Contour-Based Learning for Object Detection

Jamie Shotton
Department of Engineering
University of Cambridge
jdjs2@cam.ac.uk

Andrew Blake
Microsoft Research Ltd.
Cambridge, UK
ablake@microsoft.com

Roberto Cipolla
Department of Engineering
University of Cambridge
cipolla@eng.cam.ac.uk

Abstract

We present a novel categorical object detection scheme that uses only local contour-based features. A two-stage, partially supervised learning architecture is proposed: a rudimentary detector is learned from a very small set of segmented images and applied to a larger training set of unsegmented images; the second stage bootstraps these detections to learn an improved classifier while explicitly training against clutter. The detectors are learned with a boosting algorithm which creates a location-sensitive classifier using a discriminative set of features from a randomly chosen dictionary of contour fragments. We present results that are very competitive with other state-of-the-art object detection schemes and show robustness to object articulations, clutter, and occlusion. Our major contributions are the application of boosted local contour-based features for object detection in a partially supervised learning framework, and an efficient new boosting procedure for simultaneously selecting features and estimating per-feature parameters.

1. Introduction

We address the problem of general visual object recognition. The aim is to learn from a small set of training images a class-specific model for automatic object detection in novel images, where the term detection includes both image classification and object localisation. In this paper we present a new approach to detection, and demonstrate excellent results for both rigid and articulated classes of objects (see figure 6).

We follow many recent recognition papers (e.g. [6, 1, 5]) in using a collection of *parts* to model the object class, accounting for within-class variations in appearance and shape somewhat independently, and increasing model flexibility while decreasing training data requirements. In contrast to many existing systems, however, our model copes efficiently with a large number of parts enabling the use of an over-complete model: the in-built redundancy ensures tolerance to within-class variations (different individuals, body configurations, facial expressions), imaging conditions (different lighting, occlusion, clutter, slight variations in pose), and failures of local feature detectors.

Much recent work has concentrated on texture-based features localised with generic interest-point detectors, and while results indicate the discriminative power of texture, these features can only be detected repeatably on the object interior, and so cannot effectively exploit the extremely powerful recognition cue of *contour*.

As humans we are more than capable of recognising a wide variety of objects based on 2-D outline sketches alone. It is with this intuition that we choose to explore an object recognition system that exploits *only* contour-based information. Clearly the eventual goal of any recognition system should be to combine sensibly many different useful types of feature (contour, texture, colour, etc.), but for the purposes of this paper we deliberately ignore these to show just how powerful the contour cue is. Contour has many advantages over texture: for example, it can be matched largely invariant to lighting conditions and object colour, and can efficiently represent image structures with large spatial extents.

The paper is structured as follows. Immediately below, we discuss related work. In section 2 we detail the form of our object detector, which is learned as described in section 3. The technique is evaluated in section 4, and final conclusions are given in section 5.

1.1. Related Work

We first discuss methods that employ texture-based features. The constellation model of Perona et al [3, 27, 6] represents variation in appearance, relative position and scale probabilistically, but requires expensive learning algorithms which do not scale well with the number of parts. This forces the use of a very sparse model that will lack tolerance to slight mismatches, occlusions and missed feature detections. Their test data are largely rigid objects, and while they evaluate on spotted cats the technique could not be expected to cope well with other articulated objects with much weaker texture cues despite very distinctive shape

(e.g. horses).

Torralba et al [21] convincingly demonstrate object detection sharing features between classes. Their evaluation is limited to fairly compact classes without significant articulation. Mikolajczyk et al [15] presented good results on human body detection, but required hand-chosen (and labelled) body parts for training. Agarwal & Roth [1] present a car detection scheme that required manually cropped training images. The method of [20] classifies images based on histograms of order types, though suffers with background clutter.

Borenstein & Ullman [2] present good class-specific segmentation results but do not evaluate their technique in classification or detection performance, or indeed show results on images not containing objects of the trained class. Leibe et al [14] present a scheme for interleaved segmentation and classification, though this requires all training images to have been manually segmented. Felzenszwalb & Huttenlocher [5] present the Pictorial Structures model, another sparse model requiring hand-chosen object parts.

Many recognition systems using contour match the image against whole object templates, either for particular rigid objects (e.g. [16]), or for articulated objects (e.g. people in [10, 22], and hands in [19]). The former techniques often require a full 3-D model of the object, while the latter can require a prohibitively large set of templates to represent all joint object configurations. Alternative approaches (e.g. [17]) use *fragments* (parts) of contour. Fergus et al [7] augmented the constellation model with contour fragment features, but this only exploits fairly clean, planar curves with at least two points of inflection, and suffers the same sparsity restrictions as their previous work. In [13] contour fragments learned from video sequences are arranged in Pictorial Structures and used for detection of articulated objects. This technique offers a similarly sparse model and requires either complex tracking of video sequences or manual labelling of parts.

2. Detection

In this section we specify the form of the object detector. It is built on local contour-based features matched with an oriented chamfer measure, which in boosted combination form a location-specific classifier that is used for detection.

The features F employed are local fragments of contour (i.e. local edge templates) T, spatially arranged in a *star configuration*: the expected position of each fragment is offset from the object centroid (or centre-of-mass) by vector $\mathbf{p}$. Additionally five parameters (described in detail below) are learned for each feature: the uncertainty in position of the fragment, σ, the orientation specificity of the chamfer matching, λ, the detection threshold θ, and two confidence weights a and b.

2.1. Oriented chamfer matching

Chamfer matching has proven a capable and efficient method for matching contour, especially with whole object templates (e.g. [19, 11]). It provides a fairly smooth distance measure between two contours, tolerant to considerable misalignment in position, scale and rotation. This makes it perfect for our task of matching small rigid templates somewhat invariantly against a wide variety of images.

In its simplest form, chamfer matching takes two sets of edgels, the edge map of an image, $E = \{\mathbf{e}\}$, and the template (contour fragment), $T = \{\mathbf{t}\}$, and evaluates the chamfer score as a function of relative position $\mathbf{x}$:

$$d_{\text{cham}}^{(T,E)}(\mathbf{x}) = \frac{1}{N_T} \sum_{\mathbf{t} \in T} \min_{\mathbf{e} \in E} \|(\mathbf{t} + \mathbf{x}) - \mathbf{e}\|_2 \qquad (1)$$

where N_T is the number of edgels in T. This gives the mean distance of edgels in the template to their closest edgels in the edge map. It can be efficiently computed by first evaluating the *distance transform* (DT) of the edge map. Each pixel in the DT is the distance to the closest pixel in the edge map:

$$\text{DT}_E(\mathbf{q}) = \min_{\mathbf{e} \in E} \|\mathbf{q} - \mathbf{e}\|_2 \qquad (2)$$

The exact Euclidean DT can be computed in linear time with the algorithm of [4], and so the $\min$ operation in (1) becomes a simple look-up such that $d_{\text{cham}}^{(T,E)}(\mathbf{x})$ can be computed as a correlation.

The edge map E of image I is given by the Canny edge detector. To alleviate problems arising from missing edges in E, the cost function is made more robust by thresholding the distance:

$$d_{\text{cham}_\tau}^{(T,E)}(\mathbf{x}) = \frac{1}{N_T} \sum_{\mathbf{t} \in T} \min(\text{DT}_E(\mathbf{t} + \mathbf{x}), \tau) \qquad (3)$$

for some value τ. A constant τ was found in practice to improve results marginally over no thresholding.

A further improvement can be gained by exploiting edge orientation information, in the form of edge gradients. This alleviates problems from clutter edgels in the edge map which are unlikely to align in orientation as well as position. One popular technique from [11] is to divide the edge map and template into discrete orientation channels and sum the individual chamfer scores. However, it is not clear how many channels to use, and one has to be careful to avoid artefacts at the boundaries between channels. An alternative is to use a distance in a three-dimensional space including orientation as well as position, but this is expensive and the orientation space must still be quantised.

We propose a slightly different scheme, *oriented chamfer matching*. This incorporates an explicit cost for orientation mismatch, given by the mean difference in orientation between template edgels and their nearest edge map edgels:

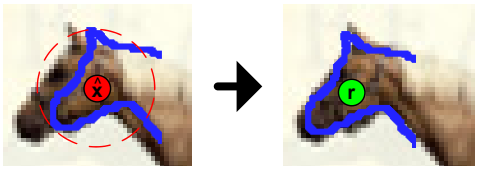


Figure 1: Illustration of detection for one contour fragment. Left: section of an image with fragment (blue) overlaid at hypothesised position $\hat{\mathbf{x}}$. The extent of $R_{\hat{\mathbf{x}},\sigma}$ is illustrated as a dashed red circle. Right: detected position $\mathbf{r}$ aligns the fragment with the image.

$$d^{(T,E)}_{\text{orient}}(\mathbf{x}) = \frac{1}{N_T} \sum_{\mathbf{t} \in T} |o(\mathbf{t}) - o(\text{ADT}_E(\mathbf{t} + \mathbf{x}))| \qquad (4)$$

where $o(\cdot)$ is the orientation of an edgel, and $|o_1 - o_2|$ gives the smallest difference in angle (modulo π)[1]. The *argument* distance transform, $\text{ADT}_E(\mathbf{q}) = \arg\min_{\mathbf{e} \in E} \|\mathbf{q} - \mathbf{e}\|_2$, can be computed simultaneously with the DT.

Our final distance measure is then the weighted sum between the distance and orientation terms:

$$d^{(T,E,\lambda)}(\mathbf{x}) = d^{(T,E)}_{\text{cham}_\tau}(\mathbf{x}) + \lambda d^{(T,E)}_{\text{orient}}(\mathbf{x}) \qquad (5)$$

where λ is an orientation specificity parameter. By learning a separate λ for each feature, one gains a fine-grained control of orientation sensitivity that could not have be obtained by dividing the edgels of E and T into discrete orientation channels.

2.2. Geometric modeling

The simple star-shaped constellation employed is, for the single pose detection problem addressed here, flexible enough to cope with large variation in shape and appearance of both rigid and articulated objects, and requires far fewer parameters than many other techniques. Thus we can build over-complete models with hundreds of parts to exploit overlap and redundancy for tolerance to occlusion and missed or incorrect part detections.

For a given centroid $\mathbf{c}$, feature $F = (T, \mathbf{p}, \sigma, \lambda, \theta, a, b)$ is hypothesised to appear at position $\hat{\mathbf{x}} = \mathbf{c} + \mathbf{p}$. This defines the centre of a search window W in the oriented chamfer distance $d^{(T,E,\lambda)}$. The weighted minimum in W then gives the best alignment of the fragment to the image:

$$\mathbf{r}(F, E|\mathbf{c}) = \arg \min_{\mathbf{x} \in R_{\hat{\mathbf{x}},\sigma}} \left(d^{(T,E,\lambda)}(\mathbf{x}) + W(\mathbf{x}|\hat{\mathbf{x}}, \sigma) \right) \qquad (6)$$

from which the feature response is trivially calculated:

$$v(F, E|\mathbf{c}) = d^{(T,E,\lambda)}(\mathbf{r}(F, E|\mathbf{c})) \qquad (7)$$

This is illustrated in figure 1.

In (6), $d^{(T,E,\lambda)}$ acts as the negative log likelihood for centroid position, as motivated by [22]. The isotropic windowing function acts as the prior, and is $W(\mathbf{x}|\hat{\mathbf{x}}, \sigma) = -\log \mathcal{N}(\mathbf{x}|\hat{\mathbf{x}}, \sigma)$, a negative log Gaussian centred at $\hat{\mathbf{x}}$ with variance σ^2. The set $R_{\hat{\mathbf{x}},\sigma} = \{\mathbf{x} \text{ s.t. } \mathcal{N}(\mathbf{x}|\hat{\mathbf{x}}, \sigma) \geq \epsilon\}$ for a small constant ϵ.

The use of the original, unweighted cost for v was seen to improve results slightly over the weighted cost. This can be attributed to the need to compare values for features regardless of where in the window they were detected. Note that σ is a parameter learned for *each feature individually* to model the differing spatial distributions that different features have.

2.3. Detection as classification

The object detector K takes the form of a location-specific classifier, returning a confidence value $K(\mathbf{c})$ for object centroid $\mathbf{c}$. Evaluating the classifier for $\mathbf{c}$ spanning a grid over the image gives a classification map (similar to the Classifier Activation Map of [1]). For efficiency, we use a grid of resolution $2\Delta_1$ pixels, i.e. classification responses are evaluated at $\mathbf{c} = (2\Delta_1 i, 2\Delta_1 j)$ for integers i and j. The classification map is smoothed slightly to reduce noise, and local maxima then give possible object detections. These are filtered to remove the weaker maxima in pairs of nearby detections, before a global threshold produces the detection result. By varying the threshold (representing the ratio of class priors) one trades between missed and incorrect detections, as visualised in the recall-precision curves given in the evaluation.

The classifier uses the learned set of features $\mathcal{F} = \{F_m\}_{m=1}^M$ where each feature $F_m = (T_m, \mathbf{p}_m, \sigma_m, \lambda_m, \theta_m, a_m, b_m)$. It takes the form of an additive model:

$$K(\mathbf{c}) = \sum_{m=1}^{M} a_m \delta(v(F_m, E|\mathbf{c}) > \theta_m) + b_m \qquad (8)$$

Each term in this sum is a *decision stump*, using a zero-one indicator function δ. The hard detection threshold is analogous to the threshold used in interest point detectors but is learned individually for each feature so can be more discriminative. The values a_m and b_m weight the classification confidence of feature F_m.

The classification map can be evaluated efficiently as follows. First the Canny edges of image I are extracted, giving E. For each feature F_m, the oriented chamfer distance $d^{(T_m,E,\lambda_m)}$ is calculated for all points in the image. Then for each hypothesised centroid $\mathbf{c}$ in the classification map, the responses $v(F_m, E|\mathbf{c})$ are calculated, and combined with (8) to give $K(\mathbf{c})$ as required. An efficient, linear-time search strategy such as [4] can be employed to find $v(F_m, E|\mathbf{c})$ for all $\mathbf{c}$ simultaneously, since W is convex.

3. Learning

The object detector is learned in a two stage, partially supervised manner[2], bootstrapping to induce labels on un-

[1] For exterior contour it is sensible to take edge directions modulo π, since the sign of the edgel gradient is unimportant. Situations where *interior* contour might have a repeatable sign, such as repeatable object shading are not explored here.

[2] We choose the term partially supervised to distinguish from the precise definitions for supervised, semi-supervised, and unsupervised that are used

labelled data.

In STAGE 1, a fragment dictionary $\mathcal{C}$, and a rudimentary detector K_1 are learned from a small set D_S of images paired with binary segmentation masks, which delineate external contour and localise object centroids. The segmentations can be generated automatically using e.g. a stereo-based system [12], semi-automatically using e.g. [18], or manually; since the number N_S of images in D_S is very small (only 10 in our evaluation) this is not a heavy requirement. A background set D_B of non-class images is also used to ensure the detector is class-specific and to regularise the parameters learned for each fragment.

The detector K_1 performs fairly well as demonstrated in figure 5(a), but performance is improved in STAGE 2. A second, larger set D_U of *unsegmented* class images is used for training with object centroids estimated by bootstrapping with detector K_1. Additionally, K_1 is evaluated on the background dataset D_B to find false detections, allowing explicit training against clutter. Finally, datasets D_S, D_U, and D_B are combined with knowledge of object and clutter locations to form labelled dataset D_L, from which detector K_2 is learned using the same algorithm as for K_1.

Note that we will refer to the number of unsegmented images as N_U, the combined number of segmented and unsegmented images as $N_\mathrm{SU} = N_\mathrm{S} + N_\mathrm{U}$, and the number of background images as N_B. Also note that while most training and test images are provided in colour, no colour information is used anywhere in our system; in fact the only image information used is given by the Canny edge detector.

3.1. Building a fragment dictionary

The first step of STAGE 1 is to randomly generate a class-specific dictionary $\mathcal{C} = \{(T, \mathbf{p})\}$ of spatially positioned contour fragments, as illustrated in Figure 2. Note that $T = \{\mathbf{t}\}$ is a set of edgels (edge-elements) where $\mathbf{t} = (x, y)$ is relative to the *fragment* centre (not the object centroid). The learning algorithm will later select a subset of this dictionary and estimate per-feature parameters $(\sigma, \lambda, \theta, a, b)$, forming the set of features $\mathcal{F}$ used for detection.

The dictionary is derived from D_S, the small training set of images paired with binary segmentation masks. From this is extracted at random a large number $N_\mathcal{C}$ of rectangular regions of mask, constrained to contain at least 5% of each of foreground and background to ensure some external contour. The position $\mathbf{p}$ of the centre of each region relative to the centroid is trivially calculated; we assume this is a good, repeatable estimate of fragment location.

somewhat loosely in the Computer Vision literature. In terms of classification, an *unsupervised* algorithm is not given any class labels for training data items, a *semi-supervised* algorithm is given only some of the class labels, whereas a *supervised* algorithm is given class labels for all data items.

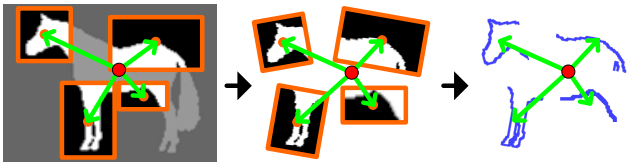


Figure 2: Building a class dictionary of spatially localised contour fragments from segmentation masks. Left: Random rectangular regions of mask are paired with their positions (green arrows) relative to the object centroid (red circle). Middle: These are randomly perturbed. Right: Edges are calculated to form contour fragments. (In practice many more than four fragments are taken from each training mask).

Each region is slightly perturbed by applying a random transformation about its centre, composed of a scaling (within a factor α, both in x and y), a rotation ($\pm\beta$), and a translation of $\mathbf{p}$ ($\pm\gamma$, both in x and y). The perturbation ensures that even for a very small set D_S a representative dictionary can be created, and improves results markedly in figure 5(a).

The final stage is to generate T, the set of edgels. Exterior contour is calculated as edges in the transformed region of mask. Interior contour is also included, derived from the Canny edge map of the corresponding transformed region of image. This allows repeatable internal contour (from e.g. car wheels or facial features) to be used.

3.2. Learning a Classifier

Boosting has developed (e.g. [8, 9]) as a simple yet powerful technique for building an accurate classifier from a set of 'weak learners,' classifiers that need only perform at a level just above random guessing. Boosting has been used very successfully in Computer Vision, primarily as a feature selection mechanism: Viola & Jones [26] used Adaboost to create a very efficient cascade of classifiers for face detection from very simple image features; Torralba et al [21] showed how to share features between multiple classes in a boosting framework.

We employ a boosting algorithm for three purposes: feature selection, parameter estimation, and learning a classifier. The authors believe that the explicit estimation of *model* parameters (i.e. excluding those directly connected with the weak learners) through boosting has not been suggested before. We present a new, efficient *parameter refinement* technique for this purpose that should be applicable to any standard boosting algorithm.

Training examples

Each training image can generate multiple training *examples*. Each example is a vector of feature responses taken at a particular centroid and is given a binary target value, positive, meaning object present in this image at this centroid, or negative, meaning object not present. This en-

Figure 3: For each training image, N_{E} example centroids are used. The positions of positive examples are shown with pluses, and negative examples are shown with minuses. In (a) a class image generates examples relative to the known centroid, to encourage localisation in the classification map. In (b) a background image generates negative examples at false detections of K_1, so clutter can be explicitly trained against.

ables two things: to encourage good localisation of detection responses in the classification map, and to explicitly train against clutter in the background images.

For class training images such as in figure 3(a), the true centroid $\mathbf{c}$ is known. To encourage the classification map to have a flat peak of width $2\Delta_1$ around $\mathbf{c}$, allowing slight translation invariance and hence efficient evaluation, positive training examples are taken at $\mathbf{c}$ and four offset centroids close by: $\mathbf{c} + (\pm\Delta_1, \pm\Delta_1)$, for small Δ_1. To encourage the classification map to then drop off rapidly away from $\mathbf{c}$, negative examples are taken surrounding the object at $\mathbf{c} + (\pm\Delta_2, 0)$, $\mathbf{c} + (0, \pm\Delta_2)$, and $\mathbf{c} + (\pm\Delta_2, \pm\Delta_2)$, for larger Δ_2, effectively regularising the spatial uncertainty parameters σ.

For background training images such as in figure 3(b), all training examples must clearly be negative, but it is not obvious which centroids to use since no object is present. In STAGE 1, $\mathbf{c}$ is set to the image centre, and the same set of offset examples as before is taken (though all marked negative) to learn detector K_1. In STAGE 2 however, K_1 is evaluated on set D_{B} to find regions of background images at which the detector is incorrectly firing, usually in areas of clutter. Hence, negative examples are taken at the N_{E} worst detections in each background image to ensure the boosting algorithm trains against these problems when learning K_2. A demonstration of the power of this technique is shown in figure 5(a).

To summarise, in STAGE 1 there are a total of $N_{\mathrm{E}}(N_{\mathrm{S}} + N_{\mathrm{B}})$ training examples, and in STAGE 2 there are a total of $N_{\mathrm{E}}(N_{\mathrm{SU}} + N_{\mathrm{B}})$ training samples. Note that $N_{\mathrm{E}} = 13$ is simply the number of example centroids used (see figure 3(a)); this number was seen to give a good trade-off between accuracy and efficiency.

Boosting algorithm

Due to space constraints we limit our explanation of the boosting algorithm to an overview. The particular variant used is called 'gentle-boost' and interested readers are referred to [9] and [21] for details. The algorithm takes a set of training examples, each with a feature vector and target value, and greedily builds a classifier K over a number of rounds, or iterations: at each round, the weak learner that most reduces a cost function J is found and added to the current classifier. After each round, all training examples are re-weighted so that those poorly classified are given more influence at the next round.

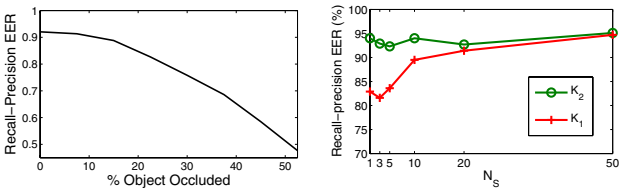
Before boosting begins, and for each example, a feature vector $\mathbf{v}$ is calculated giving the feature response $v(F, E|\mathbf{c})$ for all $F \in \mathcal{F}^*$, a set of candidate features. The weak learners used are decision stumps $h = a\delta(v^f > \theta) + b$, where a and b weight the confidence of the particular weak learner, v^f indexes dimension f in $\mathbf{v}$, and θ gives an activation threshold (cf. terms in sum (8)). Each round of boosting searches for a minimum cost J, over all f and a discrete set for θ, and then can fit a and b analytically (see [21]).

Parameter estimation and refinement

The novel aspect of our algorithm is the choice of set $\mathcal{F}^*$. If the fragment dictionary were used, $\mathcal{F}^* = \mathcal{C}$, boosting would give the standard feature selection algorithm. However, we push the concept one stage further, by letting candidate features represent combinations of contour fragments *and their parameters* σ and λ, and so expand set $\mathcal{F}^*$ to the Cartesian product between dictionary $\mathcal{C}$, a discrete set of σ parameters, and a discrete set of λ parameters. Hence each round of boosting will pull out a feature dimension f corresponding to a particular combination of $(T, \mathbf{p})$, σ and λ, in addition to the weak learner parameters θ, a and b. After M rounds of boosting a feature set $\mathcal{F} \subset \mathcal{F}^*$ is generated, containing M features.

One must therefore discretise the parameter spaces for σ, λ and θ. Care is needed to ensure this can be done efficiently; naïvely discretising the parameter spaces at an adequate resolution will be very costly. We propose a coarse-to-fine approach for each round of boosting. First, the parameter space is sampled coarsely with $\sigma_a \in \Sigma_a$, $\lambda_a \in \Lambda_a$ and $\theta_a \in \Theta_a$. This gives a relatively small set of candidate features $\mathcal{F}^*$ and allows standard optimisation of cost J.

Then, at each round, after J has been optimised but before the re-weighting of training examples, a *parameter refinement* step is performed: fixing the chosen fragment $(T, \mathbf{p})$, the parameter spaces are more finely sampled about the coarse values σ_a, λ_a and θ_a, and a new weak learner that most decreases J is chosen. The finer samplings are written $\sigma_b \in \Sigma_b(\sigma_a)$, $\lambda_b \in \Lambda_b(\lambda_a)$ and $\theta_b \in \Theta_b(\lambda_a)$. This step requires feature responses to be calculated on-the-fly for the new combinations of parameters, but in total a much smaller number of feature response evaluations are needed than had the parameter space been sampled initially at the fine resolution. The resulting parameters $(\sigma_b, \lambda_b, \theta_b)$ are guaranteed not to increase J. The refinement idea fits well with the greedy strategy that boosting already employs, and improves results (see figure 5(a)).



(a) (b)

Figure 4: (a) Occlusion performance on the horse dataset. See examples figure 6 (bottom left). (b) Comparing the performance of classifiers K_1 and K_2 on the face dataset as N_S is varied (see text).

4. Evaluation

We evaluated our technique on four classes of objects: horses, cars, faces and motorbikes. In all cases the method parameters were kept the same: $\Delta_1 = 6$ pixels, $\Delta_2 = 40$ pixels, $N_{\mathrm{E}} = 13$, $\epsilon = \frac{1}{1000}$, $\alpha = 1.2$, $\beta = 15°$, $\gamma = 5$ pixels, $M = 100$, $\tau = 15$ pixels, and $N_{\mathcal{C}} = 1000$. These were manually given sensible values, but were not hand-optimised. For each class, learning takes between 1-4 hours, after which detection can be performed in roughly 10 seconds per image (for an unoptimised implementation in C# on a 3GHz Pentium 4). Most of the time, both for training and testing, is spent evaluating the oriented chamfer distance; this could be optimised in hardware, and the algorithm is highly parallelisable.

In each experiment below, unless stated otherwise, a total of just $N_{\mathrm{SU}} = 50$ images of the class in question and $N_{\mathrm{B}} = 50$ background images were used for training, but of the class images, note that only $N_{\mathrm{S}} = 10$ were paired with segmentations.

Results are quantified in terms of recall-precision curves.[3] These plot recall (correct detection rate) against precision (the proportion of total detections that are correct) as the global detection threshold is varied. Ground truth object locations are known and a correct detection is marked when a peak lies within 25 pixels of the correct centroid. In each experiment, the detector was also evaluated on an equally sized background dataset. We give quantitative results below, and detection visualisations for horses, faces and cars are shown in figure 6.

Weizmann Horse Dataset [24]

We evaluated the performance of our detector on a very challenging dataset of side-on horse images; this dataset has been used for evaluating segmentation algorithms [2], but we do not know of published results for detection. Horses have very high within-class variation in shape, colour and texture, and present an extremely challenging class of objects with which to test a detector. Horses were investigated briefly in [7] but poor results were obtained.

[3]Note that ROC curves are not ideal for the task of detection, as opposed to classification; see [1] for more motivation.

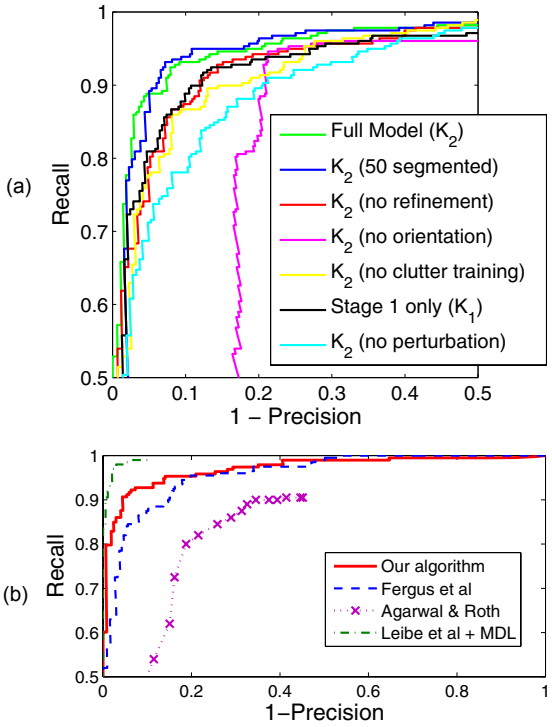


Figure 5: Recall-precision results. (a) The contribution of each aspect of our model on the horse dataset. (b) Comparative results on the car dataset.

The dataset consists of pairs of horse images and their segmentations which give ground-truth object locations (only $N_S = 10$ segmentations are used for training). The images are approximately scale normalised for this experiment.The test set consists of 277 horse images and 277 background images.

We show recall-precision results in figure 5(a) comparing the influence of various aspects of our model. Detector K_2 (green curve), using our full model, gives a recall-precision equal error rate (EER) of 92.1%. If the model is trained with $N_{\mathrm{S}} = 50$ segmented images it performs only marginally better, showing the power of the partially supervised learning methodology. Similarly, the rudimentary detector K_1 trained on only $N_{\mathrm{SU}} = 10$ positive examples performs worse than K_2. Other curves in the graph show the results for K_2 with various aspects of the model switched off individually: parameter refinement, oriented chamfer matching (setting $\lambda = 0$), explicit training for clutter in STAGE 2, and perturbation when building the fragment dictionary; in all cases the full model performs better. Note the especially large degradation when $\lambda = 0$, showing the importance of orientation in the distance measure.

The main point to note is that bootstrapping is very effective and that weakly labelled data can be combined effectively with more strongly labelled data. In fact the results suggest the partially supervised technique is almost as ef-

fective as having segmented the whole training set.

We also evaluated this dataset for tolerance to occlusion, by randomly overwriting square regions of test images such that a fixed proportion of the *object* was covered. The results are shown in figures 4(a) and 6; even when 30% of the object is occluded, an EER detection rate of almost 80% is still attainable. This shows the power of redundancy in our over-complete model.

UIUC Car Dataset [23]

In this experiment we compare our performance on the UIUC side-on car dataset against other published results. Training used only $N_{\mathrm{SU}} = 50$ images of cars facing left (with $N_{\mathrm{S}} = 10$) and $N_{\mathrm{B}} = 100$ background images. Testing was performed on 164 images containing 193 cars,and 164 background images. Since the test images had cars present facing left and right, the detector was run over each image twice, the second time with the image flipped in x. The detections (local maxima of the classification map) were combined as in section 2.3. Figure 5(b) shows our recall-precision results together with those of [6, 1, 14]. For the very small set of segmented training images required our detector gives the best results. The precision-recall EERs are compared here:

	N_{SU}	N_{S}	R-P EER
[1]	~ 550	0	$\sim 79\%$
[6]	~ 250	0	88.5%
[14]	50	50	91.0%
Ours	**50**	**10**	$\mathbf{92.8}\%$
[14] + MDL	50	50	97.5%

We are beaten in the last result by [14]. The reason for this is twofold. Firstly, they train on much higher resolution examples (not from the UIUC dataset) and use 50 segmented images. Our technique relies on edge features and higher-resolution training and test images would certainly improve results. Secondly, they employ a post-processing, minimum description length (MDL) technique to filter out false positives due to 'phantom' cars being proposed between cars parked end-to-end. The MDL criterion could be applied to our results as our detector suffers the same problem, mainly due to wheels being used as particularly informative features. However, to keep our technique completely general (since our goal is an object detector, not a car detector), we have not applied this. Without the MDL criterion, [14] only achieve an EER of 91%, which we improve upon with 92.8%.

Caltech Face Dataset [25]

A detector was learned for the Caltech face dataset. For speed of learning and testing all images were resized to 15% of their original size. The detector was tested on 400 novel face images and 400 background images. The following table gives our results (for comparison purposes an ROC EER is given). Note that we observe almost identical performance as [6], having learned from far fewer unsegmented training images, a benefit in both collecting the dataset and also training time (they state 24-36 hours training time, versus about 1-4 hours for our models).

	N_{SU}	N_{S}	ROC EER	R-P EER
[6]	~ 220	0	96.4%	-
Ours	**50**	**10**	$\mathbf{96.5}\%$	$\mathbf{94.0}\%$

As an additional experiment to investigate the improvement that classifier K_2 (learned in STAGE 2) gives over K_1 (learned in STAGE 1), we evaluated the performance of our algorithm on this dataset as the number of *segmented* training images (N_{S}) is varied while keeping the total number of training images constant ($N_{\mathrm{SU}} = 50$). The results are given in figure 4(b) and show the consistent and considerable improvement gained in STAGE 2. Note that (i) adding more segmented data improves results, (ii) the performance difference between K_1 and K_2 decreases with N_{S}, and (iii) when $N_{\mathrm{S}} = 50$, a residual improvement remains because we explicitly train against clutter.

Caltech Motorbike Dataset [25]

Finally, we evaluated our algorithm on the Caltech motorbike dataset, with roughly scale normalised images. Testing was on 380 novel motorbike images and 380 background images, and results are given in the following table. Note again very favourable performance compared to the scale-normalised evaluation on this dataset in [6].

	N_{SU}	N_{S}	ROC EER	R-P EER
[6]	400	0	95.0%	-
Ours	**50**	**10**	$\mathbf{97.1}\%$	$\mathbf{92.4}\%$

5. Conclusions & Future Work

We have presented a novel technique for object detection based on fragments of contour, and demonstrated performance at least at the level of texture based systems. No hand-selection of parts is required, and our over-complete model is tolerant to within-class variation, different imaging conditions and occlusion. We have made contributions with our partially supervised learning methodology and a new parameter refinement addition to boosting algorithms. Our evaluation gives both comparative results with related work and also introduces a new dataset to the challenge of object recognition. The system requires a comparatively small number of training images, a factor that will become important as we move toward detecting hundreds or thousands of object classes.

Our detector is learned at a particular object scale and we plan further evaluation of it in more challenging multi-scale environments and over more classes. Incorporating texture features should be straightforward in the boosting framework. We also wish to investigate using the method of [21] for multi-pose detection. Finally, the output of our detector should give a good initialisation for a bottom-up segmentation algorithm such as [18].

Figure 6: Example detection results for horses, faces and cars. In each, the left column is the input image, the middle column is the classification map (green is positive, red negative), and the right column is a visualisation of contour fragments used for detections. For horses, note excellent performance despite clutter and artificial occlusion (last two images), and especially the silhouetted horse with which a textured based model would have failed completely. For faces, note tolerance to clutter, expression, lighting conditions, occlusion, and even detection on a cartoon. For cars, note multiple object detections and tolerance to occlusion.

Acknowledgements The authors would like to thank Bastian Leibe, Shivani Agarwal, and Rob Fergus for help with figure 5(b), and Pedro Felzenszwalb and Eran Borenstein for helpful discussions. This work has been financially supported by a Microsoft Research Studentship.

References

[1] S. Agarwal and D. Roth. Learning a sparse representation for object detection. In *Proc. ECCV*, page IV: 113 ff., 2002.

[2] E. Borenstein, E. Sharon, and S. Ullman. Combining top-down and bottom-up segmentations. In *POCV04*, 2004.

[3] M.C. Burl, M. Weber, and P. Perona. A probabilistic approach to object recognition using local photometry and global geometry. In *Proc. ECCV*, pages 628–641. Springer-Verlag, 1998.

[4] P.F. Felzenszwalb and D.P. Huttenlocher. Distance transforms of sampled functions. Technical report, Cornell, 2004.

[5] P.F. Felzenszwalb and D.P. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61(1):55–79, January 2005.

[6] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proc. CVPR*, June 2003.

[7] R. Fergus, P. Perona, and A. Zisserman. A visual category filter for Google images. In *Proc. ECCV*. Springer, May 2004.

[8] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148–156, 1996.

[9] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 38(2):337–374, 2000.

[10] D. Gavrila. Pedestrian detection from a moving vehicle. In *ECCV00*, pages II: 37–49, 2000.

[11] D.M. Gavrila. Multi-feature hierarchical template matching using distance transforms. In *ICPR98*, pages Vol I: 439–444, 1998.

[12] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother. Bi-layer segmentation of binocular stereo video. In *CVPR*, 2005.

[13] M.P. Kumar, P.H.S. Torr, and A. Zisserman. Extending pictorial structures for object recognition. In *Proc. BMVC*, 2004.

[14] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *ECCV'04 Workshop on Statistical Learning in Computer Vision*, 2004.

[15] K. Mikolajczyk, C. Schmid, and A. Zisserman. Human detection based on a probabilistic assembly of robust part detectors. In *Proc. ECCV*, May 2004.

[16] K. Mikolajczyk, A. Zisserman, and C. Schmid. Shape recognition with edge-based features. In *Proceedings of the British Machine Vision Conference*, 2003.

[17] R.C. Nelson and A. Selinger. A cubist approach to object recognition. In *ICCV98*, pages 614–621, 1998.

[18] C. Rother, V. Kolmogorov, and A. Blake. Grabcut - interactive foreground extraction using iterated graph cuts. *Proc. ACM Siggraph*, 2004.

[19] B. Stenger, A. Thayananthan, P.H.S. Torr, and R. Cipolla. Filtering using a tree-based estimator. In *ICCV03*, pages 1063–1070, 2003.

[20] J. Thureson and S. Carlsson. Appearance based qualitative image description for object class recognition. In *ECCV04*, pages Vol II: 518–529, 2004.

[21] A. Torralba, K.P. Murphy, and W.T. Freeman. Sharing features: Efficient boosting procedures for multiclass object detection. In *CVPR04*, pages II: 762–769, 2004.

[22] K. Toyama and A. Blake. Probabilistic tracking with exemplars in a metric space. *IJCV*, 48(1):9–19, June 2002.

[23] `http://l2r.cs.uiuc.edu/~cogcomp/Data/Car/`.

[24] `http://www.msri.org/people/members/eranb/`.

[25] `http://www.robots.ox.ac.uk/~vgg/data.html`.

[26] P. Viola and M.J. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR01*, pages I:511–518, 2001.

[27] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. In *ECCV (1)*, pages 18–32, 2000.