MATCHING NON-ALIGNED OBJECTS USING A RELATIONAL STRING-GRAPH

Nicholas Dahm[1,3]*, Yongsheng Gao*[3]*, Terry Caelli*[2,4]*, and Horst Bunke*[5]

[1]Queensland Research Laboratory / [2]Victoria Research Laboratory, National ICT Australia
{*nicholas.dahm, terry.caelli*}*@nicta.com.au*

[3]School of Engineering, Griffith University, Brisbane, Australia
{*n.dahm, yongsheng.gao*}*@griffith.edu.au*

[4]Electrical and Electronic Engineering, University of Melbourne, Australia
tcaelli@unimelb.edu.au

[5]Institute of Computer Science and Applied Mathematics, University of Bern, Switzerland
bunke@iam.unibe.ch

ABSTRACT

Localising and aligning objects is a challenging task in computer vision that still remains largely unsolved. Utilising the syntactic power of graph representation, we define a relational string-graph matching algorithm that seeks to perform these tasks simultaneously. By matching the relations between vertices, where vertices represent high-level primitives, the relational string-graph is able to overcome the noisy and inconsistent nature of the vertices themselves. For each possible relation correspondence between two graphs, we calculate the rotation, translation, and scale parameters required to transform a relation into its counterpart. We plot these parameters in 4D space and use Gaussian mixture models and the expectation-maximisation algorithm to estimate the underlying parameters. Our method is tested on face alignment and recognition, but is equally (if not more) applicable for generic object alignment.

Index Terms— image alignment, face recognition, graph matching

1. INTRODUCTION

The localisation and recognition of objects is a mature research area with broad applications in robotics, law enforcement, and the military, including face recognition technologies. However, there still remain challenges around the issues of localisation and alignment for normalisation and recognition invariance, in general. Face localisation with respect to 2D face recognition consists of finding the ideal alignment of one face onto another, so as to maximise the performance of the face recognition algorithm. Depending on the image characteristics, good alignment estimates can be made using Active Shape Models (ASM) [1] and variants like Gabor-Wavelet ASM [2]. Alternatively, the image registration literature offers numerous techniques for alignment that may be effective for face recognition under certain conditions. For a survey of image registration techniques, the reader is directed to [3]. Despite the abundance of localisation techniques, manual alignment is still common in face recognition due to the imprecision of automatic localisation and high precision requirements of many face recognition algorithms.

The literature addressing the actual face recognition has evolved significantly from its pixel-based beginnings. While pixel-based methods are still used with great success, syntactic methods have enabled the use of higher level primitives or feature keypoints. They have proven to be a valuable addition to face recognition, especially when combined with the spatial descriptiveness of graphs, giving rise to methods such as Elastic Bunch Graph Matching [4]. Syntactically higher than keypoints are lines and line segments.

Techniques for extracting such line information range from standard edge detectors like Canny, to point detection with polygonal line fitting, to more tailored solutions like LEM (Line Edge Map) [5] and DCP (Directional Corner Points) [6].

LEM has proven to be particularly effective for face recognition. From an input image, LEM creates an edge map, comprised of edge lines which, in turn, are comprised of a number of connected line segments. The start and end point of each edge line are well-defined, but the intermediate points that separate each line segment are assigned a confidence value, based on the likelihood that the edge line contains a corner at that point. Park et al. [7] represent a face by a Face-ARG (attributed relational graph) where vertices represent line segments taken from the LEM data. Relations (edges) between these vertices characterise the angle and position difference using six parameters, four of which are invariant to global (in-plane) rotation, translation, and scale changes. Vertices between two of these ARGs are then compared using their vector spaces, which contain the parameter

sets of their relations. Using this method, the authors were able to effectively handle occlusion and expression changes, outperforming the original LEM method in all experiments. However the Face-ARG method considers all line segments independent instead of connected, losing valuable syntactic information. The more recent stringface method by Chen and Gao [8] retains this information. Also based on LEM, the stringface method maintains connectivity between the line segments in each edge line where, by encoding each line as a string, line segments are considered as labels (characters), and so the stringface method is able to use string edit operations to calculate edit distances between strings. This greatly reduces the effect of inconsistency in the point detection of LEM. Stringface was tested against a range of competitors including the p-SRC (Partitioned Sparse Representation-based Classification) method. It was shown to be competitive with, or outperform, a number of competitors when tested under ideal conditions, partial occlusions, and varying lighting conditions.

There are however, two key weaknesses in the stringface algorithm. The first is that all strings are processed independently, disregarding the spatial relationships between them. The second is that, like many other algorithms, stringface requires perfectly-aligned images to operate effectively. A translation difference of a few pixels, or a rotation difference of a few degrees, is enough to significantly degrade the results of such systems.

In this paper we present the concept of a relational string-graph, which combines the descriptive powers of stringface and Face-ARG. Furthermore, we show how to achieve alignment invariance using relation correspondence candidates and an estimation technique closely related to [9]. The paper is organised as follows. The relational string-graph and associated vertex and relational properties are described in Section 2. Section 3 describes processes for filtering the number of relation correspondences to an amount that is suitable for parameter estimation. In Section 4 we describe the process of estimating the alignment offset between the input and gallery image. The practical performance of the system is analysed in Section 5. Lastly, we discuss the results and future work of our method in Section 6.

2. RELATIONAL STRING-GRAPH

We define a relational string-graph (RSG) as a complete (fully connected) graph $G = (V, R)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is a set of vertices, and $R = \{(v_i, v_j) : v_i, v_j \in V, i \neq j\}$ is the set of all directed relations (graph edges) between those vertices. Each vertex v_i represents an edge line, composed of a string of line segments $v_i = (l_1, l_2, \ldots, l_{|v_i|})$. Each line segment l_j has three geometric properties, mean coordinate $\mathit{mid}(l_j)$, angle $\theta(l_j)$, and length $\mathit{len}(l_j)$, which are directly read from the LEM data.

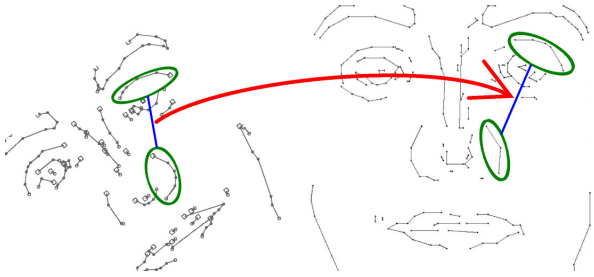

Fig. 1. Example RSG relation correspondence. The vertices are circled in green, the relations are shown in blue, and the relation correspondence in red.

2.1. Vertex Properties

The most common technique for aligning graphs from images is to establish correspondence between vertices. This is effective when vertices represent keypoints or line segments, as they can be accurately described. However given the complex nature of our vertices, the definition of the most basic elementary properties is ambiguous and sensitive to noise. For both vertices and relations, we define properties that are either invariant, or sensitive, to global RTS (in-plane rotation, translation, and scale) changes. We define the RTS-sensitive vertex properties as follows:

$$\mathit{mid}(v_i) = \frac{\sum_{l_j \in v_i} \mathit{mid}(l_j)\mathit{len}(l_j)}{\sum_{l_j \in v_i} \mathit{len}(l_j)} \tag{1}$$

$$\theta(v_i) = \mathit{average}(\{\theta(l_j) : l_j \in v_i\}) \tag{2}$$

$$\mathit{len}(v_i) = \sum_{l_j \in v_i} \mathit{len}(l_j) \tag{3}$$

As our vertices are made of multiple line segments, there is no single correct definition for the average angle. We calculate the average angle using incremental averaging, scaled by segment lengths and including bounds wrapping so that $\Delta(-170°, 170°) = 20°$. In some cases this method may give different values if the order of line segments is reversed, but it has shown to be more robust to noise than other methods.

We also define one RTS-invariant vertex property, which is used to filter vertex correspondence candidates (VCCs):

$$\mathit{len}'(v_i) = \frac{\mathit{len}(\mathit{line}(l_1, l_{|v_i|}))}{\mathit{len}(v_i)} \tag{4}$$

where $\mathit{line}(l_1, l_{|v_i|}))$ is a straight line between the mean coordinates of l_1 and $l_{|v_i|}$. The elimination of VCCs also eliminates a number of relation correspondence candidates (RCCs).

2.2. Relation Properties

The RTS-invariant relation properties are used to filter RCCs between graphs. These RTS-invariant properties for relation

$r_i = (v_j, v_k)$ are:

$$\mathit{len}'(r_i) = \frac{\mathit{len}(v_j)}{\mathit{len}(v_k)} \tag{5}$$

$$\mathit{len}''(r_i) = \frac{\mathit{len}(v_j)}{\mathit{len}(\mathit{line}(v_j, v_k))} \tag{6}$$

$$\theta'(r_i) = \begin{cases} |\theta(v_j) - \theta(r_i)| & \text{if } |\theta(v_j) - \theta(r_i)| \leq 90^\circ \\ 180^\circ - |\theta(v_j) - \theta(r_i)| & \text{otherwise} \end{cases} \tag{7}$$

where $\mathit{line}(v_j, v_k)$ is a straight line between the mean coordinates of v_j and v_k. Note that the above properties are invariant to global RTS changes, but sensitive to local changes.

The RTS-sensitive relation properties are used to determine the appropriate global RTS parameters that would have had to be used, if the two relations being compared are indeed a correct correspondence. As the relation is represented by a single straight line segment, these RTS-sensitive properties are:

$$\mathit{mid}(r_i) = \mathit{mid}(\mathit{line}(v_j, v_k)) \tag{8}$$

$$\theta(r_i) = \theta(\mathit{line}(v_j, v_k)) \tag{9}$$

$$\mathit{len}(r_i) = \mathit{len}(\mathit{line}(v_j, v_k)) \tag{10}$$

3. FILTERING AND PARAMETER EXTRACTION

In this Section we discuss three filtering techniques used to reduce the number of RCCs used for parameter estimation. These are vertex candidate filtering, relation candidate filtering, and outlier filtering, respectively. The motivation behind filtering is that between graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ there can only be at most $\mathit{min}(|E_1|, |E_2|)$ correct relation correspondences, but there are $|E_1| \times |E_2|$ RCCs prior to filtering. These additional RCCs slow down all aspects of our method, and lead to noise during parameter estimation.

3.1. Filtering Vertex Correspondences

For a relation correspondence $(v_j, v_k) \rightarrow (v_l, v_m)$ to be valid, its vertex correspondences $v_j \rightarrow v_l$ and $v_k \rightarrow v_m$ must also be valid. Therefore by filtering out unlikely vertex correspondences, we limit the number of RCCs which must be considered in Section 3.2. To ensure all nodes are given an equal chance, we allow each vertex v_i to map to the 15% of vertices in the other graph which minimise $|\mathit{len}'(v_i) - \mathit{len}'(v_j)|$. Not only does this speed up the relation filtering process, it also helps to prevent vertices being dominant during parameter estimation.

3.2. Filtering Relation Correspondences

The filtering of RCCs is possibly the most important (and time consuming) stage of our method. We filter all RCCs that do not pass our similarity test:

$$|\Psi(r_i) - \Psi(r_j)| < \epsilon(\Psi) \tag{11}$$

for $\Psi = \mathit{len}', \mathit{len}'', \theta'$, where $\epsilon(\Psi)$ is a dynamically adjusted tolerance for each RTS-invariant property.

Since we only want a selected number (β) of RCCs to pass our filtering, exactly β RCCs must pass all three similarity tests. The tolerances are automatically updated so that the number of RCCs that pass each similarity test is equal, and β pass all three.

Once we have β RCCs passing all three similarity tests, we begin parameter extraction. For each RCC, we must extract the RTS parameter set (θ, x, y, α) that transforms the input graph relation into its corresponding relation in the model graph. Using the RTS-sensitive relation properties from Section 2.2, the calculation of these RTS parameter sets is trivial. Note that since global rotation and scale changes affect local relation translations in a non-uniform manner, we must extract and (temporarily) correct rotation and scale (in relation to a global anchorpoint) before we can extract the translation parameters.

3.3. Filtering Outliers

At this stage, all vertex and relation correspondence candidates are discarded, and only the extracted RTS parameter sets are kept. To assist the Gaussian parameter estimation in the next Section, we eliminate outlying parameter sets. Parameter sets with one outlying parameter (distance to ω other parameters is high) are removed.

4. PARAMETER ESTIMATION AND NORMALISATION

Now that we have a set of transform parameters from each remaining RCC, we can estimate the global transform parameters required to align the LEM data from both graphs. We project the parameter sets as points in 4D space and model them using a Gaussian mixture model (GMM) that is fitted using the Expectation-Maximisation (EM) algorithm. After many trials, we found that the system performed well with the number of Gaussians fixed at 5, and the system did not require more than 20 iterations. As the standard E-step weight update formula:

$$p(x|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{4/2}|\Sigma_i|^{1/2}} e^{-\frac{1}{2}(x-\mu_i)^T \Sigma^{-1}(x-\mu_i)} \tag{12}$$

may lead to error states, we cap the minimum $p(x|\mu_i, \Sigma_i)$ value to 0.000001, which is also assigned to all points as a recovery weight if $|\Sigma_i| = 0$.

Once the GMM fitting is complete, we find the Gaussian j with the highest mixture percentage. We take the mean μ_j of this Gaussian as the estimated global transform parameters. Using these global transform parameters, we then transform the LEM data so it is aligned between the two images, making sure to transform scale and rotation with respect to the same global anchorpoint used in Section 3.2.

5. EXPERIMENTATION

To evaluate the effectiveness of our system, we perform a number of face recognition tests with varying degrees of geometric deformation. In our experiments, the stringface algorithm will be employed to complete the matching, either with or without our method aligning the data using the RTS estimation. The dataset we use is the AR face database [10], cropped in exactly the same manner as in [8]. We take a subset of 25 males and 25 females to make 50 test subjects. Neutral images of the test subjects from session 1 are used as gallery images for all experiments. For probe images, we use session 2 images of the subjects with either neutral conditions or scarf occlusion (see Figure 2). In the ideal case, where the parameters are perfectly estimated, the recognition rate should be similar to that of using stringface on perfectly-aligned images. However while the general shape described by LEM is quite robust, the start, end, and corner points in the edge lines are quite sensitive to noise.

Fig. 2. Example LEM data of the deformed probe with neutral expression, deformed probe with scarf occlusion, and gallery image of the same subject from the AR database.

We plot the recognition rate against a deformation multiplier, which represents the level of deformation (from RTS adjustments) in the probe image. At a multiplier value of $1\times$, the deformation is $(+5°, +5, +5, -2\%)$ for the rotation, x translation, y translation, and scale offsets, respectively. We repeat the experiments for multiplier values 0, 1, ..., 5, and 10, for both neutral (Figure 3) and occluded (Figure 4) probe images, both with and without alignment.

From these figures we can clearly see the sensitivity of the standard stringface algorithm. With only a $1\times$ multiplier, the recognition rate drops to 52% and 40% for neutral and scarf images, respectively. Using our RSG algorithm, we were able to achieve nearly total invariance to RTS deformations, with 84% and 64% accuracy from a $10\times$ deformation multiplier. Pushing the system to an extreme level of deformation ($+160°$, $+500$, -500, $+400\%$), our system had recognition rates of 80% and 64%, which is still better than $1\times$ deformation for regular stringface.

6. CONCLUSION

In this paper we presented a novel technique for object detection using a syntactically powerful relational string-graph.

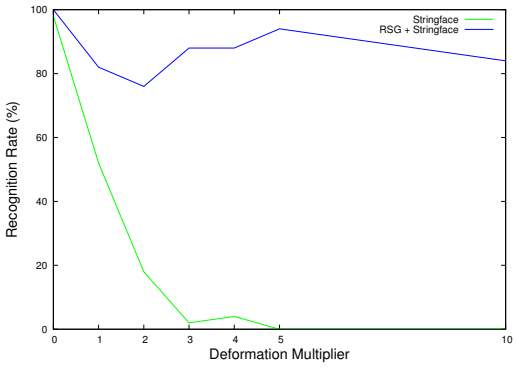

Fig. 3. Recognition rates of regular and RSG-aligned stringface on neutral images with varying stages of deformation.

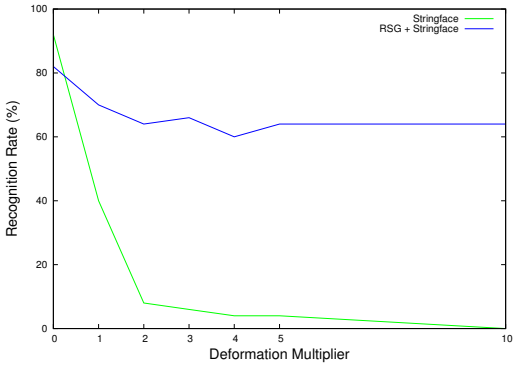

Fig. 4. Recognition rates of regular and RSG-aligned stringface on scarf-occluded images with varying stages of deformation.

The relational string-graph matching technique utilises relation correspondence instead of vertex correspondence. This makes the alignment estimation more robust, as the relational properties are more resistant to noise than the edge lines or line segments. Using Gaussian mixture models, we have been able to project the alignment parameters in 4D space and model their distribution in order to estimate the underlying parameters. We tested our method on the task of recognising non-aligned faces. The encouraging results shown in Section 5 show the validity of our method. In our upcoming work, we will be advancing our method by integrating the stringface matching algorithm into our RSG. This will allow us to utilise the intra-string line segment merging technique from stringface to determine more precise parameter estimates.

7. REFERENCES

[1] T.F. Cootes, C.J. Taylor, D.H. Cooper, and J. Graham, "Active shape models-their training and application," *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 38 – 59, 1995.

[2] Feng Jiao, Stan Li, Heung-Yeung Shum, and D. Schuurmans, "Face alignment using statistical models and wavelet features," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2003, vol. 1, pp. 321–327.

[3] Barbara Zitová and Jan Flusser, "Image registration methods: a survey," *Image and Vision Computing*, vol. 21, no. 11, pp. 977–1000, 2003.

[4] L. Wiskott, J.-M. Fellous, N. Kuiger, and C. von der Malsburg, "Face recognition by elastic bunch graph matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 775–779, jul 1997.

[5] Yongsheng Gao and Maylor K. H. Leung, "Face recognition using line edge map," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 6, pp. 764–779, June 2002.

[6] Yongsheng Gao and Yutao Qi, "Robust visual similarity retrieval in single model face databases," *Pattern Recognition*, vol. 38, no. 7, pp. 1009–1020, 2005.

[7] Bo-Gun Park, Kyoung-Mu Lee, and Sang-Uk Lee, "Face recognition using face-ARG matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 1982–1988, 2005.

[8] Weiping Chen and Yongsheng Gao, "Recognizing partially occluded faces from a single sample per class using string-based matching," in *Proceedings of the 11th European Converence on Computer Vision*, 2010, pp. 496–509.

[9] George Stockman, Steven Kopstein, and Sanford Benett, "Matching images to models for registration and object detection via clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-4, no. 3, pp. 229–241, May 1982.

[10] A. M. Martinez and R. Benavente, "The AR face database," *CVC Technical Report #24*, June 1998.