::ones(5, 5, CV_32FC1);
cv::Mat sub = m.rowRange(0, 2); // no data copy

float element = m.at<float>(1, 0);
```

Hello world example

- $\mathbf{f}(\mathbf{I}, x)$ for 2D landmark detection:

```
class HogTransform {
public:
    HogTransform(vector<Mat> images, ...HOG parameters...) { ... };

    Mat operator()(Mat parameters, size_t regressor_level, int training_index = 0) {
        // shortened, to get the idea across:
        Mat hog_descriptors = extract_hog_features(images[training_index], parameters);
        return hog_descriptors;
    }
private:
    vector<Mat> images;
};
```

Building blocks overview



```
template<class Solver = PartialPivLUSolver>
class LinearRegressor
```

Solves "y=mx+b"

```
template<class RegressorType>
class SupervisedDescentOptimiser
```

```
-template<class ProjectionFunction>
 void train(cv::Mat parameters, cv::Mat initialisations,
            ProjectionFunction projection)
```

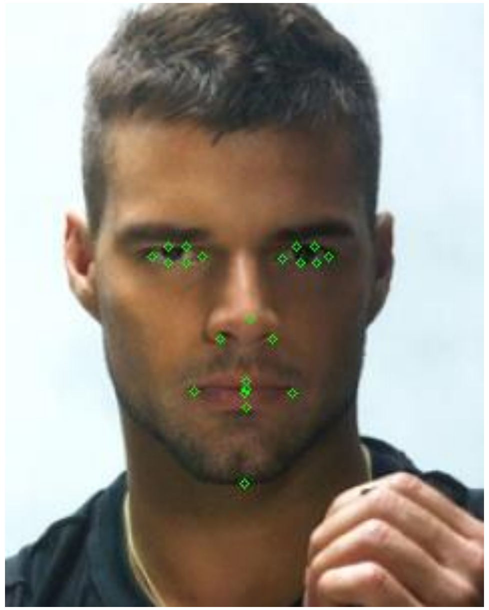
Hello world example

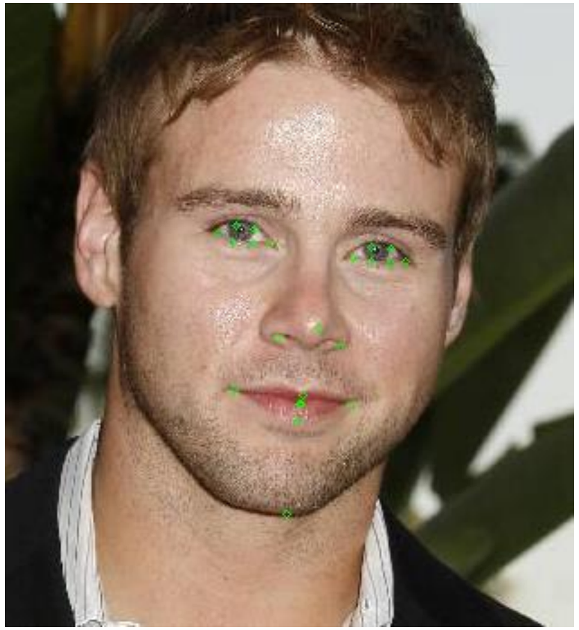
- Training the model:

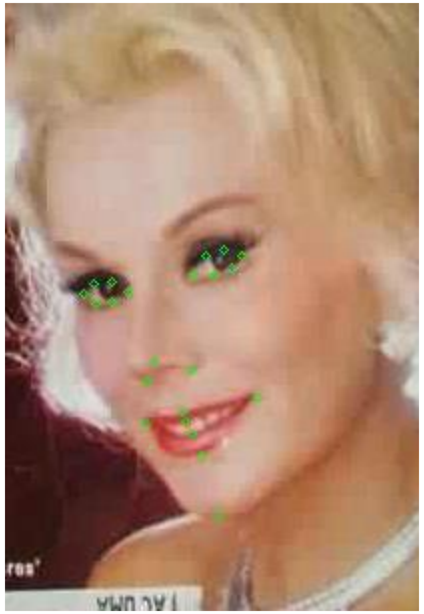
```
vector<cv::Mat> training_images = ...;
cv::Mat training_landmarks = ...; // each row is the landmarks for one image, i.e. [x_0, x_1, ...]
cv::Mat initialisations = ...; // generated initialisations

HogTransform hog(training_images, ...HoG parameters...);

vector<LinearRegressor<>> regressors(5);
supervised_descent_optimiser<LinearRegressor<>> model(regressors);

auto print_residual = [&training_landmarks](const Mat& current_predictions) {
    cout << cv::norm(current_predictions, training_landmarks, cv::NORM_L2)
        / cv::norm(training_landmarks, cv::NORM_L2) << endl;
};

model.train(training_landmarks, initialisations, hog, print_residual);

// store the model.
```

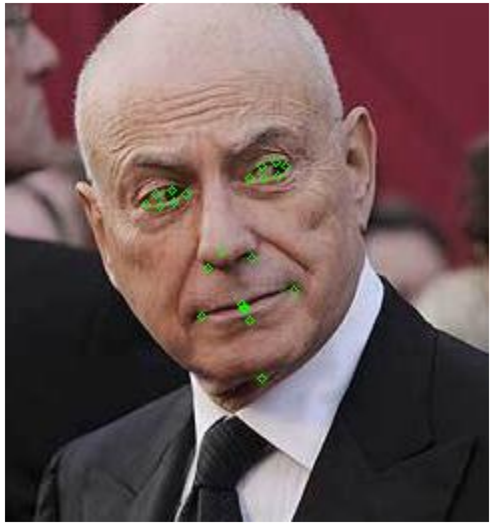
Using the model

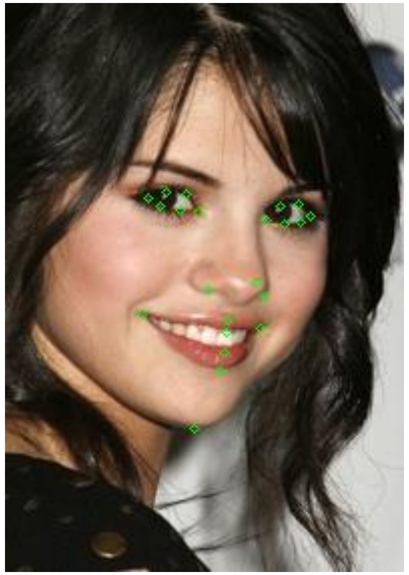
```
landmark_model<LinearRegressor<>> model;
{
    std::ifstream f(“model.bin”, std::ios::binary);
    cereal::BinaryInputArchive archive(f);
    archive(model);
}

cv::Mat image = cv::imread(imagefile);
auto landmarks = model.detect(image, hog, facebox);
```









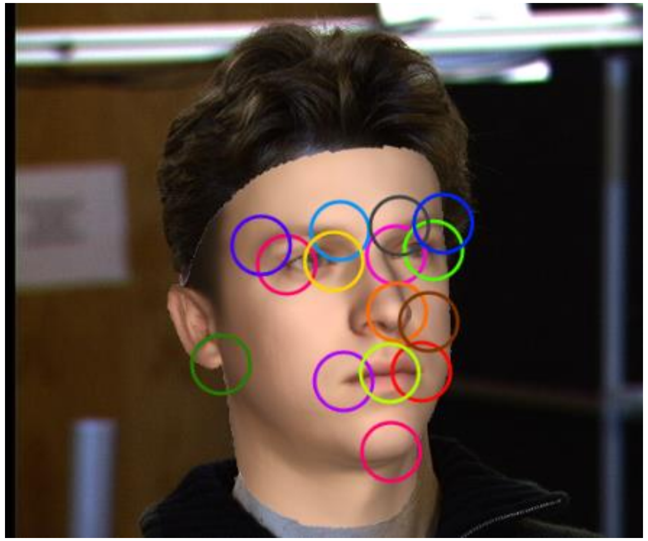


From *hello world* to the real world

- Perturbing the initialisations to get more diverse training data
- Learn the x/y update in normalised coordinates, not in pixels
- Change the size of the extraction window with each iteration

Using the generic code

- Estimate 3D face model parameters:
    - $\boldsymbol{\theta} = [r_x, r_y, r_z, t_x, t_y, t_z, \alpha_0, \alpha_1]$
    - $\mathbf{f}(\mathbf{I}, \boldsymbol{\theta})$ performs a projection from 3D to 2D using the current $\boldsymbol{\theta}$



C++ challenges

Solving the linear system of equations

- Matlab: `x = A \ b;`
  - Fast
  - Parallelised
  - Automatically chooses the best suitable algorithm (can be pseudo-inverse)
  - Warns if the system is not well-conditioned

- Our «A» can be 15000 x 8000 or larger

Solving the linear system of equations

- OpenCV `inv()`?
- Eigen?
  - `PartialPivLU?`
  - `FullPivLU?`
  - `ColPivHouseholderQR?`
  - `LDLT?`
  - `jacobiSvd?`
  - …
- …

Importing models from Matlab

- Not fun!

- Matlab SDK (ugly, hard to use C API)

- Text files and manual parsing

Storing & distributing the models in C++

- Simple task: Train a model on a local PC, upload it to repo, everybody should be able to load it

- Don’t do by hand, use a serialisation library

- XML and text format out of the question

Storing & distributing the models in C++

- Boost serialization:
    - It’s awesome!
    - But: Files stored with version x can’t be loaded with version <x
    - Older versions don’t compile on VS2015

- cereal:
    - C++11 library for serialization, header-only (https://github.com/USCiLab/cereal)
    - Can embed headers into our own code repo
    - Not so nice: No upgrade guarantee – new versions can break old files

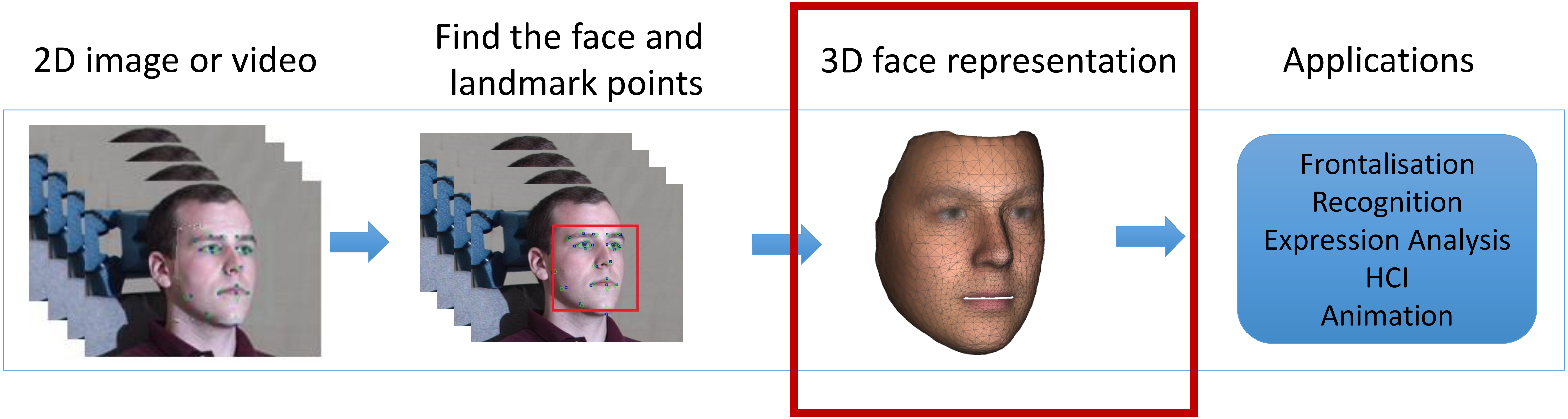
The superviseddescent library

- C++11, fully cross-platform
- Header-only
- CMake (for tests/examples)
- >=VS2013, >=gcc-4.8.2, >=clang-3.5

- License: Apache
- Repo: https://github.com/patrikhuber/superviseddescent

- Dependencies:
  - OpenCV core
  - Eigen
  - cereal

Face tracking and 3D face reconstruction



2D image or video

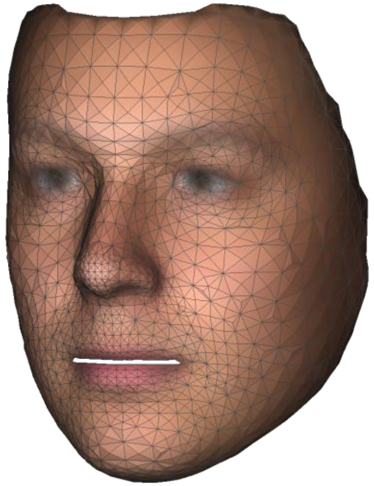
Find the face and landmark points

3D face representation

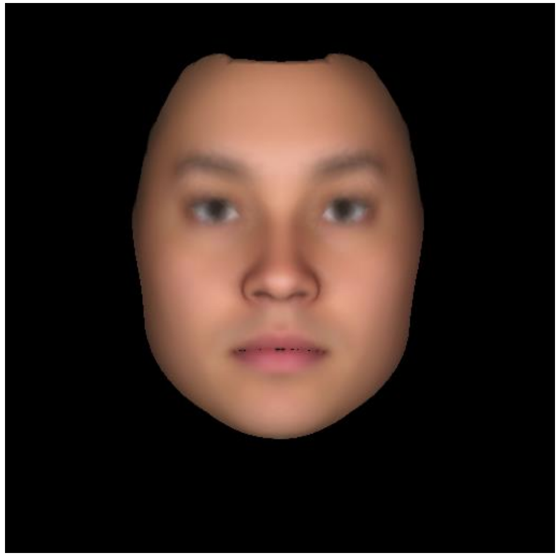
Applications

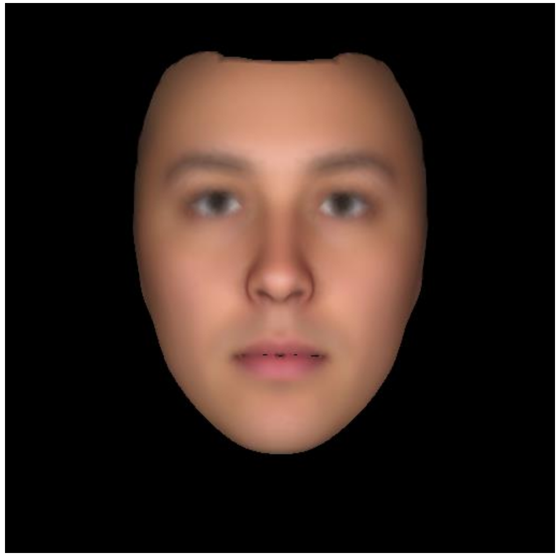
Frontalisation
Recognition
Expression Analysis
HCI
Animation

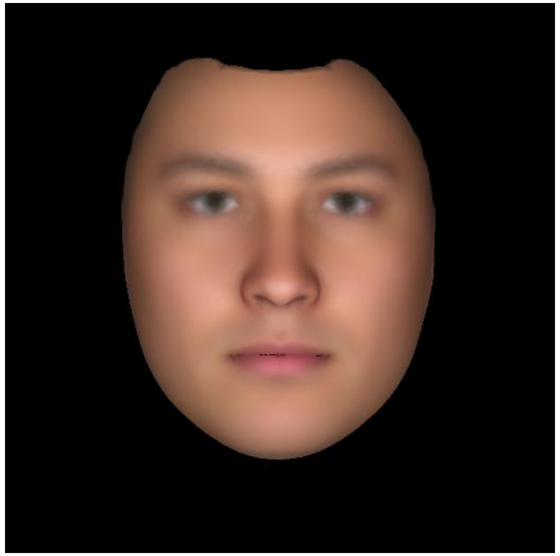
The face model



- A 3D Morphable Model
  - Shape and colour model («PCA» models), learned from 3D scans
  - Camera (rendering) parameters







Using the 3D model

```
MorphableModel morphable_model = morphablemodel::load(filename); // loaded using cereal
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");
```

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

3D model fitting library

- C++11/14, fully cross-platform
- Header-only
- Includes low-resolution 3D shape model
- CMake (for examples)
- >=VS2015, >=gcc-4.8.2, >=clang-3.5

- License: Apache
- Repo: https://github.com/patrikhuber/eos

- Dependencies:
  - OpenCV core
  - Eigen
  - cereal

Demo

Summing up...

- Other alternatives available for 2D-stuff
  - Without source code (e.g. Intraface) or only either Linux **or** Windows
  - Most source code is C-style code (e.g. clandmark)
  - A few notable exceptions: dlib, (OpenCV)
    - (and others, non-C++, e.g. menpo)

Summing up...

- Probably coming soon-ish: Better dealing with facial expressions

Summing up...

- Header-only libraries are great!

Summing up...

- OpenCV or not...
    - Try to integrate parts into OpenCV
      or...
    - Get rid of it, purely header-only easier for mobile & web

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)

UNIVERSITY OF SURREY

Hochschule Reutlingen
Reutlingen University

References & links

- Own & related publications:
  - **A Multiresolution 3D Morphable Face Model and Fitting Framework**, P. Huber, G. Hu, R. Tena, P. Mortazavian, W. Koppen, W. Christmas, M. Rätsch, J. Kittler, *in peer review (2015)*
  - **Fitting 3D Morphable Models using Local Features**, P. Huber, Z. Feng, W. Christmas, J. Kittler, M. Rätsch, *ICIP 2015*
  - **Random Cascaded-Regression Copse for Robust Facial Landmark Detection**, Z. Feng, P. Huber, J. Kittler, W. Christmas, X.J. Wu, *IEEE Signal Processing Letters, 2015*
  - **Supervised Descent Method and Its Applications to Face Alignment**, X. Xiong and F. De la Torre, *CVPR 2013*
  - **A Morphable Model for the Synthesis of 3D Faces**, V. Blanz and T. Vetter , *SIGGRAPH 1999*
- Links:
  - www.opencv.org

Thank you!

Questions?

patrikhuber@gmail.com | www.patrikhuber.ch