The Whole World in Your Hand: Active and Interactive Segmentation

Artur Arsenio* **Paul Fitzpatrick*** **Charles Kemp*** **Giorgio Metta****
*MIT AI Lab
Cambridge, Massachusetts, USA
**Lira Lab, DIST, Uiversity of Genova
Genova, Italy

Abstract

This paper presents three approaches to object segmentation, a fundamental problem in computer vision. Each approach is aided by the presence of a hand or arm in the proximity of the object to be segmented. The first approach is suitable for a robotic system, where the robot can use its arm to evoke object motion. The second method operates on a wearable system, viewing the world from a human's perspective, with instrumentation to help detect and segment objects that are held in the wearer's hand. The third method operates when observing a human teacher, locating periodic motion (finger/arm wagging or tapping) and using it as a seed for segmentation. We show that object segmentation is a key resource for development. We demonstrate that, once high-quality object segmentation is available, it is possible to train up both high-level visual modules (object recognition and localization) and to enhance low-level vision (orientation detection).

1. Introduction

The presence of a body changes the nature of perception. The body is a source of constraint on interpretation, opportunities for experimentation, and a medium for communication. Hands in particular are very revealing, since they interact directly and flexibly with objects. In this paper, we demonstrate several methods for simplifying visual processing by being attentive to hands, either of humans or robots.

Segmentation is an enabler of lots of stuff.

There are two platforms.

Protocol-based.

2. Motivation

We are highly motivated.

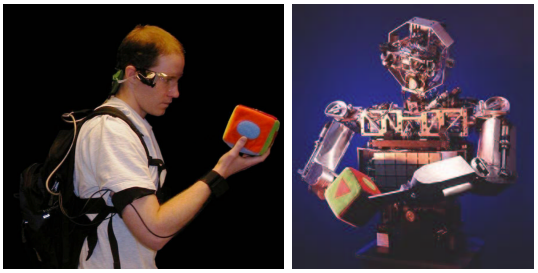

Figure 1: The platforms.

3. Segmentation on a robot

A robot (Cog) equipped with an arm and an active vision head was given a simple "poking" behavior, whereby it selected objects in its environment and struck them (3). Since the robot had a limited reach, this activity required the cooperation of a human companion to bring the robot interesting objects to poke. The behavior could also be preempted by the companion; when the robot fixated an object and was about to reach for it, the companion could choose to poke the object instead, in which case the robot would refrain from acting.

This choice of activity has many benefits. *(i)* The motion signature generated by the impact of the arm with a rigid object greatly simplifies segmenting that object from its background, and obtaining a reasonable estimate of its boundary (see Figure 2). This "active segmentation" procedure is key to automatically acquiring training data of sufficient quality to support the many forms of learning described in the remainder of this paper. *(ii)* The poking activity also leads to object-specific consequences, since different objects respond to poking in different ways. For example, a toy car will tend to roll forward, while a bottle will roll along its side. *(iii)* The basic operation involved, striking objects, can be performed by either the robot or its human companion, creating a controlled point of comparison between robot and human action.

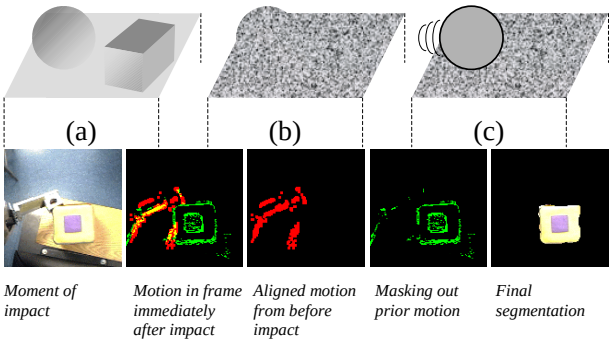

Figure 2: Cartoon motivation (top) for active segmentation (bottom). Human vision is excellent at figure/ground separation (top left), but machine vision is not (top center). Coherent motion is a powerful cue (top right) and the robot can invoke it by simply reaching out and poking around. The lower row of images show the processing steps involved. The moment of impact between the robot arm and an object, if it occurs, is easily detected – and then the total motion after contact, when compared to the motion before contact and grouped using a minimum cut approach, gives a very good indication of the object boundary. Active segmentation. The robot arm is deliberately driven to collide with an object. The apparent motion after contact, when masked by the motion before contact, identifies a seed foreground (object) region. Such motion will generally contain fragments of the arm and environmental motion that escaped masking. Motion present before contact is used to identify background (non-object) regions. An optimal object region is computed from the foreground and background information using graph cuts (1).

4. Charlie's segmentation

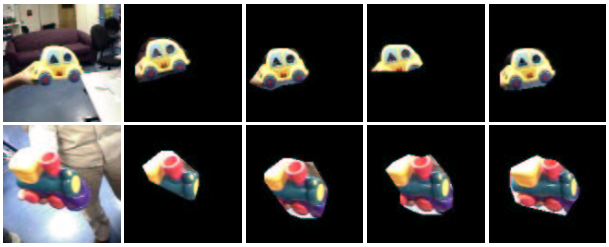

Figure 3: artur 1

5. Artur's segmentation

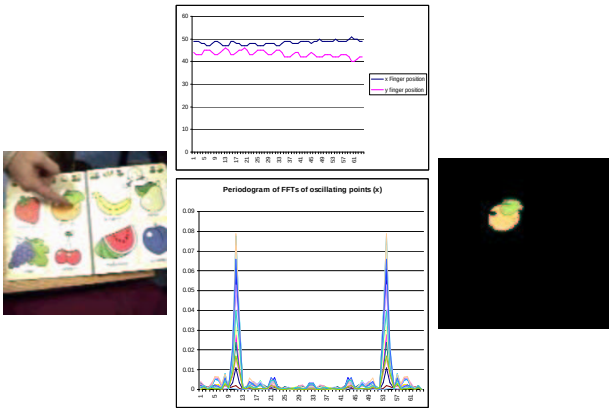

Figure 4: artur 2

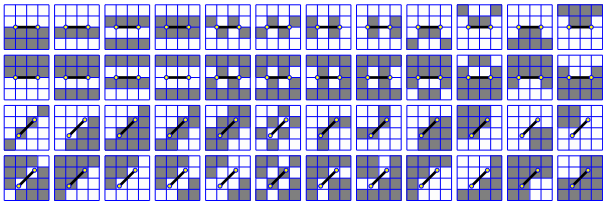

Figure 5: Edges have diverse appearances. This figure shows the orientations assigned to a test suite prepared by hand. Each 4×4 grid is a single test edge patch, and the dark line centered in the grid is the orientation that patch was observed to have in the training data. The oriented features represented include edges, thin lines, thick lines, zig-zags, corners etc. It is difficult to imagine a set of conventional filters that could respond correctly to the full range of features seen here – all of which appeared multiple times in object boundaries in real images.

6. Learning about orientation

Orientation is an important visual cue for many purposes, such as object segmentation, recognition, and tracking. It is associated with neighborhoods rather than individual points in an image, and so is inherently scale dependent. At very fine scales, relatively few pixels are available from which to judge orientation. Lines and edges at such scales are extremely pixelated and rough. Orientation filters derived from analytic considerations, with parameters chosen assuming smooth, ideal straight lines or edges (for example, (2)) are more suited to larger neighborhoods with more redundant information. For fine scales, an empirical approach seems more promising, particularly given that when the number of pixels involved is low, it is practical to sample the space of all possible appearances of these pixels quite densely.

The data collected during segmentation allows us to explore how edges truly look in "natural" images, by simply building up a catalog of edge fragments seen around the boundaries of objects. Of course, such a catalog is only practical at small scales. We worked with 4×4 pixel windows, a size is chosen to be large enough to be interesting, but small enough for the complete range of possible appearances to be easily visualized. Even at this scale, manual data collection and labelling would be extremely tedious, so it is definitely advantageous to have a robotic system to automatically compile and label a database of the appearance of oriented features. These features were extracted by sampling image patches along object boundaries, which were in turn determined using active segmentation. The resulting catalog of edge appearances proved remarkably diverse, although the most frequent appearances were indeed the "ideal" straight, noise-free edge.

7. Learning to recognize objects

With any of the active segmentation behaviors introduced here, the system can familiarize itself with the appearance of nearby objects. This section is concerned with learning to locate and recognize those objects whenever they are present, even when the special cues used for active segmentation are not available. The method described here has been applied to the poking method.

7.1 Approaches to object recognition

Physical objects vary greatly in shape and composition. This variety is reflected in their visual appearance. Unfortunately, it is not a straightforward matter to recover object properties from appearance, since there are many other factors at work – illumination, relative pose, distance, occlusions, and so on. The central challenge of object recognition is to be sensitive to the identity of objects while maintaining invariance in the face of other incidental properties. There are at least two broad approaches to recognition, geometry-based and appearance-based.

Geometry-based recognition

Image formation is a geometric process, so one way to approach recognition is to model invariant geometric relationships that hold for a particular class of object. These relationships can be between points, lines, surfaces or volumes. They may be known for many possible views of the object, or just one. When a new scene is presented, geometric relationships in it are measured and matched against the model. There are many details in what to measure and how to do the matching (there is a good review in (**?**)). The main difficulty is the combinatorics of the search involved. There are a lot of free parameters to search over when we try to match an unsegmented scene to an object model – in particular, which elements in the scene correspond to the object, and what the transformation is between those elements and the object. For high-speed performance, geometric hashing is a useful technique (for a review see (**?**)). In this method, geometric invariants (or quasi-invariants) are computed from

points in model (training) images, then stored in hash tables. Recognition then simply involves accessing and counting the contents of hash buckets. One possibility for the geometric invariants is to take a set of points selected by an interest operator, and use each pair of points in turn to normalize the remainder by scale and rotation. The position of the normalized points can be stored in a 2D hash table, as shown in Figure **??**. Invariants in 3D are more difficult to achieve, but various solutions have been proposed.

Appearance-based recognition

While the world is geometric in nature, geometric features are not particularly easy to extract reliable from images. In appearance-based recognition, the focus is shifted from the intrinsic nature of an object to properties that can be measured in images of that object, including geometric properties but also surface properties such as color or texture. For example, (**?**) proposed using the set of colors present in segmented views of an object as their representation. Regions of an image that contain the same color mix (as determined by histogram intersection) could contain the object. Histogram back-projection can be used to quickly filter out regions unlikely to contain the object, by assigning each pixel a weight based on the frequency of its color in the histogram. Some form of region-growing method then accumulates this evidence to find plausibly colored regions. This method is very fast, and for objects with distinctive colors it can be useful as a prefilter to more computationally expensive processing. Color is also intrinsically somewhat robust to scale and rotation, but is sensitive to changes in the spectral profile of illumination.

Many appearance-based methods are window-based. A classifier is built that operates on a rectangular region of interest within an image. That window is moved across the entire image, at multiple scales, and sometimes multiple orientations. Responses of the classifier across locations and scales are combined using various heuristics. Variation in orientation (rotation in depth) is typically dealt with by training up multiple recognizers for various poses. These poses can be sampled quite sparsely, but still each pose requires iteration of the search procedure. Ther are ways to speed all this up, for example using a cascade of classifiers that reject clearly non-target windows early, devoting full analysis only to plausible targets. The actual classifier itself can be based on eigenspace methods or many other possibilities.

Probabilistic methods like Schiele. Version with Roy.

7.2 Hashing with rich features

The approach used here is like geometric hashing, but uses richer features that include non-geometric information. Geometric hashing works because pairs of points are much more informative than single points. An ordered pair of points defines a relative scale, translation, and orientation (in 2D). Further points may be needed for more general transformations, such as affine or projective, or to move to 3D (**?**,). But, even staying with the 2D case, using just two points is somewhat problematic. First of all, they are not at all distinctive – any two points in an image could match any two points in the model. Hashing doesn't require distinctiveness, but it would be a useful prefilter. Secondly, there is no redundancy; any noise in the points will be directly reflected in the transformation they imply.

One possibility would be to use triplets of points, or more. Then we have distinctiveness and some redundancy. But we also have an explosion in the number of possible combinations. Pairs of points are just about manageable, and even then it is better if they are drawn from a constrained subset of the image. For example, the work of (**?**) uses histograms of the distance and angles between pairs of points on the boundary of an object for recognition.

In this work, pairs of edges (or more generally, any region with well-defined and coherent orientation) are used instead of pairs of points (Figure **??**). Pairs of edges are more distinctive than pairs of points, since they have relative orientation and size. And if used carefully during matching, they contain redundant information about the transformation between image and model. A disadvantage is that edges are subject to occlusion, and edges/regions found automatically many be incomplete or broken into segments. But in all but the most trivial objects, there are many pairs of edges, so this approach is at least not doomed from the start.

The orientation filter developed earlier is applied to images, and a simple region growing algorithm divides the image into sets of contiguous pixels with coherent orientation. For

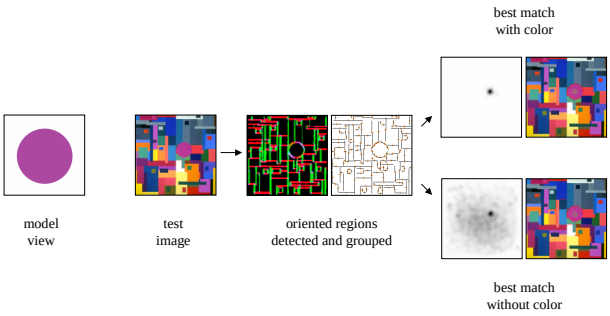

Figure 6: A simple example of object localization: finding the circle in a Mondrian.

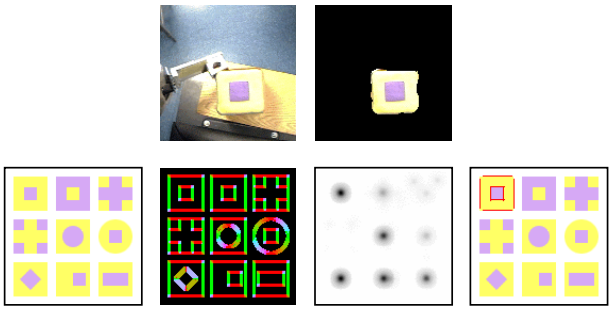

Figure 7: Looking for the best match to a cube found through poking in a synthetic image of various approximations and distractors. The superimposed red lines on the rightmost image indicate the detected position of the object and the edges implicated. The image on its immediate left shows the strength of evidence for the object across the entire image, which lets us rank the distractors in order of attractiveness.

realtime operation, adaptive thresholding on the minimum size of such regions is applied, so that the number of regions is bounded, independent of scene complexity. In "model" (training) views, every pair of regions belonging to the object is considered exhaustively, and entered into a hash table, indexed by relative angle, relative position, and the color at sample points between the regions (if inside the object boundary).

A useful extension of geometric hashing is coherency, where each match implies a particular transformation, and only "coherent" matches are aggregated (for example, see (**?**)). This could be applied in the present instance. For speed, an abbreviated version is used here, where we filter by the centroid location each match implies for the object (this is information we would like to have anyway). There is no coherence checking by scale and orientation at the matching stage. This procedure means that we perform object localization simultaneously with matching.

Oriented regions are relatively sparse. Experiments showed that on a fast machine (800MHz) and at low resolution (128×128) it is possible to use triplets of regions as features at close to real-time. These can be very distinctive, and very redundant, and non-trivial objects have very very many possible triplets. But the frame-rate was just too slow (approximately 1 Hz) to be worth using here.

At the other extreme, another possibility would be just to use single edges. But they are not very distinctive, and sampling colors at an edge (generally a high-variance area) is problematic.

8. Searching for a synthetic object in a synthetic scene

As a simple example of how this all works, consider the test case shown in Figure **??**. The system is presented with a model view of the circle, and the test image. For simplicity, the model view in this case is a centered view of the object by itself, so no segmentation is required. The processing on the model and test image is the same – first the orientation filter is applied, and then regions of coherent orientation are detected. For the circle, these regions will be small fragments around its perimeter. For the straight edges in the test image, these will be long. So finding the circle reduces to locating a region where there are edge fragments at diverse angles to each other, and with the distance between them generally large with respect to their own size. Even without using color, this is quite sufficient for a good localization in this case. The perimeter of the circle can be estimated by looking at the edges that contribute to the peak in match strength. The algorithm works equally well on an image of many circles with one square.

8.1 Searching for real objects in synthetic scenes

In Figure 7, we take a single instance of an object found through poking, and search for it in a synthetic image containing an abstract version of it along with various distractors. The algorithm picks out the best match, and lets us rand the distractors in order of salience. It is clear that a yellow square with anything in it is a good match, and having the internal purple square adds another boost. The closest distractor is a yellow square with a purple square inside it, rotated by 45°.

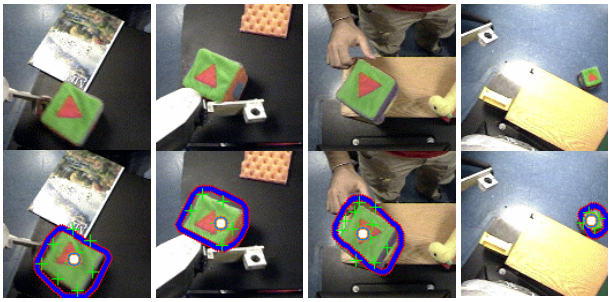

Figure 8: The cube being recognized, localized, and segmented in real images. The image in the first column is one the system was trained on. The image in the remain columns are test images. Note the scale invariance demonstrated in the final image.

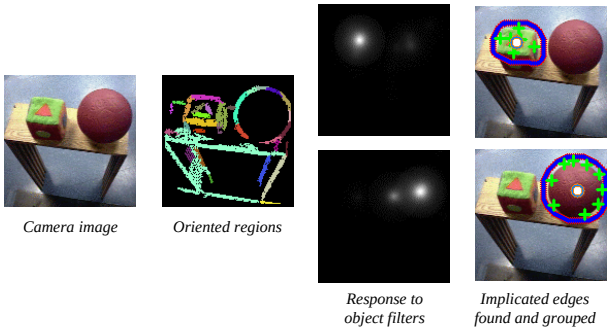

Figure 9: resegment multiple

8.2 Recognizing real objects in real images

Figure 8 shows examples of the cube being resegmented in real images.

Testing on a set of 400 images of four objects (about 100 each) being poked by the robot, with half the images used for training, and half for testing, gives a recognition error rate of about 2%, with a median localization error of 4.2 pixels in a 128×128 image (as determined by comparing with the center of the segmented region given from poking). By segmenting the image by grouping the regions implicated in locating object, and filling in, a median of 83.5% of the object is recovered, and 14.5% of the background is mistakenly included (again, determined by comparison with the results of poking).

9. Multiple objects simultaneously

See Figure 9. Don't actually do this normally, deal with foveation and egomap (described later).

9.1 Online training

Finally, Figure 10 shows recognition and training occurring online.

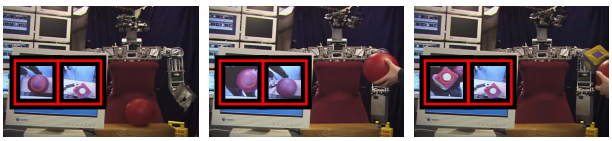

Figure 10: This figure shows stills from a short interaction with Cog. The area of the first frame highlighted with a square shows the state of the robot – the left box gives the view from the robot's camera, the right shows an image it associates with the current view. If the object confuses another object with what it has already seen, such as the ball in the first frame, this is easily to fix by poking. IMPROVE THIS DESCRIPTION

10. Discussion and conclusions

Acknowledgements

Funds for this project were provided by DARPA as part of the "Natural Tasking of Robots Based on Human Interaction Cues" project under contract number DABT 63-00-C-10102, and by the Nippon Telegraph and Telephone Corporation as part of the NTT/MIT Collaboration Agreement.

References

Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 359–374, 2001.

J. Chen, Y. Sato, and S. Tamura. Orientation space filtering for multiple orientation line segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(5):417–429, May 2000.

P. Fitpatrick and G. Metta. Towards manipulation-driven vision. In *IEEE/RSJ Conference on Intelligent Robots and Systems*, 2002.

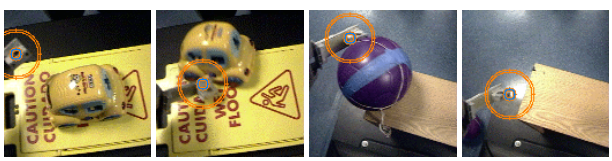

Figure 11: This doesn't really need to be in the paper.

P. Fitzpatrick. First contact: Segmenting unfamiliar objects by poking them. 2003. submitted to IROS.

B. Schiele and J. L. Crowley. Recognition without correspondence using multidimensional receptive field histograms. *International Journal of Computer Vision*, 36(1):31–50, January 2000.