Elastic Bunch Graph Matching

Laurenz Wiskott et al. (2014), Scholarpedia, 9(3):10587. doi:10.4249/scholarpedia.10587 revision #143316 [link to/cite this article]

- **Prof. Laurenz Wiskott**, Institut für Neuroinformatik, Ruhr-Universität Bochum, Bochum, Germany
- **Dr. Rolf P. Würtz**, Institut für Neuroinformatik, Ruhr-Universität Bochum, Bochum, Germany
- **Dr. Günter Westphal**, TIS Technische Informationssysteme GmbH, Bocholt, Germany

Elastic Bunch Graph Matching is an algorithm in computer vision for recognizing objects or object classes in an image based on a graph representation extracted from other images. It has been prominently used in face recognition and analysis but also for gestures and other object classes.

Contents

Figure 1: Matching at 45°

Figure 2: Matching in frontal pose

Figure 3: Matching at −45°

Introduction

Elastic Graph Matching (EGM) is a biologically inspired algorithm for object recognition in the field of computer vision. It draws its biological inspiration from two sources: (i) The visual features used are based on Gabor (http://en.wikipedia.org/wiki/Gabor_filter) wavelets, which have been found to be a good model of early visual processing in the brain, more precisely simple cells (http://en.wikipedia.org/wiki/Simple_cells) in primary visual cortex (http://en.wikipedia.org/wiki/Visual_cortex) . (ii) The matching algorithm itself is an algorithmic version of dynamic link matching (DLM) (Wiskott and von der Malsburg, 1996), which is a model of invariant object recognition in the brain.

Visual objects in EGM are represented as labeled graphs (http://en.wikipedia.org/wiki/Graph_labeling) , where the nodes represent local textures based on Gabor wavelets and the edges represent distances between the node locations on an image. Thus an image of an object is represented as a collection of local textures in a certain spatial arrangement. If a new object in an image shall be recognized, the labeled graphs of stored objects, so-called model graphs, are matched onto the image. For each model graph the locations for the nodes in the image are optimized such that local texture of the image fits the local texture of the model and distances between the locations fit the distances between the nodes of the model. The model graph with the best fit constitutes the recognized object, and with its node locations in the image an image graph can be created.

Elastic Bunch Graph Matching (EBGM) is an extension to elastic graph matching for object classes with a common structure, such as faces in identical pose. All instances of such a class are represented by the same type of graph. From these graphs a *bunch graph* of same structure is created, with the nodes representing local textures of any object in the class, e.g. all variants of a left eye, and the edges represent average distances between the node locations, e.g. the average distance between the two eyes. This permits taking advantage of the combinatorics of the local textures to represent instances of the object class not seen before. For instance, the textures of the eyes could be taken from one face and the textures of the mouth from another face to represent a new face that shares features with the two stored faces. Thus, a bunch graph is an abstraction for representing object classes rather than individual objects.

EBGM can only be applied to objects with a common structure, such as faces in frontal pose, sharing a common set of *landmarks* like the tip of the nose or the corner of an eye. For the recognition of arbitrary objects, in the absence of landmarks, the graphs are required to be dynamic with respect to both shape and attributed features. To this end, Westphal and Würtz (2009) have proposed a graph dynamics that lets generic object representations, model or bunch graphs, emerge from a collection of arbitrary objects. The idea is to extract typical local texture arrangements from the objects and provide the rules to compose them as needed to represent new objects.

Algorithm

Gabor Wavelets

The most often used elementary features in EBGM are based on Gabor wavelets, having the shape of a (co)sine wave multiplied with a Gaussian envelope function (Figure 4, Figure 5), written as

$$\psi_{\vec{k}}(\vec{x}) = \frac{\vec{k}^2}{\sigma^2} \exp\left(-\frac{\vec{k}^2\vec{x}^2}{2\sigma^2}\right)\left(\exp\left(i\vec{k}\vec{x}\right) - \exp\left(-\frac{\sigma^2}{2}\right)\right).$$

Figure 4: Cosine Gabor wavelet, which is the real part of the complex wavelet.

Figure 5: Sine Gabor wavelet, which is the imaginary part of the complex wavelet.

The first exponential is the Gaussian envelope function with σ/k determining its width. The second exponential combines a cosine-wave in the real part and a sine-wave in the imaginary part of a complex function with wave vector $\vec{k}$ determining the orientation and spatial frequency of the waves. The third exponential is a small correction term that ensures that the wavelet has zero mean. The prefactor k^2/σ^2 normalizes the wavelet. The way the wavelet is written guarantees that for fixed σ all wavelets of different orientation and frequency, i.e. different $\vec{k}$, look identical except for scaling and rotation. In one dimension with $\sigma = 1$ and $k = 1$ and dropping the small correction term the wavelet becomes $\exp(-x^2/2)(\cos(x) + i\sin(x))$, which makes the mathematical structure of the equation clearer.

Mathematically the wavelets are defined with continuous variables, but in the algorithmic version $\vec{x}$ is discretized with the image resolution, and $\vec{k}$ is discretized as follows

$$\vec{k}_{m,l} = k_{\max}\alpha^{-m}\begin{pmatrix}\cos\left(\frac{\pi l}{L}\right)\\ \sin\left(\frac{\pi l}{L}\right)\end{pmatrix} \quad m = 0,\ldots,M-1, \quad l = 0,\ldots,L-1\,.$$

Frequently used parameters are $L = 8, M = 5, \alpha = \sqrt{2}$, i.e., there are 8 orientations and 5 frequencies, which differ by half octaves. The total number of complex filters is therefore 40. The relative width of the Gaussian envelop function is usually chosen as $\sigma = 2\pi$.

Gabor Wavelet Transform

A single Gabor wavelet response at an image location $\vec{x}_0$ can be calculated by centering the Gabor wavelet $\psi_{\vec{k}}(\vec{x})$ at $\vec{x}_0$, multiplying the image grey values $I(\vec{x})$ with the values of the wavelet at corresponding locations, and integrating over all image locations. This would be done with all $L \cdot M$ wavelets to yield a textural description around $\vec{x}_0$ comparable to a local Fourier transform. During matching, however, the Gabor wavelet responses are potentially needed at all locations of the image. Thus, we apply a spatial convolution (http://en.wikipedia.org/wiki/Convolution) of the image with the $L \cdot M$ wavelets yielding the responses at all locations,

$$\mathcal{W}I\left(\vec{x}_0, \vec{k}\right) = \left(\psi_{\vec{k}} * I\right)(\vec{x}_0) = \int \psi_{\vec{k}}(\vec{x}_0 - \vec{x})\; I(\vec{x})\; d^2x\,.$$

In the algorithm a discretized version of this convolution is used. Notice that for mathematical reasons the convolution implies a flip of the wavelets compared to the intuition given above, which is irrelevant for symmetry reasons.

The calculation of the convolution is conveniently done using the Fourier transform (http://en.wikipedia.org/wiki/Fourier_transform), which is more efficient than direct computation. This also implies that the boundary conditions for the convolution are wrap-around. The Fourier transforms of the Gabor wavelets can be calculated analytically (without computation time) as:

$$\left(\mathcal{F}\psi_{\vec{k}}\right)(\vec{\omega}) = \exp\left(-\frac{\sigma^2\left(\vec{\omega}-\vec{k}\right)^2}{2\vec{k}^2}\right) - \exp\left(-\frac{\sigma^2\left(\vec{\omega}^2+\vec{k}^2\right)}{2\vec{k}^2}\right).$$

The complete algorithm for calculating the Gabor wavelet transform consists of a Fourier transform of the input image, followed by multiplication with

$\mathcal{F}\psi_{\vec{k}_{lm}}$ and inverse Fourier transform of the $L \cdot M$ products. For an image of N pixels, the computational complexity of this is $\mathcal{O}(L \cdot M \cdot N \cdot \log_2 N)$.

Jets

The Gabor wavelet transform yields a value for each wavelet at all locations of the image. Thus, with the standard parameters and discretized images it yields 80 (40 real + 40 imaginary) values at any pixel position. This set of values for a single pixel position is referred to as a *jet* J (Figure 6). Since a jet contains values from wavelets of different frequency and orientation, one can think of it as a local Fourier transform, and it is as such a representation of the local texture. It is in fact possible to reconstruct the image gray values from a jet in a small surrounding of its location, except for the mean value.

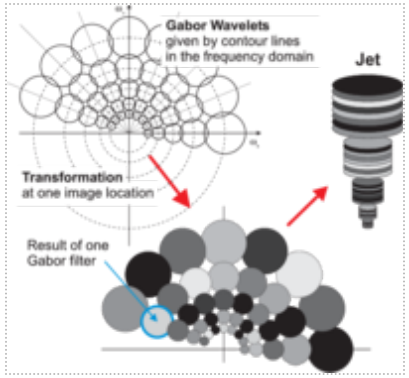

Figure 6: Creation of Jets by a Gabor wavelet transform. Upper left: The circles indicate the Gaussians in the frequency domain that represent the complex Gabor wavelets. Bottom: At a given image location each wavelet yields a complex response, only the amplitude of which is illustrated here by the gray values. Upper right: The illustrated amplitude responses can be stacked into a more compact shape to represent a jet.

The Gabor wavelets come in pairs of cosine (real part) and sine (imaginary part) filters. Each filter by itself is relatively sensitive to a small shift, either of the image or of the pixel position in a stationary image. However, squaring and adding the responses of such pairs reduces the number of values to 40 and yields the local analogue to a power spectrum, which still resolves frequencies and orientations but is insensitive to small shifts (related to the rule $\cos^2(x) + \sin^2(x) = 1$). In more technical terms one splits all complex wavelet responses into amplitude and phase resulting in amplitude vectors $|J|$ and phase vectors $\arg(J)$. Using only amplitudes is often advantageous, because it makes the jets more robust with respect to shifts and other transformations but at the price of not being able anymore to reconstruct the local texture so easily. Thus for localization and reconstruction one tends to use the full jets J with phase information, while for recognition purposes the reduced amplitude jets $|J|$ have proven to be more useful.

Graphs

A jet J is a representation of a local texture. To represent images of whole objects, one needs to combine many jets in a defined spatial arrangement. One could use jets at all pixel positions within the area of the object, but that would be redundant by a factor of 80. Thus one subsamples the jets in the image. In its simplest form this can be a rectangular array with fixed spacing from which jets are taken and stored to represent an object. In the more general case one can define a graph (with vertex set V and edge set E) that is specific to the object and permits to locate nodes at particularly salient points on the object, which are then used as landmarks, see Figure 7. The full representation of a single object then is a labeled

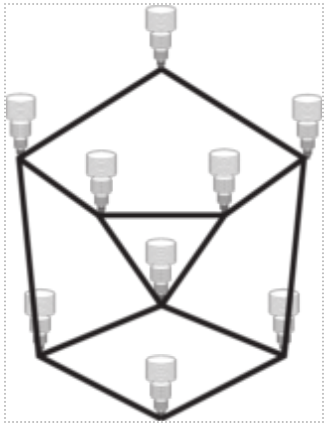

Figure 7: Schema of a model graph. On a fixed graph structure all nodes are labeled with the respective jets.

graph with vertices $i \in V$ labeled with jets J_i and edges $(i,j) \in E$ labeled with distances $\vec{\Delta}_{ij}$ between the pixel positions from which the jets have been taken in the original image. It is relatively easy to reconstruct the image of an object from such a graph representation if the phase of the jets is preserved, i.e. 80 values per jet are stored for the standard parameters.

Two such graphs G^I and G^M with the same structure can be compared by a similarity function, which should be high for identical graphs and low for graphs from very different images. It consists of a similarity for vertices and one for edges, combined linearly:

$$S_v\left(J^I, J^M\right) := \frac{|J^I| \cdot |J^M|}{\|J^I\| \, \|J^M\|} \,, \tag{1}$$

$$S_e\left(\vec{\Delta}_{ij}^{\,I}, \vec{\Delta}_{ij}^{\,M}\right) := - \frac{\left\|\vec{\Delta}_{ij}^{\,I} - \vec{\Delta}_{ij}^{\,M}\right\|^2}{\left\|\vec{\Delta}_{ij}^{\,M}\right\|^2} \,, \tag{2}$$

$$S_G(G^I, G^M) := \sum_{i \in V} S_v(J_i^I, J_i^M) + \lambda \sum_{(i,j) \in E} \mathcal{S}_e\left(\vec{\Delta}_{ij}^{\,I}, \vec{\Delta}_{ij}^{\,M}\right) . \tag{3}$$

In the similarity function (1) for the jets, the phase information is ignored and only the absolute value of the wavelet responses is used, indicated by $|J^I|$, which is somewhat more robust to small geometrical variations and changes in illumination; $\|\cdot\|$ indicates the Euclidean norm. In practice, many modifications to the jet similarity functions are feasible and can be useful for better localization or recognition performance (e.g., Günther and Würtz, 2009). The similarity function (2) for the distances between nodes can be thought of as a model of elastic springs between the nodes that not only try to keep their length but also their direction similar to those in the original image. The graph similarity function (3) combines all jet similarities and node

distance similarities, with parameter λ weighting the latter vs the former. We refer to the first as the texture term and the latter as the geometry term. A high similarity between two graphs can only be achieved if both textures and geometry are similar.

Elastic Graph Matching

Assume we are given a model graph G^M and a new image I . Elastic Graph Matching is a process by which a graph is found in the image that fits well to the model graph according to the graph similarity function (3) (Lades et al 1993). This is an optimization problem in a space with a dimension twice the number of graph vertices, because each node has a vertical and a horizontal coordinate. In the simplest version one could pick node locations $\overrightarrow{x_i}, i \in V$, in the image at random and calculate the similarity of the resulting image graph G^I with the model graph G^M . Repeating this process often and keeping only the best result should eventually yield an image graph that fits well to the model graph. However, the process can be made much more efficient by a heuristics that gradually introduces flexibility in the image graph and samples the parameter space systematically rather than randomly. One such heuristic scheme could look as follows, cf. Figure 8:

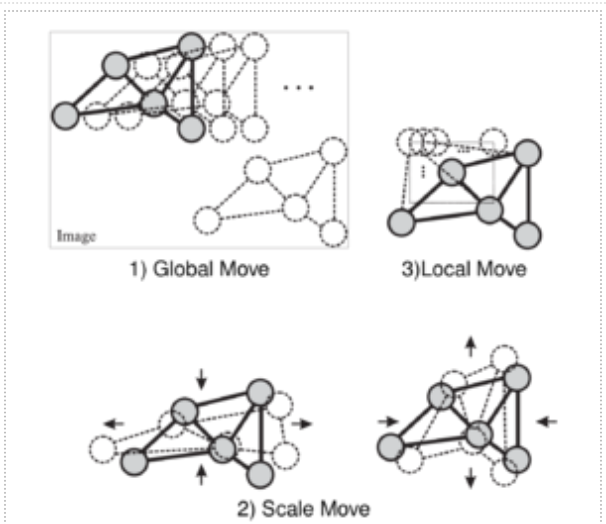

Figure 8: Heuristic scheme for elastic graph matching: (1) Move the rigid graph across the image. (2) Scale the graph horizontally and vertically. (3) Move single nodes locally.

1. **Global Move: Scan the image to find the position of the object.** First use the geometry of the model graph for the image graph, i.e. with identical distances between the corresponding nodes, but shift the graph across the image. Since the relative node positions are kept constant, the geometry term of the graph similarity function is irrelevant and only the jet similarities matter. This scanning is usually done on a rectangular grid of positions, first with a large spacing, e.g. 10 pixels, but then repeated around the best position with a finer spacing, yielding a more accurate estimate of the position of the object in the image.
2. **Scale Move: Scale the image graph to find the right size and aspect ratio of the object.** Next the image graph can be scaled horizontally and vertically to improve the similarity with the model graph. A horizontal or vertical scaling changes the aspect ratio and should probably be punished by the geometry term in the similarity function. For a global scaling however, i.e. same horizontal and vertical scaling, one can argue that this is no distortion but just a consequence of a different distance from the observer, and one could rescale the model graph correspondingly, so that the geometric transformation alone does not effect the similarity function (but notice that the jets also differ if an object has a different size).

3. **Local Move: Move all nodes locally to adapt to local distortions or differences.** Finally one can randomly move the nodes locally to find an even better match. This should be done for all nodes repeatedly, e.g. in a random order, because one node cannot be optimally positioned if the others are still misplaced.

The result of such matching is an image graph, which could be integrated into the gallery as a model graph, and a similarity value between the model graph matched to the image and the resulting image graph, which can be used to decide whether the model object is in the image or not.

Bunch Graphs

Elastic graph matching solves the problem of finding a known object in an image by matching a model graph to the image. However, it would be advantageous if one could also find an unknown object in an image and create an image graph for it in a reproducible manner. This would have at least two major advantages: Firstly, model graphs for new objects could be created without manual assistance. Secondly, there would be no need to match every single model in the gallery to the image. An image graph could be generated independently of a concrete model and then only the graph comparison would be done for each model, which would save a lot of computation.

Such a generic creation of an image graph is difficult for two reasons: Firstly, one has to segment the object from the background, and secondly, one has to decide where to place the nodes on the object. However, when processing an object class where all objects have a common structure, much is known about a new object of that class, even if we have not seen it before. Faces are a prominent example. Even if we have not seen a particular face before, we know its structure and can easily find the eyes, nose, etc., because we have seen many other faces before.

The structural knowledge of faces and their variants can be represented in a so-called *bunch graph* G^B (Wiskott, 1997, Wiskott et al, 1997) (Figure 9). Assume model graphs of identical structure are given for 100 frontal views of different faces. Since the graphs have identical structure, one can easily calculate the average distance between two nodes across all 100 model graphs. This yields the labels of the edges in the bunch graph. The nodes are simply labeled with all hundred jets from the different faces at one node. Thus, a bunch graph differs from a normal object graph in that each node is labeled with not just one but many jets, and the edges are labeled with average distances rather than distances from one concrete face.

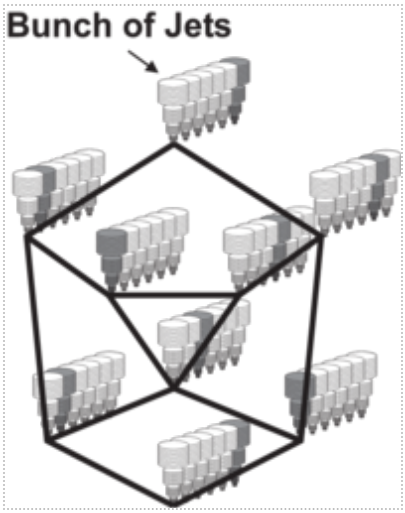

Figure 9: Schema of a bunch graph. The jets of many model graphs are combined. During matching, different jets become the local experts, here shaded in gray.

When comparing a bunch graph with a normal graph, one takes advantage of the combinatorics and always picks the best fitting jet at each node independently of the other nodes, realized by the max operation in the bunch graph similarity function (4).

$$S_{BG}(G^I, G^B) := \frac{1}{|V|}\sum_{m \in G} \max S_v(J_i^I, J_{i,m}^B) + \frac{\lambda}{|E|}\sum_{(ij)\in E} S_e\left(\vec{\Delta}_{ij,m}^I, \vec{\Delta}_{ij,m}^B\right). \quad (4)$$

Thus, the eye jets might come from one person, the nose jet from another one, and the mouth jets from a third one. This process resembles the process by which police creates an identikit picture for an unknown person they are searching for. The best fitting jet at one node is referred to as the *local expert*.

Elastic Bunch Graph Matching

Matching a bunch graph to an image (, ,) to create an image graph works exactly as elastic graph matching except that the local expert, as determined by the max operation, is used for each node and might actually change during matching as the position of the node changes (Wiskott, 1997,Wiskott et al, 1997). The result of the matching is not only the image graph but also the identity of the local experts, which can be used to further analyze the object, see below.

Applications

Face Recognition

Once we have the means to generate and compare image graphs, recognition of faces in identical pose is relatively straight-forward. Matters become more complicated when trying to recognize faces across different poses.

Recognition in Identical Pose

Assume we are given 1000 facial images of identical pose, e.g. all looking straight into the camera, and correctly labeled with the name of the person they show. This constitutes the model gallery. For face recognition we would proceed as follows:

- **Step 1: Building a face graph.** The first step to bootstrap the system is to define the graph structure for the given pose. Thus, we take the first image and manually define node locations on the face that are easy to localize, such as the corners of the eyes or mouth, the center of the eyes, the tip of the noise, some points on the outline etc. We also define edges between the nodes. This constitutes our first face graph.
- **Step 2: Building a face bunch graph.** The single face graph defined above can be viewed as a bunch graph with just one instance in it. It can be matched onto the second face image, but if the first two face images are not very similar, the match is of poor quality. For instance, the tip-of-the-nose node might by placed at the cheek, so we need to move the node onto the tip of the nose by hand. After some such manual correction the graph is acceptable and constitutes the second instance in the bunch graph. The bunch graph with two instances is then matched onto the third image, and after some manual correction we have a third instance for the bunch graph. By repeating this process, the bunch graph grows, and as it grows the match onto

new images gets more and more reliable. If we are satisfied with the quality of the matches and only little manual correction is needed, we are done with building the bunch graph. Let say this happens after having processed the first 100 images.

- **Step 3: Building the model gallery of graphs.** Since we now have a bunch graph that provides sufficient quality in finding the node locations in a new face, we can process the remaining 900 images fully automatically. To avoid the distinction between manually corrected and fully automatically generated model graphs, one can create the bunch graph on an extra set of 100 images distinct from the 1000 model images. Then all 1000 model graphs can be created automatically. We are now in the position to perform face recognition on a new probe image.
- **Step 4: Building the probe graph.** Assume we are given a new image and shall find the depicted person in the gallery. First we need to create a graph for the probe image. This process works exactly as for the model images, just that we use the probe image.
- **Step 5: Comparison with all model graphs.** The image graph is compared with all model graphs, resulting in 1000 similarity values. These form the basis of the recognition decision. Notice that this does not require EBGM anymore, only the graphs are compared according to the similarity function (3).
- **Step 6: Recognition.** For recognition it is obvious that the model graph with the highest similarity with the image graph is the candidate to be recognized. However, if the best similarity value is relatively low, the system might decide that the person on the probe image is not in the model gallery at all; and if there are more than one very high similarity values, the system might decide that the person is most likely in the model gallery but that there are several possible candidates. Only if the highest similarity is high and the next one is low can the system recognize the face on the probe image with high reliability.

This system has achieved 98% correct recognition rates on a gallery of 250 frontal view faces, 57% on half profiles, and 84% on profiles (Wiskott et al, 1997). Half profiles were presumably most difficult, because the pose was least well controlled.

Recognition Across Different Poses

EBGM is robust to small pose changes. Larger differences in pose modify the local features too much for corresponding points to be found, some features are occluded in some poses. This difficulty can be overcome by creating bunch graphs for different poses and matching all of them to the probe image. As small pose differences are tolerated by the matching process, only coarse sampling of pose is necessary. Only if the poses matched are roughly equal, high similarity values and good correspondences will result.

On the other hand, a single frontal image per person should contain enough information for successful recognition. It is therefore desirable to reduce the gallery to single pose and learn the pose variation from examples. Müller et al (2013) have done this by setting aside a *model gallery* of graphs of N_M people known in *all* poses. The actual *gallery* of all people only contains frontal poses. Each gallery graph can be matched to the models in frontal pose. From the matching result G^0 the *rank list* γ^0 of jet similarity to all models is calculated for each node. These define the similarity relations of the gallery

image to the frontal pose images. For a probe image, EBGM can be applied to the model gallery, with the best match belonging to the correct pose but not necessarily to the correct identity, because that identity need not be part of the model gallery. For the found pose again similarity rank lists π^v of length N_M can be calculated for each graph location. These are a representation of the probe image in terms of similarities to the various model identities in the appropriate pose. While no direct similarity between the probe image and the gallery images can be calculated, it can be expected that the rank lists π_v and γ_v will be similar for the same person. The underlying assumption is that faces that are similar in one pose will also be similar in other poses. With the help of the rank list similarity function

$$S_{\text{rank}}(\pi, \gamma_g) = \frac{\sum_{m=1}^{N_M} \lambda^{\pi(m)+\gamma_g(m)}}{\sum_{m=1}^{N_M} \lambda^{2m}} \ ,$$

which is averaged over all graph nodes present in both pose graphs

$$S_{\text{rec}}(g) = \frac{1}{|\mathcal{L}^v \cap \mathcal{L}^0|} \sum_{n \in \mathcal{L}^v \cap \mathcal{L}^0} S_{\text{rank}}(\pi_n^v, \gamma_{g,n}) \, ,$$

the most similar rank list to π can be identified.

With the standard jet similarity function and 500 people in the model gallery this method yielded 97.4% of correct recognitions in different poses on the CAS-PEAL database (Gao et al., 2008), with a combination of different ones the rate can be as high as 98.8%.

Recognition in different illuminations

The same method can be applied to recognition in different illumination situations. With a model gallery of 100 identities from the CAS-PEAL database (Gao et al., 2008) and 9 different illuminations the recognition rate for standard jet similarity was 82.7%, with combined similarities 88.9%.

Face Analysis

Facial Features

As a result, EBGM yields

- an overall similarity value, which can be used to decide whether the image actually contains an object of the given class, e.g. a face,
- locations of the nodes in the image, which form the basis for creating an image graph used for storage or recognition, and
- for any node the best fitting instance in the bunch graph, referred to as local expert.

For faces, the latter can be used for an analysis of some abstract properties of the face. For instance, if most local experts come from female faces, it is likely that the face on the new image is female as well; if most local experts in the lower half of the graph come from bearded faces, it is likely that the person has a beard; if most local experts in the upper half of the graph come from faces with glasses, it is likely that the person wears glasses (Figure 10). On a mixed set of 111 faces this method has achieved 92% correct discrimination between male and female faces, 94% correct detection of a beard vs no beard, and 96% correct detection of glasses vs no glasses (Wiskott, 1997).

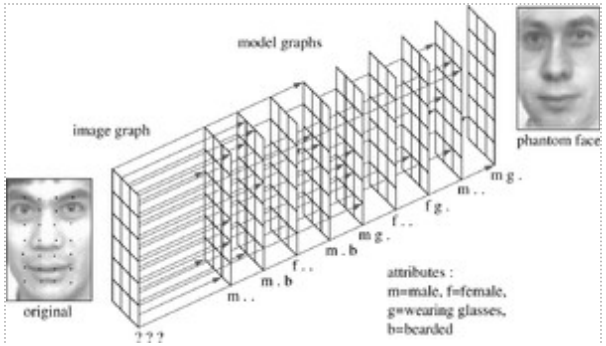

Figure 10: Face analysis with EBGM. A face bunch graph (middle) is matched onto a new face image (left). For each node (small black dots) there is a local expert in the bunch graph (pointed at by the arrow heads), from which a so called phantom face can be synthesized and abstract features such as gender, glasses, or beard can be inferred. In this case the phantom face looks Caucasian even though the original face is half Asian, because the bunch graph only includes Caucasian faces. Reproduced from (Wiskott, 1997) with permission from Elsevier (http://www.elsevier.com/journal-authors/author-rights-and-responsibilities) .

Medical Analysis

The image graph found by EBGM in a facial image is a representation of that image. It can be converted to a feature vector by simply listing all vertex and edge labels. This vector can be used as input for classification systems. Such classifiers can provide suggestions for medical diagnoses for ailments which influence the appearance of the face. Böhringer et al. (2011) have used this approach to distinguish among eleven genetically caused syndromes. Five of them could be identified in 100% of the cases, the overall correct recognition rate was 60%. The classifier used was Local Discriminant Analysis after Principal Component Analysis and dimension reduction. These high classification rates could, however, only be achieved with some manual corrections on the placement of the graph nodes on the image. Finding good landmark locations automatically remains a research challenge. In a further study (Schneider et al, 2011) the method was applied to distinguishing patients with acromegaly from a control group with correct decisions for patients in 72% and controls in 92% of the cases. Kosilek et al (2013) used the image graph features for diagnosing Cushing-syndrome in female patients with an overall accuracy of 92%. Both studies also required manual corrections of landmark placement.

Object Recognition

Cluttered Scenes

Further challenges in recognizing objects are cluttered background and partial occlusions. EGM can deal with that due to the localized feature representation in the graphs, which limits the influence of background on the object representation and permits to ignore the occluded parts. However, it is still necessary to find the object in the scene despite the clutter and to decide which parts might be occluded. It has been shown that EGM is fairly robust to clutter and occlusions in finding the object, thus localizing a known object in a scene is possible with relatively high reliability. Deciding which regions are

occluded can be done based on the local similarity values of the model jets with the corresponding image jets, visible regions have high similarity values while occluded regions have low similarity values. Thus a simple thresholding operation can already be suggestive as to which nodes are occluded. This can be further refined by regularization, e.g. one can consider a node as occluded if at least three out of four of its neighbors are occluded as well, even if its similarity value is above threshold. An object is then considered recognized with confidence if a certain minimal area of it is considered visible and if the average similarity within that area is high enough. This algorithm gave a performance of 80% correct recognition rate in cluttered scenes with extensive occlusions (Wiskott and von der Malsburg, 1993).

Further improvement can be achieved if one uses matches of other objects. For instance, if the system has recognized a known object in the foreground, it is clear that any other candidate object matched to the scene must be occluded in the area of the known object in the foreground. With that logic one can work a scene from front to back with high reliability. An algorithm using this synergy between object matches in a scene yielded a performance of 21 correctly interpreted scenes out of 30 scenes tested. For 3 scenes the objects were recognized correctly but the estimated occlusion order was wrong. In the remaining 6 scenes, 3 models were accepted although the corresponding object was not present, and 4 objects that were present were not recognized. Overall 96.7% of the objects were recognized correctly (Wiskott and von der Malsburg, 1993).

Gesture Recognition

Triesch and von der Malsburg (2001) have extended the method of EBGM to *compound features* and have applied it to hand gesture recognition. Compound features augment the jet vectors by extra dimensions provided by color features and Gabor functions applied to color channels. The similarity functions are adjusted accordingly. An additional optimization step accounts for rotation in the image plane. Bunches are built from graphs of the same gesture in front of uniform backgrounds of varying brightness. The bunch graph similarity function selects the best local match in the bunch graph. This leads to considerable invariance against structured backgrounds, because the local experts may match to different brightness values across nodes. The system has been used successfully to guide pick-and-place behavior by a robot (Becker et al., 2001).

Generalized Object Graphs

EGM and EBGM have proven to perform well when the objects to be recognized have a common structure, such as faces in frontal pose. These share a common set of fiducial points, so-called *landmarks*, such as the corners of the eyes, of the mouth, the tip of the nose and so forth. However, for the recognition of a larger repertoire of arbitrary objects such a set of landmarks cannot be defined. Therefore, EGM and EBGM, in their pure form, encounter problems when applied to the task of invariant visual object recognition. Here, object models are required to be dynamic with respect both to shape and attributed features in order to cope with object variations like changes in identity, pose, illumination and so on. Graph-like structures, and model graphs in particular, inherently fulfill this requirement as they allow for *compositionality*, the ability to construct mental representations, hierarchically, in terms of parts and their relations. What needs to be specified are the elementary parts, the *constituents*, and the rules of composition. Westphal and Würtz (2009) have employed small regular graphs of 3x3 Gabor jets on a ten times ten pixel grid as constituents (Figure 11). These are termed *parquet graphs*, inspired

by the look of ready-to-lay parquet tiles. They can be used as local image features and can be aggregated to larger graph entities that can represent entire objects. As a useful initialization of E(B)GM Westphal and Würtz (2009) have proposed that fast feature-based preprocessing is applied as far as it goes by excluding as many objects as possible and that only a manageable number of ambiguous cases really worth their while, they have called them *model candidates*, are subjected to the more accurate correspondence-based processing of EGM. In the course of this, model graphs emerge that represent the object in the input image well.

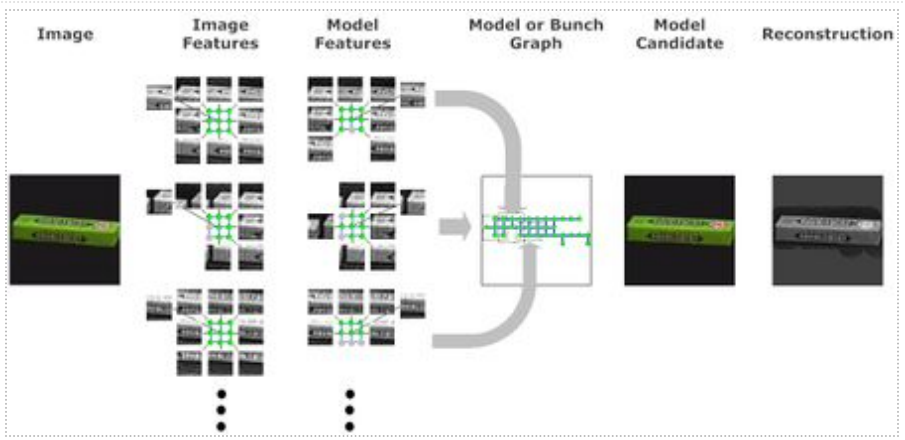

Figure 11: Composition of parquet graph features into model or bunch graphs: Upon presentation of an image (first column) a model or bunch graph (fourth column) is composed (arrows) out of memorized model parquet graph features (third column) that have a superthreshold similarity to the image features (second column). The best model candidate is given in the fifth column. The reconstruction from the model graph (column six) demonstrates that the model graph describes the object in the input image well. Reconstruction and model candidate contain the same object in the same pose, which is slightly different from the one in the input image. The nodes of the image and model parquet graphs that are known to reside in the background are excluded from calculation and are painted in grey.

The desired preselection of model candidates is achieved by a neural network, called the *preselection network*. Given an input image, it computes an *activation* for each model image based on the number of feature coincidences. The activations scale proportionally with the probability that the input and the respective model image contain the same object under similar object variations, i.e., the output neurons respond to particular object views. The pre-selection of model candidates is achieved through picking a manageable number of highly activated models. As a second result the preselection network provides a set of coincidental features for each model candidate. The rules to compose these parquet graphs into larger graph entities are provided by their memorized spatial arrangements. Therefore, for each model candidate, the coincidental parquet graph features can be aggregated to an image and a model graph, respectively. These are matched onto each other using EGM. The model graph that yields the highest similarity value is considered to be the best approximation of the object contained in the input image. The method is also capable of aggregating model parquet graph features to *bunch graphs*. To this end, the model parquet graphs are not just taken from one but from a carefully chosen selection of highly activated model candidates, e.g., from model candidates that belong to the same semantic category. Thus, in a similar way, the method exploits the combinatorics of model parquet graph features like EBGM exploits the combinatorics of the jets in the bunches.

The method has achieved high recognition rates on identity, object category, pose, and illumination type, provided that the individual object variations are sufficiently covered by learning examples. Unlike many other models this technique can also cope with varying background, multiple objects, and partial occlusion (Westphal and Würtz, 2009).

References

- Becker, M et al. (1999). GripSee: A Gesture-controlled Robot for Object Perception and Manipulation. *Autonomous Robots* 6: 203-221. doi:10.1023/A:1008839628783 (http://dx.doi.org/10.1023/A:1008839628783) .
- Böhringer, S et al. (2011). Automated syndrome detection in a set of clinical facial photographs. *American Journal of Medical Genetics Part A* 155: 2161-2169. doi:10.1002/ajmg.a.34157 (http://dx.doi.org/10.1002/ajmg.a.34157) .
- Gao, W et al. (2008). The CAS-PEAL Large-Scale Chinese Face Database and Baseline Evaluations. *IEEE Transactions on Systems, Man, and Cybernetics Part A* 38: 149-161. doi:10.1109/TSMCA.2007.909557 (http://dx.doi.org/10.1109/TSMCA.2007.909557) .
- Günther, M and Würtz, RP (2009). Face Detection and Recognition Using Maximum Likelihood Classifiers on Gabor Graphs]. *International Journal of Pattern Recognition and Artificial Intelligence* 23: 433-461. doi:10.1142/s0218001409007211 (http://dx.doi.org/10.1142/s0218001409007211) .
- Kosilek, RP et al. (2013). Automatic face classification of Cushing's syndrome in women - A novel screening approach. *Experimental and Clinical Endocrinology and Diabetes* 121: 561-564. doi:10.1055/s-0033-1349124 (http://dx.doi.org/10.1055/s-0033-1349124) .
- Lades, M et al. (1993). Distortion Invariant Object Recognition in the Dynamic Link Architecture. *IEEE Transactions on Computers* 42: 300-311. doi:10.1109/12.210173 (http://dx.doi.org/10.1109/12.210173) .
- Müller, MK et al. (2013). Learning invariant face recognition from examples. *Neural Networks* 41: 137-146. doi:10.1016/j.neunet.2012.07.006 (http://dx.doi.org/10.1016/j.neunet.2012.07.006) .
- Schneider, HJ et al. (2011). A novel approach for the detection of acromegaly: accuracy of diagnosis by automatic face classification. *Journal of Clinical Endocrinology and Metabolism* 96: 2074-2080. doi:10.1210/jc.2011-0237 (http://dx.doi.org/10.1210/jc.2011-0237) .
- Triesch, J and von der Malsburg, C (2001). A System for Person-Independent Hand Posture Recognition against Complex Backgrounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23: 1449-1453. doi:10.1109/34.977568 (http://dx.doi.org/10.1109/34.977568) .
- Westphal, G and Würtz, RP (2009). Combining Feature- and Correspondence-Based Methods for Visual Object Recognition. *Neural Computation* 21: 1952-1989. doi:10.1162/neco.2009.12-07-675 (http://dx.doi.org/10.1162/neco.2009.12-07-675) .
- Wiskott, L (1997). Phantom Faces for Face Analysis. *Pattern Recognition* 30(6): 837–846. doi:10.1016/S0031-3203(96)00132-X (http://dx.doi.org/10.1016/S0031-3203(96)00132-X) .
- Wiskott, L et al. (1997). Face Recognition by Elastic Bunch Graph Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19: 775-779. doi:10.1109/34.598235 (http://dx.doi.org/10.1109/34.598235) .
- Wiskott, L and von der Malsburg, C (1993). A Neural System for the Recognition of Partially Occluded Objects in Cluttered Scenes. *Intl. J. of Pattern Recognition and Artificial Intelligence* 7(4): 935-948. doi:10.1142/S0218001493000479 (http://dx.doi.org/10.1142/S0218001493000479) .

- Wiskott, L and von der Malsburg, C (1996) Face Recognition by Dynamic Link Matching (http://www.cs.utexas.edu/users/nn/web-pubs/htmlbook96/) . Chapter 11 in Lateral Interactions in the Cortex: Structure and Function. Eds. Sirosh, J and Miikkulainen, R and Choe, Y. Publ. The UTCS Neural Networks Research Group, Austin, TX.

Sponsored by: Jian-Gang Wang, Institute for Infocomm Research, Singapore

Reviewed by (http://www.scholarpedia.org/w/index.php?title=Elastic_Bunch_Graph_Matching&oldid=139268) : Prof. Shiguang Shan, ICT, CAS, Beijing, China

Reviewed by (http://www.scholarpedia.org/w/index.php?title=Elastic_Bunch_Graph_Matching&oldid=139268) : Prof. Ales Leonardis, University of Birmingham, Birmingham, United Kingdom

Accepted on: 2014-02-10 20:30:34 GMT (http://www.scholarpedia.org/w/index.php?title=Elastic_Bunch_Graph_Matching&oldid=139268)

Categories: Computer Vision | Biometrics