ISSN 1470-5559

Object Localization by Generative Graph Configuration

Huaijun Qiu and Shaogang Gong

RR-06-13

December 2006

Department of Computer Science

Object Localization by Generative Graph Configuration

Huaijun Qiu and Shaogang Gong
Department of Computer Science
Queen Mary, University of London
London E1 4NS, UK
{john,sgg@dcs.qmul.ac.uk}

December 2006

Abstract

State-of-art statistical models for object recognition are sensitive to large affine transformations in scale and rotation. This is due to the rather strict spacial correlation assumption made between model parts that cannot be solved using distance transform. In this work, to tackle these problems we take a different approach by utilizing topological properties of a graphical object model. Our model is a part-based generative model with each part representing a patch of an object image. The graph structure is learned by Boosting the graph nodes additively so that new parts are added in until a threshold is reached. Every time when a new part is inserted, all other parts are updated accordingly. The result of this process gives us a reliable and flexible graph structure that encodes local topological information of an object appearance. The recognition step is performed using two different methods: either by exhaustive search of a topological graph which best matches the model, or by fitting the model incrementally starting from selecting the most reliable (rather than most dominant from training) part candidates from the test image.

1 Introduction

It is difficult for all existing object recognition methods (Bar-Hillel et al., 2005a,b; Crandall et al., 2005; Crandall and Huttenlocher, 2006; Felzenszwalb and Huttenlocher, 2005; Fergus et al., 2003; Mikolajczyk et al., 2006; Torralba et al., 2004) to locate objects and their corresponding parts in a scenario like those shown in Fig. 1. Our aim is to solve this problem using a topological approach from generative graph configuration. This is motivated by the following reasons.

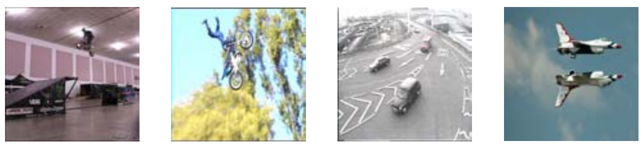


Figure 1: Difficult examples for object localization task.

1.1 Model Structure

In a statistical model, an object can be represented using a collection of parts. Each part encodes some local visual properties of the object, such as a patch from the image. Spatial constraints may be used to model coorelations between parts. This resulted in a range of model structures from the simplest spatial independent bag-of-key-words (Dance et al., 2004), to star-like models with only one reference node (Bar-Hillel et al., 2005a; Fergus et al., 2005), k-fans with k reference nodes (Crandall et al., 2005; Crandall and Huttenlocher, 2006), sparse connected graphs like trees (Felzenszwalb and Huttenlocher, 2005), K-NN graphs (Carneiro and Lowe, 2006), and a fully connected constellation structure (Fergus et al., 2003).

A major problem with a simple spatial independent model is the lack of image semantics which makes any realistic object localization highly unreliable. To the other extreme, a constellation model suffers from an exponentially increased complexity in model parameters w.r.t model parts and learning rendering it computationally intractable. In between, both star-like models and k-fans have a very restrict graph structure which has to be defined beforehand. Inherently, their use of a global geometric constraint based on the relative position of each satellite part to a reference part renders their ability to deal with large affine transformations.

Spatial relations between parts can be modeled either globally or locally. Star-like models (Bar-Hillel et al., 2005a; Fergus et al., 2005) and k-fans (Crandall et al., 2005; Crandall and Huttenlocher, 2006) all have some special nodes or parts as global geometric references for the other parts. Trees (Felzenszwalb and Huttenlocher, 2005) and K-NN graphs (Carneiro and Lowe, 2006) instead have no such global references and parts are only connected locally to their neighbors. A good property of using local spatial constraint is the ability of dealing with both local deformations and global transformations (Carneiro and Jepson, 2004). Aiming at locating objects and their corresponding parts under large affine transformation, we take 2-NN graph as our model structure. The motivation is to build a graph with triangles with relative angles and distance ratio as model primary elements since they are flexible to local deformations and also reliable to large affine changes.

1.2 Model Learning

Given a set of training images, typically a generative model learns the model parameters in a maximum likelihood framework. Features from each training image are either manually labeled (Crandall et al., 2005; Felzenszwalb and Huttenlocher, 2005), or selected in an exhaustive search (Crandall and Huttenlocher, 2006; Fergus et al., 2003). Manually labeled data suffers from the quality of the parts chosen and the accuracy of the hand-labeled ground-truth. In a bottom up approach, raw responses from a feature detector can be highly unreliable. On the other hand, a severe problem of an exhaustive search is its complexity. Learning a $|V|$ parts model from a pool with N_f features take $\mathcal{O}(N_f^{|V|})$ time. For a model with more than 20 parts, learning becomes intractable (Carneiro and Lowe, 2006). Moreover, gradient descent based optimization for exhaustive search can easily get stuck in a local minimum due to the large number of parameters. Therefore for a model with more than a few parts (e.g. 20), the maximum likelihood approach is not practical. To overcome this problem, we take the approach proposed by Bar-Hillel et al. (2005a,b), with which each part is treated as a weak classifier that can be trained by Boosting. In doing so, the computational complexity of the model becomes linear to $|V|$ and N_f. However, since there is no global constraint in our model and each part is dependent on its neighbors (to avoid over-rigidity in model configuration), errors produced in the previous parts could propagate to the following parts making them less reliable. This can also be a sever problem since typically the first few parts in model fitting are least reliable due to the lack of imposing any model structural constraints. To tackle this problem, we introduce a back-updating process which enables the old parts to be updated/replaced accordingly when a new part is allocated. As a result, the structure of our model can be learned automatically with each part added incrementally.

1.3 Localization

Existing part-based object recognition methods such as the star model (Bar-Hillel et al., 2005a; Fergus et al., 2005) and the k-fans (Crandall et al., 2005; Crandall and Huttenlocher, 2006) take typically a center voting scheme for object localization. They first compute an appearance response map by convolving the test image with model parts. Part locations are then approximated by the positions corresponding to ranked maximum values in the corresponding appearance map. An object center is then voted individually from the position of each part plus a learned offset (spatial relation). This method works well for translation transformation and for small variations in object shape, given a predominant single object in a scene. However, these methods always fail when dealing with relatively large affine transformations in scale and rotation as shown in Figure 1. This is because that rather rigid spatial relations are imposed by the model using a global geometric constraint, such as the center of the star model and the reference part of k-fans. This global geometric relation is not invariant to large affine transformations in scale and rotation.

To tackle these problems, we present a different method by utilizing scale and rotation invariant local topological constraints. Briefly, we construct an arbitrary graph model based on triangular triplet of nodes as its constituents. Every node in the model connecting to its two nearest neighbors forms a triangle. The inner angles and distance ratio between nodes in a triangle are taken as spatial relations for the corresponding model parts. These scale and rotation invariant features enable our model to deal with sever affine transformations (see experiment section). In the following we first describe our model structure.

2 Model Structure

Our model is a part-based arbitrary two nearest neighbor graph $\Gamma(V, E)$ with nodes set V and edges set E representing parts and their spacial relations respectively. A part is an image patch feature of the same category of objects extracted from training images. Spacial relations between parts are characterized by their local topological and relative geometric measures.

More precisely, let $\mathbf{x} = [x_a, x_l, x_s]$ be a part vector with components x_a, x_l and x_s representing the appearance, local geometric constraints and scale respectively. The corresponding parameters can then be given by $\theta = [\theta_a, \theta_l, \theta_s]$.

Our object part appearance is represented by a 18-dimensional feature vector extracted from image patches using the Kadir and Brady (KB) detector (Kadir and Brady, 2001). We select the best 200 high scoring features with less overlap and bigger scale for each image I. The selected features are then represented using the first 15 DCT (discrete cosine transform) coefficients of a 11×11 image patch plus its x and y image coordinates and scale. Since the learning of our model takes an incremental approach by selecting a weak classifier (i.e. part) using boosting, its cost is linear w.r.t. the total number of image features. This enables us to utilize the whole set of features as part candidates, which is an advantage over the alternative approaches (Crandall et al., 2005; Felzenszwalb and Huttenlocher, 2005). We finally model the appearance of part k using a Gaussian distribution $p(x_a^k|\theta_a^k) = \mathcal{N}(x_a^k|\mu_a^k, \Sigma_a^k)$ with mean μ and covariance matrix Σ.

An illustration of our model is shown in Figure 2(a), in which each model part is connected to its two nearest neighbor parts. Note nearestness is with regard to geometric distance, not in appearance similarity. In a similar approach, Carneiro and Lowe (2006) build their model using K-NN graphs. The difference is that they take each pair of nodes as a basic element and model the spatial relations based on their *absolute* values. In our case, we take a triplet (3 parts) and model their *relative* geometric constraints. The benefit of our approach is to have totally scale and rotation invariant spatial relations with the same number of parameters.

As in Figure 2(a), part k is connected to two of its closest neighbors i and j. Since we do not have any absolute value in our model, we can only approximate the true part location using the mean of the corresponding patches in the training images. This is ill-posed specially for the first few parts since the true position is hard to estimate due to the inherent high error rate in any local image patch detection. To overcome this problem, we introduce a back-updating process to update/replace the selection of early parts (see section 3).

Let $g(k)$ be the function returning the approximated mean position of part k, then a set of vectors can be computed, which allows us to compute the local geometric relations. For instance, we have $\mathbf{v}_{ji} = g(i) - g(j)$, $\mathbf{v}_{jk} = g(k) - g(j)$, $\mathbf{v}_{ik} = g(k) - g(i)$ and so on for $\mathbf{v}_{ij}$,$\mathbf{v}_{kj}$ and $\mathbf{v}_{ki}$. Furthermore, we decompose the spatial parameter θ_l^k of part k into individual settings by $\theta_l^k = \{\theta_d^{jk}, \theta_d^{ik}, \theta_\phi^{kij}, \theta_\phi^{kji}, \theta_o^k\}$. Here, θ_d^{jk} represents the parameters for the distance ratio of part j and k over part i and j. θ_d^{ik} represents the parameters for the distance ratio of part i and k over part i and j. θ_ϕ^{kij} and θ_ϕ^{kji} are the parameters for angle $\angle kij$ and angle $\angle kji$ respectively. Finally, θ_o^k is a constant, defined as follows:

$$\theta_o^k = \begin{cases} 1 & \text{if a walk from } k\text{'s closest neighbor to } k \text{ then} \\ & \text{to } k\text{'s second closest neighbor is clockwise.} \\ 0 & \text{otherwise} \end{cases}$$

The definition for θ_o^k is necessary and helpful. This is because without it, a mirror-flipped triangle will have exactly the same geometric configuration as the original one. And by flipping it in the object localization process, we can easily handle with mirror-flipped objects as we will show in the experiments.

With all these settings in hand, our local geometric constrains can then be given by Gaussian distributions as follows:

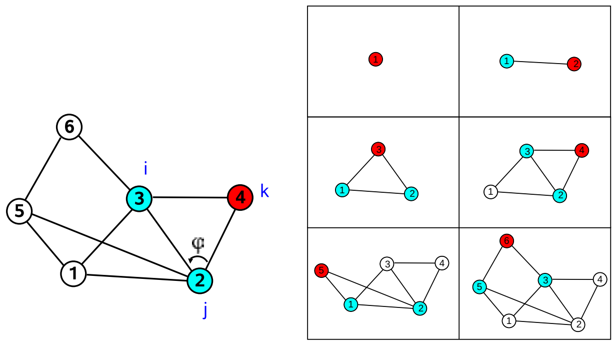
$$\begin{aligned} p(x^k|x^i, x^j, \theta_d^{jk}) &= \mathcal{N}\left(\frac{|\mathbf{v}_{jk}|}{|\mathbf{v}_{ji}|} \,\middle|\, \mu_d^{jk}, \Sigma_d^{jk}\right) \\ p(x^k|x^i, x^j, \theta_\phi^{kji}) &= \mathcal{N}\left(\arccos(\frac{\mathbf{v}_{jk}\cdot\mathbf{v}_{ji}}{|\mathbf{v}_{jk}||\mathbf{v}_{ji}|}) \,\middle|\, \mu_\phi^{kji}, \Sigma_\phi^{kji}\right) \\ p(x^k|x^i, x^j, \theta_s^k) &= \mathcal{N}\left(\log(\frac{2x_s^k}{x_s^i + x_s^j}) \,\middle|\, \mu_s^k, \Sigma_s^k\right) \end{aligned}$$

where μ and Σ are mean and covariance matrix respectively. $p(x^k|x^i, x^j, \theta_d^{ik})$ and $p(x^k|x^i, x^j, \theta_\phi^{kij})$ are also computed in a similar way. We further assume the appearance of different parts of an object is independent, i.e. $\forall i, j \in V, p(x_a^i, x_a^j) = p(x_a^i)p(x_a^j)$. As a result, for a model with $|V|$

parts, the joint probability of model parts can be factorized by

$$P(X|\Theta) = \prod_{\substack{k=1 \\ i,j \in N_k}}^{|V|} p(x_a^k|\theta_a^k) p(x_l^k|x^i, x^j, \theta_l^k) p(x_s^k|x_s^i, x_s^j, \theta_s^k) \tag{1}$$

where $N_k = \{i|(i,k) \in E\}$ represents the neighbor parts of part k.



(a) Model structure. (b) Structure learning.

Figure 2: An illustration of model structure and incremental structure learning. Each node on the graph represents a model part. Edges connecting the nodes represent spatial relations. In structure learning, newly introduced node is colored in red and its reference nodes are colored in cyan.

3 Learning

The task of learning is both building the graph structure and optimizing the parameters. The goal is to minimize a score function using a set of binary labeled ($\{+1, -1\}$) training images. The score function f is the log likelihood ratio between the probability that an image has an instance of the object and the probability that the image only consists of background.

$$f(I) = \log \frac{p(I|Model)}{p(I|BG)} - \tau \tag{2}$$

where τ is a constant threshold.

Our model is a 2-NN graph with each node representing an object part corresponding to an image patch whose appearance follows a Gaussian distribution of all corresponding patches in the training images. Every training image I has a pool of $N_f = 200$ patch features and our aim is to find a feature set $\{\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^{|V|}\}$ with each of its component corresponding to a model part. Let $F(I)$ be the feature set obtained from image I, the probability of observing an object in image I is given by averaging all these configurations under the graph model.

$$p(I|Model) = Z_0 \sum_{\mathbf{x} \in F(I)^{|V|}} \prod_{\substack{k=1 \\ i,j \in N_k}}^{|V|} p(\mathbf{x}^k|\mathbf{x}^i, \mathbf{x}^j, \theta^k) \tag{3}$$

where Z_0 is a constant. Since part appearance is independent, a complete consideration of selecting $|V|$ parts from a set of N_f features is of $\mathcal{O}(N_f^{|V|})$ complexity. In order to improve the efficiency of our learning process, we take an incremental approach as in (Bar-Hillel et al., 2005a). We approximate the average feature selecting by working on the best one resulting in a decreased computational cost from $\mathcal{O}(N_f^{|V|})$ to $\mathcal{O}(N_f|V|)$.

$$p(I|Model) = Z_0 \prod_{\substack{k=1 \\ i,j\in N_k}}^{|V|} \max_{\mathbf{x}\in F(I)} p(\mathbf{x}^k|\mathbf{x}^i, \mathbf{x}^j, \theta^k) \tag{4}$$

Model parameters optimization is achieved by maximizing the log likelihood of $p(I|Model)$. Since we allow feature repetition, the result of ML can be a repetition of the same best features. To avoid this, we take the same approach as Bar-Hillel et al. (2005a) by considering each model part as a weak classifier that can be learned using boosting (Mason et al., 2000).

Given N labeled training images $\{I_i, y_i\}_{i=1}^N$, Adaboost (Schapire and Singer, 1999) minimizes the loss function $L(f) = \sum_{i=1}^N \exp(-y_i f(I_i))$ by constructing a set of weighted weak classifiers: $f(x) = \sum_{i=1}^N \alpha_i h_i(x)$ where α_i is the weight for weak classifier h_i. Our score function can then be cast into this framework by setting

$$\begin{aligned} f(I) &= \log p(I|Model) - \log p(I|BG) - \tau \\ &= \sum_{\substack{k=1 \\ i,j\in N_k}}^{|V|} \alpha^k h^k(I, \mathbf{x}^i, \mathbf{x}^j) - \tau' \end{aligned} \tag{5}$$

and the weak classifier h^k is defined as

$$h^k(I, \mathbf{x}^i, \mathbf{x}^j) = \max_{\mathbf{x}\in F(I)} \sum_{p=1}^{3} \frac{\lambda_p^k}{\sum_{p=1}^3 \lambda_p^k} \log\left(p(x_p^k|x_p^i, x_p^j, \theta_p^k)\right) \tag{6}$$

$$\lambda_p^k > 0, p = 1, 2, 3$$

where $\frac{\lambda_p^k}{\sum_{p=1}^3 \lambda_p^k}$ measures the relative weights of the appearance, local geometric constraint and scale components. We model background probability as a constant: $\log p(I|BG) = \tau' - \tau$. A more detailed description on how to compute h^k and α^k can be found in (Bar-Hillel et al., 2005a).

Every time a new part is added, it brings informative evidence to the model. To utilize this new information, we update old parts by maximum a posteriori (MAP) of existing parts. We use belief propagation (BP) (**?**) to pass updated information (messages) between parts in the object graph configuration model. For example, to update part j in tth iteration, we have

$$\hat{\mathbf{x}}_{MAP}^j = \arg\max_{\mathbf{x}^j} p(x_a^j) \prod_{k\in N_j} M_{kj}^t \tag{7}$$

and message M_{kj}^t from part k to part j is given by

$$M_{kj}^t = \max_{x^k} \Phi(x_l^j, x_l^k) p(x_a^k) \prod_{l\in N_k - \{j\}} M_{lk}^{t-1} \tag{8}$$

where M_{lk}^{t-1} is the message passed from part l to part k in iteration $t-1$, and $\Phi(x_l^j, x_l^k)$ is a potential function defined as:

$$\Phi(x_l^j, x_l^k) = p(x^k|x^j, \theta_d^{jk})p(x^k|x^j, \theta_\phi^{kji}) \tag{9}$$

Here, we model the potentials between two parts using their geometric constraint. Not taking scale into account is because the scale of a part is referenced on both of its neighbors and considering only one part can be erroneous.

As we will show in the experiment, this back-updating process is important. Since we do not have a global constraint in our model, earlier errors can be easily propagated afterwards. With back-updating, most of the errors can be fixed in time with the new evidence.

4 Object Localization

Our aim is to tackle the difficulties in object localization due to large scale and rotation changes. In order to locate objects with large scale changes, our localization process takes a sliding window approach as follows.

Given a test image, we first compute $|V|$ appearance responses. Each response corresponds to one of the model part and obtained by convolving the test image with the part appearance. Then we slide the sliding window with various sizes through these appearance responses synchronously. Local maximums are computed and the best N_c are taken as part candidates. Ideally, we could construct a 2-NN graph for each candidate in every part and match this graph with our model. The matching score will indicate where the object is likely to lie. The problem with this approach is its computational complexity. For a model of $|V|$ parts and if we choose the best N_c candidates for each part, there are $N_c^{|V|}$ possibilities. Computational overload will increase exponentially with the number of model parts. How to speed up the process of object localization (i.e. model fitting) is a major consideration. To overcome this problem, we take an incremental fitting approach. This is performed by first finding a triangle (three parts) among the candidates which best fits the model in terms of both appearance and spatial relations. Once this triangle is located, the rest parts of the model can be added in incrementally.

The criteria of choosing a triangle is given by the weighted score for both appearance and spatial relational constraints. Score for appearance is the value from the appearance response and score for spatial constraints is obtained by comparing angles, distance ratios and scales of the candidate triangle with the corresponding one in our model. Let c be a triplet comprising of three candidates (local maximums in appearance response) from the corresponding three parts. The total triplet candidates set can be given by $C = \{c_1, c_2, ..., c_{N_c^3|V|}\}$ where $N_c^3|V|$ is the total number of possible triplets under model configuration. Our aim is to find the best triplet c^* which satisfies:

$$c^* = \arg\max_{c \in C} \left(\lambda_1' \sum_{k \in c} \alpha^k LG(x_a^k) + \sum_{\substack{k \in c, i \in c-\{k\} \\ j \in c-\{k,i\}}} \big(\lambda_2' LG(x_l^k|x^i, x^j, \theta_l^k) + \lambda_3' LG(x_s^k|x_s^i, x_s^j, \theta_s^k)\big)\right) \tag{10}$$

where $LG(x)$ stands for $\log(p(x))$ and $\lambda_i' = \frac{\sum_{k \in c} \lambda_i^k}{\sum_{p=1}^{3} \sum_{k \in c} \lambda_p^k}$ is the averaged weight of appearance, geometric constraint and scale components for the triplet. α^k is the weight of part k which has been learned in Equ. 5.

There are a few things need to be addressed in Equ. 10. First of all, after the model is learned, we will be able to compute all the relative spatial relations in every triangle in the model. This allows us to compute an average score of the spatial relations in the triangle, and the score of any adjacent part based on the established ones (Equ. 11). Second, parameter θ_o is inclusive in θ_l, which means if the orientation of a triangle is wrong, we will set its score on the spatial component be zero. Finally, The appearance score of a part candidate is weighted by the corresponding part weight α. This is reasonable since a more discriminative part is easier to be detected and as a result, it should have higher weight. However, a good score in appearance does not guarantee it is the right one. It has to be weighted by the geometric constraint. As we will show in the experiment, this property enables us to deal with the situation when the appearance response is weaken by occlusion or affine transformations.

With this triangle established, the computational cost for selecting the remaining parts in fitting the model is linear to the number of candidates. A selection of part x^* can be given by

$$\begin{aligned} x^* = \arg \max_{\substack{\mathbf{x}^k \in X - c^* \\ i,j \in N_k}} \Big(& \lambda_1^k \alpha^k LG(x_a^k) \\ & + \lambda_2^k LG(x_l^k | x^i, x^j, \theta_l^k) + \lambda_3^k LG(x_s^k | x_s^i, x_s^j, \theta_s^k) \Big) \end{aligned} \tag{11}$$

We start from the part which is adjacent to the triangle and finish when all parts have been allocated or no candidate left is above a threshold. The overall complexity now decreases to $\mathcal{O}\left(N_c^3 |V| + (|V| - 3) N_c\right)$.

We further speed up the object localization process by setting a threshold for the first triangle selection score. Sliding window without a good starting triangle will be immediately ignored. This proves to be very efficient in practice since low response areas like the sky, water etc will be quickly passed. [1]

5 Experiments

The first part of our experiment is carried out on benchmark data. The aim is to compare the performance of our method with other existing techniques using the same data. Next, to illustrate the effectiveness of our method on datasets with large affine changes in scale and orientation, we test our model on some new challenging images unused before. Finally, we gave some analysis on two failed examples from using our method.

5.1 Model Learning

For comparing with (Bar-Hillel et al., 2005a; Crandall et al., 2005; Crandall and Huttenlocher, 2006; Fergus et al., 2003, 2005), we used the Caltech datasets. We evaluated our method on four subsets from the Caltech data: motor-bikes (800 images), car-sides (123 images), airplanes (800 images) and car-rears (800 images). In each dataset, we used the same number of images for training and testing as in (Bar-Hillel et al., 2005a; Crandall et al., 2005; Crandall and Huttenlocher, 2006; Fergus et al., 2003). We initially set the number of part in our model to 6, which is compa-

[1] In our supplemental material, this process can be easily observed by the dropping of sliding windows.

rable to that of k-fan (Crandall et al., 2005; Crandall and Huttenlocher, 2006) 6 parts, Fergus et al. (2003, 2005) 6 parts and Bar-Hillel et al. (2005a) 7 parts. We then experimented with increased number of parts up to 20.

Figures 5 - 7 show learned models from three of the datasets. Notice the clear semantics in terms of object's parts in each category. In Figure 5, we also show results from a motor-bike model using 10 parts as a comparison to the 6 parts model. Figure 3 shows the learning curves of our 6 parts model for each object category. We take loss f as a function of the number of model parts and for each part, we plot three points with each of them corresponding to an iteration when a part (weak classifier) is updated. The reason why the airplane dataset has a higher loss compared to the others is because that airplanes have less discriminative features over the background and as a result, it is harder to be recognized. This corresponds well to our test error rate table in Table 1 in which almost all methods performed worst on the airplane dataset. It is also interesting to notice the small lumps on the curves which occur nearly every time when a new part is added. This is due to the belief updating process which does not have enough iterations to converge with a newly arrived evidence.

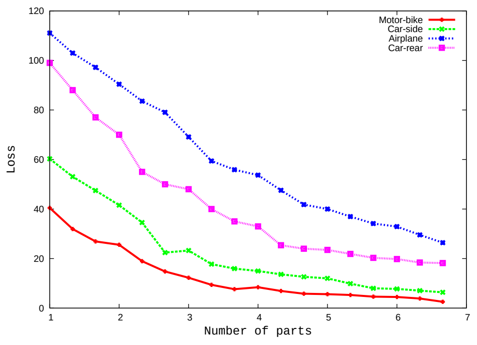


Figure 3: Learning curves for the four training datasets.

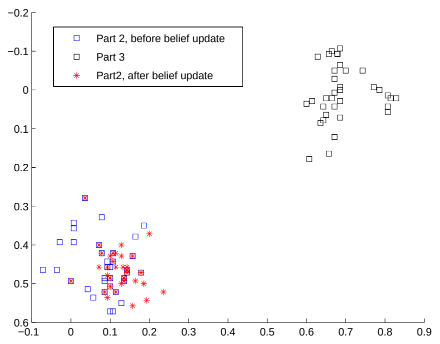


Figure 4: An illustration of BP based back-updating process.

To demonstrate the effectiveness of our back-updating process, in Figure 4, we show a simple example with two parts (part 2 and 3) taken from the motor-bike model. In this figure, blue and black squares represent the locations of corresponding patches for part 2 and part 3 respectively in the top 35 training images. Since there is only limited scale and shape variation in the motor-bike dataset, these two sets of squares should form their own clusters tightly. The red stars also represent the patch locations of part 2, but after back-updating. It is evident that the cluster formed

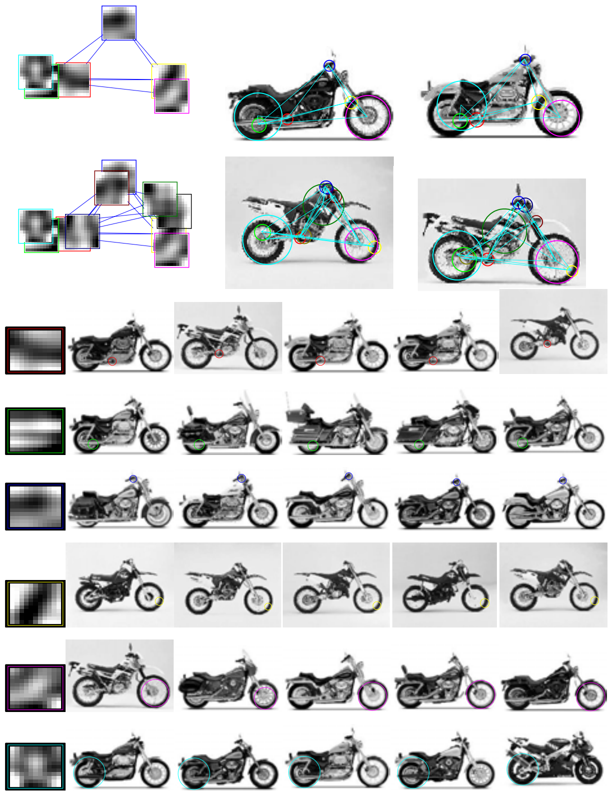


Figure 5: An illustration of our model for motor-bikes. **Top row:** A six part motor-bike model is shown on the left and its two implementations found on the test images are shown on the right. **Second row:** Corresponding to the top row, this is a ten parts motor-bike model. Notice the six part model is a subset of this ten part one. **Main panel:** An illustration of model parts and the found implementations on the test images. The leftmost column shows the mean appearance of each part and the corresponding found locations on the top five test images are shown on the rest columns.

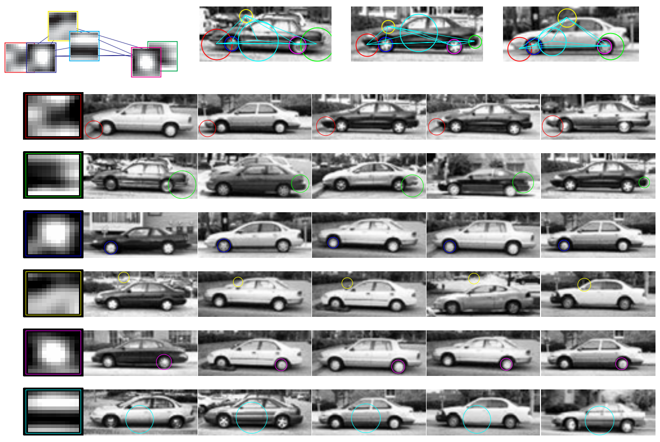


Figure 6: A six part model for cars.

by red stars is less scattered than the blue squares, representing a 23.6% decrease in the variance (scatterness) of part 2 locations.

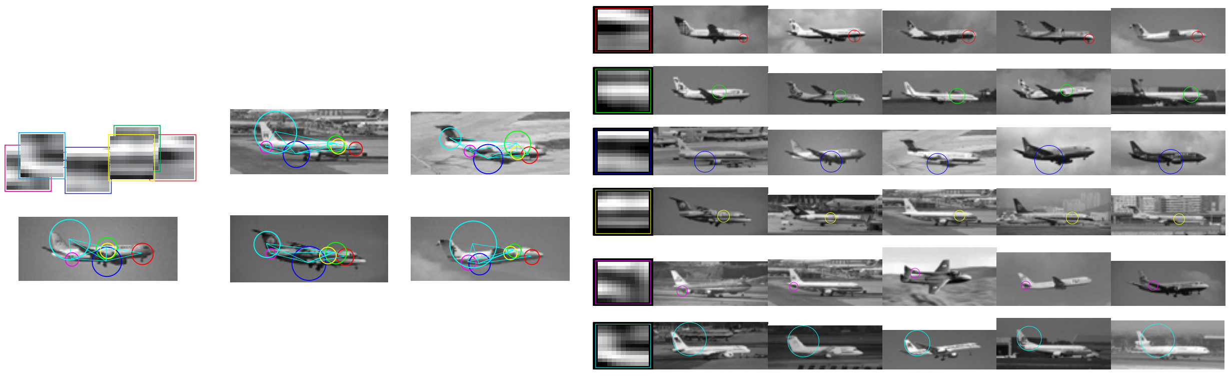


Figure 7: A six part model for planes. A plane's figure is almost observable in the model.

5.2 Recognition Results

First, we show how our localization method works on a simple example. On the left hand side of Figure 8, we show an image of a motor-bike from the Caltech dataset circulated by a red rectangle. We first rotate the image 90 degrees anti-clockwise and keep the size of the background (white) but rescaling the motor-bike object to $1/2$ of its original size. The red rectangle represents the location of a sliding window. In all our experiments, we use red rectangle for the intermediate detecting process and a blue rectangle for the final localization result. Inside a red rectangle, there are 6 colored small squares connected by blue lines. These are the detected (whether correct or not) object parts. The color pattern as well as the connectivity of the squares correspond to the motor-bike model as shown in Figure 5 (top-left). A successful localization of object parts would result in a similar geometric layout in a set of small colored squares as compared to the configuration of the model parts. For example, in Figure 8, all five parts are correctly located except the last one, the back wheel in cyan color.

On the right of Figure 8, we show 6 images with each of them corresponding to an appearance response of a part. Here 'hot' color indicates that the response is high, i.e. more likely a location of a part and 'cold' color shows the opposite. Red rectangle in each image represents the current sliding window which corresponds to the one in the object image. There are also small squares in each of the red rectangle. These are the part localization candidates, chosen from the local maximum of the appearance response and ranked according to their scores. Here we set the number of candidates to 8. Colored ones are the most preferable localization of object parts selected by our localization process according to the model graph configuration and local patch appearance score. These colored squares are also shown in the object image.

As explained in section 4, we first look for the most suitable patch triplets and then find the others in an incremental manner. In the case shown in Figure 8, the first triplet chosen with best graph configuration and local appearance are candidate 3, 6 and 2 in parts 4, 3 and 1 respectively. Candidate 7, 3 and 3 from parts 2, 5 and 6 are then added successively. To the opposite, a purely appearance based part voting scheme will give erroneous results by selecting only the best candidates in appearance.

We compare our method with other five alternatives on the same four datasets and show the test error rate in Table 1. From the table it is clear that our method performs reasonable well compared to the others, particularly for the airplane and car-rear datasets our test error is the lowest. Airplanes normally have less discriminative features (e.g. no wheels, legs or eyes) and a difficult shape for a star-like structure to model (narrow and long). Our method instead strengthens the local

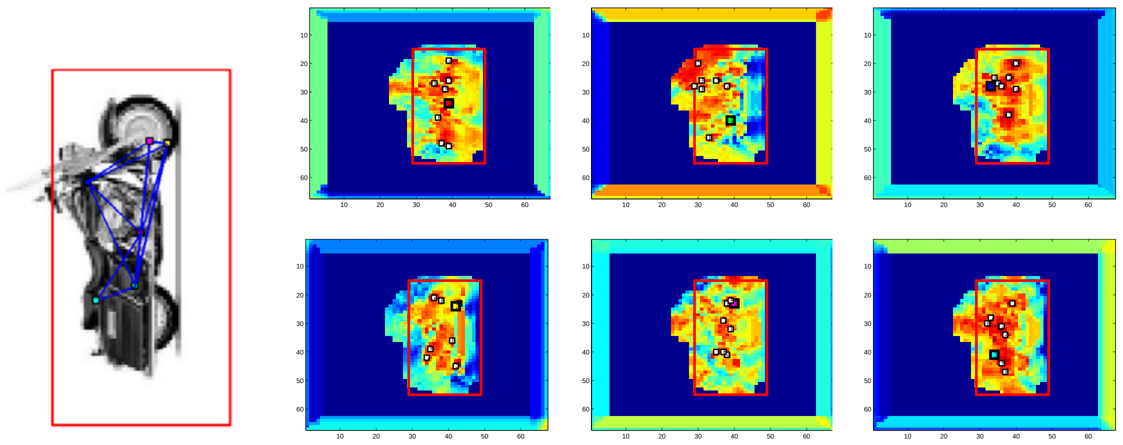


Figure 8: An example of object localization process. **Left:** Object image with sliding window on top. Parts locations and graph configuration are also illustrated. **Right:** Appearance response for each model part. They are arranged in an ascending order from top to bottom.

geometric constraints among the parts and not surprisingly, gives the best result. In addition, due to our model's inherent scale invariance, it also gives better performance on the car-rear dataset. A major cause in recognition error for car-side and motor-bike datasets comes from the ambiguity in distinguishing the front from the back wheels. Figure 5.2 shows test error as a function of the number of parts.

Table 1: **Test Error Rate (smaller the better).**

Data Name	**Constell-ation (Fergus et al., 2003)**	**Star (Fergus et al., 2005)**	**1-fan (Crandall and Huttenlocher, 2006)**	**Boosted (Bar-Hillel et al., 2005b)**	**Bar-Hillel (Bar-Hillel et al., 2005a)**	**Ours**
Motor-bike	7.5	3.0	1.4	7.2	7.8	3.6
Car-side	11.5	-	-	-	-	8.5
Airplane	9.8	8.7	5.7	14.2	8.6	4.3
Car-rear	9.7	-	5.6	6.8	1.2	1.0

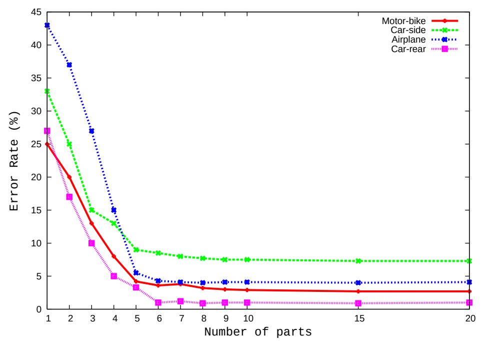


Figure 9: Error rate as a function of number of parts.

To further test our localization method, we collected a few images from the Internet containing motor-bikes, cars and airplanes with large affine variations in scale and rotation and very cluttered

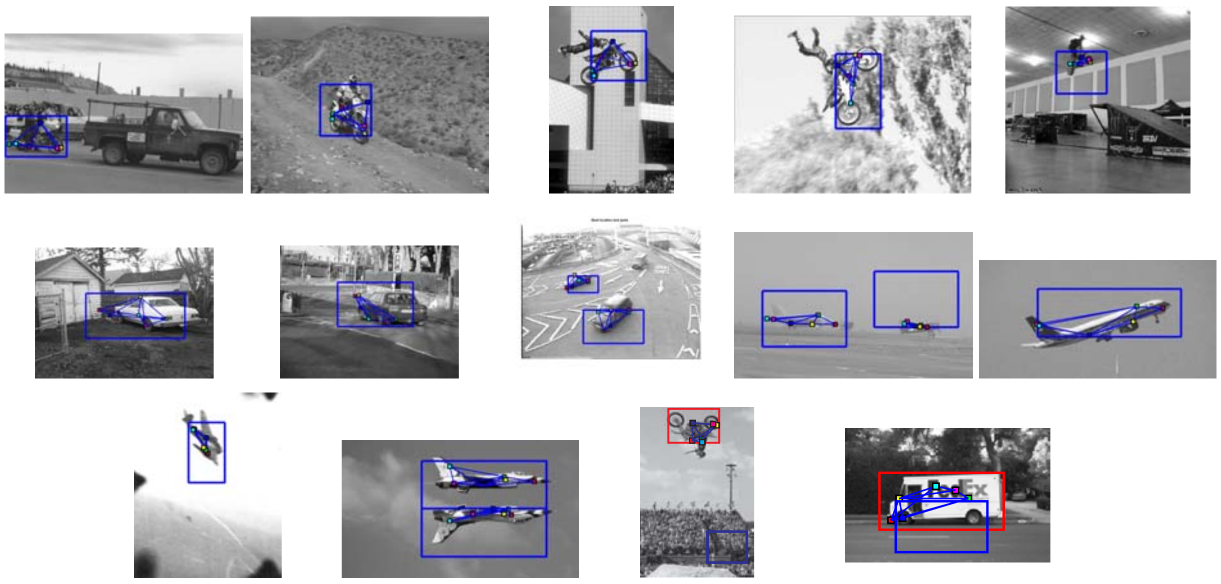


Figure 10: A few examples of our localization result on the challenging dataset. Detected object parts are shown in colored squares. Multiple objects are detected by selecting the best candidates above a threshold. Mirror-flipped object detection (like the upside-down plane) is realized by changing the triangle orientation parameter θ_o. On the last row, we show two failed examples overlaid by the detected object parts in red rectangle respectively. For the rotated motor-bike, it failed due to the weak appearance response from the front wheel and the model took the back wheel as its front instead. The van vehicle does not have an obvious rear end and the paintings on the body confuse the model to choose a letter 'd' as back wheel and letter 'E' as rear end.

background (even blurred). Images with multiple instances of objects are also considered. We tested our method on this challenging dataset and the result is shown in Figure 10. All localization processes of our model on this dataset are supplied with videos of sliding windows in the supplement material. (notice: to save space, only most preferable window sizes are shown.)

6 Conclusion

In this report, aiming at tackling the object localization problems with large affine transformations, we have developed a statistical model based on local graph configurations. We have shown how this model can be learned efficiently and effectively by taking each part as a weak classifier and using belief propagation for back-updating. Object localization is realized by incrementally model fitting, taking account of both appearance and local topological and relative geometric constraints. Experiments on standard and challenging datasets demonstrate the effectiveness of our approach.

References

A. Bar-Hillel, T. Hertz, and D. Weinshall. Efficient learning of relational object class models. In *ICCV*, pages 1762–1769, 2005a.

A. Bar-Hillel, T. Hertz, and D. Weinshall. Object class recognition by boosting a part-based model. In *CVPR*, pages 702–709. IEEE Computer Society, 2005b.

G. Carneiro and A. D. Jepson. Flexible spatial models for grouping local image features. In *CVPR*, pages 747–754, 2004.

G. Carneiro and D. Lowe. Sparse flexible models of local features. In *European Conference on Computer Vision*, pages III: 29–43, 2006.

D. Crandall, P. F. Felzenszwalb, and D. P. Huttenlocher. Spatial priors for part-based recognition using statistical models. In *CVPR (1)*, pages 10–17, 2005.

D. Crandall and D. Huttenlocher. Weakly-supervised learning of part-based spatial models for visual object recognitio. In *ECCV*, 2006.

C. Dance, J. Willamowski, L. Fan, C. Bray, and G. Csurka. Visual categorization with bags of keypoints. In *ECCV International Workshop on Statistical Learning in Computer Vision*, 2004.

P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005.

R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR (2)*, pages 264–271, 2003.

R. Fergus, P. Perona, and A. Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *CVPR*, pages 380–387. IEEE Computer Society, 2005. ISBN 0-7695-2372-2.

Timor Kadir and Michael Brady. Saliency, scale and image description. *International Journal of Computer Vision*, 45(2):83–105, 2001.

L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent in function space. In *NIPS*, pages 512–518, 2000.

K. Mikolajczyk, B. Leibe, and B. Schiele. Multiple object class detection with a generative model. In *IEEE Computer Vision and Pattern Recognition or CVPR*, pages I: 26–36, 2006.

R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *MACHLEARN: Machine Learning*, 37, 1999.

A. B. Torralba, K. P. Murphy, and W. T. Freeman. Sharing features: Efficient boosting procedures for multiclass object detection. In *CVPR (2)*, pages 762–769, 2004.