Identification of human faces using data-driven segmentation, rule-based hypothesis formation, and iterative model-based hypothesis verification

Richard Weiss
Leslie Kitchen
Julianne Tuttle

COINS Technical Report 86-53

October 1986

ABSTRACT

We present a system for identifying human faces in gray-scale television imagery. The system uses a three-stage approach to image interpretation. The first stage is data-driven image segmentation; the second is rule-based hypothesis formation; the third is model-based hypothesis verification using an iterative relaxation process.

This research was supported in part by a grant from the A.C. Nielson Company.

1 INTRODUCTION

The goal of this study is to demonstrate the feasibility of recognizing images of faces in indoor scenes. Two parts to this problem are the location in an image of where a face is likely to be found and the matching of parts of that area with a model of an individual face. In our previous report [12] we demonstrated the feasibility of locating faces based on the detection of motion. In this report we demonstrate some success in recognizing faces from models which are constructed manually. Figures 1 - 5 show the images of faces which were used in this study. There are three images of each of three people. One of these for each person was used to construct the model. This problem is described in terms of matching data from the image with the model. The image data is organized into structures which we call tokens. Tokens can be lines, regions, dark circular areas surrounded by light annular areas, etc. In our system, we have used only lines and regions for the recognition process.

Recognizing a face in an image requires finding a correspondence between tokens in the image and a model of the appearance of a face. There are four parts to this problem:

1. Construction of the models
2. Extraction of tokens from the image.
3. Formation of initial hypotheses
4. Verification of hypotheses and model matching

The construction of the face models is based on the region and line tokens which can be extracted and their geometric relationships. The model description is designed for efficiency in conjunction with the matching algorithm and is described in Section 4.

The extraction of tokens from the image uses two algorithms which have been developed in the VISIONS system and are described in Section 2.

In the formation of the initial hypothesis, possible correspondences between image tokens and model tokens are obtained, based only on intrinsic properties. These possible correspondences are the hypotheses, each with a score measuring the extent to which the correspondence satisfies the required intrinsic properties. This is described in Section 3. In the last stage, we seek to find a subset of these hypotheses that satisfy the geometric relational constraints between the face tokens in the model. For this stage, we chose a relaxation process, in which the scores of these hypotheses are adjusted in accordance with the degree to which they satisfy the constraints. It can be thought of as an iterative process of hypothesis verification by making and testing predictions about other hypotheses. This process is described in section 4.

2 REGION AND LINE SEGMENTATIONS

Region and line tokens are two fundamental ways of organizing data in an image. In this section we describe the algorithms used for locating these tokens, and we present the results for the set of test images.

The straight-line algorithm was developed by Brian Burns and is described in [2]. The algorithm has four steps:

1. Group of pixels with similar gradient direction. These groups are called ***edge-support regions.***

2. Fit a plane to the intensity surface over an edge-support region. The line associated with an edge-support region is the intersection of this plane with a horizontal plane whose height is the mean intensity for the region.

3. Compute attributes for lines, such as length and contrast.

4. Filter lines to retain the lines with the highest contrast and longest length. In our experiments the 150 lines with highest contrast were selected, and of these the longest 60 were selected.

Figures 6 - 8 shows the lines which were extracted for each of the face images.

In the previous report, it was shown that the general location of moving objects could be detected in pairs of images. Based on this result, we have made the assumption that we have a bounding rectangle for the face in each image. Since for these experiments, we have assumed a bounding rectangle for the face location, we also filter on location. The result is that we have only the lines which intersect the rectangle. The filtered lines for some of the images are shown in Figure 9.

Segmentation of the images into meaningful region tokens was particularly difficult. Nevertheless, we obtained some useful results from the Nagin-Kohler algorithm [Kohler84].

1. The image is divided into sectors and for each sector a histogram of the intensity values is constructed.

2. Clusters are identified in each histogram.

3. Small clusters are added. This is based on examining adjacent sectors for clusters which may extend across sector boundaries but may not be large enough to be detected by themselves. Once this is done, then connected component regions are formed from the clusters.

4. Regions are merged across sector boundaries.

The parameters in the algorithm can be set in a way which depends only on general image characteristics. Although there are many parameters in the algorithm, these have been collapsed into a single parameter called sensitivity, which controls the number of regions produced. Empirically, this parameter has been quantized into five levels from very low sensitivity to very high sensitivity for a wide range of images [1]. Figures 10 - 12 show the regions which were extracted using very high sensitivity for each of the face images.

3 THE RULE-BASED OBJECT HYPOTHESIS SYSTEM

The rule-based object hypothesis system, Rulesys, generates initial hypotheses for image tokens such as lines and regions based on features computed from the image. An hypothesis is an assignment of a confidence value to the matching of an image token with a model token.

Our goal was to use these hypotheses to reduce the amount of computation for the relaxation algorithm which computes the most likely matches based on geometric relationships. The object hypothesis system only relies on intrinsic properties. The intrinsic properties are the feature values of regions and lines which are computed for an image and stored in a data base for use by the object hypothesis system. The algorithm for assigning a confidence value is based on rules.

A simple rule is a piecewise linear function which defines a score based on the value of a scalar feature as shown in Figure 13. This rule specifies two veto ranges where the function is negative and a positive voting range with zero in between. In general, an hypothesis about a match between an image token and a model token will require more than one feature. Thus, simple rules are combined into complex rules. They can be viewed as defining an area of feature space which represents a *vote* for an hypothesis, which could be the identity of an image token. For example, lines which mark the side of a face are expected to be vertical and relatively long compared with other lines in the face. The vertical line rule is an example of a rule which uses line orientation:

```
vertical-lines
 Feature: (line theta)
 Rule:    medium
 Veto:     (or low high)
```

The resulting value of the function is called the score and corresponds to the confidence in a particular hypothesis based on the value of the computed feature. Figure 14 shows the result of applying this rule to the lines in the images in Figure 9 . Features can provide both positive and negative evidence. It is may be possible to veto a particular hypothesis based on the value of a feature. For example, the vertical line rule above, if the orientation of a line were close to 0 or π, would veto the hypothesis that it is vertical. This rule can be combined into a complex rule for the side of the face so that if a line is vertical, the hypothesis that it is part of the side of the face is vetoed.

The features themselves can be the location of the token or some intrinsic property. In addition it is possible to combine features for regions with those for lines. The rules are divided into five categories:

- simple rules for regions

- simple rules for lines
- complex rules for regions
- complex rules for lines
- combined region and line rules

Simple Rules for Regions – intrinsic and location rules for regions
The intrinsic feature rules are based on a single feature which is computed directly from the image. The features used for regions are:

- size: the number of pixels in a region
- intensity: the average pixel intensity for a region
- compactness: the ratio of the boundary of the region to the area of the interior. The boundary pixels are all the pixels which have at least one of their four-connected neighbors outside the region, so the most compact shape is a square.

Simple rules can be defined in terms of either absolute or relative values. For example, when specifying the rule for finding vertical lines, one can assign the value 1.0 to lines whose orientation is between 0.4 and 0.6 times π radians. Relative rules are those which can be described without reference to absolute measurements. For example in the rule which is used to find regions which form the mouth based on intensity, the values can be specified with respect to the values for all the regions in the image. The top and bottom 5 % are discarded, then the remaining range is divided into five intervals whose endpoints are designated as vlow, low, medium, high, and vhigh. Thus, the rule is described as follows:

```
mouth-regions-intensity
Feature: (intensity mu)
Rule:    low
Veto:    high
```

The location rules are based on the minimum and maximum extents of the region with respect to the minimum and maximum extents of the area corresponding to the face. The basic assumption is that the minimum and maximum x- and y-coordinates for the face are known. We have had some success with using hierarchically smoothed difference images to determine the extents of the face. For example in the following rule, the values for *mask-top-third and *mask-y-midpoint are computed from the minimum and maximum y-coordinates for the face.

```
miny-in-top-third
(:feat-vals *mask-top-third *mask-y-midpoint *mask-y-midpoint)
(:feature extents miny)
(:score-vals 100.0          0.0              -20.0)
(:type . :primitive)
```

Simple Rules for Lines

The intrinsic features used for lines are:

- contrast: the slope of the plane which approximates the intensity surface
- length: the distance between the two endpoints of the line
- orientation: the angle of the line measured from the x-axis

Since the scale of the image is already known, medium-length lines can be specified on an absolute scale, as in the following rule:

```
medium-length-lines
(:feat-vals 7.0 10.0 26.0 30.0)
(:feature line length)
(:score-vals -20.0 100.0 100.0 -20.0)
(:type . :lprimitive)
```

The location features used for lines are: x- and y-coordinate of the midpoint of the line. Example:

```
top-of-head-lines-y
(:feat-vals  *mask-top-third  *mask-y-midpoint *mask-y-midpoint)
(:feature mid-point-y)
(:score-vals 100.0          0.0              -20.0)
(:type . :lprimitive)
```

Complex Rules

Complex rules are formed by combining rules, and this may be done in an hierarchical way starting with combinations of simple rules. We used two principle methods for combining the scores from different rules. The first is a simple weighted average of the scores, where the weights are specified in the rule according to the importance of each of the component rules. The second is a weighted average with veto, so that the resulting score is a veto if any one of the component scores is a veto. For example, the following rule combines the rules long-length-lines and vertical-lines with weights 2 and 3 respectively:

```
side-of-face-lines
(:rule-list long-length-lines vertical-lines)
(:score-form weighted-average-w-veto long-length-lines 2
                                     vertical-lines 3)
(:type . :lcomplex)
```

Figure 16 shows the results of this complex rule.

Combined Region and Line Intersection Rules

With the current system it is possible to select regions based on whether they intersect a set of lines or not. For example, when attempting to identify eyes, it is useful to look for dark regions of a particular size, but it is also important to find horizontal lines which bound the region.

The set of rules which was used for generating hypotheses for regions and lines is contained in Appendix A.

4 MODEL MATCHING

4.1 Matching Technique

As mentioned in the Introduction, recognizing a face in an image requires finding a correspondence between features in the image (here lines and regions) and a model of the appearance of a face. This has two stages. In the first, possible correspondences between image features and model parts are obtained, based only on intrinsic properties. These possible correspondences are the hypotheses, each with a score measuring the extent to which the correspondence satisfies the required constraints on intrinsic properties. In the second stage, we seek to find a subset of these hypotheses that satisfy the geometric relational constraints between the face parts in the model. For this stage, we chose a relaxation process, in which the scores of these hypotheses are adjusted in accordance with the degree to which they satisfy the constraints. It can be thought of as an iterative process of hypothesis verification by making and testing predictions about other hypotheses.

Thus there are three components to model matching:

(1) A model representing both the intrinsic and relational constraints on the parts of a face.

(2) A process for forming initial hypotheses, based on intrinsic properties only. (This is what is done by the *Rulesys*, as described earlier.)

(3) A (relaxation) process for adjusting the scores of these hypotheses to reflect the degree to which they satisfy the geometric relational constraints.

The next two sections will present (1) and (3). As mentioned above, (2) was treated earlier.

4.2 Model Representation

The representation we use for models is a hybrid one: some information is stored explicitly, some implicitly. For each face part, the type of image feature it corresponds to is stored explicitly: in the current implementation this is either *region* or *line*. Further details of its intrinsic properties are stored implicitly, by giving the name of a Rulesys rule (or in general a function) that will return a list of hypotheses for possible occurrences of this face part in an image. Thus the specifics of these intrinsic properties are stored elsewhere, usually in the Rulesys knowledge base.

In the current system, the only relational constraints are those on relative position in the image. Instead of storing these many relative-position constraints explicitly, we keep only the position of each face part with respect to a face-model coordinate system (with origin midway between the eyes). The actual relative positions of the face parts are implicit in this representation, and can be computed as needed during the matching process. However, in general there are many relative-position constraints, $n(n-1)$ of them for a model with n parts. Checking them all would be computationally too expensive, and inefficient since most of them are redundant. So, while the relative positions themselves are stored implicitly, we do keep explicitly a partial list of other face parts with respect to which these positional constraints are actually checked. Put another way, the confidence of an hypothesis is updated by making predictions about the positions of other face-part hypotheses, and checking how well these predictions are verified. Instead of making predictions about the positions of all other face parts, the system makes predictions only about a subset of the other face parts. This is sufficient, and computationally far less expensive. Also, if these other parts are chosen to be nearby the given part, then the matching process will be more robust, in that it will be more tolerant of systematic and random distortions of the face in an image.

The actual Lisp specification of a face model can be seen in Figures 37, 38, and 39, which show the models used in the experiments. The face parts used were those that apeared as reasonably well defined image features (regions or lines). Six items are given for each face part:

(1) A symbolic name, used for referring to this model part from elsewhere in the model, and for labelling displays of matching results.

(2) What kind of image feature the face part appears as, currently either *region* or *line*.

(3) & (4) Respectively the X and Y coordinates of the position of this face part with respect to the face coordinate frame. (These coordinates were obtained by making measurements on images of the three subjects.)

(5) The function or Rulesys rule used to generate a list of initial hypotheses for this face part.

(6) The list of other face parts (referred to by their symbolic names) for which the relative-position constraints are checked.

4.3 Relaxation

As mentioned above, relaxation is essentially an iterative process of updating the confidence in each hypothesis based on how well predictions made from that hypothesis are verified amongst the other hypotheses. Thus a collection of hypotheses that actually make up a face will mutually support each other (in that predictions made from any one will be verified amongst the others). Hypotheses that are not part of face interpretation will not have their predictions verified, and so their confidences will be decreased. Predictions are made only for a subset of the positional constraints on a given hypothesis; this imposes an "horizon" on the prediction and verification possible on any one pass of the relaxation updating. So this process is iterated several times, in order that the effects of more distant hypotheses may be propagated, to produce a globally consistent interpretation.

Put another way, *all* hypotheses in a complete face interpretation should (ideally) be in the correct relative positions with respect to each other. On any one iteration, only a chosen subset of these relative-position constraints are checked. The repetition of the updating ensures that all the relative positions are checked. For example, suppose that a left-eye hypothesis fails to find the (supporting) left side of the face. Then its confidence will be decreased by the updating. On the next pass, the corresponding right-eye hypothesis, which may previously have received strong support from this left-eye hypothesis, will also have its confidence decreased. Thus the failure to find a corresponding left side of a face will, over the course of two iterations, affect the confidence in a right-eye hypothesis through the linkages between hypotheses, even though there are no direct constraints expressed between right eye and left side of face.

The reasons for choosing a relaxation process were discussed in a previous report [12]. Note, however, that since that report was written, the relaxation procedure has been entirely recoded and considerable improvements made. This has arisen largely from the need to interface the relaxation procedure effectively with the Rulesys, and from re-interpretation of relaxation as an iterative, local process of top-down hypothesis verification, rather than as a merely bottom-up process of adjusting confidences of match pairings. This re-interpretation has been fruitful, in that it has lead to a new understanding of how the process works, suggesting practical improvements in its implementation, and also making clearer the connections between relaxation and higher-level processes of intepretation and control.

The basic relaxation updating formula used can be described as follows. Let F be the set of face parts in our model, I be the set of image features (regions and lines), let H_{fi} be the hypothesis that matches face part $f \in F$ with image feature $i \in I$, and let $C_{fi}^{(k)}$ be our confidence in H_{fi} on the kth iteration of the relaxation process. (These confidence values will be real numbers in the range $[0,1]$.) For $k=0$, we take initial confidences derived from a linear rescaling of the Rulesys scores to the range $[0,1]$. If no such hypothesis exists, this confidence is taken to be zero. The updating rule, in its simplest form is:

$$C_{fi}^{(k+1)} = \min\left(C_{fi}^{(k)}, U\right) \tag{1}$$

where

$$U = \frac{1}{|\mathrm{nbd}(f)|} \sum_{g \in \mathrm{nbd}(f)} \max_{j \in I} \min\left(C_{fi}^{(k)}, C_{gj}^{(k)}, D\right) \tag{2}$$

and

$$D = \frac{1}{1 + \tau \|(\mathbf{p}_i + \sigma \mathbf{q}_g - \sigma \mathbf{q}_f) - \mathbf{p}_j\|_2^2} \tag{3}$$

The set $\mathrm{nbd}(f)$, for $f \in F$, the ***neighborhood*** of f, is the set of all face parts for which a prediction is made based on an hypothesis for f; $\mathbf{p}_i$, for $i \in I$, is the coordinate vector of image feature i in the image; $\mathbf{q}_f$, for $f \in F$, is similarly the coordinate vector of face part f in the face-frame coordinate system; σ is the scale factor that converts face-model distances into image distances. Thus, given H_{fi}, the hypothesis that image feature i corresponds to face part f, $(\mathbf{p}_i + \sigma\mathbf{q}_g - \sigma\mathbf{q}_f)$ is the predicted image position of face part g. So the $\|\ldots\|_2$ in (3) is the image distance by which $\mathbf{p}_j$, the position of image feature j, differs from the predicted position of face part g. Thus D measures, on a scale from 0 to 1, the degree to which the hypothesis H_{gj} fits the prediction of where g should be, under the hypothesis H_{fi}. In (3), τ is the tolerance factor that controls how much being in the wrong spatial position reduces the goodness of the match. The other two terms in (2), $C_{fi}^{(k)}$ and $C_{gj}^{(k)}$, are respectively the confidences of H_{fi} and H_{gj} from the previous iteration. In particular, $C_{gj}^{(k)}$ will normally incorporate information that has propagated in from further afield than nbd(f).

The evaluation of the updating formula requires several nested iterations, which could be computationally expensive. However, as mentioned in the previous report [12], advantage can be taken of the regular structure of this formula to make a number of short-cuts in its evaluation. Choose two critical confidence values, α and β, with the significance that confidence values less than α can be effectively treated as 0.0; similarly, confidence values greater than β can be effectively treated as 1.0. It is not possible to compute the exact value of the updating formula without complete evaluation. However, it is often possible to determine early in the computation that the resulting value must be less than α, or greater than β. (For example, in the evaluation of a minimum, if any term is found to be less than α, we can conclude that the ultimate minimum must also be less than α.) In such cases the computation can be cut off early with the result 0.0 or 1.0, as appropriate, without any significant efect on the resulting confidence values. A simple application of this idea produced a 30-fold speed-up in the updating process. It is related to the notion of alpha-beta cut-off in minimax game-tree searching (from which source it was adapted [10]).

These cut-off values α and β were used in the previous report [12]. However, this idea has been further exploited to improve the speed and reliability of the updating process. In the updating rule, as formerly described, not only was it necessary to define and search a "neighborhood" for each face part, but it was also necessary to define and search a "neighborhood" around each image feature. The initial computation of these neighborhoods in the image data was a computationally expensive process, and the correct definition of these neighborhoods was crucial. If the neighborhoods were too small, the relaxation matching might produce incorrect answers, because it might not be able to find some necessary support for an hypothesis. If the neighborhoods were too large, correctness would not be affected, but searching such excessively large neighborhoods would increase computation time inordinately. Furthermore, it was not possible to integrate such construction of image neighborhoods with use of the Rulesys in a natural way, since the image description is not directly accessible in its entirety through the Rulesys, but only sets of object hypotheses.

We have therefore made use of the lower cut-off value α to do away with the need for such image neighborhoods. In updating an hypothesis about a particular face part, we make predictions about neighboring face parts (neighboring in the face model), and check how well they are verified by searching for a best matching hypothesis to each prediction. A simple but expensive method would be, in finding a best match, to examine all hypotheses known to the system. The use of image neighborhoods was an attempt to reduce this cost. However, it is possible to do far better.

Every match to a prediction is evaluated in two respects: in its positional correctness and its confidence from the previous iteration. We are interested only in matches with confidences greater than α, so a match can fail in only two ways: its confidence from the previous iteration is less than α, or its positional correctness is less than α. Now, hypotheses with confidence less than α are automatically dropped. Thus the only significant matches are those with positional correctness greater than α. However, from the symmetry and monotonicity of the positional correctness term, it can be seen that all such matches must lie within a certain radius of the predicted position, a radius that depends simply on α, σ, and τ.

By using a special data structure that permits the rapid retrieval of hypotheses within a certain radius of a given position, we can make this verification process quite direct. This data structure and the tuning of it to give optimal performance are discussed elsewhere [8]. To further speed up the verification, separate data structures were maintained for the different kinds of hypotheses (that is, hypotheses for the different face parts), but the usefulness of this refinement may depend on the density of hypotheses in the image.

4.4 Experimental Results

We conducted a modest experiment to test the effectiveness of this technique for distinguishing human faces. From our database of images, we chosen three images each of three subjects, "Hiro", "Les", and "Mike", nine images altogether. Region and line segmentations were computed for each image, and stored in the image-feature database. We had previously developed face models for each of the three subjects, and Rulesys rules for forming initial hypotheses for the component parts of these models.

Each image (or rather the segmentations derived from it) was matched in turn against the three face models, using four iterations of the relaxation process, and an overall match-merit measure was computed for each match. For each image, the model which matched it best (according to the match merit) was chosen as the identity of that image. In order for a model to match well, there must be good matches for every one of its parts. So for each face part, we select the hypothesis (after matching) of highest confidence. The overall match merit is a function of these best-match confidences for all face parts. Two functions were used: the minimum (under the view that a match is only as good as its weakest support), or the average (under the view that the overall match merit should depend continuously on the best-match confidences).

The results of the identification experiments can be seen in the match scores in Tables 1 and 2, and in the confusion matrices in Tables 3 and 4. Using the average confidence, six out of the nine images were identified correctly. This is double the rate that would be expected by chance. Furthermore, the errors are not random, but are very systematic: All the images of "Hiro" and "Mike" were identified correctly, while all the "Les" images were misidentified as "Mike". This indicates that the errors arise from some underlying similarity between the models for "Les" and for "Mike", and that this similarity is asymmetric. The "Mike" model is more tolerant, in that not only does it match well to images of "Mike", but it also matches well to images of "Les"—better than does the "Les" model itself. This suggests that at this stage the accuracy of identification is not limited by the image features used, but by the discriminating power of the models. The identification could therefore be improved by adjusting the "Mike" and "Les" models to be more discriminating, by a slightly different choice of face parts and constraining relations. The results obtained by using the minimum confidence are similar, five out of nine correct, and again demonstrate confusion between the "Les" and "Mike" models.

Figures 17, 18, 19, 20, and 21 show the progress (over several iterations) of the relaxation process on a typical matching problem, that of matching the "Les" model against one of the "Les" images. (Here the image and model agree.) In each figure, the labels indicate the image feature that matches best to each face part, along with the confidence of the match hypothesis. Before the relaxation (Figure 17), none of the best matches are correct (or if they are correct, as in some other cases, it is largely accidental). This is understandable, since the purpose of the initial hypothesis formation is only to find a conservative set of candidate matches. After a single iteration of the relaxation updating (Figure 18), almost all (and sometimes all) the best matches have reached their final assignment. The following iterations (Figures 19, 20, 21) make only a few small adjustments in the best matches, usually just cycling between two competing best-match hypotheses that are nearly tied in their confidence scores. (This can be seen in the choice of matches for the *mouth-line* and *mouth-region*.) The numeric values of the confidences change noticeably, and continue changing (it would take many iterations for them to converge), but the actual rank ordering of the matches changes hardly at all—aside from the alternation of closely competing hypotheses just mentioned. Similar series of results can be seen in Figures 22–26, 32–36, and 27–31.

The "correctness" of the match of image data against the true corresponding model can only be evaluated by human judgement. In most cases, the image feature (region or line) chosen by the relaxation process as the best match to a particular face part would be the best match chosen by a human judge, or very close to it. (This can be seen in Figure 21, and also in the other examples shown in Figures 26, 36 and 31.) Failures can usually be attributed to poor localization of the image features by the initial segmentation and token-formation processes. This is particularly true of long line features, such as the sides of the face. (The localization of large region features, such as hair regions, was so poor that such features could not be used at all for model construction and matching.) Also, the spectacles worn by the subject "Mike" made it very difficult to detect his eyes.

As can be seen in Figure 36, his eyes are mislocated at the tops of the spectacle frames. The effect of this mistake propagates, and causes a systematic upward mislocation of other face features (mouth, chin, hair line).

The computation time for the matching, including the formation of initial hypotheses and four iterations of relaxation matching, typically took 30 to 50 minutes of CPU time on a VAX 750. Roughly a third of the time was taken up by the initial hypothesis formation (Rulesys), another third by the first iteration of relaxation (in which there are many hypotheses to consider), and the remaining time by the subsequent iterations of relaxation (which become quicker as the number of remaining, viable hypotheses decreases). The computation was done in interpreted Lisp (CLisp), written under experimental conditions with no particular care taken regarding code efficiency, and with a considerable amount of additional tracing and instrumentation code. Given this, it is not unreasonable to expect that, in practice, this computation time could be reduced to several minutes by careful programming in a language such as C.

5 CONCLUSIONS

We have shown how human faces can be recognized and identified in images with a moderate degree of success, under somewhat restricted viewing conditions (frontal view, with known orientation and range). Our technique is based on an initial, general-purpose segmentation of the image into regions and straight lines, followed by rule-based formation of initial hypotheses about the correspondences of parts in a face model to image features, and a relaxation process to find sets of hypotheses that satisfy the relative positional constraints of the face parts.

It seems that the accuracy of identification could best be improved by refinements in two parts of the system. First, the structure of the models should be modified to make them more discriminating between the individuals to be identified. Second, the initial segmentation processes should be refined in order to better localize the extracted image features. This would require some kind of perceptual-organization process, and in the long run perhaps feedback from the higher-level model matching.

An alternative to better segmentation would be to adjust the match criteria so as to be more tuned to the actual localization of image features. For example, the localization of a line feature is much better in the direction perpendicular to the line than in the direction along the line. The rating of positional discrepancy could be modified to give more weight to the perpendicular discrepancy, and be more tolerant of discrepancies along the direction of the line. This was in fact tried, but gave only slightly improved results at a greatly increased computational cost. A feasible compromise would be to apply the simple discrepancy measure in early iterations of matching, when there are many obviously poor matches to be rejected, and then to apply some more sophisticated measure in later iterations, when it remains only to refine the confidences of a few retained matches. This variant, however, has not been tried at this point.

References

[1] J.R. Beveridge, R. Kohler, A.R. Hanson, and E.R. Riseman, "Segmenting Images Using Localized Histograms", to appear.

[2] J. B. Burns, E. M. Riseman and A. R. Hanson, "Extracting Straight Lines", *Proceedings of the International Conference on Pattern Recognition*, Montreal, July–August 1984, pp. 482-485.

[3] M. A. Fischler and R. A. Eschlager, "The representation and matching of pictorial structures", *IEEE Transactions on Computers*, vol. C-22, no. 1, January 1973, pp. 67–92.

[4] A. J. Goldstein, L. D. Harmon and A. B. Lesk, "Identification of human faces", *Proceedings of the IEEE*, vol. 59, May 1971, pp. 748–760.

[5] L. D. Harmon, M. K. Khan, R. Lasch and P. F. Ramig, "Machine identification of human faces", *Pattern Recognition*, vol. 13, no. 2, 1981, pp. 97–110.

[6] M. D. Kelly, "Edge detection in pictures by computer using planning", pp. 397–409 in *Machine Intelligence 6*, B. Meltzer and D. Michie, eds, American Elsevier, New York, 1971.

[7] L. Kitchen, "Relaxation applied to matching quantitative relational structures", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-10, no. 2, February 1980, pp. 96–101.

[8] L. J. Kitchen and M. Callahan, "Optimal cell size for efficient retrieval of sparse data by approximate 2D position using a coarse spatial array", *Proc. Computer Vision and Pattern Recognition Conference*, Miami, June 1986.

[9] L. Kitchen and A. Rosenfeld, "Scene analysis using region-based constraint filtering", *Pattern Recognition*, vol. 17, no. 2, 1984, pp. 189–203.

[10] N. J. Nilsson, *Problem-Solving Methods in Artificial Intelligence*, McGraw-Hill, New York, 1971.

[11] S. Ranade and A. Rosenfeld, "Point pattern matching by relaxation", *Pattern Recognition*, vol. 12, 1980, pp. 260–275.

[12] L. J. Kitchen, R. S. Weiss and J. Tuttle, "Recognition of human faces in image sequences of indoor scenes", unpublished report, VISIONS Group, COINS Dept, University of Massachusetts, Amherst, MA 01003, September 1984.

TABLES

Image	Model		
	Hiro	Les	Mike
1	0.6914	0.5608	0.5809
Hiro 2	0.7029	0.5462	0.6691
3	0.6596	0.5317	0.5639
1	0.7420	0.7029	0.7494
Les 2	0.5932	0.6026	0.7170
3	0.5309	0.5890	0.7366
1	0.6647	0.6578	0.7031
Mike 2	0.4126	0.3726	0.5082
3	0.4575	0.3839	0.5171

Table 1: Average confidence of matches.

Image	Model		
	Hiro	Les	Mike
1	0.4322	0.4149	0.4300
Hiro 2	0.6287	0.4569	0.4928
3	0.5555	0.4003	0.4245
1	0.6754	0.6268	0.7109
Les 2	0.4536	0.3829	0.5962
3	0.3245	0.3500	0.5924
1	0.5096	0.5524	0.6337
Mike 2	0.2728	0.3109	0.2473
3	0.2263	0.3109	0.3123

Table 2: Minimum confidence of matches.

True Identification	Computed Identification Hiro	Les	Mike
Hiro	3	0	0
Les	0	0	3
Mike	0	0	3

Table 3: Confusion matrix for face identification using *average*.

True Identification	Computed Identification Hiro	Les	Mike
Hiro	3	0	0
Les	0	0	3
Mike	0	1	2

Table 4: Confusion matrix for face identification using *minimum*.

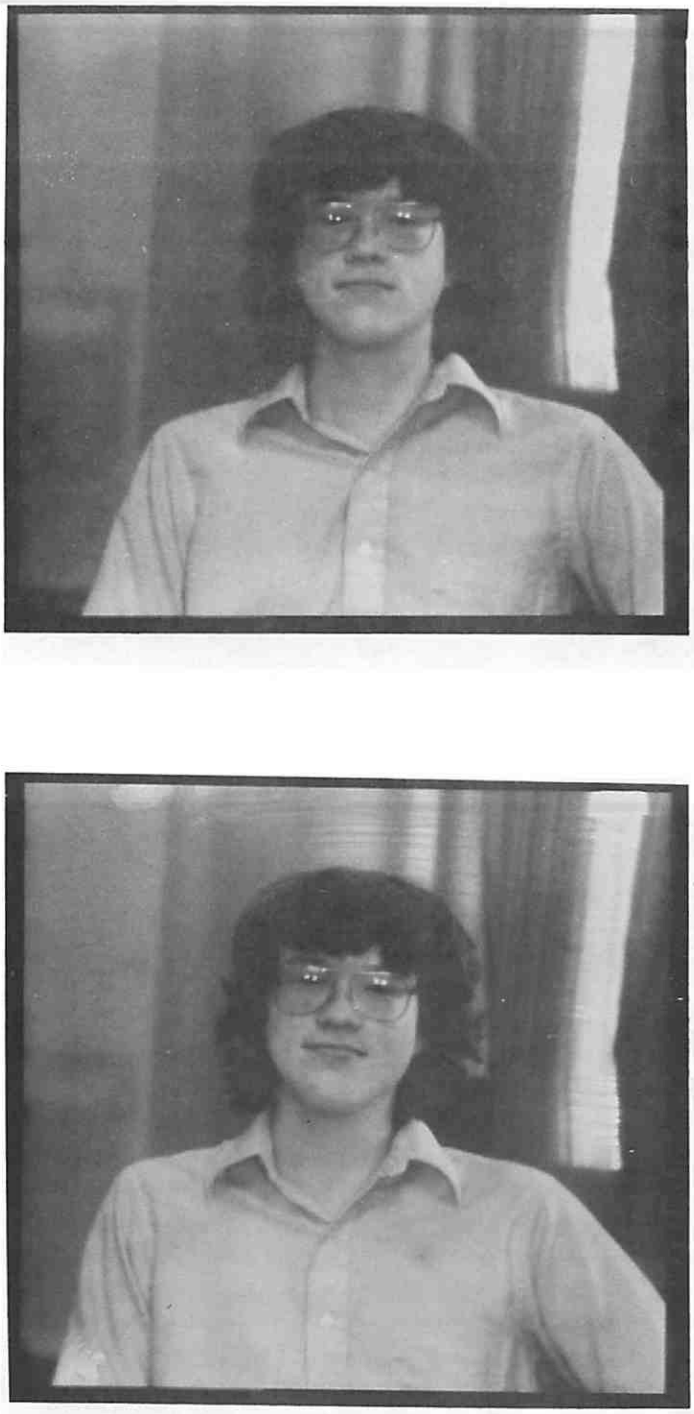


Figure 1: Images 1-2

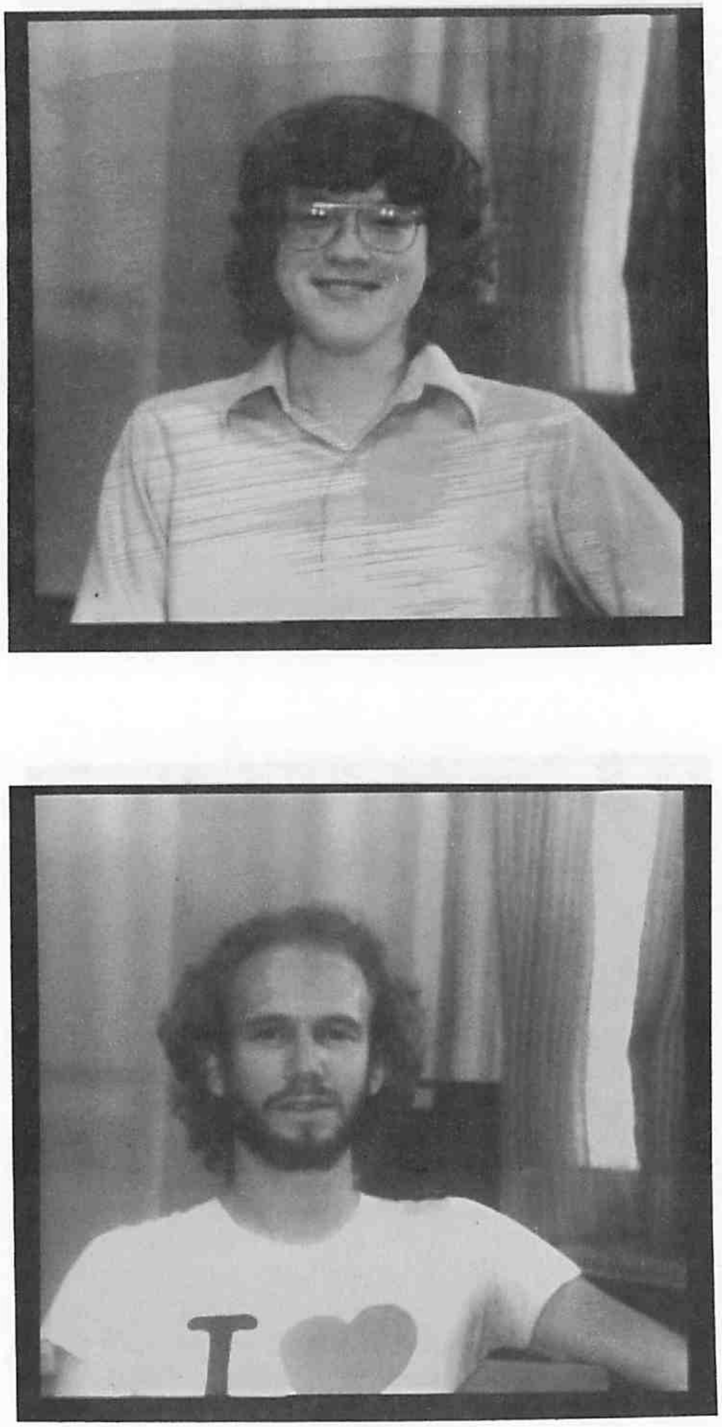


Figure 2: Images 3-4

Figure 3: Images 5-6

Figure 4: Images 7-8

Figure 5: Image 9

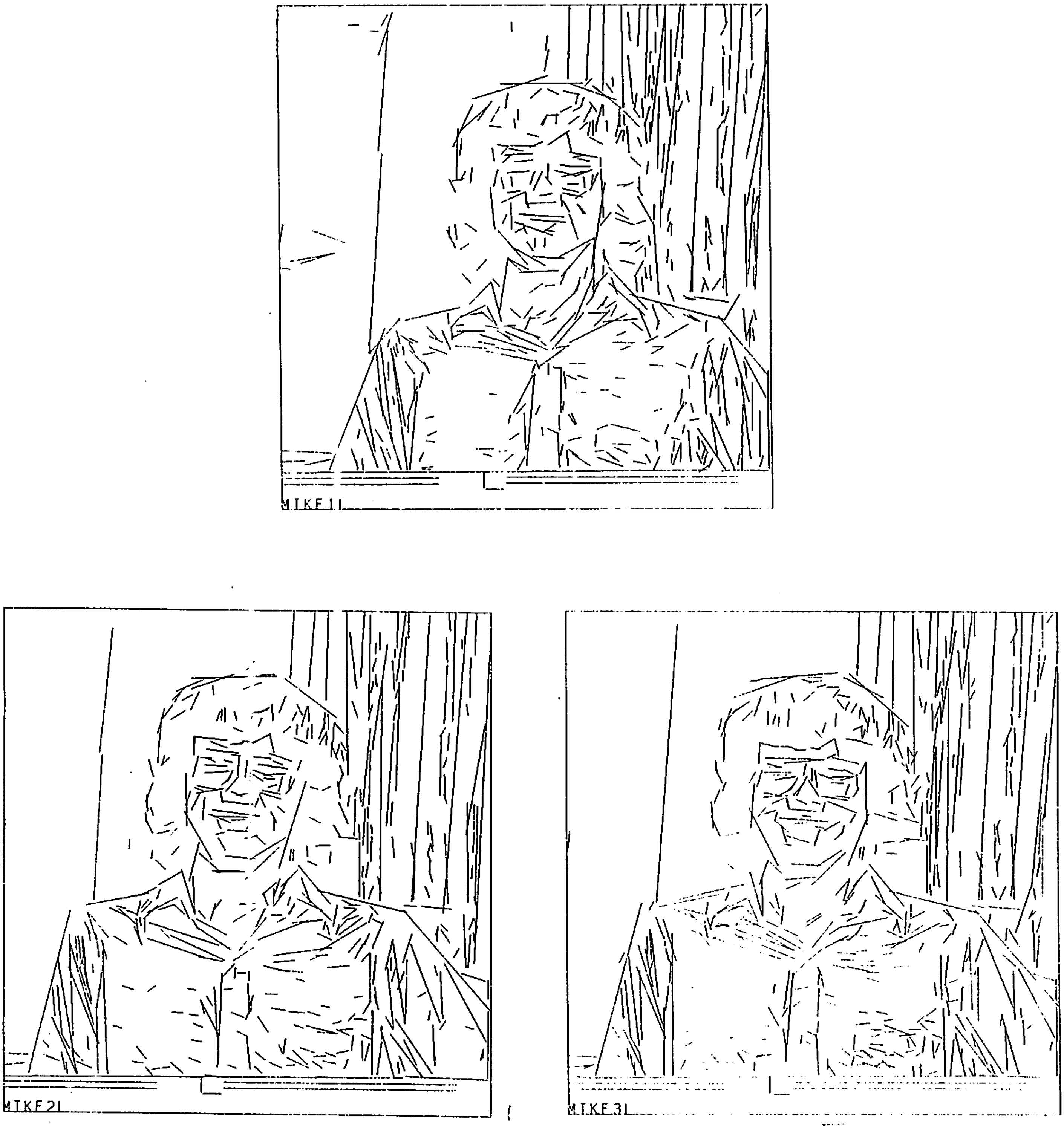


Figure 6: Lines For Images 1-3

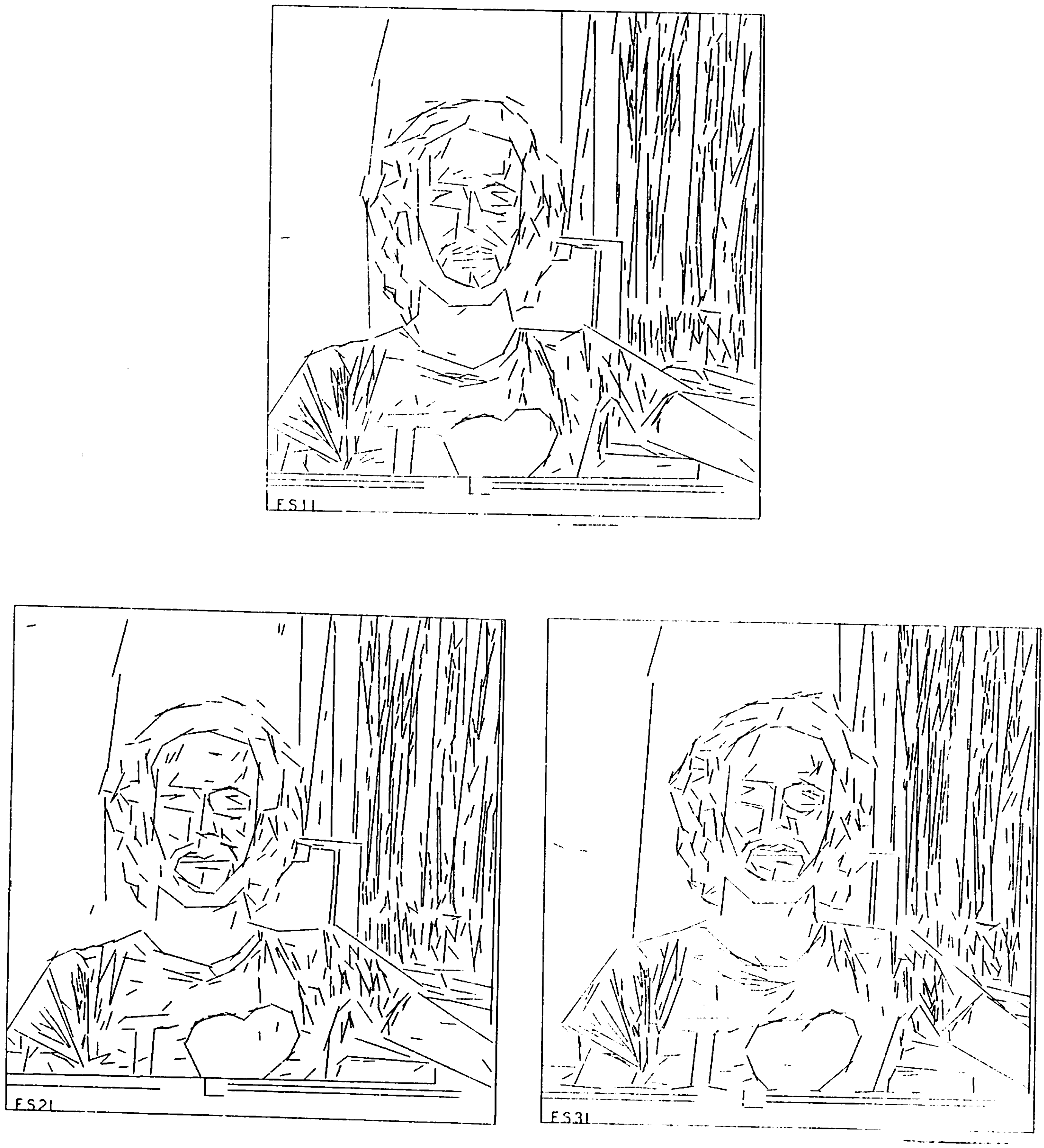


Figure 7: Lines For Images 4-6

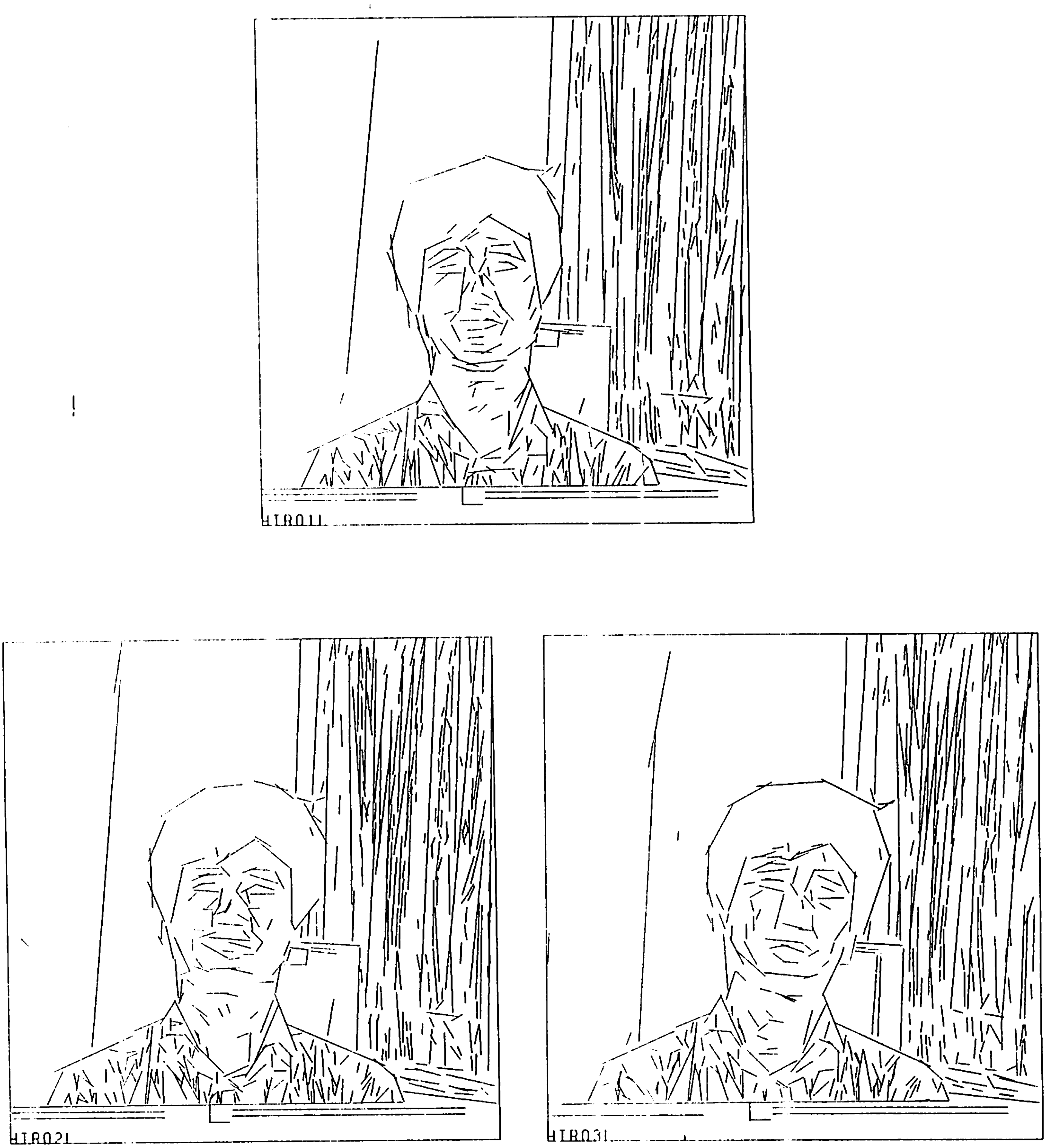


Figure 8: Lines For Images 7-9

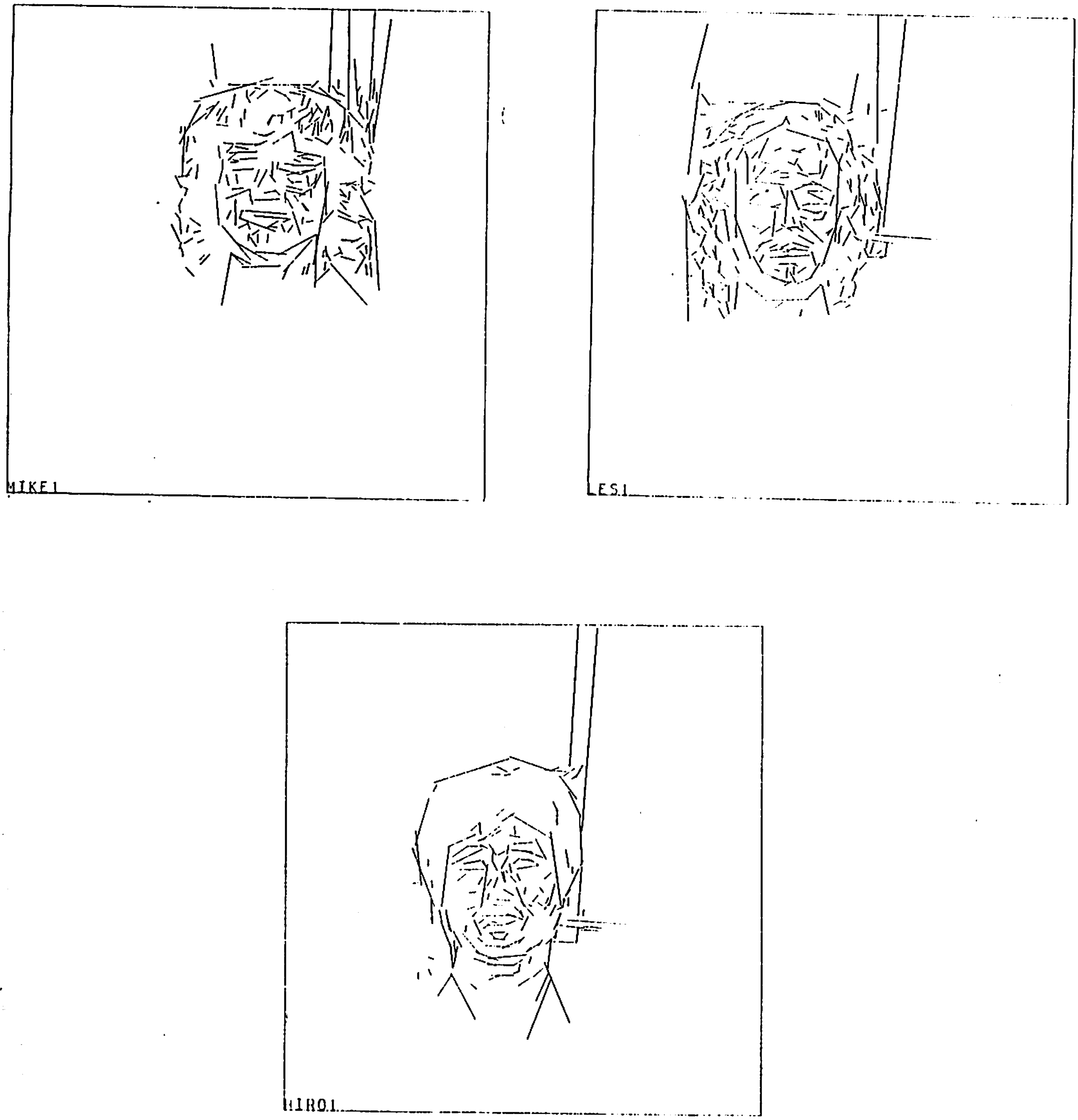


Figure 9: Lines Which Intersect the Mask for Images 1,4,7

Figure 10: Segmentation of Images 1-3

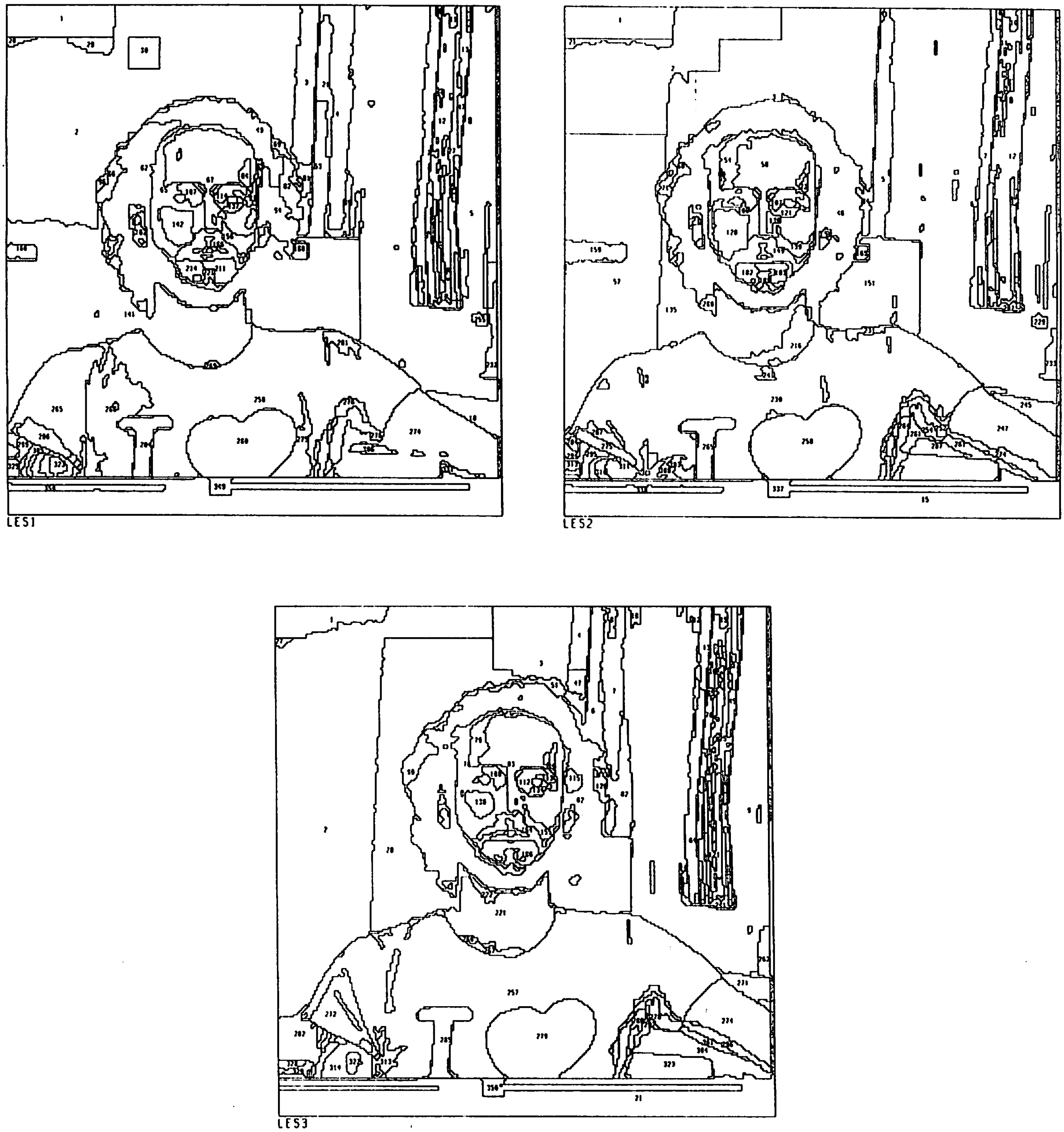


Figure 11: Segmentation of Images 4-6

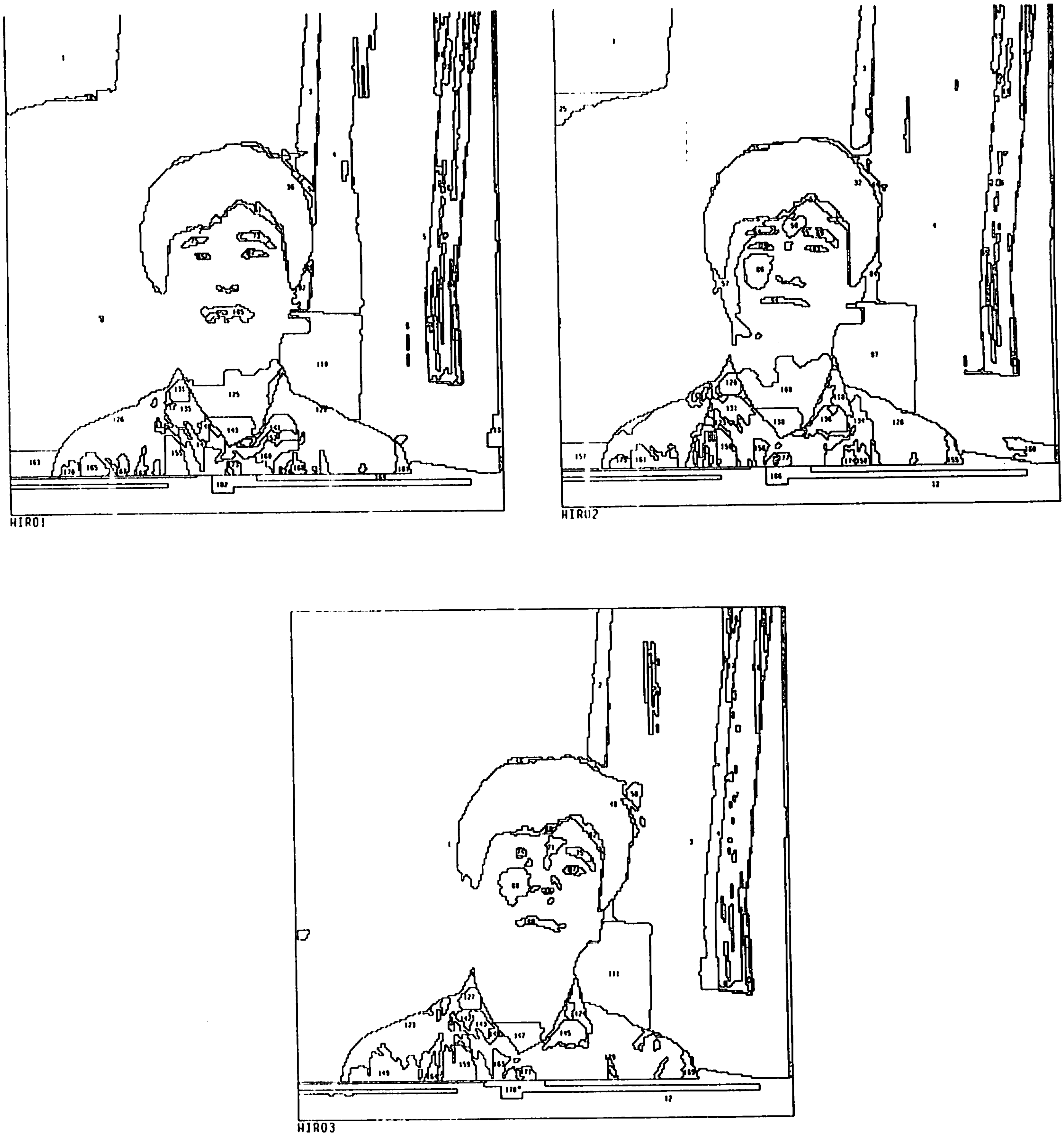


Figure 12: Segmentation of Images 7-9

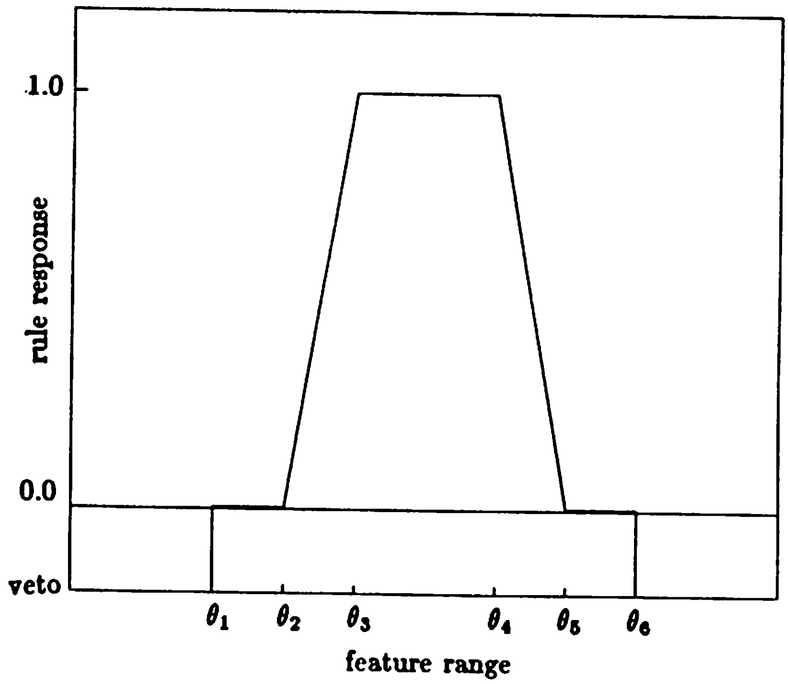
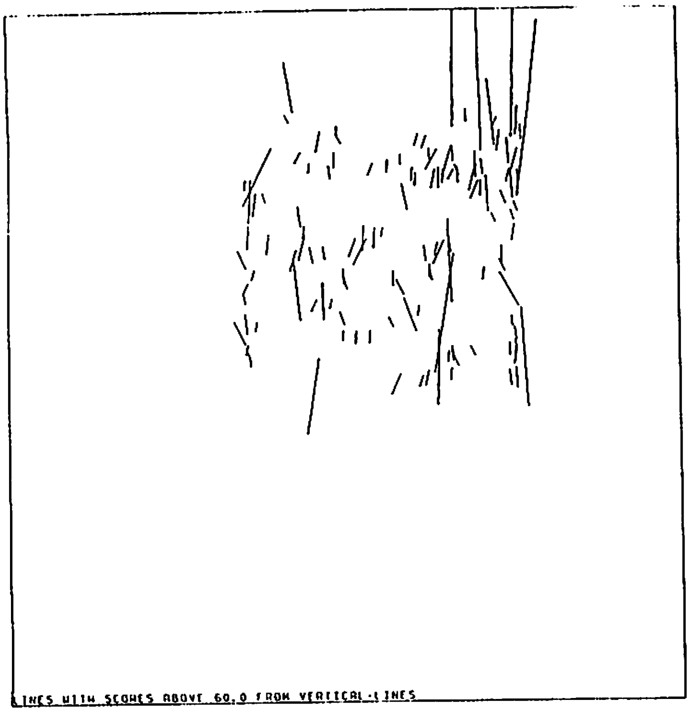
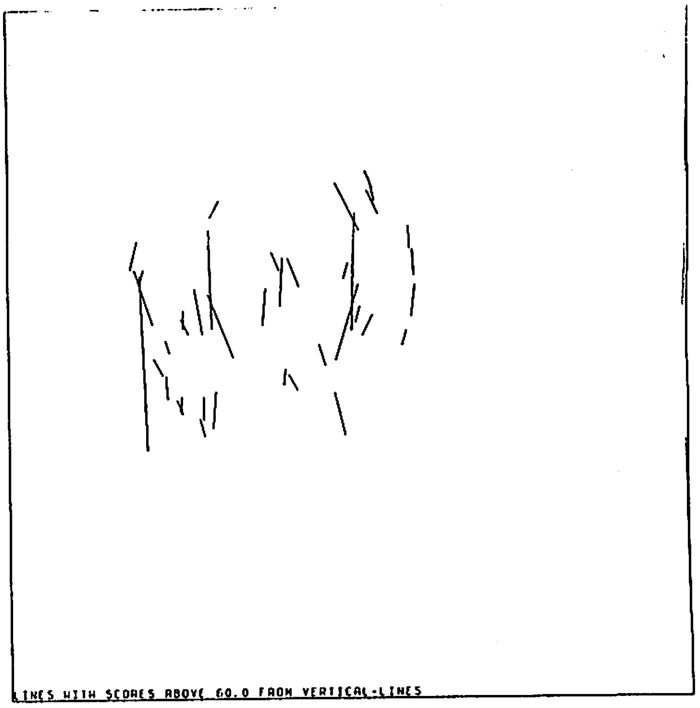


Figure 13: Structure of a simple rule for mapping an image feature measurement f_I into support for a label hypothesis on the basis of a prototype feature value. The object specific mapping is parameterized by six values, $\theta_1, ..., \theta_6$.





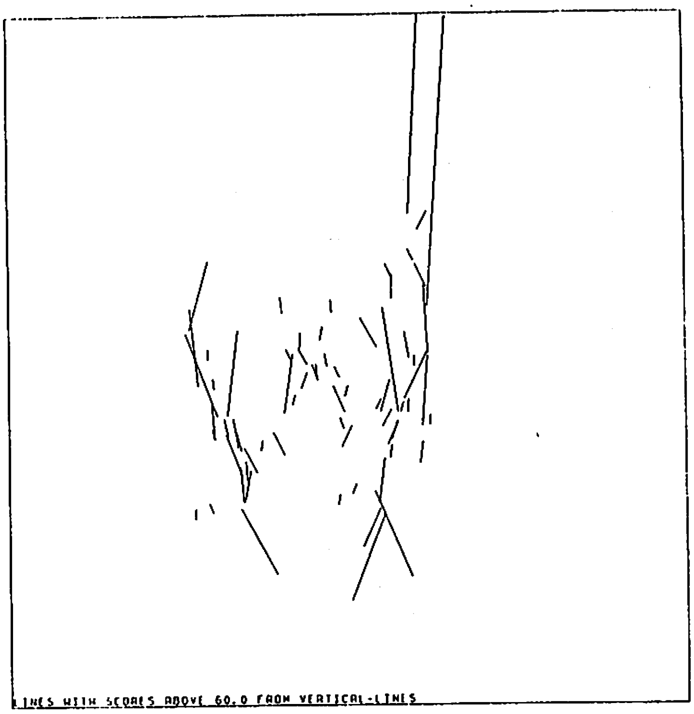


Figure 14: Vertical Lines, Images 1,4,7

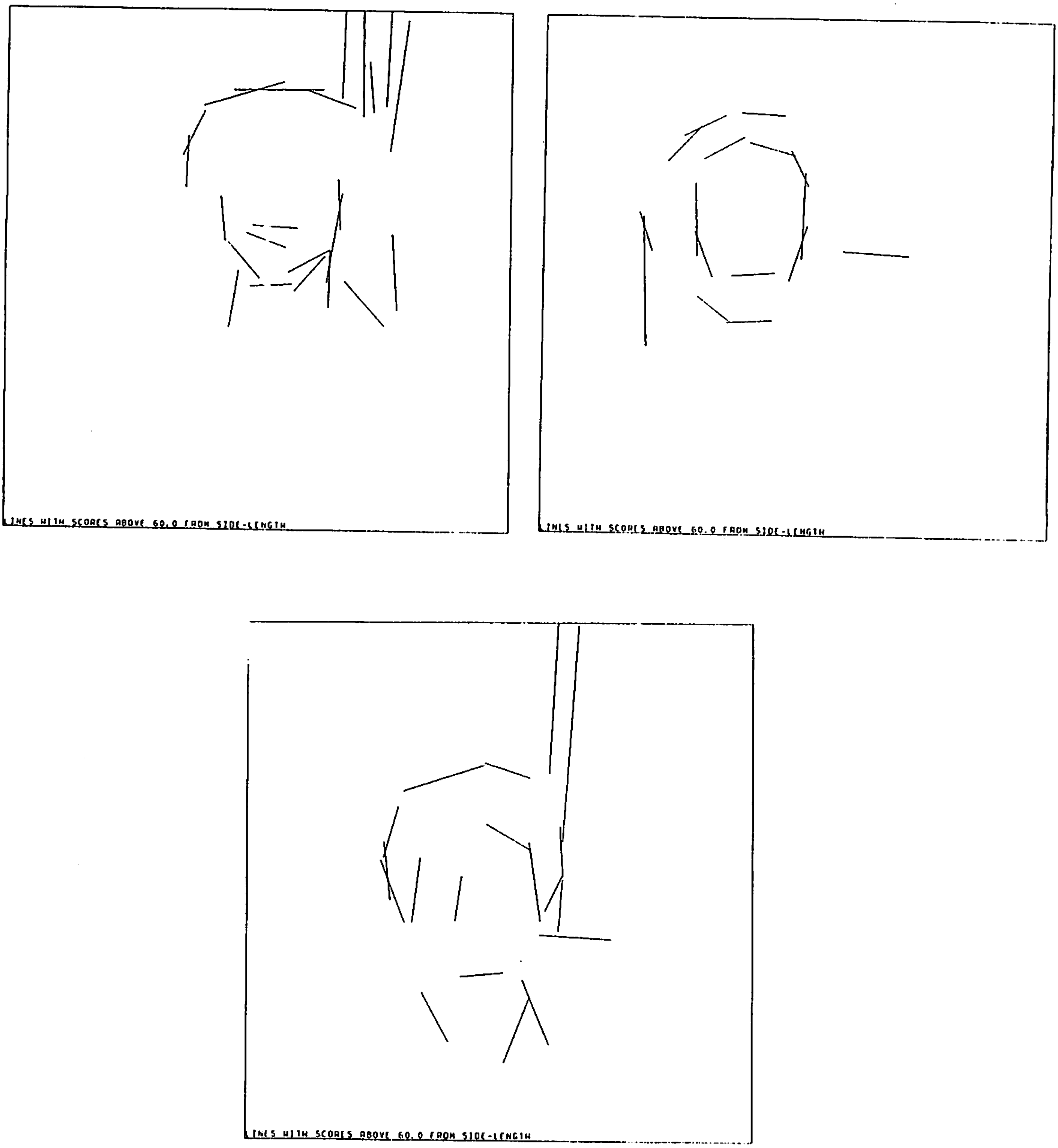


Figure 15: Lines Whose Length Satisfies Side-of-face Rule, Images 1,4,7

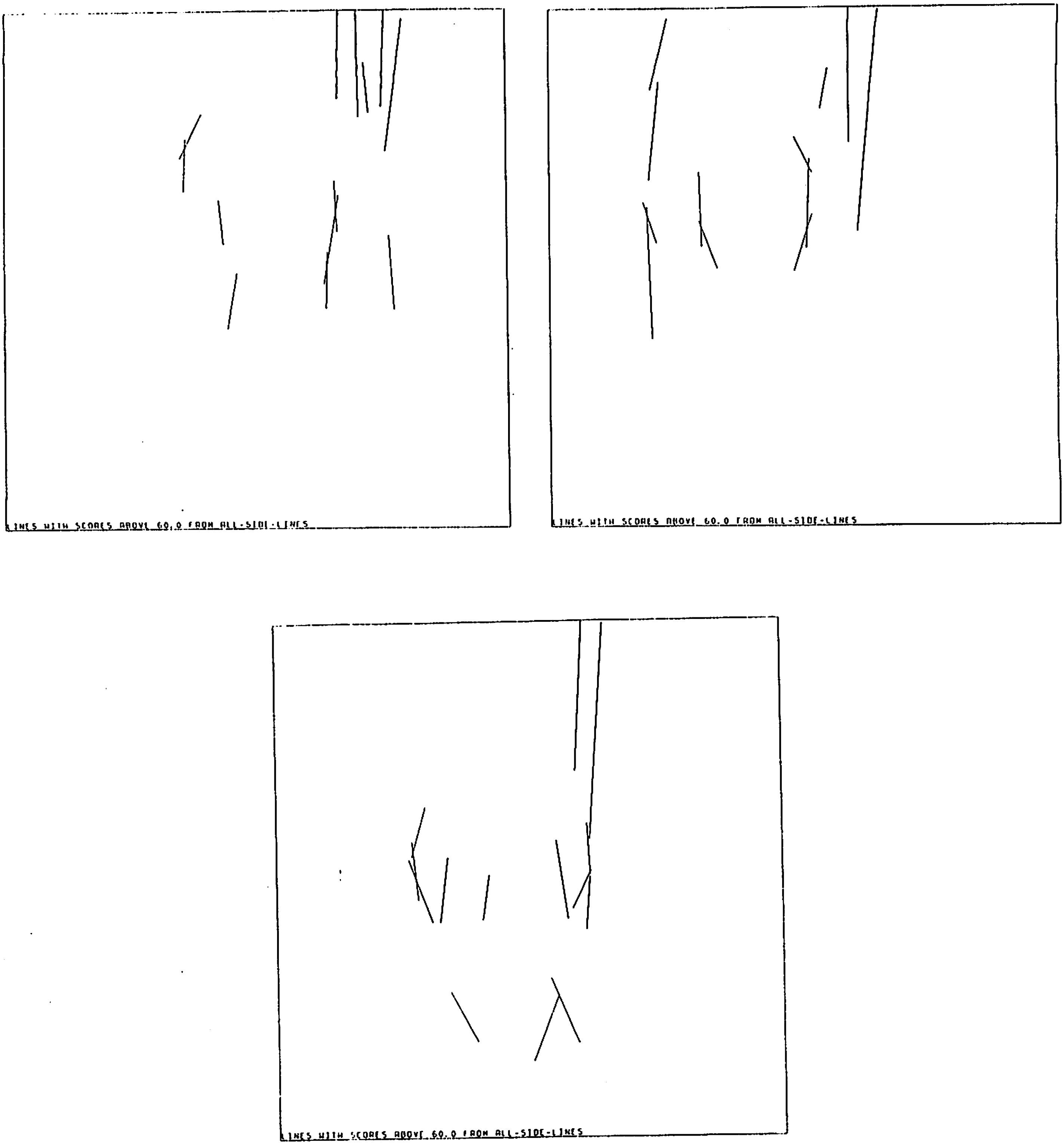


Figure 16: Lines Which Satisfy Side-of-face Rule, Images 1,4,7

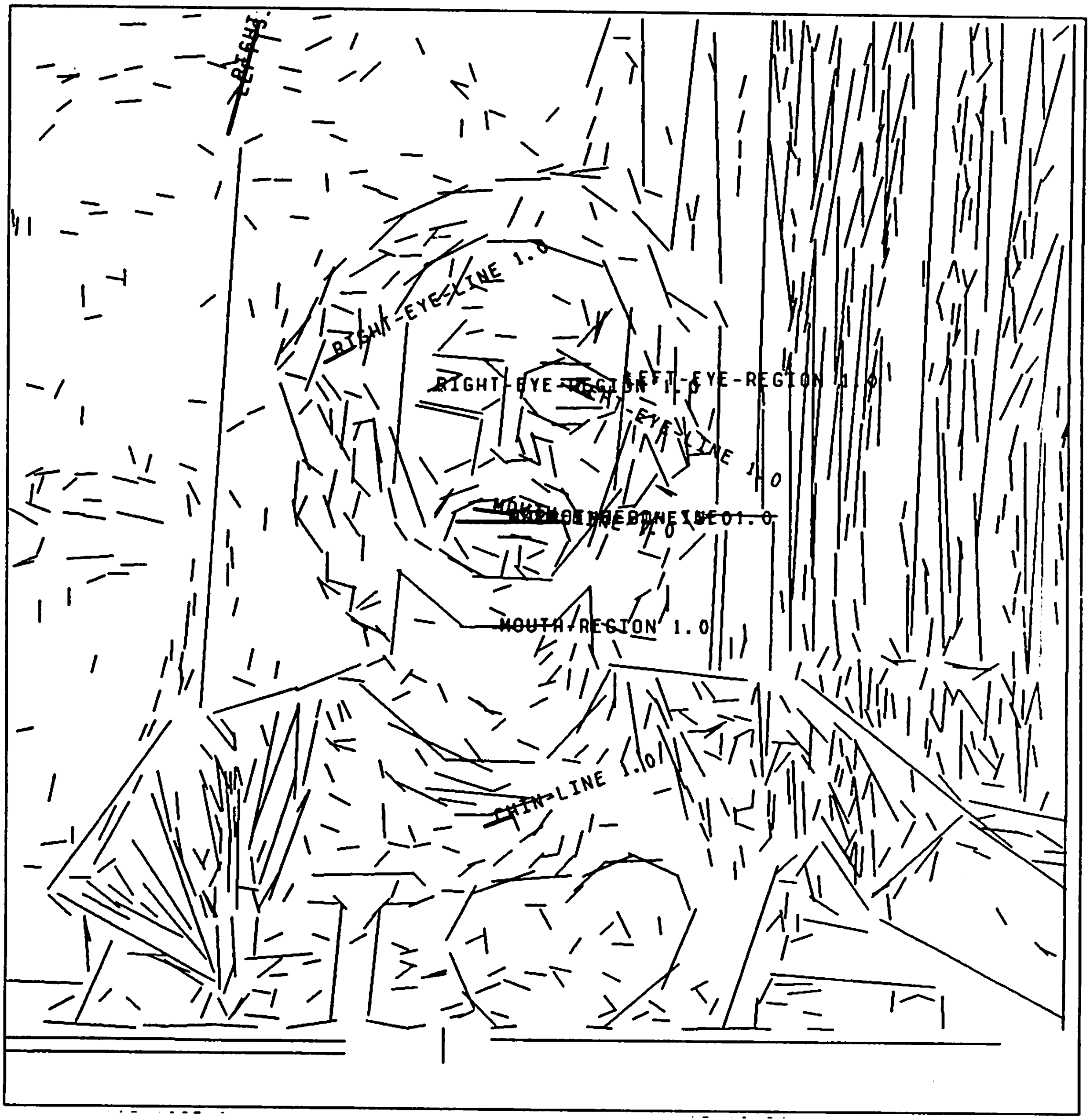


Figure 17: "Les" model against "Les" image 3: Best matches for each face part *before* relaxation matching.

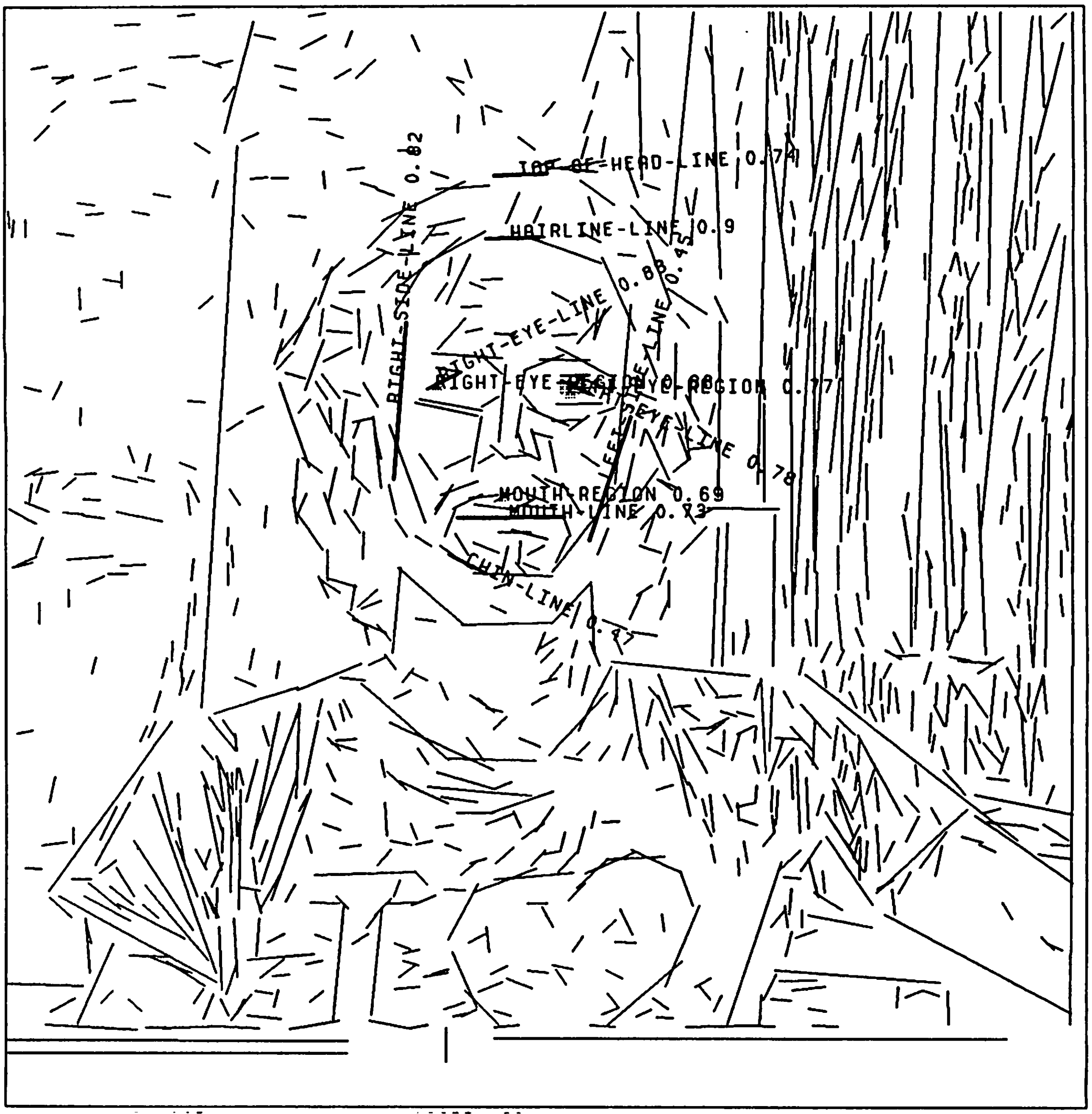


Figure 18: “Les” model against “Les” image 3: Best matches for each face part after one iteration of relaxation matching.

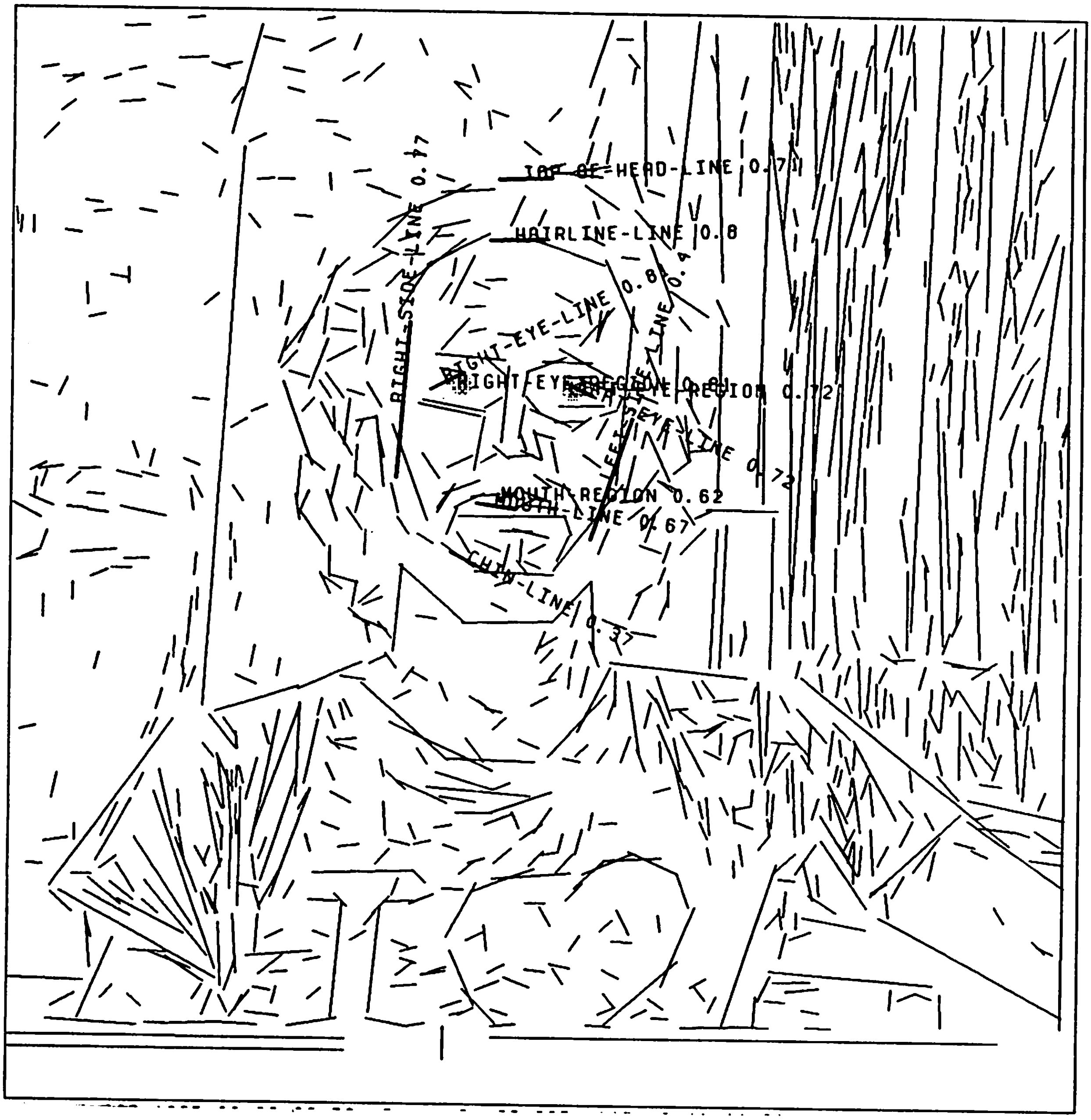


Figure 19: "Les" model against "Les" image 3: Best matches for each face part after two iterations of relaxation matching.

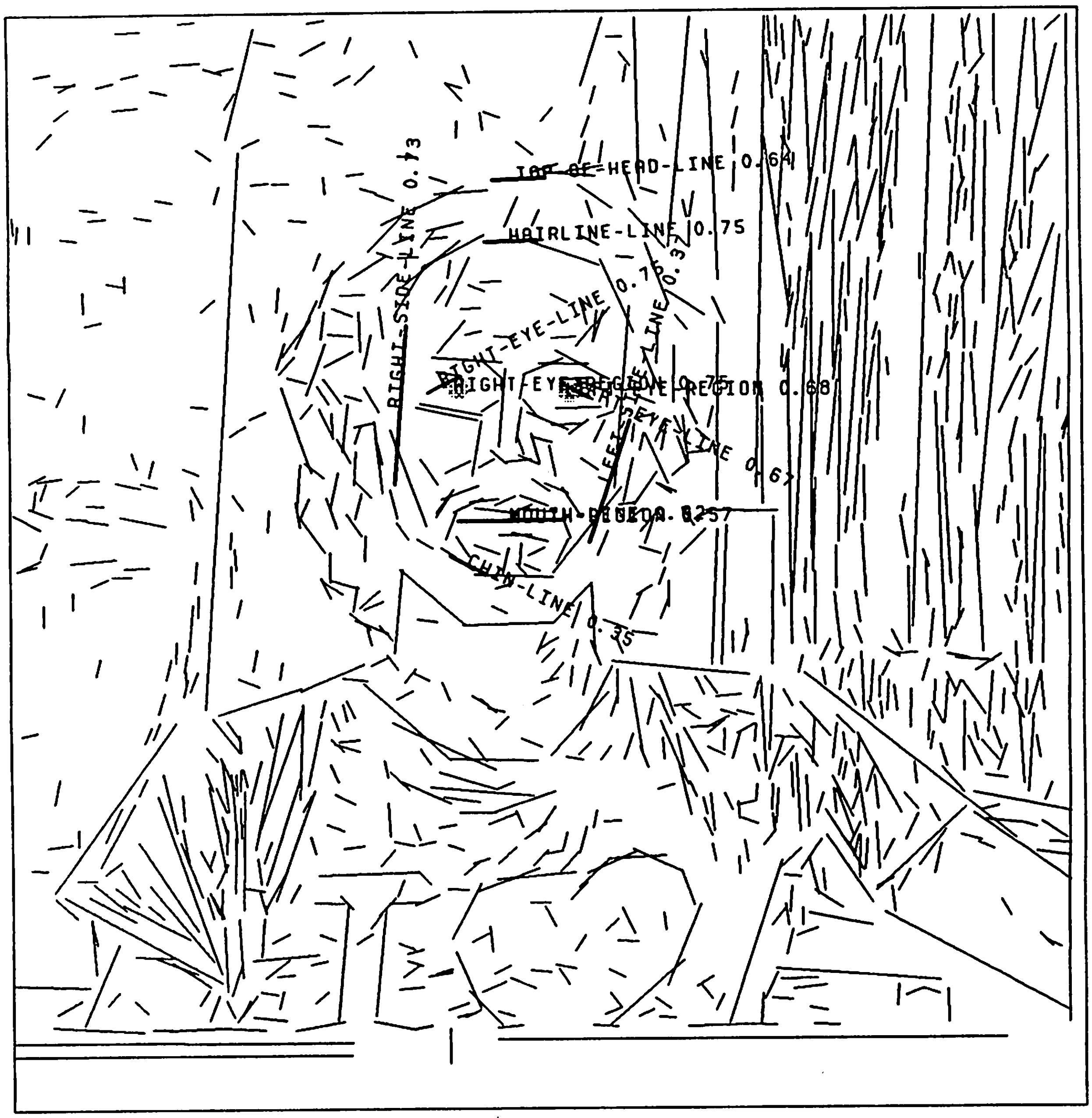


Figure 20: "Les" model against "Les" image 3: Best matches for each face part after three iterations of relaxation matching.

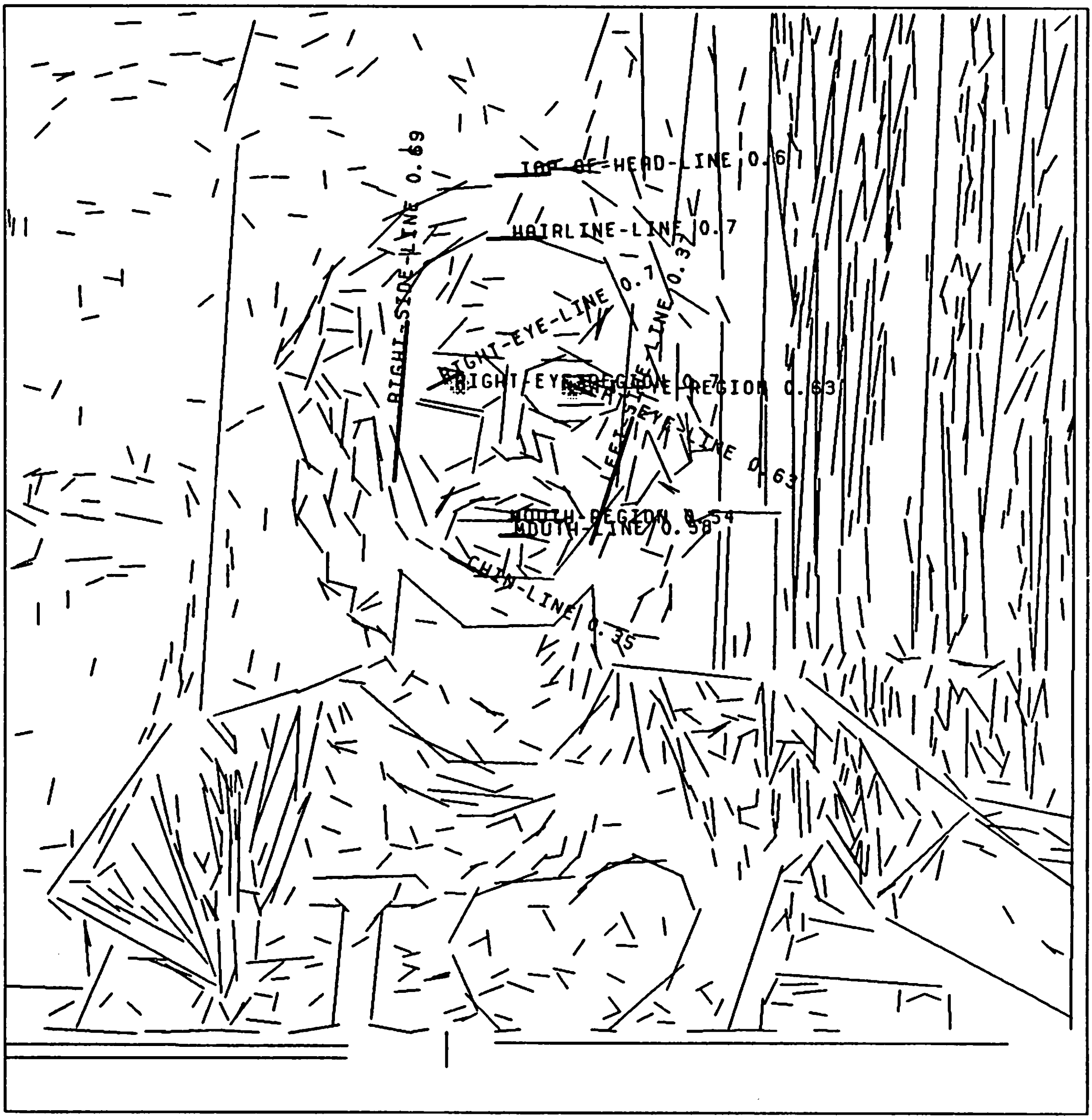


Figure 21: "Les" model against "Les" image 3: Best matches for each face part after four iterations of relaxation matching.

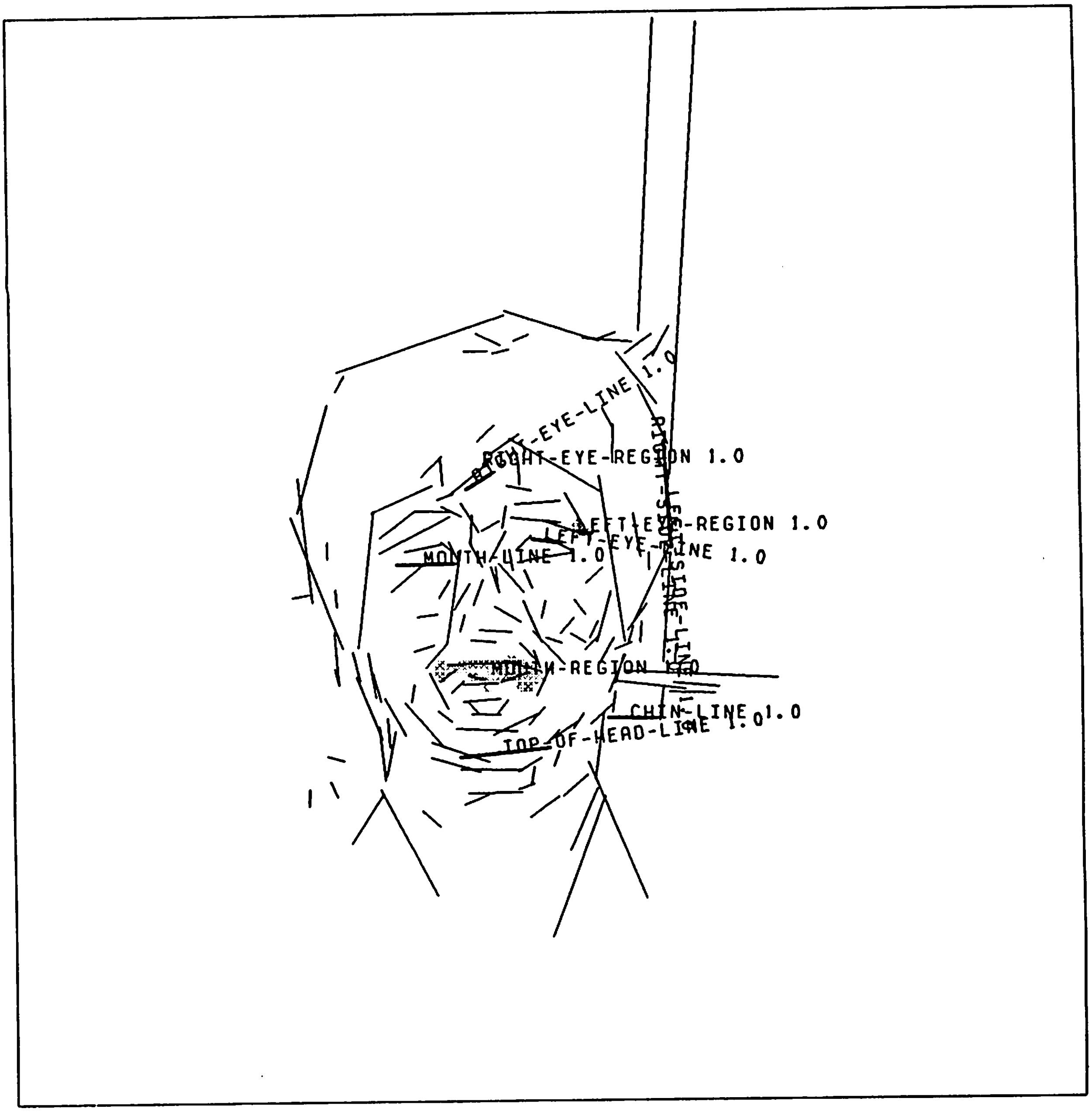


Figure 22: "Hiro" model against "Hiro" image 1: Best matches for each face part *before* relaxation matching.

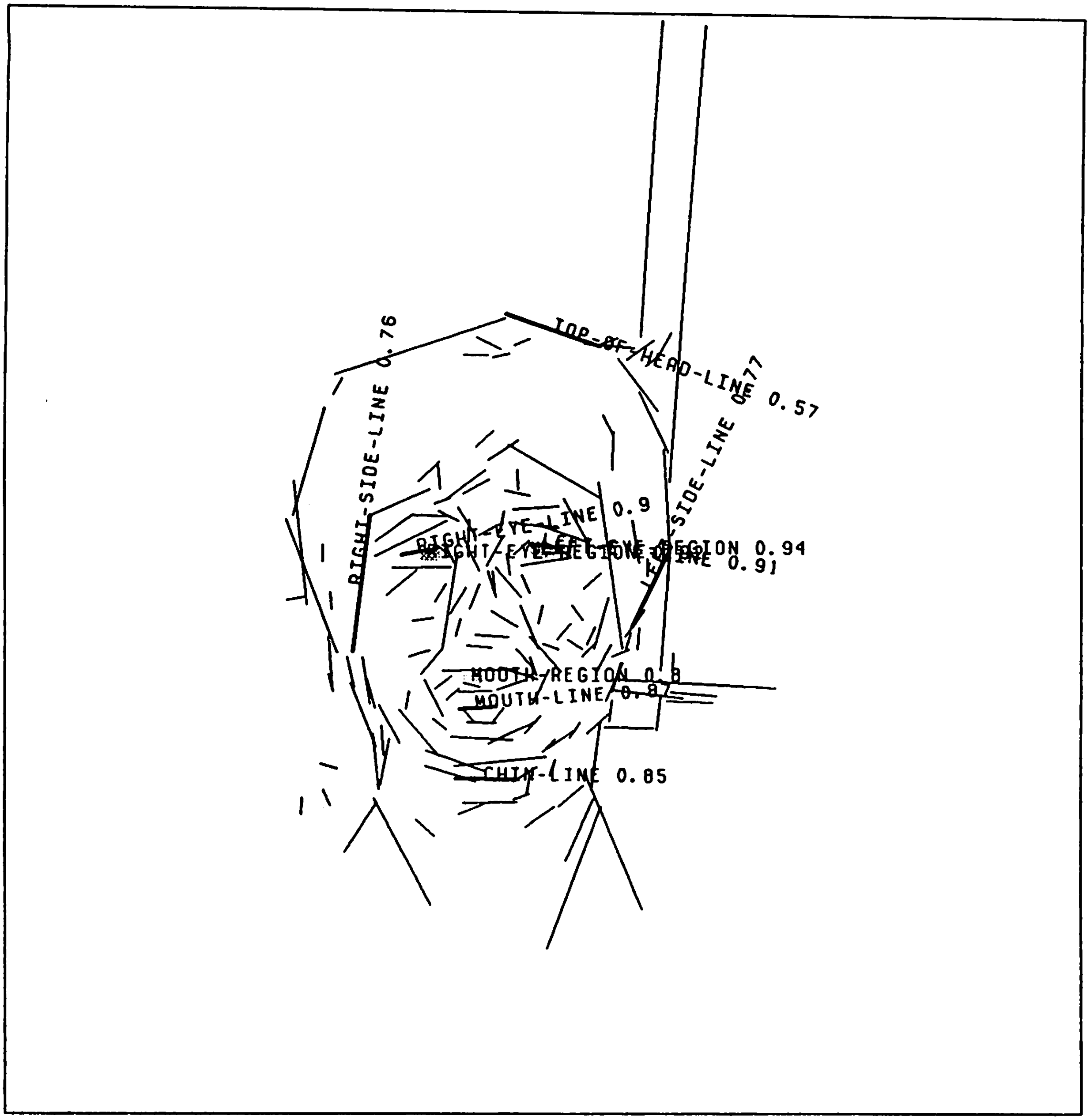


Figure 23: "Hiro" model against "Hiro" image 1: Best matches for each face part after one iteration of relaxation matching.

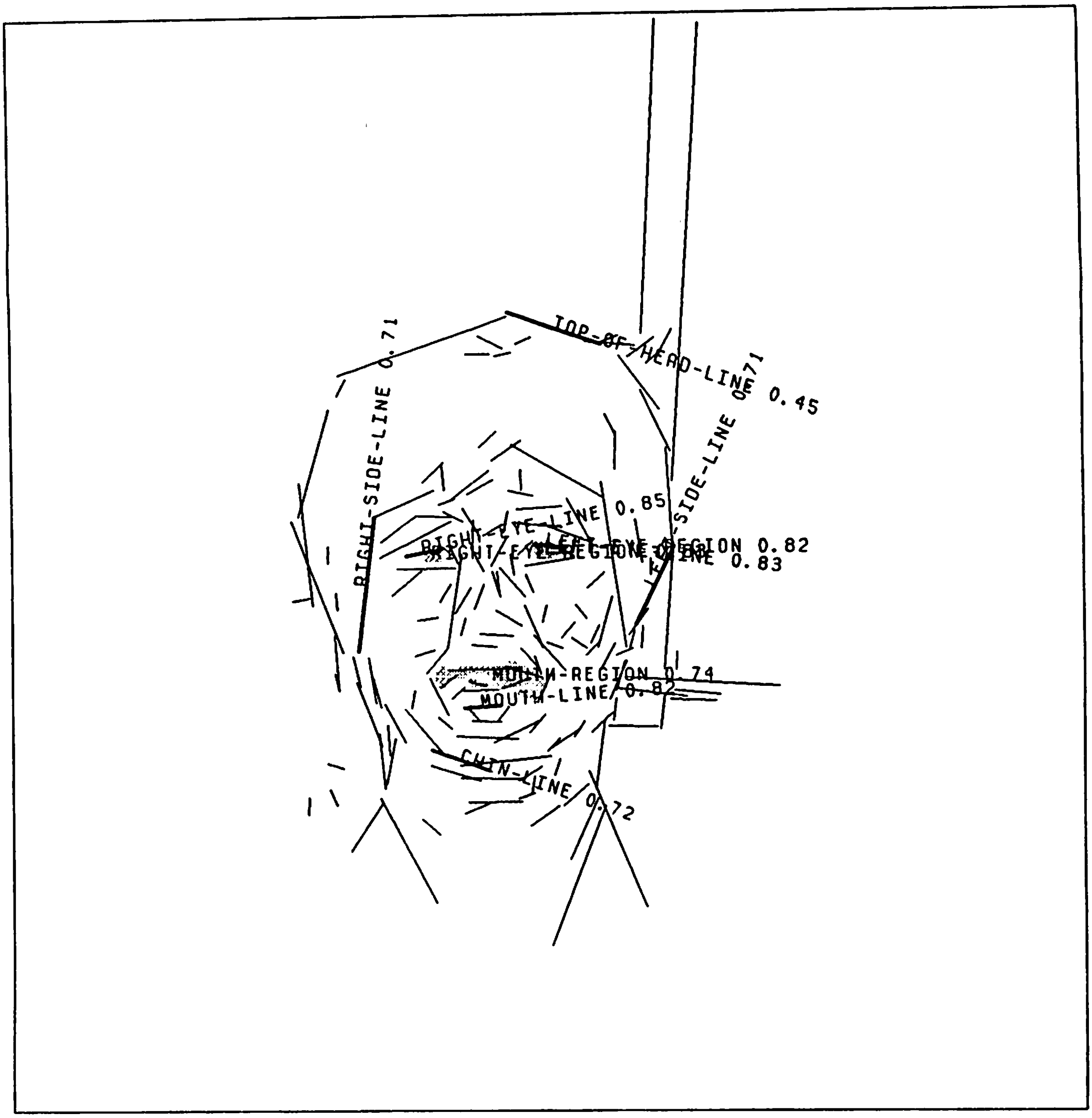


Figure 24: "Hiro" model against "Hiro" image 1: Best matches for each face part after two iterations of relaxation matching.

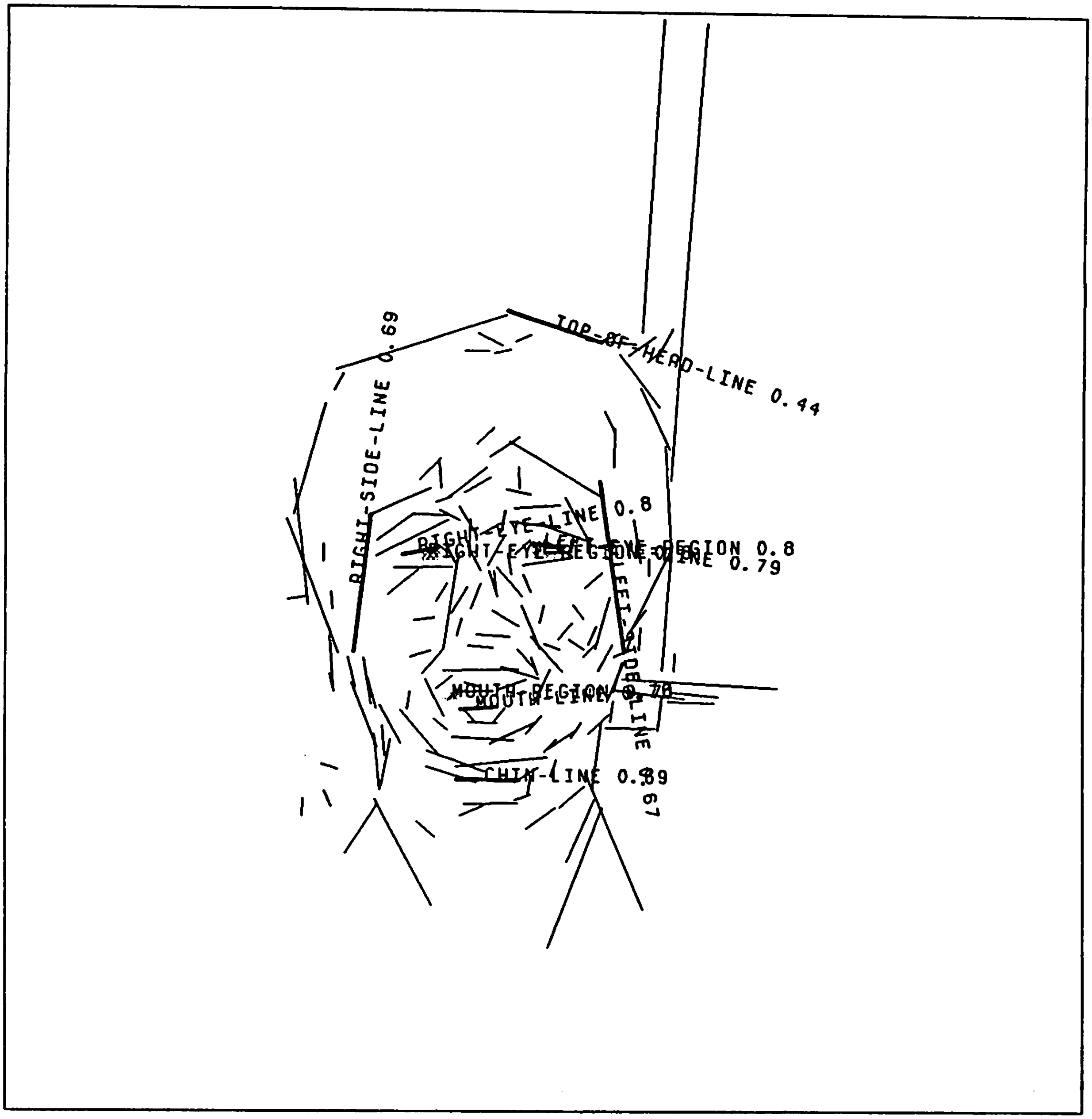


Figure 25: "Hiro" model against "Hiro" image 1: Best matches for each face part after three iterations of relaxation matching.

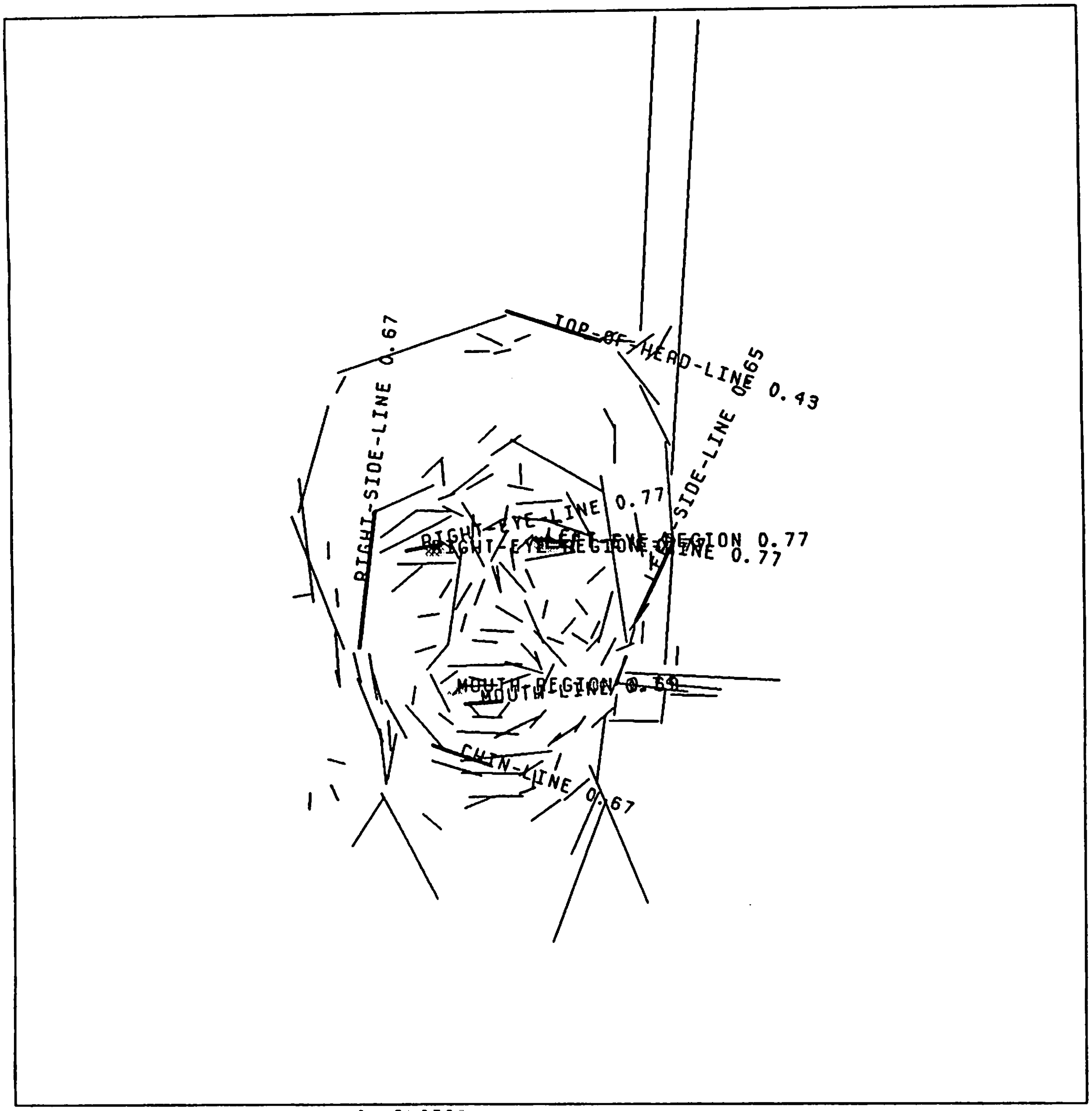


Figure 26: "Hiro" model against "Hiro" image 1: Best matches for each face part after four iterations of relaxation matching.

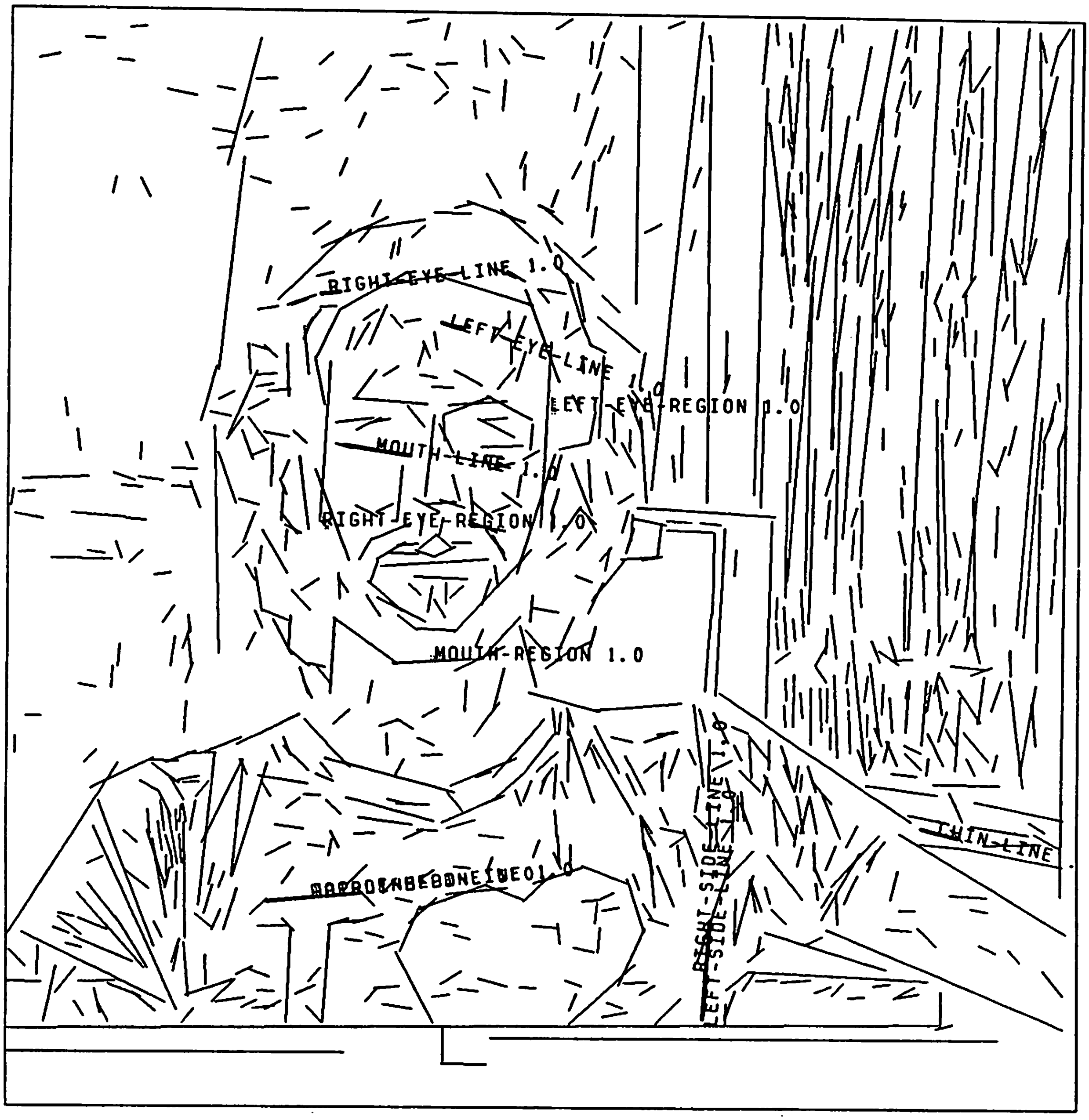


Figure 27: "Les" model against "Les" image 2: Best matches for each face part *before* relaxation matching.

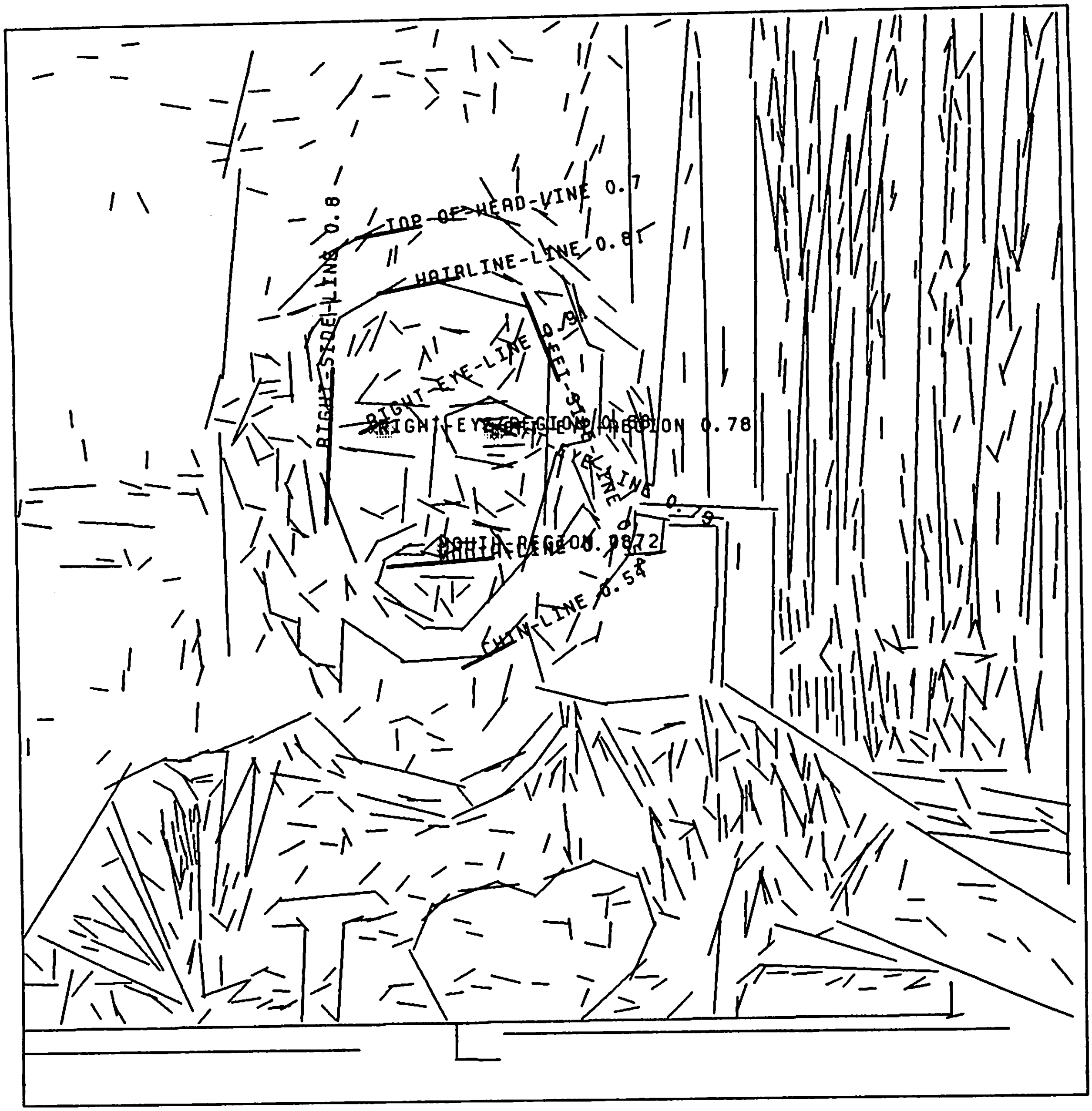


Figure 28: "Les" model against "Les" image 2: Best matches for each face part after one iteration of relaxation matching.

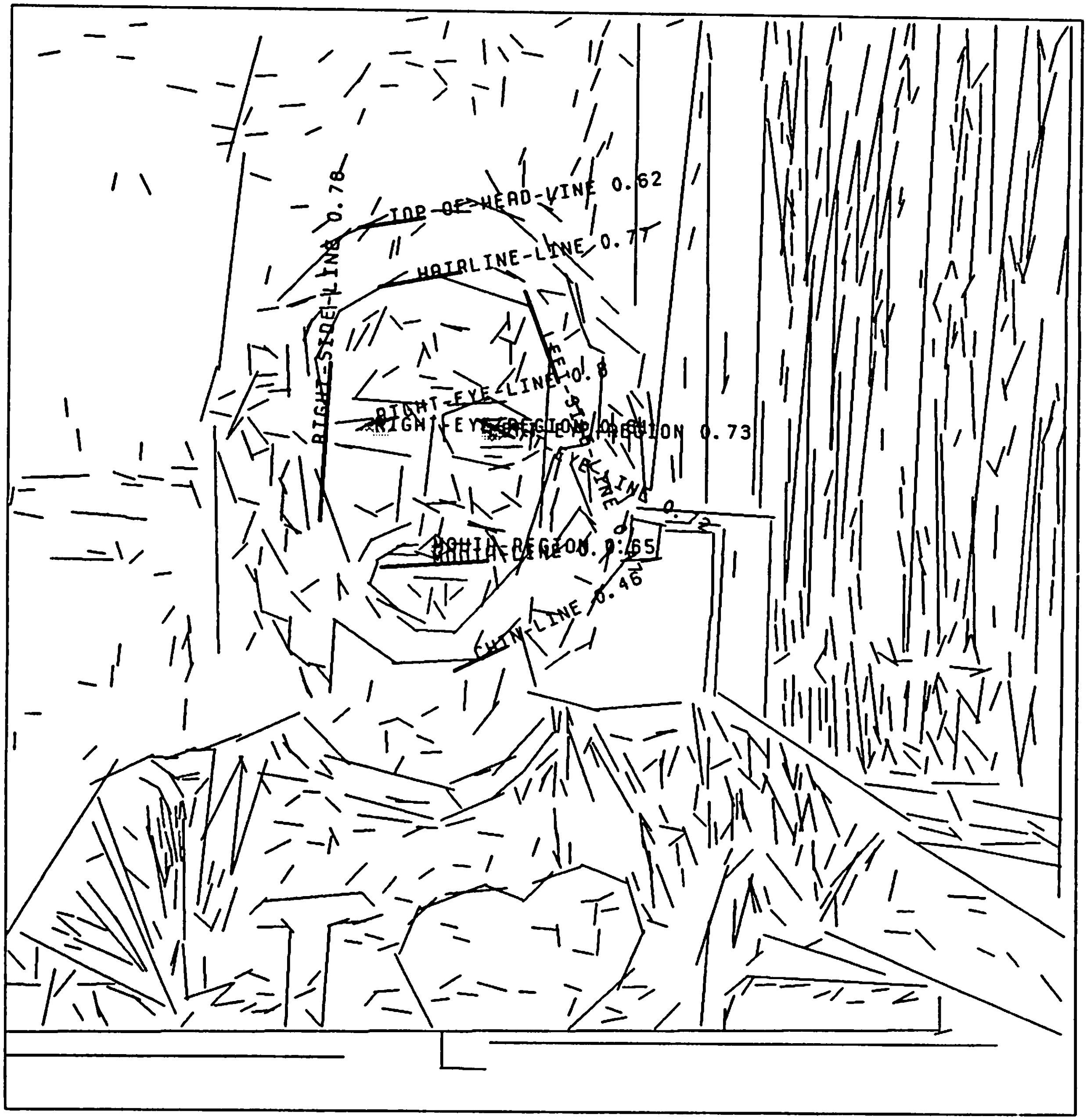


Figure 29: "Les" model against "Les" image 2: Best matches for each face part after two iterations of relaxation matching.

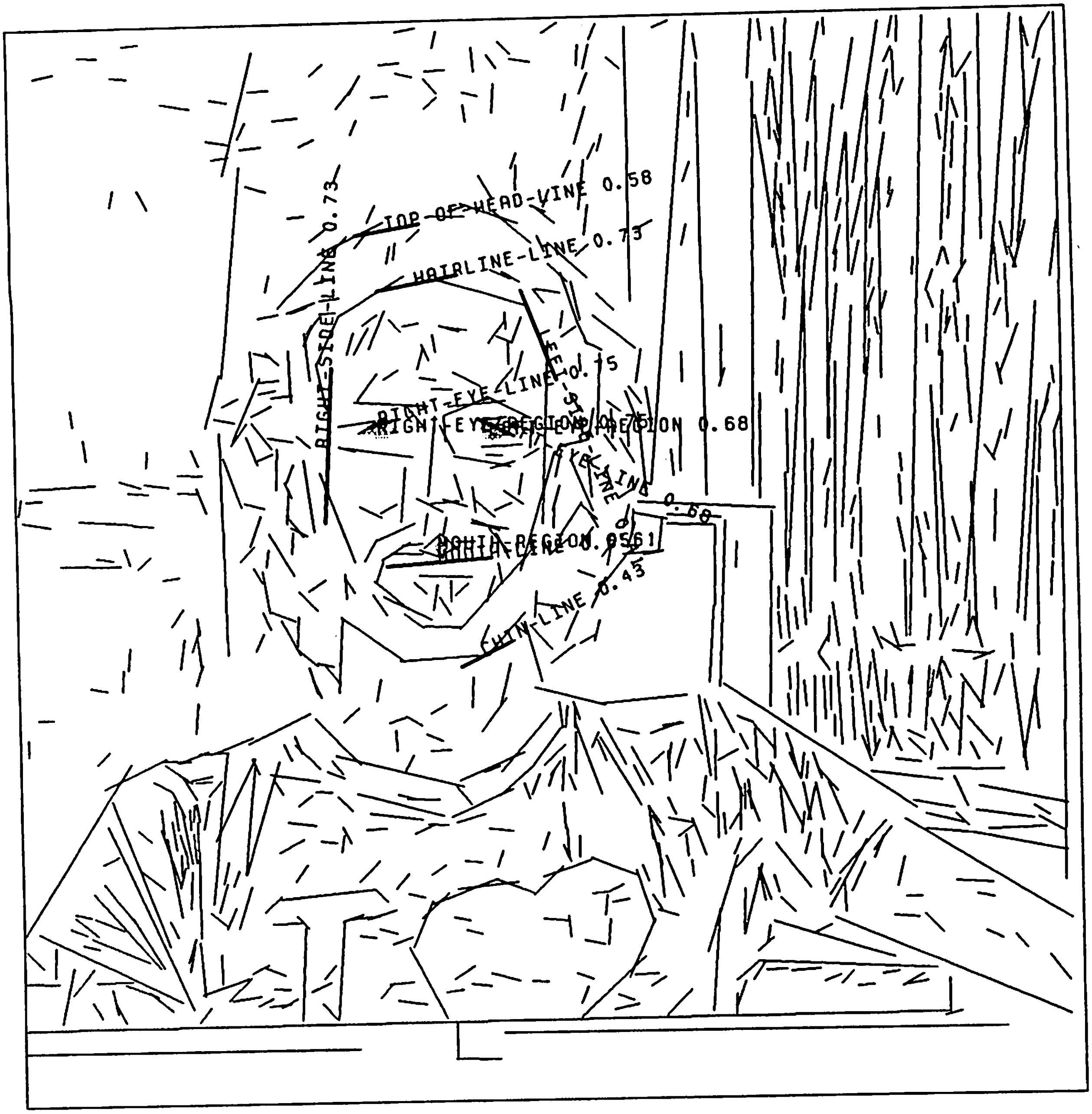


Figure 30: "Les" model against "Les" image 2: Best matches for each face part after three iterations of relaxation matching.

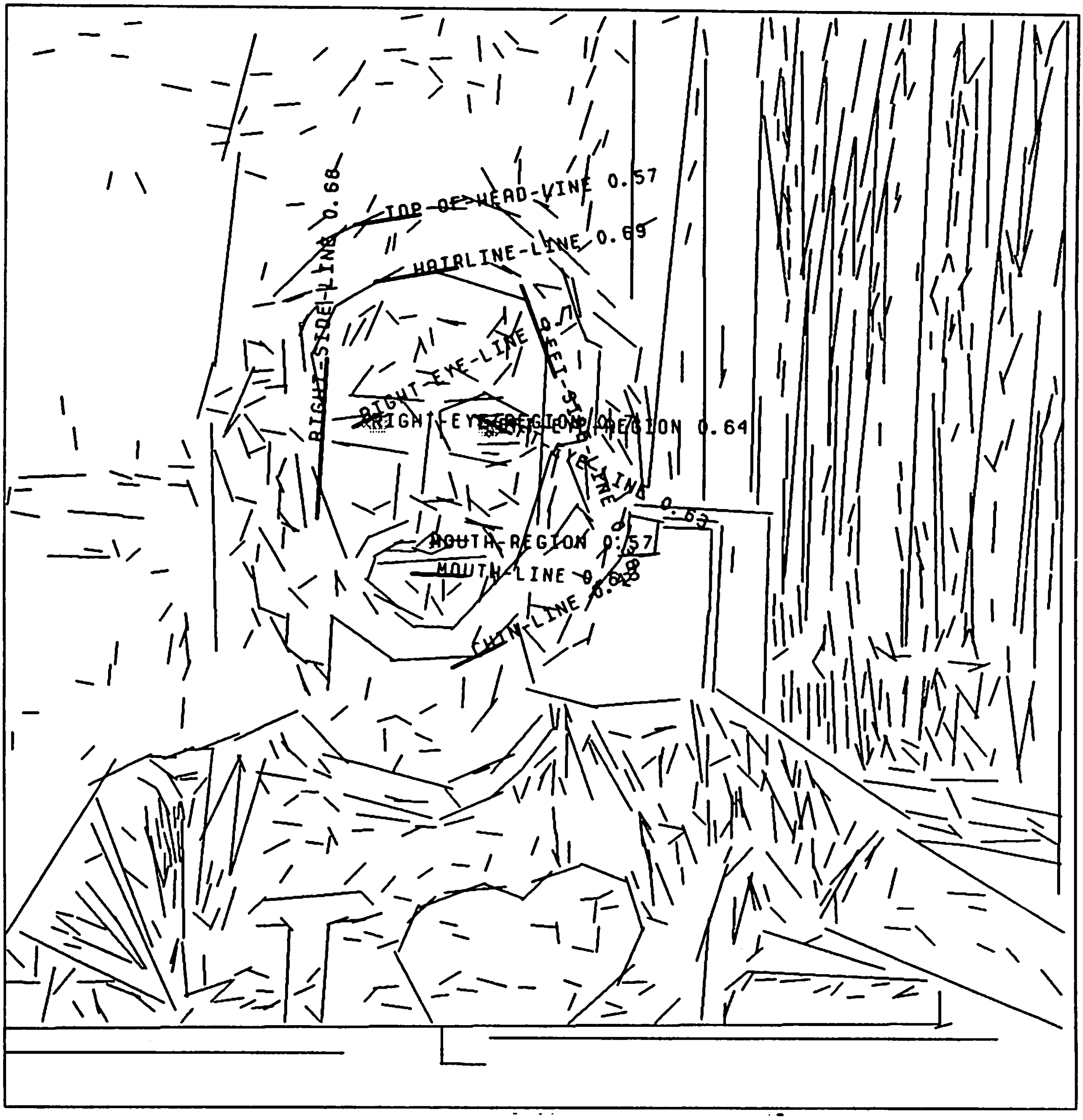


Figure 31: "Les" model against "Les" image 2: Best matches for each face part after four iterations of relaxation matching.

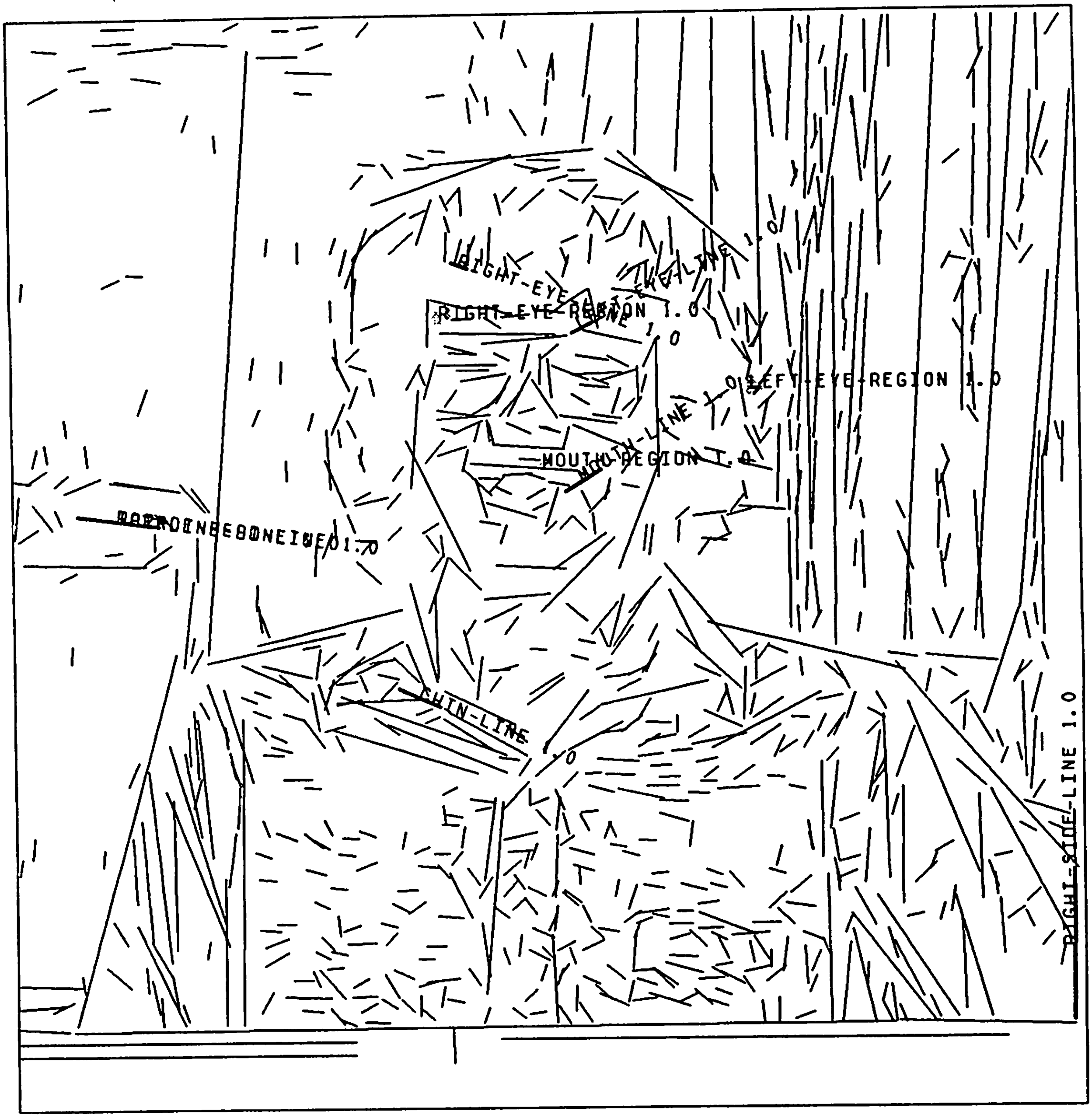


Figure 32: "Mike" model against "Mike" image 3: Best matches for each face part *before* relaxation matching.

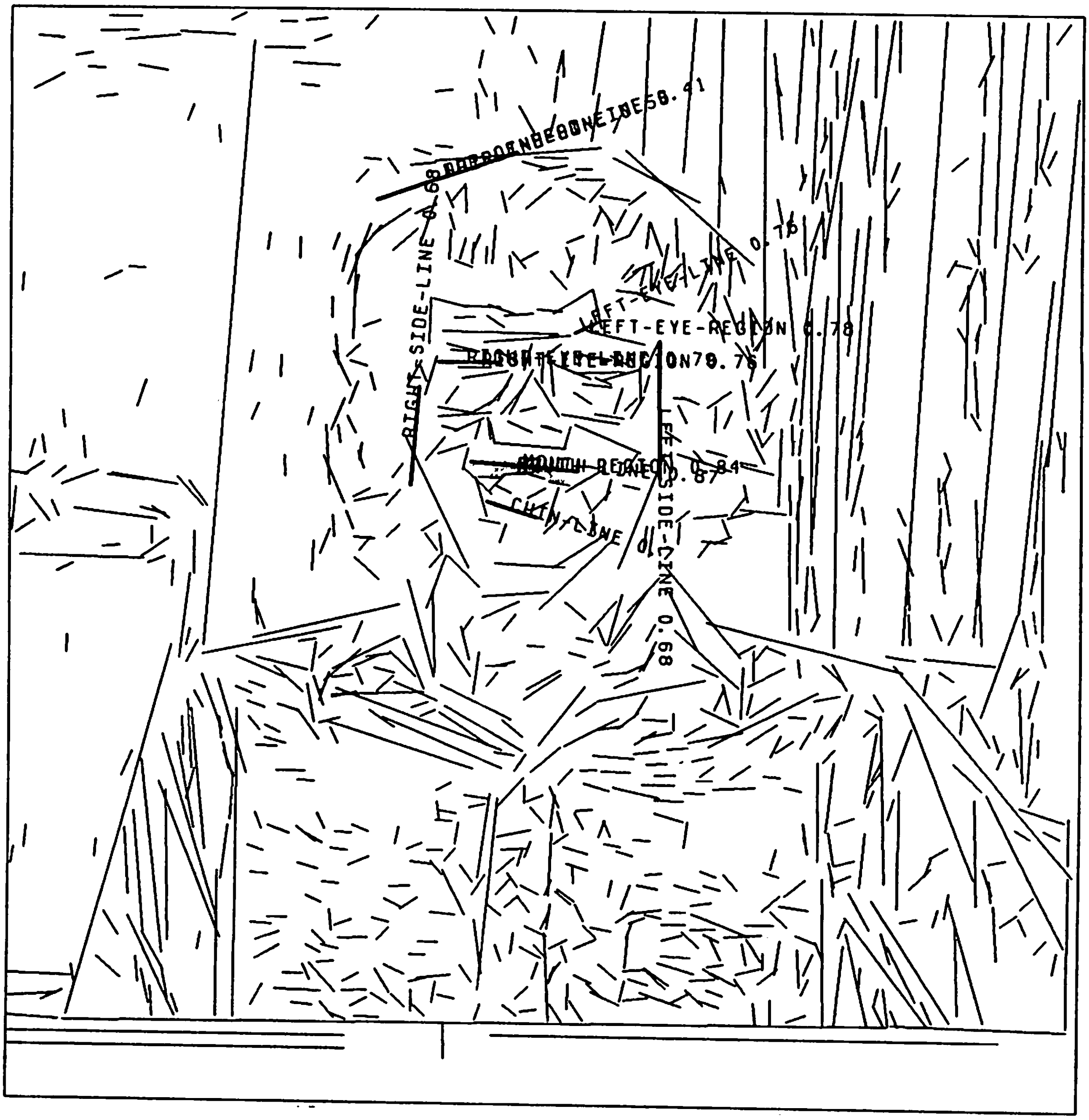


Figure 33: "Mike" model against "Mike" image 3: Best matches for each face part after one iteration of relaxation matching.

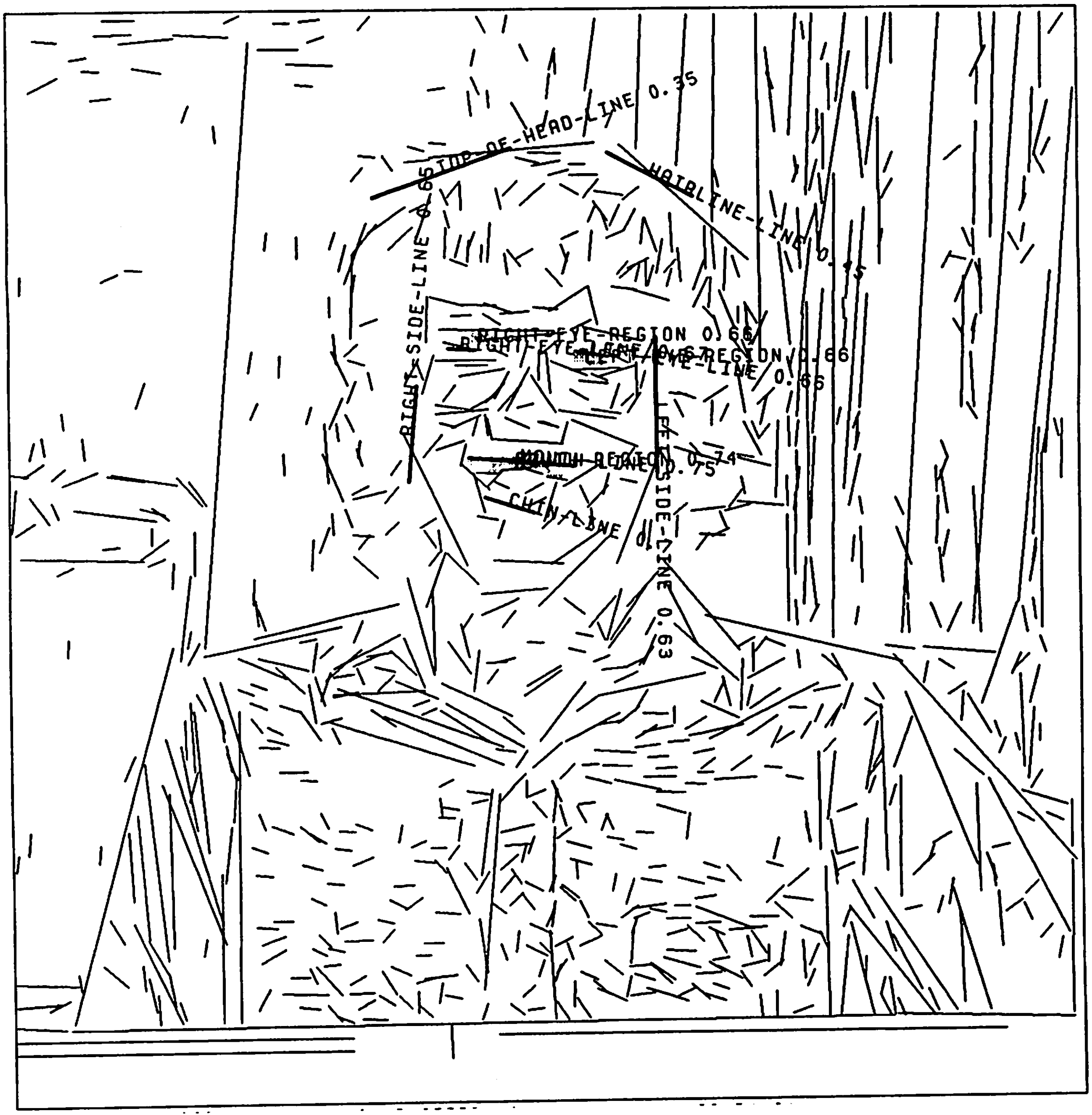


Figure 34: "Mike" model against "Mike" image 3: Best matches for each face part after two iterations of relaxation matching.

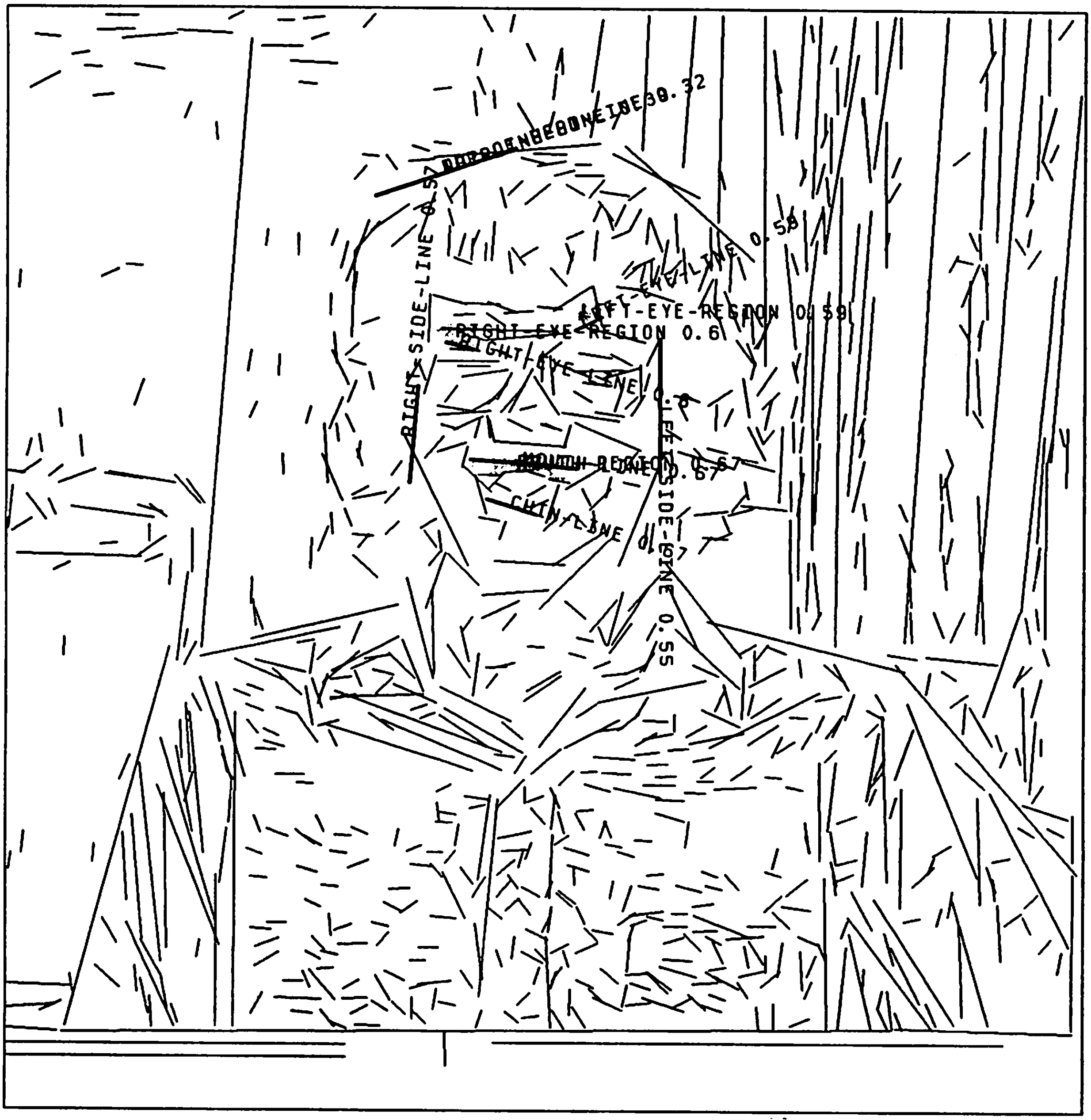


Figure 35: "Mike" model against "Mike" image 3: Best matches for each face part after three iterations of relaxation matching.

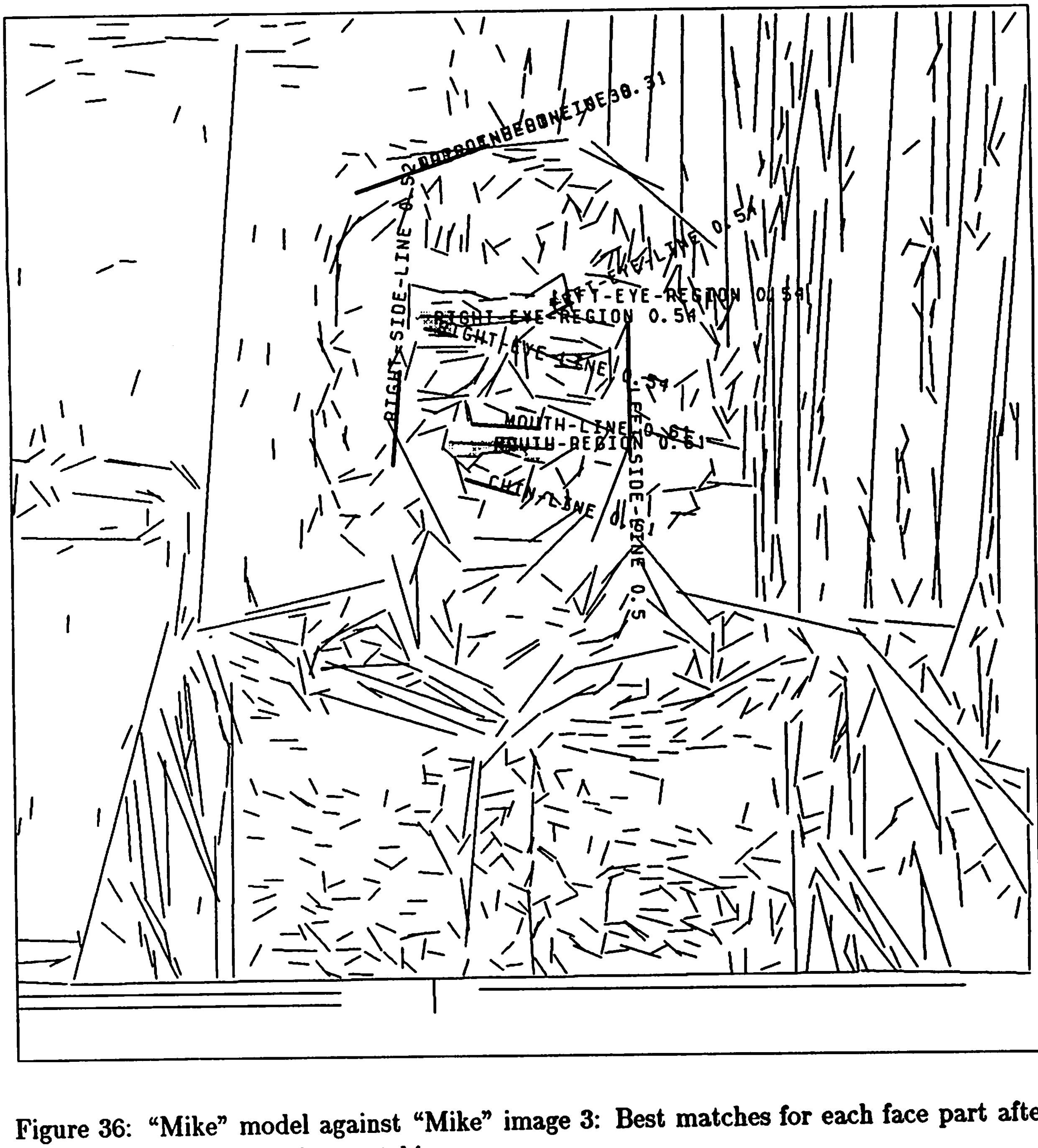


Figure 36: "Mike" model against "Mike" image 3: Best matches for each face part after four iterations of relaxation matching.

FACE MODELS

```
(define-model hiro-model
  (right-eye-line line -0.9 0.0
    (use-rule right-eye-lines-location-and-characteristics)
    (right-eye-region left-eye-line left-eye-region right-side-line
      mouth-line))
  (right-eye-region region -0.9 0.0
    (use-rule right-eye-regions-location-and-characteristics)
    (right-eye-line left-eye-line left-eye-region right-side-line
      mouth-line))
  (left-eye-line line 0.9 0.0
    (use-rule left-eye-lines-location-and-characteristics)
    (left-eye-region right-eye-line right-eye-region left-side-line
      mouth-line))
  (left-eye-region region 0.9 0.0
    (use-rule left-eye-regions-location-and-characteristics)
    (left-eye-line right-eye-line right-eye-region left-side-line
      mouth-line))
  (top-of-head-line line 0.0 3.7
    (use-rule top-of-head-lines-characteristics)
    (right-eye-line left-eye-line right-side-line left-side-line
      right-eye-region left-eye-region))
  (right-side-line line -2.0 -0.3
    (use-rule face-side-lines-characteristics)
    (top-of-head-line right-eye-line mouth-line right-eye-region))
  (left-side-line line 2.0 -0.3
    (use-rule face-side-lines-characteristics)
    (top-of-head-line left-eye-line mouth-line left-eye-region))
  (mouth-line line 0.0 -2.4
    (use-rule mouth-lines-location-and-characteristics)
    (mouth-region right-eye-line left-eye-line right-side-line
      left-side-line chin-line right-eye-region left-eye-region))
  (mouth-region region 0.0 -2.4
    (use-rule mouth-regions-location-and-characteristics)
    (mouth-line right-eye-line left-eye-line
      right-side-line left-side-line chin-line))
  (chin-line line 0.0 -3.3
    (use-rule chin-lines-characteristics)
    (mouth-line right-side-line left-side-line mouth-region))
);hiro-model
```

Figure 37: Model specification for "Hiro".

```
(define-model les-model
  (right-eye-line line -1.1 0.0
    (use-rule right-eye-lines-location-and-characteristics)
    (right-eye-region left-eye-line left-eye-region
      right-side-line hairline-line mouth-line))
  (right-eye-region region -1.1 0.0
    (use-rule right-eye-regions-location-and-characteristics)
    (right-eye-line left-eye-line left-eye-region
      right-side-line hairline-line mouth-line))
  (left-eye-line line 1.1 0.0
    (use-rule left-eye-lines-location-and-characteristics)
    (left-eye-region right-eye-line right-eye-region
      left-side-line hairline-line mouth-line))
  (left-eye-region region 1.1 0.0
    (use-rule left-eye-regions-location-and-characteristics)
    (left-eye-line right-eye-line right-eye-region
      left-side-line hairline-line mouth-line))
  (hairline-line line 0.0 2.5
    (use-rule hairline-lines-characteristics)
    (right-eye-line left-eye-line top-of-head-line
      right-eye-region left-eye-region))
  (top-of-head-line line 0.0 3.6
    (use-rule top-of-head-lines-characteristics)
    (hairline-line right-eye-line left-eye-line right-side-line
      left-side-line right-eye-region left-eye-region))
  (right-side-line line -1.8 -0.2
    (use-rule face-side-lines-characteristics)
    (top-of-head-line right-eye-line mouth-line right-eye-region))
  (left-side-line line 1.8 -0.2
    (use-rule face-side-lines-characteristics)
    (top-of-head-line left-eye-line mouth-line left-eye-region))
  (mouth-line line 0.0 -2.4
    (use-rule mouth-lines-location-and-characteristics)
    (mouth-region right-eye-line left-eye-line right-side-line
      left-side-line chin-line right-eye-region left-eye-region))
  (mouth-region region 0.0 -2.4
    (use-rule mouth-regions-location-and-characteristics)
    (mouth-line right-eye-line left-eye-line
      right-side-line left-side-line chin-line))
  (chin-line line 0.0 -4.2
    (use-rule chin-lines-characteristics)
    (mouth-line right-side-line left-side-line mouth-region))
);les-model
```

Figure 38: Model specification for "Les".

```
(define-model mike-model
  (right-eye-line line -1.1 0.0
    (use-rule right-eye-lines-location-and-characteristics)
    (right-eye-region left-eye-line right-side-line
      hairline-line mouth-line left-eye-region))
  (right-eye-region region -1.1 0.0
    (use-rule right-eye-regions-location-and-characteristics)
    (right-eye-line left-eye-line right-side-line
      hairline-line mouth-line left-eye-region))
  (left-eye-line line 1.1 0.0
    (use-rule left-eye-lines-location-and-characteristics)
    (left-eye-region right-eye-line left-side-line
      hairline-line mouth-line right-eye-region))
  (left-eye-region region 1.1 0.0
    (use-rule left-eye-regions-location-and-characteristics)
    (left-eye-line right-eye-line left-side-line
      hairline-line mouth-line right-eye-region))
  (hairline-line line 0.0 2.5
    (use-rule hairline-lines-characteristics)
    (right-eye-line left-eye-line top-of-head-line
      right-eye-region left-eye-region))
  (top-of-head-line line 0.0 3.7
    (use-rule top-of-head-lines-characteristics)
    (hairline-line right-eye-line left-eye-line right-side-line
      left-side-line right-eye-region left-eye-region))
  (right-side-line line -2.2 -1.3
    (use-rule face-side-lines-characteristics)
    (top-of-head-line right-eye-line mouth-line right-eye-region))
  (left-side-line line 2.2 -1.3
    (use-rule face-side-lines-characteristics)
    (top-of-head-line left-eye-line mouth-line left-eye-region))
  (mouth-line line 0.0 -2.1
    (use-rule mouth-lines-location-and-characteristics)
    (mouth-region right-eye-line left-eye-line right-side-line
      left-side-line chin-line right-eye-region left-eye-region))
  (mouth-region region 0.0 -2.1
    (use-rule mouth-regions-location-and-characteristics)
    (mouth-line right-eye-line left-eye-line right-side-line
      left-side-line chin-line right-eye-region left-eye-region))
  (chin-line line 0.0 -3.2
    (use-rule chin-lines-characteristics)
    (mouth-line right-side-line left-side-line mouth-region))
);mike-model
```

Figure 39: Model specification for "Mike".

A OBJECT HYPOTHESIS RULES

```
--------------------------------------------------------------------------------
********************************************************************************

                    ***R E G I O N   R U L E S***

********************************************************************************
--------------------------------------------------------------------------------


                    >>>>REGION CHARACTERISTICS<<<<
                    ------------------------------


"VIS$DISK:[TUTTLE.RULESYS.CURRENT_RULES]REGIONS_CHARACTERISTICS.RUL;1"

;;;There is a copy of these complex rules with EASIER names
;;; (they leave off the colon and everything following it) in
;;;VAX2::[TUTTLE.RULESYS.CURRENT_RULES]regions_characteristics_easy_names.rul.

mouth-region-characteristics:intensity-pixcount-horizontalrect
(:rule-list mouth-regions-intensity mouth-regions-pixcount
horizontal-rectangle)
(:score-form weighted-average-w-veto mouth-regions-intensity 4
mouth-regions-pixcount 3 horizontal-rectangle 2)
(:type . :complex)


eyebrow-region-characteristics:intensity-pixcount-horizontalrect
(:rule-list hair-regions-intensity very-small-pixcount horizontal-rectangle)
(:score-form weighted-average-w-veto hair-regions-intensity 3
very-small-pixcount 2 horizontal-rectangle 1)
(:type . :complex)


moustache-region-characteristics:intensity-pixcount-horizontalrect
(:rule-list hair-regions-intensity very-small-pixcount horizontal-rectangle)
(:score-form weighted-average-w-veto hair-regions-intensity 3
very-small-pixcount 2 horizontal-rectangle 1)
(:type . :complex)


beard-region-characteristics:intensity-pixcount-compactness
(:rule-list hair-regions-intensity hair-regions-pixcount
hair-regions-compactness)
(:score-form weighted-average-w-veto hair-regions-intensity 3
hair-regions-pixcount 2 hair-regions-compactness 1)
(:type . :complex)


hair-region-characteristics:intensity-pixcount-compactness
(:rule-list hair-regions-intensity hair-regions-pixcount
hair-regions-compactness)
(:score-form weighted-average-w-veto hair-regions-intensity 3
hair-regions-pixcount 2 hair-regions-compactness 1)
(:type . :complex)


ear-region-characteristics:intensity-pixcount
(:rule-list white-skin-regions-intensity very-small-pixcount)
(:score-form weighted-average-w-veto white-skin-regions-intensity 2
very-small-pixcount 3)
(:type . :complex)


nose-region-characteristics:intensity-pixcount
(:rule-list white-skin-regions-intensity very-small-pixcount)
(:score-form weighted-average-w-veto white-skin-regions-intensity 3
very-small-pixcount 2)
(:type . :complex)


eye-region-characteristics:intensity-pixcount-compactness
(:rule-list eye-regions-intensity very-small-pixcount compact-region)
(:score-form weighted-average-w-veto eye-regions-intensity 3
```

```
very-small-pixcount 3 compact-region 2)
(:type . :complex)

air-regions-pixcount
 Feature:   (extents pixcount)
 Rule:      (or high medium)
 Veto:      nil

hair-regions-compactness
 Feature:   (compact value)
 Rule:      (or low medium)
 Veto:      nil

spreadout-region
 Feature:   (compact value)
 Rule:      low
 Veto:      high

compact-region
 Feature:   (compact value)
 Rule:      high
 Veto:      low

mouth-regions-pixcount
 Feature:   (extents pixcount)
 Rule:      (or low medium)
 Veto:      high

very-large-pixcount
 Feature:   (extents pixcount)
 Rule:      high
 Veto:      low

arge-pixcount
 Feature:   (extents pixcount)
 Rule:      (or high medium)
 Veto:      low

very-small-pixcount
 Feature:   (extents pixcount)
 Rule:      low
 Veto:      high

small-pixcount
 Feature:   (extents pixcount)
 Rule:      (or low medium)
 Veto:      high

mouth-regions-intensity
 Feature:   (intensity mu)
 Rule:      (or low medium)
 Veto:      high

eye-regions-intensity
 Feature:   (intensity mu)
 Rule:      low
 Veto:      high

white-skin-regions-intensity
 Feature:   (intensity mu)
 Rule:      (or high medium)
 Veto:      low

hair-regions-intensity
 Feature:   (intensity mu)
 Rule:      low
 Veto:      high
-------------------------------------------------------------------------
-------------------------------------------------------------------------
```

```
                  ** ** ** LL OO CC AA TT II OO NN  ** ** **

;;;NOTE-->>>Some things to remember about ALL location rules!
;;;            1. The origin is in the top left (normal left) corner (because
;;;               the visions system flips the y-axis)
;;;            2. To add to the confusion, we refer to the person-in-the-
;;;               picture's left and right.
;;;
;;;                          _______
;;;        Jambo-------->/|||||||\
;;;                         |  _ _  |<--left
;;;                        {|  ● ●  |}
;;;                         |   ^   |
;;;            right-->|  \___/  |
;;;                          \______/

                  >>>Region Location and characteristics<<<<
                  ------------------------------------------

[TUTTLE.RULESYS.CURRENT_RULES]REGIONS_LOCATION_AND_CHARACTERISTICS.RUL

;;;combines the region-characteristics rules (above) and region-location
;;;rules (only the bounding-coord rules, see below, are used here).

hair-regions-location-and-characteristics
(:rule-list hair-regions-characteristics
hair-regions-location-using-bounding-coords)
(:score-form weighted-average-w-veto hair-regions-characteristics 3
hair-regions-location-using-bounding-coords 2)
(:type . :complex)

right-eye-regions-location-and-characteristics
(:rule-list eye-regions-characteristics
right-eye-regions-location-using-bounding-coords)
(:score-form weighted-average-w-veto eye-regions-characteristics 3
ight-eye-regions-location-using-bounding-coords 2)
(:type . :complex)

left-eye-regions-location-and-characteristics
(:rule-list eye-regions-characteristics
left-eye-regions-location-using-bounding-coords)
(:score-form weighted-average-w-veto eye-regions-characteristics 3
left-eye-regions-location-using-bounding-coords 2)
(:type . :complex)

right-ear-regions-location-and-characteristics
(:rule-list ear-regions-characteristics
right-ear-regions-location-using-bounding-coords)
(:score-form weighted-average-w-veto ear-regions-characteristics 3
right-ear-regions-location-using-bounding-coords 2)
(:type . :complex)

left-ear-regions-location-and-characteristics
(:rule-list ear-regions-characteristics
left-ear-regions-location-using-bounding-coords)
(:score-form weighted-average-w-veto ear-regions-characteristics 3
left-ear-regions-location-using-bounding-coords 2)
(:type . :complex)

right-eyebrow-regions-location-and-characteristics
(:rule-list eyebrow-regions-characteristics
right-eyebrow-regions-location-using-bounding-coords)
(:score-form weighted-average-w-veto eyebrow-regions-characteristics 3
right-eyebrow-regions-location-using-bounding-coords 2)
(:type . :complex)

ft-eyebrow-regions-location-and-characteristics
:rule-list eyebrow-regions-characteristics
left-eyebrow-regions-location-using-bounding-coords)
(:score-form weighted-average-w-veto eyebrow-regions-characteristics 3
left-eyebrow-regions-location-using-bounding-coords 2)
(:type . :complex)
```

```
mouth-regions-location-and-characteristics
(:rule-list mouth-regions-characteristics
 outh-regions-location-using-bounding-coords)
(:score-form weighted-average-w-veto mouth-regions-characteristics 3
mouth-regions-location-using-bounding-coords 2)
(:type . :complex)

moustache-regions-location-and-characteristics
(:rule-list moustache-regions-characteristics
moustache-regions-location-using-bounding-coords)
(:score-form weighted-average-w-veto moustache-regions-characteristics 3
moustache-regions-location-using-bounding-coords 2)
(:type . :complex)

nose-regions-location-and-characteristics
(:rule-list nose-regions-characteristics
nose-regions-location-using-bounding-coords)
(:score-form weighted-average-w-veto nose-regions-characteristics 3
nose-regions-location-using-bounding-coords 2)
(:type . :complex)

beard-regions-location-and-characteristics
(:rule-list beard-regions-characteristics
beard-regions-location-using-bounding-coords)
(:score-form weighted-average-w-veto beard-regions-characteristics 3
beard-regions-location-using-bounding-coords 2)
(:type . :complex)

----------------------------------------------------------------------
--------------------------------------------------------------------
```

>>>LOCATION USING COORDS<<<

```
"VIS$DISK:[TUTTLE.RULESYS.CURRENT_RULES]REGIONS_COORDS.RUL;64"

;;;these rules are for finding location based on bounding coordinates:
;;;what is a regions minimum and maximum x- and y-coordinates?
;;;They are written for the nielsen faces and assume a mask was used
;;;for the face area.  These rules use global variables that are set
;;;with respect to the mask bounds.  To setup these globals, use the
;;;functions in vax2::vis$disk:[tuttle.rulesys]rulesys.lsp.


>>>complex rules
--------------

;;;----------------BOUNDING-COORDS-COMPLEX-RULES-----------------
;;;     These rules are composed of the primitive rules defined
;;;     directly below this set of complex rules.  Hair and beard complex
;;;     rules are more complicated than the others because beard and hair
;;;     regions behave inconsistantly---sometimes they are sprawling and
;;;     sometimes they are broken into smaller pieces.
;;;-------------------------------------------------------------

;;bottom piece of beard---in case of fragmentation
beard-regions-coords-bottom
(:rule-list miny-in-bottom-third maxy-vhigh minx-in-right-third
maxx-in-left-third)
(:score-form weighted-average-w-veto miny-in-bottom-third 2 maxy-vhigh 3
minx-in-right-third 3 maxx-in-left-third 3)
(:type . :complex)

;;in case beard is one large region (not fragmented).
beard-regions-coords-full
(:rule-list maxy-vhigh minx-in-right-third maxx-in-left-third)
(:score-form weighted-average-w-veto maxy-vhigh 3 minx-in-right-third 2
maxx-in-left-third 2)
 :type . :complex)

;;in case of fragmentation
beard-regions-coords-left-side
(:rule-list maxy-in-bottom-half minx-in-left-third maxx-vhigh)
(:score-form weighted-average-w-veto maxy-in-bottom-half 2 minx-in-left-third
```

```
3 maxx-vhigh 3)
(:type . :complex)

;;in case of fragmentation
 eard-regions-coords-right-side
(:rule-list maxy-in-bottom-half minx-vlow maxx-in-right-third)
(:score-form weighted-average-w-veto maxy-in-bottom-half 2 minx-vlow 3
maxx-in-right-third 3)
(:type . :complex)


hair-regions-location-using-bounding-coords
(:rule-list hair-regions-coords-top hair-regions-coords-full
hair-regions-coords-right-side hair-regions-coords-left-side)
(:score-form max hair-regions-coords-top hair-regions-coords-full
hair-regions-coords-right-side hair-regions-coords-left-side)
(:type . :complex)


beard-regions-location-using-bounding-coords
(:rule-list beard-regions-coords-bottom beard-regions-coords-full
beard-regions-coords-right-side beard-regions-coords-left-side)
(:score-form max beard-regions-coords-bottom beard-regions-coords-full
beard-regions-coords-right-side beard-regions-coords-left-side)
(:type . :complex)

;;in case of fragmentation
hair-regions-coords-right-side
(:rule-list miny-in-top-half minx-vlow maxx-in-right-third)
(:score-form weighted-average-w-veto miny-in-top-half 2 minx-vlow 3
maxx-in-right-third 3)
(:type . :complex)

;;in case of fragmentation
hair-regions-coords-left-side
(:rule-list miny-in-top-half minx-in-left-third maxx-vhigh)
(:score-form weighted-average-w-veto miny-in-top-half 2 minx-in-left-third 3
maxx-vhigh 3)
(:type . :complex)

 ;in case of no fragmentation
 air-regions-coords-full
(:rule-list miny-vlow minx-vlow maxx-vhigh)
(:score-form weighted-average-w-veto miny-vlow 3 minx-vlow 2 maxx-vhigh 2)
(:type . :complex)

;;in case of fragmentation
hair-regions-coords-top
(:rule-list miny-vlow maxy-in-top-half minx-in-right-third maxx-in-left-third)
(:score-form weighted-average-w-veto miny-vlow 3 maxy-in-top-half 2
minx-in-right-third 3 maxx-in-left-third 3)
(:type . :complex)


mouth-regions-location-using-bounding-coords
(:rule-list miny-in-bottom-half maxy-in-bottom-half minx-in-right-half
maxx-in-left-half)
(:score-form weighted-average-w-veto miny-in-bottom-half 3 maxy-in-bottom-half
3 minx-in-right-half 2 maxx-in-left-half 2)
(:type . :complex)


moustache-regions-location-using-bounding-coords
(:rule-list miny-in-bottom-half maxy-in-bottom-half minx-in-right-half
maxx-in-left-half)
(:score-form weighted-average-w-veto miny-in-bottom-half 3 maxy-in-bottom-half
3 minx-in-right-half 2 maxx-in-left-half 2)
(:type . :complex)


nose-regions-location-using-bounding-coords
(:rule-list miny-in-middle-third maxy-in-middle-third minx-in-middle-third
maxx-in-middle-third)
(:score-form weighted-average-w-veto miny-in-middle-third 2
maxy-in-middle-third 2 minx-in-middle-third 3 maxx-in-middle-third 3)
(:type . :complex)


.eft-eye-regions-location-using-bounding-coords
(:rule-list miny-in-middle-third maxy-in-middle-third minx-in-middle-third
maxx-in-left-half)
(:score-form weighted-average-w-veto miny-in-middle-third 1
maxy-in-middle-third 1 minx-in-middle-third 1 maxx-in-left-half 1)
```

CLIP

```
(:type . :complex)

eft-eyebrow-regions-location-using-bounding-coords
(:rule-list miny-in-middle-third maxy-in-middle-third minx-in-middle-third
maxx-in-left-half)
(:score-form weighted-average-w-veto miny-in-middle-third 1
maxy-in-middle-third 1 minx-in-middle-third 1 maxx-in-left-half 1)
(:type . :complex)

right-eye-regions-location-using-bounding-coords
(:rule-list miny-in-middle-third maxy-in-middle-third minx-in-right-half
maxx-in-middle-third)
(:score-form weighted-average-w-veto miny-in-middle-third 1
maxy-in-middle-third 1 minx-in-right-half 1 maxx-in-middle-third 1)
(:type . :complex)

right-eyebrow-regions-location-using-bounding-coords
(:rule-list miny-in-middle-third maxy-in-middle-third minx-in-right-half
maxx-in-middle-third)
(:score-form weighted-average-w-veto miny-in-middle-third 1
maxy-in-middle-third 1 minx-in-right-half 1 maxx-in-middle-third 1)
(:type . :complex)


left-ear-regions-location-using-bounding-coords
(:rule-list miny-in-middle-third maxy-in-middle-third minx-in-left-third maxx-in-left-third)
(:score-form weighted-average-w-veto miny-in-middle-third 1 maxy-in-middle-third 1 minx-in-left-third 2 maxx-in-le
(:type . :complex)

right-ear-regions-location-using-bounding-coords
(:rule-list miny-in-middle-third maxy-in-middle-third minx-in-right-third maxx-in-right-third)
(:score-form weighted-average-w-veto miny-in-middle-third 1 maxy-in-middle-third 1 minx-in-right-third 2 maxx-in-r
(:type . :complex)

>>component rules

;;;------------BOUNDING-COORDS-PRIMITIVES-FOR-VHIGHS-AND-VLOWS------------
;;;vhigh and vlow are the outermost ninths of the mask (third of a third)
;;;these rules are like this:
;;;1.  from end to vhigh or vlow mark is 100.0.
;;;2.  from vhigh/vlow mark to vhigh/vlow mark +/- *medium-margin
;;;    slopes from 100.0 to 0.0.
;;;3.  remainder is -20.0.
;;;-------------------------------------------------------------

miny-vlow
(:feat-vals *mask-y-axis-vlow (plus:r *mask-y-axis-vlow *medium-margin) (
plus:r *mask-y-axis-vlow *medium-margin))
(:feature extents miny)
(:score-vals 100.0 0.0 -20.0)
(:type . :primitive)


minx-vlow
(:feat-vals *mask-x-axis-vlow (plus:r *mask-x-axis-vlow *medium-margin) (
plus:r *mask-x-axis-vlow *medium-margin))
(:feature extents minx)
(:score-vals 100.0 0.0 -20.0)
(:type . :primitive)


miny-vhigh
(:feat-vals (difference:r *mask-y-axis-vhigh *medium-margin) (difference:r
*mask-y-axis-vhigh *medium-margin) *mask-y-axis-vhigh)
(:feature extents miny)
(:score-vals -20.0 0.0 100.0)
(:type . :primitive)


minx-vhigh
:feat-vals (difference:r *mask-x-axis-vhigh *medium-margin) (difference:r
mask-x-axis-vhigh *medium-margin) *mask-x-axis-vhigh)
(:feature extents minx)
(:score-vals -20.0 0.0 100.0)
(:type . :primitive)
```

```
maxy-vlow
(:feat-vals *mask-y-axis-vlow (plus:r *mask-y-axis-vlow *medium-margin) (
plus:r *mask-y-axis-vlow *medium-margin))
(:feature extents maxy)
(:score-vals 100.0 0.0 -20.0)
(:type . :primitive)


maxx-vlow
(:feat-vals *mask-x-axis-vlow (plus:r *mask-x-axis-vlow *medium-margin) (
plus:r *mask-x-axis-vlow *medium-margin))
(:feature extents maxx)
(:score-vals 100.0 0.0 -20.0)
(:type . :primitive)


maxy-vhigh
(:feat-vals (difference:r *mask-y-axis-vhigh *medium-margin) (difference:r
*mask-y-axis-vhigh *medium-margin) *mask-y-axis-vhigh)
(:feature extents maxy)
(:score-vals -20.0 0.0 100.0)
(:type . :primitive)


maxx-vhigh
(:feat-vals (difference:r *mask-x-axis-vhigh *medium-margin) (difference:r
*mask-x-axis-vhigh *medium-margin) *mask-x-axis-vhigh)
(:feature extents maxx)
(:score-vals -20.0 0.0 100.0)
(:type . :primitive)

;;;--------COORD-PRIMITIVES-FOR-NON-MIDDLE-THIRDS (LEFT RIGHT TOP BOTTOM)----
;;;it's like this:
;;;1. the designated third gets 100.0
;;;2. in the other 2/3's, the midpoint to the start (or end) of the good third
;;;   slopes from 100.0 to 0.0
;;;3. and...the midpoint to through the rest gets -20.0.
;;;-----------------------------------------------------------------------

miny-in-top-third
(:feat-vals *mask-top-third *mask-y-midpoint *mask-y-midpoint)
(:feature extents miny)
(:score-vals 100.0 0.0 -20.0)
(:type . :primitive)


maxy-in-top-third
(:feat-vals *mask-top-third *mask-y-midpoint *mask-y-midpoint)
(:feature extents maxy)
(:score-vals 100.0 0.0 -20.0)
(:type . :primitive)


minx-in-right-third
(:feat-vals *mask-right-third *mask-x-midpoint *mask-x-midpoint)
(:feature extents minx)
(:score-vals 100.0 0.0 -20.0)
(:type . :primitive)


maxx-in-right-third
(:feat-vals *mask-right-third *mask-x-midpoint *mask-x-midpoint)
(:feature extents maxx)
(:score-vals 100.0 0.0 -20.0)
(:type . :primitive)


miny-in-bottom-third
(:feat-vals *mask-y-midpoint *mask-y-midpoint *mask-bottom-third)
(:feature extents miny)
(:score-vals -20.0 0.0 100.0)
(:type . :primitive)


maxy-in-bottom-third
(:feat-vals *mask-y-midpoint *mask-y-midpoint *mask-bottom-third)
(:feature extents maxy)
(:score-vals -20.0 0.0 100.0)
(:type . :primitive)

;;;------------BOUNDING COORD PRIMITIVES USING MIDDLE THIRD-------
;;;these rules for the middle third take into account whether the rule
;;;is for the min or max bounding coord.  For example, the rule for
```

```
;;;minx-in-middle-third is more lenient on the lower values in the
;;;histogram and maxx-in-middle-third is more lenient on the higher
;;;values in the histogram.  If this makes no sense, contemplate the 4 below
;;rules for a while.
;;
;;;an example...maxy-in-middle-third:
;;;1.  top third minus *small-margin gets -20.0 (remember y is 0 at
;;;    top---our y-coord is flipped)
;;;2.  the middle third with *small-margin as padding on both sides gets 100.0.
;;;3.  of the remaining portion (less than a third from the bottom), between
;;;    bottom-third plus *small-margin and *mask-y-axis-vhigh slopes from
;;;    100.0 to 0.0
;;;4.  and the portion between *mask-y-axis-vhigh and the end gets -20.0.
;;;
;;;One easy way to adjust these middle-third rules is to mess around with the global
;;;variables *small-margin *medium-margin and *large margin.  These globals
;;;are set by function set-more-mask-globals in vax2::[tuttle.rulesys]rulesys.evl
;;;for the standard nielsen images, small is typically 5 pixels, medium 10
;;;and large 20.
;;;------------------------------------------------------------

minx-in-left-third
(:feat-vals *mask-x-midpoint *mask-x-midpoint *mask-right-third)
(:feature extents minx)
(:score-vals -20.0 0.0 100.0)
(:type . :primitive)


maxx-in-left-third
(:feat-vals *mask-x-midpoint *mask-x-midpoint *mask-right-third)
(:feature extents maxx)
(:score-vals -20.0 0.0 100.0)
(:type . :primitive)


miny-in-middle-third
(:feat-vals *mask-y-axis-vlow *mask-y-axis-vlow (difference:r *mask-top-third
*small-margin) (plus:r *mask-bottom-third *small-margin) (plus:r
*mask-bottom-third *small-margin))
(:feature extents miny)
:score-vals -20.0 0.0 100.0 100.0 -20.0)
(:type . :primitive)


maxy-in-middle-third
(:feat-vals (difference:r *mask-top-third *small-margin) (difference:r
*mask-top-third *small-margin) (plus:r *mask-bottom-third *small-margin)
*mask-y-axis-vhigh *mask-y-axis-vhigh)
(:feature extents maxy)
(:score-vals -20.0 100.0 100.0 0.0 -20.0)
(:type . :primitive)


minx-in-middle-third
(:feat-vals *mask-x-axis-vlow *mask-x-axis-vlow (difference:r
*mask-right-third *small-margin) (plus:r *mask-left-third *small-margin) (
plus:r *mask-left-third *small-margin))
(:feature extents minx)
(:score-vals -20.0 0.0 100.0 100.0 -20.0)
(:type . :primitive)


maxx-in-middle-third
(:feat-vals (difference:r *mask-right-third *small-margin) (difference:r
*mask-right-third *small-margin) (plus:r *mask-left-third *small-margin)
*mask-x-axis-vhigh *mask-x-axis-vhigh)
(:feature extents maxx)
(:score-vals -20.0 100.0 100.0 0.0 -20.0)
(:type . :primitive)


;;;------------BOUNDING-COORD-RULES-USING-HALVES--------------------
;;;all of the HALF rules are like this...
;;;1. they give a 100 to all values in their half.
;;;2. they slope from 0.0 to 100.0  between the midpoint and the midpoint
;;;    +/- *medium-margin
;;3. they give -20 to all values NOT between the midpoint and the third
;;     mark in the OTHER half.
;;;------------------------------------------------------------

miny-in-top-half
(:feat-vals *mask-y-midpoint (plus:r *mask-y-midpoint *medium-margin) (plus:r
*mask-y-midpoint *medium-margin))
```

```
(:feature extents miny)
(:score-vals 100.0 0.0 -20.0)
(:type . :primitive)


maxy-in-top-half
(:feat-vals *mask-y-midpoint (plus:r *mask-y-midpoint *medium-margin) (plus:r
*mask-y-midpoint *medium-margin))
(:feature extents maxy)
(:score-vals 100.0 0.0 -20.0)
(:type . :primitive)


minx-in-right-half
(:feat-vals *mask-x-midpoint (plus:r *mask-x-midpoint *medium-margin) (plus:r
*mask-x-midpoint *medium-margin))
(:feature extents minx)
(:score-vals 100.0 0.0 -20.0)
(:type . :primitive)


maxx-in-right-half
(:feat-vals *mask-x-midpoint (plus:r *mask-x-midpoint *medium-margin) (plus:r
*mask-x-midpoint *medium-margin))
(:feature extents maxx)
(:score-vals 100.0 0.0 -20.0)
(:type . :primitive)


miny-in-bottom-half
(:feat-vals (difference:r *mask-y-midpoint *medium-margin) (difference:r
*mask-y-midpoint *medium-margin) *mask-y-midpoint)
(:feature extents miny)
(:score-vals -20.0 0.0 100.0)
(:type . :primitive)


maxy-in-bottom-half
(:feat-vals (difference:r *mask-y-midpoint *medium-margin) (difference:r
*mask-y-midpoint *medium-margin) *mask-y-midpoint)
 :feature extents maxy)
.:score-vals -20.0 0.0 100.0)
(:type . :primitive)


minx-in-left-half
(:feat-vals (difference:r *mask-x-midpoint *medium-margin) (difference:r
*mask-x-midpoint *medium-margin) *mask-x-midpoint)
(:feature extents minx)
(:score-vals -20.0 0.0 100.0)
(:type . :primitive)


maxx-in-left-half
(:feat-vals (difference:r *mask-x-midpoint *medium-margin) (difference:r
*mask-x-midpoint *medium-margin) *mask-x-midpoint)
(:feature extents maxx)
(:score-vals -20.0 0.0 100.0)
(:type . :primitive)

------------------------------------------------------------------------------
------------------------------------------------------------------------------


               >>>>LOCATION-USING-CENTROIDS<<<<<
               ---------------------------------


"VIS$DISK:[TUTTLE.RULESYS.CURRENT_RULES]REGIONS_CENTROIDS.RUL;7"

;;;centroid location rules to be used with mask.  global variables refered
;;;to below are setup by functions stored in
;;;vax2::vis$disk:[tuttle.rulesys]rulesys.evl.  Those functions are included
;;;in this listing (see the very bottom).

;;;NOTE-->>>these rules should be rewritten to be more restrictive.
 ;;These rules have gone unchanged because, for now, we are using the
;;;rules in regions_coords.rul for finding location.  Those rules have been
;;;rewritten to be more restrictive so, lif you want better centroid rules,
;;;use the coord rules as a model.  The centroid rules often picked up
;;;all sorts of riffraff (the hair centroid, for example, was often
;;;almost the same as the nose centroid).
```

```
>>complex rules
---------------

left-eye-regions-location-using-centroid
(:rule-list left-half-of-head middle-third-of-head-y-axis)
(:score-form weighted-average-w-veto left-half-of-head 3
   middle-third-of-head-y-axis 2)
(:type . :complex)

right-eye-regions-location-using-centroid
(:rule-list right-half-of-head middle-third-of-head-y-axis)
(:score-form weighted-average-w-veto right-half-of-head 3
  middle-third-of-head-y-axis 2)
(:type . :complex)

left-eyebrow-regions-location-using-centroid
(:rule-list left-half-of-head middle-third-of-head-y-axis)
(:score-form weighted-average-w-veto left-half-of-head 3
  middle-third-of-head-y-axis 2)
(:type . :complex)

right-eyebrow-regions-location-using-centroid
(:rule-list right-half-of-head middle-third-of-head-y-axis)
(:score-form weighted-average-w-veto right-half-of-head 3
  middle-third-of-head-y-axis 2)
(:type . :complex)

mouth-regions-location-using-centroid
(:rule-list bottom-half-of-head middle-third-of-head-x-axis)
(:score-form weighted-average-w-veto bottom-half-of-head 3
  middle-third-of-head-x-axis 2)
(:type . :complex)

moustache-regions-location-using-centroid
(:rule-list bottom-half-of-head middle-third-of-head-x-axis)
(:score-form weighted-average-w-veto bottom-half-of-head 3
  middle-third-of-head-x-axis 2)
(:type . :complex)

nose-regions-location-using-centroid
(:rule-list middle-third-of-head-y-axis middle-third-of-head-x-axis)
(:score-form weighted-average-w-veto middle-third-of-head-y-axis 2
  middle-third-of-head-x-axis 3)
(:type . :complex)

left-ear-regions-location-using-centroid
(:rule-list left-third-of-head middle-third-of-head-y-axis)
(:score-form weighted-average-w-veto left-third-of-head 3
  middle-third-of-head-y-axis 2)
(:type . :complex)

right-ear-regions-location-using-centroid
(:rule-list right-third-of-head middle-third-of-head-y-axis)
(:score-form weighted-average-w-veto right-third-of-head 3
  middle-third-of-head-y-axis 2)
(:type . :complex)

beard-regions-location-using-centroid
(:rule-list bottom-half-of-head middle-third-of-head-x-axis)
(:score-form weighted-average bottom-half-of-head 3
  middle-third-of-head-x-axis 2)
(:type . :complex)

air-regions-location-using-centroid
:rule-list top-half-of-head middle-third-of-head-x-axis)
(:score-form weighted-average top-half-of-head 3 middle-third-of-head-x-axis 2)
(:type . :complex)
```

```
>>>>component rules
------------------

;;;-------------------CENTROIDS MIDDLE THIRDS--------------------
;;they go like this:
;;;1.  the middle third along with *small-margin on each side gets 100.0.
;;;2.  On each side of that middle section, the values from *mask-<some>-third
;;;    +/- *small-margin to *mask-<some>-third +/- *mask-large-margin is
;;;    sloped from 100.0 to 0.0.
;;;3.  the remainder on each side gets -20.0.
;;;-------------------------------------------------------------------

middle-third-of-head-y-axis
(:feat-vals (difference:r *mask-top-third *large-margin)
  (difference:r *mask-top-third *large-margin) (difference:r *mask-top-third
  *small-margin) (plus:r *mask-bottom-third *small-margin) (plus:r
  *mask-bottom-third *large-margin) (plus:r *mask-bottom-third
  *large-margin))
(:feature centroid ybar)
(:score-vals -20.0 0.0 100.0 100.0 0.0 -20.0)
(:type . :primitive)


middle-third-of-head-x-axis
(:feat-vals (difference:r *mask-right-third *large-margin) (difference:r
  *mask-right-third *large-margin) (difference:r
  *mask-right-third *small-margin) (plus:r *mask-left-third *small-margin)
  (plus:r *mask-left-third *large-margin) (plus:r
  *mask-left-third *large-margin))
(:feature centroid xbar)
(:score-vals -20.0 0.0 100.0 100.0 0.0 -20.0)
(:type . :primitive)

;;;----------------CENTROIDS IN THE NON-MIDDLE THIRDS-----------
;;;these rules do it like this:
;;;1.  the designated third gets 100.0.
;;;2.  from the midpoint to the good third mark, values slope from 0.0 to 100.0
;;;3.  the midpoint and beyond get -20.0
;;;----------------------------------------------------------------

top-third-of-head
(:feat-vals *mask-top-third *mask-y-midpoint *mask-y-midpoint)
(:feature centroid ybar)
(:score-vals 100.0 0.0 -20.0)
(:type . :primitive)


right-third-of-head
(:feat-vals *mask-right-third *mask-x-midpoint *mask-x-midpoint)
(:feature centroid xbar)
(:score-vals 100.0 0.0 -20.0)
(:type . :primitive)


bottom-third-of-head
(:feat-vals *mask-y-midpoint *mask-y-midpoint *mask-bottom-third)
(:feature centroid ybar)
(:score-vals -20.0 0.0 100.0)
(:type . :primitive)


left-third-of-head
(:feat-vals *mask-x-midpoint *mask-x-midpoint *mask-left-third)
(:feature centroid xbar)
(:score-vals -20.0 0.0 100.0)
(:type . :primitive)


;;;------------------CENTROID RULES FOR HALVES----------------------
;;;these rules are as follows:
;;;1. designated half gets 100.0.
;;;2. the range between midpoint and the third mark in the OTHER half
;;;   slopes between 100 and 0.0
;;;3. and the remainder gets -20.0.
;;;-----------------------------------------------------------------

top-half-of-head
(:feat-vals *mask-y-midpoint *mask-bottom-third *mask-bottom-third)
(:feature centroid ybar)
(:score-vals 100.0 0.0 -20.0)
(:type . :primitive)
```

```
right-half-of-head
(:feat-vals *mask-x-midpoint *mask-left-third *mask-left-third)
(:feature centroid xbar)
(:score-vals 100.0 0.0 -20.0)
(:type . :primitive)


bottom-half-of-head
(:feat-vals *mask-top-third *mask-top-third *mask-y-midpoint)
(:feature centroid ybar)
(:score-vals -20.0 0.0 100.0)
(:type . :primitive)


left-half-of-head
(:feat-vals *mask-right-third *mask-right-third *mask-x-midpoint)
(:feature centroid xbar)
(:score-vals -20.0 0.0 100.0)
(:type . :primitive)

---------------------------------------------------------------------
---------------------------------------------------------------------

                >>>REGION-RECTANGLE-RULES<<<<
                -----------------------------



"VIS$DISK:[TUTTLE.RULESYS.CURRENT_RULES]REGIONS_RECTANGLES.RUL;4"

;;;these rules look for regions that are rectangular, i.e., within the
;;;region's smallest bounding rectangle, more pixels are inside the region
;;;than not.  The bounding rectangle is only calculated along the
;;;x- and y-coords; a diamond-shaped region will not be seen as
;;;very rectangular.

;;; These rules use functions stored in vax2::[tuttle.rulesys]rulesys_funs.lsp.
;;; See bottom of this listing for a printout of those fuuncions.


horizontal-rectangle
(:rule-list region-that-is-fat-and-short rectangular)
(:score-form weighted-average region-that-is-fat-and-short 3 rectangular 2)
(:trace-form lambda (region seg-dss seg-dss2) (princ Running) (princ (quote
:complex)) (princ rule) (princ (quote horizontal-rectangle)) (princ on) (princ
(cond ((eq (quote :lcomplex) (quote :complex)) line) (t region))) (printc
region) (let* (*left-margin (plus *left-margin 5) region-that-is-fat-and-short
(let (score ((lambda (region seg-dss seg-dss2) (primitive-rule-score (quote
flat-region) (quote (0.709901096941206 0.709901096941206 1.39805480506608
1.39805480506608 2.77437047929246 2.77437047929246 3.46254896134721
3.46254896134721)) (quote (-20.0 -20.0 -20.0 0.0 0.0 100.0 100.0 100.0))
region seg-dss)) region seg-dss seg-dss2)) (princ (quote
region-that-is-fat-and-short)) (princ returns) (printc score)) rectangular (
let (score ((lambda (region seg-dss seg-dss2) (primitive-rule-score (quote
rectangular-region) (quote (40.0 50.0)) (quote (-20.0 100.0)) region seg-dss))
region seg-dss seg-dss2)) (princ (quote rectangular)) (princ returns) (printc
score))) (weighted-average region-that-is-fat-and-short 3 rectangular 2)))
(:type . :complex)


region-that-is-fat-and-short
 Feature:  flat-region
 Rule:     high
 Veto:     low


rectangular
(:feat-vals 40.0 50.0)
(:feature . rectangular-region)
(:score-vals -20.0 100.0)
(:type . :primitive)

---------------------------------------------------------------------
---------------------------------------------------------------------


                >>>>REGION-HEAD-RULES<<<<<
                --------------------------
```

```
;;;these rules are used for choosing the regions that fall fully or partially
;;;within a rectangle mask designated by the global variables *top-of-head
;;;*bottom-of-head *right-side and *left-side.
;;;used with functions in vax2::[tuttle.rulesys]rulesys_funs.lsp
;;;(included in this listing, see very bottom).  The globals are set
;;;by functions (also below) in vax2::[tuttle.rulesys]rulesys.evl.

;;;for directions on using this function, see regions_head_area.rul.


head-regions
(:rule-list head-length-regions head-width-regions)
(:score-form weighted-average-w-veto head-length-regions 1 head-width-regions
1)
(:type . :complex)


head-length-regions
(:feat-vals (difference:r *top-of-head 5.0) *top-of-head *bottom-of-head (
plus:r *bottom-of-head 5.0))
(:feature centroid ybar)
(:score-vals -20.0 100.0 100.0 -20.0)
(:type . :primitive)


head-width-regions
(:feat-vals (difference:r *right-side 5.0) *right-side *left-side (plus:r
*left-side 5.0))
(:feature centroid xbar)
(:score-vals -20.0 100.0 100.0 -20.0)
(:type . :primitive)

-----------------------------------------------------------------------------
-----------------------------------------------------------------------------

                ***  RRRRR   to   LLLLLL *******
                ----------------------------------


 ;;The rulenames indicate the parameters limiting the min and max amount
;;;that the line can intersect the region: mouth-characteristics-rtol-
;;;03-min-08-max means that a line must intersect between 30% and 80% of the
;;;region to get a score of 100.  All other regions get (I think) -20.0.
;;;There are files with the same rules but with different min's and max's
;;;in vax2::[tuttle.rulesys.current_rules] for testing purposes but
;;;03 min and 08 max seem to work the best.


[TUTTLE.RULESYS.CURRENT_RULES]REGIONS_CHARACTERISTICS_RTOL_03_10.RUL

;;;do rtol on regions_characteristics.rul rules. Must intersect between 30% and
;;;100% to get a good score.

eye-regions-char-rtol-03-min-10-max
(:combo-form 0.3 1.0)
(:map-rule . eye-regions-characteristics)
(:min-max-filter . max-score)
(:type . :r-to-l)


nose-regions-char-rtol-03-min-10-max
(:combo-form 0.3 1.0)
(:map-rule . nose-regions-characteristics)
(:min-max-filter . max-score)
(:type . :r-to-l)


moustache-regions-char-rtol-03-min-10-max
(:combo-form 0.3 1.0)
(:map-rule . moustache-regions-characteristics)
(:min-max-filter . max-score)
(:type . :r-to-l)


 r-regions-char-rtol-03-min-10-max
 .combo-form 0.3 1.0)
(:map-rule . ear-regions-characteristics)
(:min-max-filter . max-score)
(:type . :r-to-l)
```

```
hair-regions-char-rtol-03-min-10-max
(:combo-form 0.3 1.0)
(:map-rule . hair-regions-characteristics)
(:min-max-filter . max-score)
(:type . :r-to-l)

beard-regions-char-rtol-03-min-10-max
(:combo-form 0.3 1.0)
(:map-rule . beard-regions-characteristics)
(:min-max-filter . max-score)
(:type . :r-to-l)

eyebrow-regions-char-rtol-03-min-10-max
(:combo-form 0.3 1.0)
(:map-rule . eyebrow-regions-characteristics)
(:min-max-filter . max-score)
(:type . :r-to-l)

mouth-regions-char-rtol-03-min-10-max
(:combo-form 0.3 1.0)
(:map-rule . mouth-regions-characteristics)
(:min-max-filter . max-score)
(:type . :r-to-l)
------------------------------------------------------------------------
------------------------------------------------------------------------

"VIS$DISK:[TUTTLE.RULESYS.CURRENT_RULES]REGIONS_LOC_CHAR_RTOL_03_08.RUL;4"

;;;lines which intersect the regions found by rules in
;;;regions_location_and_characteristics.rul

left-eye-regions-lc-rtol-03-min-08-max
(:combo-form . max-score)
(:map-rule . left-eye-regions-location-and-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :r-to-l)

right-eye-regions-lc-rtol-03-min-08-max
(:combo-form . max-score)
(:map-rule . right-eye-regions-location-and-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :r-to-l)

nose-regions-lc-rtol-03-min-08-max
(:combo-form . max-score)
(:map-rule . nose-regions-location-and-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :r-to-l)

moustache-regions-lc-rtol-03-min-08-max
(:combo-form . max-score)
(:map-rule . moustache-regions-location-and-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :r-to-l)

left-ear-regions-lc-rtol-03-min-08-max
(:combo-form . max-score)
(:map-rule . left-ear-regions-location-and-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :r-to-l)

right-ear-regions-lc-rtol-03-min-08-max
(:combo-form . max-score)
(:map-rule . right-ear-regions-location-and-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :r-to-l)

air-regions-lc-rtol-03-min-08-max
(:combo-form . max-score)
(:map-rule . hair-regions-location-and-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :r-to-l)
```

```
beard-regions-lc-rtol-03-min-08-max
(:combo-form . max-score)
(:map-rule . beard-regions-location-and-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :r-to-l)

left-eyebrow-regions-lc-rtol-03-min-08-max
(:combo-form . max-score)
(:map-rule . left-eyebrow-regions-location-and-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :r-to-l)

right-eyebrow-regions-lc-rtol-03-min-08-max
(:combo-form . max-score)
(:map-rule . right-eyebrow-regions-location-and-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :r-to-l)

mouth-regions-lc-rtol-03-min-08-max
(:combo-form . max-score)
(:map-rule . mouth-regions-location-and-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :r-to-l)
---------------------------------------------------------------------------
---------------------------------------------------------------------------
```

```
---------------------------------------------------------------------
*********************************************************************
                       L I N E    R U L E S
*********************************************************************
---------------------------------------------------------------------

              >>>>>>>>>>>>CHARACTERISTICS<<<<<<<<<<<<<<<
              ------------------------------------------


"VIS$DISK:[TUTTLE.RULESYS.CURRENT_RULES]LINES_CHARACTERISTICS.RUL;3"


nose-bridge-lines-characteristics
(:rule-list face-lines-w-short-length vertical-lines)
(:score-form weighted-average-w-veto face-lines-w-short-length 2
vertical-lines 3)
(:type . :lcomplex)


nose-end-lines-characteristics
(:rule-list face-lines-w-very-short-length horizontal-lines)
(:score-form weighted-average-w-veto face-lines-w-very-short-length 2
horizontal-lines 3)
(:type . :lcomplex)


beard-lines-length
(:rule-list face-lines-w-long-length face-lines-w-medium-length)
(:score-form max face-lines-w-medium-length face-lines-w-long-length)
(:type . :lcomplex)


beard-lines-characteristics
(:rule-list beard-lines-length beard-theta)
(:score-form weighted-average-w-veto beard-lines-length 1 beard-theta 1)
(:type . :lcomplex)


moustache-lines-characteristics
(:rule-list face-lines-w-medium-short-length horizontal-lines)
(:score-form weighted-average-w-veto face-lines-w-medium-short-length 2
horizontal-lines 3)
(:type . :lcomplex)


eyebrow-lines-characteristics
(:rule-list face-lines-w-short-length horizontal-lines)
(:score-form weighted-average-w-veto face-lines-w-short-length 2
horizontal-lines 3)
(:type . :lcomplex)


top-of-head-lines-characteristics
(:rule-list top-of-head-lines-length horizontal-lines)
(:score-form weighted-average-w-veto top-of-head-lines-length 2
horizontal-lines 3)
(:type . :lcomplex)


top-of-head-lines-length
(:rule-list face-lines-w-long-length face-lines-w-medium-length)
(:score-form max face-lines-w-medium-length face-lines-w-long-length)
(:type . :lcomplex)


hairline-lines-length
(:rule-list face-lines-w-long-length face-lines-w-medium-length)
(:score-form max face-lines-w-medium-length face-lines-w-long-length)
(:type . :lcomplex)


hairline-lines-characteristics
(:rule-list hairline-lines-length horizontal-lines)
(:score-form weighted-average-w-veto hairline-lines-length 2 horizontal-lines
3)
```

```
(:type . :lcomplex)

chin-lines-characteristics
(:rule-list face-lines-w-medium-short-length horizontal-lines)
(:score-form weighted-average-w-veto face-lines-w-medium-short-length 2
horizontal-lines 3)
(:type . :lcomplex)

face-side-lines-length
(:rule-list face-lines-w-long-length face-lines-w-medium-length)
(:score-form max face-lines-w-long-length face-lines-w-medium-length)
(:type . :lcomplex)

face-side-lines-characteristics
(:rule-list face-side-lines-length vertical-lines)
(:score-form weighted-average-w-veto face-side-lines-length 2 vertical-lines 3)
(:type . :lcomplex)

mouth-lines-length
(:rule-list face-lines-w-medium-short-length face-lines-w-medium-length)
(:score-form max face-lines-w-medium-short-length face-lines-w-medium-length)
(:type . :lcomplex)

mouth-lines-characteristics
(:rule-list horizontal-lines mouth-lines-length)
(:score-form weighted-average-w-veto mouth-lines-length 2 horizontal-lines 3)
(:type . :lcomplex)

eye-lines-characteristics
(:rule-list face-lines-w-short-length horizontal-lines)
(:score-form weighted-average-w-veto face-lines-w-short-length 2
horizontal-lines 3)
(:type . :lcomplex)

orizontal-lines
 Feature:  (line theta)
 Rule:     (or vlow vhigh)
 Veto:     medium

vertical-lines
 Feature:  (line theta)
 Rule:     medium
 Veto:     (or low high)

beard-theta
 Feature:  (line theta)
 Rule:     (or low high)
 Veto:     nil

;;;-----------------------------------------------------------------
;;;The length rules use absolute values (the histogram gets easily skewed by
;;;weirdness in the image like lots of long lines from flapping curtains
;;;in one image and lots of short lines from a plaid shirt in another
;;;image) which should some day incorporate a scale factor figured from
;;;range data or something.  I got the values used here by just measuring
;;;lines in the actual face images we have been using (vax8::[tuttle.pics]
;;;hirol.dat, debl.dat, bobl.dat, etc.)

;;;There is overlap between many of these primitive line length rules.
;;;For example, face-lines-w-medium-long-length overlaps with both
;;;face-lines-w-medium-length and face-linew-w-long-length.  If you
;;;dont like it, feel free to write your own rules.
-----------------------------------------------------------------

face-lines-w-short-length
(:feat-vals 0.0 0.0 14.0 16.0)
(:feature line length)
(:score-vals -20.0 100.0 100.0 -20.0)
 type . :lprimitive)

face-lines-w-medium-length
(:feat-vals 7.0 10.0 26.0 30.0)
(:feature line length)
```

```
(:score-vals -20.0 100.0 100.0 -20.0)
(:type . :lprimitive)


 ace-lines-w-long-length
(:feat-vals 16.0 20.0 40.0 60.0)
(:feature line length)
(:score-vals -20.0 100.0 100.0 -20.0)
(:type . :lprimitive)


face-lines-w-very-short-length
(:feat-vals 0.0 0.0 8.0 10.0)
(:feature line length)
(:score-vals -20.0 100.0 100.0 -20.0)
(:type . :lprimitive)


face-lines-w-medium-short-length
(:feat-vals 5.0 7.0 18.0 20.0)
(:feature line length)
(:score-vals -20.0 100.0 100.0 -20.0)
(:type . :lprimitive)


face-lines-w-medium-long-length
(:feat-vals 14.0 18.0 28.0 32.0)
(:feature line length)
(:score-vals -20.0 100.0 100.0 -20.0)
(:type . :lprimitive)


face-lines-w-very-long-length
(:feat-vals 22.0 28.0 55.0 60.0)
(:feature line length)
(:score-vals -20.0 100.0 100.0 -10.0)
(:type . :lprimitive)

-------------------------------------------------------------------------------
-------------------------------------------------------------------------------


                    >>>>LINE LOCATION RULES<<<<
                   -----------------------------




;;;See toplevel comments on region location rules info on flipped
;;;y-coord and flippe right and left.

                >>>>>LINE LOCATION<<<<<
                -----------------------




"VIS$DISK:[TUTTLE.RULESYS.CURRENT_RULES]LINES_LOCATION.RUL;4"


;;;these rules were written by Rich Weiss and renamed by Julianne Tuttle.
;;;The functions called are in [weiss.vis]line_location.evl (and printed
;;;near the bottom of this listing.

nose-lines-location
(:rule-list middle-lines-x nose-lines-y)
(:score-form weighted-average-w-veto middle-lines-x 3 nose-lines-y 2)
(:type . :lcomplex)


mouth-lines-location
(:rule-list middle-lines-x mouth-lines-y)
 .score-form weighted-average-w-veto middle-lines-x 3 mouth-lines-y 2)
(:type . :lcomplex)


left-eye-lines-location
(:rule-list left-eye-lines-x eye-lines-y)
```

```
(:score-form weighted-average-w-veto left-eye-lines-x 2 eye-lines-y 3)
(:type . :lcomplex)

ight-eye-lines-location
(:rule-list right-eye-lines-x eye-lines-y)
(:score-form weighted-average-w-veto right-eye-lines-x 2 eye-lines-y 3)
(:type . :lcomplex)

left-eye-lines-x
(:feat-vals *mask-x-midpoint *mask-x-midpoint *mask-x-axis-vhigh *left-side)
(:feature . mid-point-x)
(:score-vals -20.0 100.0 100.0 0.0)
(:type . :lprimitive)

right-eye-lines-x
(:feat-vals *right-side *mask-x-axis-vlow *mask-x-midpoint *mask-x-midpoint)
(:feature . mid-point-x)
(:score-vals 0.0 100.0 100.0 -20.0)
(:type . :lprimitive)

eye-lines-y
(:feat-vals *mask-y-axis-vlow *mask-y-axis-vlow *mask-y-midpoint
*mask-y-midpoint)
(:feature . mid-point-y)
(:score-vals -20.0 100.0 100.0 -20.0)
(:type . :lprimitive)

chin-lines-y
(:feat-vals *mask-y-midpoint *mask-y-midpoint *mask-bottom-third)
(:feature . imd-point-y)
(:score-vals -20.0 0.0 100.0)
(:type . :lprimitive)

toh-lines-y
:feat-vals *mask-top-third *mask-y-midpoint *mask-y-midpoint)
.:feature . mid-point-y)
(:score-vals 100.0 0.0 -20.0)
(:type . :lprimitive)

nose-lines-y
(:feat-vals *mask-y-midpoint *mask-y-midpoint *mask-bottom-third
*mask-bottom-third)
(:feature . mid-point-y)
(:score-vals -20.0 100.0 100.0 -20.0)
(:type . :lprimitive)

middle-lines-x
(:feat-vals *mask-right-third *mask-right-third *mask-left-third
*mask-left-third)
(:feature . mid-point-x)
(:score-vals -20.0 100.0 100.0 -20.0)
(:type . :lprimitive)

left-eye-lines-x
(:feat-vals *mask-x-midpoint *mask-x-midpoint *mask-x-axis-vhigh *left-side)
(:feature . mid-point-x)
(:score-vals -20.0 100.0 100.0 0.0)
(:type . :lprimitive)

right-eye-lines-x
(:feat-vals *right-side *mask-x-axis-vlow *mask-x-midpoint *mask-x-midpoint)
(:feature . mid-point-x)
(:score-vals 0.0 100.0 100.0 -20.0)
(:type . :lprimitive)

ye-lines-y
:feat-vals *mask-y-axis-vlow *mask-y-axis-vlow *mask-y-midpoint
*mask-y-midpoint)
(:feature . mid-point-y)
(:score-vals -20.0 100.0 100.0 -20.0)
(:type . :lprimitive)
```

```
mouth-lines-y
(:feat-vals *mask-y-midpoint *mask-y-midpoint *mask-y-axis-vhigh
 mask-y-axis-vhigh)
 :feature . mid-point-y)
(:score-vals -20.0 100.0 100.0 -20.0)
(:type . :lprimitive)
------------------------------------------------------------------------
------------------------------------------------------------------------
```

[TUTTLE.RULESYS.CURRENT_RULES]LINES_CHARACTERISTICS_AND_LOCATION.RUL;

```
right-eye-lines-location-and-characteristics
(:rule-list right-eye-lines-location eye-lines-characteristics)
(:score-form weighted-average-w-veto eye-lines-characteristics 2
right-eye-lines-location 3)
(:type . :lcomplex)

right-eyebrow-lines-location-and-characteristics
(:rule-list right-eyebrow-lines-location eyebrow-lines-characteristics)
(:score-form weighted-average-w-veto eyebrow-lines-characteristics 2
right-eyebrow-lines-location 3)
(:type . :lcomplex)

right-ear-lines-location-and-characteristics
(:rule-list right-ear-lines-location ear-lines-characteristics)
(:score-form weighted-average-w-veto ear-lines-characteristics 2
right-ear-lines-location 3)
(:type . :lcomplex)

left-eye-lines-location-and-characteristics
(:rule-list left-eye-lines-location eye-lines-characteristics)
 :score-form weighted-average-w-veto eye-lines-characteristics 2
 eft-eye-lines-location 3)
(:type . :lcomplex)

left-eyebrow-lines-location-and-characteristics
(:rule-list left-eyebrow-lines-location eyebrow-lines-characteristics)
(:score-form weighted-average-w-veto eyebrow-lines-characteristics 2
left-eyebrow-lines-location 3)
(:type . :lcomplex)

left-ear-lines-location-and-characteristics
(:rule-list left-ear-lines-location ear-lines-characteristics)
(:score-form weighted-average-w-veto ear-lines-characteristics 2
left-ear-lines-location 3)
(:type . :lcomplex)

nose-end-lines-location-and-characteristics
(:rule-list nose-end-lines-location nose-end-lines-characteristics)
(:score-form weighted-average-w-veto nose-end-lines-characteristics 2
nose-end-lines-location 3)
(:type . :lcomplex)

nose-bridge-lines-location-and-characteristics
(:rule-list nose-bridge-lines-location nose-bridge-lines-characteristics)
(:score-form weighted-average-w-veto nose-bridge-lines-characteristics 2
nose-bridge-lines-location 3)
(:type . :lcomplex)

beard-lines-location-and-characteristics
(:rule-list beard-lines-location beard-lines-characteristics)
(:score-form weighted-average-w-veto beard-lines-location 2
 eard-lines-characteristics 3)
  type . :lcomplex)

moustache-lines-location-and-characteristics
(:rule-list moustache-lines-location moustache-lines-characteristics)
(:score-form weighted-average-w-veto moustache-lines-location 2
```

```
moustache-lines-characteristics 3)
(:type . :lcomplex)

 airline-lines-location-and-characteristics
(:rule-list hairline-lines-location hairline-lines-characteristics)
(:score-form weighted-average-w-veto hairline-lines-characteristics 2
hairline-lines-location 3)
(:type . :lcomplex)

top-of-head-lines-location-and-characteristics
(:rule-list top-of-head-lines-location top-of-head-lines-characteristics)
(:score-form weighted-average-w-veto top-of-head-lines-characteristics 2
top-of-head-lines-location 3)
(:type . :lcomplex)

mouth-lines-location-and-characteristics
(:rule-list mouth-lines-characteristics mouth-lines-location)
(:score-form weighted-average-w-veto mouth-lines-characteristics 2
mouth-lines-location 3)
(:type . :lcomplex)
---------------------------------------------------------------------------
---------------------------------------------------------------------------

"VIS0DISK:[TUTTLE.RULESYS.CURRENT_RULES]LINES_LOC_CHAR_RTOL_03_08.RUL;1"

;;;these combine line-location, line-characteristics and
;;;region-rtol (which finds lines intersecting good regions) rules.

right-eye-lines-char-loc-and-rtol-0308
(:rule-list right-eye-lines-location eye-lines-characteristics
right-eye-lc-regions-rtol-03-min-08-max)
(:score-form weighted-average-w-veto eye-lines-characteristics 3
right-eye-lines-location 3 right-eye-lc-regions-rtol-03-min-08-max 2)
(:type . :lcomplex)

 ight-ear-lines-char-loc-and-rtol-0308
(:rule-list right-ear-lines-location ear-lines-characteristics
right-ear-lc-regions-rtol-03-min-08-max)
(:score-form weighted-average-w-veto ear-lines-characteristics 3
right-ear-lines-location 3 right-ear-lc-regions-rtol-03-min-08-max 2)
(:type . :lcomplex)

right-eyebrow-lines-char-loc-and-rtol-0308
(:rule-list right-eyebrow-lines-location eyebrow-lines-characteristics
right-eyebrow-lc-regions-rtol-03-min-08-max)
(:score-form weighted-average-w-veto eyebrow-lines-characteristics 3
right-eyebrow-lines-location 3 right-eyebrow-lc-regions-rtol-03-min-08-max 2)
(:type . :lcomplex)

left-eye-lines-char-loc-and-rtol-0308
(:rule-list left-eye-lines-location eye-lines-characteristics
left-eye-lc-regions-rtol-03-min-08-max)
(:score-form weighted-average-w-veto eye-lines-characteristics 3
left-eye-lines-location 3 left-eye-lc-regions-rtol-03-min-08-max 2)
(:type . :lcomplex)

left-ear-lines-char-loc-and-rtol-0308
(:rule-list left-ear-lines-location ear-lines-characteristics
left-ear-lc-regions-rtol-03-min-08-max)
(:score-form weighted-average-w-veto ear-lines-characteristics 3
left-ear-lines-location 3 left-ear-lc-regions-rtol-03-min-08-max 2)
(:type . :lcomplex)

left-eyebrow-lines-char-loc-and-rtol-0308
(:rule-list left-eyebrow-lines-location eyebrow-lines-characteristics
left-eyebrow-lc-regions-rtol-03-min-08-max)
(:score-form weighted-average-w-veto eyebrow-lines-characteristics 3
 eft-eyebrow-lines-location 3 left-eyebrow-lc-regions-rtol-03-min-08-max 2)
 :type . :lcomplex)

nose-end-lines-char-loc-and-rtol-0308
(:rule-list nose-end-lines-location nose-end-lines-characteristics
nose-lc-regions-rtol-03-min-08-max)
```

```
(:score-form weighted-average-w-veto nose-end-lines-characteristics 3
nose-end-lines-location 3 nose-lc-regions-rtol-03-min-08-max 2)
(:type . :lcomplex)

nose-bridge-lines-char-loc-and-rtol-0308
(:rule-list nose-bridge-lines-location nose-bridge-lines-characteristics
nose-lc-regions-rtol-03-min-08-max)
(:score-form weighted-average-w-veto nose-bridge-lines-characteristics 3
nose-bridge-lines-location 3 nose-lc-regions-rtol-03-min-08-max 2)
(:type . :lcomplex)

beard-lines-char-loc-and-rtol-0308
(:rule-list beard-lines-location beard-lines-characteristics
beard-lc-regions-rtol-03-min-08-max)
(:score-form weighted-average-w-veto beard-lines-location 3
beard-lines-characteristics 3 beard-lc-regions-rtol-03-min-08-max 2)
(:type . :lcomplex)

moustache-lines-char-loc-and-rtol-0308
(:rule-list moustache-lines-location moustache-lines-characteristics
moustache-lc-regions-rtol-03-min-08-max)
(:score-form weighted-average-w-veto moustache-lines-location 3
moustache-lines-characteristics 3 moustache-lc-regions-rtol-03-min-08-max 2)
(:type . :lcomplex)

hairline-lines-char-loc-and-rtol-0308
(:rule-list hairline-lines-location hairline-lines-characteristics
hair-lc-regions-rtol-03-min-08-max)
(:score-form weighted-average-w-veto hairline-lines-characteristics 3
hairline-lines-location 3 hair-lc-regions-rtol-03-min-08-max 2)
(:type . :lcomplex)

top-of-head-lines-char-loc-and-rtol-0308
(:rule-list top-of-head-lines-location top-of-head-lines-characteristics
hair-lc-regions-rtol-03-min-08-max)
(:score-form weighted-average-w-veto top-of-head-lines-characteristics 3
top-of-head-lines-location 3 hair-lc-regions-rtol-03-min-08-max 2)
(:type . :lcomplex)

mouth-lines-char-loc-and-rtol-0308
(:rule-list mouth-lines-characteristics mouth-lines-location
mouth-lc-regions-rtol-03-min-08-max)
(:score-form weighted-average-w-veto mouth-lines-characteristics 3
mouth-lines-location mouth-lc-regions-rtol-03-min-08-max 2)
(:type . :lcomplex)
-----------------------------------------------------------------
-----------------------------------------------------------------


                    >>>>L to R<<<<
                    --------------


"VIS8DISK:[TUTTLE.RULESYS.CURRENT_RULES]LINES_CHARACTERISTICS_LTOR_03_08.RUL;4"

;;;do ltor on rules in lines_characteristics.rul.

hairline-lines-char-ltor-03-08
(:combo-form . max-score)
(:map-rule . hairline-lines-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :l-to-r)

ear-lines-char-ltor-03-08
(:combo-form . max-score)
(:map-rule . ear-lines-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :l-to-r)

side-lines-char-ltor-03-08
(:combo-form . max-score)
(:map-rule . side-lines-characteristics)
```

```
(:min-max-filter 0.3 0.8)
(:type . :l-to-r)

 eard-lines-char-ltor-03-08
(:combo-form . max-score)
(:map-rule . beard-lines-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :l-to-r)

eyebrow-lines-char-ltor-03-08
(:combo-form . max-score)
(:map-rule . eyebrow-lines-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :l-to-r)

eye-lines-char-ltor-03-08
(:combo-form . max-score)
(:map-rule . eye-lines-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :l-to-r)

chin-lines-char-ltor-03-08
(:combo-form . max-score)
(:map-rule . chin-lines-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :l-to-r)

moustache-lines-char-ltor-03-08
(:combo-form . max-score)
(:map-rule . moustache-lines-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :l-to-r)

mouth-lines-char-ltor-03-08
 :combo-form . max-score)
 :map-rule . mouth-lines-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :l-to-r)

nose-end-lines-char-ltor-03-08
(:combo-form . max-score)
(:map-rule . nose-end-lines-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :l-to-r)

nose-bridge-lines-char-ltor-03-08
(:combo-form . max-score)
(:map-rule . nose-bridge-lines-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :l-to-r)

toh-lines-char-ltor-03-08
(:combo-form . max-score)
(:map-rule . top-of-head-lines-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :l-to-r)
------------------------------------------------------------------------
------------------------------------------------------------------------

"VIS$DISK:[TUTTLE.RULESYS.CURRENT_RULES]LINES_LOC_CHAR_LTOR_03_08.RUL;3"

;;;do ltor on rules in lines_location_and_characteristics.rul

hairline-loc-and-char-lines-ltor-03-08
(:combo-form . max-score)
(:map-rule . hairline-lines-location-and-characteristics)
(:min-max-filter 0.3 0.8)
  type . :l-to-r)

left-ear-loc-and-char-lines-ltor-03-08
(:combo-form . max-score)
(:map-rule . left-ear-lines-location-and-characteristics)
```

```
(:min-max-filter 0.3 0.8)
(:type . :l-to-r)

ight-ear-loc-and-char-lines-ltor-03-08
(:combo-form . max-score)
(:map-rule . right-ear-lines-location-and-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :l-to-r)

side-loc-and-char-lines-ltor-03-08
(:combo-form . max-score)
(:map-rule . side-lines-location-and-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :l-to-r)

beard-loc-and-char-lines-ltor-03-08
(:combo-form . max-score)
(:map-rule . beard-lines-location-and-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :l-to-r)

left-eyebrow-loc-and-char-lines-ltor-03-08
(:combo-form . max-score)
(:map-rule . left-eyebrow-lines-location-and-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :l-to-r)

left-eye-loc-and-char-lines-ltor-03-08
(:combo-form . max-score)
(:map-rule . left-eye-lines-location-and-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :l-to-r)

right-eyebrow-loc-and-char-lines-ltor-03-08
:combo-form . max-score)
:map-rule . right-eyebrow-lines-location-and-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :l-to-r)

right-eye-loc-and-char-lines-ltor-03-08
(:combo-form . max-score)
(:map-rule . right-eye-lines-location-and-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :l-to-r)

chin-loc-and-char-lines-ltor-03-08
(:combo-form . max-score)
(:map-rule . chin-lines-location-and-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :l-to-r)

moustache-loc-and-char-lines-ltor-03-08
(:combo-form . max-score)
(:map-rule . moustache-lines-location-and-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :l-to-r)

mouth-loc-and-char-lines-ltor-03-08
(:combo-form . max-score)
(:map-rule . mouth-lines-location-and-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :l-to-r)

nose-end-loc-and-char-lines-ltor-03-08
(:combo-form . max-score)
':map-rule . nose-end-lines-location-and-characteristics)
:min-max-filter 0.3 0.8)
(:type . :l-to-r)

nose-bridge-loc-and-char-lines-ltor-03-08
(:combo-form . max-score)
```

```
(:map-rule . nose-bridge-lines-location-and-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :l-to-r)

.oh-loc-and-char-lines-ltor-03-08
(:combo-form . max-score)
(:map-rule . top-of-head-lines-location-and-characteristics)
(:min-max-filter 0.3 0.8)
(:type . :l-to-r)
```

>>>functions used by lines_locations.rul<<<<

```
;;;These functions were written by Rich Weiss and reside in [weiss.vis]
;;;line_location.evl.


(defun get-ro (line-index)
   (float (get-object-feature line-index '(line ro)
                       *line-seg-dss)))

(defun get-midpoint (line-index)
  (let* (ro (get-ro line-index)
          theta (get-theta line-index)
          mint (get-mint line-index)
          maxt (get-maxt line-index)
          midt (quotient:r (plus:r mint maxt) 2.0)
          sin-theta (sine:r theta)
          cos-theta (cos:r theta)
          x0 (times:r ro cos-theta)
          y0 (times:r ro sin-theta)
          offset (quotient:r (float (subl(car(get-seg-info '*image-size
                              *reg-seg-dss)))) 2.0)
          row (plus:r x0 offset (times:r (times:r midt ro) sin-theta))
          col (plus:r y0 offset (minus (times:r (times:r midt ro) cos-theta)))

      )
        (list row col)
   )
)

(defun get-theta (line-index)
  (float (get-object-feature line-index '(line theta)
                       *line-seg-dss)))

'defun get-mint (line-index)
  (float (get-object-feature line-index '(line min-t)
                       *line-seg-dss)))

(defun get-maxt (line-index)
  (float (get-object-feature line-index '(line max-t)
                       *line-seg-dss)))
```

>>>>>functions for setting up mask globals<<<<

July 24, 1985

```
;;;see [tuttle.rulesys]rulesys.evl




;;use this function after running rule head-regions.  For directions on how to
;;run this in conjunction with rule head-regions, see comments in
;;[tuttle.rulesys.current_rules]regions_head_area.rul.
(defun get-regions-within-mask (thresh)
   (select-items-above-thresh thresh
                   (get-rule-info (getrule 'head-regions) ':args)
                   (get-rule-info (getrule 'head-regions) ':scores)))

;;these globals are for the primitive region location rules
;;(regions_coords.rul and regions_centroids.rul)
(defun set-more-mask-globals nil
   (let (w-3rd (quotient:r (difference:r *left-side *right-side) 3.0)
         l-3rd (quotient:r (difference:r *bottom-of-head *top-of-head) 3.0))
        (setq *mask-x-midpoint
              (quotient:r (plus:r *left-side *right-side) 2.0))
        (setq *mask-y-midpoint
              (quotient:r (plus:r *bottom-of-head *top-of-head) 2.0))
        (setq *mask-right-third
              (plus *right-side w-3rd))
        (setq *mask-middle-third-of-x-axis
              (plus *right-side (times:r w-3rd 2.0)))
        (setq *mask-left-third *mask-middle-third-of-x-axis)
        (setq *mask-x-axis-view
              (plus:r *right-side (quotient:r w-3rd 3.0)))
```

```
        (setq *mask-x-axis-vhigh
              (difference:r *left-side (quotient:r w-3rd 3.0)))
        (setq *mask-top-third
              (plus *top-of-head l-3rd))
        (setq *mask-middle-third-of-y-axis
              (plus *top-of-head (times:r l-3rd 2.0)))
        (setq *mask-bottom-third
              *mask-middle-third-of-y-axis)
        (setq *mask-y-axis-vlow
              (plus:r *top-of-head (quotient:r l-3rd 3.0)))
        (setq *mask-y-axis-vhigh
              (difference:r *bottom-of-head (quotient:r l-3rd 3.0)))
        (setq *small-margin
              (quotient:r (difference:r *left-side *right-side) 20.0))
        (setq *medium-margin
              (quotient:r (difference:r *left-side *right-side) 10.0))
        (setq *large-margin
              (quotient:r (difference:r *left-side *right-side) 5.0))))


(defun set-mask-bounds (top bot rside lside)
   (setq *top-of-head
                  top)
   (setq *bottom-of-head
                  bot)
   (setq *right-side
                  rside)
   (setq *left-side lside))


                >>>>functions for regions_rectangles.rul<<<<<
                -------------------------------------
                                    June 4, 1981
;;;see [tuttle.rulesys]rulesys_funs.lsp



;;what part of the minimum-bounding rectangle (returned from get-b-rect-area)
;;is actually filled by this region.  Scores returned are greater than 0.0
;and less than or equal to 100.0
.defun rectangular-region (reg-index)
   (times 100.0 (quotient (get-pixcount reg-index)
                          (get-b-rect-area reg-index))))


;;returns area of minimum bounding rectangle.
;; !!NOTE!! looks for rectangles along x and y coords only!!
(defun get-b-rect-area (reg-index)
   (get-object-feature reg-index 'bounding-rectangle-area *reg-seg-dss))

;;returns high score for regions that are longer (along x axis) than
;;they are tall (along y axis).
(defun flat-region (reg-index)
   (quotient (get-reg-width reg-index) (get-reg-height reg-index)))

(defun get-reg-width (reg-index)
   (get-object-feature reg-index 'region-width *reg-seg-dss))

(defun get-reg-height (reg-index)
   (get-object-feature reg-index 'region-height *reg-seg-dss))
```