Taking in Shape: Detection and Tracking of Basic 3D Shapes in a Robotics Context

Richtsfeld Andreas, Mörwald Thomas, Zillich Michael, and Vincze Markus

Automation and Control Institute (ACIN), Vienna University of Technology
{arichtsfeld, moerwald, zillich, vincze} @acin.tuwien.ac.at

Abstract Segmenting and tracking generic objects in an unknown dynamic scene remains an elusive goal for computer vision. In this paper we tackle a simplified problem, namely detecting and tracking objects from a class of basic shapes (cuboids, cylinders, cones, spheres) in scenes containing a ground plane. We use perceptual grouping of edges to identify generic views of basic shapes, instantiate 3D wire-frame models assuming that objects initially rest on the ground and subsequently track these using a particle-filter based tracker. During tracking we augment the wire-frame models with surface texture which substantially increases the robustness of tracking with respect to background clutter as well as occlusions and lighting effects.

1 Introduction

Perceiving 3D shape plays a major role in computer vision applications for robotics tasks. Most obviously this is true for tasks involving manipulation of objects, be it grasping, simple pushing or just observing a human handling an object. While tracking based on 3D models of objects has a long history and many increasingly sophisticated methods exist, robustness and speed remain a challenge to this date. Also the question of how sufficiently accurate object models are acquired in the first place is often left open.

In this work we present a system that aims to close this gap between model acquisition from an unknown scene and robust tracking of models. We make several simplifying assumptions. Having in mind robotics applications we assume a robot mounted camera which allows us to at least assume the ground plane (or tabletop plane) while the rest of the scene remains of course unknown. Furthermore we restrict our search to objects belonging to simple shape classes (cuboids, cylinders, cones and spheres) which are detected in generic views from edge images. Note that there are in principle no limits to shape complexity for the tracker once we have a model (in fact more “non-regular” shapes simplify the task for tracking) but that the difficulty of segmenting objects from edge images limits the scope of shapes we can handle.

Fig. 1 shows how we proceed from 2D images to tracked 3D objects.

The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 explains detection of shapes while their tracking is detailed in Section 4. Section 5 shows experimental results and Section 6 concludes the paper.

2 Related Work

Detection of geometric shapes based on perceptual grouping of edges is a well known topic in computer vision with an abundance of literature since the eighties. [1] introduced geons as a means of describing objects of arbitrary shape by components and [3] discuss the potential of geons in computer vision systems. [20] present a method based on joint statistical constraints defined on complex wavelet transforms to represent and detect geons. These representations are however not size invariant and a sliding window is needed to cover the
whole image at different positions and scales. Moreover the test images are rather artificial with grey objects on a black background. Note that our proposed approach does not aim to detect all geons but is tailored to a subset.

[9] uses perceptual grouping of edge segments to reduce the complexity of detecting 3D models in edge images and shows impressive results on highly cluttered images. This however requires precise CAD-like 3D models given a-priori which we want to avoid. Approaches such as [13], [6] and [14] use groups of edge fragments to detect learned classes of shapes and show impressive results on databases. Our models differ in that they are only topological models of generic views of basic shapes such as an arrangement of lines and ellipses forming a cylinder, where the limited number of these shapes allows us to ignore learning.

[4] combine qualitative and quantitative object models to detect and track ten object primitives (box, cylinder, tapered cylinder etc). That system is still brittle in the presence of texture and clutter. To this end [15] describe objects beyond simple shapes but still of limited complexity (cups, hats, desks) with qualitative, parts-based shape abstraction base on a vocabulary of 2D part models corresponding essentially to closed contours of various shapes. Their system can extract such representations from images containing textured objects as well as complex backgrounds.

A major challenge in perceptual grouping is the combinatorial explosion when identifying possible groups of image features as the number of possible groups grows exponentially with group size. [9] addresses the problem of exponential run-time complexity using a grid overlaid on the image indexed by line endpoints. A typical problem of indexing is the appropriate choice of bin size. [18] use further curve parameters to construct index spaces of higher-parametric models and also addressed the problem of bin size and indices close to bin boundaries. [22] proposes to use indexing in the image space where search lines emanating from the ends of image edges are used to find collinearities and junctions and finally closed contours. Search lines are grown incrementally over processing time, thus avoiding problematic distance thresholds.

An overview of monocular model-based 3D tracking is given in [8]. With respect to tracking based on Monte Carlo particle filtering the authors state the following:

We attribute this relative lack of popularity to two different causes. First a large number of particles, perhaps as many as several thousands when the motion is poorly defined, can be required, which slows down the tracking process.

To cope with this problem our particle filter based approach uses texture information, which is more distinctive than only using geometry edges and therefore requires a smaller number of particles. Secondly we implemented the major part of the particle filter on the graphics processing unit (GPU) which allows us to evaluate many particles efficiently in parallel.

Tracking based on surface texture is of course not new. [10] also use edges and texture for tracking. They extract point features from surface texture and use them together with edges to calculate object pose. This turns out to perform very fast and robust with respect to occlusion. Our approach not only uses patches around a few interest points but the whole texture. Also [17] fuse interest points (FAST features) from surface texture and geometry edges for improved tracking. Whereas in both of the above approaches texture and edges are treated explicitly with different underlying operators our approach treats them the same using one underlying operator, namely edge extraction.

Recent approaches typically rely on (possibly combinations of) edge contours, interest point descriptors and optical flow [2, 19, 21]. While interest point descriptors (such as SIFT, FAST etc.) and optical flow are becoming faster to compute thanks in part to GPU implementations edges are still considerably faster to compute but are of course inherently less discriminative than interest point descriptors.

The work presented in this paper is based on [7] where the authors also take advantage of the GPU by projecting a wire-frame model into the camera image. A particle filter with a Gaussian noise model is used to evaluate the confidence level with respect to the pose.

Our approach not only uses geometry edges but also edge features from textures which extends the class of trackable models to those that have curved surfaces as illustrated on the right of Fig. 2. This is because in a standard 3D model curvature is approximated by triangles and quadrangles which would produce virtual edges which do not correspond to the actual edges as shown on the left of Fig. 2.

We found [12] to be the closest related work to our tracking part, but instead of computing the cross-correlation of a pixel patch we propose to evaluate the match between the edge gradients of the rendered model and the camera image. This means that we have fewer comparisons for each pixel which makes it faster. Furthermore edges are more robust against changing lighting conditions. Our approach was partially pre-
sented in [11], with major modifications regarding the particle filtering.

3 Detection of Basic Shapes

In the following sections we show how we detect generic views of basic shapes in edge images by employing an incremental perceptual grouping approach. Having detected 2D shapes we then use a ground plane assumption to generate 3D shape models which are subsequently handed to the tracker.

3.1 Incremental Indexing and Anytimeness

Perceptual grouping in our system is based on the work of [22] which provides an anytime solution to finding junctions between edge segments and subsequently closed contours avoiding the need for arbitrary distance thresholds and [16] which adds higher level features. Indexing is used to efficiently identify candidates for junctions, where the indexing space is the image itself. Each edge endpoint defines a set of search lines consisting of tangential and normal search lines. These search lines are drawn into the index image using Bresenham line drawing. Whenever two lines index into the same bin, i.e., their search lines intersect, we create a new junction. Depending on the types of search lines intersecting we form an L-junction, a collinearity or a T-junction between the respective originating lines. If more than two lines intersect, the according number of pairwise junctions are created. Shortest path search in the resulting graph consisting of edges and junctions then finds closed contours.

In order to avoid the definition of certain length thresholds for search lines they are drawn incrementally, continuously checking for junctions. So the longer we search, the more junctions and eventually closed contours will be found, where “easy” cases typically pop out fast and “difficult” ones (broken edges, partial occlusions, more clutter) follow later. This allows us to stop processing anytime, e.g., after a certain frame time has elapsed or, if we happen to know that we expect precisely three cylinders in the scene, after having found three cylinders.

3.2 Perceptual Grouping

We then define a hierarchy of grouping principles to enable efficient abstraction of image edges into basic geometric Gestalts as shown in Fig. 3. Triggered by the incrementally growing search lines referred to in the above section, lower level Gestalts such as closures or ellipses are formed and in turn trigger formation of higher level Gestalts such as cuboids and cylinders. Concretely cuboids are defined as three overlapping “flaps”, where a flap is defined as two rectangles connected along one edge. Ellipses are derived from convex groups of intersecting arcs. Cylinders are defined using ellipse junctions: search lines emanating from the major semi-axes of an ellipse meeting straight lines. Two ellipses and two parallel straight lines (possibly of course comprised itself of several collinear lines) thus make up a cylinder. Cones are defined likewise.

![Figure 3: Abstraction hierarchy: from edges to basic shapes](image)

Note that as we move up the abstraction hierarchy the corresponding Gestalts get more and more distinctive. So while we will generally find lots of closures, rectangles and ellipses are already somewhat less frequent. Finally cuboids comprised of three flaps or cylinders and cones being composed of a specific topological arrangement of lines and ellipses already rarely appear accidentally. The next section explains how we further reduce the number of false hypotheses.

3.3 Ranking and Masking

With longer processing time the number of estimated hypotheses will grow exponentially, because the more junctions are found between edges the more combinatorial possibilities for higher level groups will appear. Basically this means we will start accumulating “crappy” hypotheses. However many of these higher level hypotheses will share lower level features, thus essentially providing different interpretations for the same underlying data. We are obviously only interested in the best interpretation.

So we rank all hypotheses according to their significance, which is derived from properties such as parallelism, closeness or completeness of Gestalt. We then traverse the list of hypotheses in order of decreasing significance and mark all visited lower level features. Whenever a hypothesis finds one of its constituent features already marked, it will be masked by the higher ranked owner of that feature.

This additional pruning step gets rid of most false hypotheses. Remaining accidental groupings that happen to constitute valid hypotheses could be identified by changing the viewpoint or more generally by observation over time as typically only correct hypotheses, i.e., actual shapes will be stable over viewpoints. In the context of this paper the tracker described in Section 4 takes care of these as only correct hypotheses will produce stable tracks.

3.4 From 2D to 3D

The following tracking procedure in Section 4 requires a 3D wire-frame model of the detected object shape...
as well as an initial pose estimate relative to the camera. Note that everything so far is purely 2D, i.e. we detect projections of shapes in generic views onto the image plane. Assuming a camera with known elevation and tilt angle and further assuming that detected objects (cubes, cones, cylinders and spheres) rest on the ground, allows us to convert them to 3D shapes. We intersect view rays with the ground plane and thus obtain 3D position on the plane as well as unambiguous size. Note that we restricted our search to objects belonging to a limited number of simple shape classes. This allows us to “fill in” the unseen backside from simple symmetry considerations.

4 Tracking

Once we have detected objects and generated wire-frame models along with initial pose estimates we initialize a model based 3D pose tracker. Remember that we view our work in the context of robotic applications where we expect to observe manipulated objects or manipulate them ourselves. So it is essential that we have robust and fast estimates of object trajectories.

To improve robustness we enhance the wire-frame models with surface texture, captured directly from the camera image. Tracking based solely on wire-frame edges already provides reasonable performance in many cases but is insufficient for rotationally symmetric objects and runs into problems with degenerate views and heavy background clutter. Adding texture edges provides much more dense information for the tracker to “snap” on to, allowing degenerate views as well as large scale partial occlusions.

To meet real-time requirements we harness the parallel computing power of modern graphics processing units (GPU) for image processing. Graphics boards are designed to render virtual scenes as realistically as possible. The basic idea is to compare those virtual scenes with an image captured from reality. Texturing is a common method of simulating realistic surfaces. In this paper, the edges of those textures are used for comparison. Fast progress in computer graphics will soon allow the inclusion of more and more optical effects such as shadows, reflections, shading, occlusions or even smoke, fire, water or fog. This requires of course available in-

4.1 Image Processing

In the following an object is described by the geometry of its surface $S$ (approximated by polygons and vertices $v$) and its 6 DOF pose $x$. Furthermore with the

\[ x_k = \sum_{i=1}^{N} w_{k-1}^i x_{k-1}^i \]  \hspace{1cm} (1)
the edge texture once as the edges appear for a "representative" view of the object and using that for rendering of all other (similar) views substantially reduces such effects. Of course this relies on the assumption that the various object pose hypotheses represented by the particles are in fact similar enough, which for normal tracking situations they are.

4.2 Particle Filtering

For each tracking step the particle filter executes the methods shown in Fig. 4. First the particles $x^i_0, i = 1, \ldots, N$, representing the pose of the object, are generated using Gaussian noise. Then the confidence level $c^i_k$ and importance weight $w^i_k$ of each particle $x^i$ are evaluated by matching its corresponding edge image against the edge image of the camera $I^e_C$. According to the importance weights the set of particles is resampled and importance weights the set of particles is resampled and “Resampling with Replacement” is executed several times (2-5 times depending on the power of the processor) before proceeding to the step “Weighted Mean”. We refer to this as Iterative Particle Filtering. This increases accuracy, and therefore also increases robustness especially when the object is moving.

First particles $x^i_1$ are resampled from the prior particle distribution $x^i_{k-1}$ according to the importance weights. Then $x^i_1$ is perturbed by adding Gaussian noise $n(\sigma)$ with a standard deviation scaled by the prior confidence level $c^i_{k-1}$:

$$x^i_k = x^i_1 + n(\sigma)$$

$i = 1, \ldots, N$

The standard deviation is evaluated by

$$\sigma = \sigma(c^i_{k-1}, m_w)$$

$m_w$ is the transformation from the normalized Gaussian noise in the range of $[0, \ldots, 1]$, to the metric world coordinates with respect to the object size.

Each particle is tested against the camera image and a confidence level is calculated. To this end the correlation between the gradients of the edges $g_S(u, v)$ and $g_C(u, v)$ is evaluated by comparing the direction of the edges at each image point $(u, v)$.

$$g_S(u, v) = \left( I^e_{S,x}(u, v) \right)$$

$$g_C(u, v) = \left( I^e_{C,x}(u, v) \right)$$

The angles between those vectors are calculated, producing the edge correlation image $\Phi$:

$$\phi = \arccos(g_S \cdot g_C)$$

$$\Phi^i(u, v) = \begin{cases} 
1 - \frac{2\theta}{\pi} & \text{if } \phi < \pi/2 \\
1 - \frac{2(\pi - \phi)}{\pi} & \text{if } \phi > \pi/2 \\
0 & \text{if } (u, v) \notin \mathbf{v}'_S
\end{cases}$$

Note that it is assumed that the result of the arccos() function lies within 0 and $\pi$. The image $\Phi^i$ now contains the degree of correlation between the pose suggested by the particle $i$ and the camera image. The angular deviation of the edge angles $\Phi^i$ is scaled to the range of 0 to 1.

The confidence level $c^i$ and importance weight $w^i$ are evaluated as follows:

$$c^i_k = \frac{1}{2} \left( \frac{m^i_n}{n^i} + \frac{m^i}{n_{\max}} \right) \quad (2)$$

$$w^i_k = (c^i_k)^\rho$$

with

$$m^i = \sum_{u,v} \Phi^i(u, v)$$

$$n^i = \sum_{u,v} |I^e_S(u, v)|$$

$$n_{\max} \propto \max(n^i|\pi = 1, \ldots, N)$$

The first term within the brackets of equation (2) is the percentage of matching edge pixels $m^i$ with respect to the non-matching edge pixels $n^i$. Calculating the confidence level only with this term would cause the tracker to lock when only one side of the 3D object is visible. If in this degenerate view the object is rotated slightly, another side of the object becomes visible. The particle representing this rotation would typically get less weight than the prior front facing particle. This is because the number of matching pixels $m^i$ grows slower than the number of not-matching pixels $n^i$ when rotating the object out of the front side view. This effect is amplified by the fact that edge detection for strongly tilted faces is less accurate.
The second term allocates more weight to the total number of matching pixels $m^i$ which is intrinsically higher for the rotated particle. $n_{\text{max}}$ is the maximum number of visible edge pixels in the actual area and scales the pixels to the proper range. To scale the range of the outcome to $[0, \ldots, 1]$ the terms are divided by the sum of their maximum values.

The weights of the particles are calculated by raising $c_i^k$ to the power of $p$, which controls the speed of convergence of the particles. With a higher power $p$, $w_i^k$ increases, which leads to more particles assigned to $x_i^k$ when resampling the whole particle distribution and therefore to a faster convergence.

As explained in Section 4.1 for projection and re-projection of the model, a single pose $\tilde{x}^k$ is required. We use the weighted mean of the particle distribution as in equation (1), because it shows good results, both in terms of accuracy and smoothness of the resulting pose.

5 Experimental Results and Evaluation

We made experiments and evaluations with the proposed basic shape detector and with the model tracker. The detector and the tracker are running as separated threads in a distributed framework, using shared memory for the exchange of geometry and pose information of objects. While the detector is triggered only every second to detect major changes in the scene, the tracker runs with high priority at frame rate to achieve high pose accuracy. For each new object appearing in the scene, the detector drops a new geometric model to the shared memory. On the other hand the tracker updates the pose of the models in the shared memory, which allows the system to identify a re-detection.

5.1 Detection of Basic Shapes

The incremental grouping method has been evaluated experimentally with a mobile robot moving among simple geometric 3D objects. Fig. 6 shows an example image, with several objects. The picture indicates a typical problem of grouping, namely that shadows or image noise create spurious features such as lines or arcs. A grouping into higher level Gestalts sometimes accidentally includes a wrong feature. With the incremental approach object detection depends on processing time. The first two images present the cluttered edge image and the extended search lines after 468 ms from the voting image. The following images show the detected object shapes after 328 ms and 468 ms processing time.

The proposed method is able to detect non-textured as well as textured objects in real world scenes. As can be expected, the amount of texture, background clutter and occlusion limits detection rates. Gaps in edges due to occlusion are filled in by search lines if they are not too large (otherwise filling in would take rather long). Adjoining faces with the same color tend to lead to weak edges if lighting is uniform, leading to edge detection failure. Also too much texture over shape edges (e.g. for colourful packaging) will cause problems for detection of the geometry edges. Fig. 7 shows results from two indoor table-top scenes.

To evaluate the capabilities of the perceptual grouping approach, we explored a playground scene containing several cubes with a mobile robot using different processing times. Tab. 1 shows the results of the different runs using an Intel Core2Duo with 2.5 GHz. The whole scene consists of 148 images within 417 cubes to detect. Our approach allows only detection of basic shapes in generic views. With degenerated views, when only one or two surfaces of a cube are visible, only a detection of rectangles or flaps is possible, which may indicate a cube at this position but is not sufficient to build a cube hypothesis. Therefore it is impossible to reach a perfect detection rate of 100 percent with this method of grouping.

Fig. 8 shows the detection rate graphically from the playground scene with the different processing times. As expected the detection rate increases with increased processing time but also the rate of falsely detected cubes. Under 400 ms the false positives (FP) are few but will increase steadily with increasing processing time. I.e. the longer we search the more “hallucinated” shapes will appear.

Please note that the first 150 ms of processing are
used up by Canny edge detection (we use Deriche edge extraction and self-adjusting hysteresis thresholds) and line and arc fitting. Only after that fixed amount of time the incremental processing elements take place and start filling the hierarchy of Gestalts. Hence the detection rate curve starts at 150 ms.

Table 1: True and false positive and negative (TP, FP, TN, FN) detection from a playground scene with four cubes in 148 images at different processing times.

<table>
<thead>
<tr>
<th>Processing time</th>
<th>TP</th>
<th>FP</th>
<th>TN</th>
<th>FN</th>
</tr>
</thead>
<tbody>
<tr>
<td>150 ms</td>
<td>211</td>
<td>0</td>
<td>-</td>
<td>206</td>
</tr>
<tr>
<td>200 ms</td>
<td>294</td>
<td>0</td>
<td>-</td>
<td>123</td>
</tr>
<tr>
<td>300 ms</td>
<td>340</td>
<td>1</td>
<td>-</td>
<td>77</td>
</tr>
<tr>
<td>400 ms</td>
<td>359</td>
<td>12</td>
<td>-</td>
<td>58</td>
</tr>
<tr>
<td>600 ms</td>
<td>378</td>
<td>28</td>
<td>-</td>
<td>39</td>
</tr>
</tbody>
</table>

5.2 Tracking of Textured Objects

The advantage of tracking using the edges of the object texture is its high robustness against changing lighting conditions, occlusion, reflections and background clutter. In other words, they are very distinctive, robust and only a relatively small portion of the surface needs to be visible to determine the correct pose. Of course the latter is only true if the visible surface is rich in texture.

Robustness comes with at the cost of speed, which corresponds to the number of particles used. Fig. 10 illustrates the dependency between frame rate and number of particles, where the red shaded area indicates either too low frame rate (bottom) or too few particles (left) which causes jittering and loss of object. We experienced that running the particle filtering loop several times within one image but with less particles further increases robustness without wasting calculation time.

6 Conclusion and Further Work

We presented a system for detection and tracking of basic geometric objects. Shape detection is based on a hierarchical perceptual grouping system where the use of an incremental processing approach eliminates the need for many thresholds and parameters in various Gestalt principles. Assuming known camera elevation and tilt we use the ground plane to generate 3D wire-frame models which are subsequently covered with surface texture and tracked by a particle filter based tracker.

Tracking converges quickly to the correct pose and is able to handle large deviations, for example when initializing. Also partial occlusion, reflections, light changes, shadows and cluttered background are handled thanks to the increase in robustness with the use of texture edges besides simple wire-frame edges. Exploiting the power of a graphics processing unit allows high tracking speed at frame rate.

We are currently working to remove the ground plane
Table 2: Frame rate with respect to number of polygons of the geometrical model with different number of recursions and particles, computed on a GeForce 285 GTX

<table>
<thead>
<tr>
<th>Example Objects</th>
<th>Faces</th>
<th>Frames per Second</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>2x250</td>
</tr>
<tr>
<td>Box</td>
<td>6</td>
<td>240</td>
</tr>
<tr>
<td>Cylinder (low)</td>
<td>24</td>
<td>220</td>
</tr>
<tr>
<td>Cylinder (mid)</td>
<td>96</td>
<td>210</td>
</tr>
<tr>
<td>Cylinder (high)</td>
<td>384</td>
<td>190</td>
</tr>
<tr>
<td>Complex Scene</td>
<td>1536</td>
<td>160</td>
</tr>
</tbody>
</table>

assumption required to get from 2D to 3D shape by using stereo, where we perform stereo matching not on pixel but on higher feature level. These are generally much more distinctive than small image patches used in dense stereo and matching of edges allows high accuracy in the stereo reconstruction.

Tracking rate is limited by the number of particles required to obtain sufficiently accurate pose. The number of particles can be reduced by employing a better motion model (right now we simply assume a 0-th order motion model, i.e. a static object). A first order motion model could already improve tracking during smooth trajectories. But actually it is the non-smooth trajectories that pose the real problems and here first order motion models would not help. So we plan to incorporate predictive models from a physics simulation to predict events such as sudden stops when a falling object hits the ground.

Another important issue is the ability of the tracker to report lost tracks, which would result in a re-detection request to the detector. Deciding when a tracked object is truly lost and not just partially occluded or entering strong shadow is non-trivial. To this end we are working on robust tracking confidence measures.

Acknowledgement

The research leading to these results has received funding from the European Communitys Seventh Framework Programme [FP7/2007-2013] under grant agreement No. 215181, CogX.

References