Object Detection by Contour Segment Networks

Citation for published version:
Ferrari, V, Tuytelaars, T & van Gool, L 2006, Object Detection by Contour Segment Networks. in *Computer Vision – ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006, Proceedings, Part III.* Lecture Notes in Computer Science, vol. 3953, Springer-Verlag, Heidelberg, pp. 14-28. https://doi.org/10.1007/11744078_2

Digital Object Identifier (DOI):
10.1007/11744078_2

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Computer Vision – ECCV 2006

Object Detection by Contour Segment Networks

Vittorio Ferrari[1], Tinne Tuytelaars[2] [*] , Luc Van Gool[1,2]
[1]Computer Vision Group (BIWI), ETH Zuerich, Switzerland
[2]ESAT-PSI, University of Leuven, Belgium
{ferrari,vangool}@vision.ee.ethz.ch, Tinne.Tuytelaars@esat.kuleuven.ac.be

Abstract. We propose a method for object detection in cluttered real images, given a single hand-drawn example as model. The image edges are partitioned into contour segments and organized in an image representation which encodes their interconnections: the Contour Segment Network. The object detection problem is formulated as finding paths through the network resembling the model outlines, and a computationally efficient detection technique is presented. An extensive experimental evaluation on detecting five diverse object classes over hundreds of images demonstrates that our method works in very cluttered images, allows for scale changes and considerable intra-class shape variation, is robust to interrupted contours, and is computationally efficient.

1 Introduction

We aim at detecting and localizing objects in real, cluttered images, given a single hand-drawn example as model of their shape. This example depicts the contour outlines of an instance of the object class to be detected (e.g. bottles, figure 1d; or mugs, composed by two outlines as in figure 5a).

The task presents several challenges. The image edges are not reliably extracted from complex images of natural scenes. The contour of the desired object is typically fragmented over several pieces, and sometimes parts are missing. Moreover, locally, edges lack specificity, and can be recognized only when put in the wider context of the whole shape [2]. In addition, the object often appears in cluttered images. Clutter, combined with the need for a 'global view' of the shape, is the principal source of difficulty. Finally, the object shape in the test image can differ considerably from the one of the example, because of variations among instances within an object class (*class variability*).

In this paper, we present a new approach to shape matching which addresses all these issues, and is especially suited to detect objects in substantially cluttered images. We start by linking the image edges at their discontinuities, and partitioning them into roughly straight contour segments (section 3). These segments are then *connected* along the edges and across their links, to form the image representation at the core of our method: the *Contour Segment Network* (section 4). By recording the segment interconnections, the network captures the underlying image structure, and enables to cast object detection as finding paths through the network resembling the model outlines. We propose a computationally efficient matching algorithm for this purpose (section 5).

[*] T. Tuytelaars acknowledges support by the Fund for Scientific Research Flanders (Belgium).

The resulting, possibly partial, paths are combined into final detection hypotheses by a dedicated integration stage (section 6).

Operating on the Contour Segment Network brings two key advantages. First, even when most of the image is covered by clutter segments, only a limited number is connected to a path corresponding to a model outline. As we detail in section 5, this greatly limits the choices the matcher has to make, thus allowing to correctly locate objects even in heavily cluttered images. Besides, it also makes the computational complexity *linear* in the number of test image segments, making our system particularly efficient. Second, since the network connects segments also over edge discontinuities, the system is robust to interruptions along the object contours, and to short missing parts.

Our method accommodates considerable class variability by a flexible measure of the similarity between configurations of segments, which focuses on their overall spatial arrangement. This measure first guides the matching process towards network paths similar to the model outlines, and is then used to evaluate the quality of the produced paths and to integrate them into final detections. As other important features, our approach can find multiple object instances in the same image, produces point correspondences, and handles large scale changes.

In section 7 we report results on detecting five diverse object classes over hundreds of test images. Many of them are severely cluttered, in that the object contours form a small minority of all image edges, and they comprise only a fraction of the image. Our results compare favorably against a baseline Chamfer Matcher.

2 Previous work

The construction of our Contour Segment Network (sections 3 - 4) is rooted in earlier perceptual organization works [14, 12]. However, unlike these, we do not seek to single out salient edge groups. Instead, we connect all subsequent segments in a single, global network which comprises all possible contour paths. This enables our main contribution: to perform object class detection as path search on the network.

Much previous work on shape matching has focused on class variability. Several measures of shape similarity have been proposed [2, 1]. They can distinguish objects of different classes, while allowing for variations and deformations within a class. However, these works assume the object to be in a clean image, thereby avoiding the problem of localization, and the difficulties of contour detection. Hence, the rest of this review focuses on methods handling clutter.

Our algorithm of section 5 is related to “local search” [4] and “interpretation trees” [11], as it iteratively matches model features to test image features. However, at each iteration it meets an approximately constant, low number of matching candidates (only those connected to the latest matched segment, section 5). Interpretation Trees / Local Search approaches instead, need consider a large number of test features (often all of them [4]). As a consequence, our method is far less likely to be confused by clutter, and has lower computational complexity (*linear* in the number of test segments), thus it can afford processing heavily cluttered images (with typically about 300 clutter segments, compared to only 30 in [4]). Besides, both [4, 11] expect the model to transform rigidly to the test image, while our method allows for shape variations.

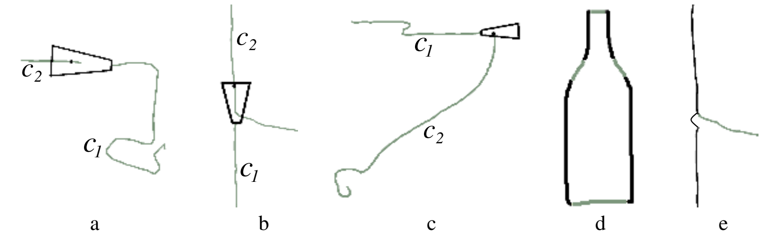

Fig. 1. *(a-c) Example links between edgel-chains. (a) Endpoint-to-endpoint link. (b) Tangent-continuous T-junction link. (c) Tangent-discontinuous link. (d) 8 segments on a bottle-shaped edgel-chain. (e) A segment (marked with an arc) bridging over link b).*

Deformable template matching techniques deform a template shape so as to minimize some energy function, e.g. diffusion-snakes [7], elastic matching [5], and active shape models [6]. These approaches require rough initialization near the object to be found. Additionally, several such methods need multiple examples with registered landmark points [6], and/or do not support scale changes [7]. Chamfer matching methods [10] can detect shapes in cluttered images, but, as pointed out by [17, 13], they need a large number of templates to handle shape variations (a thousand in [10]), and are prone to produce rather high false-positive rates (1-2 per image in [10]). Recently Berg et al. [3] proposed a powerful point-matching method based on Integer Quadratic Programming. However, the nature and computational complexity of the optimization problem require to explicitly set rather low limits on the maximal portion of clutter points, and on the total number of points considered from the test image (via a sampling scheme). This is not suitable when the objects' edge points are only a fraction of the total in the image. Besides, [3] uses real images as models, so it is unclear how it would perform when given simpler, less informative hand-drawings. The same holds for [16], whose approach based on edge patches seems unsuited in our setting. Felzenszwalb [8] applies Dynamic Programming to find the optimal locations of the vertices of a polygonal model on a regular image grid. Since the computational complexity is quadratic in the number of grid points, it is intractable to have a high resolution grid, which is necessary when the object covers a small portion of the image (while [8] has a 60×60 grid, taking 5 minutes, using a 180×180 grid would be 81 times slower).

In contrast to previous contributions, our method combines the attractive properties of dealing with highly cluttered images, allowing for shape variations and large scale changes, working from a single example, being robust to broken edges, and being computationally efficient.

3 Early processing

Detecting and linking edgel-chains. Edgels are detected by the excellent Berkeley natural boundary detector [15], which was recently successfully applied to object recognition [3]. Next, edgels are chained and a smoothing spline curve is fit to each edgel-chain, providing estimates of the edgels' tangent orientations.

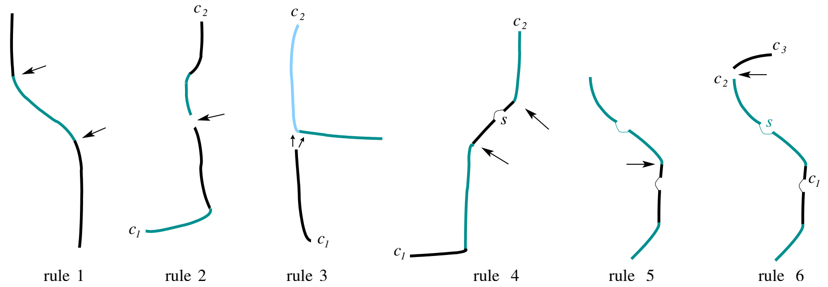

Fig. 2. *The six rules to build the Contour Segment Network. They connect (arrows) regular segments and bridging segments (marked with an arc). Rules 2-6 connect segments over different edgel-chains* c_i.

Due to the well-known brittleness of edge detection, a contour is often broken into several edgel-chains. Besides, the ideal contour might have branchings, which are not captured by simple edgel-chaining. We counter these issues by *linking* edgel-chains: an edgel-chain c_1 is linked to an edgel-chain c_2 if any edgel of c_2 lies within a search area near an endpoint of c_1 (figure 1). The search area is an isosceles trapezium. The minor base rests on the endpoint of c_1, and is perpendicular to the curve's tangent orientation, while the height points away from c_1 [1] . This criterion links c_1 to edgel-chains lying *in front* of one of its endpoints, thereby indicating that it could *continue over* c_2. The trapezium shape expresses that the uncertainty about the continuation of c_1's location grows with the distance from the breakpoint . Note how c_1 can link either to an endpoint of c_2, or to an interior edgel. The latter allows to properly deal with T-junctions, as it records that the curve could continue in two directions (figure 1b). Besides, we point out that it is not necessary for the end of c_1 to be oriented like the bit of c_2 it links to (as in figure 1b). Tangent-discontinuous links are also possible (figure 1c).

The edgel-chain links are the backbone structure on which the Contour Segment Network will be built (section 4).

Contour segments. The elements composing the network are *contour segments*. These are obtained by partitioning each edgel-chain into roughly straight segments. Figure 1d shows the segmentation for a bottle-shaped edgel-chain. In addition to these regular segments, we also construct segments *bridging* over tangent-continuous links between edgel-chains. The idea is to bridge the breaks in the edges, thus recovering useful segments missed due to the breaks.

4 Building the Contour Segment Network

Equipped with edgel-chain links and contour segments, we are ready to build the image representation which lies at the heart of this paper: the Contour Segment Network (or just *network*, for short). To this end, we *connect* segments along edgel-chains, and across links between edgel-chains. Thanks to the explicit modeling of the edgel-chains' interconnections, the network supports robust matching of shapes in cluttered images.

[1] The dimensions of the trapezium are fixed, and the same in all experiments.

Definitions. Before explaining how to build the network, we give a few definitions. First, every segment is *directed*, in that it has a *back* and a *front*. This only serves to differentiate the two endpoints, they have no semantic difference. As a convention, the front of a segment is followed by the back of the next segment on the edgel-chain. Second, every edgel-chain link is directed as well: the edgel-chain c_1, on which the trapezium search-area rests, is at the back, while the other edgel-chain c_2 is at the front. This also defines the front and back endpoints of a segment bridging between two edgel-chains. For clarity, we use the word *links* between edgel-chains, and *connections* between segments.

Rules. The network is built by applying the following rules, illustrated in figure 2. These connect the front of each segment to a set of segments, and its back to another set of segments. Thus the network structure is unconstrained and its complexity adapts to the image content.

1. The front of a segment is connected to the back of the next segment on the same edgel-chain.
2. When two edgel-chains c_1, c_2 are linked at endpoints, the segment of c_1 before the link is connected to the segment of c_2 after the link.
3. Consider a T-junction link (i.e. from an endpoint of c_1 to the interior of c_2). The segment of c_1 before the link is connected to the *two* segments of c_2 with the closest endpoints. As can be seen in figure 2.3, this records that the contour continues in both directions.
4. Let s be a segment bridging over a link from c_1 to c_2. s is connected to the segment of c_2 coming after its front endpoint, *and* to the segment of c_1 coming before its back endpoint.
5. Two bridging segments which have consecutive endpoints on the same edgel-chain are connected. Here 'consecutive' means that no other segment lies inbetween.
6. Consider a bridging segment s without front connection, because it covers the front edgel-chain c_2 until its end. If c_2 is linked to another edgel-chain c_3, then we connect s to the segment of c_3 coming after its front endpoint. An analogue rule applies if s lacks the back connection.

Although they might seem complex at first sight, the above rules are pretty natural. They connect two segments if the edges provide evidence that they could be connected on an ideal edge-map, where all edges would be detected and perfectly chained. Notice how the last three rules, dedicated to bridging segments, create connections analog to those made by the first three rules for regular segments. Therefore, both types are treated consistently.

Since each edgel-chain is typically linked to several others, the rules generate a complex branching structure, a *network* of connected segments. The systematic connections across different edgel-chains, together with the proper integration of bridging segments, make the network robust to incomplete or broken edgel-chains, which are inevitable in real images. Figure 3 shows a segment on a bottle outline, along with all connected segments up to depth 8 (those reachable following up to 8 connections). Although there is no single edgel-chain going all around the bottle, there is a path doing so, by spanning several edgel-chains. It is the task of the forthcoming matching stage to discover such desired paths.

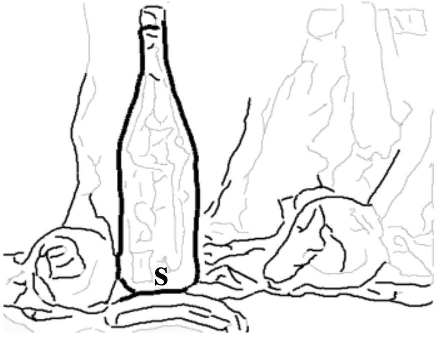

Fig. 3. *Network connectedness. All black segments are connected to S, up to depth 8. They include a path around the bottle (thick).*

5 Basic matching

By processing the test image as described before, we obtain its Contour Segment Network. We also segment the contour chains of the model, giving a set of contour segment chains along the outlines of the object.

The detection problem can now be formulated as finding paths through the network which resemble the model chains. Let's first consider a subproblem, termed *basic matching*: find the path most resembling a model chain, starting from a basis match between a model segment and a test image segment. However we do not know a priori where to start from, as the test image is usually covered by a large majority of clutter segments. Therefore, we apply the basic matching algorithm described in this section, starting from all pairs of model and test segment with roughly similar orientations. The resulting paths are then inspected and integrated into full detection hypotheses in the next section.

We consider the object transformation from the model to the test image to be composed of a global pose change, plus shape variations due to class variability. The pose change is modeled by a translation $\mathbf{t}$ and a scale change σ, while class variability is accommodated by a flexible measure of the similarity between configurations of segments.

The basic matching algorithm. The algorithm starts with a basis match between a model segment b_m and a test segment b_t, and then iteratively matches the other model segments, thereby tracing out a path in the network. The matched path $\mathcal{P}$ initially only contains $\{b_m, b_t\}$.

1. *Compute the scale change σ of the basis match.*
2. *Move to the next model segment m.* Points 3-6 will match it to a test segment.
3. *Define a set $\mathcal{C}$ of candidate test segments.* These are all successors [2] of the current test segment in the network, and their successors (figure 4a). Including successors at depth 2 brings robustness against spurious test segments which might lie along the desired path.

[2] All segments connected at its free endpoint, i.e. opposite the one connecting to $\mathcal{P}$.

4. *Evaluate the candidates.* Each candidate is evaluated according to its orientation similarity to m, how well it fits in the path $\mathcal{P}$ constructed so far, and how strong its edgels are (more details below).
5. *Extend the path.* The best candidate c_{best} is matched to m and $\{m, c_{best}\}$ is added to $\mathcal{P}$.
6. *Update* σ. Re-estimate the scale change over $\mathcal{P}$ (more details below).
7. *Iterate.* The algorithm iterates to point 2, until the end of the model segment chain, or until the path comes to a dead end ($\mathcal{C} = \emptyset$). At this point, the algorithm restarts from the basis match, proceeding in the backward direction, so as to match the model segments lying before the basis one.

For simplicity, the algorithm is presented above as greedy. In our actual implementation, we retain the best two candidates, and then evaluate their possible successors. The candidate with the best sum of its own score and the score of the best successor wins. As the algorithm looks one step ahead before making a choice, it can find better paths.

Evaluate the candidates. Each candidate test segment $c \in \mathcal{C}$ is evaluated by the following cost function [3]

$$q_c = q(m, c, \mathcal{P}) = w_{la} D_{la}(m, c, \mathcal{P}) + w_{ld} D_{ld}(m, c, \mathcal{P}) + w_\theta D_\theta(m, c) \qquad (1)$$

The last term $D_\theta(m, c) \in [0, 1]$ measures the difference in orientation between m and c, normalized by π.

The other terms consider the location of c in the context of test segments matched so far, and compare it to the location of m within the matched model segments. The first such spatial relation is

$$D_{la}(m, c, \mathcal{P}) = \frac{1}{|\mathcal{P}|} \sum_{\{m_i, t_i\} \in \mathcal{P}} D_\theta(\overrightarrow{mm_i}, \overrightarrow{ct_i})$$

the average difference in direction between vectors $\overrightarrow{mm_i}$ going from m's center to the centers of matched model segments m_i, and corresponding vectors $\overrightarrow{ct_i}$ going from c to the matched test segments t_i (see figure 4d). The second relation is analogous, but focuses on the distances between segments

$$D_{ld}(m, c, \mathcal{P}) = \frac{1}{\sigma d_m |\mathcal{P}|} \sum_{\{m_i, t_i\} \in \mathcal{P}} \left| \sigma \|\overrightarrow{mm_i}\| - \|\overrightarrow{ct_i}\| \right|$$

where d_m is the diagonal of the model's bounding-box, and hence σd_m is a normalization factor adapted to the current scale change estimate σ. Thus, all three terms of function (1) are scale invariant.

The proposed cost function grows smoothly as the model transformation departs from a pure pose change. In particular the D_{la} term captures the structure of the spatial arrangements, while still allowing for considerable shape variation. Function (1) is low when c is located and oriented in a similar way as m, in the context of the rest of the shape matched so far. Hence, it guides the algorithm towards a path of test segments with an overall shape similar to the model.

[3] In all experiments, the weights are $w_{la} = 0.7, w_{ld} = 0.15, w_\theta = 1 - w_{la} - w_{ld} = 0.15$.

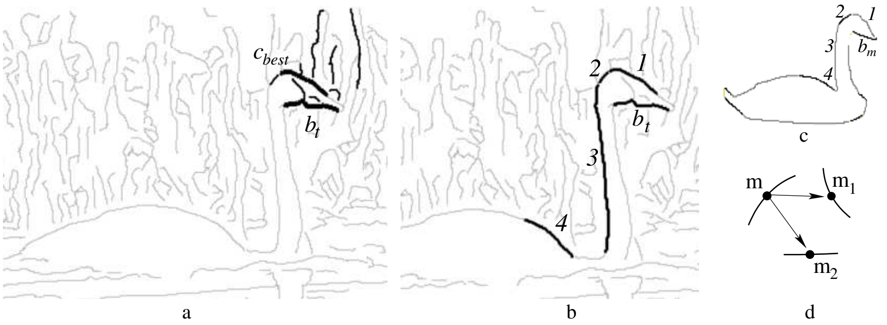

Fig. 4. *Basic matching. (a) Iteration 1: basis segment b_t, candidates $\mathcal{C}$ with $q_c \leq 0.3$ (black thin), and best candidate c_{best} (thick). (b) Matched path $\mathcal{P}$ after iteration 4. (c) Model, with basis segment b_m and segments matched at iteration 1-4 labeled. (d) Example vectors used in D_{la}, D_{ld}.*

Analyzing the values of q_c over many test cases reveals that for most correct candidates $q_c < 0.15$. In order to prevent the algorithm from deviating over a grossly incorrect path when no plausible candidate is available, we discard all candidates with q_c above the loose threshold $q_{th} = 0.3$. Hence: $\mathcal{C} \leftarrow \{c | q_c \leq q_{th}\}$.

In addition to the geometric quality q_c of a retained candidate c, we also consider its *relevance*, in terms of the average strength of its edgels $\bigtriangledown_c \in [0, 1]$. Hence, we set the overall cost of c to $q_c \cdot (1 - \bigtriangledown_c)$. Experiments show a marked improvement over treating edgels as binary features, when consistently exploiting edge strength here and in the path evaluation score (next section).

Update σ. After extending $\mathcal{P}$ the scale change σ is re-estimated as follows. Let δ_m be the average distance between pairs of edgels along the model segments, and δ_t be the corresponding distance for the test segments. Then, set $\sigma = \frac{\delta_t}{\delta_m}$. This estimation considers the relative locations of the segments, together with their individual transformations, and is robust to mismatched segments within a correct path (unlike simpler measures such as deriving σ from the bounding-box areas). Thanks to this step, σ is continuously adapted to the growing path of segments, which is useful for computing D_{ld} when matching segments distant from the basis match. Due to shape variability and detection inaccuracies, the scale change induced by a single segment holds only locally.

Properties. The basic matching algorithm has several attractive properties, due to operating on the Contour Segment Network. First and foremost, at every iteration it must chose among only a few candidates (about 4 on average), because only segments *connected* to the previous one are considered. Since it meets only few distractors, it is likely to make the right choices and thus find the object even in substantially cluttered images. The systematic exploitation of connectedness is the key driving force of our system. It keeps the average number of candidates D low, and *independent* of the total number of test segments T. As another consequence, the computational complexity for processing all basis matches is $O(TMD \log^2(M))$, with M the number of model segments. In contrast to "local search" [4] and "interpretation trees" [11], this is *linear* in T, making it possible to process images with a very large number of clutter segments (even thousands). Second, the spatial relations used in D_{la}, D_{ld} can easily be pre-computed for all possible segment pairs. During basic matching, evaluating a candidate takes but a few operations, making the whole algorithm computationally efficient. In our Matlab implementation, it takes only 10 seconds on average to process the approximately 1000 basis matches occurring when matching a model to a typical test image. Third, thanks to the careful construction of the network, there is no need for the object contour to be fully or cleanly detected. Instead, it can be interrupted at several points, short parts can be missing, and it can be intertwined with clutter contours.

6 Hypothesis integration

Basic matching produces a large set $\mathcal{H} = \{\mathcal{P}_i\}$ of matched paths $\mathcal{P}_i$, termed *hypotheses*. Since there are several correct basis matches to start from along the object contour, there are typically several correct hypotheses on an object instance (figure 5b+c+d). In this section we group hypotheses likely to belong to the same object instance, and fuse them in a single *integrated hypothesis*. This brings two important advantages. First, hypotheses matching different parts of the same model contour chain, are combined into a single, more complete contour. The same holds for hypotheses covering different model chains, which would otherwise remain disjoint (figure 5d). Second, the presence of (partially) repeated hypotheses is a valuable indication of their correctness (i.e. that they cover an object instance and not clutter). Since the basic matcher prefers the correct path over others, it produces similar hypotheses when starting from different points along a correct path (figure 5b+c). Clutter paths instead, grow much more randomly. Hence, hypothesis integration can accumulate the evidence brought by overlapping hypotheses, thereby separating them better from clutter.

Before proceeding with the hypothesis integration stage, we evaluate the quality of each hypothesis $\mathcal{P} \in \mathcal{H}$. Each segment match $\{m, t\} \in \mathcal{P}$ is evaluated with respect to the others using function (1): $q\left(m, t, \mathcal{P} \backslash \{m, t\}\right)$. Whereas during basic matching only segments matched *before* were available as reference, here we evaluate $\{m, t\}$ in the context of the entire path. The score of $\{m, t\}$ is now naturally defined by setting the maximum value q_{th} of q as roof: $q_{th} - q\left(m, t, \mathcal{P} \backslash \{m, t\}\right)$. Finally, the total score of $\mathcal{P}$ is the sum over the component matches' scores, weighed by their relevance (edgel strength $\bigtriangledown$)

$$\phi(\mathcal{P}) = \frac{1}{q_{th}} \sum_{\{m,t\} \in \mathcal{P}} \bigtriangledown_t \cdot \left(q_{th} - q\left(m, t, \mathcal{P} \backslash \{m, t\}\right)\right)$$

the normalization by $\frac{1}{q_{th}}$ makes ϕ range in $[0, |\mathcal{P}|]$. In order to reduce noise and speedup further processing, we discard obvious garbage hypotheses, scoring below a low threshold $\phi_{th} = 1.5$: $\mathcal{H} \leftarrow \{\mathcal{P} | \phi(\mathcal{P}) \geq \phi_{th}\}$.
Hypothesis integration consists of the following two phases:
Grouping phase.

1. Let $\mathcal{A}$ be a graph with nodes the hypotheses $\mathcal{H}$, and arcs $(\mathcal{P}_i, \mathcal{P}_j)$ weighed by the (in-)compatibility c_{sim} between the pose transformations of $\mathcal{P}_i, \mathcal{P}_j$: $c_{sim}(\mathcal{P}_i, \mathcal{P}_j) = \frac{1}{2}(c(\mathcal{P}_i, \mathcal{P}_j) + c(\mathcal{P}_j, \mathcal{P}_i))$, with

$$c(\mathcal{P}_i, \mathcal{P}_j) = \frac{|\mathbf{t_i} - \mathbf{t_j}|}{d_m \sigma_i} \cdot \max\left(\frac{\sigma_i}{\sigma_j}, \frac{\sigma_j}{\sigma_i}\right)$$

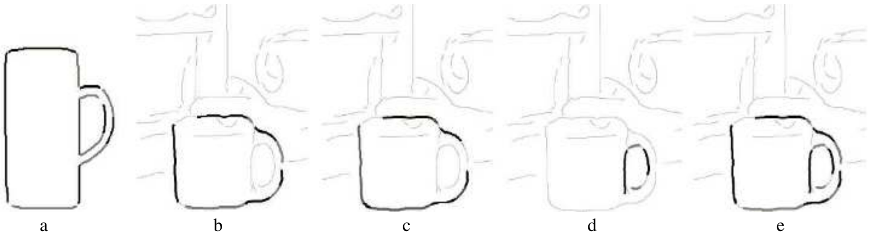

Fig. 5. *Hypothesis integration. a) mug model, composed of an outer and an inner chain (hole). b-d) 3 out of 14 hypotheses in a group. b) and c) are very similar, and arise from two different basis matches along the outer model chain. Instead, d) covers the mug's hole. e) All 14 hypothesis are fused into a complete integrated hypothesis. Thanks to evidence accumulation, its score (28.6) is much higher than that of individual hypotheses (b scores 2.8). Note the important variations of the mug's shape w.r.t the model.*

The first factor measures the translation mismatch, normalized by the scale change σ, while the second factor accounts for the scale mismatch.

2. Partition $\mathcal{A}$ using the Clique Partitioning algorithm proposed by [9]. Each resulting group contains hypotheses with similar pose transformations. The crux is that a group contains either hypotheses likely to belong to the same object instance, or some clutter hypotheses. Mixed groups are rare.

Integration phase. We now combine the hypotheses within each group $\mathcal{G} \subset \mathcal{A}$ into a single integrated hypothesis.

1. Let the *central hypothesis* $\mathcal{P}_c$ of $\mathcal{G}$ be the one maximizing
$$\phi(P_i) \cdot \left(\sum_{\mathcal{P}_j \in \{\mathcal{G} \setminus \mathcal{P}_i\}} \left| \mathcal{P}_i \bigcap \mathcal{P}_j \right| \cdot \phi(\mathcal{P}_j) \right)$$
where $|\mathcal{P}_i \bigcap \mathcal{P}_j|$ is the number of segment matches present in both $\mathcal{P}_i$ and $\mathcal{P}_j$. The central hypothesis best combines the features of having a good score and being similar to the others. Hence, it is the best representative of the group. Note how the selection of $\mathcal{P}_c$ is stable w.r.t. fluctuations of the scores, and robust to clutter hypotheses which occasionally slip into a correct group.
2. Initialize the integrated hypothesis as $\mathcal{G}_{int} = \mathcal{P}_c$, and add the hypothesis $\mathcal{B}$ resulting in the highest combined score $\phi(\mathcal{G}_{int})$. This means adding the parts of $\mathcal{B}$ that match model segments unexplained by $\mathcal{G}_{int}$ (figure 5d, with initial $\mathcal{G}_{int}$ in 5b). Iteratively add hypotheses until $\phi(\mathcal{G}_{int})$ increases no further.
3. Score the integrated hypothesis by taking into account repetitions within the group, so as to accumulate the evidence for its correctness. $\phi(\mathcal{G}_{int})$ is updated by multiplying the component matches' scores by the number of times they are repeated. Evidence accumulation raises the scores of correct integrated hypotheses, thus improving their separation from false-positives.

In addition to assembling partial hypotheses into complete contours and accumulating evidence, the hypothesis integration stage also enables the detection of multiple object instances in the same test image (delivered as separate integrated hypotheses). Moreover, the computational cost is low (1-2 seconds on average).

The integrated hypotheses $\mathcal{G}_{int}$ are the final output of the system (called *detections*). In case of multiple detections on the same image location, we keep only the one with the highest score.

7 Results and conclusions

We present results on detecting five diverse object classes (bottles, swans, mugs, giraffes, apple logos) over 255 test images [4] covering several kinds of scenes. In total, the objects appear 289 times, as some images contain multiple instances. As all images are collected from *Google Images* and *Flickr*, they are taken under varying, uncontrolled conditions. While most are photographs, some paintings, drawings, and computer renderings are included as well. The target objects appear over a wide range of scales. Between the smallest and the largest detected swan there is a scale factor of 4, while for the apple logos class, there is a factor of 6. The system is given only a single hand-drawn example of each class (figure 7, i2-j3), and its parameters are always kept fixed.

Figures 6 and 7 show example detections. In many test cases the object is successfully and accurately localized in spite of extensive clutter, and even when it comprises only a small portion of the image (e.g. b1, b3, e1, h2). The dominant presence of clutter edges is illustrated in a2, b2, c2, with the edge-maps for cases a1, b3, c3. The object contours form only a small minority of all image edges (about 1/30). The capacity of handling large scale changes is demonstrated in d1 and e1, where the mug sizes differ by a scale factor of 3. Moreover, the individual shapes of the detected objects vary considerably, and differ from the models, hence showing the system's tolerance to class variations. Compare d3 and e2 to the bottle model, or the variations among different mugs. In d1 we overlay the model after applying the best possible translation and scale.

Five of the six mugs imaged in figure c1 are found by the system, proving its ability to detect multiple object instances in the same image. As examples of the accuracy of our method, figures d2 and g2 display the image contours matched to the object for cases d1, d3, and g1 (the other cases are reported as the bounding-boxes of the matched contours).

We quantitatively assess performance as the number of correct detections (bounding-box on an instance of the target object class) and false positives (other detections). All five models have been matched to all 255 test images. The thick curves on plots i2-j3 depict the percentage of correct detections (detection-rate) versus the incidence of false-positives (number of false-positives per image FPPI). The system performs well on all five classes, and achieves a remarkable 82$\%$ average detection rate at the moderate rate of 0.4 FPPI. For a baseline comparison, we processed the dataset also with a simple Chamfer Matching algorithm [5] . The model is shifted over the image at several scales, and the local maxima of the Chamfer distance give detection hypotheses. In case of multiple overlapping hypotheses, only the strongest one is retained. As the plots show

[4] The dataset is available on our website: www.vision.ee.ethz.ch/∼ferrari

[5] While this does not include multiple orientation planes, we believe they would improve performance only moderately [13].

(thin curves) the Chamfer Matcher performs markedly worse than our approach, and reaches an average detection-rate of only 39% at 0.4 FPPI. As also pointed out by [13], the reason is that the Chamfer distance is about as low on clutter edgels areas as it is on the target object, resulting in many false-positives hardly distinguishable from correct detections. The problem is particularly outspoken in our setting, where only a single template shape is given [17]. Our approach instead, is much more distinctive and thus brings satisfactory performance even in these highly cluttered images.

In conclusion, the experiments confirm the power of the presented approach in dealing with extensive clutter, large scale changes, and intra-class shape variability, while taking only a single hand-drawn example as input. Moreover, it is robust to discontinuous edges, and is computationally efficient (the complexity is linear in the number of image segments). As one limitation, models cannot self-cross or branch, therefore excluding some objects (e.g. chairs, text). Nevertheless, every object with a distinctive silhouette can be modeled by a set of disjoint outlines, even if a detailed drawing would feature crossing/branchings (e.g. most animals, tools, and logos). Future work aims at addressing this issue, as well as learning the class variability from a few examples, to apply it for constraining the matching.

References

1. R. Basri, L. Costa, D. Geiger, D. Jacobs, *Determining the Similarity of Deformable Shapes*, Vision Research, 1998.
2. S. Belongie, J. Malik, J. Puzicha, *Shape Matching and Object Recognition Using Shape Contexts*, PAMI, 24:4, 2002.
3. A. Berg, T. Berg and J. Malik, *Shape Matching and Object Recognition using Low Distortion Correspondence*, CVPR, 2005.
4. J. R. Beveridge and E. M. Riseman, *How Easy is Matching 2D Line Models Using Local Search?*, PAMI, 19:6, 1997
5. A. Del Bimbo, P. Pala, *Visual Image Retrieval by Elastic Matching of User Sketches*, PAMI, 19:2, 1997.
6. T. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, *Active Shape Models - Their Training and Application*, CVIU, 61:1, 1995.
7. D. Cremers, C. Schnorr, and J. Weickert, *Diffusion-Snakes: Combining Statistical Shape Knowledge and Image Information in a Variational Framework*, Workshop on Variational and Levelset Methods, 2001.
8. P. F. Felzenszwalb, *Representation and Detection of Deformable Shapes*, CVPR'03
9. V. Ferrari, T. Tuytelaars, and L. Van Gool, *Real-time Affine Region Tracking and Coplanar Grouping*, CVPR, 2001.
10. D. Gavrila, V. Philomin, *Real-time Object Detection for Smart Vehicles*, ICCV'99
11. W. Grimson, T. Lozano-Perez, *Localizing Overlapping Parts by Searching the Interpretation Tree*, PAMI, 9:4, 1987.
12. D. Jacobs, *Robust and Efficient Detection of Convex Groups*, PAMI, 18:1, 1996.
13. B. Leibe, B. Schiele, *Pedestrian detection in crowded scenes*, CVPR, 2005.
14. D. Lowe, T. Binford, *Perceptual organization as a basis for visual recognition*, AAAI, 1983
15. D. Martin, C. Fowlkes and J. Malik, *Learning to detect natural image boundaries using local brightness, color, and texture cues*, PAMI, 26(5):530-549, 2004.
16. A. Selinger, R. Nelson, *A Cubist approach to Object Recognition*, ICCV, 1998.
17. A. Thayananthan, B. Stenger, P. Torr, R. Cipolla, *Shape Context and Chamfer Matching in Cluttered Scenes*, CVPR, 2003.

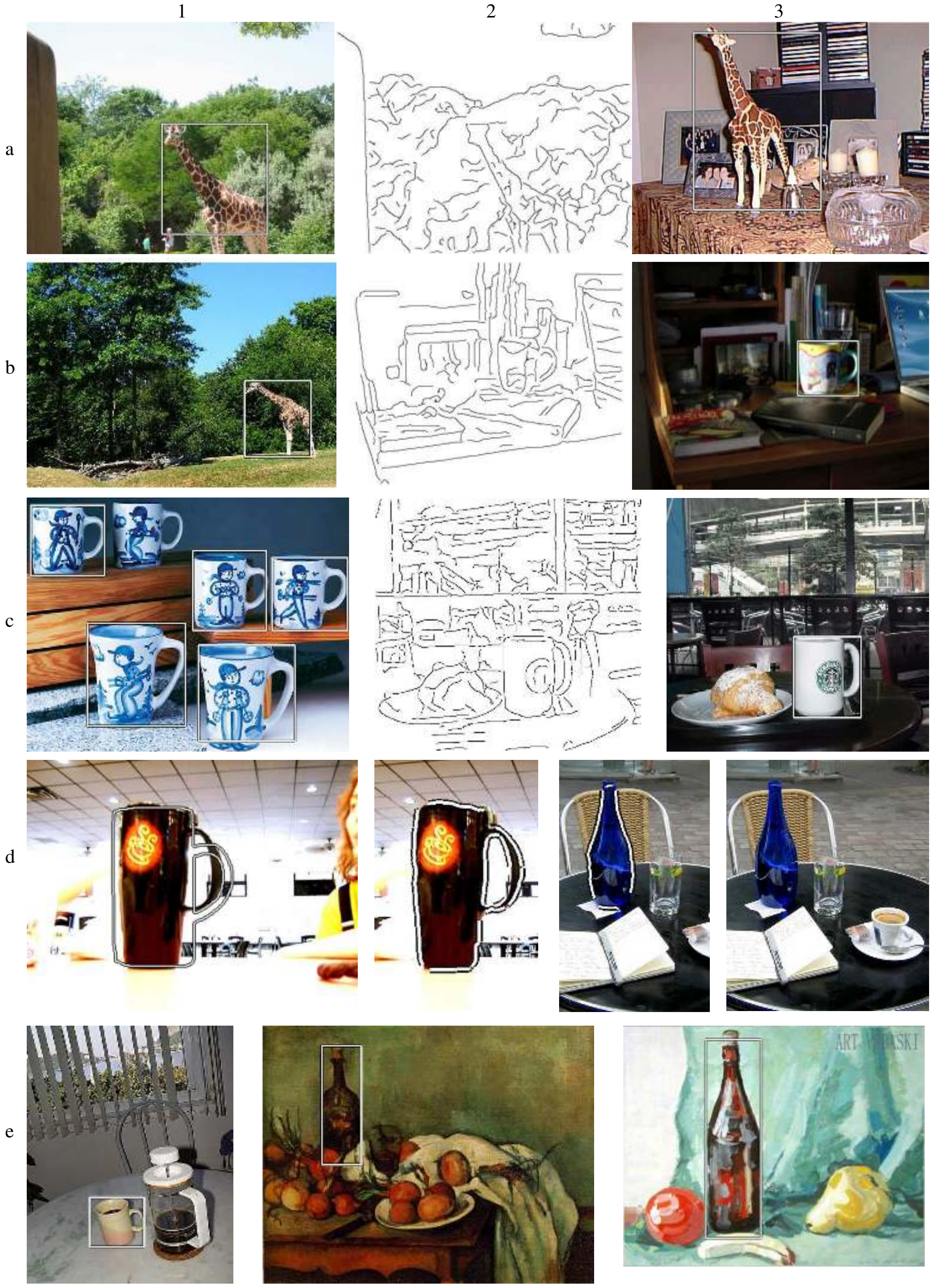

Fig. 6. *Results (first page). See text for discussion.*

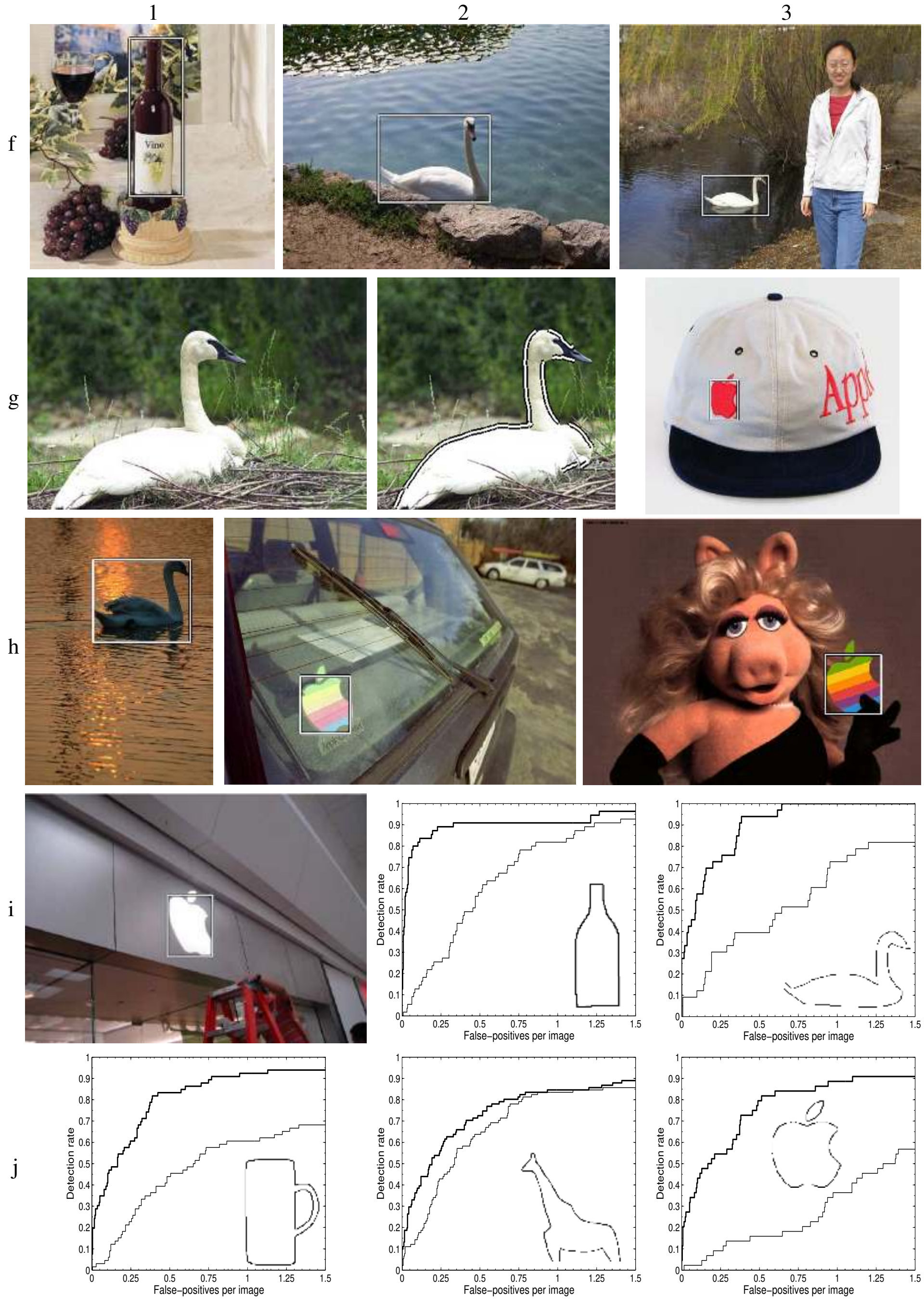

Fig. 7. *Results (second page). See text for discussion.*