Table of Contents

1 BACKGROUND

Model-based vision is a type of computer vision that makes use of some prior knowledge of the object to be modeled, how it looks, how it can change, and so on. A key issue when building models is not only the need for flexibility to cover the variations present among the objects in such a training set, but also the need to restrict this flexibility so as to be able to represent only “legal” or “plausible” examples of similar objects. Deciding how to deal with the balance between rigidness and flexibility often entails knowing something about the nature of the modeled objects, for example, finding a set of permissible transformations of the model shape.

Active shape modeling (ASM) is model-based vision that provide solutions to deal with the rigidity/flexibility problem using statistical analysis of the training corpora—a set of images of similar objects. ASM make use of patterns of variability based on shape. It captures how the object changes by statistically modeling the training set, not by considering the nature of the modeled objects.

Processing the training set produces a mean object, from which all objects in the training set may be seen to vary. Modeling allows us to define specific descriptive parameters for the objects, but because it also provides ranges for those parameters, it also allows us to evaluate whether similar objects not belonging to the original training set may be considered of the same class. It is therefore suitable for “objects” such as living creatures, which can vary widely within the strict confines of what precisely defines them as a species.

The normal distribution is a perfect stage for ASM, but this is not without limitation. Discretized behavior or nonlinear dependencies cannot be modeled with ASM without modifying them to deal with those nonlinearities.

Timothy F. Cootes and Christopher J. Taylor of the Division of Biomedical Engineering, University of Manchester, have led the field in these areas for the past few years providing methods for building shape models that we have also sought to emulate for this project.

2 DATA

2.1 Our data sets

We looked at images of the arrow mark and T mark which had been pre-annotated with two-dimensional landmark points.

These images were annotated manually using 7 and 8 landmark points lying on perimeter of the object respectively. For arrow mark and T mark each image was accompanied by a file containing the full set of landmark coordinates for that image. Landmark points were numbered consecutively in a single path. That file also contained information on the type of annotation points, such as whether points were internal or located on an outside border, and whether they lay on closed or open paths. The landmark points were given in two matrices, one for the x-coordinates, and one for the y-coordinates. No information about the paths was provided.

Our first task was bringing the data into a consistent format. For the arrow mark data sets, it was a matter of automating the loading procedure that would allow us to work with the data in MATLAB, which included manually stripping the files of any lines containing non-data (headings and other commented-out information). Finally, the arrow mark and T mark points were given in coordinates relative to the size of the images, so it was necessary to recalculate those coordinates, so that all coordinate systems corresponded.

2.2 Annotation

Annotation is the marking of images with landmark points to describe some object of interest contained in the image. These points may be set at key intersections and along object boundaries. Although the images we utilized were pre-annotated, it is worth noting that the placement of landmark points in images is not trivial. There are many techniques for annotating images, and among the many goals of model-based vision are accurate and robust methods for automating this vital step. Landmark points are commonly used for two- or three-dimensional images.

Their purpose is to describe the invariant shape of an object, one that should not change even after rotation, translation, or scaling. Each object in a set of images must be demarcated with the same number of landmark points.

In a data set of similar shapes, although object shapes themselves may vary from image to image, a landmark point in one image should correspond precisely to a landmark point in another image that fulfills the same structural description, such that all landmark points in all images correspond to one another consistently.

2.3 Paths

Our landmark points lay on artificial “paths” that that allowed us to better visualize contours when studying the shapes. Some of these paths were open, while other paths were closed. Designing one algorithm that could handle the different kinds of paths using the information provided would be a challenge. We could have ignored the paths, and simply plotted the unconnected landmark points, but in some cases, such as when we wanted to view extreme modes of variation, the result would be not more than a collection of random-looking dots, too jumbled to interpret meaningfully.

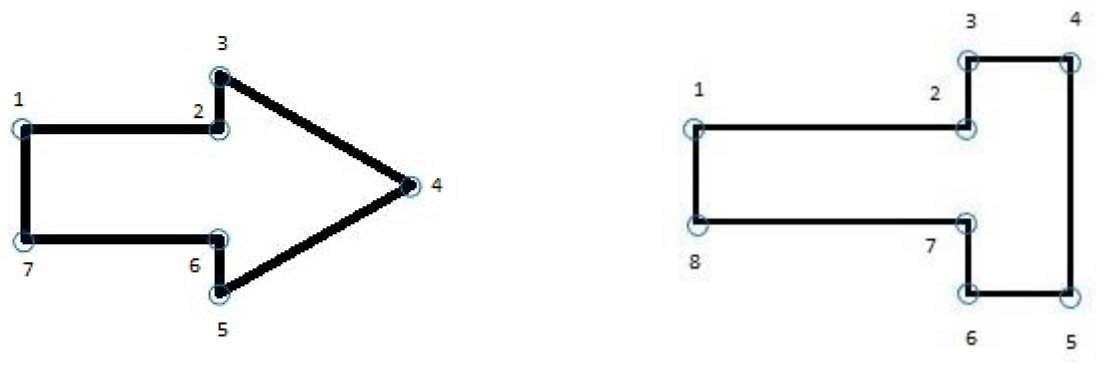


figure 2: Allocating landmark to images

3 ITERATIVE SHAPE ALIGNMENT

We built an initial shape model from the raw, unprocessed data of the arrow mark images. In both models, the two most-significant modes of variation were translations in the x and y directions. The third mode of variation was scale. To remove variations attributable to translation, rotation, and scale, it is generally desirable to align the shapes into a common coordinate frame.

Translation and rotation in face and cardiac cases were unlikely to be important factors for shape modeling, given that they were artifacts of the image-capturing process.

"Procrustes analysis" is a commonly used iterative method for aligning shapes so that each shape occupies a common coordinate system. This alignment procedure is applied to the vectors x obtained by concatenating x and y coordinates of landmark points

The following algorithm is used to align the set of shapes:

1. Each shape was rotated, scaled, and translated to align with the first shape in the set.

Repeated

2. The mean shape was calculated from the aligned shapes.
3. The orientation, scale and origin of the current mean was normalized to suitable defaults.
4. Every shape was realigned with the current mean.
5. **Until** the process converges.

Convergence is declared if the estimate of the mean does not change significantly after an iteration.

Here are some examples of the training set before align and after aligning:

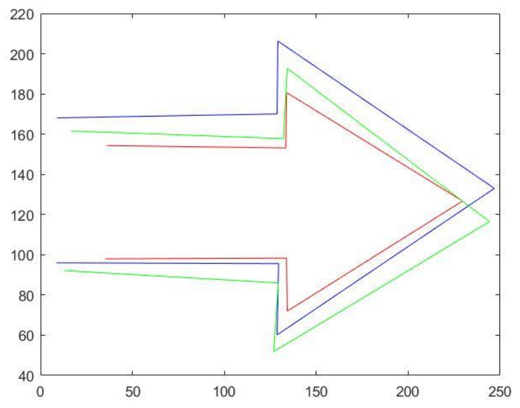


figure 3.1

In figure 3.1 the reference image is in red, the image before alignment is blue and after alignment is green. We can see that this image is closely aligned with the reference.

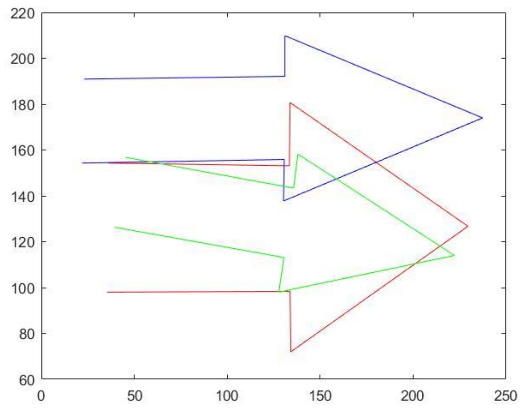


figure 3.2

In figure 3.2 we can see that the alignment of training set with reference is less. Sometimes these problems occur due to finding incorrect parameters.

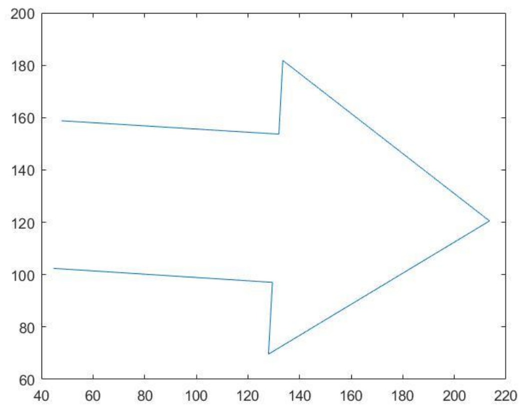


figure 3.3

In figure 3.3 the figure represents the orientation of mean of all shapes in the training set after the alignment after the first iteration during the process of convergence.

During the iterative alignment of the images in the training we were finally able to get able to make the process converge after 470 iterations with a mean square error of 0.0001. Here the error in the plot is the difference between the new mean and the previous mean. As the difference between both the mean is gradually reduced the process is said to be converged.

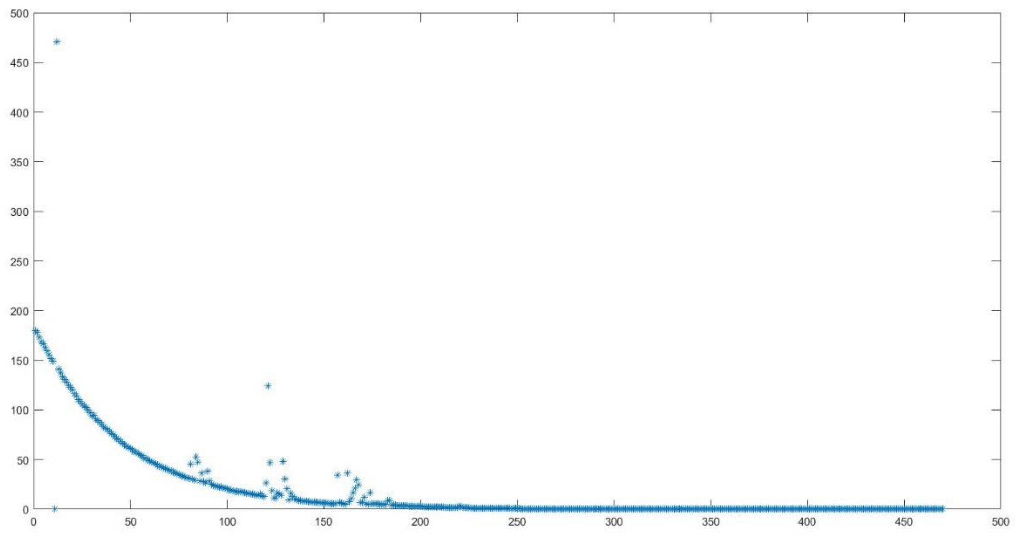


figure 3.4 Plot of error during the iterative alignment

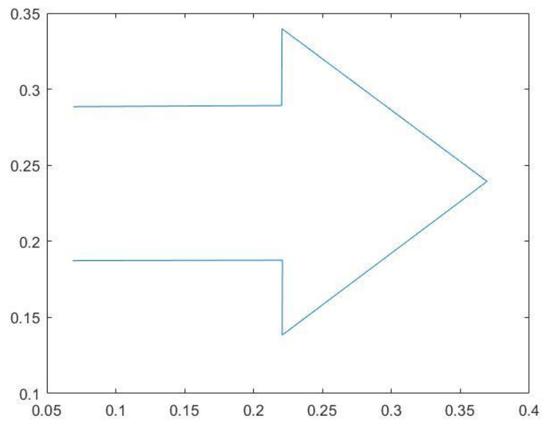


Figure 3.5 Mean shape after the final iteration

In figure 3.5 we can see the final mean shape after the final iteration when the alignment of all the shapes is done.

Example before and after aligning (T-shape):

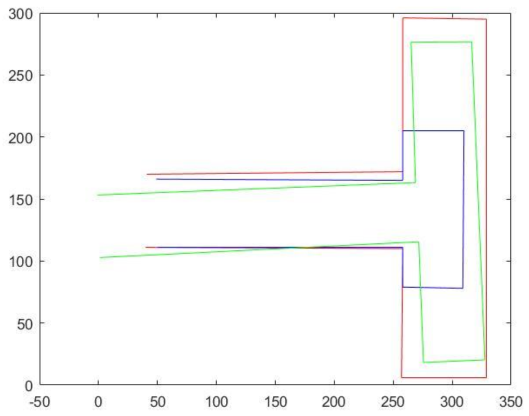


figure:3.6

In figure 3.6 the reference image is in red, the image before alignment is blue and after alignment is green. We can see that this image is nearly aligned with the reference.

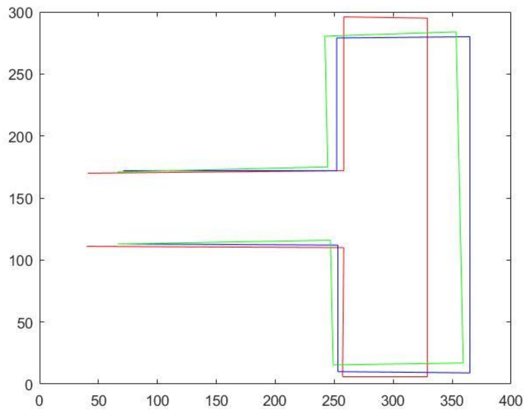


figure:3.7

In figure 3.7 the reference image is in red, the image before alignment is blue and after alignment is green. We can see that this image is closely aligned with the reference.

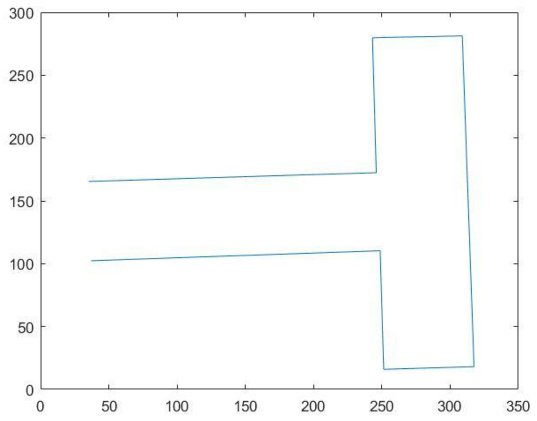


figure:3.8

In figure 3.8 it shows the first mean image of the images in the training set.

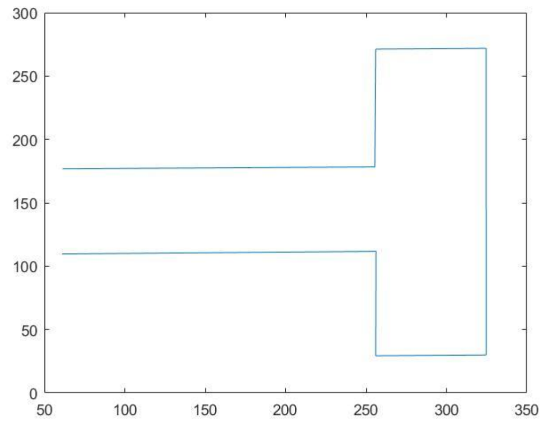


figure:3.9

In figure 3.9 shows the final mean shape of all the shapes present in the training set after all the iterations are done.

During the iterative alignment of the images in the training we were finally able to get able to make the process converge after 6 iterations with a mean square error of 0.001. As the difference between both the mean is gradually reduced the process is said to be converged.

4 BUILDING MODELS

After all the preprocessing was finished, we were left with a certain number of images, each one contributing A set of landmark points aligned to the mean shape. The term mean shape indicates the mean of all landmark points in an entire set of annotated images.

We used the sets of landmark points to build a statistical shape model. The procedures for building the shape model, the key method being principal component analysis (PCA). We begin this chapter with a short explanation of PCA, as it is obviously an important method used in building the models of statistical distribution.

4.1 Principal component analysis:

Principal component analysis (PCA) is a powerful statistical technique that is used for reducing the dimensionality of multi-dimensional data.

The basic purpose of performing PCA is to find an orthogonal coordinate system for our data cloud, in such a way that the greatest variance lies on the longest axis (first principal component), the second greatest variance lies on the second longest axis, and so on. For a data cloud with the flattened zeppelin shape, the first principal component would be the line through the middle along the length of it. We can then use the projection of the data cloud on the first axis as the initial approximation of the data cloud, and we can improve the approximation by adding more axes. The spread of the data along a certain axis provides information on how important that axis is for the approximation.

It is important to note that PCA uses a linear model to describe the data, so it can only handle linear behavior. Any nonlinear dependencies between the measurements (for example a quadratic dependency or discretized data) would present a problem.

A way of calculating principal components is by finding eigenvectors and eigenvalues of the covariance matrix.

4.2 The covariance matrix:

Given a set of N aligned shapes, the mean shape $\bar{x}$ (the center of the allowable shape domain), is calculated using:

$$\boldsymbol{\bar{x}} = \frac{\boldsymbol{1}}{\boldsymbol{N}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{x}_i$$

The principal axes of the shape fitted to the data can be calculated by applying a principal component analysis(PCA) to the data. Each axis gives a mode of variation, a way in which the landmark points tend to move together as the shape varies.

For each shape in the training set we calculate its deviation from the mean, dx_i, where

$$dx_i = x_i - \boldsymbol{\bar{x}}$$

2n × 2n covariance matrix S is calculated

$$S = \frac{\boldsymbol{1}}{\boldsymbol{N}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{dx}_i \, d\boldsymbol{x}_i^T$$

To achieve the task of PCA, one basically needs to find a coordinate system in which the covariance matrix for the data is diagonal (coordinates are not mutually dependent), and elements on the diagonal are in sorted order. This is equivalent to finding the eigenvectors of the covariance matrix and sorting them by their corresponding eigenvalues.

4.3 Eigenvectors and eigenvalues:

The principal axes give the modes of variation of the points of the shape, are described by p_k (k = 1, . . ., 2n), the unit eigenvectors of S such that

$$Sp_k = \lambda_k p_k \qquad (\lambda_k \text{ is the kth eigenvalue of S}, \lambda_k \geq \lambda_{k+1}),$$

$$p_k^T p_k = 1$$

It can be shown that the eigenvectors of the covariance matrix corresponding to the largest eigenvalues describe the longest axes of the domain, and thus the most significant modes of variation in the variables used to derive the covariance matrix. The variance explained by each eigenvector is equal to corresponding eigenvalue.

Most of the variation can usually be explained by a small number of modes, t. One method for calculating t is to choose the smallest number of modes such the sum of their variances explains a sufficiently large proportion of λ_T, the total variance of all the variables, where

$$\lambda_T = \sum_{k=1}^{2n} \lambda_k$$

Any point in our allowable shape domain can be reached by taking the mean and adding a linear combination of the eigenvectors. The kth eigenvector affects point L in the model by moving it along a vector parallel to (dx_{kl}, dy_{kl}), which is obtained from the lth pair of elements in p_k:

$$p_k^T = (dx_{k0}, dy_{k0}, \ldots, dx_{kl}, dy_{kl}, \ldots, dx_{kn-1}, dy_{kn-1})$$

Any shape in the training set can be approximated using the mean shape and a weighted sum of these deviations obtained from the first t modes:

$$x = \bar{x} + Pb, \text{ where}$$

$P = (p_1\, p_2 \ldots p_t)$ is the matrix of the first t eigenvectors, and $b = (b_1\, b_2 \ldots b_t)^T$ is a vector of weights.

The above equations allow us to generate new examples of the shapes by varying the parameters (b_k) within suitable limits, so the new shape will be similar to those in the training set. The parameters are linearly independent, though there may be nonlinear dependencies still present.

The limits for b_k are derived by examining the distributions of the parameter values required to generate the training set can be shown to be λ_k, suitable limits are typically of the order of

$$-3\sqrt{\lambda_k} \le b_k \le 3\sqrt{\lambda_k}$$

So in case of our test we choose t=2, for the arrow mark as the sum of their eigenvalues explains a sufficiently large proportion of λ_T.

Eigenvalues of the covariance matrix derived from a set of arrow shapes:

Eigenvalues	(λ_k/ λ_T)*100%
.0091	64.08%
.0044	30.98%

TABLE-1

As we get 95% of the total data variance in first two eigenvectors then for generation of any shape in the training shapes or new shape we will choose the first two values.

After applying $-3\sqrt{\lambda_k} \le b_k \le 3\sqrt{\lambda_k}$

We get the range for b1 is in between [-0.0136 ,0.0136]

We get the range for b2 is in between [-0.0066,0.0066]

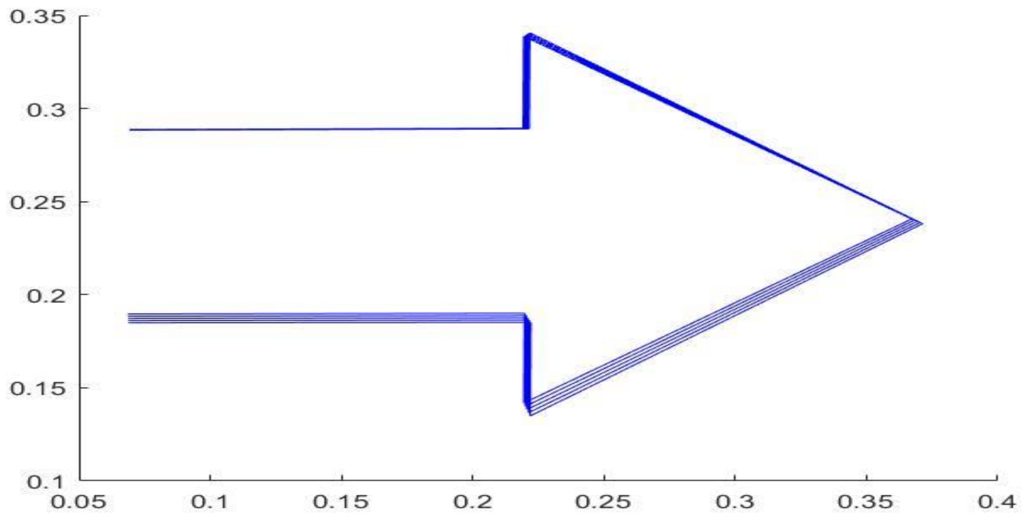


Figure 4.1 keeping b1 constant and varying b2

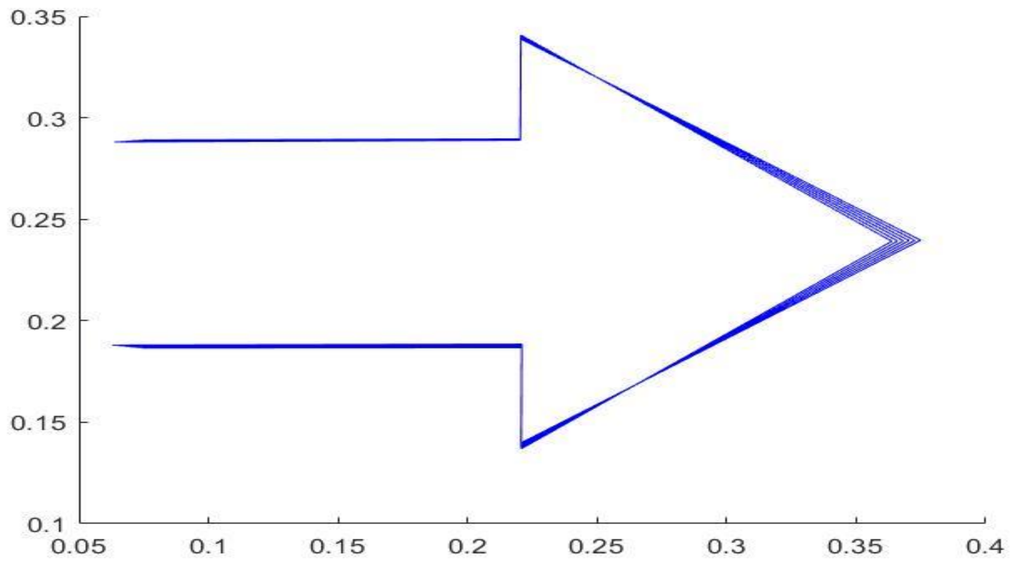


Figure 4.2 keeping b2 constant and varying b1

In the above figures we can see the change in the shape of the model as we vary the shape parameters b1 and b2. Keeping b1 constant and varying b2 the width of the model is changed. Similarly, keeping b2 constant and varying b2 the width of arrow is varied.

Similarly, in case of the T shape mark we choose t=2, for the as the sum of their eigenvalues explains a sufficiently large proportion of $\lambda_{T.}$

Eigenvalues of the covariance matrix derived from a set of arrow shapes:

Eigenvalues	$(\lambda_{k/}\ \lambda_T)$*100%
926.4	47.68%
838.4	43.15%

TABLE-2

As we get 90% of the total data variance in first two eigenvectors then for generation of any shape in the training shapes or new shape we will choose the first two values.

After applying $-2\sqrt{\lambda_k} \le b_k \le 2\sqrt{\lambda_k}$

We get the range for b1 is in between [-60.86 ,60.86]

We get the range for b2 is in between [-57.89,57.89]

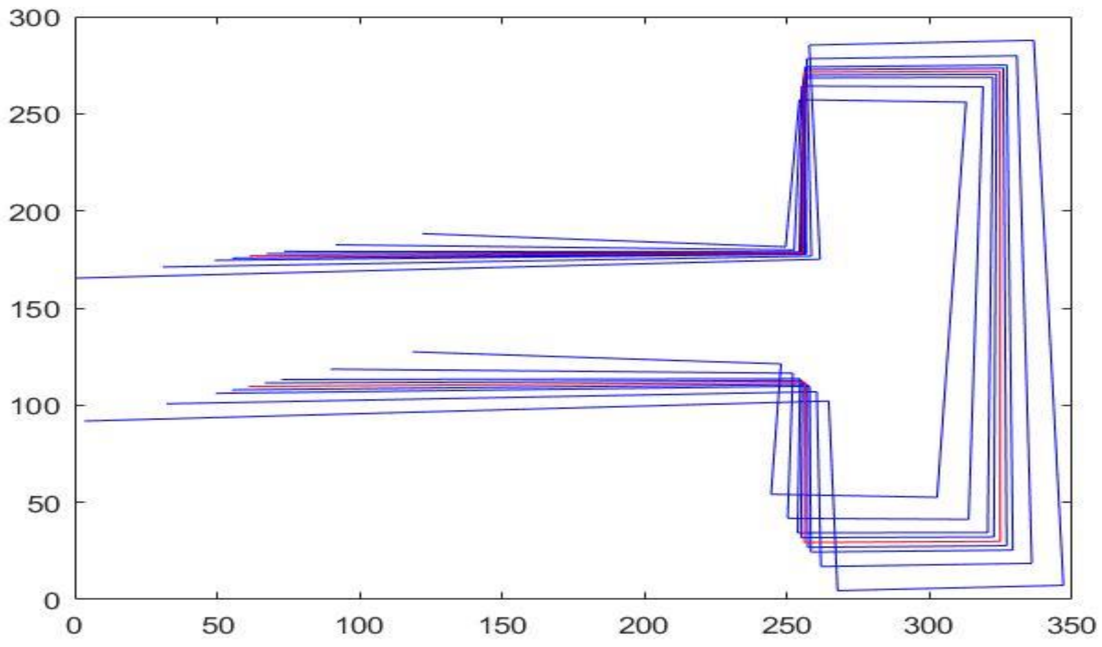


figure 4.3 keeping b1 constant and varying b2

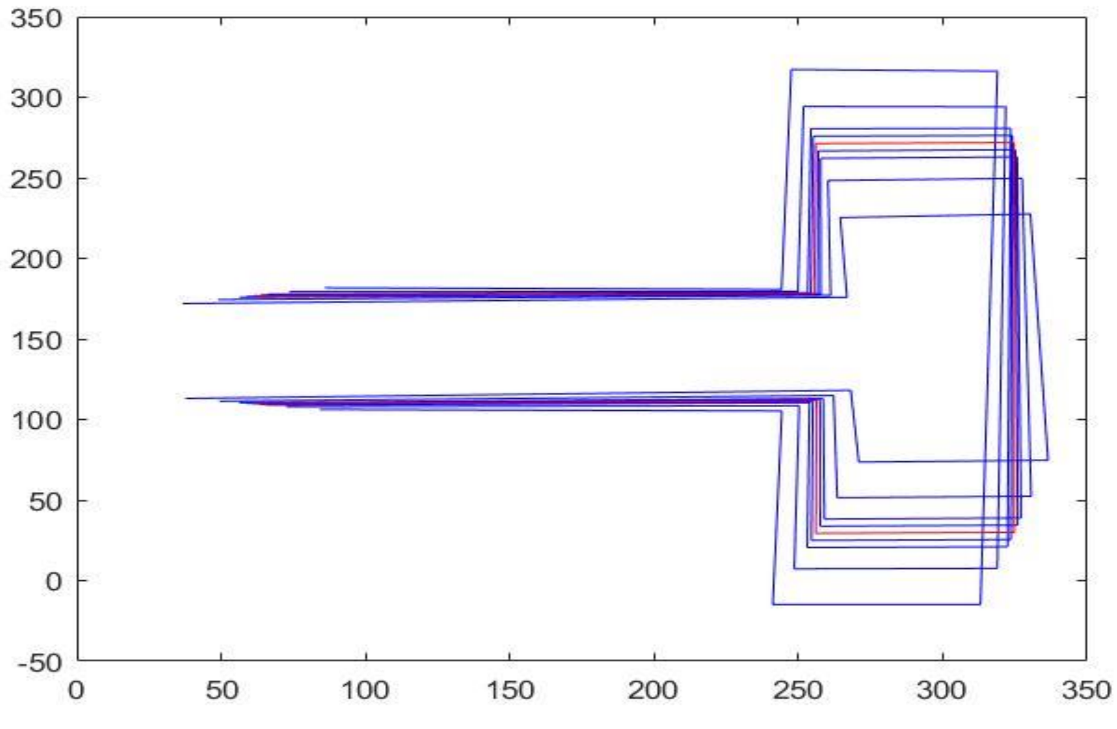


figure 4.4 keeping b2 constant and varying b1

In the above example we can see that by changing the shape parameter b1 and b2 the shape of the model image can be changed. So by varying the parameters we can get the images in the training sets as well as new images.

We can see that by keeping b1 constant and varying b2 the length and the width of the model is changed with a slight change in rotation. Similarly, by changing b1 and keeping b2 constant there is a significant change in the width of t shape that the length.

In the above part we achieved the first goal of our project that is to generate new, similar images to that of the data sets. Now moving on to the next part we have to design a GUI (Graphical User Interface) which uses the model to detect the shape of the object in an unknown image manually.

5 DESIGNING A GRAPHICAL USER INTERFACE

5.1 Introduction:

In order to detect the shape of the object in an unknown image manually, based on Active Shape Model, MATLAB-GUI is employed. Active Shape Model is analyzed by drawing the shape model and by scaling, rotating and translating the model to find the object in an unknown image manually using MATLAB; graphical user interface (GUI) is developed for interactivity. MATLAB is a high-performance language for technical computing that includes functions for numeric computation, algorithm prototyping, system simulation and operation. MATLAB also has strong data analysis tools which are useful in analysis of entered and calculated data.

In ASM finding the object, dynamic interface is developed that makes it easy to enter the required data and detecting the required object manually. The developed interface helps in understanding the ASM.

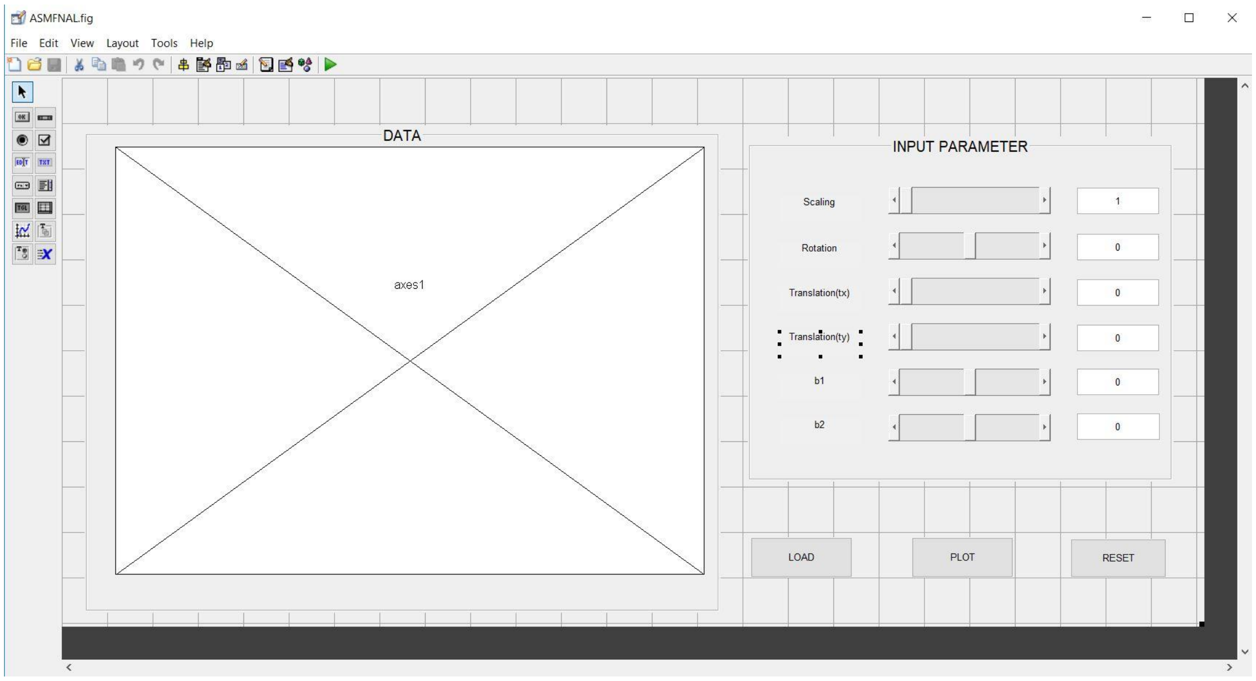
5.2 What is a GUI?

A graphical user interface (GUI) is a graphical display in one or more windows containing controls, called components that enable a user to perform interactive tasks. The user of the GUI does not have to create a script or type commands at the command line to accomplish the tasks. Unlike coding programs to accomplish tasks, the user of a GUI need not understand the details of how the tasks are performed.

GUI components can include menus, toolbars, push buttons, radio buttons, list boxes, and sliders-just to name a few. GUIs created using MATLAB tools can also perform any type of computation, read and write data files, communicate with other GUIs, and display data as tables or as plots.

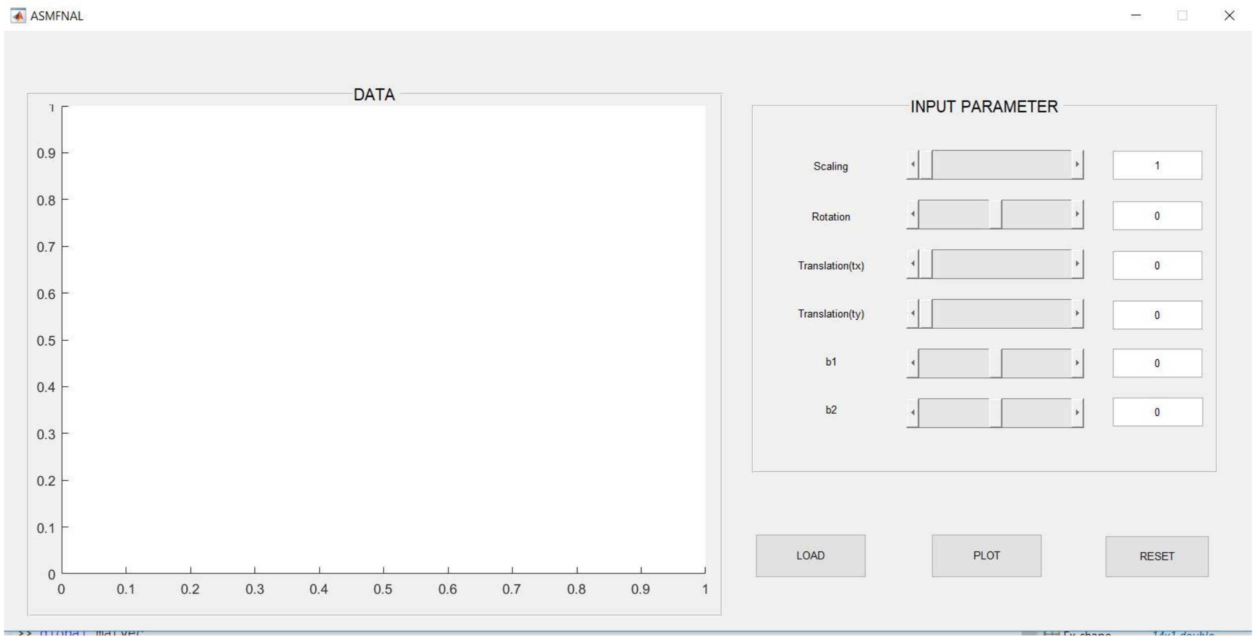
6 LAYOUT OF GRAPHICAL USER INTERFACE

Screen-1 shows layout for the GUI. Controls are placed according to requirement



Screen-1 Components (Controls) of the GUI

When the GUI was run for the first time it looked like as shown in Screen-2 below.

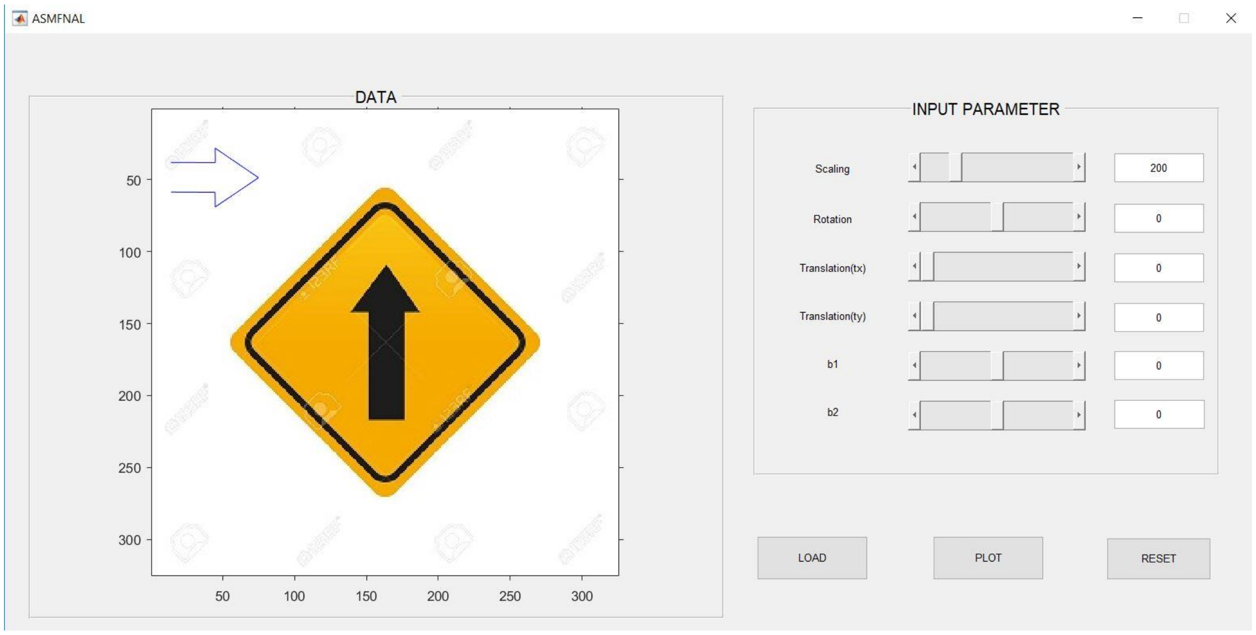


Screen-2 GUI appearance when executed

Three push buttons namely “LOAD”, “PLOT” and “RESET” were added to the GUI.

The “LOAD” push button, when clicked by mouse, Loads the workspace to the GUI for us. The “PLOT” plots the data on the axes. The “RESET” button resets all the changes done to the image and sets everything to the initial condition.

Screen-3 shows the plot when plot is pressed.



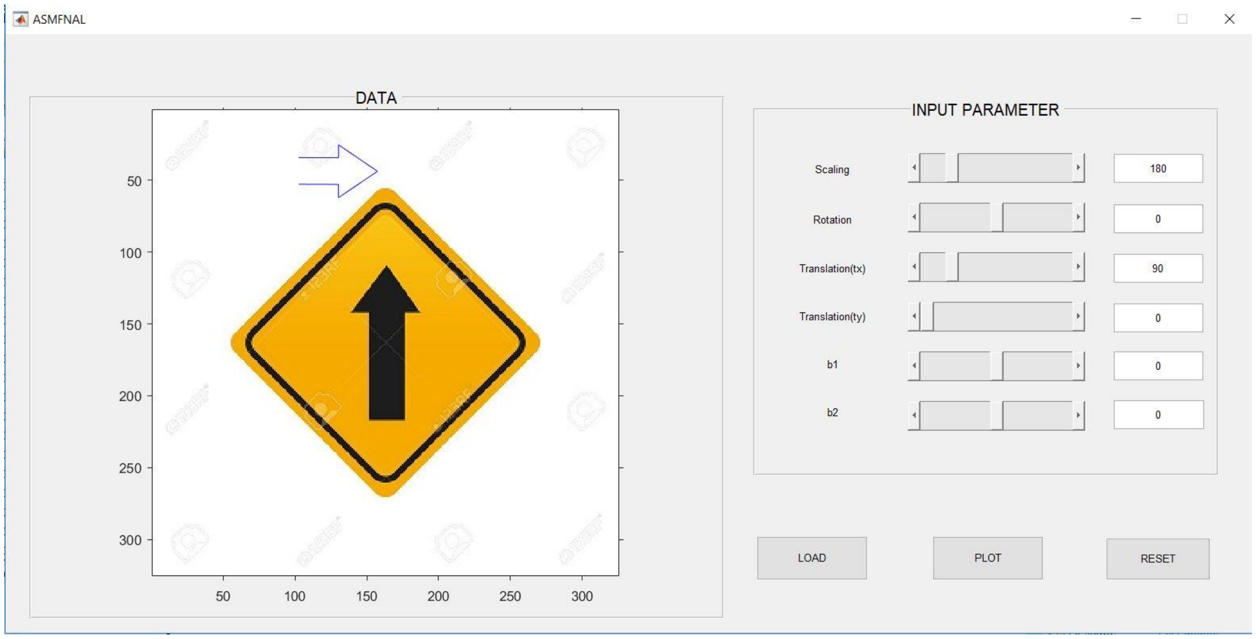
Screen-3 image is loaded in the GUI

There are six slider and six edit boxes in the GUI.

When the slider moves the corresponding value is shown in the right side of the slider. Similarly, when a value is typed the corresponding slider is moved to that value.

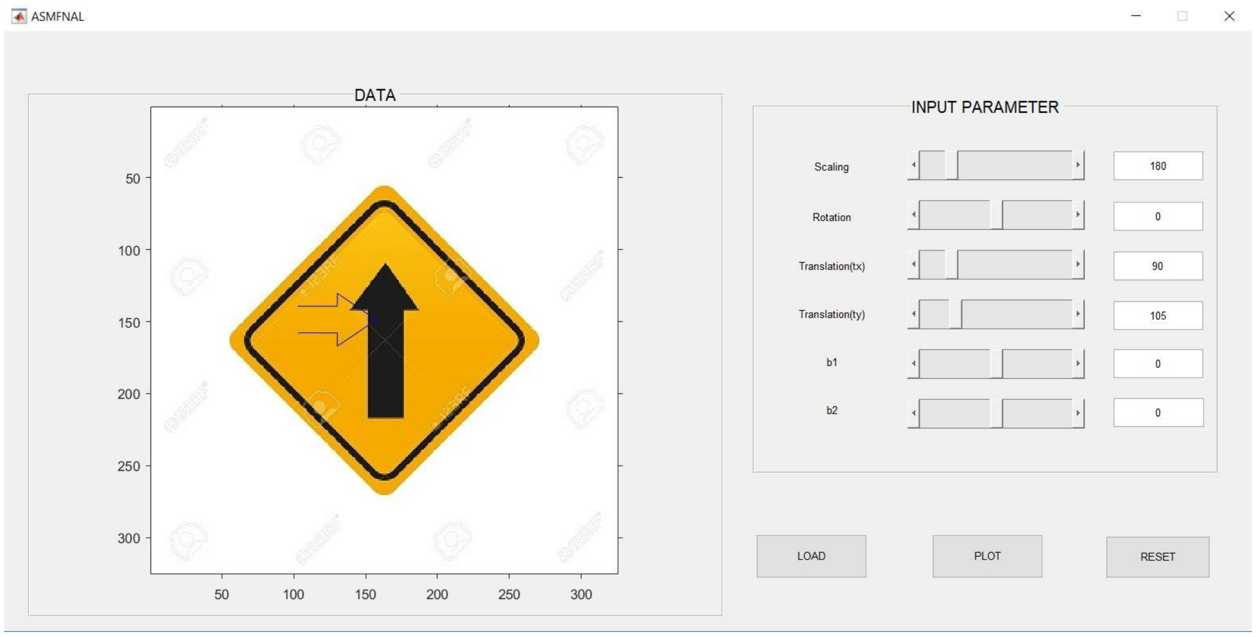
The six sliders perform six different function. The slider scaling, rotation and translation(tx,ty) perform operations as per the name suggest. Then b1 and b2 are used as shape parameters.

Screen-4,5,6 shows the performance of different sliders.



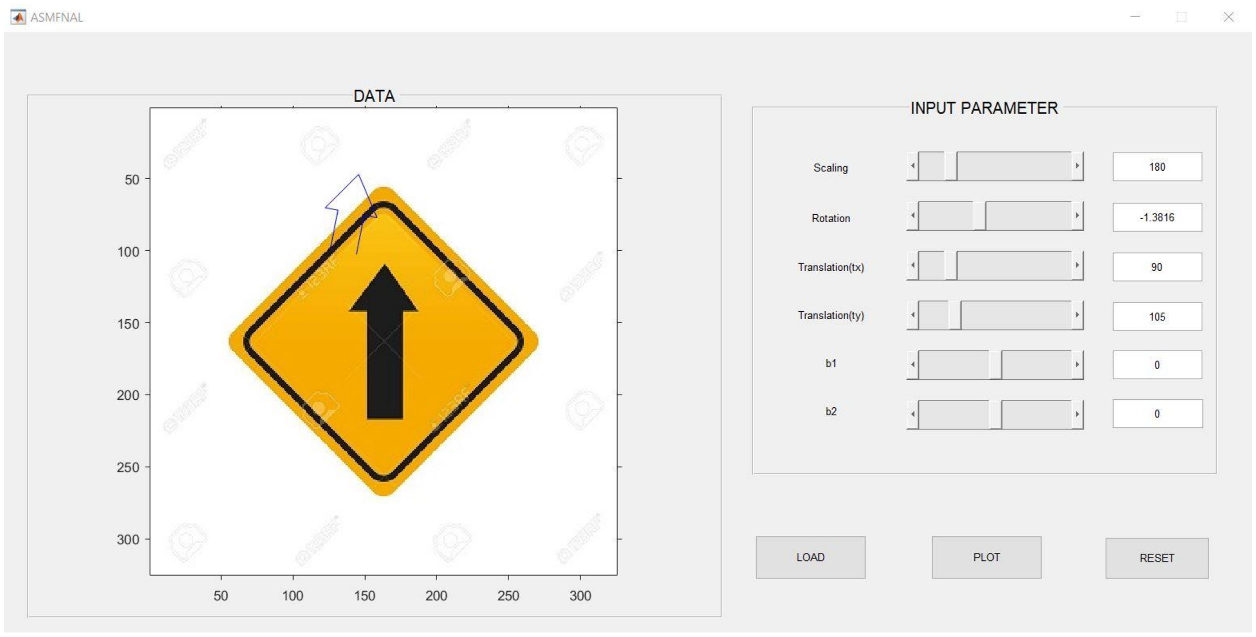
Scaling 180
Rotation 0
Translation(tx) 90
Translation(ty) 0
b1 0
b2 0

Screen-4 shows translation of model in x direction



Scaling 180
Rotation 0
Translation(tx) 90
Translation(ty) 105
b1 0
b2 0

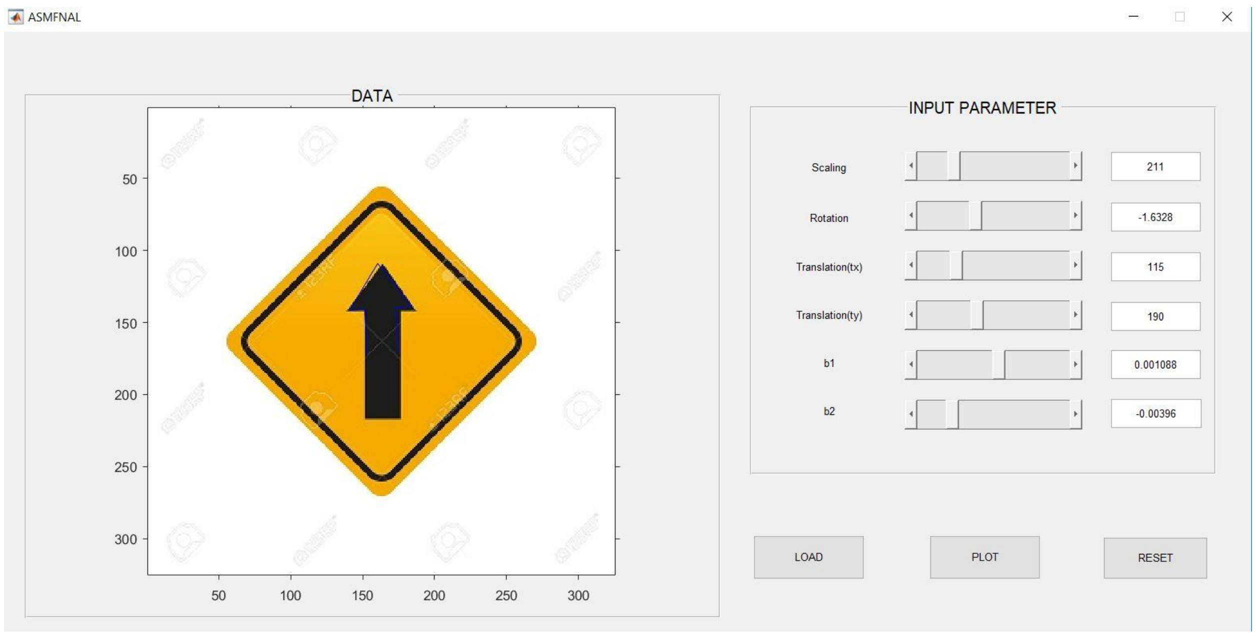
Screen-5 shows translation of model in y direction



Scaling 180
Rotation -1.3816
Translation(tx) 90
Translation(ty) 105
b1 0
b2 0

Screen-6 shows rotation of the model

After all the adjustment we can see the model is aligned in the screen-7.



Scaling 211
Rotation -1.6328
Translation(tx) 115
Translation(ty) 190
b1 0.001088
b2 -0.00396

Screen-7 shows the alignment of model on the object in the image

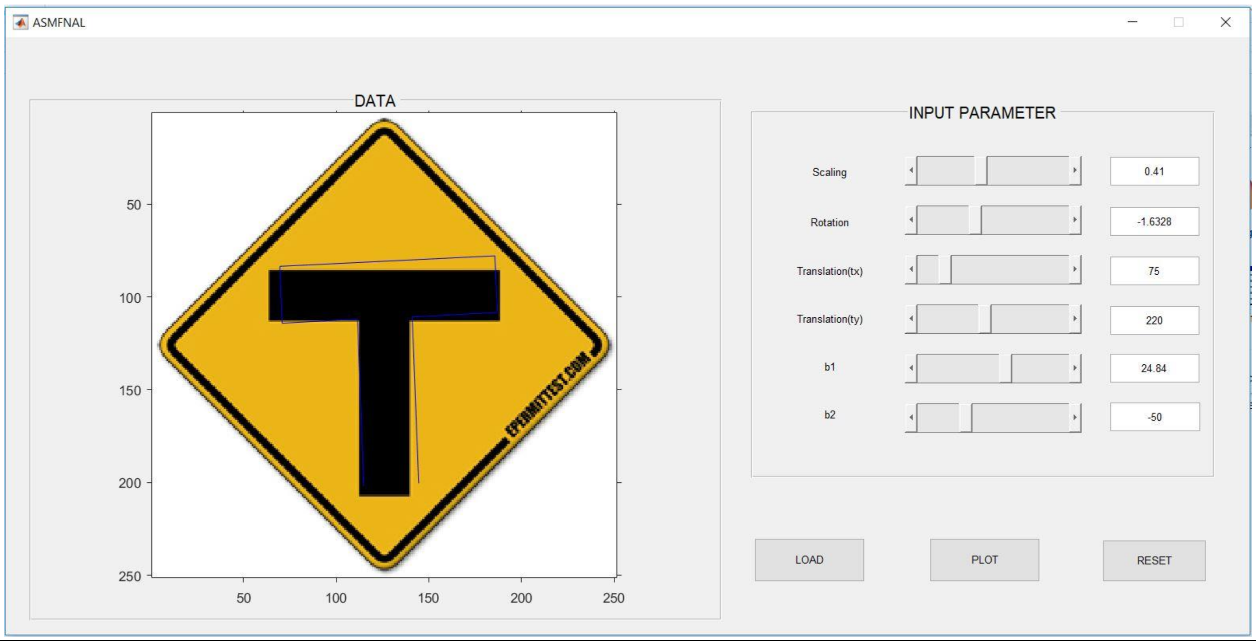
In the above GUI we achieved our second purpose that is to detect an object in an unknown image through the help of the image model.

After alignment of the model in the image the extracted ASM parameters are:

Scale=211, Rotation= -1.6328, Translation(tx)=115, Translation(ty)=190

b1=0.001088 and b2=-0.00396

In screen-7 it shows that the t shape model is fitted to the image



Scaling	0.41
Rotation	-1.6328
Translation(tx)	75
Translation(ty)	220
b1	24.84
b2	-50

Screen-8 shows the alignment of model on the object in the image

After alignment of the model in the image the extracted ASM parameters are:

Scale=0.41, Rotation= -1.6328, Translation(tx)=75, Translation(ty)=220

b1=24.84 and b2=-50

7 ADVANTAGES & DISADVANTAGES:

ADVANTAGES

- ASM is relatively fast.
- ASM is too simplistic, not robust when new images are introduced.
- It is widely applicable. The same algorithm can be applied to many different problems, merely by presenting different training examples.
- The models give a compact representation of allowable variation, but are specific enough not to allow arbitrary variation different from that seen in the training set.
- The system need make few prior assumptions about the nature of the objects being modelled, other than what it learns from the training set.

DISADVANTAGES

- The models described require a user to be able to mark 'landmark' points on each of a set of training images in such a way that each landmark represents a distinguishable point present on every example image.
- ASM does not incorporate all gray level information in parameters.
- May not converge to good solutions.

8 SUMMARY

The point distribution model concept has been developed by Cootes, Taylor *et al.* and became a standard in computer vision for the statistical study of shape and for segmentation of medical images where shape priors really help interpretation of noisy and low-contrasted pixels/voxels. The latter point leads to active shape models. Point distribution models rely on landmark points. A landmark is an annotating point posed by an anatomist onto a given locus for every shape instance across the training set population. For instance, the same landmark will designate the tip of the index finger in a training set of 2D hands outlines.

Principal component analysis (PCA), for instance, is a relevant tool for studying correlations of movement between groups of landmarks among the training set population. Typically, it might detect that all the landmarks located along the same finger move exactly together across the training set examples showing different finger spacing for a flat-posed hands collection. These landmarks are aligned using the generalized Procrustes analysis, which minimizes the least squared error between the points.

Through the use of eigenvalues, we can generate similar shape to that of the shapes in training model and also other different shapes. As the shape parameter changes we can identify different shapes of the same object.

9 CONCLUSION

We have described point distribution model (PDMs) statistical models of shape can be constructed from training set of correctly labeled images. A PDM represented an object as a set of labeled points, giving their mean positions and a small set of modes of variation, which describe how the object's shape can change. Applying limits to the parameters of the model enforces global shape constraints ensuring that any new example generated are similar to those in training set. Given a set of shape parameter, an instance of the model can be calculated rapidly.

Active shape model exploits the linear formulation of the PDM in an iterative search procedure capable of rapidly locating the modeled structures in noisy, cluttered images even if they are partially occluded. Object identification and location are robust because the models are specific in the sense that instance are constrained to be similar to those in the training set.

The extracted Active Shape Model parameters from the traffic sign boards can be further used as an input to high level computer vision which involves making sense of the extracted parameters to perform cognitive functions.

10 References

[1] "mathworks.com," MathWorks, [Online]. Available: https://in.mathworks.com/help/matlab/creating_guis/about-the-simple-guide-gui-example.html.

[2] "mathworks.com," MathWorks, [Online]. Available: https://in.mathworks.com/help/matlab/ref/fminsearch.html.

[3] T. Cootes, D. Cooper, C. Taylor and J. Graham, "Active shape Models-Their Training And Application," *Computer Vision and Image Understanding,* vol. 61, no. 1995, pp. 38-59, 1994.