Statismo – A framework for statistical shape modeling

Marcel Lüthi
University of Basel
What is Statismo

• A C++ framework for statistical shape modeling
• Open source
  – www.github.com/statismo/statismo
• Goal:
  – Make exchange of model possible
  – Establish concepts and terminology
  – Make use of statistical shape models easier
Who is behind statismo

Statismo - A framework for PCA based statistical models.

Release 0.80

Marcel Lüthi¹, Rémi Blanc², Thomas Albrecht¹, Tobias Gass³, Orcun Goksel³, Philippe Büchler⁴, Michael Kistler⁴, Habib Bousleiman⁴, Mauricio Reyes⁴, Philippe C. Cattin⁵, Thomas Vetter¹

July 24, 2012

¹Department of Mathematics and Computer Science, University of Basel, Switzerland
²IMS Laboratory, University of Bordeaux, France
³Computer Vision Lab, ETH Zurich, Switzerland
⁴Institute for Surgical Technology and Biomechanics, University of Bern, Switzerland
⁵Medical Image Analysis Center, University of Basel, Switzerland
Goal of this tutorial

• Tell you what statismo is
  – and what it is not
• Teach you the basic concepts
• Show you how you can use it in practical applications
Outline

• Fundamentals and basic principles
• Using statismo
  – Building a face model
  – Model Fitting
• PCA Models
  • Posterior models
  • Gaussian Process Models
Design principles

• Clear interpretation
  – Impose concepts on users

• Support different toolkits
  – Don’t impose choices beyond shape modeling

• Stay focused on shape modeling
  – Reuse existing components from other projects

• Keep things as simple as possible
Statistical Shape Model

• A model of the normal shape of a structure
• A probability distribution
  – (Currently multivariate normal)
Statistical Shape Model

• Sampling:
  – Explore the shape variability
Statistical Shape Model

• Computing probability
  – Prior for Image Analysis
  – Shape analysis

\[ p(s) = 0.25 \]
\[ p(s) = 0.1 \]
\[ p(s) = 0.05 \]
Statistical Shape Model

• Compute conditional distributions
  – Reconstruction of traumatized parts

\[
\text{max } p(s \mid \text{part})
\]
Why statismo

• Shape models stored as matrices / vectors
  – No semantics (vector ≠ skull)
• Application and model logic entangled
  – Difficult to exchange models and algorithms
  – Code duplication
Statismo

- Statismo provides a high level API.
- Users works with shapes, not vectors.
- Core functionality implemented only once.
- Exchanging models becomes easy.

\[
\begin{align*}
\mu &= \begin{pmatrix} 0.1 \\ \vdots \\ 0.9 \end{pmatrix},
\quad U = \begin{pmatrix} 0.3 & \cdots & 0.1 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1.1 \end{pmatrix},
\quad D = \begin{pmatrix} 1e5 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0.01 \end{pmatrix}
\end{align*}
\]
Representers

Reference

toVector : dataset -> Vector
fromVector : vector -> dataset

M. Lüthi, University of Basel, 13.06.2014
Representers

\[
\begin{pmatrix}
p_x^1 \\
p_y^1 \\
p_z^1 \\
\vdots \\
p_x^n \\
p_y^n \\
p_z^n
\end{pmatrix}
\]

\(toVector: dataset \rightarrow Vector\)

\(fromVector: Vector \rightarrow dataset\)

M. Lüthi, University of Basel, 13.06.2014
Representers

\[ v_1, v_2, v_3, \ldots, v_{n-2}, v_{n-1}, v_n \]

**Reference**

toVector : dataset -> Vector
fromVector : vector -> dataset

M. Lüthi, University of Basel, 13.06.2014
Architecture

Data manager

Model Builder

Statistical model

PCA model builder

Posterior model builder

ModellInfo

M. Lüthi, University of Basel, 13.06.2014
Architecture

Data manager → Model Builder → Statistical model

- PCA model builder
- Posterior model builder

ModelInfo

\[
\begin{pmatrix}
v_1 \\
v_2 \\
v_3 \\
\vdots \\
v_{n-2} \\
v_{n-1} \\
v_n
\end{pmatrix}
\]

\[
\begin{pmatrix}
v_1 \\
v_2 \\
v_3 \\
\vdots \\
v_{n-2} \\
v_{n-1} \\
v_n
\end{pmatrix}
\]

M. Lüthi, University of Basel, 13.06.2014
Preparation

• Meshes in correspondence
Representers

template <class DSType>
class Representer {
    static Representer* Create(DSType* reference);
    VectorType DatasetToVector(DSType* t);
    DSType* VectorToDataset(VectorType& t);
}

M. Lüthi, University of Basel, 13.06.2014
Representers

• Available Representers
  – itkStandardMeshRepresenter
    • Triangle Meshes (itk::Mesh)
  – itkStandardImageRepresenter
    • Scalar images and Vector fields (itk::Image)
  – vtkStandardImageRepresenter
    • Scalar images and Vector fields (vtkStructuresPoints)
  – vtkStandardMeshRepresenter
    • Meshes (vtkPolyData)
Basic definitions

typedef DataManager<vtkPolyData> DataManagerType;
typedef PCAModelBuilder<vtkPolyData> ModelBuilderType;
typedef StatisticalModel<vtkPolyData> StatisticalModelType;
vtkPolyData* reference = loadSurface(referenceFn);

vtkStandardMeshRepresenter* representer =
    vtkStandardMeshRepresenter::Create(reference));
Building a PCA Model

DataManagerType* dm = DataManagerType::Create(representer);

for (unsigned i = 0; i < nDatasets; i++) {
    vtkPolyData* dataset = loadMesh(filenames[i]);
    dm->AddDataset(dataset, filenames[i]);
}

ModelBuilderType* builder = ModelBuilderType::Create();
StatisticalModelType* model =
    builder->BuildNewModel(dm->GetData());

model->Save(modelName);
Statismo file format

https://github.com/statismo/statismo/wiki/The-Statismo-File-Format
Sampling

\[
\text{vtkPolyData* mean = model->DrawMean();}
\]

\[
\text{vtkPolyData* rse = model->DrawSample();}
\]

\[
\text{unsigned n = model->GetNumberOfPrincipalComponents();}
\]

\[
\text{VectorType coefficients = VectorType::Ones(n);}
\]

\[
\text{vtkPolyData* sOne = model->DrawSample(coefficients);}
\]
Statismo Viewer
Model fitting

• Fitting using the ITK Registration framework
Model Fitting, Setup

typedef itk::StandardMeshRepresenter<float, 3> RepresenterType;
typedef RepresenterType::DatasetType MeshType;
typedef itk::StatisticalModel<MeshType> StatisticalModelType;
typedef itk::StatisticalShapeModelTransform<MeshType, double, 3> TransformType;

typedef itk::EuclideanDistancePointMetric<MeshType, MeshType> MetricType;
typedef itk::PointSetToPointSetRegistrationMethod<MeshType, MeshType> RegistrationFilterType;
typedef itk::LevenbergMarquardtOptimizer OptimizerType;
MeshType::Pointer referenceMesh = model->GetRepresenter()->GetReference();
TransformType::Pointer transform = TransformType::New();
transform->SetStatisticalModel(model);

MetricType::Pointer metric = MetricType::New();
OptimizerType::Pointer optimizer = OptimizerType::New();
optimizer->SetNumberOfIterations(100);
Model Fitting (2)

RegistrationFilterType::Pointer registration =
RegistrationFilterType::New();
registration->SetMetric(metric);
registration->SetOptimizer(optimizer);
registration->SetTransform(transform);
registration->SetFixedPointSet(targetMesh);
registration->SetMovingPointSet(referenceMesh);

registration->Update();
Model builders

M. Lüthi, University of Basel, 13.06.2014
Model builders

- **PCAModelBuilder**
  - Standard PCA Models
- **ConditionalModelBuilder**
  - Include surrogate variables
- **PosteriorModelBuilder**
  - Condition on landmarks
- **GaussianProcessModelBuilder**
  - Generate models from smooth deformations
GAUSSIAN PROCESS MODELS
The normal model

- $N(\mu, \Sigma)$ with

$$\mu = \begin{bmatrix} \mu_{1x} \\ \mu_{1y} \\ \vdots \\ \mu_{nx} \\ \mu_{ny} \\ \mu_{nz} \end{bmatrix} \quad \Sigma = \begin{bmatrix} \Sigma_{1x1x} & \Sigma_{1x1y} & \Sigma_{1x1z} \\ \Sigma_{1y1x} & \Sigma_{1y1y} & \Sigma_{1y1z} \\ \Sigma_{1z1x} & \Sigma_{1z1y} & \Sigma_{1z1z} \\ \vdots & \vdots & \vdots \\ \Sigma_{nx1x} & \Sigma_{nx1y} & \Sigma_{nx1z} \\ \Sigma_{ny1x} & \Sigma_{ny1y} & \Sigma_{ny1z} \\ \Sigma_{nz1x} & \Sigma_{nz1y} & \Sigma_{nz1z} \end{bmatrix} \ldots \begin{bmatrix} \Sigma_{1xn} & \Sigma_{1xn} & \Sigma_{1xz} \\ \Sigma_{ynx} & \Sigma_{ynx} & \Sigma_{ynz} \\ \Sigma_{znx} & \Sigma_{znx} & \Sigma_{znz} \end{bmatrix}$$
The normal model

\[ N(\mu, \Sigma) \] with

\[ \mu = \begin{bmatrix} \mu_{1x} \\ \mu_{1y} \\ \mu_{1z} \\ \vdots \\ \mu_{nx} \\ \mu_{ny} \\ \mu_{nz} \end{bmatrix} \]

\[ \Sigma = \begin{bmatrix} \Sigma_{1x1x} & \Sigma_{1x1y} & \Sigma_{1x1z} \\ \Sigma_{1y1x} & \Sigma_{1y1y} & \Sigma_{1y1z} \\ \Sigma_{1z1x} & \Sigma_{1z1y} & \Sigma_{1z1z} \\ \vdots & \vdots & \vdots \\ \Sigma_{nx1x} & \Sigma_{nx1y} & \Sigma_{nx1z} \\ \Sigma_{ny1x} & \Sigma_{ny1y} & \Sigma_{ny1z} \\ \Sigma_{nz1x} & \Sigma_{nz1y} & \Sigma_{nz1z} \end{bmatrix} \]

M. Lüthi, University of Basel, 13.06.2014
The normal model

Gaussian process

- \( u \sim \text{GP}(\mu, k) \)
  - \( \mu : \Gamma_R \rightarrow R^2 \)
    - mean deformation
  - \( k : \Gamma_R \times \Gamma_R \rightarrow R^{2\times2} \)
    - kernel function

Example

- \( \mu(x) = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \)
- \( k_\sigma(x, y) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \exp\left( -\frac{\|x-y\|^2}{\sigma^2} \right) \)
Back to PCA Models

Deformation model:

\[ u \sim GP(\mu, k) \]

\[ u \sim \mu + \sum_{i=1}^{n} \alpha_i \lambda_i \phi_i \quad \alpha_i \sim N(0,1) \]

• Mercer’s theorem:
  - \( k(x, y) = \sum_{i=1}^{\infty} \lambda_i \phi_i(x)\phi_i(y) \)
  - Only the first basis functions are important
Kernels in statismo

class GaussianKernel: public MatrixValuedKernel<vtkPoint> {

public:

    MatrixType operator()(const vtkPoint& x, const vtkPoint& y)
    {
        VectorType r(3);
        r << x[0] - y[0], x[1] - y[1], x[2] - y[2];
        return MatrixType::eye(3) * exp(-r.dot(r) / m_sigma2);
    }

};
Gaussian Process model builder

```cpp
vtkStandardMeshRepresenter* representer = 
    vtkStandardMeshRepresenter::Create();

const GaussianKernel gk = GaussianKernel(gaussianKernelSigma);
vtkPolyData* reference = loadMesh(referenceFilename)

ModelBuilderType* modelBuilder =
    ModelBuilderType::Create(model->GetRepresenter());

StatisticalModelType* combinedModel =
    modelBuilder->BuildNewModel (reference,
    combinedModelAndGaussKernel, numberOfComponents));
```
MODELING WITH KERNELS
Statistical shape model

\[ k_{SSM}(x, y) = \frac{1}{n} \sum (u_i(x) - \bar{u}(x))(u_i(y) - \bar{u}(y))^t \]

M. Lüthi, University of Basel, 13.06.2014
Modeling smooth deformations

\[ k_{g70}(x, y) = 70 I_{3 \times 3} \exp\left(-\frac{\|x-y\|^2}{70}\right) \]
Composing kernels

It’s easy to build complicated kernels from simple ones:

- $k(x, z) = k_1(x, z) + k_2(x, z)$
- $k(x, z) = s \, k_1(x, z)$
- $k(x, z) = k_1(x, z) \, k_2(x, z)$
- ...
Modeling smooth deformations

\[ k_{g70}(x, y) = 70 I_{3 \times 3} \exp(-\frac{\|x-y\|^2}{70}) \]
Modeling smooth deformations

\[ k_{g_{10}}(x, y) = 10 I_{3 \times 3} \exp\left(-\frac{\|x-y\|^2}{10}\right) \]

M. Lüthi, University of Basel, 13.06.2014
Multiscale model

\[ k_{MS}(x,y) = k_{g10}(x,y) + k_{g70}(x,y) \]
Localized statistical shape models

\[ k_{local}(x, y) = \exp\left(-\frac{\|x-y\|^2}{50}\right) k_{SSM}(x, y) \]
Gaussian Process Regression

$GP(\mu, k)$

$x = \{(x_1, u(x_1)), \ldots, (x_n, u(x_n))\}$

$GP(\mu_p, k_p)$

Source: Rasmussen et al. Gaussian Processes for machine learning
Hybrid registration
Conclusion

• Statismo is a framework for creating PCA Models
  – Goes far beyond standard PCA models
• It can easily adapt to your toolkits
• It provides a standard file format
  – Easy to exchange models
  – Easy to share tools for shape modeling
Conclusion

• Statismo is a framework for creating PCA Models
  – Goes far beyond standard PCA models
• It can easily adapt to your toolkits
• It provides a standard file format
  – Easy to exchange models
  – Easy to share tools for shape modeling
How to contribute

• Fork the project on Github
  – Add model builders
  – Add representers
• Help to make it more stable
  – Report bugs
  – Improve installation procedure
• Join the mailing list
  – statismo-users@googlegroups.com