EU FP7 CogX
ICT-215181
May 1 2008 (52months)

DR 2.1: Representations of 3D shape for manipulation

Michael Zillich, Markus Vincze, Thomas Mörwald, Andreas Richtsfeld, Kai Zhou, Yasemin Bekiroglu, Marek Kopicki

TUW, Vienna
⟨zillich@acin.tuwien.ac.at⟩

Due date of deliverable:	July 31 2009
Actual submission date:	July 27 2009
Lead partner:	TUW
Revision:	final
Dissemination level:	PU

Vision is one of the primary sensory modalities for a mobile robot that is to interact with its environment. Interaction comes in two flavors: grasping objects for fetch-and-carry tasks and manipulating objects to learn object affordances, i. e. the relationship between shape, applied action and object behaviour. Both require the knowledge of 3D object shape and 6D pose. That knowledge is typically imprecise and incomplete. So the 3D shape representation and 3D vision methods must not only be robust in the face of real-world noise, clutter and occlusion as encountered in mobile robotics scenarios, but also be able to formalise their incompleteness and suggest actions to fill these knowledge gaps.

1 Tasks, objectives, results **6**
1.1 Planned work 6
1.2 Actual work performed 6
1.3 Relation to the state-of-the-art 7

2 Annexes **10**
2.1 Mörwald et al. “Edge Tracking of Textured Objects with a Recursive Particle Filter” 10
2.2 Richtsfeld et el. “3D Shape Detection for Mobile Robot Learning” 11
2.3 Richtsfeld et el. “Basic Object Shape Detection and Tracking Using Perceptual Organization” 12
2.4 Bekiroglu ”Elementary Grasping Actions for Grasping Polyflaps” 13
2.5 Kopicki et al. “Prediction learning in robotic pushing manipulation” 14

References **15**

Executive Summary

This report presents work carried out in WP2 on representation of 3D shape for manipulation, on detection and tracking of 3D shape and work on using 3D shape for manipulation, exemplified by preliminary results in planning grasping actions and learning object affordances.

There are numerous ways to represent 3D shape: Depth maps, 3D point clouds, mesh models, spin images, sparse sets of 3D features and parametric models such as superquadrics. Each has its benefits and limitations in terms of expressiveness, compactness and detectability. Manipulation of 3D objects poses several requirements on 3D shape representations.

The representation should be able to express a wide array of artificial and natural shapes. It should allow a compact description that abstracts away from unnecessary detail. It needs to be dense, i. e. provide shape information everywhere on an object in order to support things like collision detection. And it should link to several, possibly complementary vision methods that allow detection of shapes in cluttered environments. Another practical requirement is the ability to link to existing tools such as robot simulators, e. g. to support development based on learning from simulations.

Another important aspect for a system that detects gaps in its knowledge and actively self-extends is that of incomplete models. So the 3D shape representation should also be able to represent incomplete models (such as objects seen only from one side) and be able to suggest actions to supply the missing information.

We report results on reliable detection of basic shapes (such as boxes and polyflaps) [29, 28], robust 3D object tracking [21] and preliminary results on grasp planning based on polyflaps [3] and on learning object affordances from observed actions on objects [16].

The work carried out addresses Task 2.1 (Contour based shape representations) and supports Tasks 2.2 (Early grasping strategies) and 2.5 (Modular motor learning theory).

Role of 3D shape representation in CogX

Manipulation of objects rests on a representation of 3D shape. So 3D shapes are a fundamental type of information passed between components of the system. Figure 1 shows how these various components interact. Essentially 3D objects are detected, tracked and then used for manipulation.

Objects are detected by detecting generic basic shapes [29, 28] and instantiating 3D models. Basic shapes (like boxes, polyflaps - a polygon cut out of a flat material and folded once to produce a 3D object [33]) are detected in 2D edge images and either a ground plane constraint or binocular stereo is used to obtain 3D shapes and 6D poses.

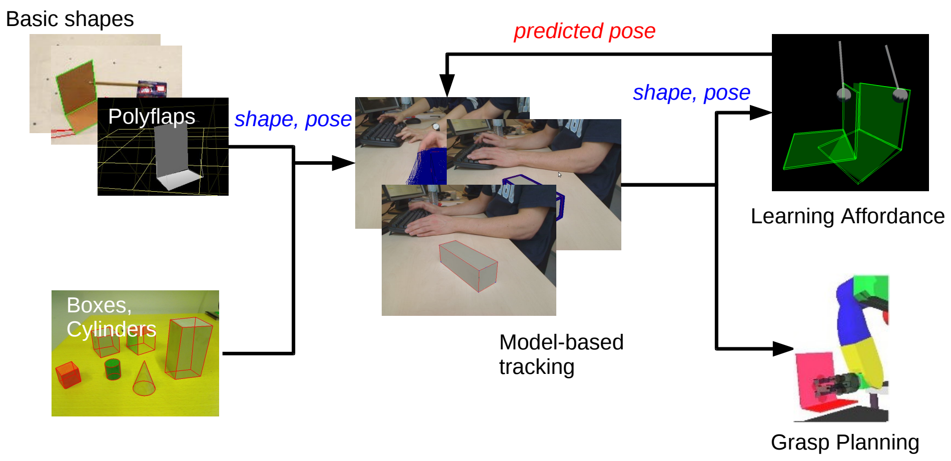


Figure 1: WP2 Overview: blue indicates forward (bottom-up) information, red indicates feedback (top-down) information.

Tracking is based on models obtained from generic basic shapes [21]. It must be sufficiently fast to enable real-time observation of manipulation actions and robust enough to recover from lost tracks due to occlusion or fast motion (a box toppling over will almost certainly be too fast to observe). Tracking is based on a particle filter which maintains a multitude of hypotheses and thus allows recovering from lost tracks. The representation of the 6D pose as a PDF allows incorporation of pose priors, such as the assumption that an object will move primarily in the ground plane. Furthermore predictions of object motion that go beyond a simple first order motion model can be supplied by the affordance learning component. The tracker is implemented to run on a GPU enabling tracking at frame rate.

Learning object affordances requires reliable object poses at constant, high frame rates. Moreover the manipulating arm will start occluding the manipulated object at some point. We need to be able to continue tracking despite possibly large occlusions as well as changing lighting conditions and shadows cast by the moving arm. Grasp planning has less severe requirements, as the manipulated object typically does not move prior to grasping and must only be tracked until a stable grasp could be established.

Once affordances have been learned and the system can make predictions on the behaviour of objects based on its own actions, these predictions can be fed back to improve the performance of the tracker by replacing a simple first order motion model with an accurate learned model.

Contribution to the CogX scenarios and prototypes

We can detect untextured boxes and polyflaps reliably on low complexity background and track them robustly at frame rate. Tracking can handle complex backgrounds, severe lighting changes and large occlusions. This constitutes the first step of the Dexter scenario (learning object affordances).

1 Tasks, objectives, results

1.1 Planned work

Work reported in this deliverable mainly concerns Task 2.1:

> Contour based shape representations. Investigate methods to robustly extract object contours using edge-based perceptual grouping methods. Develop representations of 3D shape based on contours of different views of the object, as seen from different camera positions or obtained by the robot holding and turning the object actively. Investigate how to incorporate learned perceptual primitives and spatial relations from WP5.

Although final results are only to be reported in deliverable DR 2.2, work on manipulation has already started in year 1. These tasks serve as drivers for the development of 3D shape representation and detection. These tasks are Task 2.2 (Early grasping strategies) where the goal is to, based on the visual sensory input extracted in Task 2.1, define motor representations of grasping actions for two- and three-fingered hands. And Task 2.5 (Modular Motor Learning), learning to predict object trajectories and contact relations for pushing and grasping actions.

1.2 Actual work performed

Development of 3D shape representation and detection was driven by the requirements of manipulation methods, namely grasp planning and pushing for learning object affordances. Both require information about surfaces with their normals and surface edges. Grasp points for elementary grasping actions (EGAs) are set on surfaces or edges of surfaces and surface normals are used to define friction cones, which are essential in achieving stable grasps. Learning to predict object behaviour under action requires local surface patches, again with surface normals.

A mesh model consisting of vertices and planar surfaces provides the sort of information required by these manipulation methods. It is a common form of representation in many 3D applications and links well to work in 3D object tracking as well as available tools, such as robot simulators.

It is easy to determine whether a model is incomplete (e. g. missing backside) by checking if the mesh has holes. Now a new view point can be calculated and proposed as a knowledge gathering action in order to see the missing backside. Though at this point by assuming objects are only from a limited class of objects (polyflaps and boxes) we avoid the problem of unknown backsides and always have complete models.

This generic mesh model serves as basis for more advanced models inside the various manipulation methods. Grasp planning describes a shape via elementary grasping actions (EGAs). To this end a mesh model is decomposed into sets of polyflaps, as EGAs at this point are only specified for polyflaps,

Affordance learning represents the effect of actions applied to objects as a mixture of experts, one taking into account global shape information and one taking into account local shape information in the form of local surface patches. These experts represent as probability density functions which object motions are likely/unlikely given an action.

Detection of generic 3D object shape using edge-based perceptual grouping was started with a limited class of shapes (boxes and polyflaps) (see annexes 2.2 and 2.3). Tracking of these shapes can be achieved in real-time, with high robustness to background clutter, varying lighting and large occlusions (see annex 2.1. Building up object representations continuously from different views is currently being investigated, where textures of as yet unseen sides of an object are filled in as soon as they become visible. This is ongoing work and has not been reported yet.

The first grasping strategies are defined by an approach vector (relative pose with respect to object/grasping part) and preshape strategy (grasp type), see annex 2.4. The grasping system is integrated with the real visual input and evaluated in simulation. We are currently performing dynamic simulation to evaluate the interplay between visual representation, applied actions and their effects. This is ongoing work. Further results will be reported as planned in deliverable DR 2.2.

Learning object affordances started with learning behaviour from pushing actions (annex 2.5. This was tested in simulation using a physics simulator. Tests on a physical arm with input from real vision data are the next step and will also be reported in deliverable DR 2.2.

1.3 Relation to the state-of-the-art

Representation of 3D shape has been a topic, explicitly or implicitly through the history of computer vision. Biederman [4] established the theory of Recognition-by-Components (RBC), in which objects are represented as sets of geometric primitives (GEONs). One way to describe such GEONs are superquadrics. [39] and [6] segment 3D point clouds into GEONSs and estimate superquadric parameters. [5] detect given superquadric models in cluttered 3D scenes. These methods however work on 3D data only and require rather good 3D point clouds to start from.

Saxena et al. [32] follow a completely different path. They present an approach to generic grasping without explicitly representing 3D shape at all. They learn the 2D image appearance of good grasp points from labeled training data (many simulated views of a 3D object with a specified grasp point). They then use simple motion stereo to triangulate the 3D location of a grasp point from a series of images taken with an arm-mounted camera. While they can show good results in grasping objects that contain "handle-like" structures (essentially parallel vertical edges) this approach is not general enough. Also avoiding explicit 3D shape representation does not allow checking for collisions and more sophisticated grasp planning.

Johnson [14] introduced spin-images as a representation of 3D shape generated from a 3D triangle mesh. Spin-images are a 3D equivalent of 2D interest point operators and share their benefits: a compact representation that is robust to occlusions. Good results for object detection in cluttered scenes could be shown. This representation however is also sparse in the sense that only a few distinct surface points are described with their spin-images. Checking for collisions however requires 3D shape information to be available on the whole object surface.

We chose to stick to simple mesh models as they are very general, allow modeling objects at various levels of detail and link to many existing methods in detection and tracking. More advanced models like those used in our manipulation methods can typically be easily derived from a mesh model.

A lot of progress has been made in recent years regarding detecting, recognising and tracking of 3D objects (e. g. [11], [23] [26], [20]). Most of these methods derive their speed and robustness from the use of strong interest points and are thus dependent on textured objects. This however implies that these methods are suitable only for handling specifically trained object instances. A system that explores and self-extends must however be able to handle novel, unseen objects.

Supporting learning object affordances requires the ability to detect objects of a class of shapes. E. g. in order to learn how boxes in general behave (depending on width/height ratios, standing or lying on their side) requires the ability to detect box shapes in general. Category-level object recognition (e. g. [10], [25], [35]) is inherently 2D or handles multiple views of a 3D object, each of which however is again 2D internally. To support object manipulation we require full 3D object shape

Recent advances in real-time 3D scanning technology, like time-of-flight cameras [1] or 3D laser range scanners [2] offer interesting possibilities in fast and direct perception of 3D shape. However the latter are still too bulky and expensive for mobile robot applications and the former have their own issues such as low depth resolution and problems with dark areas. Moreover the tenor of CogX is to understand principles of cognition that allow the system to overcome limited and possibly contradictory sensory information, rather than throwing more hardware at the problem (which typically introduces new problems of its own).

So we need vision methods to provide 3D mesh models from 2D images. Edge-based object detection is a well-researched area since the eighties [18].

[1]www.mesa-imaging.ch/prodview4k.php

[2]www.riegl.com/products/mobile-scanning

3D model indexing based on invariant image features [13, 2, 24, 22] is an efficient method to detect models in cluttered images. We need however to detect not only specific object instances but classes of object shapes.

Perceptual grouping approaches [31, 1, 12, 30], also a well-known topic in computer vision, can find more general shape primitives. Our approach is based on an incremental perceptual grouping method [40]. This approach avoids tuning of grouping parameters that plagues many other approaches when faced with real-world images in changing conditions. It further allows anytime-processing, i. e. results are generated as time progresses, with significant results typically popping out early. This allows keeping a fixed frame rate irrespective of image complexity.

Tracking of 3D models has been addressed numerous times in the computer vision and robotics communities. Initial approaches used edge information of CAD-like models [8, 9, 38, 7]. More recent approaches combine edge and texture information [17, 19, 36, 37, 34, 27] to improve robustness to occlusion. Another recent trend is the use of the good parallel processing capabilities of graphics cards (GPUs) [15].

Our tracking approach extends the work of [15] by adding tracking of texture to the model. This greatly increases the number of edges and thus robustness of the tracker without sacrificing real-time performance.

2 Annexes

2.1 Mörwald et al. "Edge Tracking of Textured Objects with a Recursive Particle Filter"

Bibliography T. Mörwald, M. Zillich and M. Vincze: "Edge Tracking of Textured Objects with a Recursive Particle Filter" In: Graphicon 2009 (submitted)

Abstract This paper proposes a new approach of model-based 3D object tracking in real-time. The developed algorithm uses edges as features to track, which are easy and robust to detect. It exploits the functionality of modern highly parallel graphics boards by performing hidden face removal, image processing, texturing and particle filtering. Using a standard 3D model the tracker neither requires memory and time consuming training nor other pre-calculations. In contrast to other approaches this tracker also works on objects where geometry edges are barely visible because of low contrast.

Relation to WP The model based 3D tracker presented in this paper delivers the key input to manipulation methods (grasping and pushing). The employed particle filter framework allows incorporation of context knowledge from bottom-up or top-down attention.

2.2 Richtsfeld et el. “3D Shape Detection for Mobile Robot Learning”

Bibliography A. Richtsfeld and M. Vincze: “3D Shape Detection for Mobile Robot Learning”, Technical Report, ACIN-TR-2009/1, Automation and Control Institute, Vienna University of Technology, Vienna, Austria, June 2009.

Abstract If a robot shall learn from visual data the task is greatly simplified if visual data is abstracted from pixel data into basic shapes or Gestalts. This paper introduces a method of processing images to abstract basic features into higher level Gestalts. Grouping is formulated as incremental problem to avoid grouping parameters and to obtain anytime processing characteristics. The proposed system allows shape detection of 3D such as cubes, cones and cylinders for robot affordance learning.

Relation to WP Detection of basic 3D shapes such as boxes, cylinders and polyflaps is the precursor to object tracking and thus the first step of any manipulation action. The presented anytime perceptual grouping allows real-time operation and furthermore allows incorporation of attentional mechanisms into the grouping process.

2.3 Richtsfeld et el. "Basic Object Shape Detection and Tracking Using Perceptual Organization"

Bibliography A. Richtsfeld and M. Vincze: "Basic Object Shape Detection and Tracking Using Perceptual Organization", Technical Report, ACIN-TR-2009/2, Automation and Control Institute, Vienna University of Technology, Vienna, Austria, June 2009.

Abstract If a robot shall learn object affordances, the task is greatly simplified if visual data is abstracted from pixel data into basic shapes or Gestalts. This paper introduces a method of processing images to abstract basic features and into higher level Gestalts. Perceptual Grouping is formulated as incremental problem to avoid grouping parameters and to obtain anytime processing characteristics. Furthermore we want to present a efficient method to track Gestalts using lowlevel Gestalts for motion field approximation. The proposed system allows shape detection and tracking of 3D shapes such as cubes, cones and cylinders for robot affordance learning.

Relation to WP This paper extends the above by adding tracking of 2D perceptual groups. Note that this is independent of 3D model based tracking and mainly serves to increase robustness of the 2D perceptual grouping process in dynamic scenes.

2.4 Bekiroglu "Elementary Grasping Actions for Grasping Polyflaps"

Bibliography Yasemin Bekiroglu: "Elementary Grasping Actions for Grasping Polyflaps", Technical Report, CVAP/CAS-CSC, KTH, Stockholm, Sweden, July 2009.

Abstract The ability to manipulate novel objects detected in the environment and to predict their behaviour after a certain action is applied to them is important for a robot that can extend its own abilities. The work presented in this report investigates the interplay between perception and action in the framework of object manipulation based on visual input. The work concentrates on the development of generalisable and extensible manipulation strategies for two and three fingered robot hands.

In particular, we investigate the necessary conditions for grasping of objects using basic geometric representations - polyflaps. We demonstrate how polyflaps can be grasped by using a parallel jaw gripper. A set of potential grasps is generated based on an analytical parametrisation. Generated grasps are ordered based on a metric that considers the pose state space of a given polyflap. The output of the method is either the most stable grasping action or the desired pose that can be integrated with a polyflap pushing mechanism to achieve the most stable grasping action.

Relation to WP This report presents the first design and implementation of a polyflap grasping system for two-fingered grasps. It demonstrates the idea of Elementary Grasping Actions (EGAs) and their analytic definition.

2.5 Kopicki et al. “Prediction learning in robotic pushing manipulation”

Bibliography M. Kopicki, J. Wyatt and R. Stolkin “Prediction learning in robotic pushing manipulation”, In: The 14th International Conference on Advanced Robotics (ICAR) 2009, Munich, Germany

Abstract This paper addresses the problem of learning about the interactions of rigid bodies. A probabilistic framework is presented for predicting the motion of one rigid body following contact with another. We describe an algorithm for learning these predictions from observations, which does not make use of physics and is not restricted to domains with particular physics. We demonstrate the method in a scenario where a robot arm applies pushes to objects. The probabilistic nature of the algorithm enables it to generalize from learned examples, to successfully predict the resulting object motion for previously unseen object poses, push directions and new objects with novel shape. We evaluate the method with empirical experiments in a physics simulator.

Relation to WP Learning object behaviour under manipulation and extrapolating to novel objects is a key capability for a self-extending cognitive agent. This work represents a first step in that direction by starting with the simple interaction of pushing.

References

[1] F. Ackermann, A. Maßmann, S. Posch, G. Sagerer, and D. Schlüter. Perceptual grouping of contour segments using markov random fields. *IEEE Transactions on Pattern Recognition and Image Analysis*, 7(1):11–17, 1997.

[2] Jeffrey S. Beis and David G. Lowe. Indexing without invariants in 3d object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10):1000–1015, 1999.

[3] Y. Bekiroglu. Elmentary Grasping Actions for Grasping Polyflaps. TRITA-CSC-CV 2009.4 CVAP315, CVAP/CAS-CSC, KTH, Stockholm, Sweden, July 2009.

[4] I. Biederman. Human image understanding: Recent research and a theory. *CVGIP*, 32(1):29–73, October 1985.

[5] G. Biegelbauer and M. Vincze. Efficient 3D Object Detection by Fitting Superquadrics to Range Image Data for Robot Object Manipulation. In *IEEE Conf. on Robotics and Automation*, pages 1086–1091, 2007.

[6] L. Chevalier, F. Jaillet, and A. Baskurt. Segmentation and superquadric modeling of 3D objects. In *11-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, 2003.

[7] Andrew I. Comport, Eric Marchand, and Francois Chaumette. A real-time tracker for markerless augmented reality. In *In ACM/IEEE Int. Symp. on Mixed and Augmented Reality, ISMAR'03*, pages 36–45, 2003.

[8] Tom Drummond and Roberto Cipolla. Application of lie algebras to visual servoing. *International Journal of Computer Vision*, 37(1):21–41, 2000.

[9] Tom Drummond and Roberto Cipolla. Real-Time Visual Tracking of Complex Structures. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(7):932–946, 2002.

[10] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 264–271, 2003.

[11] I. Gordon and D. G. Lowe. What and where: 3D object recognition with accurate pose. In J. Ponce, M Hebert, C. Schmid, and A. Zisserman, editors, *Toward Category-Level Object Recognition*, pages 67–82. Springer-Verlag, 2006.

[12] Gideon Guy and Gerard Medioni. Inferring global perceptual contours from local features. *International Journal of Computer Vision, Special issue on computer vision research at the University of Southern California*, 20(1-2):113–133, 1996.

[13] David W. Jacobs. Space efficient 3D model indexing. A.I. Memo 1353, Massachusetts Institute of Technology, 1992.

[14] Andrew Johnson. *Spin-Images: A Representation for 3-D Surface Matching*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 1997.

[15] Georg Klein and David Murray. Full-3d edge tracking with a particle filter. In *Proc. british machine vision conference (bmvc'06)*, volume 3, pages 1119–1128, Edinburgh, September 2006. BMVA.

[16] M. Kopicki, J. Wyatt, and R. Stolkin. Prediction learning in robotic pushing manipulation. In *14th International Conference on Advanced Robotics (ICAR), Munich*, 2009.

[17] Ville Kyrki and Danica Kragic. Tracking unobservable rotations by cue integration. In *ICRA'06*, pages 2744–2750, 2006.

[18] David G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31(3):355–395, 1987.

[19] L. Masson, M. Dhome, and F. Jurie. Robust real time tracking of 3D objects. In *Proceedings of the 17th International Conference on Pattern Recognition (ICPR)*, volume 4, pages 252 – 255, 2004.

[20] F. Moreno-Noguer, M. Salzmann, V. Lepetit, and P. Fua. Capturing 3D Stretchable Surfaces from Single Images in Closed Form. In *IEEE Conference on Computer Vision and Pattern Recognigion (CVPR)*, 2009.

[21] T. Mörwald, M. Zillich, and M. Vincze. Edge Tracking of Textured Objects with a Recursive Particle Filter. In *19th International Conference on Computer Graphics and Vision (Graphicon), Moscow (submitted)*, 2009.

[22] Randal C. Nelson and Andrea Selinger. A cubist approach to object recognition. Technical Report TR689, Dept. of Computer Science, Univ. of Rochester, 1998.

[23] S. Obdrzálek and J. Matas. Sub-linear indexing for large scale object recognition. In *Proceedings of the British Machine Vision Conference*, volume 1, pages 1–10, 2005.

[24] Clark F. Olson. Probabilistic indexing: A new method of indexing model data from 2d image data. In *Proceedings of the Second CAD-Based Vision Workshop*, pages 2–8. IEEE, 1994.

[25] A. Opelt, M. Fussenegger, A. Pinz, and P. Auer. Generic object recognition with boosting. *PAMI*, 28(3):416–431, 2006.

[26] Mustafa Ozuysal, Pascal Fua, and Vincent Lepetit. Fast Keypoint Recognition in Ten Lines of Code. In *CVPR'07*, 2007.

[27] Muriel Pressigout and Eric Marchand. Real-time Hybrid Tracking using Edge and Texture Information. *International Journal of Robotics Research*, 26(7):689–713, 2007.

[28] A. Richtsfeld and M. Vincze. 3D Shape Detection for Mobile Robot Learning. Technical Report ACIN-TR-2009/1, Automation and Control Institute, Vienna University of Technology, Vienna, Austria, June 2009.

[29] A. Richtsfeld and M. Vincze. Basic Object Shape Detection and Tracking Using Perceptual Organization. Technical Report ACIN-TR-2009/2, Automation and Control Institute, Vienna University of Technology, Vienna, Austria, June 2009.

[30] Paul L. Rosin and Geoff A. W. West. Extracting surfaces of revolution by perceptual grouping of ellipses. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 677–678, 1991.

[31] Sudeep Sarkar and Kim L. Boyer. A computational structure for preattentive perceptual organization: Graphical enumeration and voting methods. *IEEE Transactions on System, Man and Cybernetics*, 24(2):246–266, February 1994.

[32] Ashutosh Saxena, Justin Driemeyer, Justin Kearns, and Andrew Ng. Robotic Grasping of Novel Objects. *NIPS*, 19, 2006.

[33] A. Sloman. Polyflaps as a domain for perceiving, acting and learning in a 3-D world. In *AAAI Fellows Symposium, Menlo Park, CA*, 2006.

[34] Geoffrey Taylor and Lindsay Kleeman. Fusion of multimodal visual cues for modelbased object tracking. In *In australasian conference on robotics and automation (acra2003), brisbane,australia*, 2003.

[35] A. Thomas, V. Ferrari, B. Leibe, T. Tuytelaars, B. Schiele, and L. Van Gool. Towards Multi-View Object Class Detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[36] L. Vacchetti, V. Lepetit, and P. Fua. Stable Real-Time 3D Tracking using Online and Offline Information. *PAMI*, 2004.

[37] Luca Vacchetti, Vincent Lepetit, and Pascal Fua. Combining Edge and Texture Information for Real-Time Accurate 3D Camera Tracking. In *ISMAR'04*, 2004.

[38] M. Vincze, Ayromlou M., W. Ponweiser, and Zillich M. Edge Projected Integration of Image and Model Cues for Robust Model-Based Object Tracking. *International Journal of Robotics Research*, 20(7):533–552, 2001.

[39] Kenong Wu and Martin D. Levine. Segmenting 3d objects into geons. In *in ICIAP*, pages 321–334. Springer-Verlag, 1995.

[40] Michael Zillich and Markus Vincze. Anytimeness Avoids Parameters in Detecting Closed Convex Polygons. In *The Sixth IEEE Computer Society Workshop on Perceptual Grouping in Computer Vision (POCV 2008)*, 2008.

Edge Tracking of Textured Objects with a Recursive Particle Filter

Thomas Mörwald[*] Vienna University of Technology

Michael Zillich[†] Vienna University of Technology

Markus Vincze[‡] Vienna University of Technology

Abstract

This paper proposes a new approach of model-based 3D object tracking in real-time. The developed algorithm uses edges as features to track, which are easy and robust to detect. It exploits the functionality of modern highly parallel graphics boards by performing hidden face removal, image processing, texturing and particle filtering. Using a standard 3D -model the tracker requires neither memory and time -consuming training nor other pre-calculations. In contrast to other approaches this tracker also works on objects where geometry edges are barely visible because of low contrast.

Keywords: Object tracking, edge detection, 3D tracking, real-time, image processing, feature matching

Best to read in color.

1 Introduction

Tracking the pose of a three dimensional object from a single camera is a well known task in computer vision. What seems to be simple for humans turns out to be significantly more complicated for computers. While humans are able to perform highly parallel image processing, even modern *central processing units (CPUs)* have problems calculating the pose of an object with sufficient accuracy, robustness and speed. This leads to the idea of using modern graphics cards, which also work in parallel, to solve this problem. This approach exploits the parallel power of *graphics processing units (GPUs)* by comparing edge features of camera images and a 3D -model.

Graphics boards are designed to render virtual scenes as realistically as possible. The main 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 science will soon allow the inclusion of more and more optical effects like shadows, reflections, shading, occlusions or even smoke, fire, water or fog.

1.1 Related work

One of the first successful approaches of tracking objects by their edges was RAPiD [Harris 1992]. It uses points on edges and searches for correspondence to its surroundings along the edge gradient. However this method lacks robustness and several improvements were applied to overcome this problem as in [Drummond and Cipolla 1999; Philipp Michel 2008; Luca Vacchetti and Fua 2004; Klein and Drummond 2003].

Another approach is to globally match model primitives with those from the camera image [Lowe 1992; Gennery 1992; D. Koller and Nagel 1993; Kosaka and Nakazawa 1995; A. Ruf and Nagel 1997]. This method has been used for robot and car tracking applications, but was later replaced by improved versions based on RAPiD.

[Lucie Masson and Jurie 2004] also uses edges and textures for tracking. They extract point features from the texture and use them together with the edges to calculate the pose. This turns out to perform very fast and robust against occlusion. Our approach not only uses patches but the whole texture, which usually lets the pose converge very quickly to the accurate pose. Since the algorithm runs on the GPU, it is as fast as the method in [Lucie Masson and Jurie 2004].

The work presented in [M. Vincze and Zillich 2001] uses edge features to track but does not take into account texture information. This makes it less robust against occlusion. Since the search area in that approach is very small, it is also less robust against fast movement and getting caught in local minima.

The work presented in this paper is based on [Klein and Murray 2006] where they take advantage of graphics processing by projecting a wireframe model into the camera image. Then a particle filter with a Gaussion noise model is used to evaluate the likelihood distribution of the pose.

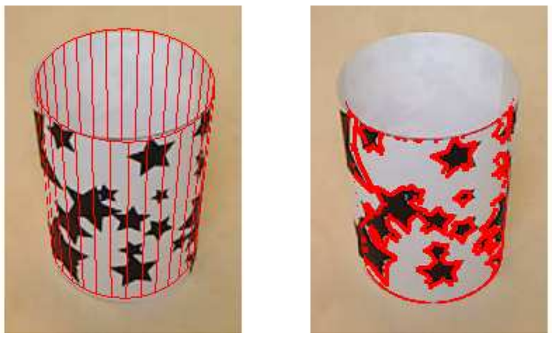


Figure 1: *Edges from geometry vs. edges from texture*

Our approach not only uses geometry edges but also edge features from textures which extends the class of trackable models by those that have curved surfaces as illustrated on the right of Figure 1. 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 Figure 1. The particle filter is extended by using it in a recursive design that evaluates a single pose estimate given by the most likely particles.

1.2 Overview

The idea of this approach is to

- extract the edges from the incoming camera image,
- extract the edges from the textured 3D -model,
- generate hundreds of slightly different views of the model relative to a pose estimate,
- calculate the most likely pose of the model by matching the edges of the camera image and the 3D -model.

The algorithm developed is separated into two parts. The *preprocessing* in Section 2, where all possible pre-calculations are made

[*]e-mail: moerwald@acin.tuwien.ac.at
[†]e-mail: zillich@acin.tuwien.ac.at
[‡]e-mail: vincze@acin.tuwien.ac.at

and the *recursive particle filtering* in Section 3, which generates several hundred views and for each of them evaluates the match likelihood. Therefore the second part is very time -crucial. A *linear Kalman filter* described in Section 4 is applied for smoothing the resulting trajectory. Section 5 gives some hints on how to implement the proposed methods. The *results* in Section 6 and the *conclusion* in Section 7 summarize the advantages and strengths of the presented tracker.

2 Preprocessing

In the preprocessing stage of the algorithm the edge image I_C^e of the incoming camera image I_C is extracted and stored for comparison later. The color surface S of the object to track is *projected* into the incoming image I_C and the edges are calculated again. This edge image of the object I_S^e is then used to *re-project* to the geometry in world space which results in the corresponding edge map of the original surface S^e.

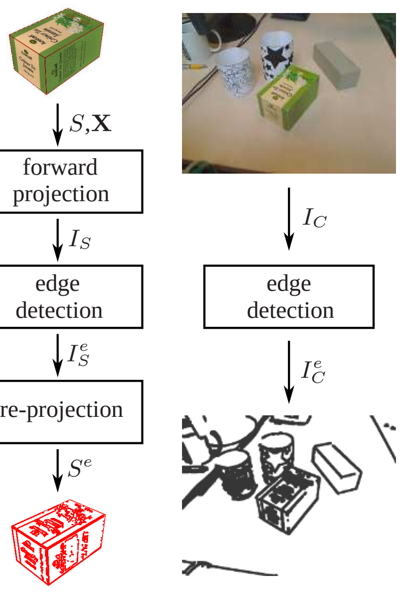


Figure 2: *Flow chart of preprocessing*

2.1 Edge detection

The edges I^e are found by convolving the original image I with a Gaussian smoothing $\mathbf{G}$ and the two Sobel kernels $\mathbf{H}_{s,x}$ and $\mathbf{H}_{s,y}$ as described in [Burger and Burge 2008].

$$I^e = \begin{pmatrix} I_x^e \\ I_y^e \end{pmatrix} = \begin{pmatrix} \mathbf{H}_x * \mathbf{G} * I \\ \mathbf{H}_y * \mathbf{G} * I \end{pmatrix} \tag{1}$$

Furthermore, the result is improved by applying thinning and spreading algorithms. Note that for the gradient calculation in Section 3.2 the x and y values are stored separately.

Figure 3 shows the different results of edge detection where the x- and y-components of the gradient are stored in the red and green color channel. The detection tolerance can be influenced by applying spreading, which broadens the edges by a specific number of pixels according to the tolerance level. This means that instead of searching for edge pixels close to each other, the line width of the edge is raised as shown in Figure 3, which broadens the matching area.

2.2 Forward projection

The 3D -model is projected into the camera image I_C as defined by Equation (2). Using the camera image I_C takes into account that edges are not visible when there are similar light and color conditions in the background. Then the edges of the image are extracted using Equation (1).

The transformation of the model from world space to image space is performed by the following matrix operations:

$$\begin{aligned} \mathbf{u}_S &= \mathbf{T}_p \mathbf{X} \mathbf{v}_S \\ I_S(u,v) &= \begin{cases} S(\mathbf{v}_S) & \text{if} \quad (u,v) \in \boldsymbol{\mathcal{U}} \\ I_C(u,v) & \text{else} \end{cases} \end{aligned} \tag{2}$$

where I_S is the camera image with the projected model. $\boldsymbol{\mathcal{U}}$ defines the geometry of the object in image space with

$$\mathbf{u}_S = [u_S, v_S] \in \boldsymbol{\mathcal{U}}$$

$\mathbf{T}_p$ denotes the projection- and $\mathbf{X}$ the model view or world transformation matrix which defines the pose of an object to track with a rotational and translational term $\mathbf{R}$ and $\mathbf{t}$.

$$\mathbf{X} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$$

The geometry of the object in world space $\boldsymbol{\mathcal{V}}$ is represented by its vectors

$$\mathbf{v}_S = [x_S, y_S, z_S] \in \boldsymbol{\mathcal{V}}$$

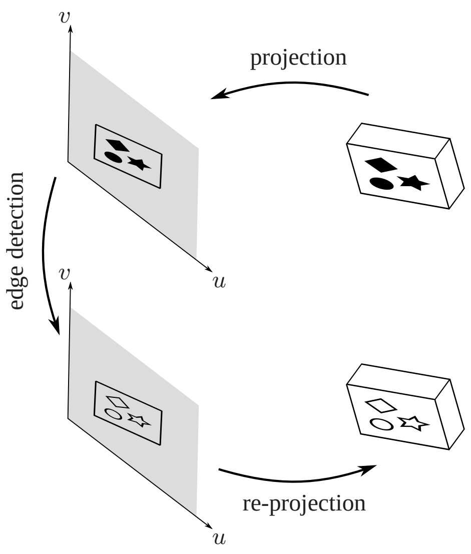


Figure 4: *Forward projection and re-projection*

2.3 Re-projection

The idea of re-projection is to replace the color surface of the 3D -model with the corresponding edge map. Note that it is not possible to do the image processing on the color surface S of the object directly, as the edge features get distorted and thinned out when they are projected to image space and therefore wrongly fail the edge matching test. Comparing the edges of the model with the camera image requires the same methods applied to the same point of view and also the same scaling of the edge width.

Using a particle filter requires drawing the model several hundred times at different poses $\mathbf{X}_i$ with $i = 1...N$. Replacing the surface

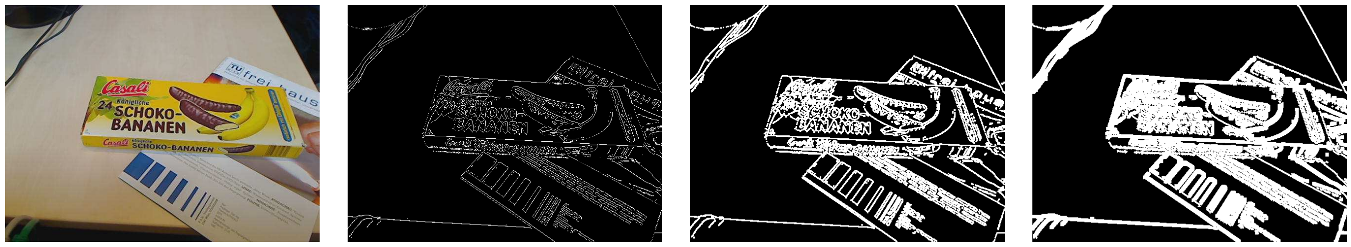


Figure 3: *Edge detection results from left to right: original, no spreading, one time spreading, three times spreading*

and running the edge detection algorithm for each particle would cause the tracker to be far away from real-time capability. For this reason, the surface of all particles is replaced by only one edge map I_S^e, calculated by using the prior tracking result $\mathbf{X}^+$. This, in principle causes the same problems as mentioned above, but assuming that the motion of the particles $\mathbf{X}_i$ remains small within one tracking pass, the distortions and thinning out can be disregarded.

3 Recursive particle filtering

For each tracking pass the recursive particle filtering executes the methods shown in Figure 5. First the particles $i = [1 \dots N]$, representing the pose of the object, are generated using Gaussian noise. Then the likelihood of each particle i is evaluated by matching it against the edge image of the camera I_C^e. If there is still processing time t_f remaining, then a further recursion step of particle generation and likelihood evaluation is performed with different parameters as described in Section 3.4. Otherwise the maximum likely particle is passed to the next step. The linear Kalman filter, including a physical motion model, is attached to the outcome of the recursive particle filter to fine tune the result and remove remaining jitter. As this additional filter is not part of the recursion it is explained in Section 4.

The reason for this setup is to benefit from the robustness and speed of a particle filter. For higher accuracy, the standard deviation of the noise is reduced in each recursion. The linear Kalman filter is attached just for fine tuning as mentioned above.

3.1 Particle generation

The prior pose $\mathbf{X}_i^-$ of each particle is calculated by perturbing the posterior $\mathbf{X}^+$ with Gaussian noise $\mathbf{n}(\boldsymbol{\sigma}^2)$ with a standard deviation scaled by the prior likelihood of the pose w_m, a scaling factor for motion effect $\mathbf{f}_m$ and a scaling factor $\mathbf{f}_c$ set by each recursion step:

$$\begin{aligned} \mathbf{X}_i^- &= \mathbf{X}^+ + \mathbf{n}(\boldsymbol{\sigma}^2(w_m, \mathbf{f}_m, f_c)) \\ i &= [1 \dots N] \end{aligned} \tag{3}$$

The standard deviation is evaluated by

$$\boldsymbol{\sigma} = \boldsymbol{\sigma_I} \mathbf{f}_m \mathbf{f}_c . (1 - w_m) \tag{4}$$

where the prior pose likelihood w_m is multiplied to the initial standard deviation $\boldsymbol{\sigma}_I$ so that the particle distribution narrows with higher likelihood. The motion effect $\mathbf{f}_m$ takes into account that motion in world space along the camera viewing axis causes less change in image space then the same motion orthogonally to the viewing axis. $\mathbf{f}_c$ becomes smaller with each recursion step in the particle filter. $\boldsymbol{\sigma}_I$ is implemented as a parameter to be set by the user, but should be evaluated automatically in the future regarding the tracking conditions.

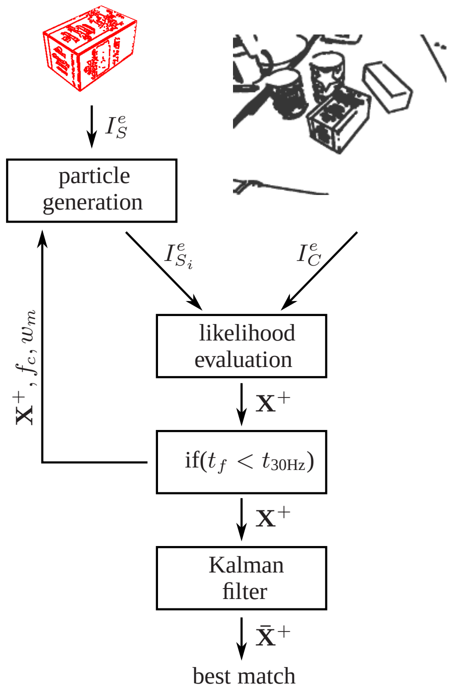


Figure 5: *Block scheme of motion with recursive particle filter and Kalman filter*

3.2 Likelihood evaluation

Each particle is tested against the camera image and a matching likelihood is calculated. Therefore the correlation between the gradients of the edges $\mathbf{g}_{S_i}(u, v)$ and $\mathbf{g}_C(u, v)$ is evaluated by comparing the direction of the edges at each image point (u, v).

$$\begin{aligned} \mathbf{g}_{S_i}(u,v) &= \begin{pmatrix} I_{S_i,x}^e(u,v) \\ I_{S_i,y}^e(u,v) \end{pmatrix} \\ \mathbf{g}_C(u,v) &= \begin{pmatrix} I_{C,x}^e(u,v) \\ I_{C,y}^e(u,v) \end{pmatrix} \end{aligned}$$

The angles between those vectors are calculated, producing the edge correlation image Φ_i:

$$\begin{aligned} \phi &= \arccos\left(\mathbf{g}_{S_i} . \mathbf{g}_C\right) \\ \Phi_i(u,v) &= \begin{cases} 1 - \frac{2\phi}{\pi} & \text{if} \quad \phi < \pi/2 \\ 1 - \frac{2|\phi - \pi|}{\pi} & \text{if} \quad \phi > \pi/2 \\ 0 & \text{if} \quad (u,v) \notin \mathbf{v}_S' \end{cases} \end{aligned} \tag{5}$$

Note that it is assumed that the result of the $\arccos()$ function lies within 0 and π. The image Φ_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 Φ_i is scaled to the range of 0 to 1.

The match likelihood w_i is calculated by integrating Φ_i and dividing it by the integrated edge image of the surface $I^e_{S_i}$.

$$w_i = \frac{\int_u \int_v \Phi_i(u,v)dvdu}{\int_u \int_v |I^e_{S_i}(u,v)|dvdu} \tag{6}$$

3.3 Determining the pose

As explained in the sections 2.2 and 2.3 for projection and re-projection of the model, a single pose $\mathbf{X}^+$ has to be defined. This is where the approach suggested in this paper differs from usual, straight forward particle filters, where the whole propability density function defined by all $\mathbf{X}_i$ is carried over into the next estimation step.

The pose $\mathbf{X}^+$ is evaluated using the mean of the top most likely particles M. $\mathbf{X}^-_k$ in Equation (7) denotes the particles sorted by likelihood w_k in descending order.

$$\mathbf{X}^+ = \frac{1}{w_m} \sum_{k=1}^{M} \mathbf{X}^-_k . w_k \tag{7}$$

with

$$w_m = \frac{1}{M} \sum_{k=1}^{M} w_k$$

Experiments have shown that increasing the number of most likely particles M to consider in the mean likelihood, while lowering the standard deviation in Equation (4) by f_c and the edge width (see Figure 3) for each further recursion obtains good results.

3.4 Recursion

The methods described in Section 3.1 and 3.2 are performed for each of the hundreds of particles. The idea of recursion is to take advantage of the information gain when calculating. This means that the N particles are divided into subranges $R_1, R_2, R_3, \ldots$ and so forth. For every range R_k, the pose estimate $\mathbf{X}^+$ and likelihood w_m of the previous particle filtering R_{k-1} is used. The standard deviation for the particle generation is reduced by the scaling factor $\mathbf{f}_c$, which narrows the *search area* of the filter. Therefore Equation (3) becomes

$$\begin{aligned} \mathbf{X}^-_i(R_k) &= \mathbf{X}^+(R_{k-1}) + \mathbf{n}\left(\boldsymbol{\sigma}^2\right) \\ i &= [1 \ldots N_k] \end{aligned}$$

with

$$\boldsymbol{\sigma} = \boldsymbol{\sigma_I} \mathbf{f}_m f_c(R_{k-1}).(1 - w_m(R_{k-1}))$$

Figure 6 shows the principal idea of recursive particle filtering in 2D. In the lower left graph the particles are perturbed using a high standard deviation of the Gaussian noise of the particles, covering a large area around the prior pose estimate. The mean of the top most likely particles is used to evaluate the rough pose of the real object. The upper left graph shows particles with lower standard deviation, where this time the most likely particles measure the real pose much more accurately.

The example in Figure 6 is drawn with 1500 particles with high and 500 with low standard deviation. This method allows the tracker to respond both quickly and accurately without wasting performance as it does not require any more particles than before.

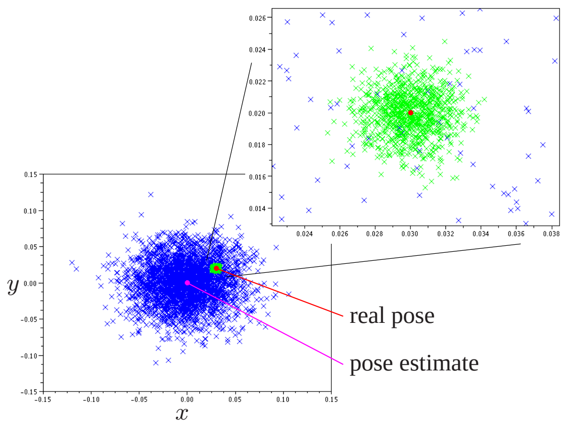


Figure 6: *Recursive particle filter*

4 Linear Kalman filtering

The discrete Kalman filter implemented for this approach uses a motion model for all six degrees of freedom of the object. The reason why the motion model is not applied in the particle filter is because this would reduce the speed and accuracy of the tracker, as modelling the velocity of the object would rise the degree of freedom from 6 to 12. This would mean that the particles have to cover 6 more dimensions. Therefore the Kalman filter is attached only to smooth the resulting trajectory and remove remaining jitter.

Time Update - "Predict"

$$\begin{aligned} \mathbf{x}^-_k &= \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k \\ \mathbf{P}^-_k &= \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{Q} \end{aligned}$$

Measurement Update - "Correct"

$$\begin{aligned} \mathbf{K}_k &= \mathbf{P}^-_k\mathbf{H}^T\left(\mathbf{H}\mathbf{P}^-_k\mathbf{H}^T + \mathbf{R}(w_m)\right)^{-1} \\ \mathbf{x}_k &= \mathbf{x}^-_k + \mathbf{K}_k\left(\mathbf{z}_k - \mathbf{H}\mathbf{x}^-_k\right) \\ \mathbf{P}_k &= \left(\mathbf{I} - \mathbf{K}_k\mathbf{H}\right)\mathbf{P}^-_k \end{aligned} \tag{8}$$

where

$$\mathbf{x}_k = \left[x_k, y_k, z_k, \alpha_k, \beta_k, \gamma_k, \dot{x}_k, \dot{y}_k, \dot{z}_k, \dot{\alpha}_k, \dot{\beta}_k, \dot{\gamma}_k\right]$$

denotes the state of the Kalman filter containing the pose and velocity of the six degrees of freedom.

$$\mathbf{A} = \begin{bmatrix} \mathbf{I} & \text{diag}(\Delta t) \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$$

is the *state matrix*,

$$\mathbf{B} = [\text{diag}(0)]$$

the *input matrix*,

$$\mathbf{H} = [\mathbf{I}, \text{diag}(0)]$$

the *output matrix*, $\mathbf{P}_k$ the *estimate error covariance*, $\mathbf{Q}$ the *process noise covariance*, $\mathbf{R}$ the *measurement noise covariance*, $\mathbf{I}$ the unity matrix and $\mathbf{K}_k$ the *Kalman gain*. Please refer to [Welch and Bishop 2004] for more details on Kalman filtering.

The process covariance matrix $\mathbf{Q}$ defines the noise of the physical model, where no information is available. However, in contrast to $\mathbf{R}$, this matrix is set to a fixed value.

The measurement covariance matrix $\mathbf{R}$ depends on the pose likelihood as follows:

$$\mathbf{R}(w_m) = \begin{bmatrix} \text{diag}((w_{t-3})^3) & 0 \\ 0 & \text{diag}((w_{t-3})^3) \end{bmatrix}$$

As shown in Equation (8), this means that $\mathbf{R}$ rises proportionally to the delayed likelihood of the measurement from the particle filter. Therefore, the Kalman gain drops, giving less weight to the measurement $\mathbf{z}_k$ and more weight to the motion model, which smooths the result. This means that jitter is removed when the likelihood is high, for example when the object to track does not move.

On the other hand, the lag behind the real object, caused by the motion model during acceleration, is decreased when the likelihood falls, which usually happens when the object to track moves fast. In this case, the Kalman gain increases the weight of the measurement, increasing the speed but also allowing jitter which is barely visible when the object is moving anyway.

Experimants have shown that a cubic likelihood function yields to a smooth tracking behaviour. The delay of the likelihood w_{t-3} removes overshooting of the motion model when the object suddenly stops moving.

5 Implementation notes

The implementation of the algorithm requires discretization which is denoted by bold letters for images in this section.

5.1 Notes for preprocessing

The preprocessing is done once per tracking pass and is not as time critical as the recursive particle filtering in the following section. However, the overall performance of the tracker needs to be as fast as possible, so this part is also implemented using the graphics processing unit. Therefore, the image received by the camera is sent to the graphics board where it is stored as texture. The convolution with the Gaussian and Sobel kernels are applied by shaders as well as thinning and spreading. The edge image is again stored as *RGB* texture where the *R*- and *G*-channels are used for the x- and y-components of the image gradient.

As the channels only allow values between 0 and 1, the normalized gradients ranging from -1 to 1 need to be adjusted.

The surface of the object to track is made up of vertices which form triangles and quadrangles, also called primitives. The projections described in Equation (2) and (9) are performed for these vertices only, because the surface points within a primitive can be determined by linear interpolation, which is optimized by the graphics adapter.

$\mathbf{T}_p$ and $\mathbf{X}$ denote the projection and model view matrix, which can be queried from OpenGL. The six degrees of freedom are stored in the vector

$$\mathbf{x}_i = [x_i, y_i, z_i, \alpha_i, \beta_i, \gamma_i]$$

Instead of transforming the edge map I_S^e back to world space by solving Equation (2) with respect to S, the coordinates for the edge map in image space are evaluated. This has to be done only once, since those coordinates do not change for the other particles.

$$\begin{aligned} \mathbf{u}_S^+ &= \mathbf{T}_p \mathbf{X}^+ \mathbf{v}_S \\ I_{S_i}^e(u,v) &= I_S^e(\mathbf{u}_S^+) \end{aligned} \tag{9}$$

The particles i are represented by the pose matrices $\mathbf{X}_i^-$. The vectors $\mathbf{u}_{S_i}$ that are used to find the corresponding point in the camera image I_C^e are calculated with

$$\begin{aligned} \mathbf{u}_{S_i} &= \mathbf{T}_p \mathbf{X}_i^- \mathbf{v}_S \\ I_{C_i}^e(u,v) &= I_C^e(\mathbf{u}_{S_i}) \end{aligned}$$

At this point for each particle i the model $I_{S_i}^e$ and camera image $I_{C_i}^e$ are ready for comparison which is described in Section 3.2.

5.2 Notes for recursive particle filtering

The calculations described in this section are very critical with respect to optimized programming, since every single line of code is called several hundreds of times. In particular evaluating the match likelihood using the *NVIDIA Occlusion Query* is definitely a bottleneck in the algorithm. This OpenGL extension is responsible for counting the pixels of the whole edge map. The matching pixels m_i and total edge pixels n_i are evaluated by summing up the correlation image $\mathbf{\Phi}_i$ and the edge map $\mathbf{I}_{S_i}^e$:

$$\begin{aligned} m_i &= \sum_{u,v} \mathbf{\Phi}_i(u,v) \\ n_i &= \sum_{u,v} \mathbf{I}_{S_i}^e(u,v) \end{aligned}$$

As this extension only supports counting pixels disregarding the value, they only can be marked to be rendered or not. Therefore, the displacement image is thresholded by the angle ϵ:

$$\begin{aligned} \phi &= \arccos\left(\mathbf{g}_{S_i}(u,v).\mathbf{g}_C(u,v)\right) \\ \mathbf{\Phi}_i &= \begin{cases} 0 & \text{if} \quad \phi < \epsilon \\ 1 & \text{if} \quad \phi \geq \epsilon \end{cases} \end{aligned} \tag{10}$$

where ϵ denotes the angular threshold. For the match evaluation shader 0 means that the pixel is discarded when rendering and does not increase the counter, whereas 1 means the pixel is drawn and therefore increases m_i.

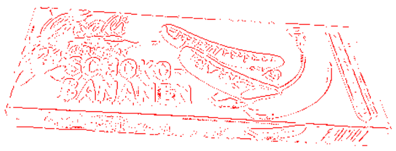
Figure 7b) shows the pixels m_i successfully passing the match evaluation. 7c) are the pixels which fail the match test of Equation (10) and 7d) is the total number of pixels n_i of the edge image of the object. Note the missing pixels on the very upper edge, as a result of the background having the same color (*yellow*) as the object. The mismatch of the edges on the front side of the box is caused by the inaccuracy of the placement of the texture on the geometry of the model which is produced manually by a 3D modeling tool.

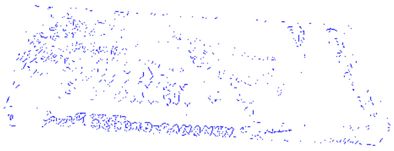
The likelihood w_i is evaluated by

$$w_i' = \left(\frac{m_i}{n_i} + \frac{m_i}{n_{max}}\right) \frac{1}{w_{pos}}$$

The first term within the brackets is the percentage of matching edge pixels m_i with respect to the total visible edge pixels n_i. Calculating the likelihood only with this term would cause the tracker to lock when only one side of the 3D object is visible. When at this special pose the object is rotated slightely, another side of the object becomes visible. The particle matching this rotation would be equal or most often less then the prior front facing particle. The reason for this is that the number of matching pixels m_i grow less than the total visible pixels n_i when rotating the object out of the front side view. This effect amplifies when taking into account that edge detection for strongly tilted faces is very faulty.

a) Object to track with cluttered background


b) Matched edges m_i


c) Mismatched edges

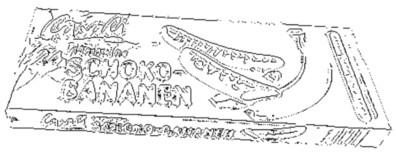

d) Total edges of model n_i

Figure 7: *Edge matching*

The second term allocates more weight to the total number of matching pixels m_i which is intrinsically higher for the rotated particle. n_{max} which are the maximum visible edge pixels in the actuall area scales the pixels to the proper range. As this summation would lead to likelihood values higher than 1, it is divided by the maximum possible likelihood w_{pos} which is updated online.

This differs from the likelihood calculation used in [Klein and Murray 2006]

$$\text{Likelihood}\left(\mathbf{X}_i^-\right) \propto \exp\left(k\frac{d_i}{v_i}\right)$$

which we experienced to lock very fast at the local minima mentioned above. Here d_i denotes the number of matching edge pixel, v_i the total edge pixels of the wireframe model, where the hidden edges are removed and k is a constant for distributing the likelihood.

6 Results

As the tracker requires fast parallel calculations, the focus of the system is on the graphics board where it is implemented. It has been tested on an NVIDIA GeForce GTX 285 with a fill rate of 50 billion pixels per second, and an Intel Core2 Quad CPU Q6600 with 2.4 GHz. To fulfill a minimum frame rate of 30 FPS (frames per second), the time for one tracking pass is 33 ms. Within this limits it is possible to draw 1500 particles for a box (6 faces textured with a 900x730 pixel image) like in Figure 7a). With the cylinder model shown in Figure 1 with 16 faces and a 900x400 pixel texture image, the tracker achieves 580 particles within the same time and 100 particles with the cylinder with 64 faces.

Figure 8 shows some results of a video sequence. The first row demonstrates the robustness of the tracker. The whole top surface and the edges of the geometry are covered by the hand and there are reflections of the checkerboard pattern on the front face and a cluttered background, but the pose of the box still can be estimated. Of course at some degree of occlusion, the accuracy of the pose drops until it cannot be determined at all.

The second row shows the fast and reliable convergence. Although the deviation of the estimated from the actual pose is very high, the tracker finds the correct alignment within a second. The third row illustrates the concept of recursive particle filtering. Especially in the second image from the left, where the speed of the moving object is high, the benefit is clearly visible (compare with Figure 6).

7 Conclusion

We presented a method for fast and robust object tracking. It converges fast to the correct pose and is able to handle relative large deviations, for example when initializing. Partial occlusion, reflections, light changes, shadows and cluttered background are no problem for the tracker, as long as enough features are visible to determine the pose. Exploiting the power of a graphics processing unit with a particle filter in a recursive design allows high tracking speed with sufficient accuracy.

However, there are several improvements possible. Firstly the mismatch of the 3D -model to the real object, as described in Section 5.2, can be learned and corrected with constraints that prevent the model from strong distortion. Secondly corner detection, color matching and so forth can be implemented which would most likely further improve the accuracy and robustness of the tracker.

The correlation map Φ_i in Equation (5) is simplified to Equation (10), because the *NVIDIA Occlusion Query* only counts visible pixels disregarding the information about the angular displacement stored within. A future work would be to implement precise likelihood evaluation as described in Equations (5) and (6).

The bottle-neck, with respect to the frame time of this approach, is definitely the particle filter with its likelihood evaluation using the OpenGL extension. The tracking error e directly correlates with the standard deviation σ and number of particles N as follows:

$$e \propto \frac{\sigma}{N}$$

with

$$N \propto t = t_{33ms}$$

This means that the tracking error e can be significantly reduced by lowering the standard deviation σ for each degree of freedom independently. When for example the z position of an object to track is known, because it lies on a table the accuracy can be increased by lowering the standard deviation for this degree of freedom.

There are several points of the algorithm where further investigation needs to be done, like finding the optimal boundary conditions and functions for the recursive particle filter and designing a better function for particle generation. Or more specifically, designing better functions for calculating the standard deviation of the Gaussian noise in Equation (4).

A further problem that needs to be solved is, that the tracker cannot supply information if tracking fails when it locks into a local minimum. There, the likelihood evaluation returns sometimes values as high as at the correct tracking pose.

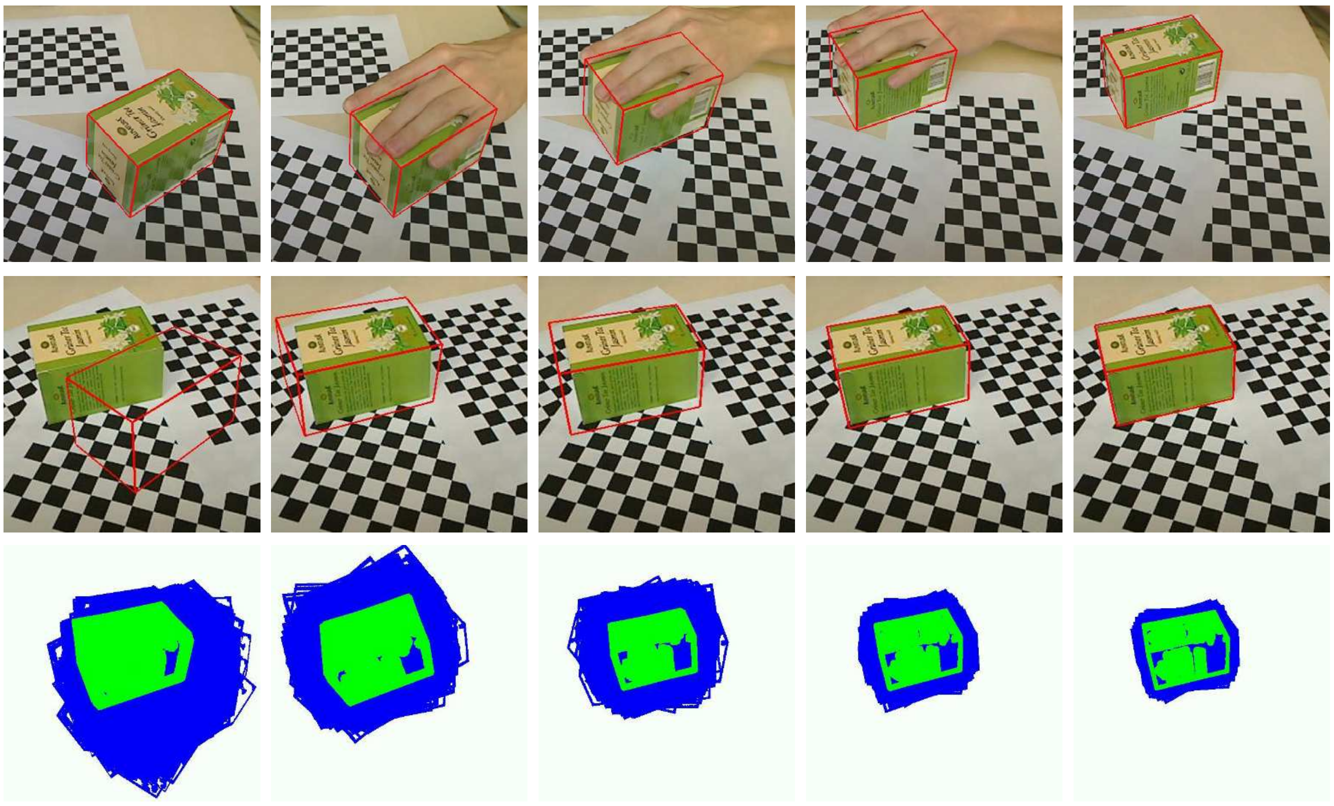


Figure 8: *First row: robustness against occlusion, reflections and background clutter; Second row: fast and robust convergence; Third row: particle distribution with three recursions*

Acknowledgements

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

References

A. Ruf, M. Tonko, R. H., and Nagel, H.-H. 1997. Visual tracking by adaptive kinematic prediction. *Proceedings of International Conference on Intelligent Robots and Systems*.

Burger, W., and Burge, M. J. 2008. *Digital Image Processing, An Algorithmic Introduction Using Java*. Springer.

D. Koller, K. D., and Nagel, H.-H. 1993. Model-based object tracking in monocular image sequences of road traffic scenes. *International Journal of Computer Vision*.

Drummond, T., and Cipolla, R. 1999. Real-time tracking of complex structures with on-line camera calibration. 574–583.

Gennery, D. 1992. Visual tracking of known three-dimensional object. *International Journal of Computer Vision*.

Harris, C. 1992. Tracking with rigid objects. *MIT Press*.

Kessenich, J. 2008. *The OpenGL Shading Language, Version 1.30*.

Klein, G., and Drummond, T. 2003. Robust visual tracking for non-instrumented augmented reality.

Klein, G., and Murray, D. 2006. Full-3d edge tracking with a particle filter. *British Machine Vision Conference Proc 17th*.

Klein, G., and Murray, D. 2007. Parallel tracking and mapping for small ar workspaces. *Proc International Symposium on Mixed and Augmented Reality (ISMAR)*.

Kosaka, A., and Nakazawa, G. 1995. Vision-based motion tracking of rigid objects using prediction of uncertainties. *International Conference on Robotics and Automation*.

L. Vacchetti, V. L., and Fua, P. 2004. Stable real-time 3d tracking using online and offline information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Lowe, D. G. 1992. Robust model-based motion tracking through the integration of search and estimation. *International Journal of Computer Vision*.

Luca Vacchetti, V. L., and Fua, P. 2004. Combining edge and texture information for real-time accurate 3d camera tracking.

Lucie Masson, M. D., and Jurie, F. 2004. Robust real time tracking of 3d objects.

M. Vincze, M. Ayromlou, W. P., and Zillich, M. 2001. Edge-projected integration of image and model cues for robust model-based object tracking. *The International Journal of Robotics Research*.

Mustafa Özuysal, Michael Calonder, V. L., and Fua, P. 2009. Fast keypoint recognition using random ferns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

P.A. SMITH, I. R., AND DAVISON, A. 2006. Real-time monocular slam with straight lines. *Proc 17th British Machine Vision Conference* (sept).

PHILIPP MICHEL, J. C. E. A. 2008. Gpu-accelerated real-time 3d tracking for humanoid autonomy.

R. KOCH, K. KOESER, B. S., AND EVERS-SENNE, J.-F. 2005. Markerless image-based 3d tracking for real-time augmented reality applications. *WIAMIS* (april).

ROST, R. J. 2006. *OpenGL Shading Language*, vol. Second Edition. Addison-Wesley.

SEGAL, M., AND AKELEY, K. 2008. *The OpenGL Graphics System: A Specification, Version 3.0*.

SUN-KYOO HWANG, M. B., AND KIM, W.-Y. 2008. Local discriptor by zernike moments for real-time keypoint matching. *IEEE Congress on Image and Signal Processing*.

VINCENT LEPETIT, JULIEN PILET, P. F. 2004. Point matching as a classifiaction problem for fast and robutst object pose estimation. *Conference on Computer Vision and Pattern Recognition* (june).

WELCH, G., AND BISHOP, G. 2004. An introduction to the kalman filter.

3D Shape Detection for Mobile Robot Learning

Andreas Richtsfeld and Markus Vincze
Technische Universität Wien
Gußhausstraße 27-19 / E376
1040 Wien, Austria
ari@acin.tuwien.ac.at, vincze@acin.tuwien.ac.at

Abstract

If a robot shall learn from visual data the task is greatly simplified if visual data is abstracted from pixel data into basic shapes or Gestalts. This paper introduces a method of processing images to abstract basic features into higher level Gestalts. Grouping is formulated as incremental problem to avoid grouping parameters and to obtain anytime processing characteristics. The proposed system allows shape detection of 3D such as cubes, cones and cylinders for robot affordance learning.

Introduction

Humans can solve even complex tasks such as manipulating or grasping objects by learning and using the relationship of object shape to the intended behaviour - also referred to as affordances. Robotics would profit considerably if a robot could learn these affordances [1] and relate them to visual percepts. However, at present visual perception delivers only image-based features. It is the objective of this work to present a method that extracts the basic 3D shape of objects for enabling more realistic robot learning. Fig. 1 gives an example of a robot that attempts to learn concepts such as moveability of objects.

There are several paradigms of making robots learn affordances. [2, 3] present an approach where a mobile robot observes the environment and autonomously learns a prediction model from its actions and observation data. An overhead camera is used to track colour blobs of the robot and the objects. Another possibility is to discover object affordances [4]. To track objects and estimate the object behaviour, they are also using colour blobs and a set of visual features to describe the 2D blob shape. As an example, a circle can roll but a rectangle cannot. Another possibility is learning high-level ontologies, as proposed in [5, 6], by clustering the point data from a laser range scanner with an occupancy grid map.

All these approaches for autonomous learning are dealing with simple vision data, which are determining relative positions of objects and use image-based point or blob features. However, to understand the usage of different objects the capability to abstract shapes from images into 3D shapes is necessary. This is particularly true for object grasping but also holds for navigation, where an abstraction of point data into geometric features reduces complexity, thus enables efficient learning and yields an effective description of affordances.

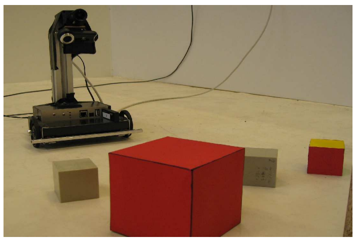
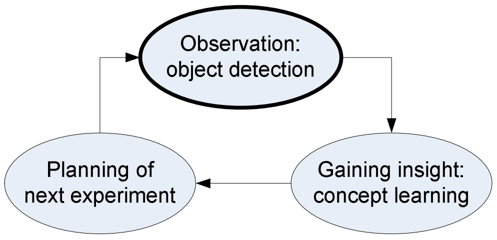


Observation: object detection
Gaining insight: concept learning
Planning of next experiment

Fig. 1 Robot attempting to learn moveability of objects and the basic learning loop of planning robot motions, detecting objects (the focus of this paper) and gaining insights [2].

Hence we propose a system using methods of perceptual organisation to estimate basic object shapes in a hierarchical manner such that affordances can link to different abstractions as required. This aims at offering a generic tool for autonomous learning of robots in their environments.

Related Work

Perceptual recognition is a well known problem in computer vision and was research topic since the Eighties. Seminal works regarding the theory of perceptual recognition of simple objects are [7] and [8]. Biederman [7] proposed that there are non-accidental differences between simple objects, which can be derived through so called geons. [9] discusses the potential of geons in computer vision systems. Perceptual grouping is a bottom-up method to estimate Gestalts, where new hypotheses will be estimated from Gestalts of lower-level Gestalt principles. It can also be seen as geometric grouping, because most of the Gestalt principles implements geometric restrictions.

The main challenge of perceptual grouping is to limit the combinatorial explosion. Indexing and thresholding of less salient hypotheses are normally used to solve this problem [8, 10]. Indexing divides the search space for relations between elements into bins and each element will be allocated to a bin depending on the grouping relationship. Further all elements in a bin can be analysed whether they fulfil the given relation or not. A method for indexing in image space is proposed in [11], where search lines for a Gestalt feature are used to find intersections. Indexing into the image pixels limits the search space and avoids the problem of comparing all combinatorial possibilities to find connections. Depending on the type of search line, different types of groupings can be found.

A dilemma of using search lines is the length definition. When the length of the search lines is too small, some important Gestalts could not be found and on the other side, if they are too long, enhancement of computation time could be provoked because many combinations may occur. This problem can be avoided with incrementally growing search lines [11], where in every processing circle another search line grows one pixel. When drawing into the indexing image, intersections with existing search lines are detected immediately in the respective bin. The biggest advantage of this method is that any preset parameters or thresholds are now moved into longer and longer processing. The results are no longer dependent to the line length, but to the processing time. The longer the process operates, the more junctions can be found and thus more other Gestalts can be detected. This gives a possibility to evaluate the best possible hypotheses for a limited processing time and on the other side this makes it feasible to obtain results at any time (a quality referred to as anytimeness). This grouping approach is suited for robotics, since it allows to sequential group higher level Gestalts such as cubes or cylinders in short processing time.

System Overview: Perceptual Grouping to Detect Object Shape

We propose an incremental grouping method to enable efficient abstraction of image pixel data into a hierarchy of basic geometric Gestalts shown in Fig. 2. Triggered by search lines in the image each Gestalt creates new hypotheses and delivers them to Gestalts of higher levels for robot learning.

When following the proposal of [11] it is possible to build a perceptual grouping system which processes most of the Gestalt principles incrementally. To extend from this work, we firstly applied the idea of incremental indexing to ellipses. Hence, we insert the Ellipse-Junction as Gestalt element. Boxes in the non-shaded area of Fig. 2 are referring to basic geometric features, whereas the boxes in the shaded area indicate Gestalts obtained by incremental processing. This shows that the essential Gestalts are the Line- and Ellipse-Junctions. They are are providing the system with incremental indexing capabilities.

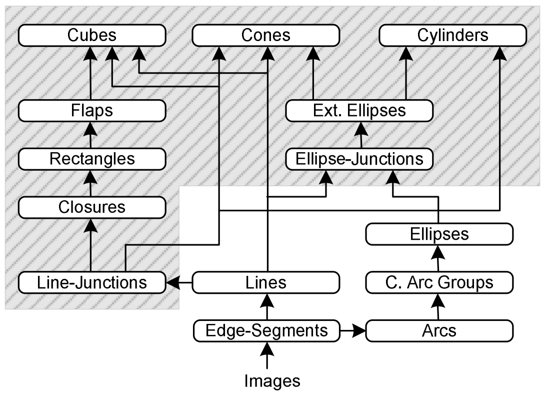


Fig. 2 Gestalt tree: basic geometric features and higher-level Gestalts incrementally built up (shaded).

In a first processing step the basic features are processed all at once and grouping uses neighbourhood constraints. Next, incremental processing of Line- and Ellipse-Junctions starts and the higher level Gestalts receive new input incrementally: each cycle extends one search line and all new hypotheses are processed at once to obtain new Gestalt hypotheses. If one of the Gestalts can hypothesise for a new Gestalt one level up the tree, the Gestalt from the next higher level begins to work, and so on. Hence the next junction will not be processed until all higher Gestalts are processed and no more hypotheses can be produced. This functionality of the processing tree guarantees that the best Gestalt hypotheses for the detected junctions are calculated in every processing circle.

The steps for building new Gestalts are the creation of new hypothesis, the ranking and masking, see Fig. 3. Whenever a Gestalt is informed about a new hypothesis, the processing starts and tries to build new hypotheses within this incoming Gestalt. The creation of new hypotheses exploits Gestalt principles from geometric constraints.

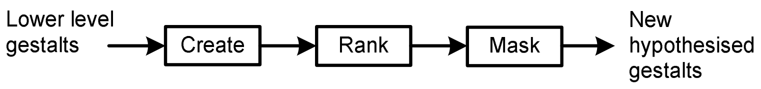


Fig. 3 Processing pipe for a Gestalt in the Gestalt tree.

After creation of new Gestalt hypotheses they will be ranked by quality criteria using the geometric constraints. Ranking is important to firstly calculate the best results and secondly for the masking of hypotheses. Masking is used to prune the poor results, e.g. when different hypotheses disagree about the interpretation of one element. Masking Gestalts is optional and is used to increase the performance and also to avoid combinatorial explosion in the following high-level principles.

Gestalt Principles

In this section we explain which Gestalt principles are used to build up the Gestalt tree of Fig. 2. The grouping of Edge-Segments into Lines, Arcs, Convex Arc Groups and Ellipses has been described in [11, 12]. Here we start with an explanation on how Line- and Ellipse-Junctions are formed to then introduce the approach to obtain the basic 3D shapes Cube, Cone and Cylinder.

Line-Junctions

The fundamental principles for incremental processing are Line and Ellipse- Junctions. Each processing circle starts with incrementing one search line of a line or an ellipse. Depending on which search line was chosen, the next hypothesis can be an Ellipse- or a Line-Junction. Line Junctions are connections between two lines. For each line six different search-lines are defined, as shown in Fig. 4, at each end one in tangential direction and two in normal direction. Different combinations of intersecting search lines are generating different junctions. We distinguish between Collinearities, L-Junctions and T-Junctions, where T-Junctions could also be interpreted as two L-Junction.

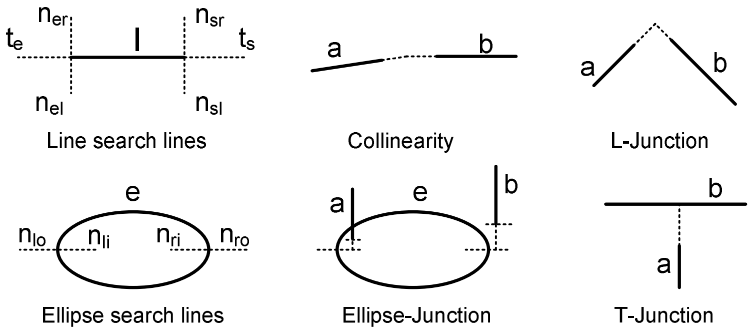


Fig. 4 Search lines for lines and ellipses and different junctions of lines and ellipses.

Ellipse-Junctions

Ellipse-Junctions are connections between a vertex of an Ellipse and a Line, which can also be found with search lines. Ellipse-Junctions have been defined to detect the higher-level Gestalts Cone and Cylinder. The search lines for Ellipses are growing on the main axis of the Ellipses, beginning at the vertices into both directions. For the growing of the search lines, we are using the same growing algorithm as for the search lines of Line-Junctions. Fig. 4 shows the search lines defined for Ellipses on both vertices.

Closures

Closures are closed convex contours, built from the Gestalts Lines and Line-Junctions, as proposed in [11]. Whenever the principle will be informed about a new Line-Junction, a new closed convex contour could be detected. The task is formalised as followed:

- Connect neighbouring lines to form a graph G = (V,E) with Lines as vertices (nodes) and Junctions between Lines as edges E of the graph.
- Perform a shortest path search on G, while making sure this path constitutes a roughly convex polygon.

To find these contours, Dijkstras algorithm for shortest path search is used, where only paths consisting of non-intersecting Lines, Collinearities and L-junctions of the same turning direction are allowed, thus ensuring simple convex polygon contours.

Rectangles

Rectangles can be directly derived from Closures by using geometric restrictions, which are given through the description of Rectangles in a perspective view. When considering these perspective projections as shown in [13] and neglecting one vanishing point (one-point projection), it is possible to recognize Rectangles, whenever a new Closure appears. We build a new Gestalt hypothesis, when:

- A Closure contains four L-Junctions, and
- Two opposing Lines between the four L-Junctions are approximately parallel.

Flaps

A Flap is a geometric figure consisting of two Rectangles that do not overlap and that have one line in common. Detecting Flaps is the intermediate step for detecting Cubes, because two sides of a Cube are always building a Flap in a perspective view of a camera. When seeing a Cube aligned to one side only a Rectangle or Flap is visible, making the Flap an obvious Gestalt element. We can formalise: Whenever a new Rectangle is hypothesised, a new Flap can be built, if

- It is possible to find another Rectangle, which shares one line with the new hypothesised Rectangle, and
- The two Rectangles do not overlap.

Cubes (Cuboids)

An image taken from a camera may show different perspective views of three-dimensional objects. In the case of a cube there may be one, two or three rectangles visible. The chance that one observed rectangle indicates a cube is small and increases for a flap, but both can occur accidentally. Only when we are able to find three adequate rectangles, we can conclude that the robot observes a cube. Cubes and cuboids can be built up from Flaps, Lines and Junctions in two different ways, shown in Fig. 5. We can formalise the detection of Cubes in the following way: Whenever a new Flap is hypothesised, build a Cube, if

- There are two Flaps (upper line in Fig. 5), which share one Rectangle with the new Flap (area 1 and 2) and the two Flaps are sharing one Rectangle (3), or
- There is one L-Junction and Lines that close the Flap from the two outer corners at the smaller inner angle of the Flap (lower line in Fig. 5).

The second method for finding cubes is more general than the method with three Flaps and leads to more poor results, because connections between corners of one Flap can also occur accidently. On the other side it gives the possibility to detect cubes, where one rectangle could not be detected.

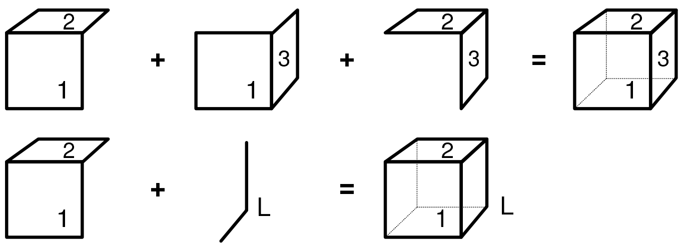


Fig. 5 Composing a cube from three Flaps or from a Flap with two Lines and an L-junction.

Extended Ellipses

Extended Ellipses are Ellipses with Lines attached trough Ellipse-Junctions. They are needed to build later the higher-level Gestalts Cone and Cylinder. We can describe the building of Extended Ellipses in the following way: Whenever a new Ellipse-Junction could be hypothesised:

- Build a new Extended Ellipse, if there is not one with the delivered Ellipse, or
- Assign the new Ellipse-Junction to the existing Extended Ellipse.

Cones

Cones are geometric figures consisting of ellipses (circles), lines and junctions between these components. We are using Extended Ellipses, Lines and Line-Junctions to find the object shape as given in Fig. 6. Building cones can be described in the following way:

- Whenever an Extended Ellipse (with at least one E-Junction at both vertices) could be hypothesised, build a Cone, if there is one L-Junction and Lines, which can close the Extended Ellipse.
- Whenever a new L-Junction can be hypothesised, build a new cone, if it is possible to close an Extended Ellipse with one Junction at both vertices.

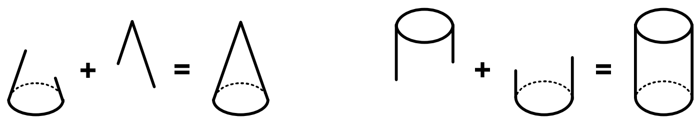


Fig. 6: Composing a cone and a cylinder from lower level Gestalts.

Cylinders

The object shape of a cylinder consists of an ellipse and lines, as shown in Fig. 6. Therefore Cylinders can be grouped from Extended Ellipses, Lines and Junctions. Building new Cylinders can be formalised to:

- Whenever an Extended Ellipse (with at least one E-Junction at both vertices) could be hypothesised, build a Cylinder, if there is another Extended Ellipse, whose lines are connected with the lines of the new Extended Ellipse.

Experiments and Results

The incremental grouping method has been evaluated with a mobile robot moving among simple geometric 3D objects. Fig. 7 shows an example image and the edge image containing several three-dimensional objects. The picture indicates the typical problem of grouping, namely that shadows or image noise creates spurios features such as lines or arcs. A grouping into higher level Gestalts sometimes accidentially includes a wrong feature though, more often the Grouping principle constrain the search to actual higher level Gestalts.

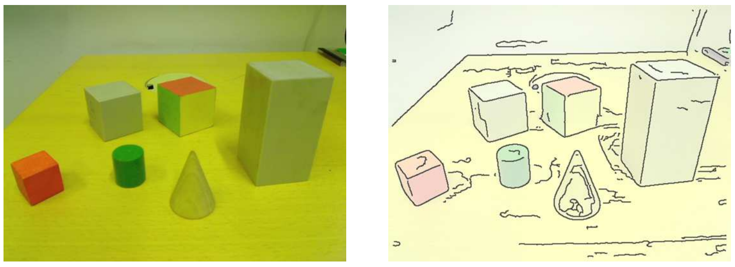


Fig. 7 Original image (left) and edge image with underlying half-transparent image (right).

With the incremental approach object detection depends on processing time. Fig. 8 shows in every row the results for different processing times. The first image (a) presents the resulting search lines, the second (b) the detected closures, the third (c) the best results of lower level Gestalts and the right column (d) the detected basic object shapes cube, cone and cylinder.

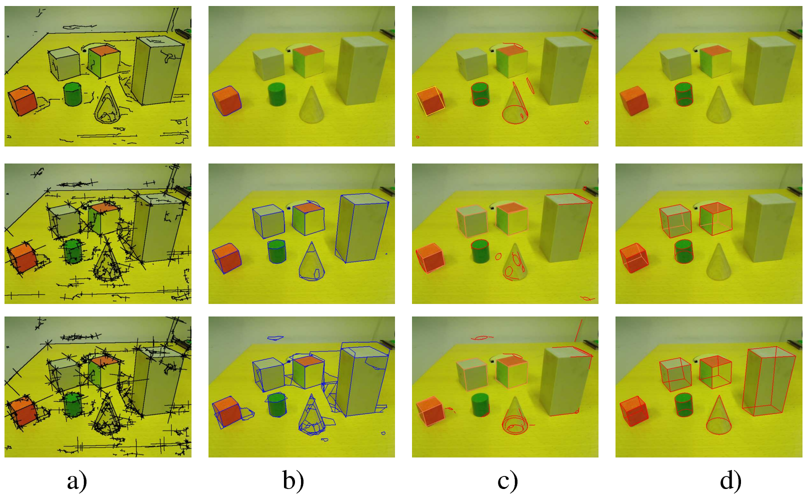


Fig. 8 Search lines (a), closures (b), the best ten lower level results (c) and simple object shapes (d) after 184ms (first row), 328ms (second row) and 468ms (third row).

Fig. 9 shows eight images from a sequence of a robot to a playground-scene with cubes (see Fig. 1). The whole sequence consists of 148 images, where 404 cubes could be observed in all images. For the detection of some cubes the processing time was too short or the view to the object too bad to detect the cube. In these cases the highest level Gestalt detected is shown. In nearly every image is at least one rectangle or a flap of a cube visible. Hence it is fair to assume that tracking of rectangles would be sufficient to follow cubes over degenerate views. For this sequence we evaluated the cube detection rate depending on processing time. The detection rate is 46% at 220ms processing time per image, 71% at 280ms (examples shown in Fig. 9) and 82% at 350ms, calculated with an Intel Core Quad

Q6600 with 2.40GHz. We can see that the detection rate grows for increased processing time as expected, but we note that increasing processing time to more than 500ms does not lead to further improvements: search lines are long enough to detect what is possible.

This clearly indicates the potential of the approach. Fig. 10 shows therefore results of real world images in an office and living room scene. While previous robot learning methods had to cope with pixel data [5] or 2D blobs [4] this could be exploited to learn from object to robot relationships in 3D in [3].

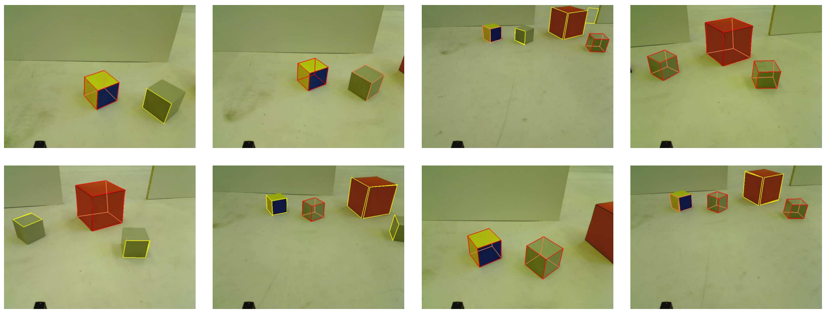


Fig. 9 Results for every 21 image from a 148 image sequence (from left to right) in a playground scene with 280ms runtime per image.

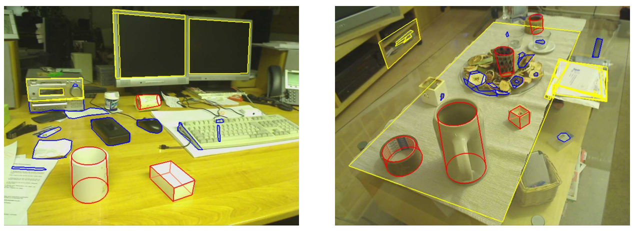


Fig. 10 Detected high-level Gestalts in an office and living room scene.

Conclusion and further work

We presented a hierarchical visual grouping system for the detection of basic geometric Gestalts such as cones, cylinders and cubes. With the incremental processing approach the problem of having parameters for each Gestalt principle is reduced to the single parameter processing time. The evaluation shows that basic shapes are detected with more than 80% detection rate. This makes it possible to follow the object motion over sequences. In [3] this has been used to learn affordances such as moveability and relate it to object size. And Fig. 10 indicates that this method is capable of extracting the outlines of objects in everyday scenes for future use in service and home robotics.

In future work it is interesting to investigate how the approach in [4] can be extended from learning affordance relations to 2D image blobs to different object shapes and their behaviour when pushed or grasped. This would also exploit the ability to calculate the relative 3D positions of the objects and also of size and orientation under the condition of knowing the mounting point of the camera to the ground plane when using a triangulation.

Presently added is shape tracking using lower-level Gestalts. With such a tracking algorithm there will be no need to observe the objects from a viewpoint where several or all sides are visible. Once a three-dimensional object is detected, it is possible to follow it, e.g., when only a rectangle or a flap of a cube is observable.

Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement No. 215181, CogX and from the European Community's Sixth Framework Programme under grant agreement No. 029427 as part of the Specific Targeted Research Project XPERO.

References

1 Gibson J (1979) The Ecological Approch to Visual Perception. Boston, Hougton Mifflin
2 Zabkar J, Bratko I, Mohan A (2008) Learning Qualitative Models by an Autonomous Robot. 22nd International Workshop on Qualitative Reasoning, 150-157
3 Zabkar J, Bratko I, Jerse G, Prankl J, Schlemmer M (2008) Learning Qualitative Models from Image Sequences. 22th International Workshop on Qualitative Reasoning, 146-149
4 Montesano M, Lopes A, Bernardino J, Santos-Victor J (2008) Learning Object Affordances: From Sensory-Motor Coordination to Imitation. Robotics, IEEE Transactions on [see also Robotics and Automation, IEEE Transactions on]. Vol 24(1):15-26
5 Kuipers B, Beeson P, Modayil J, Provost J. (2006) Boostrap learning of foundational representations. Connection Science 18.2; special issue on Developmental Robotics
6 Modayil J, Kuipers B (2004) Bootstrap Learing for Object Discovery. Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems; 742-747
7 Biederman I (1987) Recognition-by-Components: A Theory of Human Image Understanding. Psychological Review, Vol 94(2):115-147
8 Lowe D G (1987) Three-dimensional object recognition from single two-dimensional images. Artificial Intelligence, Vol 31(3):355-395
9 Dickinson S, Bergevin R, Biederman I, Eklund J et al (1997) Panel report: The potential of geons for generic 3-d object recognition. In Image and Vision Computing, 15(4):277–292
10 Sarkar S, Boyer K L (1994) A computational structure for preattentive perceptual organization: Graphical enumeration and voting methods. IEEE Transactions on System, Man and Cybernetics, Vol 24(2):246–266.
11 Zillich M (2007) Making Sense of Images: Parameter-Free Perceptual Grouping. Ph.D. Dissertation, Technical University of Vienna
12 Zillich M, Matas J (2003) Ellipse detection using efficient grouping of arc segments. In 27th Workshop of the Austrian Association for Pattern Recognition AGM/AAPR, 143-148
13 Carlbom I, Paciorek J (1978) Planar Geometric Projections and Viewing Transformations. Computing Surveys, Vol.10(4):465-502

Basic Object Shape Detection and Tracking Using Perceptual Organization

Andreas Richtsfeld and Markus Vincze

Abstract— **If a robot shall learn object affordances, the task is greatly simplified if visual data is abstracted from pixel data into basic shapes or Gestalts. This paper introduces a method of processing images to abstract basic features and into higher level Gestalts. Perceptual Grouping is formulated as incremental problem to avoid grouping parameters and to obtain anytime processing characteristics. Furthermore we want to present a efficient method to track Gestalts using low-level Gestalts for motion field approximation. The proposed system allows shape detection and tracking of 3D shapes such as cubes, cones and cylinders for robot affordance learning.**

I. INTRODUCTION

Humans can solve even complex tasks such as manipulating or grasping objects by learning and using the relationship of object shape to the intended behavior - also referred to as affordances. Robotics would profit considerably if a robot could learn these affordances [1] and relate them to visual percepts. However, at present visual perception delivers only image-based features. It is the objective of this work to present a method that extracts the basic 3D shape of objects for enabling more realistic robot learning.

There are several paradigms of making robots learn affordances. [2] and [3] present an approach where a mobile robot observes the environment and autonomously learns a prediction model from its actions and observation data. An overhead camera is used to track color blobs of the robot and the objects. Another possibility is to discover object affordances [4]. To track objects and estimate the object behavior, they are also using color blobs and a set of visual features to describe the 2D blob shape. As an example, a circle can roll but a rectangle cannot. Another possibility is learning high-level ontologies, as proposed in [5] and [6], by clustering the point data from a laser range scanner with an occupancy grid map. All these approaches for autonomous learning are dealing with simple vision data, which are determining relative positions of objects and use image-based point or blob features. However, to understand the usage of different objects the capability to abstract shapes from images into 3D shapes is necessary. This is particularly true for object grasping but also holds for navigation, where an abstraction of point data into geometric features reduces complexity, thus enables efficient learning and yields an effective description of affordances.

Hence we propose a system using methods of perceptual organization to estimate basic object shapes in a hierarchical manner such that affordances can link to different abstractions as required. Fig. 1 shows the overall processing schema of the observation system, which will be discussed in the following sections. Perceptual Grouping is used to determine 3D object shapes like cubes, cones, cylinders or balls. As recognition of 3D shapes is dependent on the view on objects, we propose an object tracking and hypothesizing strategy using an approximation of the motion field. Therefore, once a object shape was determined, it is possible to follow 3D objects over image sequences even if objects can only be detected partially in one frame. This aims at offering a generic tool for autonomous learning of robots in their environments.

A. Richtsfeld and M. Vincze are with Automation and Control Institute, Faculty of Electrical Engineering, Vienna University of Technology, 1040 Vienna, Austria {ari,vinzce}@tuwien.ac.at

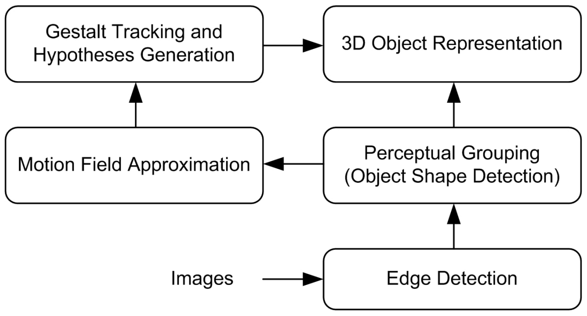

Gestalt Tracking and Hypotheses Generation
3D Object Representation
Motion Field Approximation
Perceptual Grouping (Object Shape Detection)
Images
Edge Detection

Fig. 1. Processing schema of the proposed system.

II. RELATED WORK

Perceptual organization is a well known topic in computer vision and was researched since the Eighties. Seminal works regarding the theory of perceptual recognition of simple objects are [7] and [8]. Biederman [7] proposed that there are non-accidental differences between simple objects, which can be derived through so called geons. [9] discusses the potential of geons in computer vision systems. Perceptual grouping is a bottom-up method to estimate Gestalts, where new Gestalts will be estimated from Gestalts of lower-level Gestalt principles. It can also be seen as geometric grouping, because most of the Gestalt principles implements geometric restrictions.

The main challenge of perceptual grouping is to limit the combinatorial explosion. Indexing and thresholding of less salient hypotheses are normally used to solve this problem [8][10]. Indexing divides the search space for relations between elements into bins and each element will be allocated to a bin depending on the grouping relationship. Further all elements in a bin can be analyzed whether they fulfil the given relation or not. [11] proposes to use indexing in the image space where growing search lines are used to find intersections between Gestalts. With the reduced search space, the problem of combinatorial explosion and the length definition of search lines can be avoided. When using incremental growing search lines, results are no longer dependent on the length definition of search lines, but to the processing time.

[12] investigated the role of perceptual organization in tracking of 2D structures. He pointed out, that extended structures from 2D shapes are more stable, reliable, less susceptible to noise and have more descriptive capability than point features. Therefore it is obvious to use these higher-level Gestalts for the motion field estimation, what allows later easier tracking of higher-level Gestalts.

III. Perceptual Organization

When following the work of [11], search lines will be used to find intersections between Gestalt features. Indexing into the image pixels limits the search space and avoids the problem of comparing all combinatorial possibilities to find connections. Depending on the type of search line, different types of groupings can be found. A dilemma of using search lines is the length definition. When the length of the search lines is too small, some important Gestalts could not be found and on the other side, if they are too long, enhancement of computation time could be provoked because many combinations may occur. This problem can be avoided with incrementally growing search lines, where in every processing circle another search line grows one pixel. When drawing into the indexing image, intersections with existing search lines are detected immediately in the respective bin. The biggest advantage of this method is that any preset parameters or thresholds are now moved into longer and longer processing. The results are no longer dependent on the line length, but to the processing time. The longer the process operates, the more junctions can be found and thus more other Gestalts can be detected. This gives a possibility to evaluate the best possible hypotheses for a limited processing time and on the other side this makes it feasible to obtain results at any time (a quality referred to as anytimeness).

We propose an incremental grouping method as shown in Fig. 2. Triggered by incremental growing search lines, used for Line- and Ellipse-Junctions, each Gestalt creates new hypotheses and delivers them to Gestalts of higher levels. The concept of incremental growing search lines provides the system with incremental processing capabilities. Boxes in the non-shaded area of Fig. 2 are referring to basic geometric features, whereas the boxes in the shaded area indicate Gestalts obtained by incremental processing. The fundamental principles for incremental processing are Line- and Ellipse-Junctions. Each processing circle starts with incrementing one search line of a line or an ellipse. Depending on which search line was chosen, the next hypothesis can be an Ellipse- or a Line-Junction.

In a first processing step the basic features are processed all at once and grouping uses neighborhood constraints. Next, incremental processing of Line- and Ellipse-Junctions starts

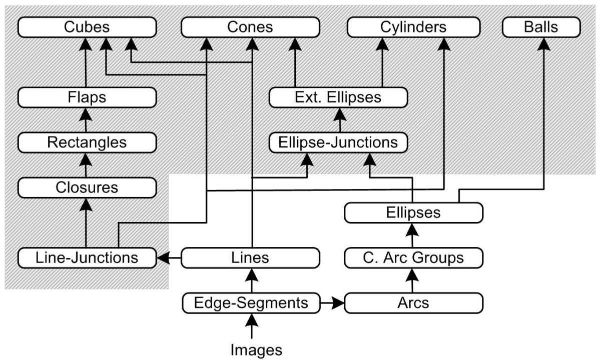


Fig. 2. Perceptual grouping system.

and the higher level Gestalts receive new input incrementally: each cycle extends one search line and all new hypotheses are processed at once to obtain new Gestalt hypotheses. If one of the Gestalts can hypothesize for a new Gestalt one level up the tree, the Gestalt from the next higher level begins to work, and so on. Hence the next junction will not be processed until all higher Gestalts are processed and no more hypotheses can be produced. This functionality of the processing tree guarantees that the best Gestalt hypotheses for the detected junctions are calculated in every processing circle.

[13] and [11] discusses the approach for grouping of the Gestalts and the processing of Gestalt principles in detail. The following list explains the grouping conditions and methods in short terms for completeness:

- Edge-Segments: Each edge segment consists of multiple neighboring edge points. The segments are estimated by canny edge detection with self-adjusting hysteresis thresholds.
- Lines: Lines are built from edge segments, by fitting straight lines to the detected segments using the method by Rosin and West [14].
- Arcs: Circular arcs can be fitted to edge segments, using the RANSAC algorithm. Fitting circles instead of ellipses is more stable and faster and locally approximates ellipses well [15].
- Convex Arc Groups: Arcs will be grouped to convex arc groups to avoid the problem of an exponentially large number of groups. Perceptual grouping based on proximity and good continuation helps to significantly reduce the number of hypotheses.
- Ellipses: In this stage, ellipse hypotheses will be formed from groups of arcs, using the direct least squares B2AC algorithm by [16].
- Line Junctions: Line Junctions are connections between two lines and can be found with intersections of defined search lines. Six different search-lines are defined for each line, one at each end in tangential direction and two in normal direction. Different combinations of intersecting search lines are generating different junctions. We distinguish between Collinearities, L-Junctions and T-Junctions, where T-Junctions could also be interpreted as two L-Junction. For growing search lines, the "smart

grow" algorithm of [11] is used.

- Ellipse Junctions: Ellipse-Junctions are connections between a vertex of an Ellipse and a Line, which can also be found with search lines. The four defined search lines for Ellipses are growing on the main axis of the Ellipses, beginning at the two vertices into both directions. For the growing of the search lines, we are using the same growing algorithm as for the search lines of Line-Junctions.
- Closures: Closures are closed convex contours, built from the Gestalts Lines and Line-Junctions, as proposed in [11]. Whenever the principle will be informed about a new Line-Junction, a new closed convex contour could be detected. To find these contours, Dijkstras algorithm for shortest path search is used, where only paths consisting of non-intersecting Lines, Collinearities and L-junctions of the same turning direction are allowed, thus ensuring simple convex polygon contours.
- Rectangles: Rectangles can be directly derived from Closures by using geometric restrictions, which are given through the description of Rectangles in a perspective view. When considering these perspective projections as shown in [17] and neglecting one vanishing point (one-point projection), it is possible to recognize Rectangles as Trapezoid, whenever a new Closure appears.
- Flaps: A Flap is a geometric figure consisting of two Rectangles which do not overlap and which have one line in common. Detecting Flaps is the intermediate step for detecting Cubes, because two sides of a Cube are always building a Flap in a perspective view of a camera. When seeing a Cube aligned to one side only a Rectangle or Flap is visible, making the Flap an obvious Gestalt element.
- Cubes: An image taken from a camera may show different perspective views of three-dimensional objects. In the case of a cube there may be one, two or three rectangles (in perspective view) visible. The chance that one observed rectangle indicates a cube is small and increases for a flap, but both can occur accidentally. Only when we are able to find three adequate rectangles, we can conclude that the robot observes a cube. Therefore, cubes are built up from flaps (rectangles), lines and junctions.
- Extended Ellipses: Extended Ellipses are Ellipses and Lines, which are assigned through Ellipse-Junctions. Extended Ellipses also assigning all Lines, which are connected to these Lines with a Collinearity.
- Cones: Cones are geometric figures, whose shape consisting of an (partly visible) ellipse, lines and junctions between these components.
- Cylinders: The object shape of a cylinder consists of two (partly visible) ellipses and lines between these ellipses. Therefore Cylinders can be grouped from Extended Ellipses, Lines and Junctions.
- Balls: Balls can be estimated directly from Ellipses, when the shape of the ellipse is nearly a perfect circle.

IV. Gestalt Based Tracking

A fundamental problem of the proposed perceptual grouping system for detection of 3D shapes is the predetermined view to objects for recognition. Thinking on a cube, there is the need to have a view on three different sides to make sure that it is the sought-after three-dimensional object. One or two visible sides are not enough to obtain a high reliability. Only a view from a position where three sides of the cube are visible gives a high significance for observing the object shape of a cube. But this is not general true for all objects, because it does not fit for the other basic object shape of cones, cylinders or balls. The radial symmetry of these objects causes always the same shape from every point around the object.

To solve the problem with the view to objects, we suggest a processing schema with additional processing modules as shown in Fig. 1. After the estimation of Gestalts by perceptual grouping with the rules of the Gestalt principle tree as explained in section III, follows a approximation of the motion field. This aims for the explanation of the motion of object shapes over following images from a video sequence, where the view to objects changes with time. The motion field allows later a reliable tracking of Gestalts and a reliable hypothesizing of higher-level Gestalts from lower-level Gestalts. So once a basic object is detected, it will be possible to recognize it in the next image without a view to the three-dimensional shape.

A. *Motion Field Approximation*

A camera mounted on a mobile robot can be seen as a moving observer in a quasi-stationary environment. When the observer moves, the projected image from the environment will change. For visually-guided navigation the task of the observer is to use this changing image to determine the motion in space. The proposed method is not accurate enough for self localization and mapping (SLAM), but this is not necessary, because we are only interested in a approximation of the motion field to evaluate the results of tracking. When the observing camera is mounted on a robot, which drives on the ground floor, it is possible to constrain the motion to two dimensions for the approximation of the motion field. This simplifies the calculation, because only forward or backward motion and respectively left or right turning is possible.

We calculate the motion field from tracked corner points and use the L- junctions already calculated as corner detectors, rather than running an additional corner detectors such as Harris or KLT. L-Junctions are good candidates, because they appear constant in images and they have strong descriptive features to distinguish between them. Needless to say that L-Junctions can occur accidentally and without relevance to the image content, but a high correctness is not mandatory. It is enough to get the motion vectors of a fraction of L-Junctions to evaluate the motion field stable.

Fig. 3 shows the motion vectors of L-Junctions. The median of all estimated directions can be considered as main direction of the motion field. This helps to prune the possible false assigned L-Junctions. The main direction is also used to distinguish between four different motion cases, shown in Fig. 3, which will be considered for the calculation.

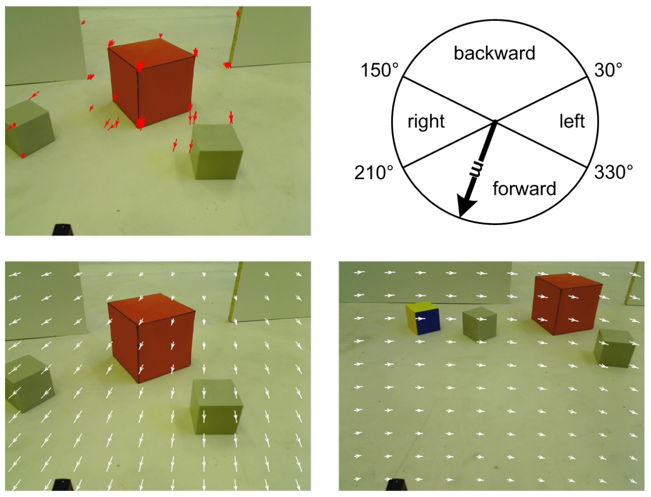


Fig. 3. Motion vectors of L-Junctions (top, left). Defined motion cases (top, right). Approximated motion field for forward motion (bottom, left) and for right turn (bottom, right).

Once the main direction and therewith the motion case is determined, the focus of expansion (FOE) or the center of rotation (COR) can be estimated when calculating all intersection points between different motion vectors, which has been extended to infinity. The FOE can be estimated directly for forward and backward motion, whereas the motion vectors must be rotated 90 degrees in case of left and right motion for the COR calculation. The FOE and COR will be determined as weighted mean value of the intersection points from the motion vectors. In most instances the FOE and COR are outside of the image. The position of the FOE or the COR and the knowledge about the direction of the motion allows a approximation of the motion field as proportional ratio between motion and distance between FOE or COR and the point of the sought-after motion:

$$motion = k * distance \tag{1}$$

To get the weighting factor k, we calculate the mean value of k from the motions and distances of all recovered L-Junctions:

$$k_{avg} = 1/n * \sum_{i=1}^{n} motion_i / distance_i \tag{2}$$

The direction of the motion depends on the position of the FOE/COR and the motion case. It is possible to distinguish between the four cases of movement, where we get following directions for the different movements:

- Forward: Direction against FOE.
- Backward: Direction to FOE.
- Left: Direction to COR + 90 degree.
- Right: Direction to COR - 90 degree.

The left bottom image of Fig. 3 shows the resulting motion field for a forward motion and the right bottom image shows it for a left turn. In both cases the FOE/COR is outside the image.

B. Tracking of Gestalts and Hypothesis Generation

The motion field allows us to predict the motion of all following Gestalts and therefore makes tracing more reliable. Nevertheless prediction of motion is not enough to assign Gestalts from the former image to the next, it is recommended to compare Gestalts by descriptive features to ensure that they match. Once all possible shapes are tracked, the system tries to create hypotheses for objects which could not be tracked as a whole. For the latter the system tries to identify partial objects from lower-level Gestalts, because these still strongly indicate the presence of the whole object. Fig. 4 shows grouping, tracking and hypothesis generation for an example of a turning cube over three images.

The oldest cube on the left side of the image is created by perceptual grouping, whereas the next cube in the middle of the image could not be detected cause of the missing rectangle, which should be on the top of the cube. The cube can be hypothesized using the tracked flap, because it belongs to the cube of the former image. The last cube can not be hypothesized in one step, because only one rectangle was detected. The system hypothesizes first the flap from the single rectangle and in the next step the cube from the hypothesized flap. It is still sure, that the cube is at this position, because the rectangle could be tracked from the former two images. With this method is it now possible to follow a Gestalt, even if it is not possible to detect the whole shape with the perceptual grouping tool.

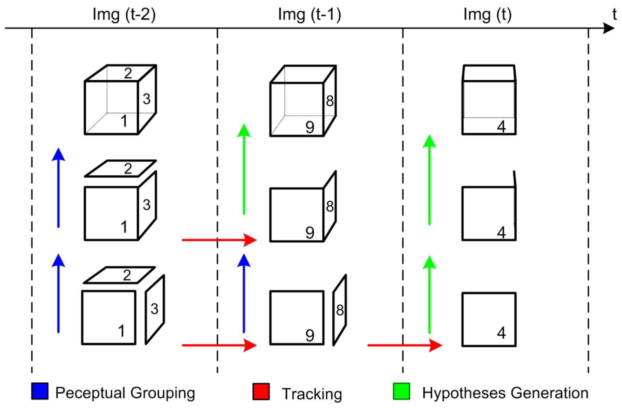


Fig. 4. Hypotheses generation from tracked Gestalts.

C. 3D Object Representation

Using a ground plane constraint enables the estimation of 3D points when object points on the ground plane are known as image points. It is therefore possible to get the position and the size of the basic object shapes in a three dimensional coordinate system. When we are hypothesizing objects after tracking of lower-level Gestalts, we have to recalculate the 3D position and the size of objects dependent on the current position and orientation of the tracked low-level Gestalt. When using the model (e.g. corner points, size, orientation) of the former detected object, it is enough to know two points or a point and a orientation to recalculate the new position and orientation of the object and we can show the model again in the two dimensional image.

V. Experimental Results

The incremental grouping method has been evaluated with a mobile robot moving among simple geometric 3D objects. Fig. 5 shows an example image, with several three-dimensional objects. The picture indicates the typical problem of grouping, namely that shadows or image noise creates spurious features such as lines or arcs. A grouping into higher level Gestalts sometimes accidentally includes a wrong feature though, more often the grouping principle constrain the search to actual higher level Gestalts. With the incremental approach object detection depends on processing time. The first two images present the original and the cluttered edge image. The left image of the second row shows all extended search lines after 468ms and the next three images are showing the results after 184ms, 328ms and 468ms processing time.

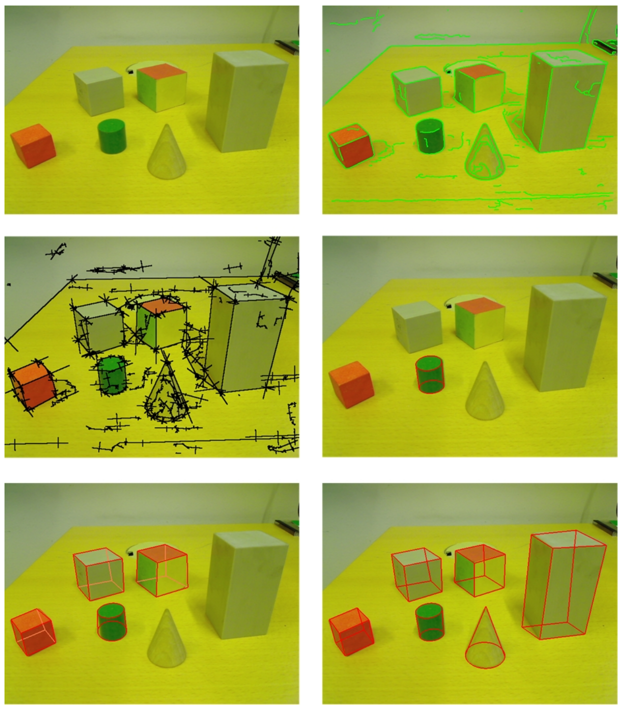


Fig. 5. Perceptual grouping for a single image: edge image, voting image and basic objects after 184ms, 328ms, 468ms.

The left image in Fig. 6 shows two cubes which are tracked over ten images. The tracked cubes are laid over the last image to show the motion of the cubes. The right image shows the calculated motion of the cube center points (red line) and the real motion (cyan) over ten images.

To evaluate the capabilities of the proposed perceptual grouping with the Gestalt based tracking method, we processed a playground scene with different processing times. Tab. I shows the results of runs without tracking and with tracking, using a Intel Core2Duo with 2,5GHz. The processing time for detection with tracking is between 1.5% and 5,0% higher than without tracking, because the non-incremental processing of the motion field and the tracking algorithm needs some more processing time after the perceptual grouping of Gestalts. The whole scene consists of 148 images within 417 cubes to detect. As expected, the detection rate increases with increasing processing time, but also the rate of falsely detected cubes.

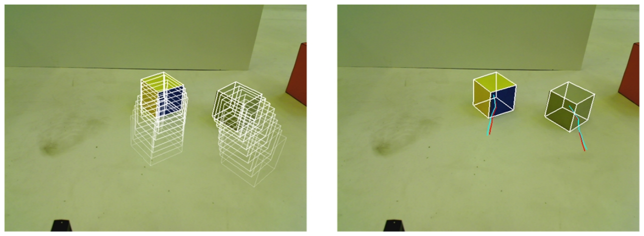


Fig. 6. Tracking of cubes. Deviation of calculated (red) and real motion of a cube (magenta) after 10 tracking steps.

TABLE I

Detection of cubes in a playground scene.

Without Tracking			With Tracking		
Time [ms]	True	False	Time [ms]	True	False
150	211	0	151,66	223	1
200	294	0	203,03	344	5
300	340	1	305,37	383	5
400	359	12	412,8	390	17
600	378	28	630,1	406	31

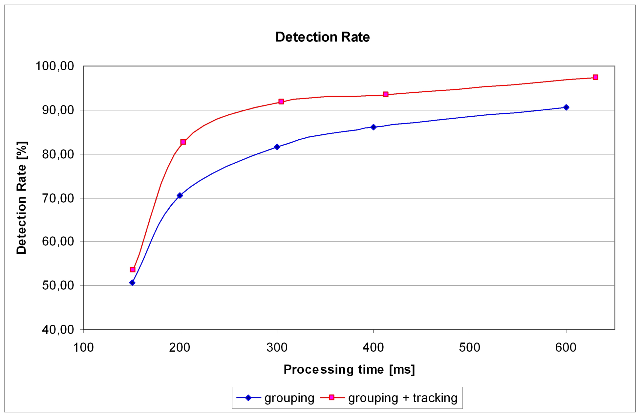


Fig. 7. Detection rate of a playground sequence of 148 images with 417 possible cube detections.

Fig. 7 shows the detection rate graphically over processing time. The blue curve shows the detection rate for processing without tracking, whereas the red presents the detection rate using the tracking method. The second curve shows the advantage of tracking. For processing times around 250ms increases the detection rate about twelve percent, while processing time increases just about two percent. Note that Canny edge detection alone takes approximately 150 ms. So near 150 ms processing time only edges and very few L-junctions and accordingly very few higher-level Gestalts are detected.

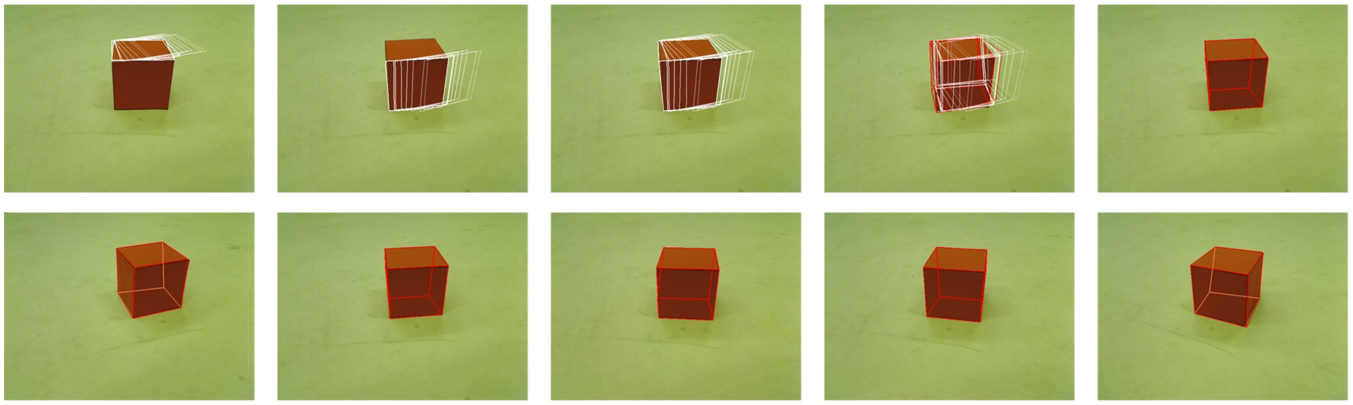


Fig. 8. First row: Tracked (white) and hypothesized (red) rectangles, flaps and cubes from a single image of the sequence. Second row: Five sample images with estimated hypotheses from tracking of lower-level Gestalts, when driving a robot around a cube.

The goal of the motion field approximation was to track cubes if not all three sides of their shapes are visible, which are needed to detect them as three-dimensional objects with the proposed grouping method. Fig. 8 presents a showcase, where a robot drives around a cube with some degenerate views, where only two or a single side of the cube are visible. The first row shows tracked (white) and hypothesized (red) rectangles, flaps and cubes, and the last image of the first row the resulting cube hypothesis. The second row shows that the Gestalt based tracking algorithm allows to follow the cube through the whole scene, which consists of sixteen images, and create hypotheses even if there is only one flap or a single rectangle visible.

VI. Conclusions and Further Work

We presented a efficient method for detection and tracking of basic geometric Gestalts, based on a hierarchical visual grouping system. With the incremental processing approach the problem of having parameters for each Gestalt principle is reduced to the single parameter processing time. A approximation of the motion field makes tracking of Gestalts possible, whereas tracking Gestalts and creating hypothesis allows us to follow basic object shapes even for degenerate views, where not all sides of an object are visible for an observer. The evaluation shows that the proposed methods increase the detection rate without a essential increase of processing time.

In future work it is interesting to investigate how the approach in [4] can be extended from learning affordance relations to 2D image blobs to different object shapes and their behavior when pushed or grasped.

VII. ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement No. 215181, CogX and from the European Community's Sixth Framework Programme under grant agreement No. 029427 as part of the Specific Targeted Research Project XPERO.

References

[1] J. Gibson, *The Ecological Approch to Visual Perception*, Hougton Mifflin, Boston; 1979.

[2] J. Zabkar, I. Bratko, A. Mohan, "Learning Qualitative Models by an Autonomous Robot", *22nd International Workshop on Qualitative Reasoning*, 2008, pp 150-157.

[3] J. Zabkar, I. Bratko, G. Jerse, J. Prankl, M. Schlemmer, "Learning Qualitative Models from Image Sequences", *22th International Workshop on Qualitative Reasoning*, 2008, pp 146-149.

[4] M. Montesano, A. Lopes, J. Bernardino, J. Santos-Victor, "Learning Object Affordances: From Sensory-Motor Coordination to Imitation", *Robotics, IEEE Transactions on [see also Robotics and Automation, IEEE Transactions on]*, Vol 24(1), 2008, pp 15-26.

[5] B. Kuipers, P. Beeson, J. Modayil, J. Provost, "Boostrap learning of foundational representations", *Connection Science 18.2; special issue on Developmental Robotics*, 2006.

[6] J. Modayil, B. Kuipers, "Bootstrap Learing for Object Discovery", *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004, pp 742-747.

[7] I. Biederman, "Recognition-by-Components: A Theory of Human Image Understanding", *Psychological Review*, Vol 94(2), 1987, pp 115-147.

[8] D. G. Lowe, "Three-dimensional object recognition from single two-dimensional images", *Artificial Intelligence*, Vol 31(3), 1987, pp 355-395.

[9] S. Dickinson, R. Bergevin, I. Biederman, J. Eklund, et al, "Panel report: The potential of geons for generic 3-d object recognition", *In Image and Vision Computing*, Vol. 15(4), 1997, pp 277-292.

[10] S. Sarkar, K. L. Boyer, "A computational structure for preattentive perceptual organization: Graphical enumeration and voting methods", *IEEE Transactions on System, Man and Cybernetics*, Vol 24(2), 1994, pp 246-266.

[11] M. Zillich, *Making Sense of Images: Parameter-Free Perceptual Grouping*, Ph.D. Dissertation, Technical University of Vienna, 2007.

[12] S. Sarkar, "Tracking 2D Structures using Perceptual Organizational Principles", Proceedings of the International Symposium on Computer Vision, 1995, pp 283-288.

[13] A. Richtsfeld, M. Vincze, "3D Shape Detection for Mobile Robot Learning", *accepted at German Workshop on Robotics*, 2009.

[14] P. L. Rosin, G.A.W. Westl, "Non-parametric segmentation of curves into variuos representations", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 17(12), 1995, pp 1140-1153.

[15] M. Zillich, J. Matas, "Ellipse detection using efficient grouping of arc segments", *In 27th Workshop of the Austrian Association for Pattern Recognition AGM/AAPR*, 2003, pp 143-148.

[16] A. W. Fitzgibbon, M. Pilu, R. B. Fisher. "Direct least square fitting of ellipses", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 21(5), 1999, pp 476-480.

[17] I. Carlbom, J. Paciorek, "Planar Geometric Projections and Viewing Transformations", *Computing Surveys*, Vol.10(4), 1978, pp 465-502.

KTH Computer Science
and Communication

School of Computer Science and Communication

CVAP - Computational Vision and Active Perception

Elementary Grasping Actions for Grasping Polyflaps

Yasemin Bekiroglu

TRITA-CSC-CV 2009.4 CVAP315

Yasemin Bekiroglu
Elementary Grasping Actions for Grasping Polyflaps

Report number: TRITA-CSC-CV 2009.4 CVAP315
Publication date: July, 2009
E-mail of author: yaseminb@kth.se

Reports can be ordered from:

School of Computer Science and Communication (CSC)
Royal Institute of Technology (KTH)
SE-100 44 Stockholm
SWEDEN

telefax: +46 8 790 09 30
http://www.csc.kth.se/

Elementary Grasping Actions for Grasping Polyflaps

Yasemin Bekiroglu
Centre for Autonomous Systems
Computational Vision and Active Perception Lab
School of Computer Science and Communication
KTH, Stockholm, Sweden
yaseminb@kth.se

15th July 2009

Contents

1 Introduction **2**

2 Related work **3**

3 System Setup **5**

4 Elementary Grasping Actions for Polyflaps **7**
4.1 EGA 1 8
4.2 EGA 2 9
4.3 EGA 3 13

5 Using EGAs **15**

6 Discussion and Conclusion **16**

Abstract

The ability to manipulate novel objects detected in the environment and to predict their behaviour after a certain action is applied to them is important for a robot that can extend its own abilities. The work presented in this report investigates the interplay between perception and action in the framework of object manipulation based on visual input. The work concentrates on the development of generalisable and extensible manipulation strategies for two and three fingered robot hands.

In particular, we investigate the necessary conditions for grasping of objects using basic geometric representations - polyflaps. We demonstrate how polyflaps can be grasped by using a parallel jaw gripper. A set of potential grasps is generated based on an analytical parametrisation. Generated grasps are ordered based on a metric that considers the pose state space of a given polyflap. The output of the method is either the most stable grasping action or the desired pose that can be integrated with a polyflap pushing mechanism to achieve the most stable grasping action.

1 Introduction

Robots that move about in natural environments need the ability to detect and manipulate known and novel objects. Learning about objects and their affordances is also central for understanding the context and execution of various delivery tasks. Given an object, the embodiment of the robot and the task itself, the amount of potential grasps that can be applied to that object is huge. How to choose a feasible grasp and, at the same time, deal with incomplete information about e.g. the object geometry is not a trivial task to program. Although humans master this skill easily, no suitable representations of the whole process have yet been proposed in the neuroscientific literature. It is thus difficult to develop artificial control systems that can demonstrate human grasping behaviour.

Although there exist robotic hands with several degrees of freedom, most common solutions for mobile robots have only two or three fingers. With only two fingers and a single degree of freedom to be controlled, the manipulation capabilities of a robot are quite constrained. Since the shape and size of object in natural environments varies a lot, determining a good grasping strategy is a complex task. The goal in this project is for the robots to be able to manipulate novel objects in natural environments and there are potentially infinitely many grasping strategies that need to be considered.

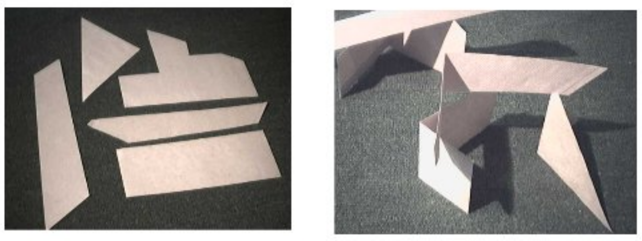


Figure 1: Polygons to form polyflaps and sample polyflaps [1].

With the aim of reducing the complexity of the problem, minimal models for objects should be determined which requires as few assumption as possible. Humans can perceive and act on unfamiliar and unrecognisable objects without the need of identifying or describing spatial structures in order to perceive them. Therefore, for a robot to be able to mimic this ability, the choice of the representation of objects is crucial. For this purpose, polyflaps [2] have been considered in this work. A polyflap is a polygon (concave or convex) cut out of a flat material and folded once to produce a 3D object. Figure 1 shows a few sample polyflaps. Based on the shape of the original polygon and how it is folded, different polyflaps are created and thus there is a large number of possible configurations of a single polyflap. The domain of polyflaps was chosen since they may be easily fitted to a variety of everyday objects, [1]. We can also define polyflap grasping strategies for two and three fingered robot hands.

In general, grasping cannot simply be achieved by opening the gripper, approaching an object and then closing the fingers around the object. Grasping requires an understanding of the spatial configuration of the environment to determine the approach direction, the orientation of the gripper and the gripping location on the object. Therefore, when planning a suitable grasping strategy, the robot needs prior knowledge about its own embodiment and scene it is supposed to act in.

In some cases, the robot will not be able to achieve a position and orientation in the environment from which a grasp should be applied. This may be due to the obstacles in the scene or if the number of degrees of freedom of the robot is low. However, the robot may still be able to change the pose (position and orientation) of the object through pushing. Thus, pushing-for-grasping can change the state of an object from an ungraspable to a graspable pose. This is a natural behaviour in humans to: if we want to grasp a cup by the handle and the handle is not directly in reach of our hand, we will rotate the cup in a position from which a suitable handle-grasp can be performed. Thus, moving an object from “out of reach” to “in reach” is an additional requirement closely related to grasping. Pushing an object may also be helpful when the sensory input cannot deliver a useful model of the object to plan a good grasp. The object may then be pushed to a new pose to afford new views and provide more sensory information about the object.

In order to specify a better pose in terms of grasping, there are two problems to be solved. The first one is how to quantify a pose according to the grasp quality and the other is the need of a method to specify the desired pose as a bounded region in the pose state space that enables a performable grasp. A suitable approach is to categorise object poses and then choosing among them. When a model for an object is given, a choice among possible poses needs to be made that enables pushing to a suitable grasping pose. Thus, a pose needs to be quantified according to its “graspability”.

We have developed a method for generating suitable grasps using a polyflap representation that relies on analytic evaluation of grasp quality. Good grasps related to each pose are listed and sorted according to the chosen quality criteria. After determining which potential poses afford good grasps, the one with the best grasp is chosen. Nearness with respect to the current pose or being achievable with less pushes can also influence the decision.

This report is structured as follows. In Section 2 we outline some of the related work. In Section 3, we present the initial implementation of the grasping system. In Section 4 we describe the idea of elementary grasping actions for polyflaps and show their use in Section 5. Finally, we provide a discussion and conclusions in Section 6.

2 Related work

In robotic object grasping there has been a lot of effort during the past few decades. It has been widely recognised that high-level task-related grasp planning is difficult due to the large search space resulting from all possible hand configurations, grasp types, and object properties that occur in realistic settings.

Early work on grasping considered generation of force-closure grasps: grasps resisting any arbitrary force and/or moment that may act on the object externally. A good discussion and review on force-closure grasp can be found in [3] and [4]. Necessary and sufficient conditions for force-closure grasps can be defined ([5]). [6], [7], [8] have identified properties that a force-closure grasp must possess in order to be able to perform everyday tasks similar to those performed by human hands.

[9] extracts four mutually independent properties for grasp generation: dexterity (how should grasping fingers be configured), equilibrium (how hard to squeeze the grasped object), stability (how to remain unaffected by external disturbances), dynamic behaviour (how soft a grasp should be for a given task). During task execution, the fingers should be controlled in a way that the grasped object possesses these four essential properties. Here, it is required to have methods to compute finger parameters such as positions and forces of fingertips and joints. Such algorithms are essential parts of robot hand control and are called grasp synthesis algorithms. Synthesis means determining the required finger properties in order for the grasp to have desired properties.

Numerous approaches have been proposed for the problem of automatic grasp synthesis. The nature of human grasps inspired many works [10], [11]. For the study of robotic grasping and manipulation, works in [12], [9], [13], [14] provide a good starting point.

Approaches considering known objects have a detailed object model in 2D or 3D. They aim to estimate object's pose and retrieve a suitable grasp ([15], [16], [17]). There are also examples of generation of grasp databases. [18] treats grasping as a shape matching problem where a database of grasp examples are used and given a model of a new object, shape features of the object are compared to shape features of hand poses in examples to identify a grasp. The work relies on the availability of a 3D object model.

Approaches considering unknown objects represent the shape of an unknown object and try to reduce the potential grasps. Related to our work, several authors have investigated different geometrical representations for object shape. In [19], the shape of an unknown object is approximated by box decomposition and used to choose feasible grasps. The simple geometry of a box helps to reduce the number of potential grasps. Also, in [20] and [21], a similar approach is proposed which contains fitting superquadrics to 3D data and then grasping a primitive shape is performed. In [22], grasping of unknown objects is considered. Vision based object segmentation and shape extraction are demonstrated and integrated with four elementary grasping actions. An interesting approach for grasping unknown objects is to use genetic algorithms. In [23], tries to solve the problem of automatic grasp synthesis of unknown planar objects by means of a genetic algorithm and objects to be grasped are represented as superellipses, a family of deformable 2D parametric functions. The space of possible grasp configurations is analysed using the genetic algorithm in which resulting grasp optimises several predefined criteria. Here, a neural network is trained on a set collected off-line using genetic algorithms. [24] proposes an evolutionary computation method to compute force-closure grasps from surface points. The method searches for grasping configurations without prior knowledge of objects geometry. The object is presented as set of points.

There are also approaches trying to use grasp experience about certain objects for grasping similar objects. The assumption here is that new objects similar in terms of shape, colour or texture can be grasped in a similar way. Since in practice, it is difficult to extract full 3D geometry of the object very accurately, most experience based approaches that have been demonstrated for real scenes rely on 2D data. [25] proposed a system that detects a point on the object to grasp directly as a function of its image. The method uses machine learning on labelled images of different objects and the classification is based on the features containing colour, texture and edge information extracted from the image.

Lots of work on grasping has also concentrated on generation of planar grasps. The geometry of force-closure grasps has been studied in [26] and results on polygonal planar objects are presented. [27] presents an algorithm for computing force-closure two-finger grasp on curved

2D objects. [28] and [29] show an algorithm for force-closure three-finger grasps for polygonal objects. Other work that has a similar approach are [30], [31], [32], [33], [34], [35]. In [36] and [37], the quality of 2D grasps are evaluated by several criteria. [38] propose a grasp criterion that yields the best grasping position for a robot hand with two soft fingers for planar objects. [39] defines candidate grips for a parallel-jaw gripper that are resistant to slipping and torque about the object's centre of mass and orders them according to the defined criteria. In our work, this method is adapted to be applicable on polyflaps.

3 System Setup

In order to demonstrate the use of the grasp generation system on real objects, a vision system that delivers the pose of the detected polyflap models is described below. The vision system relies on stereo images of the scene and constructs the 3D models of the detected objects. The object is represented by the list of vertices of the surfaces and the pose. The vision system is able to detect polyflaps with convex surfaces and track the detected model. The polyflap models are then reconstructed using the simulator GraspIt, [40] to generate the grasps chosen for the models as shown in Figure 2 and Figure 3. The next section describes the definition of grasps for a two-fingered gripper which we name Elementary Grasping Actions (EGAs). EGAs are crafted for grasping of polyflaps but generalise to objects that have planar regions. In our work we consider three EGAs. The EGAs are formed by the approach vector and contact points. From the determined contact points the orientation vector and the position of the gripper are then derived.

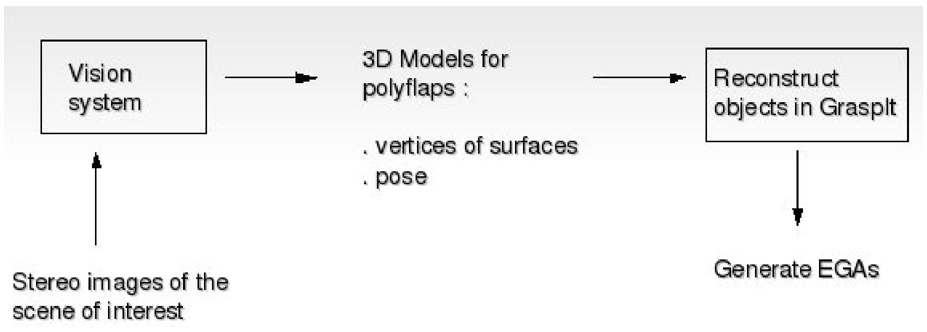

Vision system
3D Models for polyflaps :
. vertices of surfaces
. pose
Reconstruct objects in GraspIt
Stereo images of the scene of interest
Generate EGAs

Figure 2: System's components.

The EGAs defined for a two-fingered, parallel jaw gripper are used with a three-fingered hand based on the idea of virtual fingers, [41]. We do not use the full dexterity of the three-fingered hand but use two fingers in unison to act as a parallel jaw gripper. The reason for this is the special kinematics of the three-fingered Barrett hand since one of the fingers is static and only a spread angle between two other fingers can be actively controlled. The procedure uses the information obtained for the two-finger grasps, which is the locations of the grasp contact points. One of the contact points is selected to act as a thumb and the other is the virtual finger of the three-fingered hand. The real contacts are placed symmetrically on both sides of the virtual contact. This also requires that there is enough space in the grasp region around the virtual contact point to place the two fingers. Figure 4 shows the idea of virtual fingers and Figure 5 and 6 show some example grasps generated on polyflaps with a two-fingered gripper and the Barrett hand.

The position of the contact points is defined by the pose of the polyflap and grasp quality requirements. Since the grippers are attached on a robot arm, the position of the hand needs

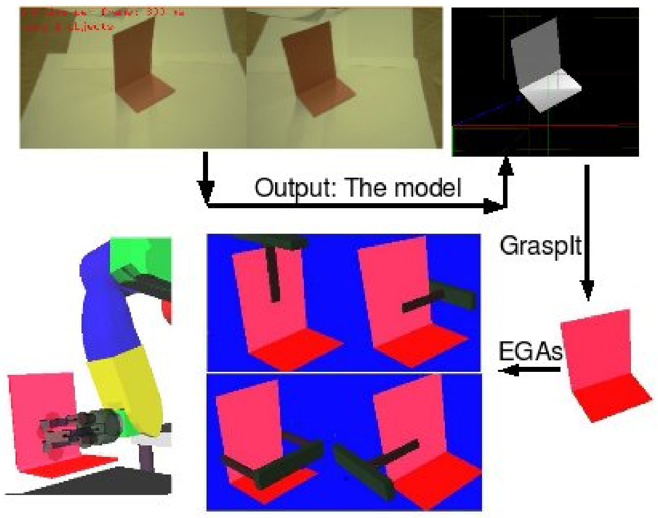


Figure 3: Demonstration of the grasping system with the example of the visual feedback.

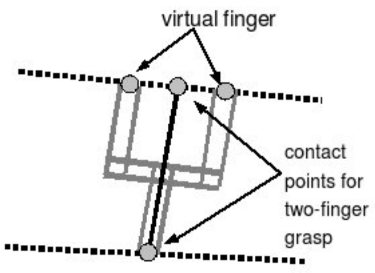


Figure 4: Mapping between two- and three-finger grasps using virtual fingers.

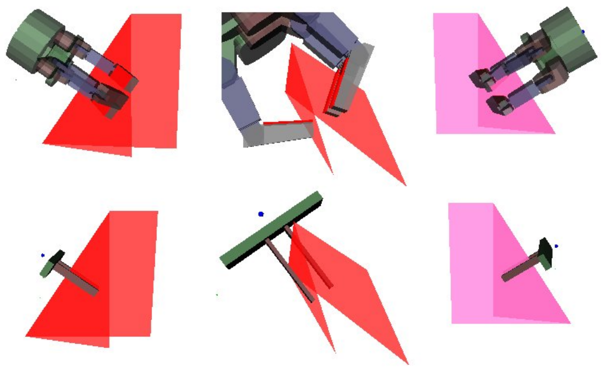


Figure 5: Example polyflap grasps using the Barrett hand and a parallel jaw gripper.

to be estimated based on the position of the contact points. The position of the wrist for a two-fingered hand is easy to determine since the fingers of the hand are made of a single link. The position of the wrist for the Barrett hand is defined through a simple geometric estimation based on the finger angles.

We start by estimating the opening between the opposing fingers of the Barrett hand. This

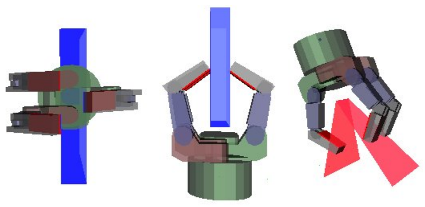


Figure 6: A few examples of Barrett preshapes as a two-fingered gripper.

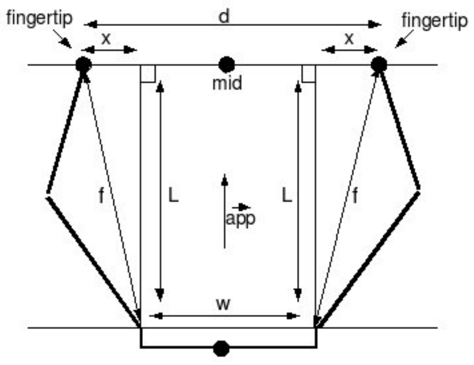


Figure 7: Finding wrist position.

is based on the Euclidean distance d between the two original contact points. As seen from Figure 7, the length x is obtained by subtracting the wrist width w from the distance between the two contact points ($x = (d - w)/2$). If the distance between the fingertips is less than the width of the wrist, then the distance is $x = (w - d)/2$. Since the link lengths and the angle between them are known, f and consequently the distance L are also known. The Barrett hand is then placed at that distance from the middle point of the two contact points along the chosen approach direction. In Figure 6, the Barrett hand grasps two contact points on two parallel surfaces using this approach.

4 Elementary Grasping Actions for Polyflaps

To be able to define qualified grasps, several conditions have to be satisfied. The gripper should not cause the object to slide due to a torque when it closes its fingers around a polyflap. The evaluation is based on the concept of friction cones and makes use of angles between the normals to the object's contour at the grasping points and the grasping line. As stated by [26], force closure with two friction contact points is achieved when the grasping line lies inside both friction cones. The grip axis, the line joining the contact points, must lie in the friction cones at each contact edge in order to prevent the slippage.

The friction cone is defined by the half angle $arctan(\mu)$, where μ is the coefficient of friction. We consider the model of contact with friction. According to Coulomb friction model, the directions of forces exerted on the surface by the fingers should not differ from the normal direction for an angle larger than $arctan(\mu)$ to prevent finger slippage, and the value of coefficient depends on the materials of both the finger and the object surface. The directions of forces must meet this condition regardless of their magnitude.

The objects considered in this work are assumed homogeneous so the friction coefficient is the same on all surfaces. The grip should have minimal dependence on friction, so the grip axis is aligned as close as possible to the corresponding normal of the each contact. The grip should minimise the amount of torque when lifting the object thus the length of the moment arm with respect to the centre of the mass of the object needs to be minimised.

Different polyflaps afford different types of grasps considering their geometry. As mentioned previously, in our work we consider three EGAs. The EGAs are formed by the approach vector and contact points. From the determined contact points the orientation vector and the position of the gripper are derived.

EGA_1 is defined to grasp a surface as close as possible to the centre of mass of the object. EGA_2 and EGA_3 are very similar to each other. EGA_2 is defined for one surface, while EGA_3 is considering two surfaces. EGA_3 can be applied with different approach vectors to make it more applicable. The main motivation for choosing these grasps is that they represent the simplest possible two fingered grasps humans commonly use.

The EGAs are parameterised by the final pose and gripper configuration which can easily be obtained from the desired fingertip positions. For the simple parallel jaw gripper, an EGA will thus be defined by seven parameters: EGA$(x, y, z, \gamma, \beta, \alpha, \delta)$ where $p = [x, y, z]$ is the position of the gripper "centre" according to Figure 8; γ, β, α are the roll, pitch and yaw angles of the vector $\vec{n}$ (in order to achieve the desired approach and orientation vector); and δ is the gripper configuration, see Figure 8. Note that the gripper "centre" is placed in the middle of the gripper.

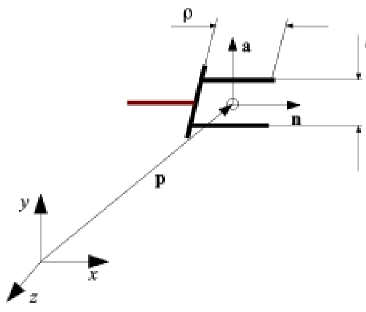


Figure 8: Parameterisation of grasps using the approach vector.

In the following sections, we describe each of the EGAs in more detail.

4.1 EGA 1

The EGA_1 is defined in order to grasp a given polyflap using only one surface and one edge. The aim is to grasp the object from a point which is as close as possible to the centre of mass of the object. That point should be reached in order to minimise the amount of torque. In Figure 9, a few example grasps are shown where the chosen contact point on the surface is the closest point to the centre of mass of the polyflap.

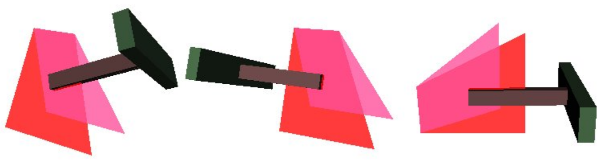


Figure 9: Example of EGA_1.

The orientation of the hand is parallel to the surface normal and the approach vector is derived as following. After determining possible edges that can be grasped, for each of them the approach direction is found based on the chosen point to be grasped. As seen in Figure 10, $d1$ and $d2$ are the end points of the chosen edge and the vector $\vec{A}$ is projected on the unit vector $\vec{B}$ to reach point d which is used to form the approach vector $\vec{APP}$ estimated as following:

$$\begin{aligned} \vec{A} &= (c - d_1) \\ \vec{B} &= \frac{(d_2 - d_1)}{\|(d_2 - d_1)\|} \end{aligned} \tag{1}$$

Given the distance between d and d_1 as $dist = \vec{A} \cdot \vec{B}$, the approach and orientation vectors are estimated as:

$$\begin{aligned} d &= \vec{B} * dist + d_1 \\ \vec{APP} &= \frac{(c - d)}{\|(c - d)\|} \\ \vec{ORI} &= \vec{N} \end{aligned} \tag{2}$$

where $\vec{N}$ represents the surface normal.

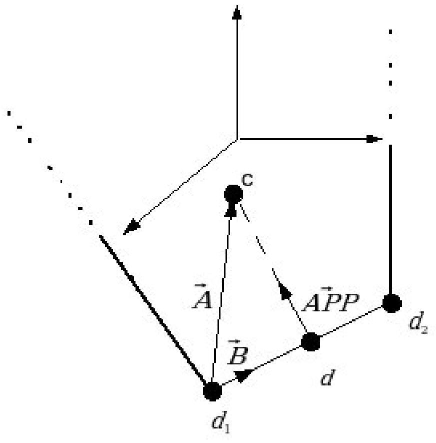


Figure 10: Approach vector for EGA_1.

If the aimed contact point is not reachable by the fingers due to the fact that the distance between the outer edge and the point c is bigger than the finger length, then the reachable point along the approach direction is chosen for grasping.

4.2 EGA 2

The EGA_2, shown in Figure 11, is defined in order to grasp a given polyflap from one surface using two surface edges. The angle between normal to the edge and the grip axis should be less than or equal to friction angle $arctan(\mu)$. The aim is to minimise both angles for two edges at the same time, to have minimum dependence on friction. In order to achieve a balance for minimising the angles at the same time, the grip axis should be perpendicular to the bisector of the two edges. In other words, the angle between the grip axis and the normal to the bisector should be zero in the optimal case. In order to achieve a good orientation for the hand, the grasp axis is chosen as close as possible to the normal of the bisector of two contact edges. If they are parallel, the orientation perpendicular to both edges is the best choice.

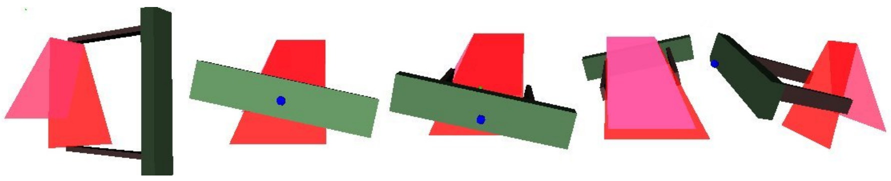


Figure 11: Example of EGA_2.

In summary, the grip axis should be aligned as close as possible to the normal of the each contact edge in order to have minimal dependence on friction. A good candidate is therefore parallel to the normal of the bisector of contact edges which provides minimal dependence on friction as shown in Figure 12. In that way, both angles between the grip axis and the edge normals are minimised.

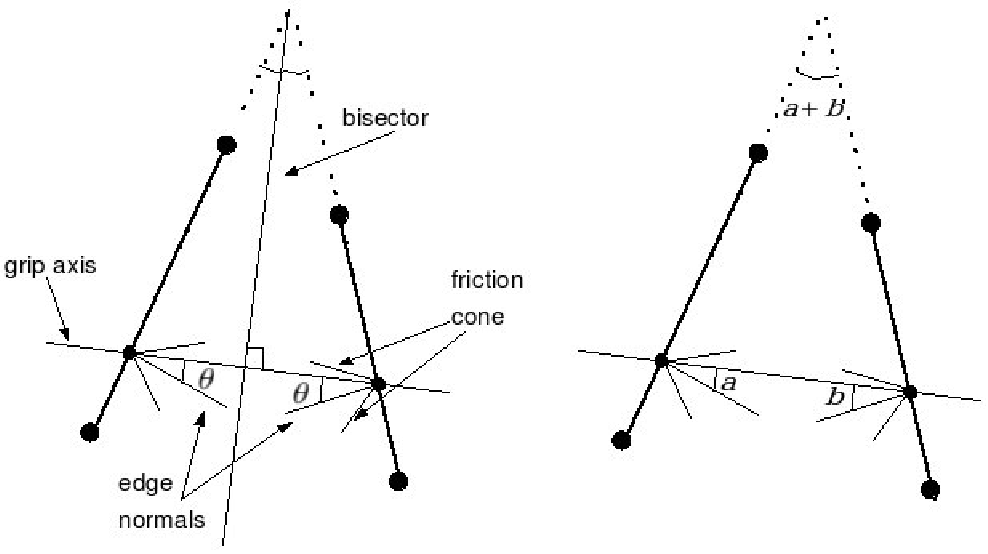


Figure 12: Definition of grip axis for EGA_2.

The grip should also minimise the amount of torque when lifting the object. Thus, the length of the moment arm with respect to the centre of the mass should be minimised. The grip axis should thus go through the centre of the mass of the object and it should be perpendicular to the bisector of the contact edges as shown in Figure 13 (left). If the line that passes through the centre of the mass is perpendicular to the bisector, but it fails to hit one or both edges then we must adjust the grasp to meet all criteria. To be able to have the grip axis within the friction cones, the angle between the edge normal and the grip axis should be less than or equal to $arctan(\mu)$ for both contacts. As seen in Figure 12, the angles a and b should satisfy this condition by $a <= arctan(\mu)$ and $b <= arctan(\mu)$. Therefore the angle between the two edges $a + b$ should satisfy the condition $a + b <= 2 * arctan(\mu)$, otherwise the grip axis is no longer within the friction cones.

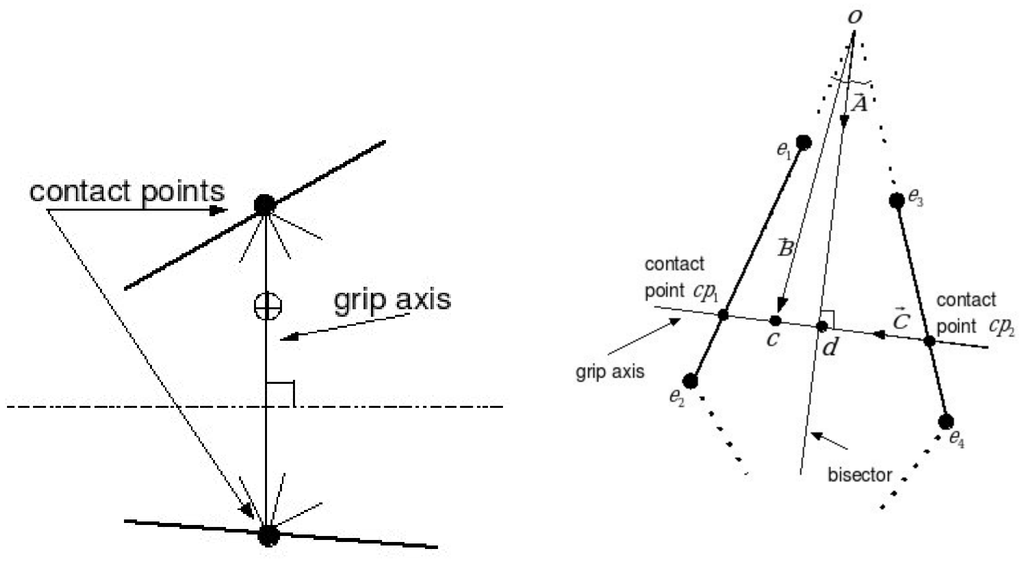


Figure 13: The best grip axis (left) and contact points for EGA_2 (right).

The aim is to achieve a grip axis which is both as close as possible to the centre of mass of the object and has minimum angle with both edge normals at the same time. Therefore, the contact points for EGA_2 are determined by considering the centre of mass of the polyflap and the best possible angle with the edge normals. If the angle (α) between two edges is less than or equal to $2 * arctan(\mu)$, then for this pair the contact points are calculated as following.

As seen in Figure 13 (right), the point o is the intersection point of the chosen edge pair. By using o, the point d on the intersection of grip axis and the bisector is obtained. The point c on the surface is the closest point to the centre of mass of the object and the grip axis goes through this point. The definitions are as follows:

$$\begin{aligned}
\vec{u_1} &= \frac{(e_1 - o)}{\|(e_1 - o)\|} \\
\vec{u_2} &= \frac{(e_3 - o)}{\|(e_3 - o)\|} \\
\alpha &= acos(\vec{u_1} \cdot \vec{u_2}) \\
\vec{A} &= \frac{(\vec{u_1} + \vec{u_2})}{\|(\vec{u_1} + \vec{u_2})\|} \\
\vec{C} &= \frac{(\vec{u_1} - \vec{u_2})}{\|(\vec{u_1} - \vec{u_2})\|} \\
d &= (\vec{A} \cdot \vec{B}) * \vec{A} + o
\end{aligned} \tag{3}$$

The distance between cp_1 and d is denoted x:

$$\begin{aligned}
x &= (A \cdot B) * \tan(\frac{\alpha}{2}) \\
cp_1 &= \vec{C} * x + d \\
cp_2 &= -\vec{C} * x + d
\end{aligned} \tag{4}$$

The orientation vector for the hand is obtained by using the two contact points:

$$\vec{ORI} = \frac{(cp_1 - cp_2)}{\|(cp_1 - cp_2)\|} \tag{5}$$

and the approach vector is simply $\vec{APP} = -surface\_normal$. It can also be seen that, after finding a point on the desired grip axis like d by using any point on the surface as o, the contact points are the intersections of the two contact edges with the line passing through d and c.

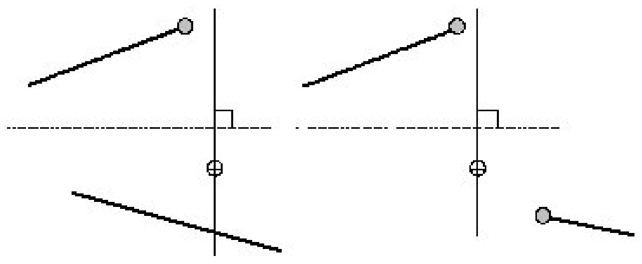


Figure 14: Two examples where the best grip axis does not intersect the chosen edges.

When the best grip axis, namely the optimum grasp which has the minimum torque and friction, can not be applied, then the grip axis needs to be modified satisfying the requirement that the grip axis lies within the friction cone of each contact edge by minimising friction and torque. The change to the grip axis can be performed in two ways: by minimising either frictional dependence or torque. To minimise friction, the grip axis should have a suitable angle with both contact edges at the same time. Since it is perpendicular to the bisector of the edges in the optimal case, then the aim is to have the minimum deviation from that case. To minimise the torque, the grip axis should be as close as possible to the centre of the mass of the object. To achieve this, the contact points are searched in the neighbourhood of the optimum contact points. The result of the search is then added to the list of possible grasps. The procedure for each of the cases is as follows:

- Minimising friction: We start by finding i) critical edge(s) not intersected by the line perpendicular to the bisector through the centre of the mass and ii) critical point(s) as the end point of the critical edge nearest to that line, as shown in Figure 14. We then need to check for a grip axis parallel to the normal of the bisector through each critical point. If none exists, then the best grip has one grip point at a critical point and the other at one of the endpoints of the other edge. The one closest to the normal is chosen and it should be in friction cones of edges as shown in Figure 15.

- Minimising torque: If there is no friction minimising grip then there is no torque minimising grip. Otherwise, we need to check for a grip axis through both the com and a critical point similarly, to find the closest axis to the centre of the mass. If the axis is not within the friction cones then perturb it to have a direction that satisfies the friction criteria as shown in Figure 16.

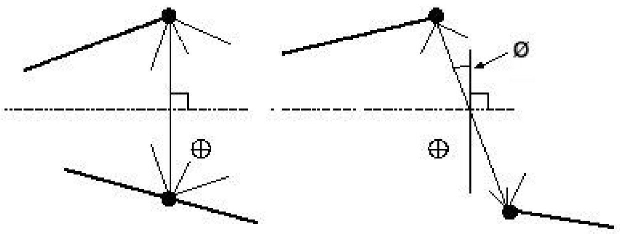


Figure 15: Minimising Friction: in the first case, the grip axis is normal to the bisector; in the second, minimum deviation from normal is achieved.

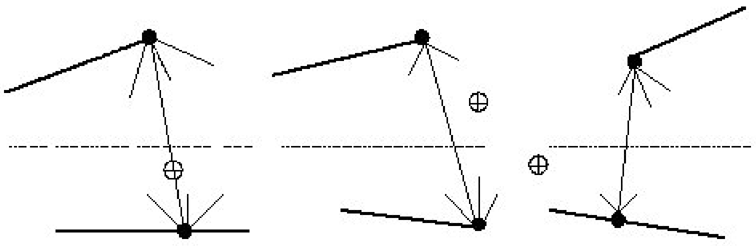


Figure 16: Minimising Torque: in the first case, the grip axis goes through com; in the second, minimum distance to com is achieved and in the third, the grip axis is perturbed to be in the friction cones.

4.3 EGA 3

EGA_3 can be applied in three ways where the way the contact points are calculated is the same but the approach vectors or the way fingers get in contact with the object may differ as explained below. These are denoted EGA_{3a}, EGA_{3b} and EGA_{3c}, respectively.

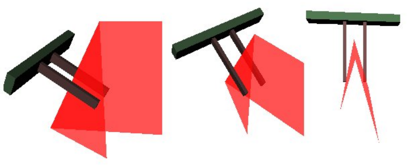


Figure 17: Example of EGA_{3a}.

The contact points are derived similarly to EGA_2 and some examples are shown in Figure 17, Figure 18, Figure 19 and Figure 20. The grip axis is formed by considering the centre of mass of the polyflap and the best possible angle with the surfaces. The approach vector $\vec{APP}$ for

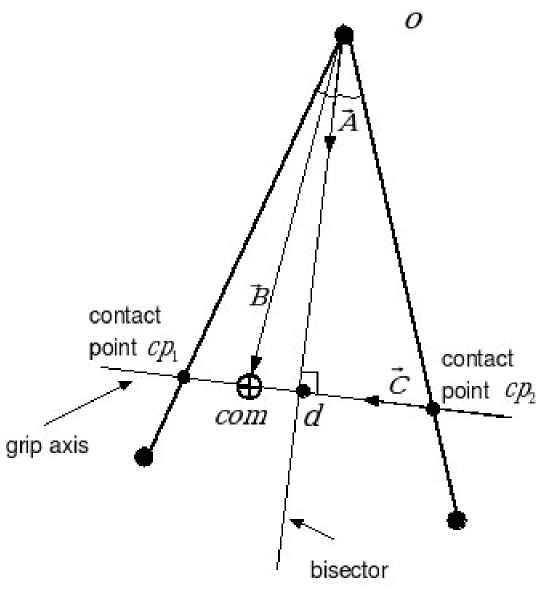


Figure 18: Contact points for EGA_3.

EGA_{3a} is derived from the surface normals ($\vec{N_1}$ and $\vec{N_2}$).

$$A\vec{P}P = \frac{-(\vec{N_1} + \vec{N_2})}{\|(\vec{N_1} + \vec{N_2})\|} = \vec{A} \tag{6}$$

This grasp can be applied if the angle $(180 - \alpha)$ between surfaces is less than or equal to $2 * \arctan(\mu)$:

$$\alpha = acos(\vec{N_1} \cdot \vec{N_2}) \tag{7}$$

The point o lies on the common line of two surfaces and is chosen as the closest point to the centre of mass of the object. After choosing o, similar calculations to EGA_2 are performed in order to find contact points on two surfaces (Figure 18).

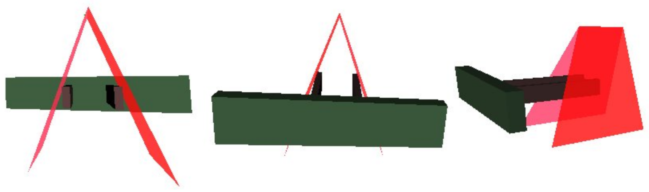


Figure 19: Example of EGA_{3b}.

EGA_{3b} is considered when the common line is parallel to the planar surface on which the polyflap stands and the distance between contact points is big enough to place the fingers. It can be preferred when the orientation calculated for the hand is parallel to the planar obstacle surface. The approach vector is computed in the same way as in EGA_{3c}, and it is parallel to the common line between the two surfaces and can also be defined as:

$$A\vec{P}P \parallel \frac{(\vec{N_1} * \vec{N_2})}{\|(\vec{N_1} * \vec{N_2})\|} \tag{8}$$

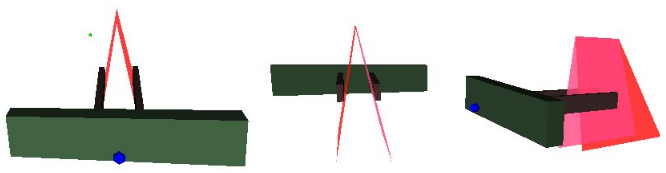


Figure 20: Example of EGA_{3c}.

The fingers are opened until contacts are obtained. The fingertip positions should be at the centre of mass of the object at least or beyond that to be able to perform this type of grasp.

The only difference between EGA_{3a} and EGA_{3c} is the approach vector, since one of them can be easier to be reached than the other. If the gripper can not be opened enough to grasp according to the chosen grip axis then the grip axis is moved along the two contact edges without changing its orientation for EGA_2 and EGA_3. Obviously, for EGA_2, when parallel edges are to be grasped and the fingers can not be opened enough, the grasp is eliminated, since moving the axis does not change the required finger opening.

5 Using EGAs

A polyflap can be in three discrete qualitative states: i) one surface down, ii) one edge next to the common line from each surface down and iii) one edge from each surface other than the neighbour edges to the common edge down, see Figure 21. In case a much better grasp is applicable in a qualitative state other than the current one, in order to increase grasp quality, a mechanism that plans pushes to move the polyflap from the current state to the desired one can be triggered. Pushing may also help when a chosen grasp is good enough but not reachable for the hand.

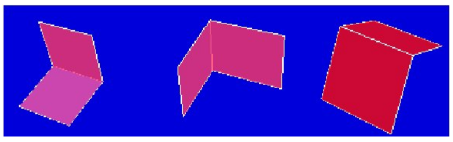


Figure 21: Qualitative pose states of a polyflap.

For each state, all possible EGAs are generated and compared according to two criteria. To achieve this, edges in contact with the resting surface are marked as blocked and only reachable parts of the polyflap are considered. For the polyflap to stand on the considered edges, those edges should be lying on the same plane and also polyflap should be at rest (in a state of equilibrium). In the calculations, weights are proportional to surface areas, since we consider objects for which weight is evenly distributed.

The quality measures are as follows. The grip axis should be as close as possible to the centre of mass of the polyflap and the angle between the normal at the contact and the grip axis should be as close as possible to 0. Therefore, the values for the measure should be as close as possible to 0 and these are normalised into [0-1] accordingly. Thus, we can define two measures accordingly:

$$DTC = distance(grip\_axis, com) \tag{9}$$

where DTC denotes the distance to the centre of mass of the polyflap. Then

$$\begin{aligned} b &= angle(grip\_axis, corresponding\_normal) \\ Angle &= \frac{(b1+b2)}{2} \end{aligned} \tag{10}$$

where b is the angle between the normal and the grip axis at the contact point. The normal is the surface normal for EGA_3 and EGA_1, and it is the normal to the edge for EGA_2.

The orderings with respect to the distance to com and the angle are simply combined like in the following:

$$Quality = \frac{(DTC + Angle)}{2} \tag{11}$$

For instance, the polyflap below (Figure 22) should be pushed to move to one of the listed states where EGA_1, EGA_3a and EGA_3c provide the best quality in three different qualitative states. The selection can favour the one requiring the least effort for pushing. The quality measures calculated for this polyflap and all possible EGAs are listed in Figure 23 and Figure 24.

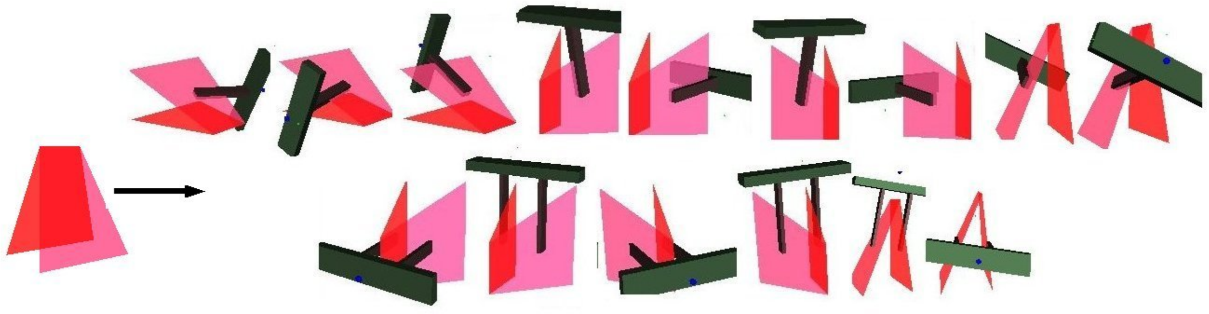
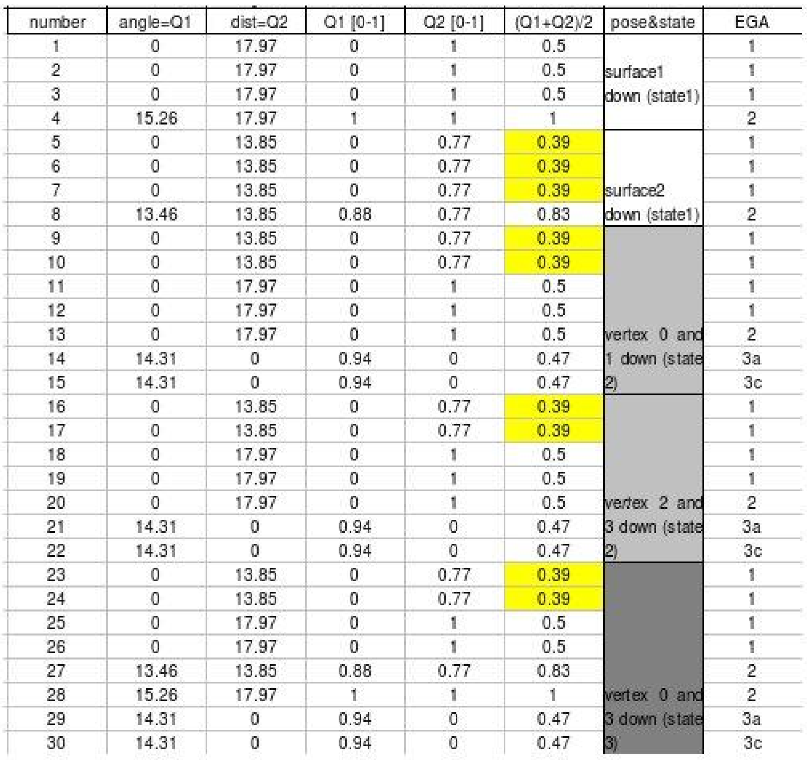


Figure 22: The best EGAs for the given polyflap.

6 Discussion and Conclusion

In this work, EGAs for single polyflaps are studied. Using the models delivered by the vision system for detected polyflaps in the scene, polyflaps are reconstructed and the corresponding EGAs are generated. These are then evaluated using two quality measures. The polyflaps are evaluated considering all possible states and better alternatives to the current state in terms of making it reachable or more graspable can be suggested. Realising that the present situation does not afford a desired grasp (or maybe view) means the system is able to understand the existence of a need for a change, and take steps to turn (extend) the current situation into a new one that affords the desired grasps. Additionally, according to seeing world composed of polyflaps, the system knows what kind of objects (polyflaps) can be grasped due to kinematics of the hand which means the system is aware of whether or not an object can be grasped according to the visual input.

At this point, if a grasp that is expected to succeed fails, some assumptions about the object can be made such as its weight distribution not being even, the visual input representing



number	angle=Q1	dist=Q2	Q1 [0-1]	Q2 [0-1]	(Q1+Q2)/2	pose&state	EGA
1	0	17.97	0	1	0.5		1
2	0	17.97	0	1	0.5	surface1	1
3	0	17.97	0	1	0.5	down (state1)	1
4	15.26	17.97	1	1	1		2
5	0	13.85	0	0.77	0.39		1
6	0	13.85	0	0.77	0.39		1
7	0	13.85	0	0.77	0.39	surface2	1
8	13.46	13.85	0.88	0.77	0.83	down (state1)	2
9	0	13.85	0	0.77	0.39		1
10	0	13.85	0	0.77	0.39		1
11	0	17.97	0	1	0.5		1
12	0	17.97	0	1	0.5		1
13	0	17.97	0	1	0.5	vertex 0 and	2
14	14.31	0	0.94	0	0.47	1 down (state	3a
15	14.31	0	0.94	0	0.47	2)	3c
16	0	13.85	0	0.77	0.39		1
17	0	13.85	0	0.77	0.39		1
18	0	17.97	0	1	0.5		1
19	0	17.97	0	1	0.5		1
20	0	17.97	0	1	0.5	vertex 2 and	2
21	14.31	0	0.94	0	0.47	3 down (state	3a
22	14.31	0	0.94	0	0.47	2)	3c
23	0	13.85	0	0.77	0.39		1
24	0	13.85	0	0.77	0.39		1
25	0	17.97	0	1	0.5		1
26	0	17.97	0	1	0.5		1
27	13.46	13.85	0.88	0.77	0.83		2
28	15.26	17.97	1	1	1	vertex 0 and	2
29	14.31	0	0.94	0	0.47	3 down (state	3a
30	14.31	0	0.94	0	0.47	3)	3c

Figure 23: Quality values for the listed EGAs.

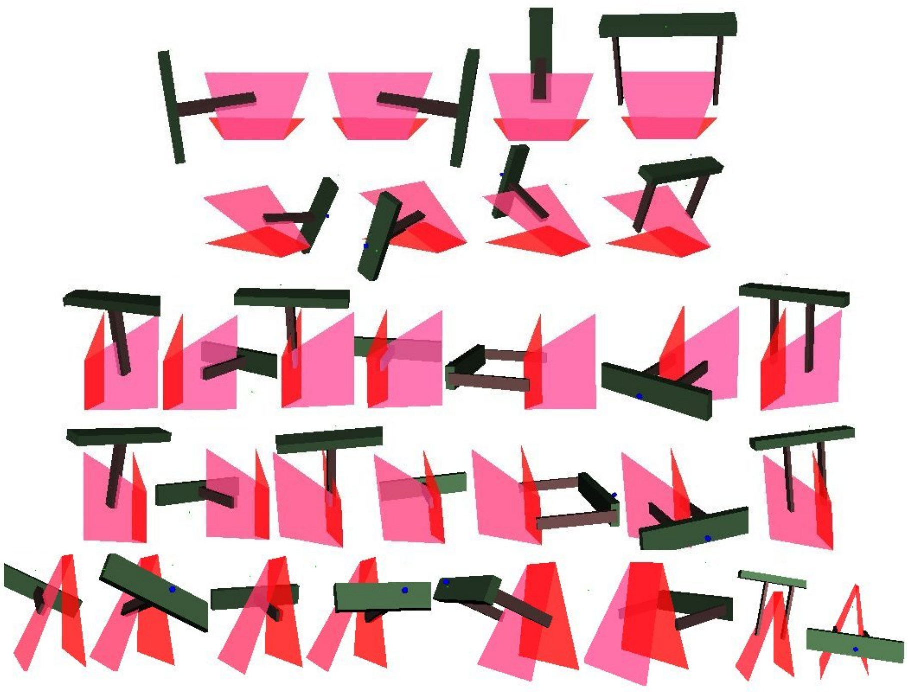


Figure 24: All EGAs for the given polyflap.

the objects shape, size and pose is not accurate enough or the real surface friction coefficients are different than the expected ones. The polyflap world abstracts away from a huge amount of complex detail of the real world. Therefore, polyflaps are good candidates to minimise the representation of an object model when it is possible to fit polyflaps to real world objects of interest, although the lack of curved surfaces and edges in the polyflap domain might be a restriction.

However there may be some problems when using EGAs on polyflaps detected on different objects. For example, in case of a scene with a box, EGAs as shown in Figure 25 can be generated if no constraints are taken into account. In order to eliminate unfeasible ones, there can be two alternatives which are trying to detect another polyflap or support the input model with additional point cloud information to include more knowledge from the scene that means seeking additional information in order to compensate for this situation and extend its knowledge about the scene. In the former case, the EGAs may not be directly applied to the combinations of polyflaps, therefore a need for a hierarchical representation of polyflaps (putting polyflaps together) may arise so that the relations between them are kept to form EGAs.

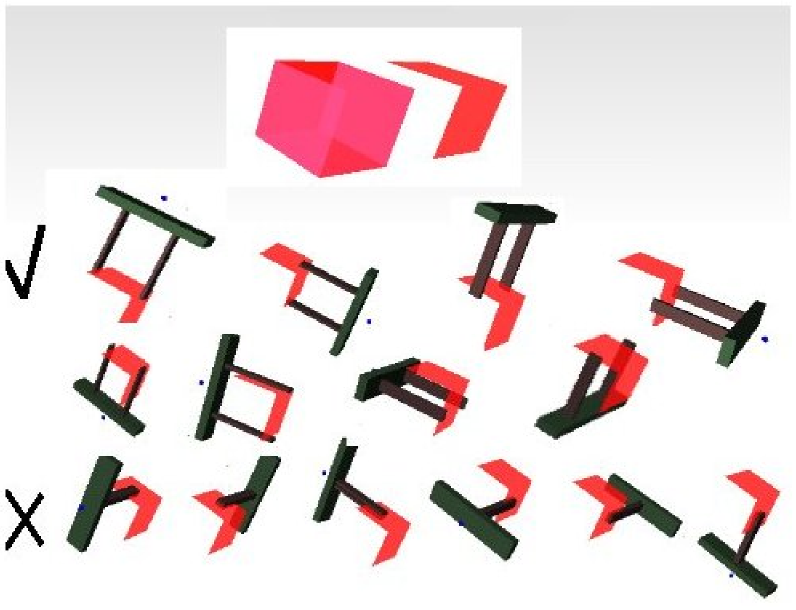


Figure 25: Example of fitting polyflaps.

Looking at grasping generally, failures can come from a number of things such as object shape, size or position and orientation wrongly observed depending on the level of accuracy of the vision system. If the situation where weight of object not evenly distributed is not considered, failure may occur as well. Another reason could be that surface texture is too smooth, therefore slippage occurs. In case object is detected as one object but separates from other parts when grasped, vision system detects failure but tactile sensors still detect success.

There may be other reasons due to embodiment such as limitations in the hand where objects a parallel gripper can grasp a multifinger may fail at and vice versa. Therefore, sometimes an exploratory stage may be needed where the robot hand pushed objects in order to confirm that there is one or more than one object, or to rotate the object to eliminate occlusions. In doing so, the object may be turned and a handle appears which increases the likelihood for choosing a better grasp. For example, a cup where the handle is facing backwards and is occluded from the vision system may appear cylindrical in nature and when the robot tries to grasp it fails because the back part is sticking out and results in not good enough contact between the hand

and the object, slippage can occur and result then in complete failure.

In terms of using polyflaps as a type of representation, the additional possibilities to those mentioned above can be listed as in the following. First of all, depending on the chosen work environment, polyflaps may not be detected on all graspable objects due to their shape being too different than a polyflap in nature. Another situation where grasp results in failure might be that when at least one polyflap is fit on the object by the vision system, but the chosen grasp for that polyflap is not stable enough due to the centre of mass of the object having a large distance from the grip centre causing a large amount of torque or as mentioned before, if the vision system can not provide enough information about the object, a grasp which is not feasible can be chosen.

In summary, the next step for our work will be to first integrate the presented system with a pushing module to demonstrate the feasibility of achieving poses from which good grasps can be performed. In parallel, we will work on extending the method to three fingered grasps and explore the use of the polyflap representations on real objects.

References

[1] Sloman, A., *The domain of polyflaps and other domains for acting and learning*, http://www.cs.bham.ac.uk/~axs/polyflaps/

[2] Sloman, A., *Polyflaps as a domain for perceiving, acting and learning in a 3-D world*, AAAI Fellows Symposium, Menlo Park, CA, (2006)

[3] Misra, B., and Silver, N., *Some discussion of static gripping and its stability*, IEEE Trans. Sys. Man Cybernet, 19(4):783-796, (1989)

[4] Troccaz, J. P., *Grasping: A state of the art*, The Robotics Review, Vol. 1, 71-98, MIT Press, Cambridge, MA, (1989)

[5] Li, J., Jin, M., and Liu, H., *A new algorithm for three-finger force-closure grasp of polygonal objects*, Proc. IEEE Int. Conf. on Robotics and Automation ICRA '03, IEEE Computer Society Press, Los Alamitos, (2003)

[6] Cutkosky, M. R., *On grasp choice, grasp models, and design of hands for manufacturing tasks*, IEEE Trans. Robotics Automation, RA-5(3):269-279, (1989)

[7] Liu, H., Iberall T., and Bekey, G. A., *The multidimensional quality of task requirements for dexterous robot hand control*, Proc. 1989 IEEE Int. Conf. on Robotics and Automation, 452-457, (1989)

[8] Iberall. T., *The nature of human prehension: Three dexterous hands in one*, Proc. IEEE Int. Conf. Robotics and Automation, 396-401, (1987)

[9] Shimoga, K. B., *Robot grasp synthesis algorithms: A survey*, Int. Journal of Robotics Research 15(3), 230-266, (1996)

[10] Cutkosky, M., and Howe, R., *Human grasp choice and robotic grasp analysis*, Dextrous robot hands, 5-31, New York, Spring-Verlag, (1990)

[11] Iberall, T., and Mackenzie, C., *Opposition space and human prehension*, Dextrous robot hands, 32-54, (1990)

[12] Bicchi, A., and Kumar, V., *Robotic grasping and contact: a review*, Proc. IEEE Int. Conf. on Robotics and Automation ICRA '00, vol. 1, IEEE Computer Society Press, Los Alamitos, (2000)

[13] Li, Z., and Sastry, S., *Issues in dextrous robot hands*, Dextrous robot hands, 154-186, (1990)

[14] Yoshikawa, T., and Nagai, K., *Analysis of multi-fingered grasping and manipulation*, Dextrous robot hands, 187-208, (1990)

[15] Ekvall, S., and Kragic, D., *Learning and Evaluation of the Approach Vector for Automatic Grasp Generation and Planning,* IEEE Int. Conf. on Robotics and Automation, 4715-4720, (2007)

[16] Morales, A., Azad, P., Asfour, T., Kraft, D., Knoop, S., Dillmann, R., Kargov, A., Pylatiuk, C., and Schulz, S., *An Anthropomorphic Grasping Approach for an Assistant Humanoid Robot,* 37th Int. Symposium on Robotics, 149-152, (2006)

[17] Glover, J., Rus, D., and Roy, N., *Probabilistic Models of Object Geometry for Grasp Planning,* IEEE Int. Conf. on Robotics and Automation, Pasadena, CA, USA, (2008)

[18] Li, Y., and Pollard N. S., *A shape matching algorithm for synthesizing humanlike enveloping grasps*, 5th IEEE-RAS Int. Conf. on In Humanoid Robots, 442-449, (2005)

[19] Heubner, K., and Kragic, D., *Selection of robot pre-grasps using box-based shape approximation*, IEEE Int. Conf. on Intelligent Robotics and Systems, 1765-1770, (2008)

[20] Biegelbauer, G., and Vincze, M., *Efficient 3D Object Detection by Fitting Superquadrics to Range Image Data for Robot's Object Manipulation*, IEEE Int. Conf. on Robotics and Automation, 1086-1091, (2007)

[21] Goldfeder, C., Allen, P. K., Lackner, C., and Pelossof, R., *Grasp Planning via Decomposition Trees*, IEEE Int. Conf. on Robotics and Automation, 4679-4684, (2007)

[22] Kraft, D., Pugeault, N., Baseski, E., Popovic, M., Kragic, D., Kalkan, S., Worgotter, F., and Krueger, N., *Birth of the Object: Detection of Objectness and Extraction of Object Shape through Object Action Complexes*, Int. Journal of Humanoid Robotics

[23] Chella, A., Dindo H., Matraxia, F., and Pirrone, R., *Real-time visual grasp synthesis using genetic algorithms and neural networks*, AI*IA '07: Proc. of the 10th Congress of the Italian Association for Artificial Intelligence, 567-578, Springer-Verlag, Rome, (2007)

[24] Sangkhavijit, C., Niparnan, N., and Chongstitvatana, P., *Computing 4-Fingered Force-Closure Grasps from surface Points Using Genetic Algorithm*, Proc. of the IEEE Int. Conf. on Robotics, Automation and Mechatronics, 1-5, (2006)

[25] Saxena, A., Driemeyer, J., Kearns, J., and Ng, A. Y., *Robotic Grasping of Novel Objects,* Neural Information Processing Systems, 19, 1209-1216, (2007)

[26] Nguyen, V. D., *Constructing force-closure grasps*, Int. Journal of Robotics Research, 7(3), (1988)

[27] Faverjon, B., and Ponce, J., *On computing two-finger force-closure grasps of curved 2D objects*, IEEE Int. Conf. on Robotics and Automation, 424-429, (1991)

[28] Park, Y. C., and Starr, G. P., *Grasp synthesis of polygonal objects using a three-fingered robot hand*, Int. Journal of Robotics Research, 11(3):163-184, (1992)

[29] Ponce, J., and Faverjon, B., *On computing three-finger force-closure grasps of polygonal objects*, IEEE Transactions on Robotics and Automation, 11(6):868-881, (1995)

[30] Markenscoff, X., Li, L., and Papadimitriou, C. H., *The geometry of grasping*, Int. Journal of Robotics Research, 9(1):61-74, (1990)

[31] Ferrari C., and Canny, J., *Planning optimal grasps*, IEEE Int. Conf. on Robotics and Automation, 2290-2295, Nice, France, (1992)

[32] Chen I. M., and Burdick, J. W., *Finding antipodal point grasps on irregularly shaped objects*, IEEE Int. Conf. on Robotics and Automation, 2278-2283, (1992)

[33] Guo, G., Gurver, W.A., and Zhang, Q., *Optimal grasps for planar multifinger robotic hands*, IEEE Transactions on Systems, Man and Cybernetics, 22(1):193-198, (1992)

[34] Mishra, B., and Teichmann, M., *Three finger optimal planar grasp*, IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Grenoble, France, (1994)

[35] Mirtich, B., and Canny, J., *Easily computable optimum grasps in 2D and 3D*, IEEE Int. Conf. on Robotics and Automation, 739-747, (1994)

[36] Park, Y., and Starr, G., *Optimal grasping using a multifingered robot hand*, Proc. IEEE Int. Conf. on Robotics and Automation, 689-694, IEEE Computer Society Press, Los Alamitos (1990)

[37] Morales, A., Sanz, P., and del Pobil, A., *Vision-based computation of three-finger grasps on unknown planar objects*, IEEE/RSJ Int. Conf. on Intelligent Robots and System, vol. 2, (2002)

[38] Sugiyama, S., Koeda, M., Fujimoto, H., and Yoshikawa, T., *Measurement of Grasp Position by Human Hands and Grasp Criterion for Two Soft-Fingered Robot Hands*, IEEE Int. Conf. on Robotics and Automation Kobe International Conference Center, Japan, (2009)

[39] Smith, G., Lee, E., Goldberg, K., Bhringer, K., and Craig, J., *Computing Parallel-Jaw Grips*, Proc. IEEE Int. Conf. Robotics and Automation, vol. 3, 1897-1903, (1999)

[40] Miller, A. T. and Allen, P. K. *Graspit! A Versatile Simulator for Robotic Grasping*, Robotics and Automation, 11(4), 110-122, (2004).

[41] Mackenzie, C., and Iberall, T., *The Grasping Hand*, North Holland, (1994)

Prediction learning in robotic pushing manipulation

Marek Kopicki, Jeremy Wyatt, Rustam Stolkin
School of Computer Science
University of Birmingham, UK

Abstract— **This paper addresses the problem of learning about the interactions of rigid bodies. A probabilistic framework is presented for predicting the motion of one rigid body following contact with another. We describe an algorithm for learning these predictions from observations, which does not make use of physics and is not restricted to domains with particular physics. We demonstrate the method in a scenario where a robot arm applies pushes to objects. The probabilistic nature of the algorithm enables it to generalize from learned examples, to successfully predict the resulting object motion for previously unseen object poses, push directions and new objects with novel shape. We evaluate the method with empirical experiments in a physics simulator.**

I. Introduction

In early childhood, humans and animals learn models for predicting the interactions with the environment [1]. It seems unlikely that these models comprise an explicit encoding of Newtonian physics, and so must instead rely on a learned relationship between observed actions and their outcomes.

This paper addresses the problem of learning to predict the motion of one body, which results from a forced interaction with another. We have chosen to investigate this problem in the context of robotic "poking" or "pushing" operations, because this includes a large number of unstable manipulations and hence provides interesting situations. However the work is potentially more general.

An algorithm is presented which learns to predict the motions of a rigid object that will result from an applied robotic pushing action. The algorithm does not rely on any understanding or encoding of Newtonian mechanics, but can be trained in simple online experiments in which a robot arm applies random pushes to objects of interest and extracts the resulting motions using a vision system. Properties of objects, and their interactions, are learned as distributions.

Pushing operations are encountered frequently in robotics, but have received relatively little attention in the research community. They are important in that robotic grasping frequently involves a pushing phase, when one finger or jaw of a gripper contacts the workpiece before another. Furthermore, pushing may often be preferable to pick and place type operations if the robot lacks the size or strength necessary to lift an object.

[2] was the first to identify pushing operations as fundamental to manipulation, especially grasping. Mason develops a detailed analysis of the mechanics of pushed, sliding objects and determines conditions required for various 2D motions of a pushed object. [3] attempts to put quantitative bounds on the rate at which these predicted motions occur. [4] developed a method for finding the set of all possible motions of a sliding object, in response to an applied push. More recently, [5] has developed path planning techniques for push manipulation of 2D sliding objects, based on the use of a physics simulator for prediction.

The above work is restricted to planar sliding motions of effectively 2D objects. In contrast, there is comparatively little literature which addresses the far more complex problems of predicting the results of push manipulations on real 3D bodies, which are free to tip or roll. It is possible to use physics simulators to predict the motions of interacting rigid bodies, however this approach is reliant on explicit knowledge of the objects, the environment and key physical parameters. It is therefore not generalizable to new objects or novel situations.

Machine learning approaches have been developed to learn pre-specified binary affordance classes, e.g. rolling versus non-rolling objects [6], or liftable versus non-liftable objects [7]. [8] present experiments where a robot arm coupled to a vision system learns affordances (e.g. rolling or sliding) of various different objects by applying pushes and then observing the resulting motions. This kind of approach is limited, in that affordances learned for a specific object and push action, may not be generalizable to a new object, pose or push direction. Furthermore, although certain primitive classes of motion, e.g. "rolling", may be predicted, such systems cannot predict an explicit 6-DOF rigid body motion for the pushed object.

In contrast, we present a system which can learn to predict the explicit 3D rigid body transformations that will result when an object in an arbitrary orientation is subjected to an arbitrary push. The probabilistic nature of the learning enables generalization to previously unseen push directions and object poses. Furthermore, the system is often able to successfully predict the behaviors of novel objects with previously unencountered shapes.

II. Representing the interaction of rigid bodies

Consider three reference frames A, B and O in a 3-dimensional Cartesian space (see Figure 1). While frame O is fixed, A and B change in time and are observed at discrete time steps $..., t-1, t, t+1, ...$ every non-zero Δt. A frame X

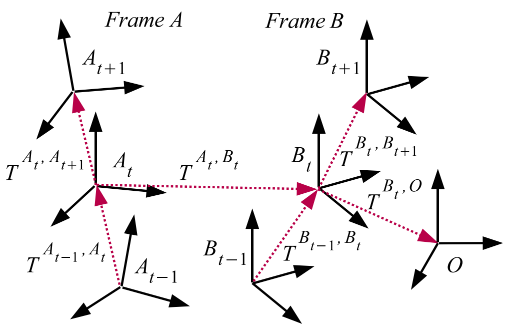


Fig. 1. A system consisting of two interacting bodies with frames A and B in some constant environment with frame O can be described by six rigid body transformations T^{A_t,B_t}, $T^{B_t,O}$, T^{A_{t-1},A_t}, $T^{A_t,A_{t+1}}$, T^{B_{t-1},B_t}, and $T^{B_t,B_{t+1}}$.

at time step t is denoted by X^t, a rigid body transformation between a frame X and a frame Y is denoted by $T^{X,Y}$.

From classical mechanics we know that in order to predict a state of a body, it is sufficient to know its mass, velocity and a net force applied to the body. We do not assume any knowledge of the mass and applied forces, however the transformations of a body, with attached frame B, over two time steps T^{B_{t-1},B_t} and $T^{B_t,B_{t+1}}$ encode its acceleration - the effect of the applied net force. Therefore, if the net force and the body mass are constant, the transformations T^{B_{t-1},B_t} and $T^{B_t,B_{t+1}}$ provide a complete description of the state of a body at time step t in absence of other bodies. A triple of transformations $T^{B_t,O}$, T^{B_{t-1},B_t} and $T^{B_t,B_{t+1}}$ provide a complete description of a state of a body in some fixed frame of reference O which accounts for a constant or stationary environment. Similarly, transformations $T^{A_t,O}$, T^{A_{t-1},A_t} and $T^{A_t,A_{t+1}}$ provide such a description for some other body with frame A.

The state of a system consisting of two bodies with frames A and B in some constant environment with frame O can be described by the six transformations as it is shown in Figure 1, where $T^{A_t,O}$ has been replaced by a relative transformation T^{A_t,B_t}. The transformation $T^{B_t,O}$ can be omitted, if the environment does not affect the motion of the bodies or it is explicitly modeled by one of them.

The prediction problem can now be stated as: given we know or observe the starting states and the motion of the pusher, $T^{A_t,A_{t+1}}$, predict the resulting motion of the object, $T^{B_t,B_{t+1}}$. This is a problem of finding a function:

$$f : T^{A_t,B_t}, T^{B_t,O}, T^{A_{t-1},A_t}, T^{B_{t-1},B_t}, T^{A_t,A_{t+1}} \rightarrow T^{B_t,B_{t+1}} \tag{1}$$

Function 1 is capable of encoding all possible effects of interactions between rigid bodies A and B, providing their physical properties and applied net forces are constant in time. Furthermore, it can be learned purely from observations for some fixed time delta Δt. There are two important problems related to relying on such a function:

1) **Limited or no generalization capability.** A function approximating interactions between bodies A and B cannot be used for any other bodies of e.g. different shape or mass. This is because function 1 implicitly encodes information about the surfaces of A and B, which play a critical role in collisions. In this way a slight change of the objects' shape can cause a dramatic deviation of the predicted transformation $T^{B_t,B_{t+1}}$.
2) **Dimensionality problem.** For a rigid body transformation represented as a set of 6 or 7 numbers, the domain of function 1 has 30 or 35 dimensions.

III. Combining Local and Global Information

It is clear that we need to enable generalization of predictions with respect to changes in shape. We also assume quasi-static conditions, i.e. we ignored all frames at time $t-1$. Consider two objects lying on a table top. In Figure 2 there are two situations that are identical except for the shape of the object A, yet it is clear that the same transformation of A's position will lead to quite a different motion for object B. How can we encode the way that the shapes of A and B alter the way they behave? We use a product of several densities to approximate the density over the rigid body transformation given in the function 1.

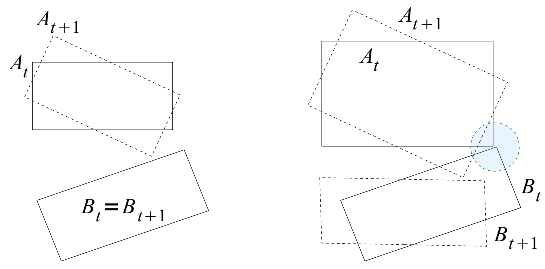


Fig. 2. Two scenes, each with two objects on a table top, viewed from above. Between the two scenes only the shape of A is different. Yet when A moves the resulting transformation $T^{B_t,B_{t+1}}$ will be quite different. This shows that our predictors must take some aspect of the shape of A and B into account.

To do this we approximate two densities, conditioned on local and global information respectively. We define the global information to be the information about the pose, but not the shape, of the whole object. We define the local shape we consider here to be the pose of the surfaces of A and B at the contact point, or the point of closest proximity, between the object and the finger. We model this local shape as a pair of planar surface patches, of limited extent (see Figure 3). Statistically, the greater the starting distance between these local surface patches of A and B, and/or the smaller the magnitude of the transformation $T^{A_t,A_{t+1}}$, the less likely it is that the objects will collide, and hence the less likely it is that the pose of shape B will change between t and $t+1$, or equivalently the more likely that the transformation $T^{B_t,B_{t+1}}$ will be an identity transformation Id. On the other hand, if the local surfaces A and B are close a large portion of possible transformations $T^{A_t,A_{t+1}}$ will cause collisions.

Transformations T^{A_t,B_t}, $T^{A_t,A_{t+1}}$ and $T^{B_t,B_{t+1}}$, observed over many experimental trials for many different

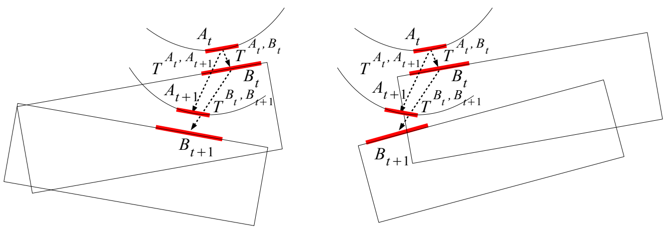


Fig. 3. Two scenes, each with two objects on a table top, viewed from above. Local shapes A and B, transformations $T^{A_t,A_{t+1}}$ and T^{A_t,B_t} are the same in each scene. Still, the transformation $T^{B_t,B_{t+1}}$ is different because local shapes belong to different parts of objects.

objects form a distribution. A particularly useful distribution is a conditional distribution:

$$\{T^{B_t,B_{t+1}}|T^{A_t,A_{t+1}},T^{A_t,B_t}\} \tag{2}$$

While conditional distribution 2 for global frames may become unimodal, for local shapes is highly multi-modal. To see this consider two scenes with two objects, where the initial conditions are identical (Figure 3). Local shapes A and B, transformations $T^{A_t,A_{t+1}}$ and T^{A_t,B_t} are the same in each scene. Still, the transformation $T^{B_t,B_{t+1}}$ is different because local shapes belong to different parts of objects.

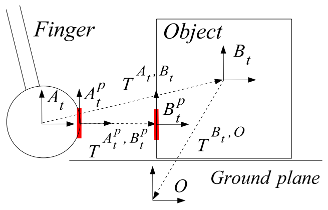


Fig. 4. 2D projection at time t of a robotic finger with global frame A_t, an object with global frame B_t, and a ground plane with constant global frame O. Local frames A_t^p and B_t^p describe the local shape of the finger and an object at their point of closest proximity.

Consider a 2D projection at time t of a robotic finger with global frame A_t, an object with global frame B_t, and a ground plane with constant global frame O (Figure 4). Similarly, local frames A_t^p and B_t^p describe local shapes belonging to a finger and an object. The global conditional density function can be defined as:

$$p(T^{B_t,B_{t+1}}|T^{A_t,A_{t+1}},T^{A_t,B_t},T^{B_t,O}) \tag{3}$$

and similarly a local conditional density function as:

$$p^c(T^{B_t^p,B_{t+1}^p}|T^{A_t^p,A_{t+1}^p},T^{A_t^p,B_t^p}) \tag{4}$$

Because both objects are rigid, $T^{A_t,A_{t+1}} \equiv T^{A_t^p,A_{t+1}^p}$ and $T^{B_t,B_{t+1}} \equiv T^{B_t^p,B_{t+1}^p}$. To predict the rigid body transformation of an object when it is in contact with others we are faced with how to represent the constraints on motion provided by the contacts. We do this using a product of experts. The experts represent by density estimation which rigid body transforms are (in)feasible for each frame of reference. In the product, only transformations which are feasible in both frames will have high probability. For the finger-object scenario a prediction problem can then be defined as finding that $T^{B_t,B_{t+1}}$ which maximizes the product of the two conditional densities (experts) 3 and 4:

$$\max_{T^{B_t,B_{t+1}}} \quad p(T^{B_t,B_{t+1}}|T^{A_t,A_{t+1}},T^{A_t,B_t},T^{B_t,O}) \times \\ p^c(T^{B_t,B_{t+1}}|T^{A_t,A_{t+1}},T^{A_t^p,B_t^p}) \tag{5}$$

The prediction problem cannot be solved using regression approach. Two regression estimates could only be combined linearly since they each make only a single prediction. Without information about the density around each of these predictions there is no ability to find compromise predictions in a principled way. In principle it is possible to fit unimodal densities using regression, but even this approach will lead to failure if the conditional distribution is multi-modal. In this case the conditional distributions are indeed highly multi-modal. Another way of saying this is that since the constraints are clearly highly non-linear the regression approach will fail for even very simple situations.

Starting with some initial state of the finger T^{A_0} and the object T^{B_0}, and knowing a trajectory of the finger $A_1,\ldots A_N$ over T time steps, one can predict a whole trajectory of an object $B_1,\ldots B_N$ by sequentially solving a problem of maximization of the product 5.

There are two major advantages of using such products of densities, e.g. over attempting to directly approximate the function of equation 1:

1) **Efficient movement encoding and learning.** Combining information from both local and global frames, allows objects' properties to be separated into those that are common to many objects and those that are specific to the particular object in question. Common properties (e.g. impenetrability) tend to be encoded in the local surface patches distribution, function 4, whereas the global density function 3 encodes information specific to the workpiece, such as its overall shape. The global density function 3 tends not to require many learning trials to provide accurate predictions, when combined with the local density function 4, which is shared or common to many different objects or situations. Thus this combination provides a movement encoding and learning method which is highly efficient.
2) **Generalization.** Even small differences in a local object surface can cause very different reactions $T^{B_t,B_{t+1}}$ for some given action $T^{A_t,A_{t+1}}$. However, such changes are unlikely to be predicted by a global density function alone. Hence, computing $T^{B_t,B_{t+1}}$ as the maximizer of the product of densities, equation 5, enhances the ability of the system to generalise between different objects and actions, because both local and global densities must simultaneously support the predicted motion hypothesis $T^{B_t,B_{t+1}}$.

IV. Learning as density estimation

We use memory-based learning in which all *learning samples* are stored during learning. The learning samples create a global joint distribution:

$$\{T^{A_t,B_t}, T^{B_t,O}, T^{A_t,A_{t+1}}, T^{B_t,B_{t+1}}\} \tag{6}$$

and local joint distribution:

$$\{T^{A_t^p,B_t^p}, T^{A_t,A_{t+1}}, T^{B_t,B_{t+1}}\} \tag{7}$$

We address $3D$ rigid bodies, subject to 6-DOF transformations, so that distributions 6 and 7 have $4 \times 6 = 24$ and $3 \times 6 = 18$ dimensions respectively. During prediction conditional densities 3 and 4 are created online from learning sample sets (i.e. from distributions 6 and 7).

Consider N D-dimensional sample vectors X_i drawn from some unknown distribution. We would like to find an approximation of this distribution in the form of a density function $p(X)$. Kernel density methods with Gaussian kernels (see e.g. [9]) estimates the density $p(X)$ for any given vector X as a sum of N identical multivariate Gaussian densities centered on each sample vector X_i:

$$p(X) = C_{norm} \sum_{i=1\ldots N} \exp\left[-\frac{1}{2}(X - X_i)^T \mathbf{C}^{-1}(X - X_i)\right] \tag{8}$$

where a constant $C_{norm} = [N(2\pi)^{D/2}|\mathbf{C}|^{1/2}]^{-1}$ and $\mathbf{C}$ is a $D \times D$ sample covariance matrix. For simplicity, we assume that $\mathbf{C}$ is diagonal. The above equation can be re-written in a new simpler form ([9]):

$$p(X) = \frac{1}{N} \sum_{i=1\ldots N} \left[\prod_{j=1\ldots D} K_{h_j}(X^j - X_i^j)\right] \tag{9}$$

where K_{h_j} are 1-dimensional Gaussian kernel functions:

$$K_{h_j}(X^j - X_i^j) = \frac{1}{(2\pi)^{1/2}h_j} \exp\left[\frac{X^j - X_i^j}{h_j}\right] \tag{10}$$

and D parameters h_j are called bandwidth $H \equiv (h_1, \ldots, h_D)$. The bandwidth H is estimated from all distribution learning samples using the "multivariate rule-of-thumb", see [9].

Let us decompose each D-dimensional sample vector X_i into two vectors: K-dimensional Y_i and L-dimensional Z_i so that $X_i \equiv (Y_i, Z_i)^T$ and $D = K + L$. Knowing bandwidth H or equivalently diagonal covariance matrix $\mathbf{C}$ for sample set $\{X_i\} \equiv \{(Y_i, Z_i)^T\}$, we can compute conditional density $p(Z|Y)$ for some given vectors Y and Z using the following two step procedure:

1) Find a set of M weighted samples $\{(Z_i, w_i)\}$ representing a conditional distribution for given vector Y, such that Y_i which corresponds to Z_i lies within some predefined maximum Mahalanobis distance d_{max} to vector Y. Mahalanobis distance d_i between sample vector Y_i and vector Y is defined as:

$$d_i = (Y - Y_i)^T \mathbf{C}_Y^{-1}(Y - Y_i) \tag{11}$$

where diagonal covariance $\mathbf{C}_Y$ is defined as:

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_Y & 0 \\ 0 & \mathbf{C}_Z \end{bmatrix} \tag{12}$$

Weights w_i are computed from distance d_i as:

$$w_i = \exp[-d_i/2] \tag{13}$$

and normalized for all M weights w_i. Normalized weight w_i can be interpreted as a probability of generating Y_i from a multivariate Gaussian centered at Y with covariance $\mathbf{C}_Y$.

2) Compute conditional probability density $p(Z|Y)$ as:

$$p(Z|Y) = \sum_{i=1\ldots M} w_i \exp\left[-\frac{1}{2}(Z - Z_i)^T \mathbf{C}_Z^{-1}(Z - Z_i)\right] \tag{14}$$

The density product 5 is maximized using the differential evolution optimization algorithm [10]. This requires the ability to evaluate and sample from each distribution comprising product 5.

All conditional distributions are represented as a weighted set of samples $\{(Z_i, w_i)\}$. Computation of a probability density for some given vector Z is realized as in Equation 14. Sampling consists of a two step procedure:

1) Choose vector Z_i from a set of samples $\{(Z_i, w_i)\}$ using an importance sampling algorithm with importance weights w_i ([11]).
2) Sample from a multivariate Gaussian centered at Z_i with covariance $\mathbf{C}_Z$.

V. Results

We evaluated the prediction algorithm with experiments in a physics simulator. Multiple experimental trials are performed, in which a 5-DOF robotic arm equipped with a finger performs a random movement of length approximately 25 cm towards an object at a random initial pose (Figure 5). In each experiment, learning samples comprising distributions 6 and 7 are stored for a particular object over a series of such random trials. Each experimental trial lasts 10 seconds, while learning samples are stored every 0.1 seconds. Further, new random trials are then generated and the learned distributions are tasked with predicting the resulting motions. Although random trials are independently generated for the learning and prediction phases, the same level of variability in pose and pushing action is used for each phase.

We take the output of a physics simulator to be ground-truth, and compare this with predictions made by our statistical learning method according to an average prediction error E defined as:

$$E = \frac{1}{K} \sum_{k=1\ldots K} \frac{1}{T} \sum_{t=1\ldots T} \frac{1}{N} \sum_{n=1\ldots N} |p_n^1 - p_n^2| \tag{15}$$

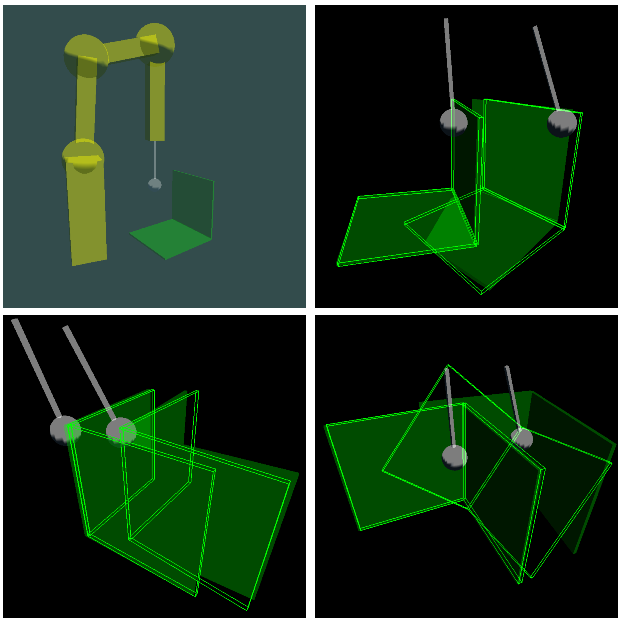


Fig. 5. A 5-DOF robotic arm equipped with a finger performs forward movements towards an object (top left). Object behavior varies depending on the initial object pose and finger trajectory. Physics simulator predictions are rendered as solid, while predictions obtained from our prediction algorithm are rendered as wired. A majority of the algorithm predictions are qualitatively plausible (top right and bottom left). Bottom right panel shows a qualitative error.

where K is a number of experiment trials, T is a number of discrete time steps in each trial (i.e. trial duration), N is a number of pairs of 3D points $\{p_n^1, p_n^2\}$, $|\cdot|$ denotes Euclidean distance between points in a pair. Points p_n^1 are rigidly attached to an object controlled by a physics simulator, while points p_n^2 to an object controlled by the prediction algorithm. All points are randomly generated at the beginning of each trial so that for $t = 1$, $p_n^1 = p_n^2$ for all n.

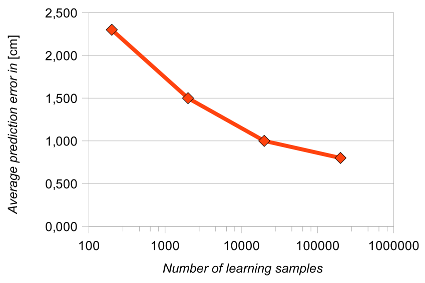


Fig. 6. Average prediction error for a polyflap in a function of a number of learning samples.

In the first experiment a robot pushes a simple symmetric $14cm \times 14cm$ polyflap[1] (Figure 5) placed randomly on a ground plane in arbitrary stable poses.

Figure 6 shows the average prediction error as a function of the number of samples collected during learning (the same for local distribution 7 and global distribution 6). The error decreases as the number of learning samples increases and predictions are reasonably good for just a few thousand learning samples. Even in cases where the prediction errors are large, the majority of predictions are qualitatively plausible, for example, correctly predicting whether a polyflap will slide, tilt or topple (Figure 7).

[1]Polyflaps are objects consisting of a number of connected flat surfaces. Their behavior can be very complex as compared to e.g. a simple box.

Polyflap shape modification	Prediction error [cm]
none (learnt shape)	0.76
narrowed by 50%	0.72
widened by 40%	1.3
skewed by 15°	1.16
skewed by 30°	1.26
skewed by 40°	1.35

TABLE I
PREDICTION ERROR IN POLYFLAP EXPERIMENTS.

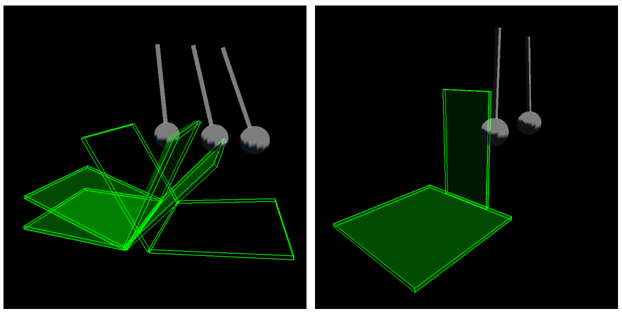


Fig. 7. After the modification of the polyflap shape most of predictions are still qualitatively correct. For example the algorithm predicts that a polyflap tips instead of returning to its initial pose after tilting (left), furthermore it makes no errors if some parts of the surface are removed (right).

In the second experiment we attempted to generalise to novel objects, by using learning samples from the first experiment to predict the trajectory of a new polyflap with a previously unseen shape. We experimented with 5 types of modified polyflap shapes which, together with corresponding prediction error, are collected in Table I. Most predictions are qualitatively correct (Figure 7), however there are more coarse errors compared to the previous experiment.

Box shape modification	Prediction error [cm]
none (learnt shape)	1.68
narrowed by 40%	1.72
widened by 30%	2.15
enlarged by 30%	2.75

TABLE II
PREDICTION ERROR IN BOX EXPERIMENTS.

In a third experiment, we learn on a box ($16cm \times 5cm \times 12cm$ parallelepiped) instead of a polyflap, and try to generalize to predict the motions of distorted boxes that are differently shaped to the one used in learning. We considered 3 types of box modifications which are also compared to the unmodified box shape in Table II. Note that the absolute prediction error is larger than for the polyflap experiment,

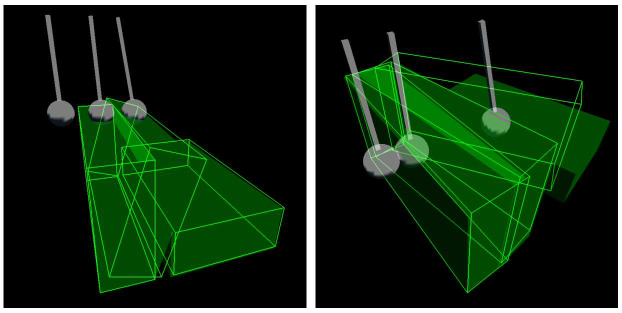


Fig. 8. For a modified box shape version many predictions are correct (left) while if the modified shape extends beyond the learnt one the predictor tends to make more errors (right).

but this is mostly due to the larger box dimensions, and more frequent rotational box movements (see Figure 8).

Prediction	Learning samples	Prediction error [cm]
polyflap	box	2.38
box	polyflap	3.13

TABLE III
PREDICTION ERROR FOR SWAPPED LEARNING SAMPLES.

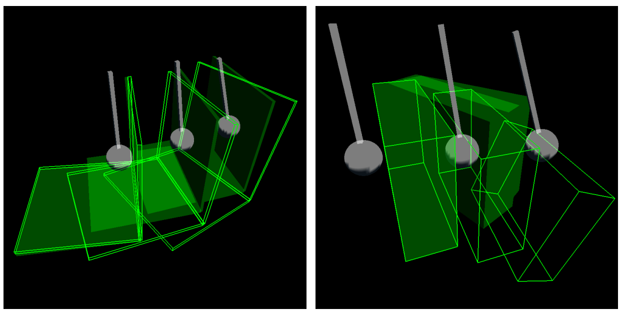


Fig. 9. After swapping learning samples between a box and a polyflap, the majority of predictions are still qualitatively plausible (left). However there is a relatively larger percentage of coarse errors (right).

In the final experiment we attempted to generalise between qualitatively different objects, using box learning samples to predict a polyflap trajectory, and polyflap learning samples to predict a box trajectory (Table III). Again, a majority of predictions are still qualitatively plausible, however there is a relatively larger percentage of coarse errors (Figure 9)

Although the results of these experiments are promising, the algorithm displays errors which are especially visible in the shape generalization experiments and where there are motions involving large amounts of rotation. The major sources of these errors are thought to be:

1) **Density estimation algorithm.** Uniform covariances, used for all kernels, are not completelly adequate for approximating local densities in so many dimensions. Kernel density estimators are unable to handle rank-deficient data [9], whereas such data is clearly present in the introduced distributions.
2) **Correlation problem.** The density product 5 express a correlation (functional relationship) between a finger and an object. While they are correlated if they are in contact, they are no longer strongly correlated when they lose contact following a push. An example is when a polyflap tips over after being pushed by the finger.

VI. Conclusions

We have presented a statistical framework for learning to predict the motions of interacting objects. By decomposing the prediction task into a product of two distributions, each encoding different kinds of information, we have demonstrated a degree of generality in terms of handling variations in shape, poses and actions. We have also shown that it is possible to produce reasonable predictions for the behaviour of a novel shape (e.g. a box), having learned on a quite different one (e.g. a polyflap). This is despite the very small number of densities we use to encode the spatial relationship and shape of the two objects. We are now extending this approach to a product of many densities to give an improved representation of object shape. Future work will also look at using this prediction system for path planning and control during robotic pushing operations.

VII. Acknowledgments

We gratefully acknowledge the support for this work of the EC funded FP7 project CogX, number ICT-215181-CogX.

References

[1] A. Berthoz, *The Brain's Sense of Movement*. Harvard University Press, 1997.
[2] M. Mason, "Mechanics and planning of manipulator pushing operations," *IJRR*, vol. 5, no. 3, pp. 53–71, 1986.
[3] M. Peshkin and A. Sanderson, "The motion of a pushed, sliding workpiece," *IEEE Journal of robotics and automation*, vol. 4, no. 6, 1988.
[4] K. Lynch, "The mechanics of fine manipulation by pushing," in *Proc. IEEE ICRA*, 1992.
[5] D. Cappelleri, J. Fink, B. Mukundakrishnan, V. Kumar, and J. Trinkle, "Designing open-loop plans for planar micro-manipulation," in *Proc. IEEE ICRA*, 2006.
[6] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, and G. Sandini, "Learning about objects through action - initial steps towards artificial cognition," in *Proc. IEEE ICRA*, 2003.
[7] L. Paletta, G. Fritz, F. Kintzler, J. Irran, and G. Dorffner, "Learning to perceive affordances in a framework of developmental embodied cognition," in *Proc. IEEE Int. conf. on development and learning*, 2007.
[8] B. Ridge, D. Skocaj, and A. Leonardis, "A system for learning basic object affordances using a self-organizing map," in *Proc. Int. conf. on cognitive systems*, 2008.
[9] D. W. Scott and S. R. Sain, *"Multi-Dimensional Density Estimation"*, pp. 229–263. Elsevier, 2004.
[10] R. Storn and K. Price, "Differential evolution. a simple and efficient heuristic for global optimization over continuous spaces," *J. of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.
[11] C. Bishop, *Pattern recognition and machine learning*. Springer, 2006.