Creating a 2D Active Shape Model Using itk::ImagePCAShapeModelEstimator

John R. Durkin[1], David Miller[2], Kenneth L. Urish[3*]

1 University of Pittsburgh School of Medicine, Pittsburgh, PA
2 The Pennsylvania State University Department of Electrical Engineering, State College, PA
3 The Penn State Milton S. Hershey Medical Center, Hershey, PA

Abstract

Although many variations of active contour segmentation algorithms exist, most are based on solely edge criteria and breakdown or leak at weak boundaries. One solution to this problem is constraining the segmented area to only statistically possible shapes with the guidance of a shape model. The purpose of this document is to fill the void in the ITK user guide on building active shape models. We describe how to create a 2d active shape model of articular femoral knee cartilage using ITK's ImagePCAShapeModelEstimator. Sample code and example images are provided for displaying the initial principle components of variation. Shape models built with our code can be used for segmentation with itk::GeodesicActiveContourShapePriorLevelSetImageFilter.

Keywords: Active shape model, segmentation,

1 Introduction

The Insight Toolkit (ITK) is an excellent example of an open source framework in extreme programming. Strengths of the toolkit include superb documentation and list serve support providing a supportive and vibrant programming community. One missing area of documentation in the ITK software guide includes the implementation of itk::GeodesicActiveContourShapePriorLevelSetImageFilter. The generation of the active shape model (ASM) in this example is provided with no example code or images. Here, we provide sample images and commented code to generate an ASM using itk::ImagePCAShapeModelEstimator. We use the challenging problem of segmenting the cartilage of the femoral condyle of the knee in our example.

Many classic described segmentation algorithms such as active contours (snakes), level set, and watershed rely mostly on edge based criteria [1-4]. For segmentation of articular femoral knee cartilage, a solely edge-based algorithm is inadequate because of the poor contrast at the cartilage and soft tissue interfaces. One approach is to use a priori information to create an active shape model to help guide segmentation [5].

The goal of an active shape model is to appropriately describe all of the allowed statistical variation of some generic shape. The model is formed by using a training set of objects that are already segmented either manually or by some other automated or semi-automated method. The variation among the set of these previously segmented shapes is used to describe the variation of the shape model and therefore the training set needs to be a good representation of the overall distribution of allowable shapes. This can be accomplished by a large sample size or good variation among the training images.

Principal component analysis (PCA) is used to decompose the large variation in the natural shapes of femoral cartilage or other shapes into a set of variables that describe the majority of the variation. The basic steps of PCA include aligning the training images, finding the corresponding landmarks, computing the covariance matrix of these landmarks, and finally finding the determinate of the covariance matrix.

In the first step of PCA, the training images should be transformed iteratively to maximize their overlapping area. In the classical implementation, the N_0 iteration consisted of the training images being registered with the first training image using a similarity transform that can rotate, scale, and translate the image. After this first iteration, the mean image is computed. For the following iterations, the training images are registered to the mean image and the mean image is recomputed. This process eventually converges, and the result is a set of training images aligned with maximal overlap. A subset of aligned training images used to create the femoral cartilage shape model are shown in figure 2-1.

From this set of transformed, registered training images, a mean model is made by taking the mean of each pixel in the training set.

$$\bar{I} = \frac{\sum_{i=1}^{N} I_i}{N}$$

In classical shape model construction, the first step of the creation of an active shape model is the identification of a set of landmarks along the outline of the shape or in this case, femoral cartilage. Landmarks are a set of points (x,y) in the training images that describe the outline of the shapes. Each image should have the same number of landmarks and they should correspond to the same location on the general shape. The itk:ImagePCAShapeModelEstimator, however, implements automatic landmark detection by sending the images through a distance filter that changes pixel values to their distance from the border of the shape. Pixels inside the shape are given a negative distance and pixels outside the shape are given a positive distance. Each pixel location (i,j) in the set of training images corresponds to a landmark.

Each of these landmarks is then tracked on each training image. The N landmarks are represented by variables X_0, X_1, …, X_{n-1}. Each training image is subtracted from the mean image so that the data is normalized for principal component analysis.

$$I'_i = I_i - \bar{I}$$

The nxn covariance matrix is then computed. The size of the covariance matrix is equal to N^2 where N is the number of landmarks.

$$\text{Cov} = \begin{pmatrix} Cov_{x0x0} & \cdots & Cov_{x0xn-1} \\ \vdots & \ddots & \vdots \\ Cov_{xn-1x0} & \cdots & Cov_{xn-1xn-1} \end{pmatrix}$$

After principal component analysis is performed by finding the determinate of the covariance matrix, we have a set of eigenvalues and corresponding eigenvectors [6,7]. All of the training images can be reconstructed by combining a linear combination of the eigenvectors and the mean image. An ample number of eigenvectors are chosen such that the sum of their eigenvalues yield a large portion of variation.

This model is then used with a normal active contour algorithm to help constrain the region growing to only statistically likely representations of the shape [8]. For an implementation of this segmentation see the ITK Software Guide's description of GeoesdicActiveContoursWithShapeGuidance [9].

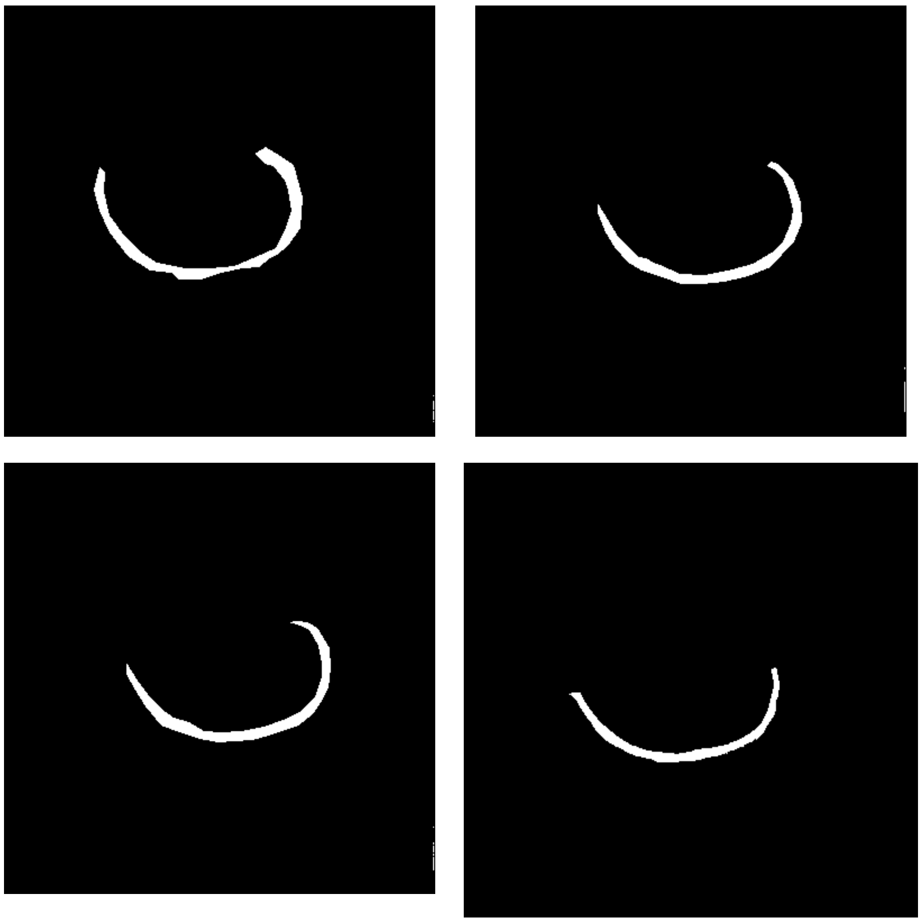

Figure 1-1 Subsample of the training images used

2 Implementation

Before a set of training images can be used, they must be registered to maximize overlap. This can be done manually, by centering each image over a predefined landmark or other methods, but for large datasets it is desirable to have an automated registration method. In this example, itk::TranslationTransform based on ImageRegistration1.cxx in the itk example library was used. Each training image was registered through multiple iterations. For the first round of registration, the fixed image was chosen to be the first image in the training set, and the remaining training images were the moving images. For the nth iteration, the mean image of the registered n-1 iteration was used as fixed image and the registered n-1 iteration images were used as the moving images. This process will eventually converge to a set of training images registered to each other.

Once the registration process is complete, the training images are ready to be used to create the active shape model.

First we set the number of training images we are using along with the prefix of training images' file names. In this example we assume there are six training images names fixed1.jpg, fixed2.jpg, fixed2.jpg … fixed6.jpg. We also set the number of principal component images we wish to output.

```
int NUMTRAININGIMAGES = 6;
int NUMPC = 3;
std::string IMAGENAME = "fixed";
```

The input to the PCAShapeModelEstimator is expected to be signed distance images. The signed distance image is the distance to the edge of the shape. A negative distance is given for pixels inside the shape and a positive distance is given for pixels outside the shape.

```
typedef itk::SignedDanielssonDistanceMapImageFilter< OutputImageType,
MapOutputImageType> FilterType;
```

Finally we create a typedef for the ImagePCAShapeModelEstimator and a smart pointer instance of this filter. The number of training images and principal components are set.

```
typedef itk::ImagePCAShapeModelEstimator<MapOutputImageType,
InternalImageType> PCAEstimatorType;
PCAEstimatorType::Pointer model = PCAEstimatorType::New();

model->SetNumberOfTrainingImages(NUMTRAININGIMAGES);
model->SetNumberOfPrincipalComponentsRequired(NUMPC);
```

Next, each trainig image is first read and tresholded to ensure it is a binary image, sent through the signed danielsson distance filter, and then set as the kth input to the PCAShapeModelEstimator.

```
for ( unsigned int k = 0; k < NUMTRAININGIMAGES; k++ )
{

  ThresholdingFilterType::Pointer thresholder =
  ThresholdingFilterType::New();

  thresholder->SetLowerThreshold( 255 );
  thresholder->SetUpperThreshold( 255 );
  thresholder->SetOutsideValue(  0  );
  thresholder->SetInsideValue(  255 );

  ReaderType::Pointer reader = ReaderType::New();
  FilterType::Pointer filter = FilterType::New();

  reader->SetFileName(trainingImageNames[k].c_str());
  thresholder->SetInput(reader->GetOutput());
  filter->SetInput(thresholder->GetOutput());
  filter->Update();
```

```
      model->SetInput(k, filter->GetOutput());
    }
```

Next we call update to envoke the estimator. When it is finished we call its print method to output information about the filter.

```
    model->Update();
    model->Print(std::cout);
```

Now the principal component and mean image must be saved. The 0th output of the ImagePCAShapeModelEstimator is the mean image and the nth principal component image is the n+1th output of the filter. The principal component images names are created using itkNumericSeriesFileNames.

```
    WriterType::Pointer writer = WriterType::New();
    writer->SetFileName("meanImage.mha");
    writer->SetInput(model->GetOutput(0));
    writer->Update();
```

We print the eigenvalues to the console.

```
    vnl_vector<double> eigenValues(NUMPC);
    eigenValues = model->GetEigenValues();
    std::cout << eigenValues << stdd:endl;
```

Finally, each of the principal components are written.

```
    for(unsigned int i = 0; i < NUMPC; i++)
    {
          //normalizing the images
          DivideFilterType::Pointer divider = DivideFilterType::New();
          divider->SetInput(model->GetOutput(i+1));
          divider->SetScale(1.0/sqrt(eigenValues(i)));

          WriterType::Pointer myWriter = WriterType::New();
          myWriter->SetFileName(outputImageNames[i].c_str());
          myWriter->SetInput(model->GetOutput(i+1));
          myWriter->Update();
    }
```

3 Viewing Variations of the Shape Model

The variations of the shape model can be viewed to verify accuracy and to visually appreciate the source of deviation in the model. In theory, we can view any linear combination of the principal components, but in practice it is easier to view each principal component individually. The following code can be found in ViewShapeModel.cxx and is used to view any real number multiple of standard deviations of a principal component.

The mean image and principal component image are read. We use the MultiplyByConstantImageFilter to multiply the scale constant to the principal component image.

```
Typedef itk::MultiplyByConstantImageFilter<InternalImageType,float,
InternalImageType> MultType;
MultType::Pointer multFilter = MultType::New();
MultType::Pointer multFilter2 = MultType::New();

multFilter->SetInput(reader2->GetOutput());
multFilter->SetConstant((float)SCALE_CONSTANT);
multFilter2->SetInput(reader3->GetOutput());
multFilter2->SetConstant((float)SCALE_CONSTANT);

try
{
      multFilter->Update();
      multFilter2->Update();
}
catch( itk::ExceptionObject & err )
{
      std::cout << "multFilter exception caught !" << std::endl;
      std::cout << err << std::endl;
}
```

The AddImageFilter is used to add the offset to the mean image to get the positive deviation from the mean shape.

```
typedef itk::AddImageFilter<InternalImageType, InternalImageType,
InternalImageType> AddFilterType;
AddFilterType::Pointer addFilter = AddFilterType::New();
addFilter->SetInput1(reader1->GetOutput());
addFilter->SetInput2(multFilter->GetOutput());
```

The SubtractImageFilter is used to subtract the offset from the mean image to get the negative deviation from the mean shape.

```
typedef itk::SubtractImageFilter<InternalImageType, InternalImageType,
InternalImageType> SubFilterType;
SubFilterType::Pointer subFilter = SubFilterType::New();
subFilter->SetInput1(reader1->GetOutput());
subFilter->SetInput2(multFilter2->GetOutput());
```

The output of the filters are thresholded. A value of zero should corrispond the the boudnary of the shape.

```
typedef itk::BinaryThresholdImageFilter<
InternalImageType, OutputImageType> ThresholdingFilterType;

ThresholdingFilterType::Pointer thresholder =
ThresholdingFilterType::New();
ThresholdingFilterType::Pointer thresholder2 =
ThresholdingFilterType::New();

thresholder->SetLowerThreshold( low );
thresholder->SetUpperThreshold( high );
thresholder->SetOutsideValue(  0  );
thresholder->SetInsideValue(  255 );
```

```
      thresholder->SetInput(addFilter->GetOutput());

      thresholder2->SetLowerThreshold( low );
      thresholder2->SetUpperThreshold( high );

      thresholder2->SetOutsideValue(  0  );
      thresholder2->SetInsideValue(  255 );
      thresholder2->SetInput(subFilter->GetOutput());
```

The output images are then written

```
      writer->SetInput( thresholder->GetOutput() );
      writer->SetFileName(outputfile1);

      writer2->SetInput( thresholder2->GetOutput() );
      writer2->SetFileName(outputfile2);
```

4 Results

The mean image for the femoral cartilage is shown in figure 4-1. It clearly resembles the average of all the training images. The eigenvalues for each principal component of the femoral cartilage shape model are given in table 4-2. The first principle component contains the majority of the variation at about 70%. Using the first four principle components, about 96% of the variation of the knee cartilage can be represented [8-9]. Figure 4-3 shows the first four principle components influence on the mean shape. The first principle component controls the shift in the mass of the cartilage from the left to the right side. The influence of the other components is less easily describable but can be seen.

There are some common pitfalls of using active shape models for segmentation. One is not having an adequate size training set. Here our training image set contains only 6 images as an example for implementation. In practice, an image set should contain approximately 100 images to capture the necessary statistical variance in the shape to have an acceptable segmentation accuracy. Another pitfall is lacking significant overlap of the training images. This will manifest itself as a completely black mean image or a mean image with a smaller than expected mass. Finally, attempting to segment an object that is outside the deformability of the shape model can result in poor results. For example, trying to segment greatly degraded cartilage with an active shape model created with all healthy cartilage.

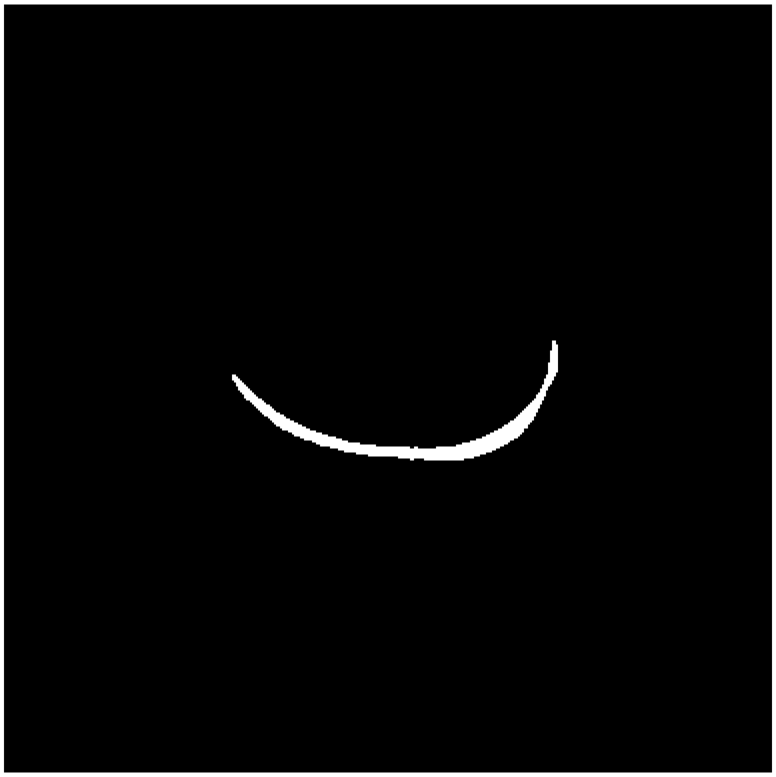

Figure 4-1 mean image for femoral articular knee cartilage.

λ in order of decreasing magnitude	$\frac{\lambda}{\sum \lambda}$
8.13E+06	54%
4.34E+06	29%
1.76E+06	12%
4.23E+05	3%
1.32E+05	1%
9.03E+04	1%
7.59E+04	1%
4.81E+04	0%
3.89E+04	0%
3.42E+04	0%
2.22E+04	0%
1.57E+04	0%
1.36E+04	0%
1.01E+04	0%
8.51E+03	0%
6.48E+03	0%
5.84E+03	0%
5.02E+03	0%

Table 4-2 Eigenvalues for each principal component and the percent of the sum of all eigenvalues.

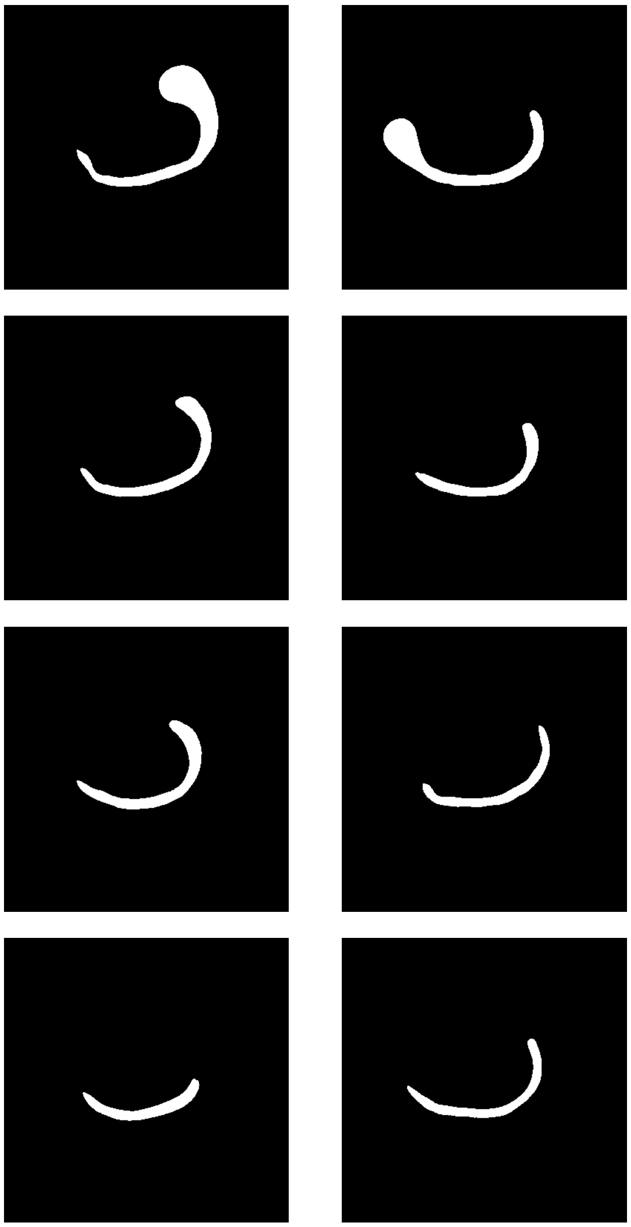

Figure 4-3. $\bar{X} + \square 2\sigma$ for λ1-λ4 from top to bottom

Reference

[1] Sethian, J.A. "Level Set Methods: An Act of Violance. Cambridge University Press, 1996.

[2] Caselles V, Kimmel R, and Sapiro G. "Geodesic Active Contours". International Journal of Computer Vision. 22(1), 61-79(1997).

[3] Kass M, Witkin A, Terzopoulos D. "Snakes: Active Contour Models". International Journal of Computer Vision. 1(4), 321-331(1988).

[4] Beucher S and Meyer F. "The Morphological Approach of Segmentation: The Watershed Transformation". Mathematical Morphology in Image Processing, E. Dougherty, ed., chapter 12, pp. 43-481, New York: Marcel Dekker, 1992.

[5] Cootes T, Taylor C, Cooper D, and Graham J. "Active Shape Models—Their Training and Application". Computer Vision and Image Understanding Vol. 61, No. 1, January, pp. 38-59, 1995.

[6] Flury, Bernhard and Riedwyl, Hans. *Multivariate Statistics a practical approach.* New York, NY: Chapman and Hall 1988.

[7] Smith, Lindsay. "A tutorial on Principal Component Analysis". 26 February, 2002. <http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf>

[8] Leventon M, Grimson W, and Faugeras O. "Statistical shape influence in geodesic active contours. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition.* Volumne 1. Pp. 316-323, 2000.

[9] Ibanez L, Schroeder W, Ng L, Cates J. The ITK Software Guide Second Edition. November 21, 2005.