TU

Technische Universität Wien

DISSERTATION

Making Sense of Images: Parameter-Free Perceptual Grouping

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Doktors der technischen Wissenschaften unter der Leitung von

A.o.Univ.-Prof.Dipl.-Ing.Dr.techn. Markus Vincze

E376

Institut für Automatisierungs- und Regelungstechnik

eingereicht an der Technischen Universität Wien
Fakultät für Elektrotechnik und Informationstechnik

von

Dipl.-Ing. Michael Zillich

E9056241

Webergasse 21/26-28, 1200 Wien

Wien, im Mai 2007

Abstract

Perceptual grouping is a well studied area in visual psychophysics and offers a principled, general way to study vision in sighted animals and humans as well as machines. Computational approaches in the past however have often been hampered by complexity issues and brittleness in the presence of clutter and noise. Especially the reliance on tuning parameters renders many approaches impractical for real world applications.

This work aims to address complexity and robustness issues by proposing an incremental processing scheme for the perceptual grouping of edges, where the only parameter is runtime. This allows interrupting processing at any time, returning the most significant perceptual groups that could be found up to that point and leads to graceful degradation with increasing amounts of noise or clutter.

We furthermore propose a probabilistic measure of visual significance based on the principle of non-accidentalness. This significance measure is used to guide grouping of convex contours as well as a relative depth ordering of contours. Varying the significance measure of edges, for example based on regions of interest, allows to focus attention on specific parts of the scene, which will subsequently be allocated a larger share of the available processing time.

Experiments were carried out on a wide range of real world images with varying scene content and complexity. For detection of convex contours we could show that identifying candidate edges and junctions for grouping can be performed in runtime linear to the number of edges. More significant contours typically popped out faster with less significant ones appearing as runtime progresses. We demonstrated how two attentional mechanisms, based on regions of interest and colour, lead to faster detection of objects of interest. Relative depth ordering of contours based on energy minimisation in a Markov Random Field was presented as an initial form of finding a globally consistent scene interpretation and was shown to work for scenes of limited complexity.

Acknowledgements

This work would not have been possible without the continued support of my family, friends and colleagues and most notably my supervisor Markus Vincze. I am grateful for the intellectual freedom he allowed me and for his never-ending patience. I also want to thank my external reviewer Vaclav Hlavac for his time and for his encouragements.

Even the most interesting scientific work would be boring without such great office mates as Minu Ayromlou, Wolfgang Ponweiser, Stefan Chroust, Dietmar Legenstein, Andreas Pichler, Rainer Danzinger, Georg Biegelbauer, Matthias Schlemmer, Peter Einramhof, Peter Gemeiner and Johann Prankl. I will miss our tabletop soccer matches.

I very much enjoyed working with Danny Roobaert and Jan-Olof Eklundh at KTH, Stockholm. And I also want to express special thanks to Jiri Matas, Czech Technical University, Prague for his inspiring ideas and many interesting discussions.

Not least I have to thank the general public for funding this work via various EU and nationally funded projects.

Finally I want express my love and gratitude to Nina for keeping me going and giving me strength whenever things looked bleak. I also want to thank my parents for setting me onto a path leading to science early on (mostly by providing my brother and me with countless LEGO kits) and for enabling my studies.

Thanks to all of You for bearing with me.

Contents

1 Introduction **1**
- 1.1 Context of Work . . . 2
- 1.2 What is Vision? . . . 3
- 1.3 The Role of Vision . . . 4
- 1.4 Requirements . . . 12
- 1.5 Thesis . . . 13

2 Related Work **15**

3 A Case of Object Recognition **25**
- 3.1 Support Vector Machine Learning . . . 25
- 3.2 Pedagogical learning . . . 27
- 3.3 Online recognition . . . 28
- 3.4 Experiments . . . 29
- 3.5 Discussion . . . 34

4 Significance **37**
- 4.1 The Helmholtz Principle . . . 37
- 4.2 Poisson Process . . . 38
- 4.3 Markov Chain . . . 41
- 4.4 Bernoulli Process . . . 43
- 4.5 Log-Likelihood . . . 45

5 Detecting Ellipses **47**
- 5.1 State of the Art . . . 47
- 5.2 Overview . . . 50
- 5.3 A Remark on Edges . . . 51
- 5.4 Experimental setup . . . 53
- 5.5 Detecting Circular Arcs . . . 53
  - 5.5.1 Sequential Growing (GROW) . . . 54
  - 5.5.2 Recursive Splitting (SPLIT) . . . 55
  - 5.5.3 RANSAC . . . 59
  - 5.5.4 Comparison of Arc Segmentation Methods . . . 61

5.6 Finding Convex Groups of Arcs . . . 63
5.6.1 Convexity Criteria . . . 63
5.6.2 Comparison of Convexity Criteria . . . 65
5.6.3 Identifying Convex Arcs - Image Space Indexing . . . 66
5.6.4 Growing Convex Groups . . . 70
5.6.5 Comparison of Growing Methods . . . 76
5.7 Fitting Ellipses . . . 77
5.7.1 The B2AC Algorithm . . . 77
5.7.2 Assessing Ellipse Quality . . . 78
5.8 Discussion . . . 83

6 Incremental Grouping **85**
6.1 State of the Art . . . 86
6.2 Overview . . . 89
6.3 Junctions . . . 90
6.3.1 Collinearities . . . 91
6.3.2 L-Junctions . . . 92
6.3.3 T-Junctions . . . 92
6.3.4 Interactions Between Junctions . . . 93
6.3.5 Adding Colour . . . 94
6.3.6 Remark on Adding Cures . . . 95
6.4 Indexing and Voting . . . 95
6.4.1 Image Space Indexing . . . 97
6.4.2 Runtime Complexity . . . 97
6.5 The Kanizsa Square Problem . . . 99
6.6 Incremental Processing and Anytimeness . . . 101
6.6.1 Incremental Growing of Search Lines . . . 102
6.6.2 Equal Growth . . . 103
6.6.3 Length-Weighted Growth . . . 103
6.6.4 Boosting/Inhibition . . . 103
6.6.5 Tangents over Normals . . . 104
6.7 Closed Path Search . . . 106
6.7.1 Graph Construction . . . 106
6.7.2 Graph Search . . . 106
6.7.3 Runtime Complexity . . . 107
6.8 Results . . . 109
6.8.1 Occlusion, Amodal Completion . . . 109
6.8.2 Adding Attention . . . 111
6.9 Discussion . . . 114

7 Depth Ordering of Surfaces **117**
7.1 Markov Random Field for Depth Ordering 119
7.2 Markov Random Fields 120
7.3 Problem Formulation 121
7.4 Highest Confidence First Energy Minimisation 123
7.5 Results 124
7.6 Discussion 127

8 Summary and Discussion **131**

A Data Set for Ellipse Detection **133**

B Detecting Closures **139**

Chapter 1

Introduction

Computer vision has fragmented into many specialised sub-fields, each developing and perfecting methods to solve specific tasks such as optical character recognition or face recognition, some of which have been successful enough to result in commercial products. Yet many of these methods are not applicable to other domains and it seems that not much insight into a broader understanding of vision can be gained from them.

One of the original dreams of AI to solve generic computer vision has not materialised. Yet it should be evident that without such a generic solution, we can not claim to have a solution at all. Let us not be fooled into believing that an assortion of (however well developed, excellent) specialised techniques can constitute a coherent whole, or ever will. A proper theory of vision still seems far away.

How lucky in contrast is for example the field of control theory. After countless ad-hoc techniques to control various types of machinery (such as steam engines or chemical processes) a proper theory of control was finally developed (starting with a dynamics analysis of the centrifugal governor by James Clerk Maxwell in 1868). We now have a theory of control which can describe control problems in a coherent framework. And we can use this framework to develop implementations. Obviously nature chose different implementations. But both can be described using the same theory. Unfortunately a theory of cognitive processes, of which vision is one example, will not be as easy to obtain. We consider it important however to keep in the back of ones head, that it is ultimately such a theory that we should be after, rather than a collection of shortcuts for isolated sub-problems.

This work presents an attempt to look at the problem of computer vision from the ground up, find what are the questions that need to be answered and propose some solutions to a small set of these questions. Note that when we say "from the ground up" we do not mean starting at the level of neurons in biological vision systems. That might be one method of *implementation* (chosen by nature) out of many possible methods. When we say from the ground up we refer to the types of questions posed. A question might be "How can we expand a method of tracking an object in cluttered scenes to tracking several objects?". But the questions that

in our view should be asked prior to that are of the sort "What constitutes an object?".

Needless to say, the attempt largely failed, in unison with the tradition of the field. But we hope nonetheless that we developed some useful techniques on the way, which alleviate some of the problems that plagued earlier systems and which might indicate possible routes for further developments.

1.1 Context of Work

We place our work in the broader context of mobile robotics and mobile manipulation (which nowadays might be called embodied cognitive system, but without even a hint at a proper theory of cognitive processes we rather avoid overstretching that term). This context defines the types of scenes to be expected and the nature of visual features in these scenes. Concretely we expect indoor scenes such as offices or apartment rooms, containing lots of man-made structures. The fact that these are more structured and controllable than outdoor scenes can (but need not necessarily) make our task easier, but of course carries the risk of leading us in a wrong direction. We must be careful to be lured into over-simplified problems and solutions.

A common characteristic of many structures in the above scenes is that they are rigid and opaque and thus have well-defined boundaries, sometimes (but not necessarily) definable by simple geometric properties. These boundaries when imaged using a perspective camera give rise to discontinuities or edges. We are therefore mainly concerned with processing of edge information, namely the grouping of edge segments based on principles of perceptual grouping and Gestalt theory.

Note that the situation would be different for e.g. medical imaging, natural outdoor scenes or aerial imaging. Different characteristics of the depicted scenes and the imaging process may make other methods, such as region based methods, better suited in those cases and we do not claim that our proposed edge-based approach is applicable to those domains.

In the following when we mention vision, we mean computer vision or machine vision. If we talk about human or animal vision we will indicate that explicitly. Furthermore vision system will be abbreviated by system. Also the terms perceptual organisation, perceptual grouping and Gestalt theory are used synonymously. When reviewing related work by various authors one will come across these different terms, which essentially mean the same. "Gestalt theory" is typically found in psychological literature while "grouping" tends to be used when it comes to the algorithmic level.

1.2 What is Vision?

Quite generally and informally the approach to solving a problem could be divided into the following steps: What is Your problem, what do You want to solve? How do You model the problem? How do You implement a solution using that model?

For the above example of control theory the answers to the above questions would be: What is control? There is a supplier of some resource (e.g. force, torque, current, voltage) and some user of that resource (e.g. load on motor, some electrical circuit drawing current). Control serves to balance supply and need of that resource. How do You model control? You can use Laplace transformations and Bode diagrams to design controllers, You can use phase space diagrams to study stability of control systems etc.. How do You implement a solution? You can use mechanical implementations like the centrifugal governor, time-continuous electronic controllers, digital controllers, fuzzy controllers and so on. The multitude of seemingly totally different solutions is held together by one coherent theory.

Applied to the problem of computer vision we would pose the following questions: What is vision? What is our theory of vision? How can we implement that theory in a technical system? Unfortunately already the first question is unanswered: What is vision? We can state what it is not: Vision is not edge detection, vision is not region segmentation, vision is not recognition, vision is not 3D-reconstruction, vision is not tracking, fill in any vision technique You like (or dislike). All of these do (or seem to) happen somewhere along the way. But here we are already at the implementation side of things. This multitude of methods are all different aspects of ... - well and here we lack of course the theory. But sit back and reconsider the first basic question: What *is* vision? (Or more generally what is perception?)

The answer is at the same time obvious and elusive. Well, vision is to see ... things ... . But what would we call a thing, as opposed to not a thing? How many things are there? Are things always the same? Are there always things? As a first working hypothesis we might answer: Vision is making sense of what You see, in terms of what You need.

Probably we might want to extend that to: Vision is making sense of what You see in terms of what You know and what You need. There will certainly be knowledge in the form of what You saw earlier or in the form of a priori knowledge such as physical properties. But it is important to keep in mind that vision is not about the application of knowledge (as in comparing stored patterns to current input). It is about making sense of that knowledge together with current visual (and possibly proprioceptive and other) input.

Coming back to "things", we can say that the introduction of the concept of things probably is a useful abstraction to structure the process of vision. Typically we can chunk the world into distinct bits that have specific uses (serve specific needs). So ordering (and therefore seeing) the world in terms of things seems to be useful and could be an aspect of a possible theory of vision. Seemingly our own

(human) visual system is structured such that we at least typically speak about things (objects). So there is some truth in the above first (the obvious) answer. But vision is not about the things. Rather things are a by-product. A theory of vision should be able to explain why things are a useful concept rather than starting from things.

"... in terms of what You need" implies that vision plays a role within a larger system. In order to clarify the above arguably vague statements, we will next consider the role of vision in our chosen context of mobile robotics.

1.3 The Role of Vision

We will now look into some aspects of the role of vision in the context of mobile robotics scenarios. What is the role of vision in that context? And what is clearly not its role?

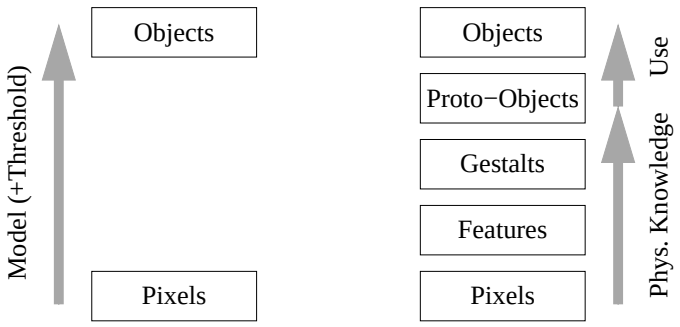
Why Object Recognition Doesn't Make Sense

A good deal of work in computer vision addresses object recognition. Recognition of already seen objects is certainly important but cannot be the only means of seeing. Not all objects can be known beforehand. A system must also be able to *see* objects it has never seen before. An object recogniser is blind for almost everything except the limited class of objects it was trained on.

Generally object recognisers make the mistake of attaching meaning to visual percepts too soon, often in one step (see Figure 1.1). An input percept pattern is compared to stored patterns, the equivalence compared to a threshold and a true/false answer is given. The stored patterns are typically labelled as "car", "cow", "mug", "face" etc. . Attaching those names attaches the meaning that those stored percept patterns have for us (the designer or user of the system). The result is that a car detector will always detect cars, a face detector always detect faces and so on. False positives are pruned by the designer by adjusting thresholds. A structuring of percepts according to *Sinnhaftigkeit* (whether they make sense) is omitted in those systems.

On the contrary we argue that a vision system should not be an object detector but a "sense detector". The vision system can not know and therefore must not decide by itself what is an object. Decisions are always dangerous. If You can't make them, don't make them (And certainly don't introduce thresholds to force them!). What is an object is solely defined by its meaning (in the current task, situation, context). And this meaning is introduced from elsewhere. So as far as vision is concerned, there are no objects.

Consider a chair. You would probably regard it as an object to sit on. A carpenter would maybe see it as many objects. A small vacuum-cleaning robot would perhaps see its four legs as four obstacle objects and so on. We always



(a) Object recognition (b) Levels of abstraction

Figure 1.1: From pixels to objects.

decompose the world into objects as we see fit in the current situation. Detecting objects is therefore an ill-defined task for the vision system alone.

If it were indeed the vision system's task to decide about objects, then we are done. Such a vision system is trivial. It is a one-liner. Let us construct one. Upon being asked "What do You see?" the system will answer "The universe." and it will always be correct. That is too coarse an answer? Well, then "A hydrogen atom.". Also always correct. No, no that is too fine. You mean something in between - ah! *You mean*. There we are again.

So the system only orders its visual percepts such that they make sense (considering physical properties of the world and the imaging process). This Sinnhaftigkeit is defined by Gestalt principles. Rather than saying "This is a mug." the vision system would say "This is something and grouping it made sense in describing (explaining) the image.". It makes sense insofar as image features are grouped which are likely to belong to the same physical object. This means that the system orders its visual percepts such that it could form objects, so that it can answer questions about objects, where objects are defined in terms of the Gestalt principles that the system knows. But the system does not know anything about objects before being asked. Objects are introduced into the system by the questioner and no sooner.

Also Pylyshyn [Pyl00, Pyl01] argues that we "... have reason to suspect that although the visual system does not 'know' about physical objects, it nevertheless can track certain visual patterns that are typically associated with physical objects.". And furthermore "... Thus, on initial contact, objects are not interpreted as belonging to a certain type or having certain properties; in other words, objects are initially detected without being conceptualized."

Where Does Meaning Come from?

Very well, this all leaves the vision system out of the tricky task of making decisions, as objects and meaning come from elsewhere. But where then does meaning

come from? Actually this question does not bother the vision system, but we will consider it nonetheless because it is of relevance to the overall system, namely the mobile robot in our scenario.

For the robot the need for meaning, and thus the need to talk about "objects", only arises when something appears in several sensor modalities. Say the robot can grasp something in a particular way and it looks like this. Then *it* has a meaning in two sensor modalities, namely grasping and seeing. *It* deserves being an object, and we call it "mug". The co-occurrence of that percept in two sensor modalities lends meaning to the source of those percepts. The meaning actually *is* the fact that it does appear in two sensor modalities. Only if it appears in two sensor modalities does it have a meaning. And this meaning must be given a name to reference to it over different sensor modalities. Objects are the names for meaning. Objects carry meaning.

Meaning arises when two sensor modalities mean the same thing. Then that thing has a meaning and therefore deserves being called an object and be given a name e.g. "mug".

But still, after the vision system has given the answer "This can be a mug", who decides whether it is actually a mug? Whether it is enough of a mug to be considered one? The answer is that if it looks like a mug and i can grasp it like a mug, it *is* a mug for me. Period. No thresholds. Because the meaning "mug" is only introduced by the fact that it looks so *and* I can grasp it so. If it looks so and I can grasp it so, it *means* a mug to me.

Now we have moved the responsibility yet another level up and the question remains: Why is there something like a mug? How did the meaning "mug" arise? Answer: From the purpose of the overall system. Because the human user wants the robot to be able to carry mugs. Because to the human user a mug has a meaning. The purpose attaches meaning. This meaning or object "mug" is mapped into the sensor modalities of the robot and referred to as term "mug". (The logical next level of what the mug means to the user is left to Philosophers.)

To relate the above rather abstract statements to real life and hint at biological systems let me give a small example. (Note that I am well aware that the following is just an anecdotal example and not a proper experiment.)

My cat loves to throw things off my cluttered desk whenever I sit there. She places herself on the mouse pad, studies her surroundings and starts pushing whatever objects are in reach until they fall of the table, which greatly amuses her. Recently she did that with the Compact-Flash slot cover of my PDA, a squarish, flattish grey plastic thingy. She has never seen it or something visibly similar before. She could not *re*cognise it as an object, because she has never seen it before. Yet amongst all the clutter her visual system (and only that!) identified *it* as *something*. Namely as something that can catch her attention, without knowing what it is. To put it another way, she did figure-ground segmentation without having objects. Note that it has no meaning yet, it is no object yet. As soon as

she pushes it, as her paw physically interacts with it, it becomes an object. It will get the meaning "toy". After playing she will recognise it next time, she will have learned it and stored it under the category "toy" (she has in the meantime learned rubber, pencil, paper clip ...).

And what Does that all Mean for Computer Vision?

After this excursion into the deeper meaning of meaning let us come back into the realm of the technically possible. And as capabilities such as grasping and feeling are even less technically developed than vision, we have to be realistic and for now have to accept that meaning will basically be hard coded. I.e. we can not hope for now to put together a robot manipulator and a vision system, specify a task like "Grasp the mug" and hope that meaning and everything will magically emerge. Vision will be the only sensor modality that we deal with. And meaning in the form of object descriptions will just be handed to us. And from the point of the vision system that is perfectly alright, as we do not care where meaning comes from.

So what can we learn from everything said above for the construction of a vision system? It boils down to: avoid thresholds, defer decisions, structure data. The vision system should concentrate on the structuring of data for higher level processes rather than trying to make all decisions by itself. It should structure input data so that it makes sense by itself without attaching meaning to it.

Why Gestalt Principles?

Vision has to infer from a two-dimensional image something about the three-dimensional world. This problem is ill-posed, as the perspective projection of a camera cannot simply be reversed. Therefore heuristics are needed to arrive at an approximation. What information can these heuristics exploit?

First the physics of the world. Our world consists mostly of solid opaque things, with smooth surfaces, with different shapes, colours, textures. Different worlds are conceivable. Maybe one which solely consists of spheres of different colour. Perceiving form in this world would be irrelevant, because all things have the same form anyway, form does not help to distinguish between them. One would only need to perceive colours (and perhaps centre points). In our world however things have different shapes, surfaces have distinct boundaries etc..

Second the physics of the imaging process. We are using cameras which project light onto photosensitive surfaces using lenses. Again different imaging processes are conceivable. For example we might observe things in a cube, where the cubes walls are covered with photosensitive surfaces (a strange camera indeed). One would see all sides of things simultaneously etc. . But we happen to use normal cameras, with a pinhole model, with perspective projection. Occluding edges of objects project to intensity (and colour) edges on the image plane.

Third the definition of objects. In our world (or actually our human use of it) typically that what we call objects is defined by its use, or function. This function of an object furthermore is typically defined via its form, or structure. This need not be the case for all objects. For example the function of a traffic light is defined by its colour rather than shape. Still most objects are defined via their 3D structure (cups, chairs, scissors, cars, human bodies). So perceiving objects comes down to perceiving 3D structure. We cannot do this directly because the imaging process projects onto a 2D plane. All we can do is trying to infer 3D structure from 2D image structure.

Gestalt principles do just that. Given what we know about the physical world, definition of objects therein and their projection onto images, Gestalt principles describe the most likely guesses we can make about how the image corresponds to the physical world. They group parts that exhibit visual structure in the image which is likely to stem from 3D structure in the physical world. Perceptual grouping can be seen as a reasoning process on a geometrical level. When image elements are perceptually grouped there is *reason* to belief that they belong to the same physical object. Perceptual grouping is therefore not just a means of data reduction to extract more compact representations (as e.g. dimensionality reduction with principal component analysis and similar techniques). It is a more active process leading to Gestalts which are not just more compact than their originating data but also more than their parts.

Furthermore in order to be effective these Gestalt laws should concentrate on those parts of an image which carry most information. This data reduction, or first abstraction is performed by extracting certain image features. So Gestalt principles operate on abstractions, typically edge and point features, or possibly higher level abstractions already grouped from lower level features.

A good justification for the emergence of Gestalt principles is given in Albert [Alb01]. The author explains Gestalt principles as expressions of the generic view principle (GVP): Vision assumes that qualitative (topological) image structure is stable with respect to small changes of viewpoint. This is another formulation of the Helmholtz principle [Hel67] which states that we perceive what is most likely under normal viewing conditions.

Similarly Lowe [Low87] lists the following two constraints for perceptual grouping:

1. The viewpoint invariance condition: Perceptual features must remain stable over a wide range of viewpoints of some corresponding three-dimensional structure.

2. The detection condition: Perceptual features must be sufficiently constrained so that accidental instances are unlikely to arise.

Sarkar and Boyer [SB93, SB94] recognise the following three characteristics of perceptual organisation:

1. It is essentially symbolic, with a top-down (attentive) and a bottom-up (preattentive) component.

2. Organisation proceeds in a hierarchical fashion, with coherent global structures gradually forming from local features.

3. Organisation produces an information reduction in the encoding of a given structure due to the hierarchical nature of the abstraction.

On Learning

Learning plays an increasingly important role in computer vision. Be it learning of object classifiers, colour models or receptive fields. Is not a system which does not learn, as we are going to propose, a step backwards? Why do we leave out learning? Actually we don't.

Looking at nature and the vision (and of course other) systems it developed, we can basically distinguish between two types of learning: evolutionary learning and individual (epigenetic) learning. Evolutionary learning took place over millions of years and produced all those abilities which are innate in newborn individuals. Individual learning happens as an individual interacts with its environment. How much visual capabilities are innate vs. learned is still an open question. We just want to briefly mention some related work, far from an exhaustive literature study, only to indicate the wide array of possibilities and sometimes contradictory findings.

Several studies show that indeed even very basic visual capabilities found in the primary visual cortex (V1) are to a large degree learned rather than intrinsic or innate. Rearing kittens in a visual environment consisting of alternating black and white stripes restricts pattern vision, and seems to shift the orientation preference of cortical cells towards the experienced orientation [BC70]. Moreover it seems that self-induced motion, rather than passively experienced motion is necessary to develop a fully functional vision system [HH63]. However the results could be explained as disturbed maturation. Much as the muscles in the cats legs will not develop properly, if it is strapped to the floor during its infancy - a general efficiency principle of nature ("If You don't use it, You lose it."). Yet we would not say the cat learns to grow muscles for walking (It will however have to learn *how to use* these muscles).

Studies by [SAS00, vMPS00, SGR88] show the impressive plasticity of the brain even for seemingly "hard-wired" abilities. In experiments with new-born ferrets the authors routed fibres from the retina to the auditory pathway, such that the primary auditory cortex (A1) now received visual rather than auditory input. Remarkably, structures and receptive fields very much similar to those in V1 then developed in A1, namely sharp orientation selectivity of cells (orientation columns), orientation maps (which were however less orderly than in V1) and long-range horizontal connections typical for V1. Furthermore, behavioural experiments

suggest that the ferrets could actually see (rather than hear) with the 'rewired' auditory cortex, i.e. behaved as they would normally behave in response to visual percepts. These results suggest that the function and even sensory modality of cortical regions is instructed to a significant extent by its extrinsic inputs during development.

On the other hand [CK99, KC02] show a different picture with regard to ocular dominance columns. Cortical cells are activated differently by the left and right eye (ocular dominance). Cells with similar eye preference are grouped together into columns, and eye dominance shifts periodically across the cortex. The initial establishment of ocular dominance columns traditionally is believed to be based on patterns of correlated visual input. However, in experiments with cats, monkeys and ferrets the authors show that ocular dominance columns can develop even in the absence of any retinal inputs. This indicates a role of innate mechanisms.

Also without going into neurophysiological details, we can see quite different mechanisms by generally looking at altricial species (such as cats) and not so much to precocial species (such as gnus). The former are born rather immature and helpless and depend on the care of their parents for an extended period of time (the extreme case being marsupials), while the latter are born more or less fully developed and can for example walk with the herd only hours after birth. The visual capabilities of a newborn gnu must be developed far enough for it to be able to walk on possibly uneven ground fast enough to catch up with the rest of the herd. It seems unlikely that these capabilities are learned within only a few hours. Why nature chose pre-configured capabilities in precocial species and individual learning of basically the same capabilities in altricial species is an interesting (and open) research question of its own but not our concern here. The bottom line is that there are examples of highly developed (mammalian) biological vision systems which are fully developed at birth.

Sighted animals (humans, cats, birds, cows, etc.) perform so astoundingly well in navigation and manipulation, that we are forced to believe that they build up some (limited) form of representation of the environment in which they move and act. This can not be achieved simply by learning with many repetitions of every single aspect of that environment. ("I have bumped into that sort of visual percept so and so many times, so it must signify an obstacle. I have died so and so often whenever I saw something like this, so it must be a cliff."). Some more general perceptual ability must be at play. Note that of course animals do learn what are dangerous or beneficial situations. But as we would argue, on a more abstract level of *situations* rather than directly visual percepts.

So the issue of innate versus learned, or put another way, the relative contributions of intrinsic and extrinsic factors to the functional specification of cortical areas, is still unresolved.

Another, more theoretical point about learning can also be made. Suppose a system is in a state where it is has a certain capability. How that capability was

acquired does not matter. Whether we as the designers created the system in just such a way that it has said capability or whether we equipped the system with a learning algorithm and training data and let it learn that capability, the result is the same. If we know how the system learns (which we do, since we equipped it with the learning algorithm) and if we know the training data (which we do, since we provided it) then we will know the systems state after learning. So we might just as well create the system in that state. From a theoretical point of view there is no difference.

From a practical point of view of course the learning system seems more appealing. It can be adjusted more easily to different environments, tasks etc.. We however choose the simpler approach of directly equipping the system with its visual capabilities. Why simpler, one might ask. In both cases, *what* the system is supposed to do is given. In the direct case we explicitly define *how* the system does it, making sure that the system does what it is supposed to do. In the learning case we have to make sure that the learning algorithm together with the training data really implements the desired capability, a task which has proved to be not always that simple . Even if learning is to be incorporated at a later stage, starting by explicitly implementing what the system is to learn, will make the learning problem a better defined task.

Too often research happens the other way round. Somebody might take some learning algorithm that seems sort of promising or fashionable, let it learn with data that seems quite reasonable and then claim that the outcome is some approximation to the desired capability. That however amounts to alchemy rather than a scientific approach and little or no insight into the problem of vision is gained (see own work in Chapter 3).

See however e.g. Fidler et al. [FBL06] as an example of how (some) Gestalt principles can emerge from a statistical learning framework. Or Olshausen and Field [OF96] who explain receptive fields as learning sparse codes and Bell and Sejnowski [BS97] who use independent component analysis instead. So we see that receptive fields as well as Gestalt principles (at least rather local ones) can very well be learned (or explained as being learned). And this is a well-defined learning problem, because we know beforehand (from neurophysiological or psychological findings) exactly what we want to learn and when we have successfully learned it.

So in some sense our system does use learning, namely the Gestalt principles, which are general purpose visual concepts learned by evolution and discovered by psychologists and learnable using e.g. the above methods. We however concentrate of the use of these principles and accept them as given to the system in one way or another.

Limits, Complexity and Anytimeness

Sometimes the complexity of an algorithm will place a limit on e.g. the number of features one can process in a given time. “On a xyz computer we can process

10 items per frame." is typically followed by "But given that computer speed doubles every year or so . . . ", which serves to justify the "slowness" of the proposed method. But this is in fact the wrong answer. No matter how fast a system may be, there are always limits on how much information it can process "simultaneously", the human visual system being no exception. We can only follow so many moving objects or perceive so many objects at the same time (e.g. four [SX01]). We can not expect a computer vision system to see everything at once in all detail. The achievement of biological vision systems is in fact how to deal with such limitations by use of attention mechanisms.

We do not expect the system to analyse an image and immediately reply "There are 457 objects. Object 1 is a pencil, object 2 is a door, . . . object 324 is a cup, . . . ". What we do expect is that if there is e.g. a cup covering a large part of the image, the system should report immediately "There is an upright rotationally symmetric object and . . . wait . . . a horizontally elongated object to the left and . . . wait . . . a rectangular object behind, . . . ". Big, obvious, easy objects should pop out fast. More detail should only be reported later as time permits. We refer to this ability as *anytimeness*. The system should be able to deliver *some* result at *any time*. And this result should be the visually most significant one.

1.4 Requirements

This directly leads to a list of requirements we have for our proposed system. Having set out our goal in the above sections, what do we feel is necessary to arrive at a solution?

Explain the image Not in terms of objects ("I see a table with a mug on it") but in terms of what makes sense, what is possible. This is our high-level goal. Though it remains to be seen how high we get in that explanation.

Anytimeness The system should produce results anytime. The more processing time is granted the finer the results will be. Most prominent results should pop out very fast.

Graceful degradation We should not have situations where the addition of noise (however nastily placed) inhibits the creation of a good hypothesis. Delaying, yes, that is very much acceptable: The more noise, the more difficult (= slow) is the extraction of good hypotheses (anytimeness). Even more so if the (nastily placed) noise appears to form good hypotheses (appears to be locally consistent) at first. Then only a more global consistency check can decide what is noise and what is actually a good hypothesis. (Noise is that which does not take part in (is not needed to create a) consistent global picture. Noise is "left over"). This means that noise can not simply be thresholded away locally, which leads to our next requirement.

No thresholds Usually there is only one way to set a threshold, namely *wrong*. The system performance shall not depend on correctly tuned settings of various thresholds, which has been the bane of countless approaches before.

Single/Multiple cues The proposed system must be able to succeed on the most primitive type of image, namely a single, monocular, grey level image. It must not *require* more if it is ever going to be successful at all. Any system that requires more than a grey level image is flawed be design, because all the structure the world exhibits is present in a grey level image. A colour or depth image does not contain more or does not contain a different kind of structure. If a system can not make use of the structure in a grey level image it can not make use of any structure at all. And colour or depth won't change that. The system should however be able to incorporate more image cues if available. Such cues are the mentioned colour, depth, motion or texture. The incorporation of more cues should make the system more robust or perhaps faster, i.e. should improve performance. Note that I am well aware that 3D information such as depth images are a richer depiction of the scene than a grey level image. But first of all 3D sensors are (at the time of writing) still not widely available, making experiments difficult and expensive. But the main argument is that just *more* information will not save us. We still require the proper methods to process that information. Now if such a method is powerful enough to even work on the simplest, poorest type of information (a grey level image) it should of course work the better on richer information. In that sense the poor information drives us to perfect the method rather than enhance the available information. (So the argument "But in so-and-so many years we will have laser range scanners with 1 mm resolution and then our system will work." is invalid. Much as the argument that "in so-and-so many years computers are 1000 times faster and then our system can work in under 1 second...").

1.5 Thesis

By now it must seem to the reader that we are claiming to have solved all the shortcomings of vision systems so far, which is of course not the case. Also the presented system has many severe limits and only partly works in very limited scenes. We believe however that we are tackling the right sorts of questions and can provide some insights into possible solutions. Concretely, the thesis put forward by this work is as follows.

1. Recognition of known objects is not enough. We need a means of extracting information at a level of abstraction before objects.

2. Perceptual grouping is such a means, which takes into account the structure

of the physical world and of the imaging process, without explicitly referring to objects, by grouping what makes sense. Perceptual grouping is well studied in psychology and known to underlie (parts of) human vision.

3. Perceptual grouping can be implemented efficiently, dealing with cluttered real world scenes.

4. Perceptual grouping can be based on clear probabilistic principles.

5. The results of a perceptual grouping process can be used to eventually make statements about objects, once the notion of object is introduced from outside.

The remainder of this work is organised as follows.

Chapter 2 reviews related and inspiring work in the area of perceptual grouping and extraction of 3D shape (which is ultimately the goal of vision, though not in this particular work).

To make a clear point about learning and object recognition, the temptation of easy successes and the inherent limitations of such approaches Chapter 3 presents an example of an object recognition system based on Support Vector Machine learning. Although the results will be seen to be promising we will also realise at the end that we are no wiser with regard to a theory of vision.

Chapter 4 introduces the principle of non-accidentalness and proposes significance as a quantitative measure, which will be used throughout the remaining chapters.

Chapter 5 addresses the problem of detecting ellipses in images, where perceptual grouping will help to tackle complexity in cluttered, real world images.

Chapter 6 revisits the well studied problem of finding closed convex contours in an edge segmented image. We look at these issues with a different paradigm in mind: Incremental processing helps avoiding parameters and leads to anytimeness.

Chapter 7 finally puts contours detected in the previous chapter into a global context and aims to find a globally consistent ordering of surfaces in depth.

In Chapter 8 we will look back at the presented work and discuss how many if any at all of the goals and requirements we set out in this introduction we could actually meet.

Chapter 2

Related Work

We are not going to elaborate on Gestalt principles themselves here. We assume that the reader is sufficiently familiar with the basic concepts. For an excellent introduction into the topic see e.g. Rock [Roc97] and Palmer [Pal99]. Instead we are going to concentrate on computational aspects and implementations.

Also see Boyer and Sarkar [BS99] for a concise overview over the state of the art in perceptual organisation. It lists approaches along two axes, using the classificatory structure presented in Sarkar and Boyer [BS93]: dimensionality of the sensor signal (2D, 3D, with or without motion) and level of abstraction (signal level, primitive level, structure level and assembly level).

Interpretation of line drawings to extract 3D information has a long history in computer vision, much of it owing to the seminal work of Marr [Mar82]. But there were earlier attempts, for example the work by Roberts [Rob65], a system to detect known wire-frame models of 3D objects in images. Naturally (owing to the limited computing power of the time) the objects were very simple and images very "clean".

An early system for globally consistent interpretation of scenes without the need for known object models was proposed by Waltz [Wal75]. It aims to infer a 3D interpretation from line drawings of scenes containing trihedral objects (i.e. polyhedra where no more than three faces meet) with shadows and extends an earlier line labelling system by Huffman [Huf71]. Huffmans approach posed the labelability problem of a line drawing as a satisfiability problem for a set of boolean expressions, which is NP-complete. Waltz overcame those problems by introducing a filtering scheme (Waltz-filtering) and reduced run time to approximately linear in the number of lines. The basic idea is to achieve local consistency in the following way: given a junction, rule out all legal labellings of the junction for which there is no labelling of the neighbour junctions which is compatible with it. Repeat this procedure until no further progress can be made. Furthermore Waltz added illumination information to achieve a finer labelling of lines as well as distinguishing between interior lines and foreground/background lines. The system also deals to a limited amount with missing edges and unlabelable accidental alignments. In

the latter case virtual eye movements are used to transform unlabelable accidental alignments of junctions into common cases.

Of course the idealised assumptions built into the system are very strong and prevent any use in all but the simplest, cleanest scenes. Even if one does not mind that the system can only correctly interpret (i.e. *see*) trihedral objects, the system is sufficiently confused by non-trihedral objects or surface texture that it subsequently fails to see the trihedral ones. Also the assumption of a single, directed light source which casts sharp shadows is too unrealistic for real scenes. There is also a strong notion of background, which is supposed to be a horizontal table large enough to fill the entire image (i.e. the table itself causes no edges). But actually every object or surface can be foreground or background for another surface.

On the implementation side the very elaborate enumeration of possible cases which have to be considered leads to a proliferation of junction labels (up to 3256!) and becomes increasingly unfeasible as idealising constraints are relaxed. The main difficulty is certainly that the system aims to reconstruct a full 3D description directly from lines in one step. Waltz however mentions the use of eye movement, stereo or depth images to resolve ambiguous labels. It seems that 3D information should rather come from these sources than solely form line labelling.

A tentative notion of affordance "supports" is introduced which is deduced from edge labels. Again this is a very local decision based only on line labels and suffers from the many possibilities of label combinations. The most interesting aspect of the work is the simple, yet powerful filtering scheme, which transforms an NP-complete problem into a (mostly) linear one. The scheme is however still very "crisp" and can not itself deal with ambiguities and uncertainties.

Barrow and Tenenbaum [BT78] introduced the notion of intrinsic images. Assuming known lighting conditions (ambient light plus a point light source at infinity) and smooth Lambertian surfaces of constant albedo they propose a method to derive from an intensity image intrinsic images of distance, reflectance, orientation and illumination. Constraints on local surface orientation at extremal edges (contour where the smooth surface turns away from the viewer) are propagated inwards to interpolate intrinsic properties of regions. In [BT81] the authors also propose a technique to extract polyhedra from line drawings based on assumptions of regularity. Although the approach is very clean and elegant, the very limiting assumptions make it interesting more from a theoretical point of view. It might be interesting however to combine their work with a perceptual grouping step to produce the very "clean" line drawings their approach requires.

Malik and Maydan [MM89] [Mal87] present a system to recover a three dimensional interpretation of a scene composed of piecewise smooth surfaces from a single edge-segmented image. They combine constraints from a (possibly ambiguous) line and junction labelling step with shape from shading algorithms. The approach however depends on unrealistically strong assumptions, namely no sur-

face markings or texture, a known reflectance map (e.g. Lambertian surfaces with a known point light source) and perfect edge segmentation.

The theory to use structural representations was advocated by [Bie87]. Objects are described as being composed of one or several out of 36 qualitative volumetric primitives called GEONS. See [DBB+97] for an overview of recognition by component approaches. Typically these approaches also suffer from the necessity of a very good line drawing extraction which makes them difficult to apply in real world images, where perfect segmentation can never be assumed.

The SCERPO system of Lowe [Low85, Low87] used perceptual grouping of possibly poorly detected lines in real world images to overcome combinatorial explosion when matching 3D wire-frame models to those lines. Scale independent measures for the significance of proximity, parallelism and collinearity are based on geometric and probabilistic considerations. The significance of a grouping is based on the probability that it is due to actual structure in the scene and not to an accident of viewpoint. Concretely the significance is the inverse of the probability that a grouping arises accidentally, where a uniform distribution of lines in the image is assumed. Furthermore an efficient implementation is proposed for finding close lines based on line endpoints indexing a 2D grid. Higher level groupings such as trapezoids and ribbons (joined pairs of parallel lines) are found by combining the lower level groupings, where the significance of the high level group is simply the product of part significances. The most significant high level groupings are matched against the model base and subsequent pose estimation aligns corresponding image and model lines. According to the author this procedure is successful as long as the complexity of the groupings is sufficiently high to be discriminative for the model base. Impressive results are obtained in cluttered real world images with models of significant complexity.

In [SB93] and [SB94, SB95] Sarkar and Boyer describe a framework for perceptual organisation consisting of a preattentive part based on voting and graph-theoretic methods and an attentive part based on Bayesian belief networks. Their system detects structures such as ribbons, rectangles and ellipses in real-world indoor and outdoor images.

We will first describe the preattentive part. The basic concept is to organise tokens (such as lines) of one hierarchy level into groups and represent such a group along with its emergent properties by a more complex structure (a pair of parallel lines), which is then again a token at the next higher level.

At the heart of their method lies a voting scheme. Tokens fulfilling a compatibility relation (e.g. lines being parallel) vote for the same bin in a parameter space. Thus for n tokens all compatible groups of tokens are found in $O(n)$ time rather than $O(n^2)$ as in exhaustive search. Voting always has the problem of finding a good bin size, balancing accuracy (which calls for small bins) with memory requirements. Especially in higher-parametric voting spaces (e.g. (x, y, θ, κ) for position, orientation and curvature of an edge segment) memory requirements grow

prohibitively fast for fine quantisations. Moreover, no matter how fine quantisation is, there is still a problem with data points near bin boundaries. Two data points might be very close just on opposite sides of a bin boundary. So although their distance is smaller than the bin size d, they are not grouped within one bin. The solution proposed by the authors is *offset quantisation*. A second voting space shifted by $\frac{d}{2}$ ensures that data points being separated less than $\frac{d}{2}$ are grouped in at least one of the two voting spaces. This idea is expanded to N dimensions and multiple order offset quantisation, using more than two interleaved quantisers. This approach allows rather coarse quantisation without missing compatible tokens. Once candidates of compatible tokens are found, exact fulfilment of the compatibility relation is checked in a subsequent screening process.

The parameters for the compatibility relations (e.g. when are two lines considered parallel) are chosen to reflect the least degree of significance allowable. Significance is defined as the non-accidentalness of a relation using Lowes formulation [Low85, Low87]. E. g. average line length is used to compute a distance threshold of 12 pixels or an orientation histogram to compute an orientation threshold of 15°.

Tokens and compatibilities then form attributed graphs, where tokens are nodes and compatibilities define edges. Each type of compatibility reflects a Gestalt principle and gives rise to a Gestalt graph (e.g. continuity graph, proximity graph). Gestalt principles realised are proximity (tokens having points close together with similar orientation), end point nearness, smooth continuity, similarity (of geometric or photometric properties) and common region (tokens bordering a common region). Graph theoretic operations such as finding connected components or cycles are used to create tokens at the next higher level.

So Gestalt principles are implemented in two stages. First, voting is used to associate pairs of compatible tokens, creating edges between graph nodes. Then graph theoretic operations are used to create Gestalts, i.e. meaningful wholes.

For the given example of an aerial image the following steps are performed. Individual edgels are grouped into straight lines using a voting space of x, y and gradient direction. Line fragments are grouped into continuous lines voting with proximity, orientation and edge contrast similarity. The continuity graph is searched for connected components to replace series of continuous lines by a single line. Next apars (antiparallel lines, i.e. parallel lines having opposite contrast polarity) are searched, using proximity, orientation and edge contrast. Apars are further grouped into continuous apars using again proximity and orientation and further width and sign (whether the apar is brighter or darker than its surrounding). Continuous apars are grouped into papars (parallel apars, think of striped structures like zebra crossings). Furthermore, intersecting apars define rectangles. Using these grouping operations structures such as buildings, runways and roads can be reliably extracted from aerial images. Generalising lines to edges of constant curvature also allows detection of arcs and subsequently ribbons, continuous

ribbons etc. Further examples show detection of rectangular structures on the facade of a building and significant structures in an indoor scene.

The strength of the above method is its generality. The same voting scheme and graph theoretic operations are used on all levels for various kinds of tokens.

Before going to the attentive part of the overall system, we would like to add three remarks: First, continuations of lines are based on proximity. This is useful when "repairing" lines, fragmented due to the edge detection process. But when grouping lines fragmented by occlusion the gap between line endpoints can be arbitrarily large. Also the definition of apars favours close-by lines. Generally there is a tendency for local groupings. When manipulating objects on a table occlusions are inevitable (as opposed to the situation in aerial imaging). This might require grouping of features which belong to the same object, yet are not close but separated by an occluding object. Generally, the above approach targets "flat" scenarios where no occlusions take place. We want to explicitly model within the grouping process the three-dimensionality of the physical world, which means concretely that grouping principles make use of the hints provided by occlusion to infer 3D structure.

Second, the notion of apars is useful only for objects in front of a uniform background, such as dark roads within a brighter landscape or vice versa. For our typical indoor scenarios, backgrounds are rarely uniform. Background intensity can vary arbitrarily and so edge contrast polarity is inadequate for grouping edges.

Third, there are many parameters for grouping. These are selected adaptively, based on significance considerations and image statistics. But they are still hard cut-off thresholds. While two lines with an orientation difference of, say, 14.9° are grouped as parallel, lines with a difference of 15.1° are not. We advocate avoiding thresholds and instead keep multiple hypotheses which are ranked according to their significance. Hypotheses which are weak at a local scale (and would thus be pruned when thresholding) can still become valuable within a larger context at later stages.

In [SB93] the authors describe the attentive part of their system. A knowledge base encodes geometric figures such as circles, ellipses, rectangles and polygons. While the bottom-up processes of the previous paragraphs group candidates for such structures, top-down reasoning using the knowledge base enables to infer undetected features from partially instantiated structures, e.g. to overcome noise and occlusions. The basic formalism for reasoning is that of Bayesian networks. This formalism is modified to reflect properties of spatial reasoning.

Each node in the network represents a feature. Parent-child relationships define a hierarchy, e.g. closed curve, polygon, quadrilateral, trapezoid, parallelogram, where features become more specific by adding constraints such as parallelity or symmetry. In order to reduce the complexity of the network, composite nodes represent not a single feature, but a feature type. This reflects the fact that all features of a type share the same neighbourhood structure (e.g. a quadrilateral

has the parent polygon and is itself a parent of trapezoid). Each composite node internally has a list of instantiated feature hypotheses. Such a modified Bayesian network is called a perceptual inference network (PIN). The structure of a PIN is fixed and independent of a image data. Image data just adds instances within composite nodes.

The conditional probability of a feature given its parents is defined as the product of two functions: the locational compatibility function and the conditional belief function. The former is a binary valued function which is 1 only if the constituent features of a given feature are compatible. This compatibility is checked point-wise in the image and basically constitutes yet another voting procedure, albeit the authors do not point that out. A threshold of 5 pixels is used, but the authors show that the system is robust against the exact choice of that value. The conditional belief function is a sigmoid function which measures the degree to which the constituent features match the definition of a given feature (e.g. the degree of parallelity, closedness or symmetry). The authors chose an exponential $1 - e$ function and again show that the exact choice of the function does not matter. This is important, because one can never expect to model probabilities exactly. But the authors could show that any sigmoid function suffices, whether exponential or part-wise linear. The authors further show that the choice of prior probabilities for the root nodes of the network does no influence the final result, which justifies the assumption of equal priors.

The higher level feature hypotheses detected by the PIN system are ellipses, circles, ribbons, (equilateral) triangles, trapezoids and parallelograms. The authors propose to then apply photometric or other more expensive custom feature detectors to verify hypothesised features.

We make two remarks: First, the attentive part of the system is meant to serve as a top-down knowledge base. A PIN describes what are the structures we are looking for (e.g. parallelograms, ellipses) and how they are formed by their constituent parts. But so also do the voting and graph-theoretic methods of the preattentive system. The benefit of switching from voting and Gestalt graphs to the PIN formalism at a certain level is not entirely clear. It seems that the structures detected by Bayesian reasoning over PINs might just as well be detected by yet another voting process.

Second, the authors make no attempt to find a global most likely interpretation. All feature hypotheses are retained. But certainly some hypotheses have higher beliefs than others. These could be used to explain away unlikely hypotheses. For example their indoor experiment shows many intuitively wrong ellipses. While locally (within the ellipse composite node) there might be evidence suggesting such ellipses, but attributing the underlying constituent features to other nodes (such as trapezoids) with higher belief values would be a globally better interpretation. The authors however leave that step to external custom feature detectors. We feel that it is one the strengths of Bayesian belief networks to resolve such ambiguities.

Stricker and Leonardis [SL95] propose *Exsel++*, a general and expandable framework for accurate and robust extraction of parametric models of different types from a set of boundary elements. They make no assumptions about the type of data, i.e. sparse or dense (though it seems they prefer sparse data) or generally about the domain of application. Any model type (lines, ellipses, planes, spheres etc.) and fitting technique can be plugged into the framework. There are four steps: exploration, selection, fit and a final selection. The interesting parts are exploration and selection. Exploration can be thought of as a RANSAC [FB81] with short term memory for masking already explained data. When choosing a new random sample to produce a hypothesis, only such boundary elements are allowed which have not been chosen recently and have not been supporting another hypothesis recently (where recency is defined via expiration rates). This dynamic data driven approach minimises the probability that the same hypothesis is generated twice while still maintaining the possibility to return to any part of the data and find a possibly better hypothesis. Selection chooses those hypotheses which explain the input data in a globally optimal way. Optimality is formalised as a Minimum Description Length (MDL) principle: that set of hypotheses which overall explains the data with the shortest possible encoding is optimal. This essentially means that hypotheses with many supporting elements, high goodness-of-fit and low number of parameters are selected. The combinatorial explosion (for n hypotheses where are 2^n different subsets) is handled via tabu search [GT93]. Tabu search is a heuristic procedure for a discrete global optimisation and can be described as an extension to steepest ascent. When exploring the search space, moves which have been executed recently are marked as illegal. This enables escaping local maxima.

One of the strengths of the above method can also be regarded as a weakness, namely not exploiting the density of the data if available. The selection is global and does not take into account any local connectivity, when in fact this is often readily available, at least in images of natural scenes taken by perspective cameras. Typically edge elements are not isolated but form connected strings of edgels, with occasional gaps. Note that the authors give only toy examples with four and ten models present and two types of models for demonstrating the idea of the method. In natural scenes we can expect thousands of models of dozens or hundreds of different types. As a weakness of the above method we also mention the necessity to choose parameters (expiration rates for selection and tabu search, weighting factors for the MDL objective function).

The work of Ackermann, Schlueter et al. [AMP+97, SWSP00, Sch01] presents an approach combining perceptual grouping with a more global reasoning based on Markov Random Fields. The system is based on contour segmentation (edge detection) and perceptual grouping according to the Gestalt principles of good continuation, proximity, parallelity, closedness and similarity. Grouping is hierarchical and recursive, complex groups can themselves again be grouped in linear

structures according to good continuation. Contour segmentation is augmented by colour based region segmentation where regions hint at missing contour fragments. On the other hand grouped contours are used to connect regions separated by occlusions. This is used to aid a region based object recognition. A Markov Random Field (MRF) is used to get from local grouping hypotheses to a globally consistent interpretation of the image. (A MRF is also used to segment contours in image sequences, but we will not go into detail here.)

Using a clearly defined setting (the Baufix domain - brightly painted wooden toys, with mostly uniform grey backgrounds) contour and region segmentation are rather simple, depending on thresholds which seem only suitable for this specific domain. Grouping of edge segments is limited to neighbours within a certain area of attention. Probability distributions of e.g. line endpoints w.r.t. a given line are learned from hand labelled training images and normalised w.r.t. feature sizes. Thresholding then gives the areas of attention, where the thresholds seem to be chosen such that the resulting areas look "reasonable". Later during grouping some of these areas are further scaled with an "empirically" obtained parameter. So although the principal idea is good, the implementation actually amounts to ad-hoc setting of parameters. Correct settings of these parameters are critical as grouping is strictly limited to the regions of attention, i.e. everything outside is discarded.

Grouping proceeds in a hierarchical fashion. Contours are grouped into collinearities (generally co-curvilinearities) then into parallel and symmetric groups. Finally closures are defined from end-point proximities between these groups. Again various thresholds are employed to discard less significant groups (even though these hypotheses might become more significant in the later global reasoning step!). At the highest level the closures themselves can be further grouped into linear and similarity groups. The definitions of nearness and similarity between polygons for this step use several ad-hoc definitions, none of which works satisfactorily. So they are combined into a feature vector and the relative contributions learned from labelled training images. This highlights the difficulty of obtaining meaningful measures of proximity and similarity for features higher than simple contours. Collections of (however well motivated) ad-hoc measures break down once different types and levels of perceptual groupings are mixed

To increase robustness the authors propose to merge contour and region information. However merging of contours and regions only happens after their respective segmentations. Merging during segmentation would clearly be a better choice. Instead the method requires matching of separated information that was originally unseparated. Again several measures of similarity between contours and regions are defined, resulting in a feature vector, and a classifier is learned to distinguish between cases. And yet again many thresholds are needed.

The most interesting part of their work is the use of a Markov Random Field (MRF) for globally consistent interpretation. Hypotheses at all levels (single con-

tours to grouped closures) created in the previous steps form the nodes of an MRF. The labels of these nodes are discretised significance values, which seems an elegant choice. Two types of energies enter the optimisation procedure. Data terms are defined as the difference between the currently assigned label (= significance) and the data-driven significance as obtained from geometrical considerations during the grouping process. Clique potentials are formed for clique sizes of two and can either represent agreement or contradiction between two groups. Agreement exists between a group and its parts (e.g. a parallelity and its lines). If both significances are high, the resulting clique potential is negative, indicating a desireable state. If one significance is high and the other low, a positive potential indicates that one of them must be wrong. Contradiction exists between groups of the same type which share a part (e.g. two closures). Here high significances of both nodes result in a positive potential, expressing the contradiction. If one significance is low and the other high, a negative potential indicates that the contradiction is resolved. Unfortunately again thresholds are used extensively to determine when an overlap between groups (e.g. the overlap between closures) defines a supporting or contradicting relationship. Thresholding significances at a value of 0.8 reduces the number of grouping hypotheses roughly by half. A truly globally consistent interpretation of the image can however not be obtained.

It would be nice to see formation of local perceptual groups and their global consistency check be interleaved. In the above work in order to keep the number of nodes of the MRF at a manageable size, conservative thresholds throw away all less significant groups. So only groups which are already significant enter the global step. However it could well be that groups only ever become significant within the more global picture. E.g. a large gap between short lines would result in in a collinearity of very low significance (or even prohibit the formation of a collinearity in the first place), while the same gap might seem totally insignificant once theses lines are part of a larger closure.

Despite the shortcomings resulting from the abundance of tuning parameters, the applications of probabilistic global reasoning in the form of a Markov Random Field is a very interesting idea.

Desolneux et al. [DMM00] propose a computational method to decide whether a given Gestalt is meaningful. They define an observed geometric event (such as the alignment of pixels) as ϵ-*meaningful* if the expectation of the number of occurrences of this event is less than ϵ under a uniform random assumption. This is yet another realisation of the Helmholtz principle in an inverse formulation stating that a grouping is perceptually "meaningful" if its number of occurrences would be very small in a random situation. Geometric structures are then characterised as large deviations from randomness. A slightly problematic aspect of their approach is that it is necessary to assume a maximum number of events (e.g. lines, clusters). Their choices of maximum numbers are somewhat justified but typically seem rather ad hoc. Sometimes one choice seems to work better than another one, and

it is not clear how one is to choose this value correctly without trial and error.

In Desolneux et al. [DMM03] the same authors consider the collaboration of Gestalt principles. Each Gestalt principle detects *partial Gestalts* (e.g. alignments, parallelisms) which can subsequently be combined to form *global Gestalts* (e.g. a square). They show the collaboration of 3 Gestalt principles (boundary, region size, alignment). Other combinations (mean grey level or region orientation instead of region size) result in the same overall grouping. The problem of how to select proper combinations however remains unsolved and combinations are chosen ad hoc. Regarding the combination of Gestalt principles the authors also remark [DMM02]: *"We cannot hope any reliable explanation of any figure by summing up the results of one or several partial Gestalts. Only a global synthesis, treating all conflicts of partial Gestalts, can give the correct result."*

Although the work by Granlund and Moe [GM04] is not directly relevant for our work from a technical point of view, their emphasis on the importance of perception-action cycles is worth mentioning. Object representation must be linkable to the actions which resulted in particular percepts. Something is regarded an object which is separate from other objects or the background if there is an action that has a separate influence on the particular object. If the system can perform some action that makes certain parts of the visual field behave differently to other parts, then this part can be considered as belonging to a separate object. This is an important mechanism for learning new objects. Depending on what actions the system can perform on its environment, it will come up with a different partitioning into objects. E.g. a book shelf might be one object or the individual books might be objects themselves.

Furthermore an object is characterised by the context when it was learned. The context is an important part of the object definition. Recognition of an object in the wrong context is generally a difficult problem (even for humans). So on one hand a known context will simplify recognition of an object, on the other hand the identification of a particular object will evoke the context in which the object was learned, or the state of the system at the time. So context, actions on and appearance of objects should be treated together when learning and recognising objects.

Geisler et al. [GPSG01] show how the Gestalt principle of good continuation can be justified from image statistics of natural scenes. Statistical analysis of the co-occurrence of edge elements in 20 natural images show a high probability for edge elements which show good continuation. This serves to explain why this Gestalt principle evolved in order to detect contours in images. Contour detection performance by humans in psychophysical experiments is well predicted by a computational contour detection scheme based on these co-occurrence statistics.

Chapter 3

A Case of Object Recognition

Although we argued against (appearance based) object recognition methods in Chapter 1, a point can be made in favour of them. Sometimes descriptions of objects in features and parts, i.e. surfaces and relations between them, is so difficult that a pictorial representation is more adequate or at least more efficient. Moreover the prospect of having a system that does not require prior knowledge and is able to learn any object or even object category seems exciting. So we give it a try in this chapter and see how far we can get. At the end of this chapter we will reconsider in detail, using the presented examples, the obvious problems of such an approach.

We want a system which is able to recognise objects in real-world scenes, which have typically the following characteristics: (1) the objects to recognise often occupy a small fraction of the image, (2) the colour appearance of the objects varies with different illumination situations, (3) object appearance on pixel-level can change dramatically with varying rotation, translation and scale of the objects, (4) the objects can be partially occluded.

The presented approach (more details on which can be found in [RZE01]) is based on Support Vector Machine (SVM) learning in conjunction with a specific choice of training examples (pedagogical learning). We specifically avoid any preprocessing step such as feature extraction or dimensionality reduction and present a pure learning approach to 3D object recognition. The absence of any representations or other object knowledge implies that the system is completely unbiased with respect to the objects it can recognise.

3.1 Support Vector Machine Learning

Support Vector Machines are binary maximal margin classifiers. Figure 3.1 illustrates the basic principle. Suppose we want to separate a class A (denoted by '−' in the figure) and a class B (denoted by '+'). I.e. we want a classifier that classifies a point as either belonging to class A or class B. Given that the classes are linearly

separable, we can find infinitely many lines which separate them. An SVM finds the line which maximises the *margin*. The margin is the distance from the line to the closest data points (called *support vectors*). In the case of a two-dimensional parameter space three support vectors are needed to define the separating line.

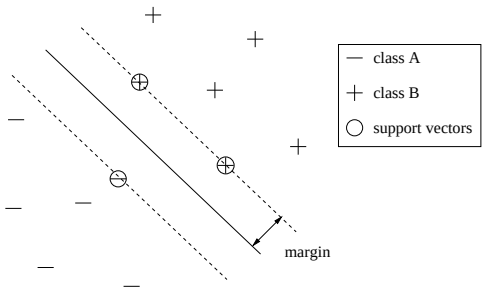


Figure 3.1: SVM principle.

The above example illustrated an SVM for the case of a two-dimensional parameter space, where the classifier is a line. Generally the parameter spaces are much higher-dimensional (several thousand in our case) and the classifiers are hyperplanes accordingly.

More formally, the capacity of a learning algorithm can be described by its Vapnik-Chervonenkis dimension (VC dimension) h, which is defined as the cardinality of the largest point set that the algorithm can separate, for all assignments of labels to those points. For our case of a linear classifier (i.e. a hyperplane) in a d-dimensional parameter space, it is thus given as the maximum number of points that are guaranteed to be linearly separable, e.g. 3 for a 2D space. Generally $h = d+1$ for a d-dimensional space. Vapnik [Vap98] shows that for a given number of learning examples the VC dimension of the classifier limits the generalisation of the classifier on future training examples. The risk of mis-classification on future test examples (which is a quantification of the generalisation performance) is bounded by the number of training examples and the VC dimension of the classifier. Concretely, given a number of training examples, the future risk can be decreased by reducing the VC dimension of the classifier. A classical approach to reduce the VC dimension is to reduce the dimensionality of the learning space (i.e. by compressing the data). However, the VC dimension of a classifier is not necessarily dependent on the dimensionality of the learning space.

The Support Vector Machine directly decreases the VC dimension (without requiring to compress the learning space) of a classifier by adding a *margin* to the hyperplane. A hyperplane with a margin has a VC dimension bounded by:

$$h \leq \min\left\{d, \left(\frac{R}{M}\right)^2\right\} + 1 \tag{3.1}$$

with d the dimensionality of the space, R the radius of the data and M the margin.

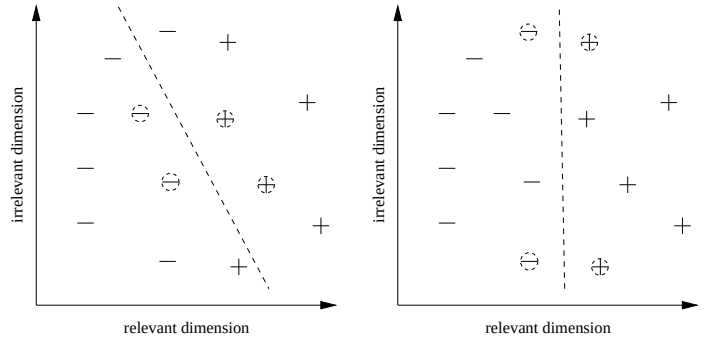
This implies that the larger the margin, the smaller the bound on the VC dimension. Hence the SVM searches for the hyperplane with the largest margin and thus for the classifier with the smallest VC dimension to achieve optimal generalisation performance.

Algorithmically, the standard implementation of the SVM, requires solving a quadratic optimisation problem [Vap98, BGV92]. Quadratic optimisation is computationally expensive. In contrast, DirectSVM is a fast and simple iterative algorithm that implements SVM learning with similar accuracy as the standard version [SSW+98]. We refer to [Roo00] for a detailed description of the learning algorithm.

3.2 Pedagogical learning

Pedagogical learning aims to minimise the number of training examples necessary in order to obtain a classifier with good generalisation capabilities. The basic idea of pedagogical learning is to provide well chosen training examples which contain extreme values in irrelevant dimensions and thus "tilt" the separating hyperplane such that good classification in relevant dimensions is achieved.

Figure 3.2 illustrates this idea. We can see that only the x-dimension is relevant in classifying A versus B. Both classes occupy large portions of the y-dimension, which is therefore irrelevant for classification. Suppose that we want to learn a classifier with only two training examples per class. Figure 3.2(a) shows a bad choice of training examples. Although the trained classifier successfully separates the training examples, it mis-classifies some members of A and B, i.e. it generalises poorly. The training examples of Figure 3.2(b) in contrast are a better choice. Now all members are classified correctly.



(a) Bad training examples (b) Good training examples

Figure 3.2: Pedagogical learning.

Concretely Roobaert [Roo99] proposes the Black/White (BW) training method which learns background invariance by choosing training images once with a black

and once with a white background. The irrelevant dimensions in this case are the background pixels.

The problem for learning classifiers then shifts to finding sets and/or methods for finding high-quality training examples. Hence we propose to search for *pedagogical learning strategies* i.e. finding heuristics to reduce the number of training examples necessary.

Training Methods

We select training examples to train the following invariances:

- Background invariance: As mentioned above, the BW background training method is applied.
- Translation and scale invariance: Although the later online recognition stage will work with image pyramids to cover wide ranges of scale, we also want a certain local translation and scale invariance. So we present training images at several scales slightly differing.
- 3D rotation invariance: Every object is presented in a number of different rotations.
- Illumination invariance: To capture lighting variations each object is presented in different illumination conditions.
- Sampling density invariance: The sharpness of the images varies with different scales and different illumination conditions. To accommodate for these variations, we provide smoothed images of the training examples as well.

Note that we consider the background as a separate "object". Given that n objects and a background have to be recognised, we construct for each pair of different objects a separate classifier. Hence $\frac{n(n+1)}{2}$ classifiers are trained. This approach has a higher accuracy experimentally [RVH99] than training only n classifiers (a classifier for each object against the background).

Note further that we typically have linearly separable classes, because the number of training examples (in the hundreds) is much smaller than the number of dimensions (in the thousands). So a linear separator (hyperplane) is adequate for our purposes.

3.3 Online recognition

We first build an image pyramid from the input image using scales factors from one pyramid level to the next between $1.15(= 2^{\frac{1}{5}})$ and $1.26(= 2^{\frac{1}{3}})$.

At each pyramid level, the image is scanned and sub-image windows are classified. In order to speed up the image scan, we used an adaptive scan step. First we scan the image coarsely using typically a scan step of 4 pixels. If the area is not classified as background with certainty, the area is explored with a step of 1 pixel. Since, the classifier exhibits some local translation and scale invariance, we do not miss an object during the coarse search. The finer search provides a more accurate classification result and localisation.

To classify a sub-image a two-step process is used, we first decide whether the sub-image is an object or background. Hence in the first step, only the object/background classifiers are used. Only if one of these classifiers identifies the image as non-background all the other object/object classifiers are invoked to classify more accurately the object. This way we avoid using all $\frac{n(n+1)}{2}$ classifiers every time. We use only n classifiers most of the time and only in the rare cases where we detect non-background, the other classifiers are invoked. Hence the runtime of the system scales nearly linearly in the number of objects.

Each classification returns a confidence value. The confidence is a measure of the distance of the classified vector to the hyperplane. If the distance is smaller than the margin a confidence < 1 is returned, because in this case the classifier can actually make no justified classification. In the other case, a confidence value ≥ 1 is returned. The final confidence of an object hypothesis is the minimum confidence of all classifiers used on this sub-image.

As the classifiers are not perfect, we obtain several object hypotheses with varying confidences. In the experiments, we simply pick the strongest one for each object over all scales. Hence, in the experiments reported, we assume that each object is contained in the image exactly once (using a winner-takes-all mechanism), if the confidence level is above 1.

3.4 Experiments

Training

We used 32x32 pixel raw RGB images (see Figure 3.3). So each image is represented as a vector in a $d = 32 \times 32 \times 3 = 3072$-dimensional space. For 3D object recognition we took about 40 views for each of five objects. The object was placed on a turntable and rotated around its vertical axis. For each view we used a black and a white background and two different illuminations (fluorescent light and additional incandescent light). This results in about 160 images per object. For face and gesture recognition we took about 25 images of a closed and an open hand, again in front of a black and a white background and with two illuminations. For face recognition 25 images of frontal views of faces of two persons were taken, again with two different backgrounds and two different illuminations. Additionally to these physical images, *virtual* images were created. Images shifted by 1 pixel were

created to train for improved local translation invariance and scaled by a factor of $1.09(= 2^{\frac{1}{8}})$ for improved local scale invariance of the classifier. To simulate more different lighting conditions, we added and respectively subtracted the pixel-values of the images with a fixed value (15% of white). We added smoothed versions of the physical images to the training set to compensate for sampling variations.

Figure 3.3: A few of the training images for objects *cleaner, fanta, mug, raisins* and *rice*.

For the "background-object", four arbitrary pictures (see Figure 3.4) of the room under normal fluorescent illumination were taken and scanned at different scales to provide approximately 2000 training examples.

Figure 3.4: The four images used to generate images for the "object" *background*

Results

The experiments were carried out on a Pentium III 450 MHz PCs. All experiments took place in normal fluorescent illumination.

3D Objects

Figure 3.6 shows a scene with 5 objects on a table from several views. For each view two lighting conditions were tested (with and without table lamp). The same procedure was repeated without a table cloth on the table. Figures 3.6(a) and

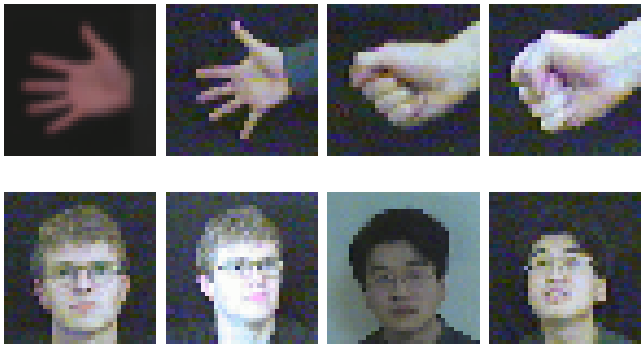
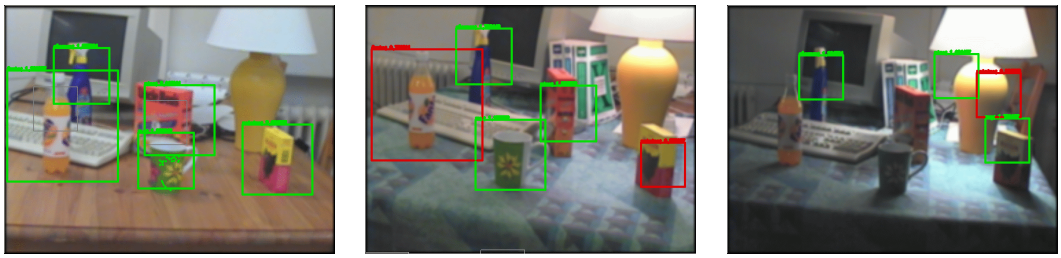


Figure 3.5: A few of the training images for objects *open_hand, closed_hand, michael* and *danny*.

3.6(b) show successful recognition. Note for example the partial occlusion of the *cleaner* and the *rice box*). Table 3.1 lists the number of correct classifications and mis-classification for each object. Correct classification means the selection of the correct location of the object in the scene. An incorrect classification means not necessarily that the classifier did not detect the object, but that another (false) location has a stronger response (false positive). Considering the large amount of classifications performed over a single scene (approx. 43000), one can regard the numbers in table 3.1 as quite satisfactory since the failure rate per classifier is in the order of 10^{-6}. Figure 3.6(c) shows a scene where the system breaks down: it was not trained for such a drastic change in illumination.



(a) Successful recognition (b) ... with different background (c) Breakdown due to insufficient lighting

Figure 3.6: 3D object recognition in different cluttered scenes and with different illuminations.

Faces and Gestures

In order to test the generality of the approach, we carried out a feasibility experiment on face and gesture recognition. The task is to recognise two persons (Figure 3.5) and two different gestures (open hand and closed hand (Figure 3.5)).

	cleaner	fanta	mug	raisins	rice
correct	23	23	23	22	23
incorrect	1	1	1	2	1

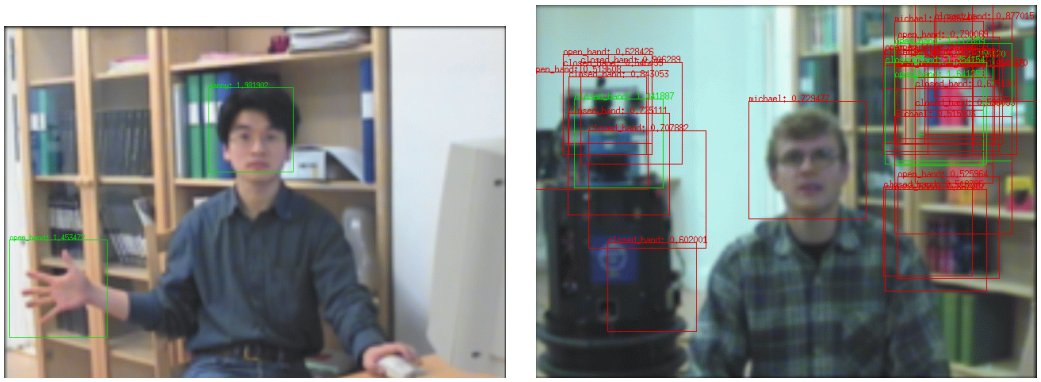
Table 3.1: Correct and incorrect classifications for the 3D object recognition task.

Figure 3.7(a) shows successful recognition of face and the open-hand gesture in front of a cluttered background. In Figure 3.7(b) the system produces a correct face hypothesis, but stronger false positives in the background cause the system to fail (in the winner-takes-all setting used).

The overall performance is lower than in the previous experiment. We assume that this is due to the lower number of training examples, while also the objects in this case might be considered more difficult to discriminate. To improve performance, the number of training examples could simply be increased ad hoc, or alternatively by:

- providing *false negatives* as extra object training examples.
- providing *false positives* as extra background training examples (a learning method introduced by [SP98] as bootstrapping).

After retraining, the system will never make the (exact) same errors again (as DirectSVM learns error-free on the training set).



(a) Successful face and gesture (*open-hand*) recognition

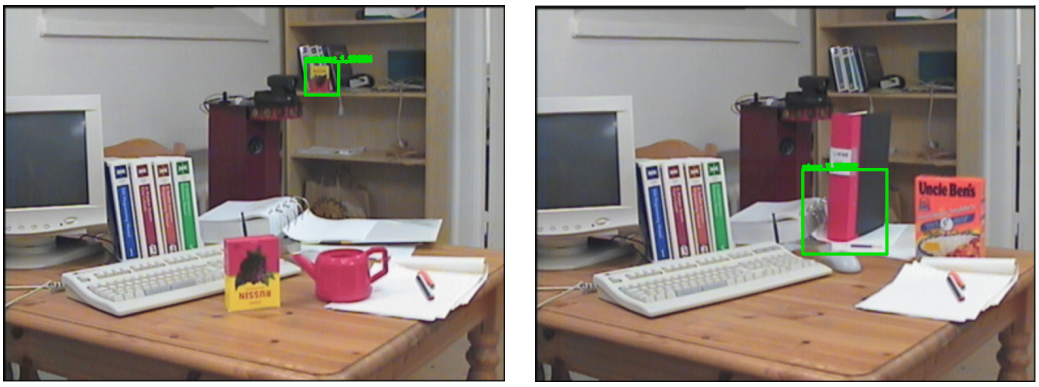
(b) False positives win.

Figure 3.7: Faces and gestures.

In follow-up experiments, we provided additional training examples and performed a few bootstrapping steps. This indeed improved performance, but more cycles than the additional 5 that we performed up to now would be needed.

Distractors

The system relies on an implicit combination of shape, texture and colour to recognise objects. The system does not simply rely on colour only to recognise the objects. This is amongst others illustrated by the gesture recognition system: hands have the same colour, but different gestures have different shapes. Another illustration is shown in the following result. In Figure 3.8(a) an upside-down raisins box is placed on the table as well as a red teapot, which serves as a distractor of similar colour. As the raisins have been trained only upright, the system finds the small upright box of raisins on the shelf (in the background).



(a) Correct box on the shelf is recognised. (b) Rice box and distractor

Figure 3.8: Distractors.

Analysis of the Support Vectors

The support vectors selected by the learning algorithm, are the only training examples that are used by the learning algorithm. All other training examples are not used and are hence non-relevant for the system. We obtained the following results:

- For the background-object classifier, about 200 to 400 background images are support vectors. If the number of background images is drastically increased e.g. by a factor 10 the amount of background support vectors does not increase substantially. Note also that different classifiers have several background support vectors in common. The number of support vectors for the object class is in the order of 15 to 40 images. We can conclude that only a small fraction of the training images are relevant for the learning system.

- For the object-object classifiers, the number of support vectors is lower and in the order of 5 to 15 support vectors per object. Since also the classifiers margin is wider than for the background-object classifiers, we can conclude

that it is easier to discriminate different objects from each other, than to discriminate an object from the background.

- The proportion of smoothed object images that are support vectors, is larger for the background-object classifiers than for the object-object classifiers. This is in agreement with intuition: in order to distinguish objects from the background, larger features (smoothed images) are discriminatory, while to discriminate between different objects, details (sharp images) have more importance.

Timing

Training a system varied between 5 minutes and a couple of hours (for the system with 10 times more background images).

Recognising two objects in an image of size 640×460 took the system about 3.5 seconds on a Pentium III 450. This includes building up the image pyramid for 7 scales. About 43000 classifications were performed, which means that one classification takes 0.08 ms.

3.5 Discussion

The experimental results of the proposed pedagogically-trained pure learning object recognition system seem to indicate that a wide range of 3D objects can successfully be detected and recognised in cluttered real-world scenes under varying illuminations. We found that the system is also robust to smaller partial occlusions of the objects.

Several severe problems however remain. Sometimes the system is distracted by other objects in the background, that are similar to the trained object in a certain view. E.g. in Figure 3.8(b) the system recognises a bright red binder as a rice box. These false positives can be overcome by retraining the classifiers and applying bootstrapping. Several iterations of this interactive learning typically train the system not to make the same error again. It is not clear however whether these iterations eventually converge, or whether learning to disambiguate one distractor would in fact unlearn another previously learned. There is no practical measure which would tell us when we are done.

Another issue is the problem of how to detect that the trained object is actually *not* in the scene. Finding a good threshold for the confidence of a hypothesis seems difficult as the confidence value is affected by e.g. lighting conditions (and thresholds are to be avoided anyway).

It is furthermore not clear how the approach scales to a larger number of objects. 30 objects could be achieved in [RVH99, RNE00] with a similar approach, but only for objects of the COIL-database [NNM96], which is a significantly simpler task. But even if we could extend the object database to 1000 objects: If I have

learned 1000 objects, what do they tell me about the 1001^{st}? Note that this plagues of course all object recognition or categorisation schemes.

Also learning for more invariances (such as in-plane rotation - we might actually want to recognise the upside-down box as well) is not guaranteed to scale well. Once the number of training images approaches the number of dimensions, linear separability is no longer guaranteed.

But there are of course more fundamental problems. First of all, using an object detector/recogniser it is not possible to make statements other than "object is there", "object not there". Although pose can be inferred if e.g. the object was learned using a turntable. Then appearance X can be mapped to orientation Y. But note that the starting angle of the orientation is arbitrary: appearance X might just as well map to orientation Z, depending on where one started rotating the table. The point is that nothing *inherent* in this pictorial representation (a point cloud in some parameter space) tells anything about orientations, no orientation is distinct. Compare that to a surface based representation of e.g. a box. Here there are distinct orientations that are inherent in the representation. One can ask: "What are the two largest, most parallel and closest surfaces (because these are the ones I would like to grasp with my two-finger gripper)?" Pictorial representations are a shortcut from pixels to objects, bypassing structure (see Figure 1.1(a)). As such they are of limited usefulness if one wants to interact with objects, which necessarily involves their structure.

Finally, looking at the example images with overlaid recognition results, we must say the system is blind, except for the few highlighted squares. The rest of the scene is non-existent as far as the object detector/recogniser is concerned. This is the most severe limitation. No matter how many objects we can eventually learn, the system will always be completely ignorant of the rest of the scene. This can hardly be called vision.

So after a few meters into this dead-end road, we return in the remaining chapters to a more promising path towards a more general vision system. And although we might not get far there either, we show that at least it is in the right direction.

Chapter 4

Significance

In this chapter we introduce the concept of significance which will be used in later chapters. Our aim is to find a general way of specifying when an image feature or a group thereof can be considered important, rather than accidental.

Numerous quality measures for all sorts of features exist, based on fit errors of some model or on geometric considerations. Typically they are carefully designed for a specific type of feature. We would however like a measure that is general in that it is comparable across different types of features or groupings. One possible common frame of reference is probability theory.

4.1 Probabilities, Non-Accidentalness and the Helmholtz Principle

Probabilities have been used before as measures for significance (saliency), e.g. the significance measures for collinearities, parallelism etc. of [Low87], the notion of maximal meaningful events by [DMM00, DMM03] and the co-occurrence statistics of edge elements in [GPSG01].

[DMM00] and [GPSG01] however seem to indicate a contradiction. While the former advocate the desireability of low probabilities (low number of expectation of events respectively) the latter report on findings that common Gestalt principles are motivated by high co-occurrence probabilities. So are we looking for low or high probabilities?

Let us restate what we actually want the vision system to do: We want the most plausible (i.e. most likely) explanation of the image. Then why are [DMM00] looking for the least probable groupings? First of all, what do we mean by most likely explanation? To be precise, we want a high probability that grouped Gestalts make sense, namely describe image structure which corresponds to structure in the world, i.e. projections of world structures, i.e. have a cause. So we want a high probability that groupings have a cause and are therefore not accidental (random). Now the less likely the null hypothesis H_0 that a grouping is purely accidental, the

more likely the alternative H_1 that it has a cause.

The less likely randomness, the more likely an underlying cause.

Once we accepted H_1 and thus assume an underlying cause, we want to select the most likely explanation for it. Which relates to high probability values in the co-occurrence statistics of [GPSG01].

The above conclusion was first drawn by Hermann von Helmholtz as early as 1867 [Hel67]. The Helmholtz likelihood principle states that

> "Die allgemeine Regel, durch welche sich die Gesichtsvorstellungen bestimmen, die wir bilden, wenn (...) ein Eindruck auf das Auge gemacht worden ist, ist die, dass wir stets solche Objecte als im Gesichtsfelde vorhanden uns vorstellen, wie sie vorhanden sein muessten, um unter den gewoehnlichen normalen Bedingungen des Gebrauchs unserer Augen denselben Eindruck auf den Nervenapparat hervorzubringen."

Roughly stated:

> "We perceive that which would in our normal life most likely have produced the effective stimulation we have received."

That is our visual system chooses the most likely explanation assuming typical normal viewing conditions. A process which Helmholtz calls *unbewusstes Schliessen - unconscious reasoning*, by which our visual system infers from observed effects the underlying cause of effects.

Coming from the opposite side, a grouping is perceptually "meaningful", or makes sense, if the probability of its occurrence would be very small as a result of accidental arrangement (i.e. non-typical viewing conditions). Pre-defined geometric structures, Gestalts, are then characterised as large deviations from randomness.

In accordance with the terminology of probability theory let us call an image feature or grouping of features a *visual event*. (Note that of course an image feature such as an edge is itself again a grouping, namely of pixels.)

Our general H_0 hypothesis is that there is no structure and visual events are governed by some random process. Whenever an event is highly unlikely under that assumption, we regard this event as non-random (non-accidental) and accept the alternative hypothesis H_1. We will now explain the types of random processes we assume.

4.2 Poisson Process

A typical situation which often arises in grouping is to decide whether a cluster of points in the image or in some parameter space is significant or not. For the time

being we are not interested in how to form (good) clusters, but just to measure the non-accidentalness of a given cluster. The null-hypothesis is a random distribution of points in a continuous space, where points are independent and evenly distributed. This situation can be modelled as a Poisson process.

Definition 1 *A Poisson process with parameter λ is a chance experiment with the following properties:*

1. *Possible outcomes of a trial are* 0 *and* 1.
2. *Trials are independent.*
3. *The probability of obtaining outcome* 1 *within an infinitesimally small interval $[x, x + dx]$ is λdx.*

Be N_A the random variable denoting the number of successful trials in interval A. Then N_A is $P_{\lambda|A|}$-distributed, where $|A|$ is the size of interval A and P_α is the Poisson distribution with the following probability density function (PDF) and cumulative distribution function (CDF):

$$p_\alpha(x) = Pr(X = x) = e^{-\alpha}\frac{\alpha^x}{x!} \tag{4.1}$$

$$P_\alpha(x) = Pr(X \leq x) = \sum_{i=0}^{x} e^{-\alpha}\frac{\alpha^i}{i!} \tag{4.2}$$

For our purposes we will need Poisson processes over higher dimensional domains, which are defined analogously with the interval $[x, x + dx]$ replaced by $[x_1, x_1 + dx_1] \times \ldots \times [x_n, x_n + dx_n]$

The spaces we consider are e.g. the image plane, the parameter space of line angles or the 3-dimensional colour space. We use the Poisson distribution when assessing the likelihood of a certain number of points falling into a given interval or of distances between points. Figure 4.1 illustrates the idea.

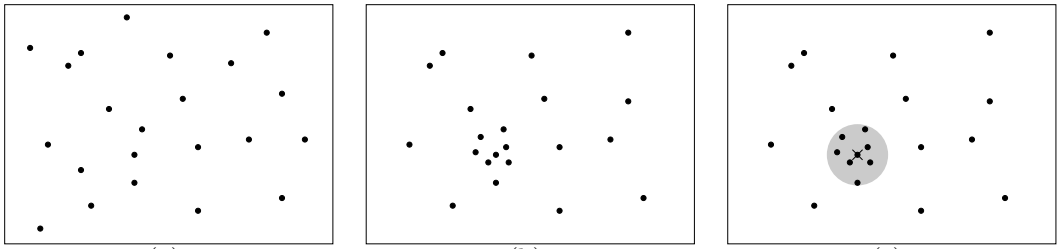


Figure 4.1: A 2D Poisson process (a) and a rather unlikely configuration of points (b).

Let $d = \frac{N}{wh}$ be the point density in a $w \times h$ image containing N points. Figure 4.1(a) shows what a typical distribution of points following a Poisson process might look like. Figure 4.1(b) shows a distribution of points which would be rather unlikely for points following a Poisson process. The probability of finding at least a number of n points in the shaded area A in Figure 4.1(c) is given by 1 minus the probability of finding at most $n-1$ points and thus

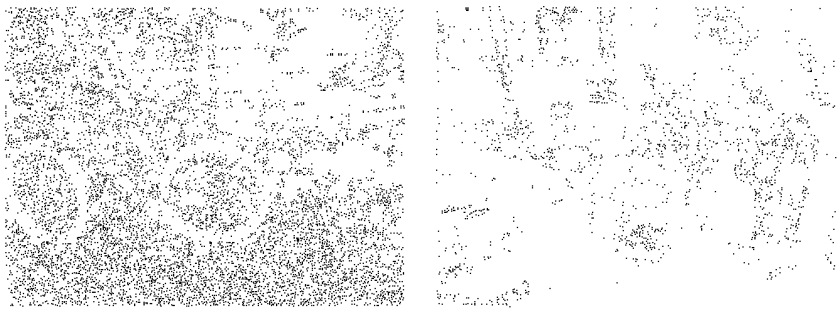
$$Pr(\text{at least n points in area A}) = 1 - P_{d\,|A|}(n-1) \tag{4.3}$$

Especially

$$Pr(\text{at least one point in area A}) = 1 - P_{d\,|A|}(0) \tag{4.4}$$

$$= 1 - e^{-p|A|} \tag{4.5}$$

Note that we do not simply use $Pr(\text{exactly one point in area A}) = p_{d\,|A|}(1)$. Suppose A were really big, maybe actually covering most of the domain. Now of course the expected number of points within A is also large. The probability of actually only finding a single point is very small: $P_{d\,|A|}(1) \ll 1$, making this a highly significant event. But that is not what we are interested in. We are typically not interested in the emptiness of space around a point (though that might be of interest in some other settings), but we are interested whether the closeness of two points is significant. I.e. what is the probability of finding *any* point in A. And in that case $1 - P_{d\,|A|}(0) \approx 1$, therefore not significant at all, as we would expect.



(a) bike (b) office

Figure 4.2: Distribution of line endpoints in actual images (see figure 5.3 for the original images)

As an example of our use of Poisson processes Figure 4.2 shows the distribution of line endpoints in two actual images. Line endpoints will be used for measuring the significance of e.g. collinearities. We can see that the Poisson assumption is roughly true in the rather unstructured lower part of the bike image 4.2(a) containing leaves and grass. Significant structure (such as the bike wheels) pops out as local violations of this assumption. We also see that the office scene 4.2(b) is more structured than the outdoor bike scene.

So the Poisson assumption holds on a rather global scale, and line endpoints far apart are indeed independent, while local deviations hint at underlying structure.

Remark

The significance measure of Lowe [Low87] uses a scale-invariant line endpoint density D. D is set to 1, because significance is only used for ranking, so absolute values are not important. Rather than using an arbitrary scale-invariant endpoint density we just measure the actual endpoint density d. If the image resolution is reduced, say by a factor of 2, endpoint density d increases by a factor 4. At the same time areas A calculated from endpoint distances are reduced by a factor 4. So the product $d\,|A|$, which is used in the calculation of non-accidentalness, is again scale-invariant.

Furthermore our measure of significance is meant to be more general, being applicable to any distance between points in an n-dimensional parameter space. Lowes measures are specifically designed for grouping lines of a certain length.

4.3 Markov Chain

The previous section mentioned line endpoints as one example for visual events that follow a Poisson process. Quite clearly edgels (edge elements) must follow a different process. The definition of a Poisson process states that events are independent, i.e. in our case two points are independent. But by the very definition of an edgel, it can never occur isolated. Edgels always occur as strings of edgels with varying length, where typically long edges are less likely.

We can view the situation like this. Starting with a string of length one we keep extending our string of edgels by one as long as we can find a connected neighbour. Let us call the situation *"string is expandable (in any direction)"* state s_1 and the situation *"string is terminated"* state s_2. And let $S^{(n)}$ denote the random variable of being in either s_1 or s_2 after n steps. We know that strings have a finite length. So given a string of n edgels, there is a certain probability that we stay in s_1 (the string can be extended to $n+1$) edgels and a certain probability that we move to s_2 (string terminates) and stay there. If we assume that the probability of continuation depends only on state n we have the Markov property that the state at step $i+1$ only depends on the state at step i..

$$Pr(S^{(n+1)} = s \mid S^{(n)} = s^{(n)}, \ldots, S^{(0)} = s^{(0)}) \;=\; Pr(S^{(n+1)} = s \mid S^{(n)} = s^{(n)}) \quad (4.6)$$

Note that this assumption is reasonable as edgels are computed locally (typically using a 3×3 convolution mask) and an edgel does not directly affect another edgel further away. So we can model edges as Markov chains.

Definition 2 *A Markov chain is a stochastic process defined over a discrete state space $S = \{s_1, s_2, \ldots, s_k\}$ where the sequence of random variables $S^{(0)}, S^{(1)}, \ldots$ has*

the Markov property. It is specified by an initial state $s^{(0)}$ *and a transition matrix* $\mathbf{P}$*, where entry* p_{ij} *gives the probability of moving from state* s_i *to state* s_j*, and* $S^{(n+1)} = \mathbf{P}S^{(n)}$*.*

In our case we can never leave state s_2 (string is terminated) once we entered it. Such a state is called *absorbing*.

Definition 3 *A state* s_i *of a Markov chain is called* absorbing *if it is impossible to leave it (i.e.,* $p_{ii} = 1$*). A Markov chain is absorbing if it has at least one absorbing state, and if from every state it is possible to go to an absorbing state. A state which is not absorbing is called* transient*.*

The transition matrix of an absorbing Markov chain with t transient and r absorbing states is of the form

$$\mathbf{P} = \begin{pmatrix} \mathbf{Q} & \mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \tag{4.7}$$

Here $\mathbf{I}$ is an $r \times r$ identity matrix, $\mathbf{0}$ is an $r \times t$ zero matrix, $\mathbf{R}$ is a nonzero $t \times r$ matrix, and $\mathbf{Q}$ is a $t \times t$ matrix. The expected number of times the system is in a transient state s_j if it is started from a transient state s_i is given by n_{ij}, where $\mathbf{N}$ is called the fundamental matrix and is given as

$$\mathbf{N} = (\mathbf{I} - \mathbf{Q})^{-1} \tag{4.8}$$

In our simple case we have one transient state s_1 and one absorbing state s_2. The transition matrix is given as

$$\mathbf{P} = \begin{pmatrix} p_{ee} & 1 - p_{ee} \\ 0 & 1 \end{pmatrix} \tag{4.9}$$

where p_{ee} is the (unknown) probability of continuing an edge, i.e. the conditional probability of an edgel, given an edgel. The expected number of times the system stays in state s_1 starting from s_1, i.e. the expected length of an edge is given by equation 4.8 as

$$\bar{l} = \frac{1}{1 - p_{ee}} \tag{4.10}$$

We can estimate the mean edge length $\bar{l}$ in an actual image and thus calculate the parameter p_{ee} as

$$p_{ee} = \frac{\bar{l} - 1}{\bar{l}} \tag{4.11}$$

For example an average edge length of 20 edgels, which is a typical value for many of our indoor scenes, p_{ee} turns out to be 0.95. The probability for staying in state s_1 for l steps, i.e. for an edge having length l is then given by

$$Pr(S^{(n+1)} = s_1 \mid S^{(0)} = s_1) = p_{ee}^{l} \tag{4.12}$$

We will use the above conditional edgel probability when assessing the significance of a model being supported by a given number of edgels.

4.4 Bernoulli Process

While a Markov chain is useful when regarding the likelihood of getting further support for an instantiated model, the following process is suited for describing accidental support for a possibly hallucinated model.

Suppose we assume the existence of say an ellipse being fitted to a minimum of 5 points (see Chapter 5 for more on ellipses). This hypothesised ellipse will probably get some support from various edges in the image. Some edges will actually belong to the ellipse (especially if we chose the original points to fit wisely), other edges will just be intersected by the ellipse, thereby providing accidental support. The bigger the ellipse, the more support it will pick up accidentally. Given the ellipse has a circumference of l, how significant is a support of $k \leq l$ edgels? That is how likely is it, that the ellipse would accidentally pick up k supporting out of l possible edgels?

Note the difference to the situation in the previous section, where the events we were interested in were *"there is/is not an edgel so that we can extend the edge in any direction"*. While here we have a hypothesised model (e.g. the above ellipse) which selects a specific set of pixels from the image and we ask how many of these pixels are supporting the model (or are correctly predicted by the model).

Although as we have just seen in the last section edgels are not independent, an arbitrary line drawn across an image containing edges at various angles will most likely intersect most of the edges, rather than being aligned. Therefore when looking at the individual pixels along the line and observing whether they are edgels or not, the event *"pixel i is an edgel"* can be considered independent of *"pixel $i+1$ is an edgel"*. This is of course not exactly true, because some edges will be intersected at acute angles. So the chances of picking up another edgel if the previous pixel was an edgel are higher than picking up an isolated edgel. To be precise we would have to model these events as absorbing Markov chains with very short expected lengths. For the sake of simplicity however allow us this slight inaccuracy and let us assume that the events *"pixel i is an edgel"* and *"pixel $i+1$ is an edgel"* are actually *independent*. In that case the above situation can be described using a Bernoulli process.

Definition 4 *A Bernoulli process is a discrete stochastic process consisting of a sequence of independent random variables $X_1, X_2, X_3, \ldots$, such that*

1. *For each i, the value of X_i is either* 0 *or* 1*;*

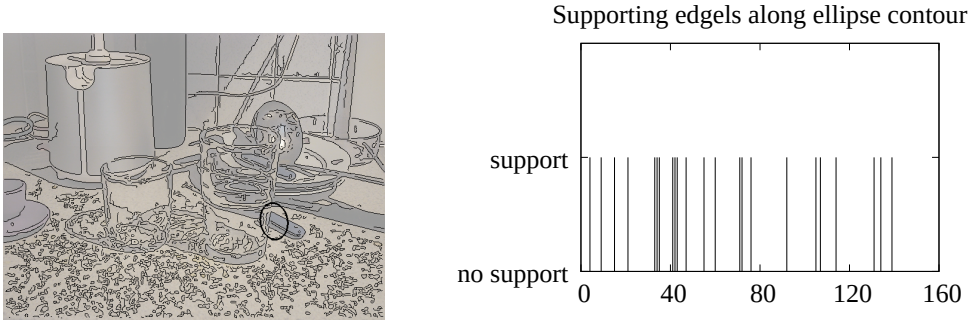
2. The probability of $X_i = 1$ is p for all i.

The random variable describing the number of successes (outcome 1) in the first l trials has a binomial distribution, given by the following probability density function (PDF) and cumulative distribution function (CDF):

$$b_{l,p}(k) = Pr(X = k) = \binom{l}{k} p^k (1-p)^{l-k} \tag{4.13}$$

$$B_{l,p}(k) = Pr(X \leq k) = \sum_{i=0}^{k} \binom{l}{i} p^i (1-p)^{l-i} \tag{4.14}$$

In our case p is the probability of a pixel being an edgel, p_e, and can be estimated as $p_e = \frac{E}{N}$, where E is the number of edgels and N the total number of pixels in the image.



(a) (b)

Figure 4.3: A hallucinated ellipse (a) and its accidental support (b).

Figure 4.3(a) shows a kitchen scene with extracted edges and an arbitrary hallucinated ellipse. Figure 4.3(b) shows the "rolled out" contour of the ellipse with positions of accidental supporting edgels. We see that the occurrences of accidental supporting edgels are in fact roughly independent.

The probability for picking up *at least* k supporting edgels is then given as

$$Pr(X \geq k) = \sum_{i=k}^{l} \binom{l}{i} p^i (1-p)^{l-i} = 1 - B_{l,p}(k-1) \tag{4.15}$$

i.e. the tail of the Binomial distribution, which can be expressed in terms of the the regularised incomplete beta function $I_x(a, b)$. Generally

$$B_{l,p}(k) = I_{1-p}(n-k, k+1) \tag{4.16}$$

$$1 - B_{l,p}(k) = I_p(k+1, l-k) \tag{4.17}$$

where we use the property

$$I_x(a,b) = 1 - I_{1-x}(b,a) \tag{4.18}$$

So

$$Pr(X \geq k) = I_p(k, l-k+1) \tag{4.19}$$

4.5 Log-Likelihood

The above sections introduced probabilities of certain visual events. We want these events to be non-accidental, i.e. have a small probability of happening accidentally. As these probabilities can sometimes become quite small, it is more convenient to use the negative logarithm of these probabilities. We call the probability of an event happening by accident its *accidentalness* a and define *significance* as follows

Definition 5 *The significance σ of a visual event with probability a is given as*

$$\sigma = -\log a.$$

A small accidentalness corresponds to a high significance. This definition also fits the common use of the word significance. The less likely an event under the uniform assumption, the more surprising and thus significant is its occurrence.

Note the difference to significance levels in statistical test theory, where roughly stated, one assumes a distribution of a certain type with given parameters and after drawing samples accepts or rejects this assumption. A typical example from quality control would be a factory producing screws with a length of e.g. 20 mm. Due to inaccuracies in the production process the actual lengths of screws will vary. Now a (common) assumption might be that the random variable describing screw length follows a normal distribution with $\mu = 20$ and, say, $\sigma = 0.1$. During quality inspection a number of samples is drawn with a mean of e.g. 19.8 mm. A statistical test will now decide whether this difference is a result of chance (the null hypothesis H_0) or whether there actually is a difference (the alternative hypothesis H_1). In the above example, if H_0 is true, we accept the assumption that screw length is in fact normally distributed with the $\mu = 20$ and $\sigma = 0.1$. In this context the significance level of the statistical test is the probability of making a Type I error (falsely rejecting the null-hypothesis, i.e. falsely accepting the alternative, or false positive).

In our case we are perfectly aware that the assumption of a uniform distribution is wrong. Our aim is not to correctly estimate the parameters of a true distribution. We can not hope to capture the actual statistics of the image, which is the result of its structure, in a simple probability distribution. Also we do not draw many samples to prove/disprove the distribution hypothesis. We know the assumed distribution is wrong and want a measure of *how* wrong it is for a given sample.

By using the above random processes we essentially assume a uniform distribution of e.g. line endpoints, line angles, colours in colour space and so on. One might argue that we should model the actual distribution more closely, for example by building histograms. If we could in fact precisely model the actual distribution of data in the image (which would be very hard indeed!) then this distribution should enable us on average to predict the actual image. This would however render the principle of non-accidentalness useless, because data would behave as expected (as modelled) and there would be no "surprises", i.e. no unlikely events. Again, it is the null hypothesis' job to be proved wrong, therefore there is no point in trying to model it correctly.

Chapter 5

Detecting Ellipses

Rotationally symmetric objects are common in many man made environments like home and office scenarios. Such objects with a circular cross-section give rise to elliptical features in the image. Elliptical features are highly non-accidental and their detection in the image is therefore a strong, distinctive cue of the presence of a circular or elliptical structure in the 3D world. In contrast straight lines are weaker cues as they are generally more abundant in artificial environments. Moreover, under the assumption that a detected ellipse is an axis-normal cross-section of an object of revolution, 3D pose of that object can be partly inferred. If we have a model of said object, specifying size and position of the 3D circle relative to an object coordinate-ordinate system, we can determine 6D object pose up to a rotational degree of freedom.

We will first review related work in the next section. We then give a short overview of our proposed approach in Section 5.2 and briefly remark on our choice for an edge-based method in Section 5.3. Section 5.4 explains the setup used for experiments throughout this chapter. We then present our approach for ellipse detection which consists of three steps: Fitting circular arcs to edges (Section 5.5), convex grouping of arcs (Section 5.6) and finally fitting ellipses to convex groups of arcs (Section 5.7). Section 5.8 concludes with a discussion.

5.1 State of the Art

Ellipse detection has received considerable interest in the machine vision literature but remains an open problem. The problem of robustly fitting an ellipse into a set of points can be considered solved [Boo79, NR79, Sam82, Por90, Tau91, EAB92, Ros93, GGS94, Kan94, FF95, Ros96, Zha97, FPF99]. Of course the problem remains how to get to these points from a general set of image edges.

The ellipse detection problem can be posed very generally as follows:
Given a set of 2D points, find subsets of points that support ellipses and estimate ellipse parameters from these subsets using some fitness function.

In this formulation the input information available in the image is treated as a set of points. This situation is suited for parameter space clustering approaches like the Hough transform (HT) [Hou62], although the high dimensionality of the parameter space in the case of ellipses poses a significant problem. Some methods try to reduce the dimension of the parameter space by first estimating ellipse centres or angles, leaving a lower dimensional parameter space.

Yuen et al. [YIK89] propose a two-step Hough transform, each solving a sub-problem with a reduced set of parameters. The first HT identifies ellipse centres using a geometric constraint (see also Section 5.6.4). The second HT step determines the remaining three parameters. If ellipse parameters are to be determined with high accuracy, i.e. a fine quantisation of the parameter space, the resulting high number of bins for a 3-dimensional parameter space still poses a problem. Therefore an adaptive Hough transform (AHT) of Illingworth and Kittler [IK87] is used, where the parameter space is iteratively subdivided into finer bins until the desired accuracy is met. The above method runs into problems if more then one ellipse is present in the image as peaks in the parameters space get smeared, especially if ellipses are overlapping or close. Some thresholds regarding expected size and separation of ellipses are introduced to overcome this problem. Moreover the $O(n^2)$ complexity of finding pairs out of n points for the centre finding stage limits the method to rather simple and clean images.

McLaughlin [McL98] proposes a combination of the Randomised Hough Transform (RHT) [XOK90, XO93] with the above ellipse centre constraint. Rather than accumulating parameters for all possible ellipses going through each single point, RHT chooses 5 random points (i.e. just enough to calculate ellipse parameters), calculates ellipse parameters from these and places a single vote in parameter space for this ellipse. As a result the five-dimensional parameter space becomes only sparsely populated and efficient list structures can be used instead of the 5-D histogram. The approach by [McL98] however, for reasons not entirely clear, chooses three random points and estimates tangents using points in a small neighbourhood. The above constraint by Yuen et al. is used to define the ellipse centre from three points and three tangents, then the remaining three parameters are found by solving a system of three linear equations. The author could just as well have picked five points and solved a system of five linear equations. The results are not very impressive. The number of correct ellipses drops below 50% for more than 3 ellipses in the image (with no noise and no other image structure present). Also adding more that 0.5% speckle noise reduced the number of correctly identified ellipses to below 50%. As with all HT based methods, the more data points there are (resulting from more structure or noise) the less reliable solutions become. Again test images are very clean and simple.

Kanatani and Ohta [KO04] present another approach based on the Hough transform. Their aim is to detect ellipses at high precision under the assumptions that only a small number of circular objects are present in the image and circular

objects are sufficiently large in the image (they specifically do not claim to have a system of universal merit). They first detect an osculating circle to an elliptic arc using a two-step Hough transform. The first step votes for the circle centre, the second step for circle radius. Accordingly parameter spaces are kept small with two and one dimensions respectively. Several ad-hoc parameters and weighting factors are chosen to enhance the saliency of peaks in Hough space. They then iteratively deform the circle into an ellipse. A δ-region around the current ellipse is formed and the edge segment with the largest number of consecutive edgels inside that region is used for the next ellipse fit. Outliers during ellipse growth are removed using LMedS. To search for other supporting edges, they randomly select edges within a region around the ellipse and add them if it improves the fit (using again some ad-hoc parameters). The above procedure finds one ellipse at a time. Removing the corresponding edges from the image enables search for the next ellipse and so on. As long as their restrictive assumptions hold (to which the various parameters seem to be tuned) they can show impressive results even for considerably occluded ellipses.

So much for the Hough transform.

Stricker and Leonardis [SL95] propose an algorithm which is explicitly general and independent of the domain (optical flow, intensity, 2D, 3D) and type of input data (sparse, dense) as well as type of model (straight line, ellipse, plane, sphere, cylinder) and fitting algorithm (least squares, robust) to fit parametric models. The authors employ a hypothesise and verify method. The first step, exploration, is based on a random sampling of points lying on boundary elements. In order to cope with the huge size of the search space exploration is biased by recency (recently used data are less likely to be selected). The problem of a stopping condition though remains: When do we have enough random samples? The risk of missing a hypothesis vs. accepting too many hypotheses (and thus long computation time) becomes apparent when the images contain significant amount of background clutter.

Treating boundary edgels as single points however neglects the fact that edgels are actually connected. This leads to an unnecessary increase of search space size as vital relation information is thrown away. If we are interested in processing perspective images of natural scenes, ellipses will be detected in the output produced by an edge detector and thus connectivity information is at hand. Edge extraction however can never be expected to be perfect, so we rarely get complete ellipses, but rather short arcs with gaps in between. Moreover part of an ellipse might be occluded and there will be many distracting edges from background clutter. So given a set of extracted image edges the problem is now to identify those groups of disconnected edges that together form an ellipse. Although using edge segments instead of single edge points reduced the number of groupable elements, the problem still is the huge number of possible combinations of edges.

Rosin and West [RW95] segment lists of connected edgels into straight lines and

elliptical arcs. Their algorithm also joins adjacent segments to form possibly larger segments. They do not however group segments possibly belonging to the same ellipse if the segments are separated by a gap, as happens with partial occlusions or simply due to locally poor contrast.

Qiang and Haralick [QH99] pick up at just that point and deal with merging pairs of disconnected elliptical arcs. They introduce scale-invariant statistical criteria to test whether two arc segments belong to the same ellipse. Furthermore these criteria adapt to the noise characteristics of the image and need not be hand-tuned. Pairs of arcs are first selected based on geometric criteria and subsequently validated or rejected based on these statistical criteria. There are two clustering steps. In-list-clustering connects short arcs lying on the same list of connected edgels into longer arcs. The second between-list-clustering connects arcs from different lists. Candidates for merging must first satisfy a proximity criterion, which is trivially fulfilled for arcs on same list. For between-list-clustering an arbitrary (and certainly not scale-invariant) proximity threshold needs to be defined. Furthermore the arcs must be geometrically compatible. The criterion for deciding whether two arc candidates should be merged is based on the statistics of the residual errors when fitting an ellipse through both arcs. If the residual error follows a normal distribution, the merge is accepted. A remaining problem is that the size of the variance of this distribution is not considered in the test. I.e. merging can occur even if the residuals are large, as long as they are normally distributed. Furthermore the authors did not mention *how* the candidates for between-list-clustering would be selected. Testing all pairs of arcs in the image leads to combinatorial explosion.

Kawaguchi and Nagata [KN98] use genetic algorithms to deal with this huge search space. However their selection of arcs from generic edges is rather ad hoc. Furthermore the search makes no explicit use of the underlying structure of the problem, namely that arcs belonging to the same ellipse form a convex group and neighbouring arcs are usually close to each other.

5.2 Overview

In the approach proposed we aim at the very common situation, where connectivity information is available but is not sufficient to establish reliable ellipse hypotheses. Typically edges are broken due to poor contrast or occlusions leading to elliptic edge segments rather than whole ellipses.

Figure 5.1 shows an overview of the approach. The main steps are:

Detecting edges Our approach is based on edges, so the first step is edge detection. We rely on an off-the-shelf edge detector.

Detecting circular arcs (Section 5.5) We take the output of edge detection and fit circular arcs to edges. Fitting circles instead of ellipses is more stable

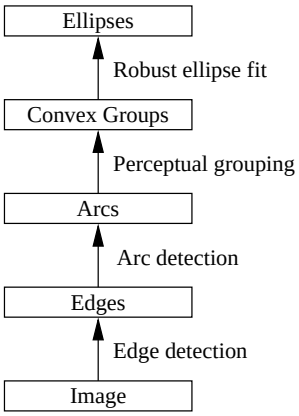


Figure 5.1: Overview of ellipse detection.

and faster and locally approximates ellipses well.

Finding convex groups of arcs (Section 5.6) In the second stage we form ellipse hypotheses from groups of arcs. Here we face 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.

Fitting ellipses (Section 5.7) In the third stage robust standard techniques are then used to fit ellipses to convex groups of arcs and ellipses ranked according to different quality criteria.

5.3 A Remark on Edges

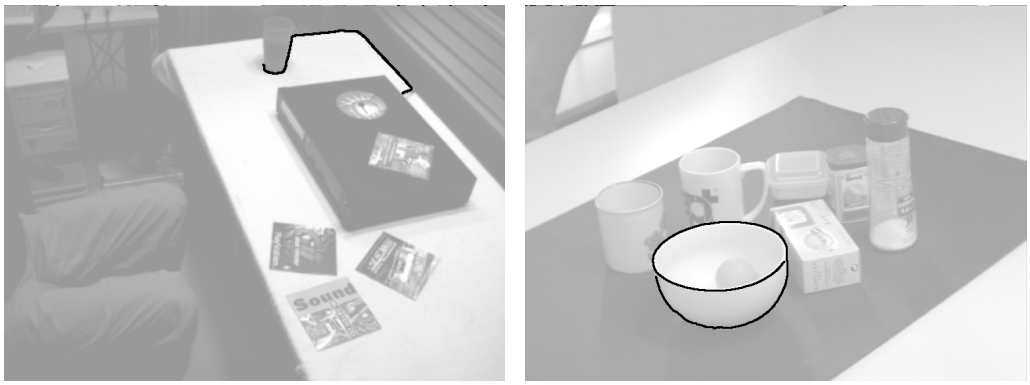
Edges (= discontinuities) carry more information than the homogeneous regions in between. Interest points carry even more information. The Kadir-Brady interest point operator [KB01, KZB04] for example looks for salient regions which are defined in an information-theoretic way as regions of high entropy over various scales. The SIFT [Low04] interest point extractor and descriptor stresses invariance to scale and orientation to extract distinctive image features. In fact interest point operators extract information in such an efficient manner that the mutual information between interest points is minimised (i.e. redundancy is minimised and thus information content is maximised). That is exactly their aim, no two interest points should be the same. If two similar interest points are detected in two images they should refer to the same physical object feature. I.e. any similarity between interest points (in the same image) which might hint that they in fact belong to the same surface is deliberately eliminated. Thus interest points are inherently not connected. They can not be (perceptually) grouped. (Except of course according to consistent motion when they are tracked. But here grouping is not based on

similarities inherent in the interest points but on their trajectories.)

Edges form an intermediate step. They represent reduced data, i.e. extracted information, but enough mutual information is retained to enable grouping based on similarities (e.g. orientation, colour). This allows perceptual grouping of edges which are likely to belong to the same surface and thus the same physical object. Only this mutual information, and thus redundancy, allows the vision system to formulate statements about objects.

Likewise points of high curvature on edges carry more information than the low curvature parts in between. That is why high curvature points also play an important role in shape perception by humans. This is our justification to break up edges at high curvature points and describe them using these points and parametric descriptions of the smooth curves in between. For a start we use straight lines (in Chapter 6) and circular arcs. Elliptic arcs or even higher order curves such as bsplines might also be considered. We restrict ourselves however to curves with low numbers of parameters as these are generally easier to to fit reliably as we will demonstrate shortly.

Why do we need parametric models and cannot just use the edges themselves, as connected strings of edgels? Look at Figures 5.2 (a) and (b). They both show a single edge as detected by the edge segmentation process.



(a) (b)

Figure 5.2: Meaningless long edges.

We want to regard edges as something separating one surface from another. Clearly for the highlighted edge in Figure 5.2 (a) this is not the case. This edge is completely meaningless. Breaking up such an edge at high curvature points will in most cases break the edge down into meaningful parts. Figure 5.2 (b) shows another problematic edge. How could we define left/right, inside/outside, start/end for such an edge? In order to make statements such as “have same orientation” or “have same colour inside” which are essential for grouping, we have to be able to define geometric relations and thus need parametric models.

5.4 Experimental setup

We used 280 images of various scenes for the experiments in this chapter. The scenes fall into five categories: office, bikes, cars, kitchen and traffic signs (four examples of which are shown in Figure 5.3). The images with overlaid Ground Truth (GT) ellipses can be seen in appendix A. All images are 640×480 pixels, except where indicated otherwise. We chose the scenes such that they contain objects which are typically identifiable by their rotational symmetry. As already indicated by the example images so far, these objects are cups and various kitchen items as well as wheels of bikes and cars. Ground Truth contains only these objects and not the other hundreds, thousands or millions of ellipses one might also be inclined to see, and which would be very tedious to include into Ground Truth.

Note that some GT ellipses are near to impossible to detect (glass in office scene, glass behind glass in kitchen scene, wheels in many-bikes scene), which explains the high FN figures in those scenes.

5.5 Detecting Circular Arcs

All subsequent processing is based on edges. So the first step is to detect edges using a Canny [Can86] edge detector. The implementation we use automatically sets internal thresholds for hysteresis thresholding and typically extracts reasonable edges. Often however edges are broken due to poor contrast or occlusion. Moreover the image is occasionally over-segmented resulting in a vast number of tiny edges. Subsequent stages of processing must be able to deal with these conditions efficiently. I.e. we do not assume perfect segmentation and make no attempt to set favourable edge detection parameters for each image. Figure 5.3 shows some prototypical images from our evaluation database and Figure gives 5.4 the edges detected in these images Note that depending on the background (e.g. grass and leaves) quite a lot of edges are generated.

The first step after edge detection is to segment edges into circular arcs. We tested three different methods. The first, simplest method starts with a minimal circular arc which is grown until the fit error exceeds a certain threshold. The second method by Rosin and West [RW95] starts with a maximal arc which is recursively split to obtain arcs with optimal significance. The third method uses RANSAC for fitting. Each of these methods uses a different heuristics to find an optimal segmentation.

In the following we will present the three methods and compare their results on some prototypical scenes. Please note that we do not seek perfect arc fits, which are impossible as arcs only approximate ellipses locally. The goal is to approximate relevant parts of edge segments by arcs such that subsequent grouping of these arcs will lead to good candidates for actual ellipses.

(a) kitchen

(b) office

(c) bike

(d) car

Figure 5.3: Four images with elliptical image structures.

5.5.1 Sequential Growing (GROW)

The first method puts emphasis on efficiency rather then accuracy. Therefore we do not fit a circle through all pixels but only use the minimal set of 3 pixels to find the circle centre and radius, which is illustrated in Figure 5.5. Given pixels i to k, find pixel $j = (i + k)/2$ (Figure 5.5(a)). Then intersect the lines bisecting lines $\overline{ij}$ and $\overline{jk}$ to find the circle centre c and radius (Figure 5.5(b)).

Given an edge with l pixels we want to decompose it into subsets of consecutive pixels which fit a circular arc reasonably well. We start with $i = 0$ and $k = 2$, i.e. with three points which by definition lie on an arc. Then we increase k, adjusting the fit until one of the pixels i to k lies outside a fixed distance threshold d_{max} of the fitted arc. We then store pixels i to k as an arc, set $i = k+1$, $k = i+2$ and repeat. The whole procedure ends when $k = l$ is reached. d_{max} was set to 1 pixel.

Figure 5.8(a) shows some piece of kitchen equipment with one detected edge highlighted. The edge follows the top rim of the piece and then picks up part of an occluding object. Situations like these often arise when the background is uniformly coloured and object/background edges are strong compared to ob-

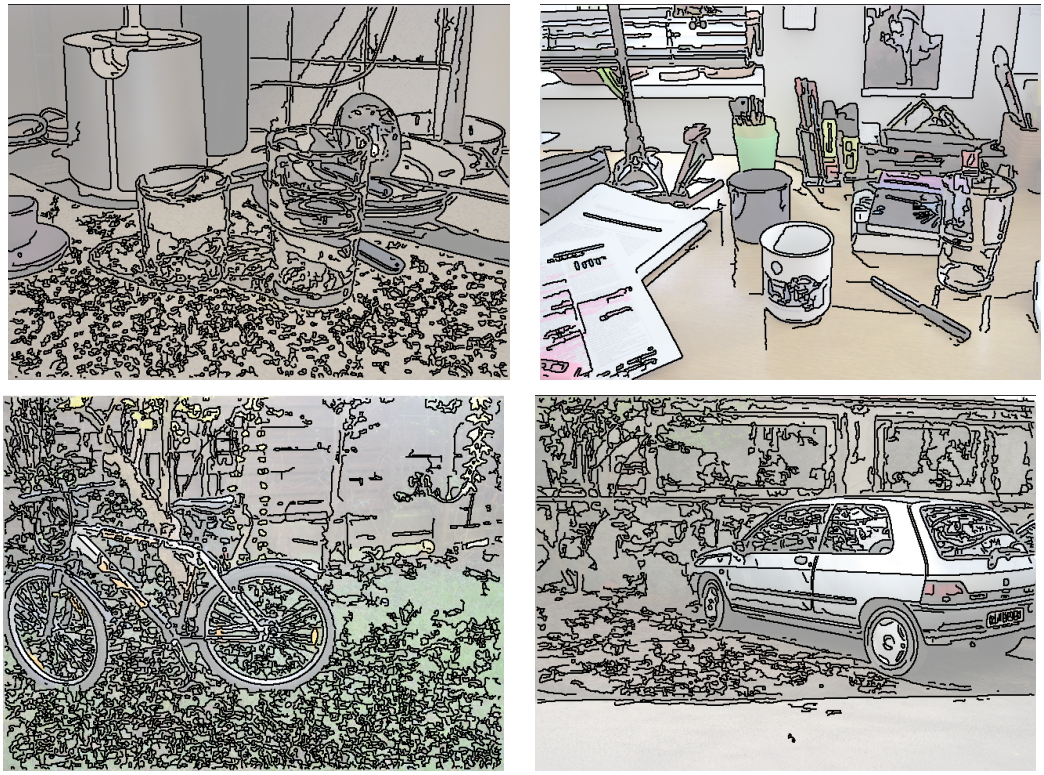


Figure 5.4: Edges detected in the above images.

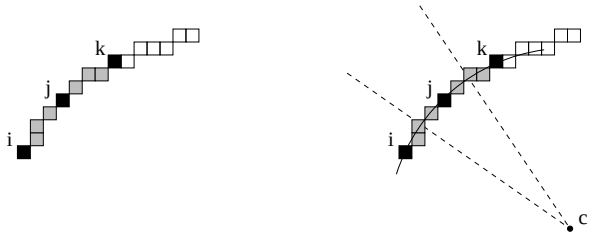
ject/object edges. Figure 5.8(b) shows the result of method GROW on the edge highlighted in Figure 5.8(a). Note that arcs a to c approximate the top rim ellipse reasonably well, while arc d gets distracted by the occluding object and thus points in the wrong direction.

The above method has the advantage of being truly simple and thus efficient. However the segmentation result depends on which pixel is labelled as 0, i.e. at which end the algorithm starts. Moreover a distance threshold is required, which makes the approach sensitive to scale. The next method alleviates these problems.

5.5.2 Recursive Splitting (SPLIT)

The method by Rosin and West [RW95] is based on recursively splitting a curve, given as a list of pixels, until a near optimal segmentation into a set of given models is achieved. A model can be any kind of parametric model. The authors used lines, circular arcs and elliptical arcs. Also a mixed representation of different models is possible. We are however at the moment only interested in arcs.

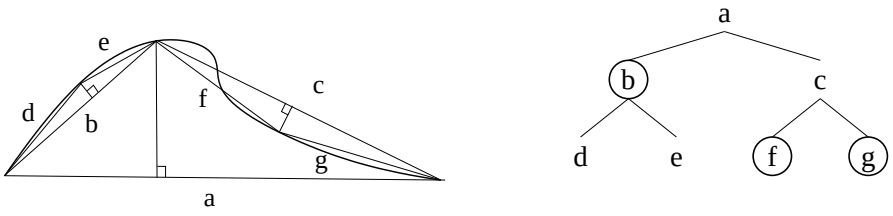
We will first show the idea for the case of lines, illustrated in Figure 5.6. First the whole list of pixels is approximated by a single line. The list is then split into two at the point of maximum deviation between the approximation and the pixel data. This process of approximation and splitting is repeated recursively



(a) (b)

Figure 5.5: Fast arc fitting: Given pixels i to k lie on an arc (a), find center c and check pixel $k+1$ (b).

until lists of size two are obtained. The result of this segmentation process is a segmentation tree, where each level describes the whole curve by increasingly fine segmentations and approximations. Now the problem remains to find the best level of representation. Each node (i.e. fitted line) is assigned a significance value (the significance measure of Lowe [Low87]). The significance is calculated as the ratio of the length of the fitted line divided by the maximum deviation of the curve from the line. The tree is now searched for the best set of lines. Traversing up the tree from the leaves, a line is retained if it is more significant than all its children, otherwise it is replaced by its children. After traversing the tree the remaining lines form the best segmentation of the curve (the circled lines in Figure 5.6(b)).

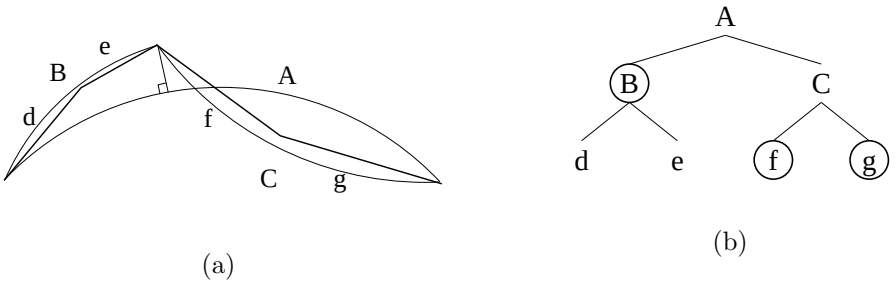


(a) (b)

Figure 5.6: Recursive fitting of lines and resulting segmentation tree.

Fitting arcs is performed in a two stage process. First the curve is segmented into lines, as outlined above. Then the sequence of line segments is segmented into arcs by fitting to line endpoints (see Figure 5.7(a)). Sequences of lines are again segmented at the line endpoint with maximum deviation and a mixed representation tree is constructed with arcs (denoted by capital letters) forming nodes and lines forming the leaves (see Figure 5.7(b)). The tree is again traversed up from the leaves and features with maximal significance are retained. In Figure 5.7 lines e and f are more significant than arc C and thus the final segmentation is Bef. The result of the two stage procedure is a mixed representation of lines and arcs. Note that for our purposes we only need arcs, therefore all lines are replaced by

arcs.



(a) (b)

Figure 5.7: Recursive fitting of arcs and resulting segmentation tree.

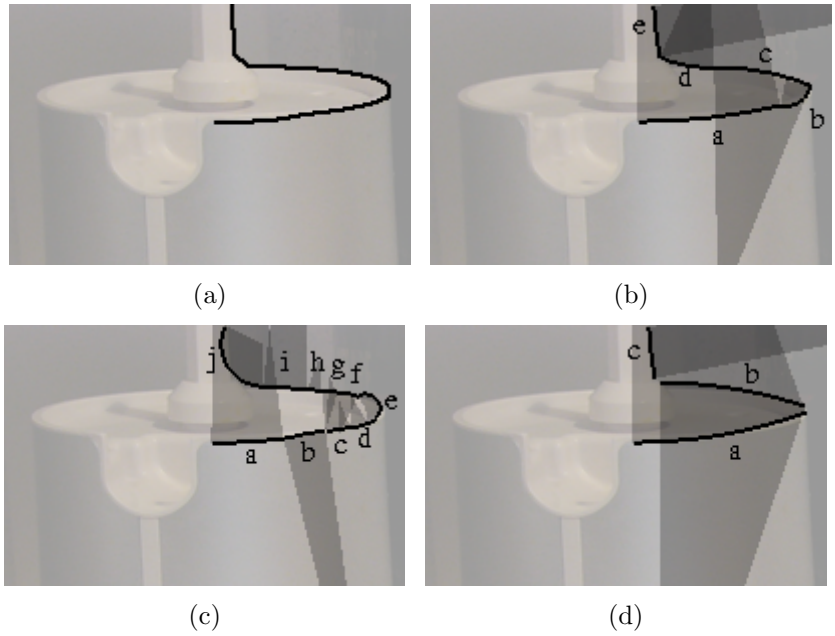
The above method has a list of desirable properties. First of all it is parameter-free. Segmentation is purely based on significances, which are calculated automatically. This also makes the method invariant to scale. Moreover the same principle of approximating and splitting can be applied to any parametric model.

While the segmentation result is invariant with respect to translation, in-plane rotation and scale, the global shape of the curve does affect the local segmentation results. For a complicated curve shape the first approximation will be quite meaningless and consequently the first split point. This can lead to a rather sub-optimal segmentation as can be seen in Figure 5.8(c). Several poor choices of split points lead to a fragmentation of the edge segment into many very short arcs, which often point in the wrong direction. Like GROW also SPLIT is distracted by the occluding object leading arc j to point in the wrong direction.

Remark

Why do we not fit elliptic arcs as the authors propose? After all we eventually want to detect ellipses, so detecting elliptic arcs seems a natural first step.

The first answer is that the simpler the model the more robust is a local fit. Note that we often have rather short segments for fitting, which can be too short for robustly fitting a model with many parameters. Consequently split points (points of maximum deviation) are chosen arbitrarily and the performance of the method breaks down. Figure 5.9 (images produced using original software by Rosin and West [RW95]) shows a detail of the CD player scene of Figure 5.2 with edges detected by a Canny edge detector (Figure 5.9). Figure 5.9(b) shows shows that fitted straight lines approximate the physical straight edges reasonably well and (as is to be expected) split curved edges into many small lines (crosses mark line ends). Figure 5.9(c) shows how 4 fitted arcs approximate the CD quite well while still approximating the straight edges with reasonable arc hypotheses. Figure 5.9(d) shows fitted ellipses. Although the CD actually is an ellipse it is approximated even worse than in Figure 5.9(c). Moreover very bad approximations are found for



(a) (b)

(c) (d)

Figure 5.8: Detail of a kitchen scene with one edge segment highlighted (a). Results of method GROW (b), SPLIT (c) and RANSAC (d).

straight edges. Note also the spurious circle next to the CD, which is the result of a very poor fit to a short segment.

An obvious solution to the above problem is to segment an edge into straight lines *or* arcs, which is also proposed by Rosin and West An edge is initially split up recursively into straight lines. Then arcs are fitted through endpoints of consecutive lines. Whenever an arc spanning some lines has a higher significance than these lines, the lines are replaced by the arc. Figure 5.10 (images produced using the authors original software) shows the problems arising when applying that method to the kitchen scene of Figure 5.2(b). Some of the edges which make the top rim of the mugs are approximated satisfactorily by elliptic arcs (Figure 5.10(b)) while for others the line representation is kept (Figure 5.10(a)). It is not clear when one or the other representation is chosen. Rosin and West provide a weighting factor to bias decision in favour of ellipses. Which in turn raises the question of how to choose this weight. The problem is that significances are not easily comparable across different geometric models. The main problem however is that this decision based on local evidence comes too early. For low curvature segments there will always be ambiguity between straight lines and arcs. Depending on the weight factor one will tend to make errors in one or the other direction.

This is the reason we restrict fitting to lines and arcs, where the above method works very well. We then keep both representations. One representing a segment as a sequence of lines (to be used in Chapter 6 and one representing it as a sequence of circular arcs. We chose circular arcs as we have seen that elliptic arcs give poor

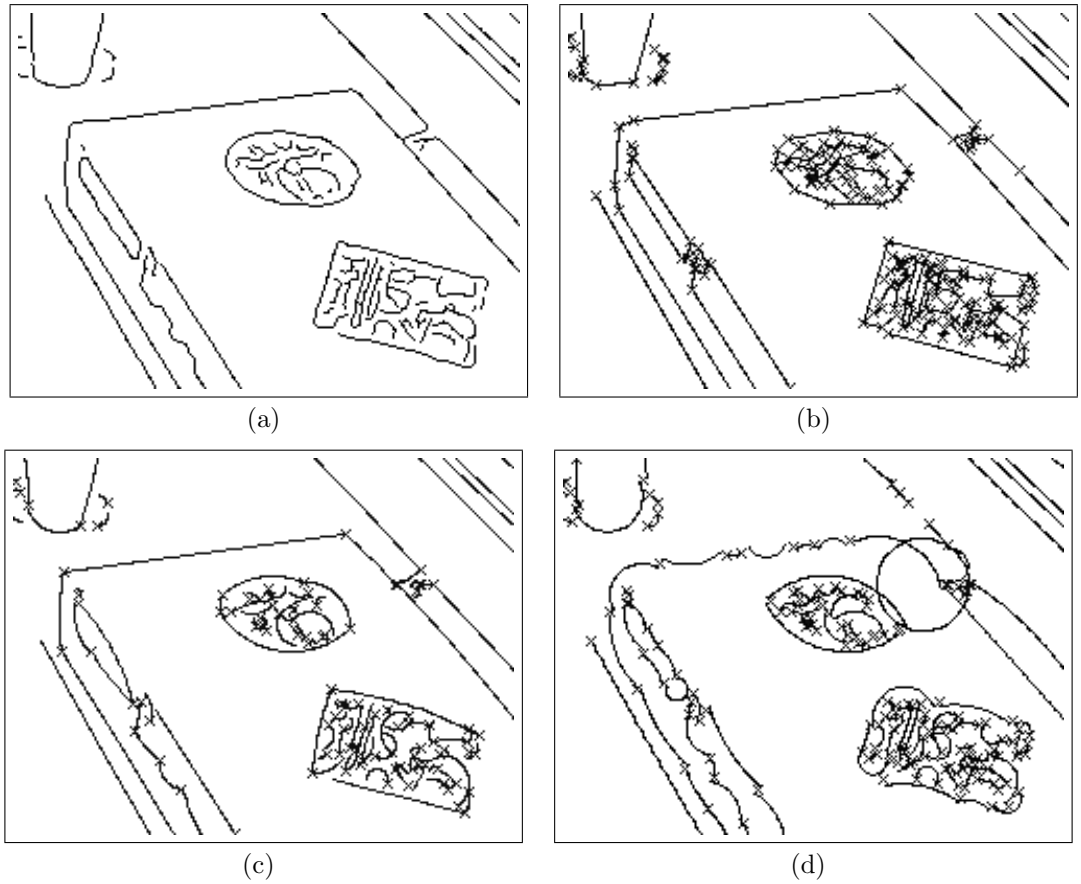


Figure 5.9: Edge image (a) and fitted straight lines (b), circular arcs (c) and elliptic arcs (d).

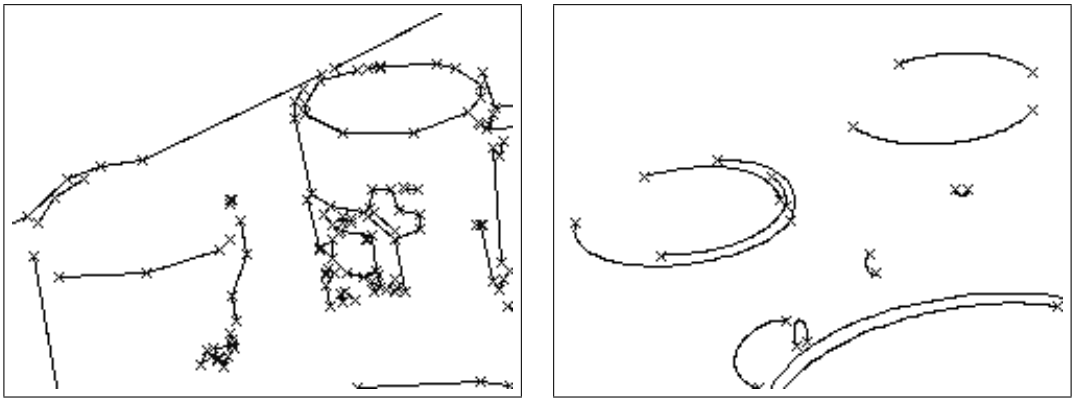
fits for low curvature segments.

The second answer for choosing circular rather than elliptic arcs is that we fit geometric models to edges in order to make geometric statements, i.e. formulate geometric relations. Circular arcs allow the same geometric statements as elliptic arcs (e.g. inside/outside, tangent directions). The (typically, in good situations) more accurate fit and higher expressiveness of an elliptic arc are outweighed by the problems with elliptic arcs mentioned above.

5.5.3 RANSAC

Method GROW is affected by the start point, SPLIT by the global shape of the curve. The last method uses random sampling to avoid such biases. RANSAC [FB81] is a well known robust estimator and has found many applications in computer vision.

RANSAC proceeds as follows. Repeatedly subsets of the input data are selected randomly and the model fitting the sample is computed. Then the quality of the



(a) (b)

Figure 5.10: Fitting lines (a) *or* elliptic arcs (b).

estimation is evaluated on the rest of the input data, typically by counting the number of inliers, i.e. data points supporting the model. The process is terminated when the probability of finding a better model becomes lower than a user-defined probability. RANSAC can deal with any percentage of outliers - it will only require more samples.

In our case we proceed as follows. Given an edge segment with pixels 0 to l we repeatedly select 2 random points i and k and calculate a third point $j = (i+k)/2$. We then fit a circular arc as in method GROW and calculate the number of inliers, where we require all inliers to be pairwise connected. Concretely this means that all points i to k must be inliers and we then extend the range $[i, k]$ in both directions until an outlier is met, resulting in range $[i', k']$. We draw a fixed number of n samples and remember the model with the highest number of inliers. As in method GROW inliers are defined via the distance threshold $d_{max} = 1$. We require a minimum of $l_{min} = 6$ inliers, as this is the minimum number of points needed for ellipse fitting in a later processing step. We then recursively call RANSAC on the remaining parts $[0, i' - 1]$ and $[k' + 1, l]$ of the segment. If no arc could be found after n trials, we split the edge segment at a random point j' and recursively call RANSAC on intervals $[0, j']$ and $[j' + 1, l]$.

Note that we use a slightly modified version of RANSAC. In the general case the number of samples depends on the outlier ratio and the desired probability to find the optimal solution. We however only draw a small fixed number of samples. We found that $n = 10$ is enough. Why is this small number of samples enough? In the optimal case where the whole edge segment actually is an arc just one sample is enough. In the general case of a wiggly shaped edge segment as in Figure 5.8(a) only small parts of the whole segment can be fitted by arcs. In that case if we do not find a satisfactory model after a few samples, there is no hope anyway that a single arc will fit the whole edge segment. Therefore we split the problem, knowing that this will necessarily get us nearer to the solution. I.e. there is no need to try

image	edges	GROW		SPLIT		RANSAC	
		arcs	time [s]	arcs	time [s]	arcs	time [s]
kitchen	3614	1772	0.220	1332	0.340	1603	0.260
office	1050	773	0.170	691	0.280	693	0.190
bike	6630	3001	0.280	2311	0.430	2730	0.410
car	2995	1587	0.220	1287	0.320	1453	0.250
avg/img			0.223		0.343		0.278

Table 5.1: Runtimes for fitting circular arcs.

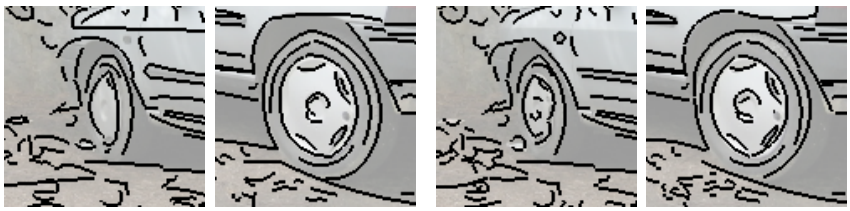
harder to find the optimal solution on the whole set of points.

Figure 5.8(d) shows the result of RANSAC. We see that the top rim ellipse is approximated very well with only two arcs and is kept separate from the occluding object. Note that there are small gaps between the arcs: For these short segments no satisfactory arc could be fitted. This can be considered a drawback of the method. In our case however later stages of processing are not affected by these small gaps.

5.5.4 Comparison of Arc Segmentation Methods

Table 5.1 shows the runtimes of the three arc segmentation algorithms for the images in Figure 5.3. Method GROW is always fastest, followed by RANSAC and SPLIT. Figure 5.11 shows that all three methods typically produce visually similar results, especially if segments tend to be short. Difficult cases like Figure 5.8 are handled better by RANSAC.

(a) edges (b) GROW



(c) SPLIT (d) RANSAC

Figure 5.11: Car details with detected arcs.

It is difficult to properly evaluate segmentation at this low level. Obtaining

ground truth by labelling all arcs in an image is tedious and moreover what would we consider a good segmentation? Many small arcs with small fit errors or fewer but longer arcs with larger fit errors? That ultimately depends on what we want to do with the segmented arcs. We therefore judge the quality of arc segmentation by how well it serves the overall goal of ellipse detection. Evaluation of the above methods on the complete evaluation data set is shown in Figures 5.12 and 5.13, showing true positive rate (TPR) of detected ellipses and run times for the complete ellipse detection process respectively. TPRs and run times are median values taken over all combinations of the other parameters of the complete ellipse detection process (such as convexity criterion and search method, to be explained later) and over all images of a scene type (see Section 5.4 for details on the evaluation data set).

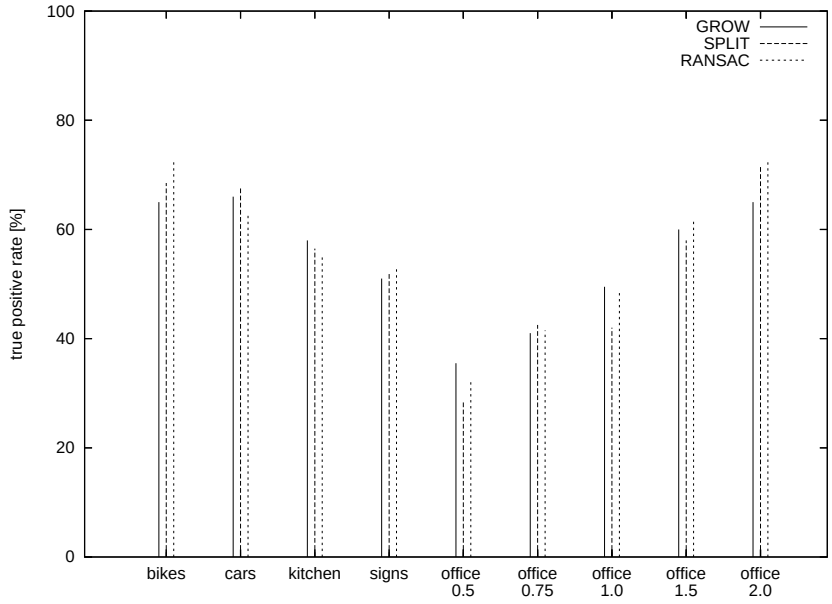


Figure 5.12: Arc segmentation methods: true positive rate.

We notice that the arc segmentation method plays a minor role in overall performance and run time. SPLIT typically performs best followed by RANSAC. GROW is typically fastest again followed by RANSAC. RANSAC therefore is a good compromise between performance and speed and seems to be the better choice especially in larger (more detailed) images such as the larger versions of the office images ("office 1.5" and "office 2.0" in Figure 5.13).

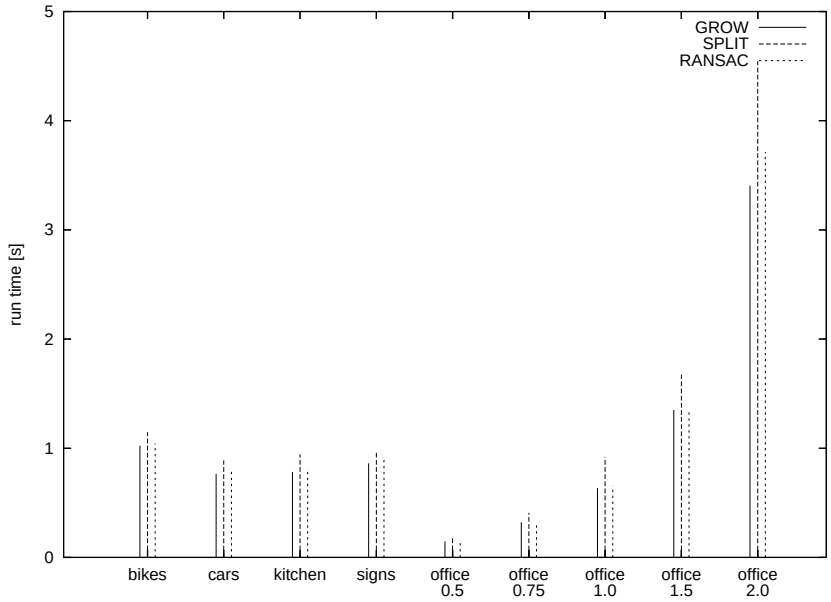


Figure 5.13: Arc segmentation methods: run times.

5.6 Finding Convex Groups of Arcs

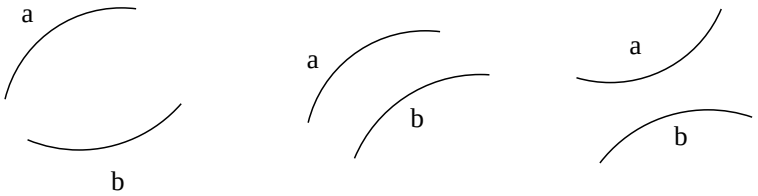
We have now identified strings of edgels which can be approximated by circular arcs and thus are likely to be part of an ellipse. But circular arcs approximate ellipses only locally. Moreover edges are sometimes broken due to poor contrast. Consequently the edge delineating the contour of an elliptical image structure will be broken up into several circular arcs. The next step thus is to identify groups of arcs, which form good candidates for ellipses.

5.6.1 Convexity Criteria

A necessary condition for such a group of convex arcs is that they be pairwise convex (see Figure 5.14). I.e. given two arcs a and b, arc a must lie "inside" arc b and vice versa.

We test convexity using two simple heuristics. The idea behind the convexity tests is to define half-planes using the arc tangents at end points and test whether all points of the other arc lie within these half planes (see Figure 5.15(a)). Let arc a be defined by start point S, end point E and centre C, where points are counted counter-clockwise along the arc. Point S and normal vector $\vec{SC}$ define a half plane. Each point P with $\vec{SC} \cdot \vec{SP} > 0$ lies "inside". Likewise with point E and normal vector $\vec{EC}$.

Some points might be classified incorrectly, as point R in Figure 5.15(a) and Figure 5.15(b) shows that this condition becomes increasingly meaningless as the



(a) (b)

Figure 5.14: Convex (a) and non-convex arcs (b).

angular coverage of the arc increases. We therefore add a second test: We require a point to lie "outside" the angular coverage of the arc. For angular coverage $< \pi$ this means $\vec{SC} \times \vec{SP} > 0$ *or* $\vec{EC} \times \vec{EP} < 0$, where $\times$ is the cross product between 2-D vectors, i.e. point P must be left of S *or* right of E (see Figure 5.15(c)). For angular coverage $\geq \pi$ this means $\vec{SC} \times \vec{SP} > 0$ *and* $\vec{EC} \times \vec{EP} < 0$, i.e. point P must be left of S *and* right of E (see Figure 5.15(d)).

As it is too time consuming to test for every point of arc b whether it lies "inside" of arc a we just test two points. Testing the end points of b, which might be a first choice, leads to problems as in Figure 5.16, a detail of Figure 5.3(b), which shows the arcs detected on the top rim of the dark blue mug. Here two two good arcs a and b are mis-classified as non-convex because end point P of arc b lies "outside" of arc a. This typically happens if the object of revolution has rounded edges, as exemplified by the mug in Figure 5.16. Then due to changing lighting conditions around the rim several slightly offset arcs are detected. These arcs are strictly speaking not convex but they nonetheless are part of the same elliptical structure and should thus be grouped.

One way to overcome this problem is to explicitly allow for a small offset δ by changing the criterion to $\vec{SC} \cdot \vec{SP} > -\delta$, where δ would e.g. be 2 pixels. Such a threshold however is problematic. The offset between two arcs in the image depends on several factors: the local structure of the object of revolution (e.g. the thickness of the rim), the lighting conditions (highlights and shadows along the contour) and image resolution. Introducing a further parameter δ and setting it to an arbitrary value therefore is not a good solution.

The actual problem is of course that the end points are most sensitive to small offsets. We therefore choose points further away from the ends, namely at one third and two thirds of arc length. This will allow small overlaps and thus makes the test more robust.

An even "weaker" test is to use only the half-plane defined by mid point M (see Figure 5.15(e). This will of course admit very non-convex cases as in Figure 5.15(f) but makes sure that no possibly convex pair of arcs is missed.

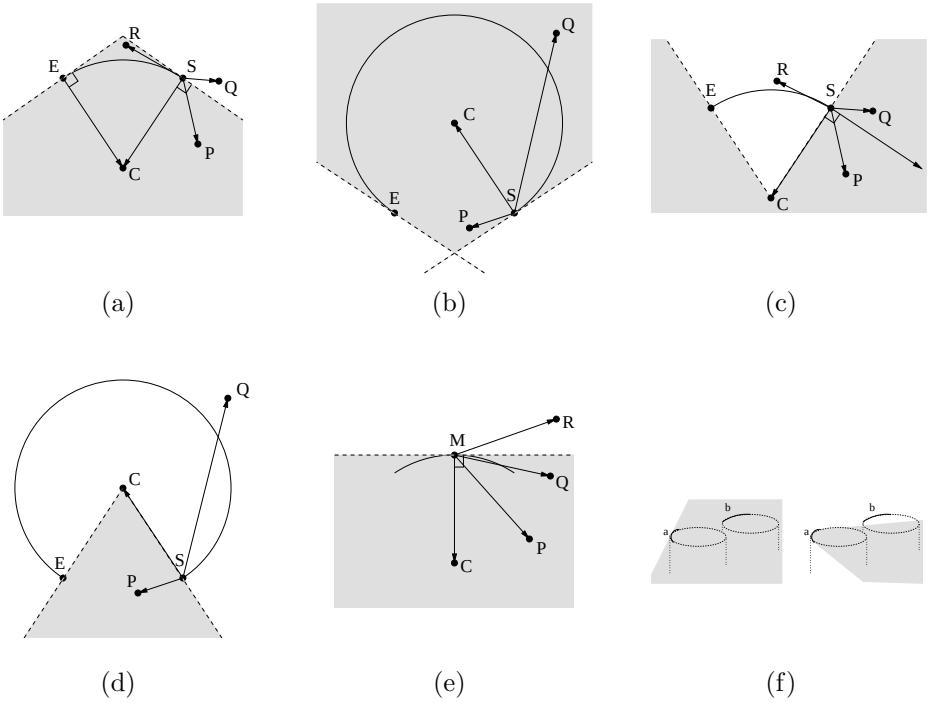


Figure 5.15: Convexity criteria.

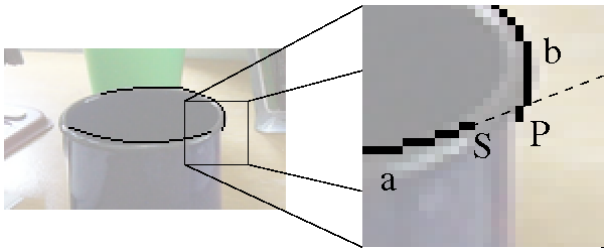


Figure 5.16: Problem with offset arcs.

5.6.2 Comparison of Convexity Criteria

Figure 5.17 shows the TPR of ellipse detection using the weak and strong convexity criterion. Again values are medians taken over all combinations of the other parameters of the complete ellipse detection process (such as arc segmentation method and search method) and over all images of a scene type. We see that the weak criterion consistently achieves a higher performance because the risk of rejecting a "good" set of arcs as non-convex is smaller. This comes with a price however. Figure 5.18 shows the median and maximum runtimes. Ellipse detection using the strong convexity criterion is always faster, because fewer convex groups of arcs are formed (see next section). The difference becomes especially apparent in the maximum runtime, which can grow substantially using the weak criterion.

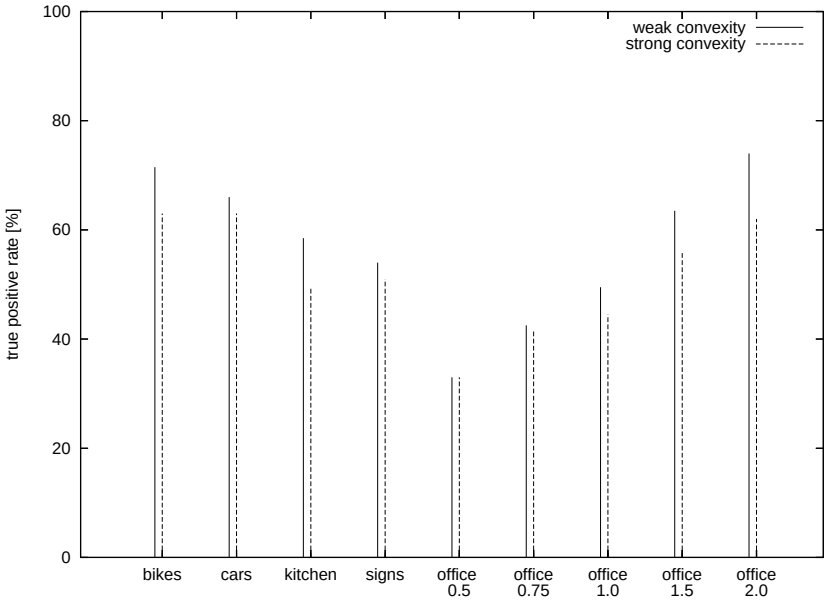


Figure 5.17: Convexity criteria: true positive rate.

Note for example the maximum runtime for the "bikes" scenes, where the rims and tyres typically form many concentric ellipses. Here the weak criterion allows creation of a vast number of convex groups. We can see a runtime/TPR trade-off: better TPR has to be paid for by higher runtimes.

5.6.3 Identifying Convex Arcs - Image Space Indexing

Now that we have a robust convexity criterion the problem remains to identify those arcs which should be tested for convexity. Testing each arc against each other arc results in a runtime complexity of $O(n^2)$, where n is the number of arcs, which is prohibitive for large numbers of arcs. We note that most arcs checked are non-convex anyway. Moreover typically only close arcs are of interest and will eventually result in useful ellipse hypotheses.

Looking at the examples of convex arcs given so far we notice that for a pair of arcs a and b which are neighbours on the contour of an ellipse (and it is only those that we are interested in) at least one of the following situations will occur: Tangents of opposing ends will intersect (Figure 5.19(a)) or (for parallel tangents and offset arcs) tangents and radial lines will intersect (Figure 5.19(b)). We therefore construct four search lines for each arc: Tangents t_s and t_e at the start and end point and radial lines r_s and r_e, where again start and end are counted counter-clockwise along the arc (see Figure 5.19(c)). To identify pairs of arcs for further convexity testing we now look at intersections between the search lines of different

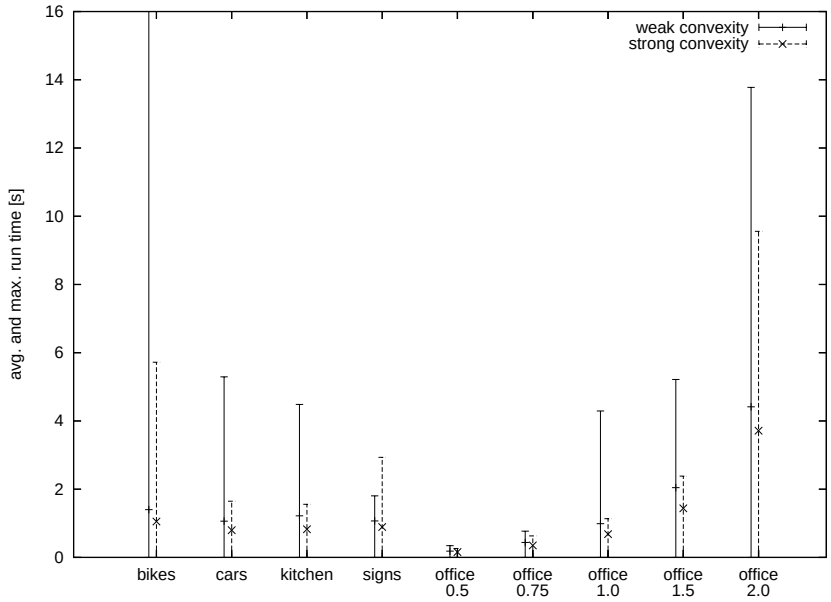
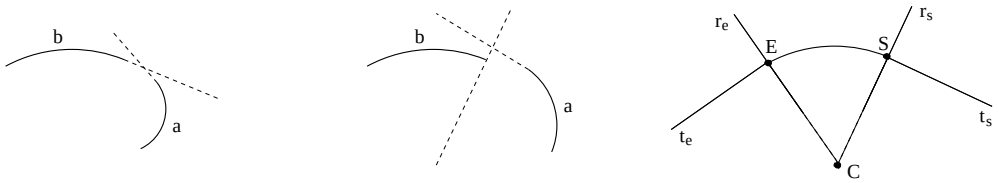


Figure 5.18: Convexity criteria: average and maximum run times.

arcs.

Finding intersections can be realised very efficiently using indexing. Typically indexing (often the term voting is used synonymously) is performed in the parameter space of a model which is to be fitted to a set of data points, as e.g. in the classical Hough transform. In contrast we perform indexing directly in the image space, i.e. the bins are the image pixels themselves. For each arc we draw the search lines t_s, t_e, r_s and r_e into the image, pixel by pixel using the well-known Bresenham line drawing algorithm. The "colour" of a pixel is the identification number of the originating arc plus an identifier denoting the type of line (tangential or radial) and the end (start or end). In that sense each search line casts votes for intersections. Whenever we colour an already coloured pixel this means that two arcs vote for the same intersection.



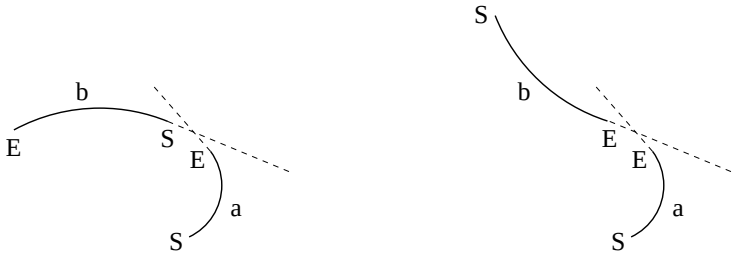
(a) Two tangents intersect (b) Tangent and radial line intersect (c) Search lines

Figure 5.19: Relations between neighbouring arcs.

If we found such an intersection we first perform a fast "daisy chain" convexity

test, which ensures that the correct ends of two arcs meet: Only an intersection between a *start* and an *end* search line can lead to a pair of convex arcs (see Figure 5.20). This test will reject many intersections which would not pass the convexity test anyway.

If a pair of arcs passes all tests a link will be created between these arcs and stored for later stages of processing.



(a) Possibly convex arcs (b) Arcs can not be convex

Figure 5.20: Daisy chain constraint.

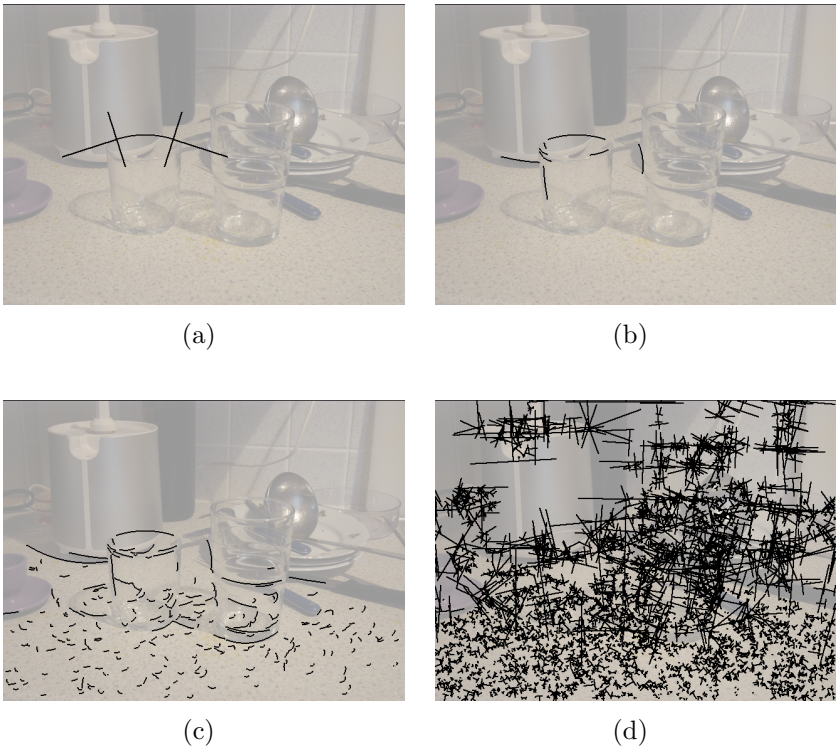
Parameters

The only parameter to choose is the length of the search lines. It should be large enough to capture all promising arcs in the neighbourhood of an arc. By promising we mean those arcs which are likely to stem from the same ellipse. Setting the length to infinity would just again test each arc against each other, because the search lines of any pair of arcs intersect somewhere, making the whole procedure pointless. Practically of course line length is limited by the image border. Still, drawing lines of maximal length will waste a lot of effort into testing many small arcs which lie far apart and have a very low likelihood of eventually forming an ellipse.

Therefore search lines should be limited to a "useful" length. To be invariant against scale line lengths for each arc are set to $s = min(r, l)$, where r is the arc radius and l is the arc length.

Figure 5.21(a) shows a selected arc of Figure 5.3(a) with search lines of length s. Figure 5.21(b) shows all convex arcs found using the above procedure and search lines of length s, while Figure 5.21(c) shows all convex arcs found using search lines of maximal length (drawn to image border). Considering the selected arc we want to find the top rim ellipse of the glass, we see that maximally extending the search lines does not give us any new promising arcs and thus makes little sense. Figure 5.21(d) shows all search lines of length s in the image. Note that showing all search lines of maximal length just results in a black image as each pixel is coloured several times.

We will see later in Chapter 6 how to get rid of this parameter as well.



(a) (b)

(c) (d)

Figure 5.21: Search lines of length $s = min(r, l)$ (a), convex arcs found using search lines of length s (b) and using search lines of maximal length (c). All search lines of length s (d).

Runtime Complexity

The complete algorithm for finding convex pairs of arcs is listed in Figure 5.22. We see that we have to perform in the average $4s_{avg}n$ pixel operations, where n is the number of arcs and s_{avg} the average length of a search line, which can never exceed the image size and can thus be regarded as a fixed constant. The runtime complexity is therefore $O(n)$.

Note that this approach does of course not find *all* pairs of convex arcs, as opposed to checking all pairs in quadratic runtime. But that is exactly what we want: a heuristics to efficiently find only promising pairs. (Setting the length of search lines to infinity we would indeed find all pairs (see Figure 5.21(c)) resulting again in a quadratic complexity.)

Figures 5.23 and 5.24 show the runtimes for various images with different numbers of arcs (the example images of Figure 5.3 are indicated). Compare the quadratic runtime of testing all arc pairs, corresponding to Image Space Indexing with infinitely long search lines (Figure 5.23) versus the linear runtime of Image Space Indexing with search lines of length s (Figure 5.24). Note that linearity of runtime in the latter case is maintained for a wide range of arc numbers.

```
FindConvexPairs(n)
  for i = 1 to n
    DrawSearchLine(i, TANG_START)
    DrawSearchLine(i, TANG_END)
    DrawSearchLine(i, RAD_START)
    DrawSearchLine(i, RAD_END)

DrawSearchLine(id, end)
 for each pixel (x,y) on line
   if Free(x,y)
     SetPixel(x, y, id, end)
   else
     if End(x,y) != end
       if Convex(Id(x,y), id)
         CreateJunction(Id(x,y), id)
```

Figure 5.22: Finding convex pairs of arcs

A Small Remark on Voting The (Randomised) Hough Transforms (RHT) of [YIK89, McL98] collects evidence for ellipses before actual ellipse fitting. (R)HT is a voting scheme in parameter space, accumulating evidence globally from the whole image. The voting points are totally unrelated (hence *randomised*), except for the common geometric model (the ellipse). I.e. nothing can be said about those points without a specific model. Moreover (R)HT has problems with noisy images. The globality of the accumulation means that distant (in terms of image distance) noise will affect the solution.

Our method represents a voting in image space, collecting evidence locally. Voting arcs are related by Gestalt principles expressing physical properties of objects (e.g. locally smooth surfaces). They are related *before* applying the specific geometric model (ellipse). Their relatedness is of a more general form. The perceptual grouping is valid in any case, with or without a specific geometric model.

5.6.4 Growing Convex Groups

The above section presented a robust and efficient way to identify convex pairs of arcs in images possibly containing large amounts of clutter. Now that we have identified arc pairs we can grow groups of pairwise convex arcs which will then serve as candidates for ellipses. Starting from a group which initially consists of only one arc we extend the group in both directions until the ends meet or no more arc can be added without violating the convexity of the group.

Every arc typically has several convex neighbours at each end (see Figure 5.21(b)). So starting from an initial arc we can form many convex groups. Finding the optimal group is a combinatorial search problem. We evaluated two different search procedures: exhaustive search and greedy search. Moreover for the greedy search we evaluated three different heuristic functions.

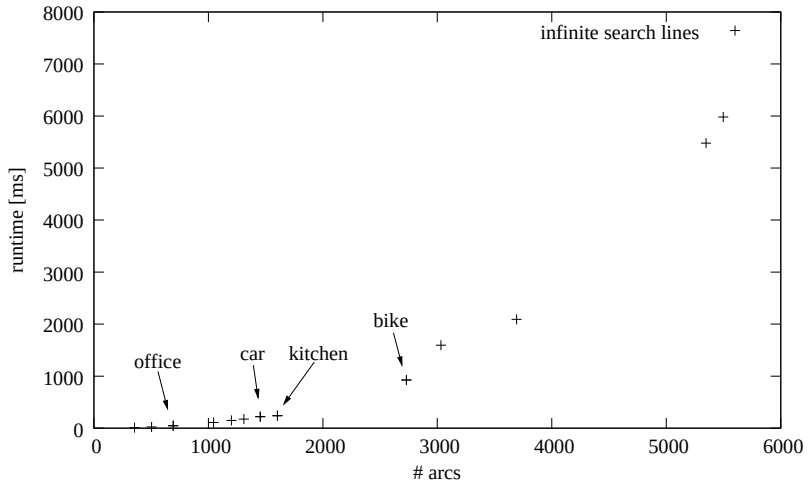


Figure 5.23: Identifying pairs of arcs for various images with different numbers of arcs (including example images of Figure 5.3): Quadratic runtime for infinite search lines (i.e. testing all arc pairs).

Exhaustive Search

Given a group initially consisting of only one arc exhaustive search creates an extended group for each convex neighbour of both ends of the current group and recursively again extends each of these groups. A group is finished if either the ends meet or no more arc can be added without violating the convexity of the group

This seems like a bad idea. Say each arc has 3 neighbours on each side. Extending the initial arc m times will result in 3^m groups, i.e. a combinatorial explosion. Luckily the mean number of neighbours per arc end is relatively small. The mean over the images in Figure 5.3 was 0.66. Most of the very short arcs which make up the majority find no neighbour at all. Still the maximum number of neighbours per arc end was 10, which could very well lead to combinatorial explosion. However the convexity constraint increasingly limits the number of admissible neighbours as the group grows. Each arc in the group must be convex with each other arc. So the more a growing group reaches closure the fewer new arcs are admissible.

Figure 5.25(a) shows a detail of Figure 5.3(d) with all arcs detected. Figure 5.25(b) shows all arcs explored during exhaustive search starting form the indicated arc. We see that actually only a fairly manageable set of arcs was explored (9 in this case) and 17 groups were formed starting from the indicated initial arc. A total of 2351 groups were formed for the whole image.

Greedy Search

Greedy search does not explore all neighbours of an arc when extending the group, but only the best neighbour according to some heuristic function. Given a group

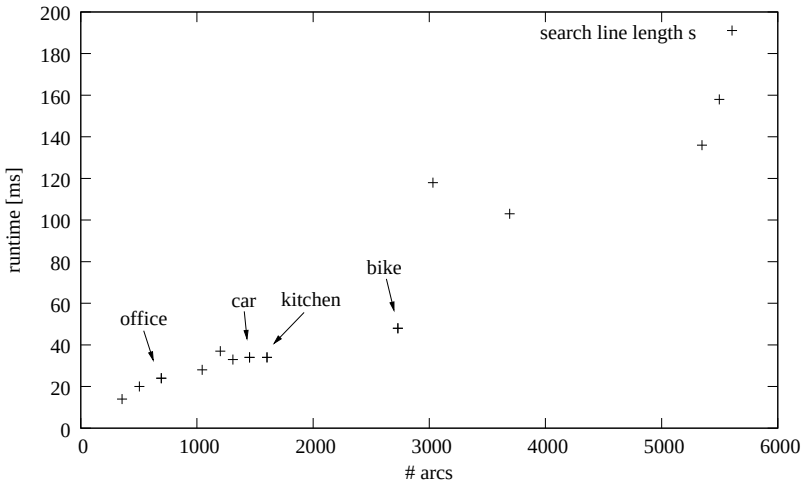
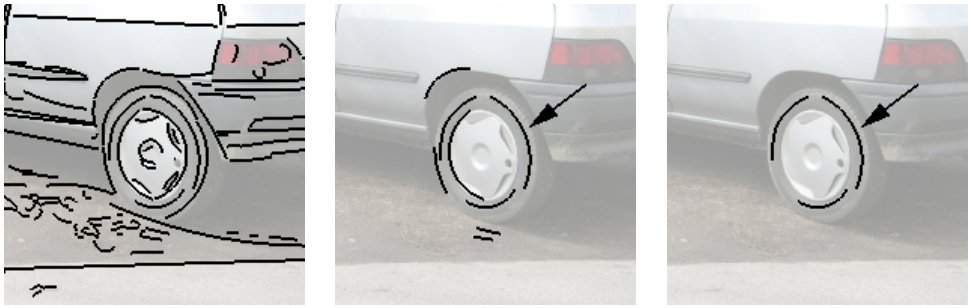


Figure 5.24: Identifying pairs of arcs for various images with different numbers of arcs (including example images of Figure 5.3): Linear runtime for search lines of length s.



(a) (b) (c)

Figure 5.25: Arcs explored by exhaustive search (b) and greedy search (c) starting at the indicated arc.

initially consisting of only one arc we extend the group with the best convex neighbour of both ends until either the ends meet or no more arc can be added without violating the convexity of the group. For defining the best neighbour we evaluated three different heuristic functions.

1. Co-curvilinearity of arcs.
2. Constraint on ellipse centre.
3. Quality of ellipse fit.

Moreover a group is only extended if its quality improves. We don't want to spoil a good group by adding a background arc which happens to be convex with the current group. For measuring group quality we also use the above heuristics.

Figure 5.25(c) shows all arcs explored during greedy search (using the co-curvilinearity heuristics explained below) starting from the indicated arc. In that case 4 arcs were explored to find the best group (note that the two arcs left of the indicated arc are so close as to appear as one arc). A total of 1325 groups were formed for the whole image.

Co-curvilinearity of arcs Given the list of all convex neighbours at both ends of the current group the first heuristic function selects the neighbour which best satisfies a co-curvilinearity criterion with the respective end of the group. Proximity and direction of endpoints as well as angular difference between tangents (see Figure 5.26) are used to define a significance measure σ_C for the co-curvilinearity between an arc and its neighbour (see Section 6.3 for details on the calculation of significances). The most significant co-curvilinearity is selected.

$$Pr_C = 1 - e^{-2nf_{prox}f_{angle}} \quad (5.1)$$

$$\sigma_C = -\log Pr_C \quad (5.2)$$

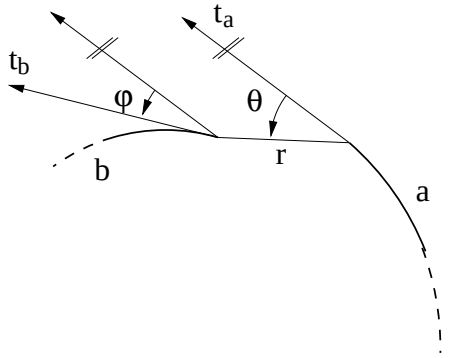
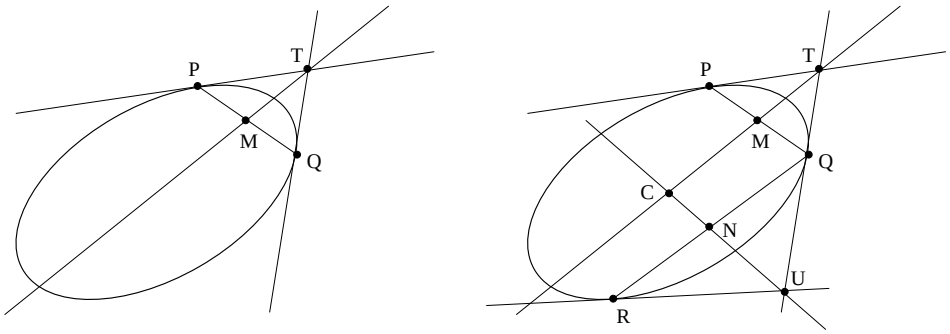


Figure 5.26: Calculating co-curvilinearity of arcs.

Constraint on ellipse centre. Co-curvilinearity only considers the ends of a group and is thus a rather local criterion. The second heuristic function considers the whole group. It extends a given group of n arcs with that neighbouring arc which results in the best extended group of $n + 1$ arcs. The quality of a group is defined by how good the arcs in the group agree over the ellipse centre. Concretely, several sets of arcs are selected from the group and the ellipse centre is estimated for each set. If the arcs are indeed part of the same ellipse, the estimated centres should coincide or at least lie close.

The centre of an ellipse can be found using a feature of ellipse geometry noted by Yuen et al. [YIK89] (see Figure 5.27(a)). Take two points on an ellipse and find their midpoint M and the intersection of their tangents T. Then the ellipse centre

must lie on the line $\overline{TM}$. Given three points P, Q, R on an ellipse one can thus estimate the ellipse centre C (see Figure 5.27(b)).



(a) (b)

Figure 5.27: Ellipse centre constraint.

If however the three points cover only a small angle on the ellipse the tangents will intersect at a very obtuse angle, resulting in an unstable intersection. Moreover the distance $\overline{TM}$ in such a case is very small, amplifying the error introduced by the obtuse intersection. An ellipse centre estimated from such a line $\overline{TM}$ will be fairly useless.

To cope with these cases we introduce weighting factors for intersections

$$w_{ab} = |d_a \times d_b| \tag{5.3}$$

where $\times$ is the cross product between 2-D vectors and d_a, d_b are unit direction vectors of the two lines to intersect.

Moreover we try to choose points from the group of arcs such that they are far apart, i.e. cover a large angle. For each arc consisting of m pixels we select 2 pixels $m/3$ and $2m/3$ and estimate tangents to the original Canny edge segment at these points. So for n arcs in a group we will have $k = 2n$ points, each arc creates 2 centre points. We now estimate k ellipse centres from points i, $i + k/3$ and $i + 2k/3$, where $i = 1..k$, and we wrap around at the end.

Each estimated ellipse centre C_i is weighted with the product

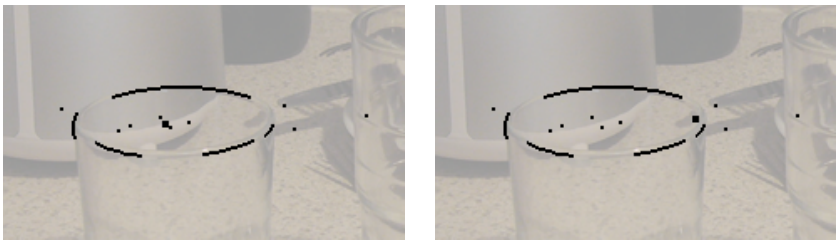
$$w_i = w_{PQ} w_{QR} w_{MN} \tag{5.4}$$

where w_{PQ} is the weighting factor from tangents to P and Q, w_{QR} is the weighting factor from tangents to Q and R and w_{MN} is the weighting factor from lines through M and N (see Figure 5.27(b)).

We then calculate the weighted mean of all k ellipse centres.

$$\bar{C} = \sum_{i=1}^{k} w_i C_i \tag{5.5}$$

De-weighting unstable intersections using the above weighting factors makes the estimation of the ellipse centre significantly more robust. Figure 5.28(a) shows the individual estimated ellipse centres and the weighted mean. Figure 5.28(b) shows how the arithmetic mean (without weights) is strongly affected by outliers resulting from unstable intersections.



(a) (b)

Figure 5.28: Ellipse centre estimated from a group of arcs using weighted mean (a) and arithmetic mean (b).

What remains to be done is to get a measure of the compactness of the "point cloud" around the mean. We could just measure the sum of weighted differences, using again the above weights. To be consistent with the rest of this work however we again use the non-accidentalness principle and the assumption that centre points are distributed following a Poisson process (see Chapter 4).

The parameter λ of the Poisson process is given by the density of ellipse centres $d = K/A$, where K is the total number of estimated centres C_i and A the image area. Note that we can not know K while we are still growing groups so we approximate K by $K' = 2N$, where N is the total number of arcs. Here we assume that on the average each arc is part of one group, and each arc creates 2 centre points (see above). This approximation typically comes close to within 5% of the true K. (Note that of course estimated centres might actually lie outside the image. This does not affect the above numbers if we consider an infinite image plane. Within each area A of that infinite image plane K' centre points are created lying anywhere in the image plane. This means that of course on the average within each area A there are K' centre points.)

We now sort points C_i in order of increasing distance r_i from $\bar{C}$ and calculate the probability $Pr(X \geq i)$ that *at least* i points lie within an area of size $A_i = \pi r_i^2$ around $\bar{C}$, for i ranging from 1 to k. That is one minus the probability that *at most* $i-1$ points lie in that area, which is given by the CDF of the Poisson distribution with parameter $\alpha = \lambda A_i$. As usual the significance is taken to be the negative

log-likelihood.

$$P_\alpha(X \leq i-1) = \sum_{j=0}^{i-1} e^{-\alpha} \frac{\alpha^j}{j!} \quad (5.6)$$

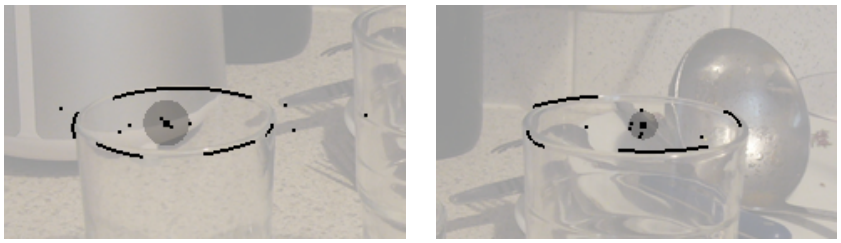
$$Pr(X \geq i) = 1 - P_\alpha(X \leq i-1) \quad (5.7)$$

$$\sigma_i = -\log Pr(X \geq i) \quad (5.8)$$

A high number of points within a small area will result in a high significance. Outliers (large r and thus large A_i) will result in a low significance. The significance of a group of convex arcs is the maximum

$$\sigma = \max_i \sigma_i \quad (5.9)$$

I.e. we find that set of points around $\bar{C}$ which is least likely to occur randomly and is thus most significant. The significance of that set of points defines the significance of the group of arcs. Figure 5.29 shows two examples of convex arc groups with shaded circles indicating the size of the area considered significant.



(a) (b)

Figure 5.29: Significant clusters of estimated ellipse centre points.

Quality of ellipse fit. The last heuristic uses the ellipse fit itself. Even before we have a complete group of convex arcs we can fit ellipses to partially completed groups. We extend a given group of n arcs with that neighbouring arc which results in the best ellipse fit through the extended group of $n+1$ arcs. Fit quality is measured by relative support (to be explained in the next section).

5.6.5 Comparison of Growing Methods

Figure 5.30 shows the true positive rates and Figure 5.31 the average and maximum runtimes of different methods. Again we see the runtime/TPR trade-off.

Figure 5.32 finally shows TPR vs. runtime for all combinations of methods (not distinguished in the plot) applied to different sizes of the office scene. We can see that generally TPR increases with image size, as does of course runtime, indicating again a runtime/TPR trade-off.

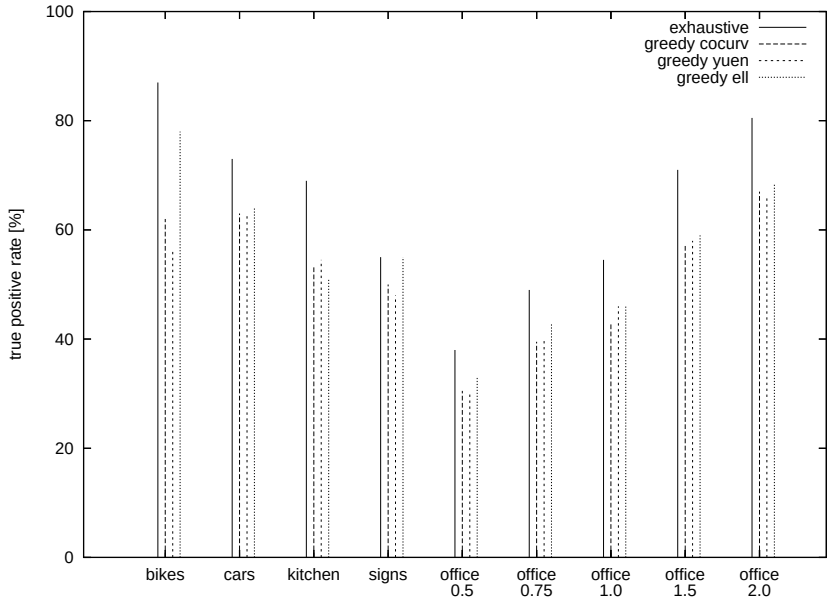


Figure 5.30: Search methods: true positive rate.

5.7 Fitting Ellipses

What we have done so far is to select groups of edgels from the image such that they are likely to belong to the same elliptical structure. What remains to be done is estimating the actual ellipse parameters. Because of the grouping of edgels so far we can consider all edgels to be inliers. Therefore the fitting method need not be robust with respect to outliers and we can use any least squares fitting method. We chose the direct least squares B2AC algorithm by [FPF99].

5.7.1 The B2AC Algorithm

Given data points $(x_i, y_i), i = 1..N$ we want to find the set of ellipse parameters $\mathbf{a} = [a\ b\ c\ d\ e\ f]^T$ for the ellipse given by the general conic equation

$$F(\mathbf{a}; \mathbf{x}) = \mathbf{a}.\mathbf{x} = ax^2 + bxy + cy^2 + dx + ey + f = 0 \tag{5.10}$$

where $\mathbf{x} = [x^2\ xy\ y^2\ x\ y\ 1]^T$ minimising the sum of squared algebraic distances

$$\mathcal{D}_A = \sum_{i=1}^{N} F(\mathbf{a}; \mathbf{x}_i)^2 \tag{5.11}$$

In order to avoid the trivial solution $\mathbf{a} = \mathbf{0}$ and to enforce that the estimated conic is an ellipse B2AC introduces the constraint

$$4ac - b^2 = 1 \tag{5.12}$$

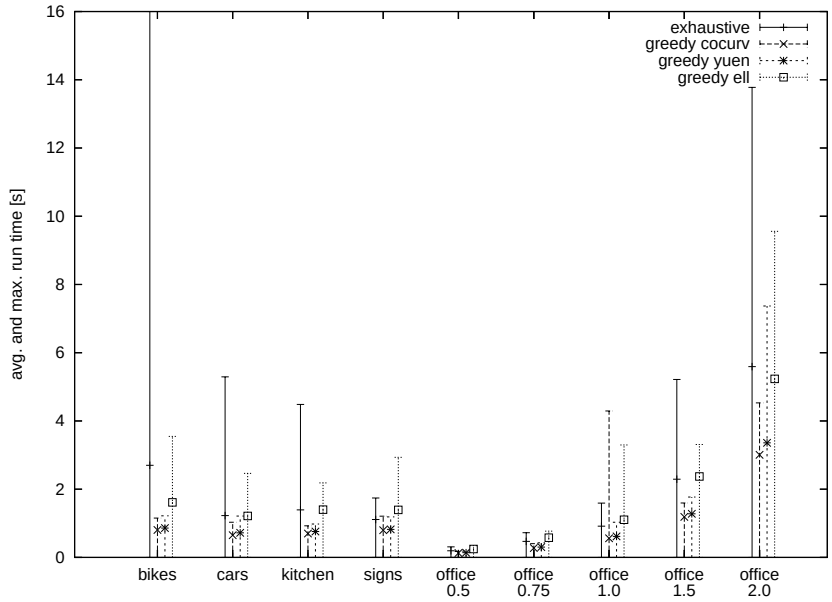


Figure 5.31: Search methods: average and maximum run times.

The system of simultaneous equations to solve becomes

$$S\mathbf{a} = \lambda C\mathbf{a} \tag{5.13}$$

$$\mathbf{a}^T C\mathbf{a} = 1 \tag{5.14}$$

where S is the scatter matrix $D^T D$ and D the design matrix $D = [\mathbf{x}_1\ \mathbf{x}_2\ \cdots\ \mathbf{x}_n]^T$. C is the constraint matrix and λ a Lagrange multiplier. The solution is given as

$$\hat{a}_0 = \mu_0 \mathbf{u}_0 \tag{5.15}$$

$$\mu_0 = \sqrt{\frac{1}{\mathbf{u}_0^T S \mathbf{u}_0}} \tag{5.16}$$

where $\mathbf{u}_0$ is the single positive generalised eigenvector of 5.15.

B2AC has been shown to be superior to iterative and non ellipse-specific methods and is robust to Gaussian noise. As noted above we have no non-Gaussian noise as we eliminated outlier edgels in preceding grouping steps.

5.7.2 Assessing Ellipse Quality

Figure 5.33 shows the ellipses fitted to the edgels of convex arc groups for the images of Figure 5.3. For this and all subsequent experiments in this section RANSAC was used for arc segmentation, the strong convexity criterion, and greedy search with ellipse criterion for convex grouping. Arguably the results look rather

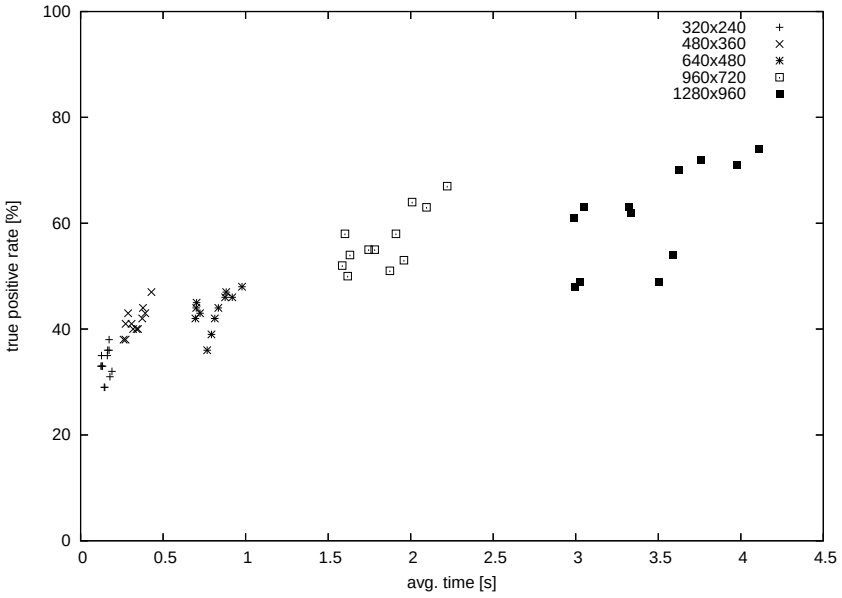


Figure 5.32: Runtimes vs. true positive rate for various image sizes of office scene.

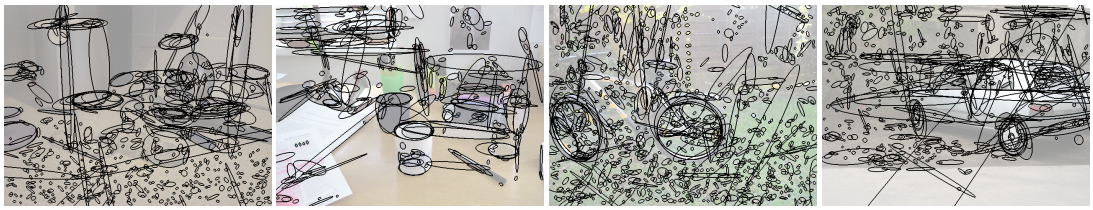


Figure 5.33: All detected ellipses.

pathetic: Some of the ground truth ellipses are missed while dozens of very poor ellipses populate the image. What happened?

One problem remains (or actually only becomes more apparent now): How do we judge the quality of the detected ellipses? Note that we never thresholded away any result, so we kept dragging along a lot of low quality "junk": Arcs that are actually almost straight lines or just tiny segments; convex groups of arcs distributed over the whole image coincidentally being pairwise convex; ellipses which happen to get support from some pixels anywhere in the image.

Figure 5.34 shows the results of Figure 5.33 after applying various thresholds regarding size, quality of fit or support. The results certainly *look* nicer but of course the thresholds only work for a particular setting (image size, expected size and number of ellipses etc.). So by applying cleverly chosen thresholds, derived from the user's constraints on what is an acceptable ellipse, one can "clean" the results. But this is certainly no general solution.

Keep in mind that we avoided thresholds for good reasons: An ellipse with low

support but a very low fit error might very well be the only un-occluded part of a wheel. On the other hand the fit error can be rather high if the circular structure does not project to a perfect ellipse (typically for rounded edges like mug rims or tyres). Enforcing lower thresholds on any value (support, fit error) will necessarily throw away some correct ellipses.

Figure 5.34: "Good" ellipses.

Therefore all we do is rank the ellipses according to some quality criterion. We evaluated four different criteria for assessing the quality of detected ellipses:

1. Weighed absolute support
2. Weighted relative support
3. Mean fit error
4. Significance

Masking Once we have a quality criterion we can employ *masking* to significantly reduce the number of ellipse hypotheses. Typically there will be many partly overlapping ellipse hypotheses around strong elliptical structures such as the wheels in the bike scene. Each arc however can only belong to one ellipse. Therefore if two ellipses claim the same arc, one of them must necessarily be wrong. Using the quality criterion we can decide which hypothesis is more likely to be correct. I.e. the stronger ellipse hypothesis will mask the weaker one. This is achieved easily be traversing the list of ranked hypotheses starting with the strongest and keeping a list of already claimed arcs.

Weighted Absolute Support The absolute support is simply the number of edgels lying within a distance threshold of the ellipse. Again we choose a distance threshold $d_{max} = 1$. Note that the actual value of d_{max} is not that important. A higher or lower value will increase or decrease the support values for all ellipses alike. And as we are only interested in the relative ranking of ellipses, the actual values of support are not of interest.

Just counting the number of supporting edgels however would give high support for any large randomly drawn ellipse which happens to cut through an area of high edge density. We therefore weight the contribution of each edgel by the relative

support of its owning arc. I.e. the contribution of each edgel of an arc a is weighted by a factor s_a/n_a, where s_a is the number of supporting edgels on a and n_a the total number of edgels on a. This factor de-weights edgels coming from poor arcs. I.e. if an arc intersects an ellipse more or less accidentally as in Figure 5.35(a), rather than supports it (Figure 5.35(b)) its contributing edgels will be weighted low. Whereas the same number of pixels coming from (shorter) supporting arcs (Figure 5.35(c)) will get weights close to 1.

(a) (b) (c)

Figure 5.35: Accidental (a) and supporting arcs (b, c).

The top row of Figure 5.36 shows the 10 best ellipses in each image according to absolute support, while in the bottom row the masked ellipses are removed. We see that this criterion obviously favours large ellipses. While it nicely picks the two wheels in the bike scene, it is less successful in the other scenes, e.g. the office and car scene. Any long curved edge can give rise to a "hallucinated" ellipse.

Figure 5.36: 10 best ellipses according to absolute support without (top) and with masking (bottom).

We note that a major part of these typically very elongated ellipses has no support at all. This leads to the notion of relative support.

Weighted Relative Support Weighted relative support is defined as weighted absolute support divided by ellipse circumference. I.e. the more complete an ellipse the higher its rank.

Figure 5.37 shows again the 10 best ellipses in each image according to relative support (note that obviously no masking takes place here). We see that many small speckles which happen to be circular or elliptic (like the leaves in the bike image or the texture on the kitchen table) will get a high ranking, as they form small ellipses with a few pixels but almost 100% support. Large, partly occluded ellipses or "difficult" fragmented ellipses e.g. on the rims of glasses will only get weak support. For example the rear wheel of the bike only gets rank 41 of 951 with a relative support of 86%. And the cooking spoon in the kitchen scene ranks at place 286 of 595 with a rather low relative support of 46% as a result of partial occlusion.

Figure 5.37: 10 best ellipses according to relative support.

Mean Fit Error Another criterion which does not take into account the absolute or relative number of supporting edgels is the mean fit error $\bar{d} = \frac{1}{N}\sum_{i=1}^{N} d_i$, where d_i is the geometric distance of edgel i to the ellipse.

Figure 5.38: 10 best ellipses according to mean fit error.

Figure 5.38 shows the 10 best ellipses in each image according to mean fit error. We see that the results are very similar to Figure 5.37. Tiny ellipses fitted to a small number of edgels typically have a lower fit error than larger "real" ellipses. For example the rear wheel of the bike is ranked as number 138 of 951 with a mean fit error of 0.240, where the smallest fit error in the image is 0.085 And the rear wheel of the car is number 102 of 529 with a mean fit error of 0.269, where the smallest fit error is of 0.095.

Significance The next criterion aims to combine absolute and relative number of supporting edgels and is given by the significance σ of ellipse support. We are

interested in the significance of the event that at least k edgels lie on an ellipse of circumference l, which is given (see Section 4.4):

$$Pr(X \geq k) = 1 - B_{l,p}(k-1) \quad (5.17)$$

$$\sigma = -\log Pr(X \geq k) \quad (5.18)$$

where $p = \frac{E}{N}$ is the probability of a pixel being an edgel, E is the number of edgels and N the total number of pixels in the image.

Figure 5.39: 10 best ellipses according to significance without (top) and with masking (bottom).

Figure 5.39 again shows the 10 best ellipses in each image according to significance. We see that we do not get bogged down by the small (but well supported) ellipses (such as relative support and fit error) and successfully pick significantly large ellipses, while not hallucinating arbitrarily large ellipses (such as absolute support).

The reason can be seen in table 5.2. Support $k = 10$ for a circumference (length) of $l = 20$ is a better relative (though same absolute) support than for $l = 40$; accordingly the former significance is higher. And while $k = 10$, $l = 20$ and $k = 20$, $l = 40$ correspond to the same relative support, the latter has higher absolute support (corresponds to a bigger ellipse), and accordingly gets a higher significance.

5.8 Discussion

It shows that the most important aspect of the ellipse detection approach presented in this chapter is the perceptual grouping per se, i.e. that arcs are grouped into convex groups. The exact segmentation of arcs, whether the grouping is done

k	l	abs.sup.	rel.sup	σ
10	20	10	50%	21.343
10	40	10	25%	12.745
20	40	20	50%	38.238

Table 5.2: Different significances σ for k pixels supporting an ellipse of length l.

exhaustively or greedily and the type of heuristics for greedy search do not make a very big difference. There is always a runtime/TPR trade-off.

The evaluations of convexity criteria, search strategies and image sizes show basically:

- Look harder and You will see more.
- Look closer and You will see more.
- Look longer and You will see more.

In the next chapter we will see how we can collapse this all into one parameter: runtime.

Chapter 6

Incremental Grouping of Lines into Convex Polygons

Many man-made objects or parts of objects can be described by convex contours. Moreover convexity is preserved under perspective transformation. This makes detection of convex contours an important issue for computer vision. In simple cases with uniformly coloured object surfaces and uniform background, an edge segmenter might already find complete contours of surfaces and we are done. Most often however contours are incomplete due to locally poor contrast, textured surfaces, cluttered background or simply partial occlusions by other objects. Moreover, depending on the local contrast the edge segmenter might "follow" the wrong edge, as it works on a strictly local level and has no notion of contours or surfaces. It is therefore common first to break edges into "meaningful" segments at curvature maxima, and represent these segments by some parametric model such as a line, circular or elliptic arc. Then these segments, which are parts of contours, are grouped into complete contours. This grouping will follow Gestalt principles such as proximity and good continuation.

We will start with reviewing related work in Section 6.1 and give an overview of the proposed method in Section 6.2. Section 6.3 introduces the types of junctions we are going to use and the calculation of their significances. Section 6.4 introduces Image Space Indexing as an efficient way to identify junctions. Section 6.5 highlights the problems caused by setting fixed thresholds and Section 6.6 shows how those thresholds can be avoided using incremental, anytime processing. In Section 6.7 finally closed convex contours are found via closed path search on a graph defined by lines and junctions. Section 6.8 shows results for amodal completion and how attentional mechanisms can be incorporated into the proposed approach. Section 6.9 finally concludes with a discussion.

6.1 State of the Art

Detection of closed convex contours is a well-studied problem in computer vision and perceptual organisation. Jacobs [Jac95] recognises perceptual grouping as an important step for figure-ground segmentation and recognition and presents a system for robust and efficient detection of convex groups of lines. He also shows its applicability in an indexing based 3D object recognition system. The approach can deal with moderate amounts of clutter and occlusion. It depends however on appropriate settings of various thresholds in order to cope with the $O(n^2 \log n + mn)$ runtime complexity, where n is the number of lines and m the number of groups to detect.

Huttenlocher and Wayner [HW91] present a system for finding convex line groupings which avoids thresholds in defining neighbourhood relations. Given n segmented lines they perform a constrained Delaunay triangulation in time $O(n \log n)$, where original image lines are required to be part of the triangulation. Thus it is ensured that two end points which are separated by an image line cannot be neighbours. The (non-image) lines of the triangulation then define scale-invariant neighbourhoods between line endpoints, which are used to build a planar graph, the local neighbourhood graph. Lines form nodes and neighbourhoods form graph edges, typically several per node, where edges are divided into two groups, those linking the start or end point of a line. To capture the notion of convexity, cost functions over pairs of lines are introduced. They assign an infinite value for counter-clockwise turning angles and a value measuring closeness or straightness for clockwise angles (or the other way round when intending to find counter-clockwise closures). The local convexity graph (LCG) is then constructed from the local neighbourhood graph by taking only the least cost edges. A valid convex path through the LCG is defined by an alternating sequence of edges, i.e. when entering a node through a start edge it must leave via an end edge and vice versa. When such a valid path closes back on itself, it forms a convex polygon. Choosing a single best locally convex edge for each image end point restricts the number of possible convex polygonal chains and thus runtime to $O(n)$. However this does come at a cost. Selecting only the best neighbour constitutes an early, purely local pruning of alternative hypotheses. I.e. clutter can easily "steal" the neighbourhood from the actual (desired) neighbour. Moreover the triangulation defining the neighbourhoods explicitly disallows neighbourhoods across image lines, so gaps produced by partial occlusions can not be bridged (amodal completion). Furthermore a couple of thresholds need to be introduced to cope with quantisation noise. A further angular threshold is used to define nearly collinear relations, which satisfy clockwise as well as counter-clockwise cost functions. Overall the authors successfully attacked the problem of runtime-complexity and overly depending on thresholds.

Early work on perceptual grouping by Lowe [Low87] already addressed the problem of quadratic run-time complexity (in the number of features) using a grid

overlaid on the image indexed by line endpoints. A typical problem of indexing is the appropriate choice of bin size. Sarkar and Boyer [SB94] use further curve parameters to construct index spaces of higher-parametric models and also addressed the problem of bin size and indices close to bin boundaries. Ackermann et al. [AMP+97] use areas of perceptual attentiveness in the earlier stages of their MRF-based grouping approach, which they learn from hand-labelled training sets. These areas however still serve as hard thresholds, no grouping candidates outside an attentive area are considered, which still leaves the system brittle. A small increase in the gap size between two line endpoints will prevent the formation of a grouping hypothesis. Guy and Medioni [GM96] *avoid* thresholds and use an infinitely large extension field. Edgels and segment endpoints vote with directional vectors weighted by the length of segments and decreasing with distance. Runtime complexity however is $O(k^2)$, where k is the number of edgels. Casadei and Mitter [CM98] present an elaborate system for contour estimation and introduce emission points at curvature maxima and endpoints. Search lines for finding good contour continuations are "emitted" from these points. Their system however too has problems with thresholds. Powerful region-based segmentation schemes such as work by Malik et al. [MBLS01] avoid contour grouping by always finding closed contours in the first place. Naturally however these systems can not deal with amodal completion of contours.

Even if runtime complexity is addressed successfully, most systems process the input image in one pass and limit the number of generated hypotheses by setting some saliency threshold. An early system by Sha'ashua and Ullman [SU88, US88] finds image curves that will optimise a cost function based on total curvature and number and size of gaps in a curve. They use an iterative dynamic programming approach in a network of kn^2 nodes, where k is the number of edge orientations and $n \times n$ the image size. They show impressive results on sometimes very cluttered natural images. Furthermore their approach has a nice anytime quality: processing can be stopped after any number N of steps (having performed $O(Nkn^2)$ computations) to find the smoothest curves of length N. Each pixel at each orientation then is assigned the saliency value of the length N curve starting from it. Even though N might be much smaller than the number of pixels in a contour, the saliency measure of length N curves often already picks out good salient contours. The system does not require parameters apart from the choice of discretisation k of angles. Although the optimisation is global (for the curve) decisions are rather local. In order to find e.g. the most salient curve in the image, one starts with the most salient node and follows along its most salient neighbour and so on. The system does not handle multiple hypotheses in cases where it might not be clear which neighbour to follow, but always (locally) chooses the best. And, depending on the structures in the scene, it is not always the smoothest edges which corresponds to an object contour.

Jacot-Descombes and Pun [JDP97] present a system for asynchronous percep-

tual grouping of edges. They start with Canny-segmented edges on which they perform smoothing at different scales. Edges are broken up into smooth *intervals* using a (scale-invariant) threshold. A line or circular arc is then fitted to each interval. These intervals with their extremities (= end points) form the basis for subsequent steps. Saliency of intervals is calculated based on smoothness and length, the relative contributions of which are weighted with an arbitrary weighting factor (note that Sha'ashua and Ullman use a parameter-free saliency measure expressed as a cost function). The intervals are then ordered by their saliencies and the resulting ordered list called a data flow, and traversed in order of decreasing saliency. This is the asynchronous aspect of their approach. The current interval is checked whether it is neighbour to a preceding interval. If yes, it is checked whether it fulfils a set of geometrical relations with the grouping of its neighbour. If it does, it is added to the group, else it is discarded. If the current interval does not fall into the neighbourhood of a preceding interval, it forms a new group. The (very elaborately described) geometrical relationships are collinearity, curvilinearity and "local" parallelity (claiming that checking for "global" parallelity would be too costly - where one would actually expect the asynchronicity to handle just that problem). The neighbourhood as well as the acceptance of geometrical relations are based on various ad-hoc parameters and "decision functions". Accepted geometrical relations form links (between two intervals), one or more links then form *junctions* (between any number of links). A further ad-hoc measure (with accompanying threshold) finally captures perceptual relevance of junctions. Junctions (as nodes) and intervals (as edges) then form a *rigid graph*: edges are ordered according to the orientation when leaving a mode. The rigid graph thus captures morphology as well as topology, which allows subsequent search for minimal cycles to be linear in the number of nodes. This search handles convex and non-convex cycles and also finds the "hull" of a set of connected minimal cycles. Cycles are then labelled according to an alphabet of known structures (such as quadrilateral, parallelogram, half-square, cone etc.), which is indexed by interval curvatures. This last step requires quadratic runtime, while everything so far was linear. Note that the cycle search only happens after the asynchronous grouping process, i.e. after the complete graph has been built. So their system is not actually anytime. Although the rigid graph formulation and cycle search are elegant, the heavy dependence on thresholds throughout the system makes the approach very brittle, as shown by the authors themselves when they set a different neighbourhood threshold for each test image. The authors claim their system to depend on only two parameters when in fact it depends on no less than 23, not including Canny parameters and filter widths for smoothing. Overall, the authors do not make use of the potential power of anytime-ness, the purpose of which should be foremost to avoid the need for thresholds in the first place.

Mahamud et al. [MWTX03] present an interesting system for segmenting salient

smooth closed contours. Contrary to most other approaches their saliency measure explicitly takes into account the global property of closure of contours. Rather than using only local geometric properties between a pair of edges their saliency measure gives the probability that a closed contour joins a pair of edges. Proximity and good continuation are modelled as a distribution of smooth curves traced by particles emanating from edge endpoints and moving with constant speed undergoing Brownian motion. The transition probability between edge i and j is the sum of the probabilities of all paths that a particle can take between the two edges, and is denoted as P_{ij}. The saliency measure then is based on the largest positive real eigenvalue of the transition matrix P. The search for closed contours finally corresponds to finding strongly connected components in the edge graph and is closely related to this specific formulation of saliency. Contours are segmented successively in order of decreasing saliency. The more salient a contour and the smaller the number of edges it contains, the faster it can be segmented. Segmentation can be stopped after any number of detected contours, so the approach shows anytime-ness. In the presented experiments segmentation was stopped after a fixed number of contours has been found. The authors present good results especially for highly fragmented contours and show that the globality of the saliency measure is crucial for performance. The elegance of this global property however comes at a cost, as it requires finding the largest positive real eigenvector of a $2N \times 2N$ matrix, with N the number of edges. The authors propose a special technique exploiting the sparseness and symmetry properties of the transition matrix, which significantly speeds up computation. The whole approach however is still computation intensive and generally does not seem to scale well to bigger problems. Average time for segmenting a single contour in a 480×480 image was 10 seconds on a SGI R10000 workstation. Furthermore the method is specifically designed for smooth contours (such as fruits, stones, coins) and does not work for contours containing sharp turns.

6.2 Overview

In Chapter 5 we formed convex groups of circular arcs as a precursor to ellipse fitting. Using Image Space Indexing we could achieve runtimes linear in the number of arcs. The length of search lines was however fixed, again introducing problematic thresholds.

We are now going to look at the more general case of grouping straight lines into convex polygons. Figure 6.1 shows an overview of our approach consisting of the following steps:

Detecting edges As with the ellipse case, the first step is edge detection. We use the same edge detector here.

Detecting lines Edges are then segmented into straight lines using the method

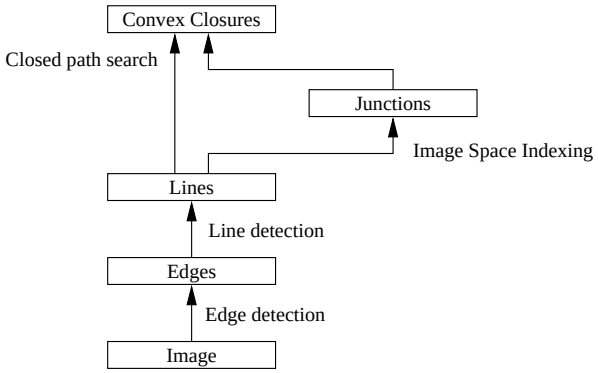


Figure 6.1: Overview of detection of convex polygons.

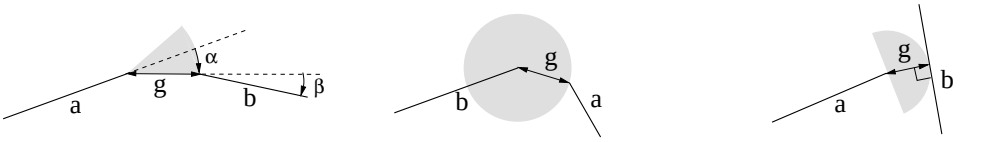
by Rosin and West [RW95].

Forming junctions (Sections 6.3 to 6.6) Junctions between lines are identified efficiently using Incremental Image Space Indexing and a graph $G = (V, E)$ is formed, with lines as vertices (nodes) V and junctions between lines as edges E of the graph.

Finding closed convex polygons (Section 6.7) Searching for closed paths on G while keeping an eye on the bending direction of the path finally yields closed convex polygons.

6.3 Junctions

We use three basic junction types: Collinearities, L-junctions and T-junctions. Note that we only consider junctions of two lines as opposed to junction labelling schemes like [Huf71]. Each higher order junction (Y, X, K etc.) can always be constructed as a series of simpler junctions. Moreover we are basically interested in finding contours of single surfaces. Each line segment of a contour can (at each end) only meet a single other line segment of that same contour. Junctions with more than two lines would thus span several contours. The T-junction is of course an exception, as it is generated by one surface occluding another.



(a) Collinearity *Cab* (b) L-Junction *Lab* (c) T-Junction *Tab*

Figure 6.2: Junction types.

6.3.1 Collinearities

Collinearities are formed by lines which are good continuations of each other (see Figure 6.2(a)). We denote a collinearity between lines a and b, where a is the longer line, as Cab. Situations like this arise if edge extraction fails locally due to poor contrast or parts of an edge are occluded. The closer and smoother the collinearity, the more significant. This corresponds to a small gap g, and small angles α and β. We are now going to define the significance of a collinearity based on its non-accidentalness (see Chapter 4) and therefore ask: What is the probability of finding a collinear configuration of two lines by accident? I.e. what is the probability of finding at least one endpoint of another line within the wedge-shaped area defined by opening angle α and gap g and with line orientation within $\pm\beta$?

Assuming that line endpoints and orientations are independent and evenly distributed we can model their distribution by a Poisson process. In Section 4.2 we introduced the Poisson process using an example of a two-dimensional parameter space and were interested in points lying within an area A. Now we have a higher-dimensional parameter space and more generally look at points within a Volume V. Endpoints are represented as points in a position/orientation parameter space of size $w \times h \times 2\pi$, where w and h are image width and height. For n lines there are $2n$ endpoints. Average point density in this parameter space is therefore $d = \frac{2n}{wh2\pi}$. The number of points within V is $P_{d|V|}$-distributed, with $|V|$ the size of V. The probability of finding a collinearity is thus given by the probability of finding at least one point within V, which is the same as 1 minus the probability of finding no such point (see Eq. 4.5):

$$\begin{aligned} Pr_C &= 1 - P_{d|V|}(0) && (6.1)\\ &= 1 - e^{-p|V|} && (6.2) \end{aligned}$$

V in our case is given as the volume defined by wedge area αg^2 and angle interval 2β.

$$Pr_C = 1 - e^{-\frac{2n}{wh2\pi}\alpha g^2 2\beta} \quad (6.3)$$

This can be rewritten using separate factors for proximity and orientation as:

$$\begin{aligned} Pr_C &= 1 - e^{-2n f_{prox} f_{ori}} && (6.4)\\ f_{prox} &= \frac{\alpha g^2}{wh} && (6.5)\\ f_{ori} &= \frac{\beta}{\pi} && (6.6) \end{aligned}$$

Following the definition in Section 4.5 the significance of a collinearity is then given as

$$\sigma_C = -\log(1 - e^{-2n f_{prox} f_{ori}}) \quad (6.7)$$

We can see that significance increases as the exponent of e decreases. So a collinearity becomes more significant with fewer lines and smaller factors. Note that each of the above factors $f_{\_} \in [0, 1]$. So also addition of more factors increases significance. This corresponds to the intuitive notion that the more specific we are about a observed visual event the less likely it will be observed by accident.

6.3.2 L-Junctions

L-junctions are formed by contour discontinuities (see Figure 6.2(b)). We denote an L-junction between lines a and b, where a is the left line, as Lab. Either the edge segmentation process split an edge segment into two lines at a point of high curvature, or the segmentation process was locally broken for some reason, leading to a gap.

Significance is defined very similarly to collinearities. Again we assume that line endpoints are independent and evenly distributed in the image. This time however line orientation plays no role. The parameter space is the $w \times h$ image, point density is $d = \frac{2n}{wh}$ and $V = \pi g^2$. Accordingly

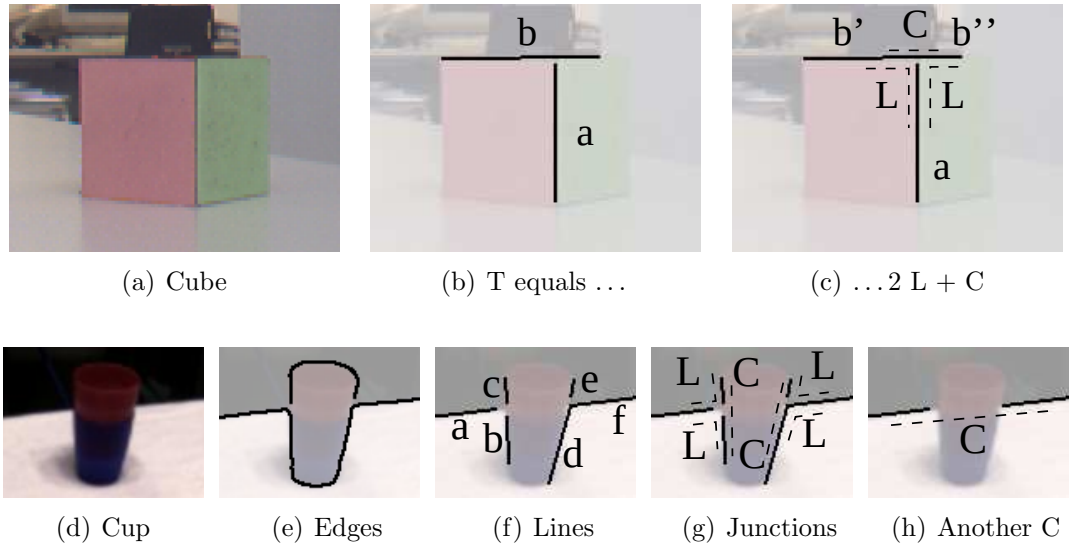
$$\begin{aligned} Pr_L &= 1 - e^{-\frac{2n}{wh}\pi g^2} && (6.8)\\ &= 1 - e^{-2nf_{prox}} && (6.9)\\ f_{prox} &= \frac{\pi g^2}{wh} && (6.10)\\ \sigma_L &= -\log(1 - e^{-2nf_{prox}}) && (6.11) \end{aligned}$$

We can see that an L-junction is less significant, as there is no orientation factor and the proximity factor is larger.

6.3.3 T-Junctions

T-junctions typically occur whenever the contour of one surface is occluded by another nearer surface (see Figure 6.2(c)). We denote a T-junction between lines a and b, where a is the upright pole and b is the horizontal bar, as Tab. To determine the significance of a T-junction we have to answer a slightly different question than in the case of the above junctions: What is the probability of finding at least one edgel of another line within the area defined by gap g? Again the parameter space is the $w \times h$ image, but now we do not need the density of line endpoints in the image but the density of edgels. We therefore replace $2n$ by N, the total number of edgels in the image. Gap g here defines a half circle as we only look for other lines"in front of" the pole (others could not form a T-junction).

Note that we do stretch the Poisson-assumption for the distribution of edgels quite a bit, as edgels are clearly not independent of each other. This dependence however would only become manifest if we asked for the probability of finding *more* than one edgel in a given area, because the actual probability would be



(a) Cube (b) T equals ... (c) ...2 L + C

(d) Cup (e) Edges (f) Lines (g) Junctions (h) Another C

Figure 6.3: Interactions between junctions.

much higher than the probability predicted by the Poisson process. As long as we are only asking for the presence of *any one* edgel of a line, we are on the safe side, as the lines themselves *can* be regarded as following a Poisson process (as we did with their endpoints above). Therefore we have

$$\begin{aligned} Pr_T &= 1 - e^{-\frac{N}{wh}\frac{\pi g^2}{2}} && (6.12)\\ &= 1 - e^{-Nf_{prox}} && (6.13)\\ f_{prox} &= \frac{\pi g^2}{2wh} && (6.14)\\ \sigma_T &= -\log(1 - e^{-Nf_{prox}}) && (6.15) \end{aligned}$$

We can see that a T-junction is again less significant than an L-junction. This follows from the fact that edgels are more common then line endpoints ($N \gg n$).

6.3.4 Interactions Between Junctions

There are certain dependencies and interactions between junctions of different types. Look for example at Figure 6.3(b). Obviously we would like to detect the two faces of the cube. This would require four lines and L-junctions to form each contour. The edge extractor however chooses to follow the "wrong" edge leading to line b. Together with a we have a T-junction which would indicate that there is a surface whose lower edge is b and which occludes contour a. This is of course a wrong interpretation and is caused by the purely local original decision of the edge extractor. To resolve such issues we simply break down each T-junction Tab into

two L-junctions Lab', $Lb''a$ and a collinearity $Cb'b''$ (see Figure 6.3(c)) and store these as alternative representations.

Figures 6.3(d) to 6.3(h) show the opposite situation. We would like to segment the contour of the cup standing at the edge of a table as well as the contour of the table surface. Figure 6.3(e) shows that again the edge extractor has locally chosen to follow the "wrong" edges (though remember that of course at the level of the edge extractor there is no right and wrong, just local contrast, the maximum of which the edge extractor follows). Figure 6.3(f) shows some of the lines segmented from the edges and Figure 6.3(g) the L-junctions and collinearities formed between them. Considering the line formed by collinearity Cbc we can now form a new T-junction $TaCbc$, and likewise with lines d, e and f. Whenever two L-junctions Lac, Lba and a collinearity Cbc meet, we form a new T-junction $TaCbc$ and again store it as an alternative representation.

Figure 6.3(h) shows yet another type of interaction. Also lines a and f form a collinearity. The gap however is rather large, making the significance of Caf quite small. However the fact that we just established T-junctions at the facing ends of lines a and f actually *explains* this gap as an occlusion rather than an accidental gap. We therefore reduce the gap size by the distance between the two T-junctions resulting in an occlusion-corrected gap and subsequently in a higher significance. In the above example we could reduce the gap size from 43 pixel to 6 pixel, thereby increasing significance of the collinearity from 4.08 to 8.03.

6.3.5 Adding Colour

Up to now the significance measures of junctions used only geometric relations. Colour can sometimes be very discriminative. Using the above formulation for significances using factors in the exponent, it is now quite convenient to add another colour factor:

$$f_{col} \quad = \quad \frac{d_{col}}{D^3} \tag{6.16}$$

where d_{col} is the colour distance e.g. between the inside colours of lines forming an L-junction. We assume again that colours are Poisson-distributed in the RGB colour space, with values ranging from 0 to $D-1$ (e.g. $D = 256$). Regarding the choice of an RGB colour space (as opposed to YUV etc.): We do not intend to mimic human colour perception (as many alternative colour spaces do). The characteristics of colour cameras are quite different from human colour perception apart from the fact that 3 colour channels are used. But these channels have different sensitivities and dynamics. E.g. many cameras are more sensitive on green because the RGGB Bayer pattern has twice as many green colour cells as red or blue ones. An "optimal" colour space would have to be adapted to a specific camera. For properly dealing with colour, cameras using Bayer filters are inadequate anyway, and high-quality 3-chip colour cameras would be needed. But as colour

is a weak cue compared to structure and we are interested in relative similarities rather than absolute colour distances, it is not worth the effort investigating further into that direction.

6.3.6 Remark on Adding Cures

Note that as a relation becomes less significant, e.g. colour distance or distance between endpoints of an L-junction increases, the corresponding factor in the exponent approaches 1, thereby not committing anything to the overall significance. So it is safe to "throw together" various cues. If a cue is not significant at all, it just does not contribute. I.e. the fact that colours in a junction do not match at all does not harm its significance, matching colours can only improve it.

Now that we have defined the junctions we are still left with actual detection of these junctions in the image. Checking all pairs of n lines for junctions has a runtime complexity of $O(n^2)$. This quadratic runtime complexity quickly becomes prohibitive for a larger number of lines. A well known technique to overcome such a combinatorial explosion is indexing.

6.4 Indexing and Voting

A common problem arising in perceptual grouping is to find relationships between pairs of primitives (or even worse triples or generally m-tuples). Exhaustively searching all pairs within n primitives has a complexity of $O(n^2)$ ($O(n^3)$, $O(n^m)$ respectively). One well known method to overcome such combinatorial explosions is indexing. Indexing serves as a heuristic to find those pairs (triples, m-tuples) of primitives which are likely to share the relationship of interest.

The indexing space is constructed such that primitives sharing a relationship index into the same bin. Thus one only has to perform the indexing operation n times to find all good candidates for pairs. As indexing is only a heuristic, all primitives of one bin are then further analysed whether they actually fulfil the given relation. If the indexing space is well constructed the number of incorrect matches is low and the complexity in all the above cases is reduced to $O(n)$.

Note that often indexing and voting are used synonymously. Authors interested in parameter estimation seem to prefer the term voting, while authors in the field of perceptual grouping prefer indexing. Technically there is no difference. In any case one wants to group a number of primitives (pixels or higher order features) into a group sharing a specific relationship (geometric parameters of a curve or higher level Gestalt principles such as proximity or parallelism). However in voting the emphasis lies in finding the parameters of the group, while one is less interested in *which* primitives make up that group. In indexing the point actually is to

(a) Original image (b) All edges

(c) Single edge segments (d) Segmented lines

Figure 6.4: Office wall scene.

identify which primitives form a group, where the parameters of the group (e.g. the orientation of parallel lines) are only a by-product.

One well known example for voting is the Hough transform [Hou62], which in its original implementation uses the (k, d) parameter space to group points (x, y) belonging to the same line $y = kx + d$. A better parameter space is obtained using the Hessian parameters $s = x \cos\theta + y \sin\theta$. Besides choosing appropriate parameters, discretisation of the parameter space is crucial to find a good trade-off between storage size (number of bins) and accuracy. While the Hough transform groups pixels into features (lines, circles, ellipses) other voting (or rather indexing according to the above distinction) methods group features into larger groups. [Low87] uses a grid-like structure, which is indexed by line endpoints. Also line orientation and length are used to form indices. [SB94] use further curve parameters to construct index spaces of higher-parametric models and also addressed the problem of bin size and indices close to bin boundaries.

6.4.1 Image Space Indexing

In Section 5.6.3 we saw how we can speed up the process of finding convex groups of arcs by the use of *Image Space Indexing*, which reduces the runtime complexity of finding "promising" pairs of arcs from $O(n^2)$ to $O(n)$, where n is the number of arcs. We are now going to have a closer, more principled look at Image Space Indexing in the context of grouping lines.

One last small remark on terminology to avoid confusion. In the following sections we refer to visible straight lines, fitted to extracted edges in the image simply as *lines*, whereas search lines drawn in the process of image space indexing are denoted explicitly as *search lines*.

Looking at a the lines extracted from a typical image such as Figure 6.4(d) it becomes evident that there is of course no need to check all pairs of lines for possible junctions, but only those being "close" to each other. While [Low87, SB94] used a grid of bins of appropriate size to capture the closeness of line endpoints, we choose a slightly different approach. In our case the index space is the image itself, i.e. single pixels serve as bins. Furthermore each line endpoint indexes a lot of specific bins, rather than only one.

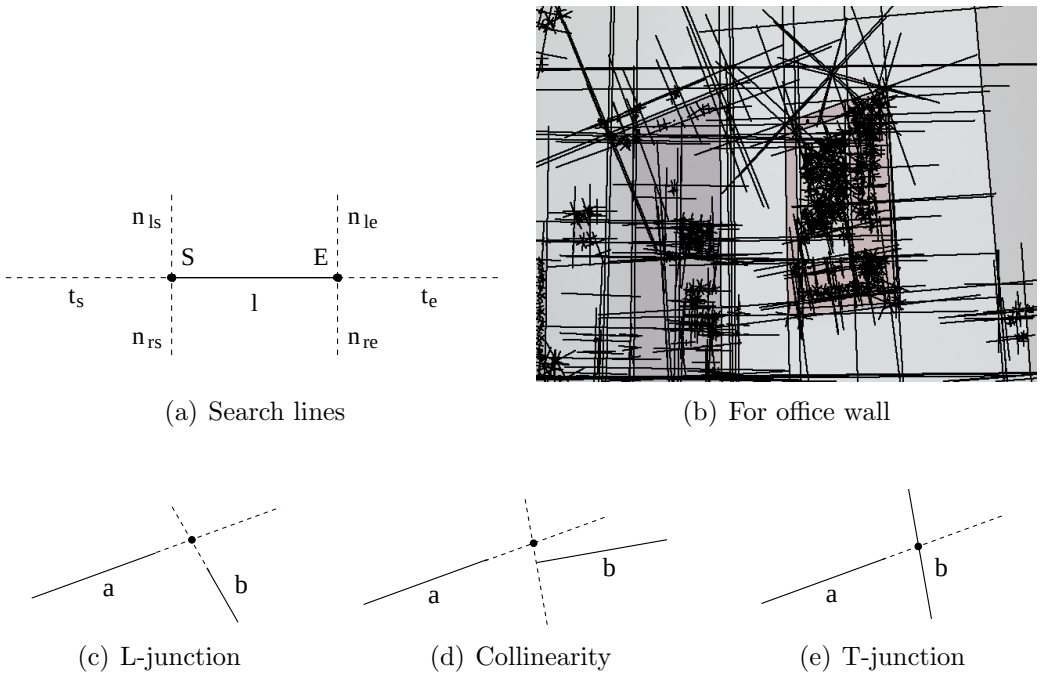
Each line endpoint defines a set of *search lines* consisting of one *tangential* and two *normal* search lines. This results in a total of seven search lines: line l itself, start and end tangent (t_s, t_e), left and right normal, again for start and end $(n_{ls}, n_{rs}, n_{le}, n_{re})$ (see Figure 6.5(a)). These search lines are drawn into the index image using the Bresenham line drawing algorithm, with a slight modification to draw "dense" lines in order not to miss any intersections. This constitutes the indexing step. Figure 6.5(b) shows the search lines drawn for the office wall scene, where search line length was chosen to be $s = l$, i.e. equal to the length of the originating lines. In this case a total of 68190 pixels were drawn for 686 lines.

Whenever two lines a and b index into the same bin, i.e. their search lines intersect, we create a new junction between lines a and b. Depending on the types of search lines intersecting we form an L-junction (Figure 6.5(c)), a collinearity (Figure 6.5(d)) or a T-junction (Figure 6.5(e)) between the respective originating lines.

6.4.2 Runtime Complexity

At first sight the above procedure seems rather wasteful and the simple elegance of indexing lost: Each one of n elements casts one vote, resulting in $O(n)$ complexity. Now each element casts many votes. Note however that still linear runtime is maintained. Assuming that search lines have an average length of s, the expected runtime is of order $O(ns)$. s is a constant limited by the image diagonal. So the runtime is of order $O(n)$. Moreover the Bresenham algorithm is of course very efficient, so per-pixel costs are small.

Note also that we are not falling victim to the curse of dimensionality. It is



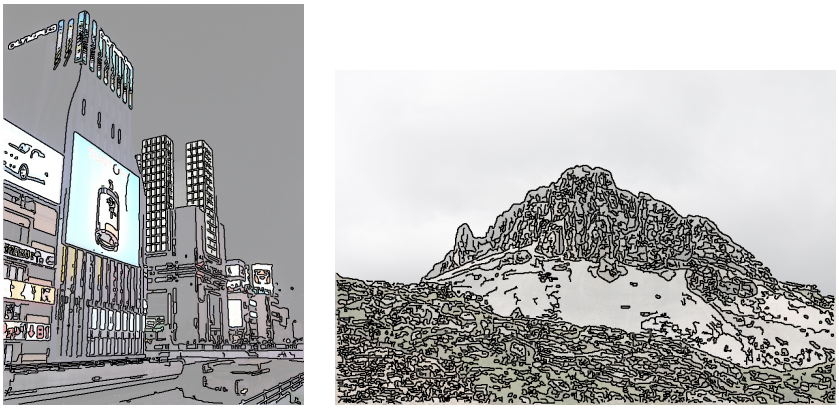
(a) Search lines (b) For office wall

(c) L-junction (d) Collinearity (e) T-junction

Figure 6.5: Search lines and junctions.

well known that high dimensional voting spaces pose a problem as the number of bins grows exponentially with the number of dimensions. Our voting space is only two-dimensional, with a rather fine binning. But as we just saw the number of votes is only growing linearly in the number of lines.

Figure 6.7 shows results for detection of junctions for two different scenes: an urban scene with rather long, straight lines (Figure 6.6(a)) and a landscape scene with many very short lines (Figure 6.6(b)). Search line length was chosen to be $s = l$, i.e. equal to the length of the originating lines. Detection was run on different resolutions of these images ranging from 384×256 to 3072×2048 pixels on a 2.16 MHz PC. The times shown are runtimes in milliseconds for junction detection only, i.e. excluding time for edge and line segmentation. The number of lines grows from a few hundred in the low resolution images up to around 100000 for the highest resolution. The linearity of runtime versus number of lines is plainly visible (Figure 6.7(a)). It reflects the linear growth of the number of junctions versus number of lines (Figure 6.7(b)). The latter indicates that each line on average forms a fixed number of junctions, which depends of course on the length of the search lines and thus the length of the lines themselves. Therefore fewer junctions are created for the mountain image, which contains many short lines, and the regression line has a smaller slope than the urban scene. The main point here is that the time needed on average to detect a single junction remains constant, irrespective of image complexity.



(a) Urban scene (b) Mountain scene

Figure 6.6: Two different types of scene.

We are now still left with the choice of a "useful" length for search lines. In the above examples we set search line length equal to length l of the respective originating line. As we will see in the next section however this typically reasonable choice can lead to problems even in typical, simple viewing conditions. We will first look at a motivating problem

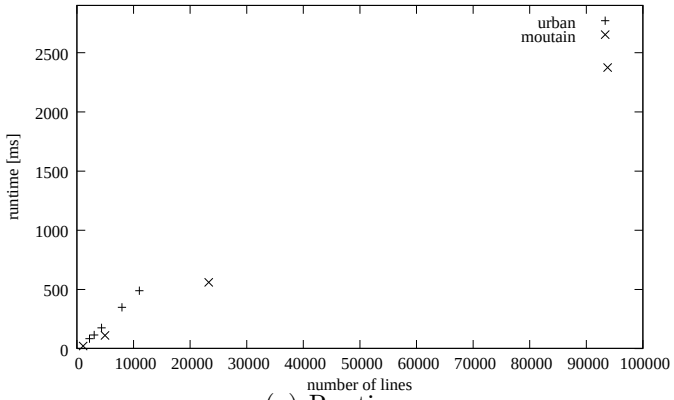
6.5 The Kanizsa Square Problem

Figure 6.8(a) shows the well known Kanizsa square, an often cited example of amodal completion. The human observer can easily perceive a white square that seems to float above four black disks. If we move the disks further apart (Figure 6.8(c)) we can still perceive a white square, albeit slightly less prominent.

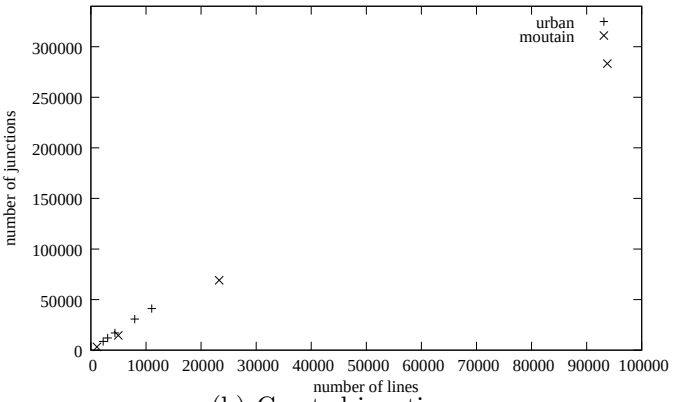
In the spirit of Image Space Indexing we can now draw search lines (the dotted lines in Figure 6.8(b)), and the intersections would create pairwise junctions which would subsequently enable use to find the closed contour which is a square. We choose the length of the search lines to be the same as the line length itself, which seems to be a reasonable "region of interest" for a line.

If we continually increase the spacing of the disks we realise that at some point suddenly the search lines are too short (see Figure 6.8(d)). Suddenly, because the increase of one pixel in spacing is enough to let the pairwise grouping and subsequent search for closures fail. This example shows yet again how brittle systems become as soon as we employ thresholds.

This also becomes readily visible in Figure 6.9(a), which we can easily perceive as a closed contour composed of line fragments. Fragmented contours like this can occur if an object or the background is highly textured, or if shadows or poor contrast lead to poor edge segmentation. Again the gaps are larger than the line fragments and thus the search lines would be too short. Figure 6.9(b) shows yet



(a) Runtime



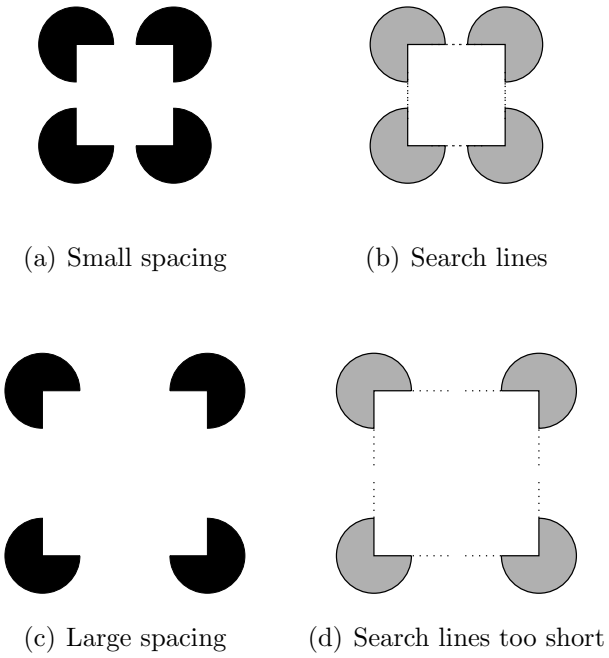
(b) Created junctions

Figure 6.7: Creating junctions for two different scenes.

another example of large gaps, this time as a result of occlusion. Certainly we would like our system to perceive the occluded rectangle as a whole, as human perceivers do.

The obvious solution is of course to make the search lines, say, twice as long. But this only postpones the same problem to a slightly later point. There will again be some spacing of lines, which is *just* too big to be bridged by the search lines. Moreover increasing search line length arbitrarily will inevitably collect more "junk" intersections (think of an actual real edge image with possibly thousands of lines), loosing the benefit of Image Space Indexing and increasing the computational costs of later processing stages which would have to reject lots of hypotheses.

So one could argue the proper search line length depends on the context or scene content. In a cluttered scene the length should be shorter to avoid an explosion of grouping hypotheses. In a sparse scene the length can be longer in order to also handle larger gaps. But what exactly would we mean by "shorter" and "longer"? Moreover also a cluttered scene could contain a contour similar to Figures 6.9(a) and 6.9(b). The point is that we should not attempt to set the search line length in the first place and instead let the length be determined automatically.



(a) Small spacing (b) Search lines

(c) Large spacing (d) Search lines too short

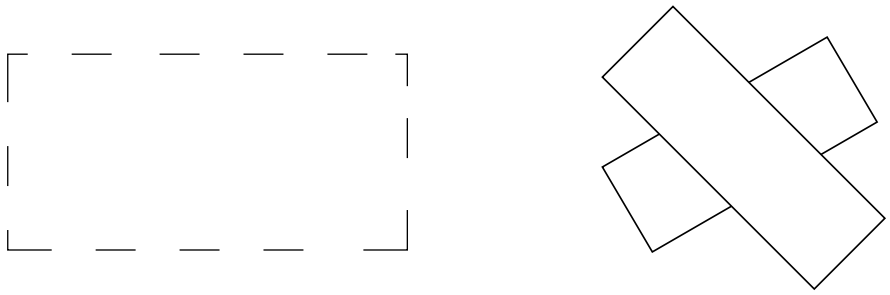
Figure 6.8: Kanizsa square problem.

6.6 Incremental Processing and Anytimeness

The solution proposed is to change the order of processing. Rather than drawing each search line to its full length (whatever that may be) for each line and then checking for intersections, we incrementally grow each search line pixel by pixel, continuously checking for junctions.

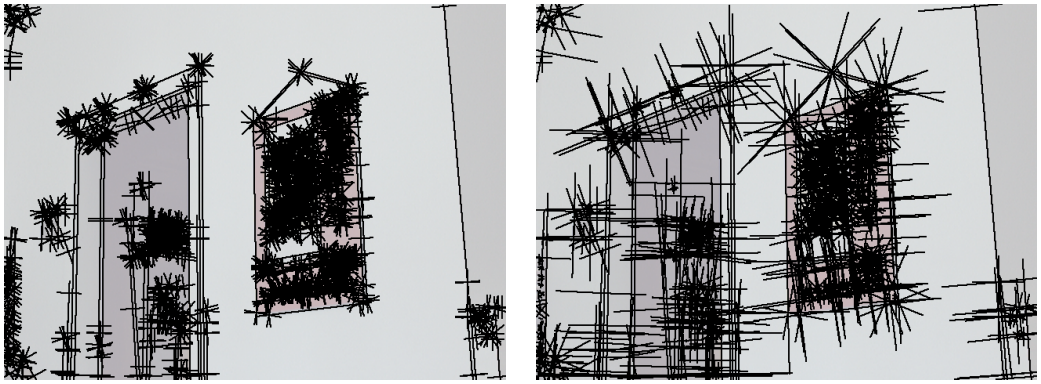
Small gaps will be traversed quickly. So the most prominent groupings will pop out early, even in cluttered scenes. Sparse scenes like the Kanizsa square figure are handled equally well. As there are not many search lines to grow, the large gaps here are also traversed quickly. Large gaps in cluttered scenes obviously pose a bigger challenge. Bridging those is more difficult amongst all the clutter, which uses up processing time as the system vainly tries to hallucinate groupings into the clutter. Eventually however even the largest gap (e.g. an occlusion as in Figure 6.9(b)) will be bridged. It may take longer, but it will not be missed.

What we have achieved now is to get rid of the troublesome parameter *search line length*. Actually we have replaced it with another parameter: *run time*. The longer we let the process of Image Space Indexing operate, the more junctions we will find. If we let indexing operate indefinitely, i.e. grow search line sup to the image border, it will eventually find all $O(n^2)$ possible junctions (most of them rather rubbish). If indexing is only given a limited amount of time, e.g. some fixed frame rate, it will only find the closest junctions.



(a) Broken contour with large gaps (b) Large gaps due to occlusion

Figure 6.9: Large gaps in typical situations.



(a) Equal growth (b) Length-weighted growth

Figure 6.10: Office wall scene with growing search lines.

Therefore we do not actually avoid combinatorial explosion, but we rather have it explode in a controlled way, always being able to interrupt the process if things get too complicated. After interruption we have as a result the best that was achievable up to now.

Incremental Image Space Indexing gives us the possibility to evaluate the best possible hypotheses given a limited amount of time. Or to put it the other way round, allows us to interrupt processing at any time while making sure that the best hypotheses have been evaluated.

6.6.1 Incremental Growing of Search Lines

Once we have adopted the incremental growing scheme, there are several strategies for controlling the growth of search lines.

6.6.2 Equal Growth

The simplest strategy is to grow all search lines equally fast. Loop over all lines and grow each of its emanating search lines by one pixel. Figure 6.10(a) shows the search lines thus created for the office wall scene. We chose to draw the same total number of pixels as in Figure 6.5(b) to enable a direct comparison. We see that short lines get too much attention and it would take rather many iterations until for example a larger gap could be bridged.

6.6.3 Length-Weighted Growth

The original, non-incremental version of search line drawing used line length to give preference to longer, thus more significant lines. We can achieve the same effect in the incremental scheme be making sure that longer lines are selected more often for growing. To this end we first rank lines according to length and then randomly select lines for growth using an exponential distribution over the rank,

$$Pr(X = x) = \lambda e^{-\lambda x} \quad (6.17)$$

$$Pr(X \leq x) = 1 - e^{-\lambda x} \quad (6.18)$$

where x is the rank of a line ranging from 0 to x_{max}. Parameter λ is chosen such that the probability mass under $[0, x_{max}]$ is 99%, i.e. we almost always select a line within the valid rank range: $\lambda = -\log 0.01/x_{max} = 4.6/x_{max}$. Once a line is selected we grow all of its emanating search lines by one pixel. Figure 6.10(b) shows the result of growing again the same number of pixels as in Figure 6.5(b) using the length-weighted growing scheme. We can see that clearly longer lines get a bigger share of the growth.

6.6.4 Boosting/Inhibition

Typically once a line has formed a junction at one end, it is not necessary to grow search lines further at that end. This is no strict rule however as sometimes the first encountered junction might not be the "correct" one leading eventually to a closed contour. As always we must be careful not to make uncorrectable local decisions. We still want to select "closed" line ends sometimes, but rather spend more time on "open" ends. So we de-weight ends with junctions.

First for each line we prefer that end with fewer junctions already formed. Probabilities of selecting the front or back end are inversely proportional to the number of junctions at the respective end: $Pr(front)/Pr(back) = n_{back}/n_{front}$, where n_{front} and n_{back} are the total numbers of junctions at the front and back end respectively. T-junctions play the opposite role of L-Junctions and collinearities. Whenever there is a T-junction at a line end, there is a good chance that this line might continue after the occlusion which caused the T-junction. So we want to favour search at ends with T-junctions. This leads to the following expressions for

n_{front} and n_{back}: $n_{front} = 1 + c_{front} + l_{front} - 2t_{front} + t_{back}$ and $n_{back} = 1 + c_{back} + l_{back} - 2t_{back} + t_{front}$. c_X, l_X and t_X are the numbers of collinearities, L-junctions and T-junctions respectively. Note that each T-junction is accompanied by two L-junctions (see Section 6.3.4). These shall not be taken into account, and are compensated by the terms $-2t_X$. The constant 1 refers to the open line end itself and avoids divisions by zero.

After randomly selecting an end we decide further if the pixel to be drawn should be awarded the line itself or, if there are junctions at this end, to one of its neighbours. Each neighbour and the line itself are chosen with equal probability. Note that we could base probabilities on junction significances, but for the time being use the simpler equal probabilities. If the line itself is chosen, one of its search lines at this end is extended (see above). Otherwise the pixel moves to the selected neighbouring line and the whole procedure is repeated. I.e. we again decide randomly whether the pixel should be awarded this new line or move along further junctions.

If at some point the pixel has travelled full-circle (i.e. we meet the original line again), we give it to a line randomly selected from all lines, where in this case we ignore rank or junctions. This selection scheme will lead to rapid closure of contours, as available partial contours "boost" growth at their end. On the other hand once closed contours are found, further growth is inhibited and "unused" pixels spread to other lines.

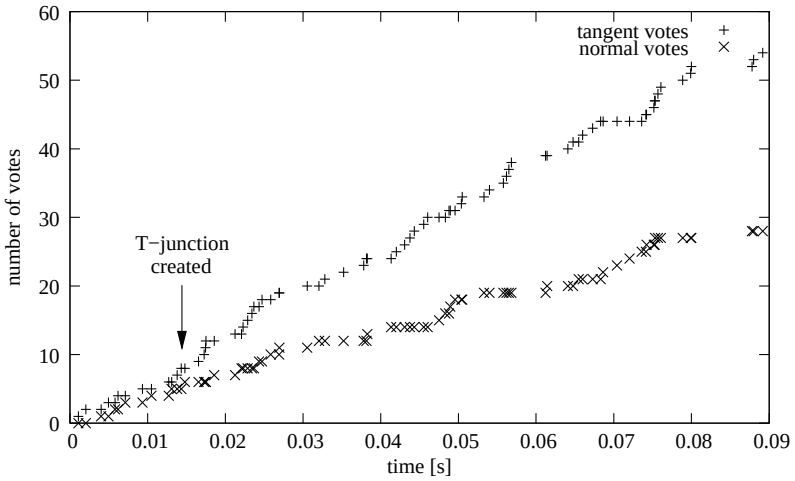
6.6.5 Tangents over Normals

Thinking about the typical grouping situations of lines we realise that areas in line direction are more interesting than areas perpendicular to it. Growing long normal search lines $n_{ls}, n_{rs}, n_{le}, n_{re}$ (see Figure 6.5(a)) is therefore slightly wasteful. So we introduce another random selection step. Once a line has been selected by the above selection, we now decide whether to grow a tangent or a normal search line. Probabilities are $Pr(tangent)/Pr(normal) = (1 + t)/1$, where t is the number of T-junctions at the respective line end. If we choose to grow a normal search line, we again randomly select whether LEFT or RIGHT.

Figure 6.11 shows the effect of T-junctions on growth rates. The line highlighted in Figure 6.11(a) forms T-junctions with other lines at some point. Figure 6.11(b) shows how the growth rate of the tangential search line at the respective line end increases after the T-junction was created. The above probabilities ensure that gaps caused by occlusions are closed faster.

Note that for this rather simple office scene we used as an example here all of the above efforts might seem slightly disproportionate. Much shorter search lines of some fixed length would certainly do. But generally we do not know the scene complexity. There might be partial occlusions requiring to bridge rather large gaps. Also we might have tens of thousands of lines rather than the few hundred in the above example (which would have made visualisation of the index image

(a) Search lines



(b) Growth rates

Figure 6.11: Effect of T-junction on search lines.

very confusing indeed). And it is in these cases where the benefits of the more elaborate growing schemes become visible. Note that earlier we remarked that we would not actually *avoid* combinatorial explosion, but rather *control* it. Basically, the more we can "steer clear" of unnecessary groupings the longer we can hold off combinatorial explosion. So generally we want to make the most sensible local decision for *most* cases, but not rule out less likely cases, as no local decision is ever guaranteed to be correct. We just "nudge" the grouping process into favourable regions.

All the above amounts to a sort of degenerate and dynamic version of extension fields [GM96] or areas of attention [SWSP00]. Our "area" is collapsed into a few lines, but these are chosen such that they capture the same relationships as would a real area. Moreover our "area" is not fixed in size but grows with time, where we (probabilistically) control its shape and growth rate.

6.7 Closed Path Search

The previous sections dealt with detection of junctions between lines and paved the way for an efficient method to construct a graph consisting of lines and junctions. We now turn our attention to this graph and the search for closed paths (cycles). Concretely we perform the following steps:

1. Connect neighbouring lines to form a graph $G = (V, E)$, with lines as vertices (nodes) V and junctions between lines as edges E of the graph.
2. Find closed paths on G, while making sure these paths constitute convex polygons.

6.7.1 Graph Construction

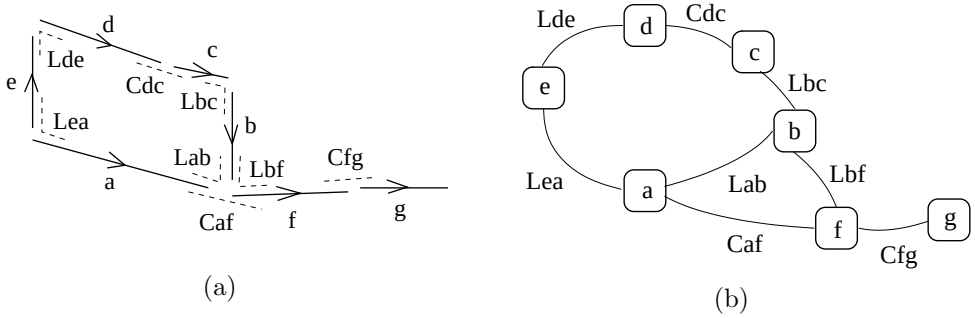


Figure 6.12: Lines and junctions (a) and the corresponding graph (b).

Figure 6.12(a) shows a simple example of several lines (the arrows indicating direction and thus start and end point) and junctions and Figure 6.12(b) shows the corresponding graph constructed from them. Nodes are formed by lines, edges are formed by L-junctions and collinearities. T-junctions do not take part in this closure search, because a closure can not have a T-junction with itself. Similar to Huttenlocher and Wayner [HW91] edges leaving a node are distinguished into several groups. They can be formed by an L-junction (e.g. Lab) or a collinearity (Caf). Furthermore L-junctions can bend to the LEFT or RIGHT, as seen from the end of a line (Lab bends to the LEFT, Lea bends to the RIGHT as seen from line a). Finally junctions connect the START or END of a line to another line (Lea and Lab respectively). This results in $2 \times 3 = 6$ groups of edges per node: START or END and one of LEFT, COLL, RIGHT.

6.7.2 Graph Search

In order to find a closed path we pick two lines which are joined at one end and search the shortest path between the two open ends. For shortest path search we

use Dijkstras algorithm [Dij59].

Remember that the graph is built incrementally. Initially we just have n nodes and 0 edges. As search lines grow and junctions between lines are created, we gradually add new edges to the graph. Whenever a new L-junction Lst or collinearity Cst is created, a new edge $s-t$ is added. This new edge could lead to the closure of a path, so we perform a shortest path search between nodes s and t. To avoid unnecessary searches, we do not start a search if a junction is isolated, i.e. one of its lines has an unconnected end. During search we only follow edges which maintain convexity of the path so far. To this end we maintain the current *bend* of the path. The first L-junction encountered on the path sets the bend to LEFT or RIGHT. Subsequent extensions of the path must be compatible with the current bend. We also enforce that a path entering via a START edge leaves via an END edge and vice versa. Finally we check whether a closed path would lead to a self-intersecting polygon, in which case we reject it. Figure 6.13 shows the complete adapted Dijkstra algorithm. `V` and `E` are the set of nodes and edges, `s` and `t` are start and terminal node. If the new edge is an L-junction, `s` and `t` are ordered such that `s` is the LEFT arm. `Q` is a heap implementing a priority queue where nodes are ordered according to their distance and `MakeHeap(Q)` is the function maintaining the heap property. `Intersecting(v)` checks whether extending the current path with node `v` would lead to a self-intersecting path. `Cost(u,v)` is a cost function and is given by the accidentalness of the junction between lines `u` and `v` (a small accidentalness corresponds to a high significance).

Note that we do not find strictly convex polygons. A collinearity is never perfectly straight. So a collinearity will sometimes actually bend somewhat "inwards" as seen from the current convex path (such as Cdc in Figure 6.12(a)). This is however a highly desired behaviour. Many "obviously" convex surfaces, such as book covers, often appear slightly concave (e.g. a paper-back with a slightly bent cover). We would not want to rule these out. This tolerance in the notion of convexity is well suited for real world situations.

6.7.3 Runtime Complexity

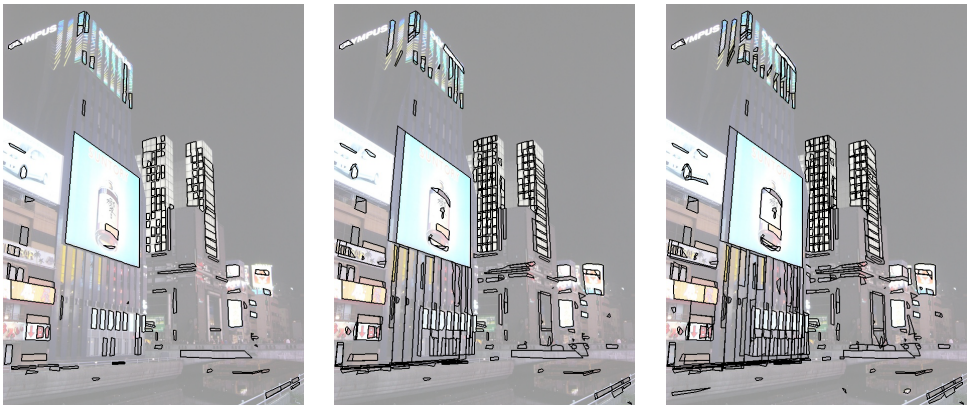
Search of one shortest path on a graph with n lines and m junctions using Dijkstras algorithm has a runtime complexity of $O((m+n)\log n)$. This is worse than the $O(n)$ complexity for finding all minimal cycles in the planar graph of Huttenlocher and Wayner [HW91] or Jacot-Descombes and Pun [JDP97]. The difference between their approach and ours is that they perform search once the graph is completed after defining all nodes and edges, while we build the graph incrementally and search for closures as we build the graph.

Figures 6.14 and 6.15 show the runtime behaviour of closure search for the urban scene of Figure 6.7. Figure 6.15(a) shows the number of detected closures over time. We see that the rate of finding new closures drops as time progresses. This is a consequence of the increasing average number of nodes visited during a

```
V .. set of graph nodes
E .. set of graph edges
s .. start node of path
t .. terminal (goal) node of path

ShortestPath(V, E, s, t)
  set dist to INF
  set prev to UNDEF
  set bend to NONE
  dist[s] = 0
  prev[s] = UNDEF
  if L-junction
    bend[s] = LEFT
  else
    bend[s] = NONE
  Q = V
  MakeHeap(Q)
  while unvisited nodes and path extendable
    u = Front(Q)
    if dist[u] != INF
      if u == t and not Intersecting(t)
        NewClosure(s, t)
      else
        for all collinearities Cuv
          if dist[u] + Cost(u,v) < dist[v] and not Intersecting(v)
            dist[v] = dist[u] + Cost(u,v)
            prev[v] = u
            bend[v] = bend[u]
        if bend[u] == NONE or bend[u] == LEFT
          for all LEFT junctions Luv
            if dist[u] + Cost(u,v) < dist[v] and not Intersecting(v)
              dist[v] = dist[u] + Cost(u,v)
              prev[v] = u
              bend[v] = LEFT
        if bend == NONE or bend == RIGHT
          ... analogously
        MakeHeap(Q)
    else
      path not extendable
```

Figure 6.13: Dijkstra algorithm to find shortest paths in graph (V, E) between start node s and terminal node t



(a) Closures after 1 s (b) ... 5 s (c) ... 10 s

Figure 6.14: Runtime behaviour for closure search.

search (Figure 6.15(b)), and therefore the increasing path lengths. These in turn mean more expensive checks for non-intersecting paths, which are of $O(k^2)$, with k the number of nodes in the path.

6.8 Results

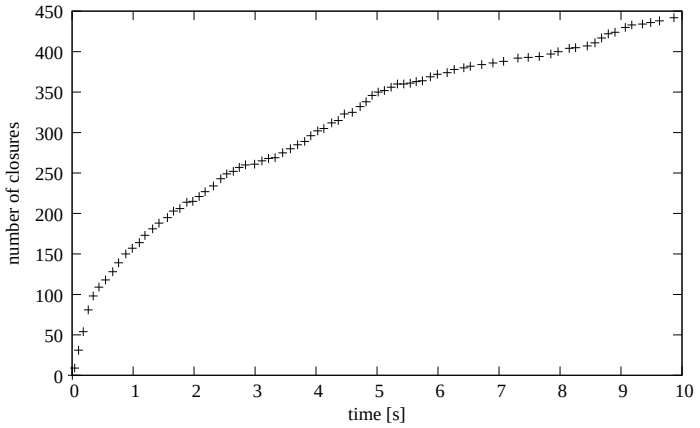
We will now evaluate the effects of different incremental indexing schemes together with the search for closed contours on a number of test images.

6.8.1 Occlusion, Amodal Completion

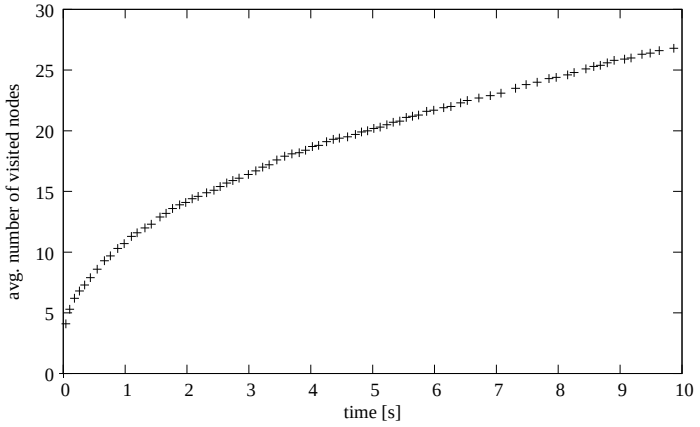
Our task is to find closed contours. This is comparatively easy as long as the contour is completely visible and we only have to bridge small gaps due to local failures of the edge detection process. However we also want to be able to handle partially occluded contours. Taking into account T-junctions when growing search lines was aimed at exactly this issue.

Figure 6.16 shows a pencil with various occlusions: none, small, large, and multiple occlusions (Note that the additional occlusions in the last case were added artificially). All images are 600×400 pixels. Tests were run on a 2.16 GHz PC. The top row shows the edge segmentation result. We can see some shadows and reflections on the table and clutter in the background. The bottom row shows the pencil contour we want to find. We are interested in the time it takes to find that contour and abort processing when it is found. Finding this closed contour requires bridging the occlusion(s) with collinearities.

Table 6.1 shows the runtimes until we find the pencil contour and how many closed contours were found until then. We evaluated equal growth (EQUAL, see Section 6.6.2), length-weighed growth (WEIGHTED, see Section 6.6.3) and what



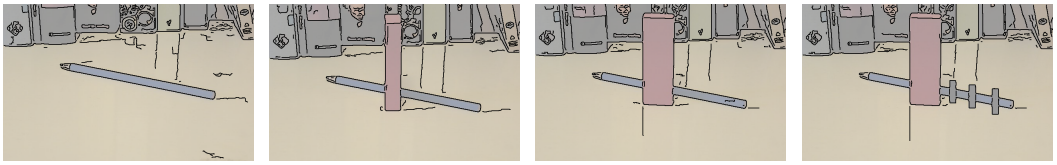
(a) Created closures



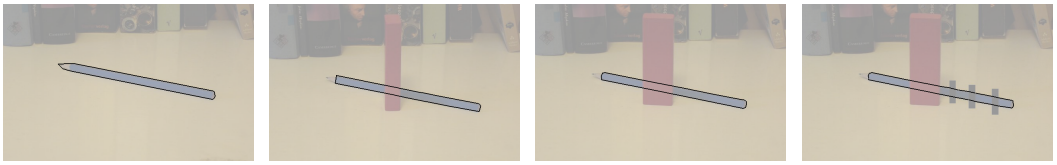
(b) Average number of visited nodes

Figure 6.15: Runtime behaviour for closure search.

we boldly term "smart" growth (SMART, the combination of Section 6.6.4 and 6.6.5). EQUAL fails to find the pencil contour even for the small occlusion in less than 10 seconds and we abort the run. The search gets completely lost in small details in the (very moderately cluttered) background. WEIGHTED and SMART have no problems with small and large occlusions, and the simpler WEIGHTED scheme actually performs better. This is no surprise as the visible parts of the contour are formed by long lines and are therefore favoured by the length-weighted selection. To see the effect of a more fragmented contour we added some more occlusions (Figure 6.16(d)). We can see that runtime for the WEIGHTED scheme is affected much stronger than SMART, which uses T-junctions at occlusions to favour tangential search lines and boosting of partially completed boundaries. Appendix B shows search lines as well as all created closures for all cases to give an indication where each scheme spends its search time and how much clutter we pick up, i.e. how much "junk" we have to accept before we get to the contour we are



(a) None (b) Small (c) Large (d) Multi



(e) (f) (g) (h)

Figure 6.16: Pencil with different amounts of occlusion: Edge image (top row) and detected pencil contour (bottom row).

Occlusion	EQUAL		WEIGHTED		SMART	
	time [s]	cnt	time [s]	cnt	time [s]	cnt
None	0.3	72	< 0.1	10	< 0.1	8
Small	> 10	(550)	0.2	61	0.5	106
Large	> 10	(560)	0.3	74	0.8	140
Multi	> 10	(500)	4.4	201	1.4	181

Table 6.1: Occluded pencil: Runtimes for finding the pencil contour and total number of contours found until that point.

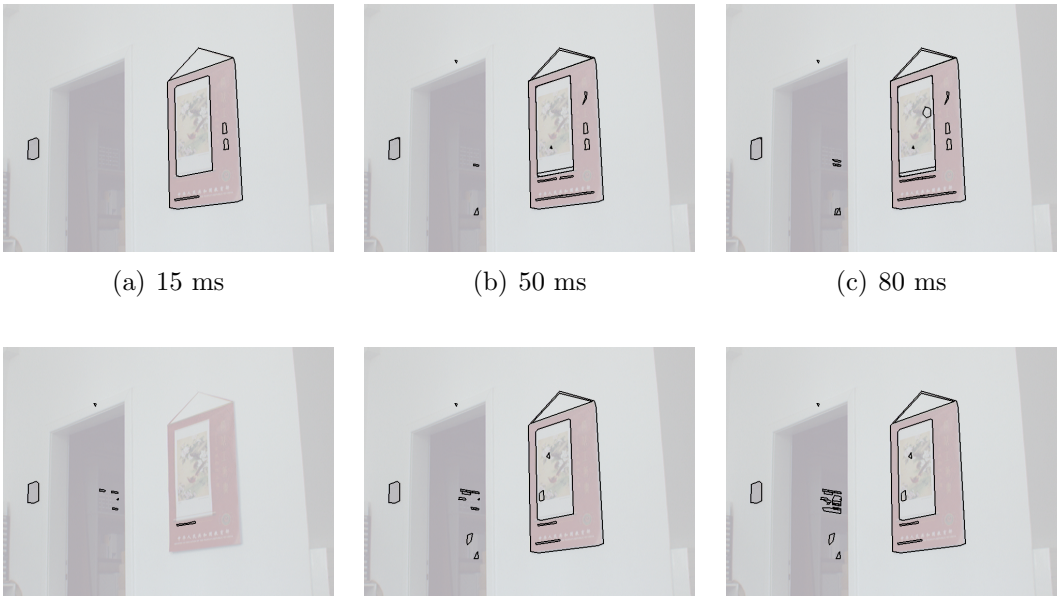
interested in.

6.8.2 Adding Attention

Up to now we based concentration of processing based on line saliency and some general considerations. Sometimes however we might have external clues. The user of our system might provide us with regions of interest (ROIs), perhaps deduced from other vision modules, e.g. saliency detection. Also higher level knowledge might be available such as the gaze direction of a human or the position of a grasping hand. The following images are all 640×480 pixels and experiments were carried out on a 2.16 GHz PC.

Regions of Interest

It is quite straight-forward to include such regions of interest into the above selection scheme. Rather than ranking lines according to their length, we rank them



(a) 15 ms (b) 50 ms (c) 80 ms

(d) 15 ms (e) 50 ms (f) 80 ms

Figure 6.17: Normal grouping (top) and grouping biased towards door as region of interest (bottom).

according to their nearness to the provided region(s) of interest.

For example we might say "look into the open door". Provided we can identify the door (perhaps the camera is mounted on a mobile robot and the robot floor-plan indicates a door) we can place a region of interest there. We choose a Gaussian weight function with peak in the centre of the doorway and standard deviation $\sigma = 150$ pixel, so that the one-σ area roughly covers the whole doorway. We then calculate the weight of a line as the sum of its pixel weights and use these line weights for ranking.

Figure 6.17 shows in the top row three images with detected closures after 15, 50 and 80 ms, using length-weighed growth. We can see that the large red poster is quickly picked up and smaller closed contours coming in later. The bottom row uses ROI-weighted growth. We can see that early on we start picking up the small boxes on the shelf in the next room, then large contours further away. After some time both methods will have found the same contours. The ROI allows us to specify where to look first and more "carefully". Note that we do not *exclude* other regions of the image, as would be the case if we simply cut out the ROI and then work on the sub-image. If there is some really prominent structure outside the ROI (such as the big red poster in the above image) we certainly still want to be able to detect it. And as always, the initial placement of the ROI might be incorrect or at least imprecise. Cutting out sub-images would mean that this error is not correctable. In our case it only means that the ROI is less efficient

in pointing out the interesting part of the image and therefore less successful in guiding processing.

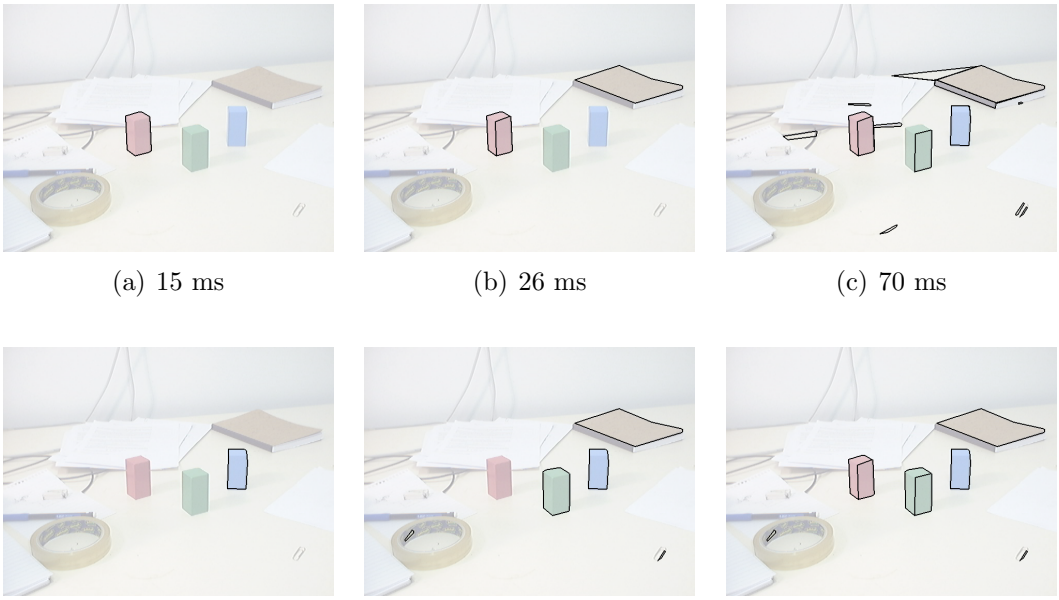
Colours of Interest

The previous example showed how we can use a region of interest to point out small details in the background which otherwise would get little attention. We can also use other available information of interest. In the scene depicted in Figure 6.18 we might be asked to "find the red block". Performing colour segmentation would be the equivalent to a cut-out ROI, with similar problems. Getting the colour model slightly wrong (e.g. due to unknown lighting conditions) will affect the segmentation and lead to uncorrectable errors. And again it is not necessarily *only* the red block which we would like our system to perceive.

Instead we use a very simple colour model: "red" is equivalent to RGB value $(255, 0, 0)$. We then weight each line with the inverse of its euclidian distance to this colour and again use these weights for ranking. (Note that there are certainly more sophisticated colour spaces and distance measures, but that is not our concern here.) The top row of Figure 6.18 shows the results of detecting closed contours for consecutive time steps. The bright red block is detected first, followed by the brownish-red book to the right and then the other blocks, which also constitute prominent contours. Compare that to the bottom row of Figure 6.18. Here we set the colour of interest to blue $(0, 0, 255)$. This time the blue block is found first, followed again by the other prominent structures.

Influencing the ranking of lines, and thus the amount of processing spent on each line, by various weights is a flexible way to provide the system with a notion of attention. We see attention not as exclusively limiting processing to a certain part of the image. We would not want the system to be blind to everything else. And of course we certainly do not want to provide a hard threshold for interesting/non-interesting. This inevitably carries the risk of missing something possibly interesting, especially if the model for attention is itself not very precise. Instead we only require a distance measure from the point of highest interest (in image space, colour space). And as we have seen neither this point of highest interest nor the distance measure need to be very precise.

Further sources of attention are conceivable. Certainly motion is known to provide attention for many sighted animals, and we could use the magnitude of an optical flow field as weight (provided of course that motion blur does not adversely affect edge detection too much). And provided with a rough notion of depth, e.g. from stereo (which by the way would carry the convenient advantage that stereo is typically good precisely at edges), we could use nearness as weight, therefore concentrating on the foreground.



(a) 15 ms (b) 26 ms (c) 70 ms

(d) 19 ms (e) 37 ms (f) 45 ms

Figure 6.18: Looking for red (top) or blue (bottom).

6.9 Discussion

The preceding chapter presented a simple idea to address issues in the detection of closed convex contours. Typically there are two ways to limit combinatorial explosion inherent in such grouping tasks: indexing and early thresholding of hypotheses. We showed how the adoption of an incremental indexing scheme removes the need for thresholds. We do not rely on any threshold or tuning parameter in the above method, but just rank grouping primitives according to saliency or external weights based on attention, and use these ranks to control incremental processing. Note that we actually do not *avoid* combinatorial explosion but rather *control* it. The more we can "steer clear" of unnecessary groupings the longer we can hold off combinatorial explosion. We are always able to interrupt the process any time if things get too complicated. After interruption we have as a result the best that was achievable up to that point. Note also that we do not prune less salient hypotheses. The longer processing runs the more awkward hypotheses will be "hallucinated", in accordance with a famous quote by Max Clowes that *vision is controlled hallucination*.

When to Stop

Having an incremental, any-time procedure for perceptual grouping, an interesting question now is when to actually stop this process. The longer it runs, the more

rather odd Gestalts are hallucinated. And obviously humans do not do that. Optical illusions are the exception, not the norm. The general answer is that grouping can be stopped as soon as sufficiently meaningful Gestalts are found, i.e. the image is sufficiently well described in terms of Gestalts. For humans we can consider this to be the case when we could form a globally consistent picture of the scene. This is still beyond the capabilities of an artificial system. A possible shortcut for an artificial system could be to stop when certain more local criteria are met or when a given specific task is solved, e.g. looking for very specific things like boxes or cylinders, or simply when time is up.

On L-Junctions and Collinearities

We have to confess that we smuggled a *threshold* into one of the above procedures. The decision whether a junction between two lines should be an L-junction or a collinearity is actually based on an angle threshold of $\pi/6$. How dare we? Looking at Figure 6.5(c) we can imagine that in some cases two tangential search lines can meet "head on", which are almost collinear, forming an L-junction where we would expect a collinearity. Likewise in Figure 6.5(d) we can imagine cases where we form a collinearity with an angle close to $\pi/2$, which should of course be an L-junction. Note that these cases are rare and in typical situations we get the junction we should expect. Nevertheless we check for those "pathological" cases. Note however that this choice has actually very little effect. In subsequent steps we treat L-junctions and collinearities basically the same. Except when we determine the bending direction in the closure search, where we disallow right turns for a left bending path and vice versa.

This leads us to the conclusion that we should abandon the distinction between L-junction and collinearity altogether and replace them with a single concept of continuation (of varying smoothness). If we have difficulties with a (early) decision, let us just not take it. What are the implications? T-junctions are no longer formed from two L-junctions and a collinearity, but from three continuations, where the straightest one constitutes the top bar of the "T". For the shortest path search instead of having three lists per line end (left L-junctions, right L-junctions, collinearities) we now only have one list of continuations, ordered according to angle. We can say which is the straightest, which is more left or right than another. So once we have e.g. a left bend in the path, we continue search along the straightest continuation and all further left of that. We thus have essentially the same behaviour as we had previously.

For the time being however we did not spend the effort of changing the program accordingly. This is left for future work. So the results presented here still have L-junctions plus collinearities, distinguished by an ad-hoc threshold.

Chapter 7

Depth Ordering of Surfaces

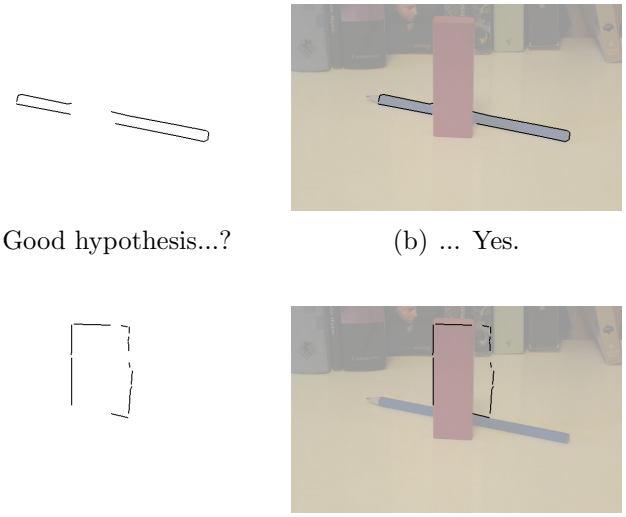
Chapter 6 showed how to find closed contours robustly without the need for tuning parameters. The result is a number of closed contours which only depends on runtime. We regard these closed contours as *surface* hypotheses. Many of these hypotheses will be contradicting, i.e. they will share parts of their contours and thus explain the same image data differently. So what remains left is to find a globally consistent grouping of edges into surfaces.

There is an abundant amount of literature on this topic, starting with Roberts [Rob65], and line/junction labelling approaches by Huffman [Huf71] and Waltz [Wal75]. Malik [MM89] propose a combination of line labelling and shape from shading. Sarkar and Boyer [SB93, SB94, SB95] propose the Perceptual Inference Network (PIN), a fixed Bayesian network independent of image data, where nodes represent known structures such as quadrilaterals and trapezoids. Schlueter et al. [SWSP00] use an MRF-based scheme, where MRF nodes represent groupings already found in the image and significances of groupings are adapted according to support and contradiction relations. But this really is an ill-posed problem. What should a globally consistent interpretation look like?

Note that we deliberately do not threshold away surface hypotheses which are too small or exceed a certain gap size vs. circumference ratio. Figure 7.1 shows two closures from the occluded pencil examples of Section 6.8. Both hypotheses have a roughly similar support (gap size(s) vs. circumference), but only Figure 7.1(b) corresponds to an actual scene structure. Locally however, by just looking at the lines making up the closures (Figures 7.1(a) and 7.1(c)) we can not decide.

Also masking of "weaker" hypotheses by "stronger", more significant ones proves to be problematic. It is difficult to capture the significance of a surface hypothesis in a single number. Moreover what would be regarded as significant is not obvious.

Figure 7.2 illustrates the problem for the case of a textured CD cover. Figure 7.2(b) shows all edges created (due to texture on the cover as well as the outline) and Figure 7.2(c) shows the closures found after some time. If we used masking and based significance on internal colour coherency, the smaller more homogeneous



(a) Good hypothesis...? (b) ... Yes.

(c) Good hypothesis...? (d) ... No.

Figure 7.1: Closure hypotheses.

texture regions would mask the outline. If we based significance on size, we would mask the texture. But clearly human observers can see both, a rectangular CD cover with texture on it.

(a) Original image (b) Edges (c) All closures (d) Contour

Figure 7.2: CD cover and detected closures.

A globally consistent interpretation is only possible once we know what *objects* we are interested in, i.e. what image content we want to communicate or act upon. This could be the CD cover outline of Figure 7.2(d) (e.g. for picking it up) or the smaller orange (top) and blue (bottom) rectangles inside (e.g. for finding the "blue-and-orange" CD cover).

So instead of forming one globally consistent interpretation of the scene, we propose to make a much weaker statement. Again we defer a (dangerous, early) decision and just order the data acquired so far. Regarding for example Figure 7.2 we would say there is one big surface, which is occluded by several smaller ones, which are "painted on", where occlusion is inferred from T-junctions. The result is a relative depth ordering of surfaces: Objects occluding each other are placed at different depths and surface markings make up different "layers" of objects.

The general rationale is that we hope to separate foreground objects from the background clutter they occlude.

7.1 Markov Random Field for Depth Ordering

Given a number of surfaces we want to find a relative depth ordering. Common contours between surfaces hint at both surfaces having same depth, T-junctions hint at one surface occluding the other (see Figure 7.3). Common contours start at a certain point, typically where two L-junctions co-incide, e.g. surfaces B and D. Sometimes these hints are contradicting, such as the common contour between surfaces C and E (starting at the indicated co-inciding L-junctions) and the T-junction between other parts of the same contours.

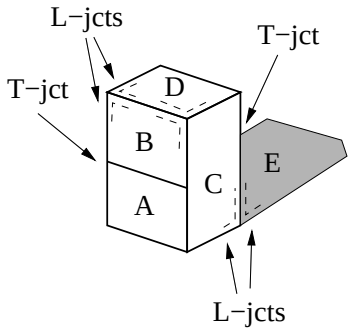


Figure 7.3: Surfaces at different relative depths.

Note that of course in real images situations are more complicated and relative depth orderings have to be consistent over the whole image. If we regard relative depth as discrete labels we can pose the search for a consistent depth labelling as a combinatorial optimisation problem. We construct a graph consisting of surfaces as nodes and depth relations (common contours and T-junctions) as edges and search for a labelling which minimises the contradictions in the depth relations.

Our problem is similar to the labelling problem in Schlueter et al. [SWSP00]. They aim to find a consistent labelling of a graph too. Their nodes are various groupings, relations are either "part of" (supporting) or "sharing the same primitive" (contradicting), and labels are significances. And they want to find *one* optimal interpretation of the scene (maximising significances). Our aim is on one hand less ambitious: we do not seek one single optimal interpretation of the scene. We consider this out of reach for now. On the other hand we want to make a stronger statement, as we infer some notion of depth from the two-dimensional image, whereas the above authors stay in 2D.

We will use the same framework as the above authors, namely a Markov Random Field (MRF) together with the Highest Confidence First (HCF) optimisation method.

7.2 Markov Random Fields

First we give a brief introduction into Markov Random Fields. See for example the book by Li [Li95] for a thorough introduction into the topic. The formalism adopted in the following sections follows that of Li.

We are given a *labelling problem*, where each site of a set $\mathcal{S} = \{s_1, \ldots, s_m\}$ is to be assigned a label from a set $\mathcal{L} = \{l_1, \ldots, l_k\}$. A *labelling* $f = \{f_1, \ldots, f_m\}$ is a set which assigns each site s_i a label f_i.

The inter-relationship between sites is represented by a *neighbourhood system* $\mathcal{N} = \{\mathcal{N}_i \,|\, \forall\, i \in \mathcal{S}\}$, where $\mathcal{N}_i$ is the set of sites neighbouring i. A *clique* c for $(\mathcal{S}, \mathcal{N})$ is a subset of $\mathcal{S}$, where each clique member is a neighbour to all other clique members. There are size-one cliques (single sites), size-two cliques, size-three cliques etc.. The set of size-one cliques is termed $\mathcal{C}_1$, the set of size-two cliques $\mathcal{C}_2$ etc. and $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2 \cup \ldots$ is the set of all cliques.

In a Markov Random Field labels are treated as random variables. $F = \{F_1, \ldots, F_m\}$ is a family of random variables defined on the set $\mathcal{S}$, where each random variable F_i takes a value f_i in $\mathcal{L}$. The joint event $f = \{f_1, \ldots, f_m\}$ where F_i takes on the value f_i is called a *configuration* of F and corresponds to a labelling. $P(F_i = f_i)$ is the probability that random variable F_i takes on value f_i, and $P(F = f)$ is the probability of the joint event. For a Markov Random Field on $\mathcal{S}$ with respect to neighbourhood system $\mathcal{N}$ the following conditions must be satisfied:

$$P(f) > 0, \quad \forall f \in \mathbb{F} \qquad \text{(positivity)} \tag{7.1}$$

$$P(f_i \,|\, f_{S-\{i\}}) = P(f_i \,|\, f_{\mathcal{N}_i}) \qquad \text{(Markovianity)} \tag{7.2}$$

where $f_{\mathcal{N}_i} = \{F_{i'} \,|\, i' \in \mathcal{N}_i\}$ denotes the set of labels at the sites neighbouring i. The Markovianity states that only neighbouring labels have direct influence on a label.

Generally it is difficult to specify the joint probability $P(f)$ or conditional probabilities $P(f_i \,|\, f_{\mathcal{N}_i})$, except for a specific class of MRFs: The random variables F of a *Gibbs random field* (GRF) on $\mathcal{S}$ with respect to $\mathcal{N}$ obey a *Gibbs* distribution:

$$P(f) = \frac{1}{Z} e^{-\frac{1}{T}U(f)} \tag{7.3}$$

$$Z = \sum_{f \in F} e^{-\frac{1}{T}U(f)} \tag{7.4}$$

Z is a normalising constant called the *partition function* and constant T is called the *temperature* (showing the Gibbs distributions origin in Physics problems). $U(f)$ is called the *energy function* and is given by the sum of clique potentials:

$$U(f) = \sum_{c \in \mathcal{C}} V_c(f) \tag{7.5}$$

$P(f)$ measures the probability of a particular configuration f. The more likely configurations are those with lower energies. By minimising the energy function we can find the most likely configuration. So as long as we are only interested in the configuration minimising the energy function and not the specific probability of this configuration, the particular values of Z and T do not matter and can be set to 1.

We will only consider cliques up to size two and can write the energy function as

$$U(f) = \sum_{i \in \mathcal{S}} V_1(f_i) + \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{N}_i} V_2(f_i, f_j) \tag{7.6}$$

7.3 Problem Formulation

We will now describe the MRF formulation of our problem. $\mathcal{S}$ is the set of surfaces. The neighbourhood system $\mathcal{N}$ is defined by common contours and T-junctions. For the time being we use a fixed number of depths, from $l_{min} = -10$ to $l_{max} = 10$, which define the label set $\mathcal{L}$ (Note that this ad-hoc setting of a fixed number of labels of course contradicts our mantra of parameter-free methods and, as is to be expected, will get us into trouble later on). We have two special labels: a default depth $l_d = 0$ and the uncommitted label l_u (see Section 7.4).

The data energy term is used to place surfaces at the default depth l_d as long as no depth relations indicate otherwise. So this energy term is only used for new, uncommitted sites.

$$V_1(f_i) = \begin{cases} |f_i - l_d| & \text{if } f_i = l_u \\ 0 & \text{else} \end{cases} \tag{7.7}$$

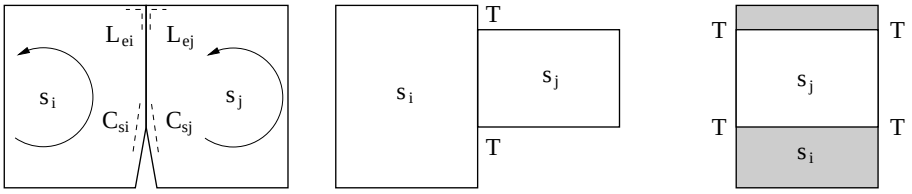
Each site s_i has two (overlapping) sets of neighbours: contour neighbours ($\mathcal{N}_C$) and T-neighbours ($\mathcal{N}_T$). Contour neighbours are surfaces which share part of their contour with s_i. Surfaces with contours forming a T-junction with the contour of s_i are T-neighbours. Accordingly we define energies for size-2 cliques: common contour energies (V_C) and T-energies (V_T). Then $V_2(f_i, f_j) = V_C(f_i, f_j) + V_T(f_i, f_j)$.

Surfaces which share a common contour should be at the same depth, i.e. have the same label:

$$V_C(f_i, f_j) = \begin{cases} \min(\sigma_{si}, \sigma_{sj}) + \min(\sigma_{ei}, \sigma_{ej}) & \text{if } f_i \neq f_j \\ 0 & \text{else} \end{cases} \tag{7.8}$$

where σ_{xy} are the significances of the junctions which start and end the common contour on s_i's and s_j's side respectively. These junctions can be L-junctions or collinearities (see Figure 7.4(a)). Start and end are defined by s_i, where junctions are counted counter-clockwise. So the energy of a common contour is defined by the minimum significances of the junctions causing it.

If the contours of s_i and s_j form a T-junction they should be at different depths, because one surface occludes the other. Let us suppose that the contour of s_i forms



(a) Common contour (b) Outside T (c) Inside T

Figure 7.4: Depth relations.

the bar and the contour of s_j forms the pole of a T-junction. Now there are two cases: the contour of s_j can intersect the contour of s_i from *outside* (see Figure 7.4(b)) or from *inside* (see Figure 7.4(c)). In the first case we consider s_i occluding s_j and demand $s_i < s_j$. In the second case we consider s_j to be "painted on" s_i and therefore occluding it, and we demand $s_i > s_j$.

$$V_T(f_i, f_j) = \begin{cases} \sigma_T & \text{if } f_i \geq f_j \text{ and } s_j \text{ outside } s_i \\ & \text{or } f_i \leq f_j \text{ and } s_j \text{ inside } s_i \\ 0 & \text{else} \end{cases} \tag{7.9}$$

where σ_T is the significance of the T-junction. Note that each T-junction corresponds to a collinearity and two L-junctions (see Figure 6.3). The common contours started by such L-junctions are not considered in $\mathcal{N}_C$.

Figure 7.5 shows a sample MRF for Figure 7.3. Solid lines represent common contour relations (i.e. same depth) and a dashed arrow from X to Y means that X is above Y. We can see several contradicting relations already in this simple example. The optimisation method presented in the next section will find the labelling minimising the global energy and thus find the least contradicting labelling.

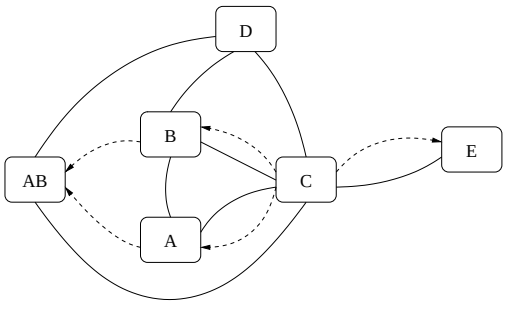


Figure 7.5: Sample MRF for Figure 7.3. Solid lines represent "same depth", dashed lines represent "above".

7.4 Highest Confidence First Energy Minimisation

Highest Confidence First (HCF) [CB90, CCS+93] is a serial, local, deterministic algorithm for combinatorial minimisation with a discrete label set. Sites are selected successively and a label is chosen which minimises the local energy E_i of that site.

$$E_i(f_i) = V_1(f_i) + \sum_{c:i\in c} V_c(f_i) \tag{7.10}$$

where $c : i \in c$ denotes all cliques of which i is part. The algorithm introduces a special *uncommitted* label $l_u \notin \mathcal{L}$. Initially all sites are uncommitted. Once a site is committed, i.e. has a label other than l_u, it can not go back to the uncommitted state. The uncommitted label is introduced in order to avoid problems with dependence on initialisation, which other algorithms such as Iterated Conditional Modes (ICM) have.

Clique potentials V_c are slightly modified

$$V_c' = \begin{cases} 0 & \text{if } \exists j \in c \; f_j = l_u \\ V_c & \text{else} \end{cases} \tag{7.11}$$

So a site has no effect on its neighbours until it is committed. Commitment and label changes are based on a *stability* measure. The stability of i with respect to f is defined as

$$S_i(f) = \begin{cases} -\min_{l\in\mathcal{L}, l\neq l_{min}}[E_i(l) - E_i(l_{min})] & \text{if } f_i = l_u \\ \min_{l\in\mathcal{L}, l\neq f_i}[E_i(l) - E_i(f_i)] & \text{otherwise} \end{cases} \tag{7.12}$$

where $l_{min} = \text{argmin}_{l\in\mathcal{L}} E_i(l)$. The stability $S_i(f)$ measures the maximum energy reduction that can be achieved by changing the label of s_i. For an uncommitted site, the stability is given by the difference between the lowest and second lowest energies. For a committed site it is given as the difference between the current energy and lowest possible energy due to any other label. A negative stability means that the local energy can still be reduced, a positive value indicates that no further improvement is possible. A higher positive value of $S_i(f)$ means a more stable configuration, the distance to the next best label is large, whereas a small positive value means that it does not matter much which label to chose. A negative value with larger magnitude gives us a higher confidence to reduce the global energy in changing f to a new configuration. Hence the name Highest Confidence First: we successively change the site with the highest confidence, i.e. the lowest stability value.

Figure 7.6 shows the complete HCF algorithm. `MakeHeap` sorts S values such that the minimum value is on top, `Top` retrieves this top value. `ChangeState(k)`

```
HCF()
  set f to UNCOMMITTED
  MakeHeap
  while S(Top) < 0
    k = Top
    ChangeState(k)
    UpdateStability(k)
    for each j in neighbourhood of k
      UpdateStability(j)
    MakeHeap
```

Figure 7.6: Highest Confidence First (HCF) algorithm.

changes f_k to f'_k

$$f'_k = \begin{cases} \text{argmin}_{l \in \mathcal{L}} E_k(l) & \text{if} f_k = l_u \\ \text{argmin}_{l \in \mathcal{L}, l \neq f_k} [E_k(l) - E_k(f_k)] & \text{else} \end{cases} \tag{7.13}$$

If there are no more negative stabilities the MRF is in a (local) minimum, because no further change of a label can reduce the global energy further.

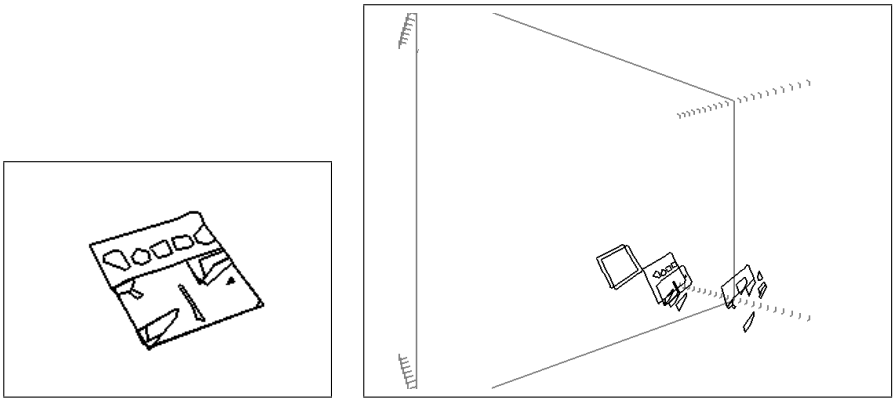
Junctions and therefore depth relations as well as closures are built incrementally (see previous chapter). So also depth ordering is performed incrementally. Sites are added to the MRF one by one as new closures are formed and a new energy minimisation started after each one. As the existing sites are already labelled consistently and only the new site is uncommitted convergence is typically fast.

7.5 Results

We tested the depth ordering on a number of table-top scenarios with varying success. All tested images are 640×480 and experiments were carried out on a 2.16 GHz PC. Runtimes are total runtimes including edge segmentation, closure detection and depth ordering. We will show the original images overlaid with detected closures and a quasi-three-dimensional view showing the ordering of surfaces in different depth layers. The "photo corners" indicate the different discrete depth layers. Darker surfaces are nearer, the lighter the grey value, the further away. Note that the spacing of depth layers is of course meaningless and is set to allow discerning the various surfaces. Unfortunately the (static) 3D views sometimes fail to adequately capture the quasi-three-dimensional organisation of the scene, especially if multiple hypotheses clutter the display. Rotating theses views interactively obviously gives a much better impression.

Figure 7.7 shows the depth ordering of closures detected for the CD cover of Figure 7.2 after 0.302 s. In this case we only show the CD cover and hide the rest

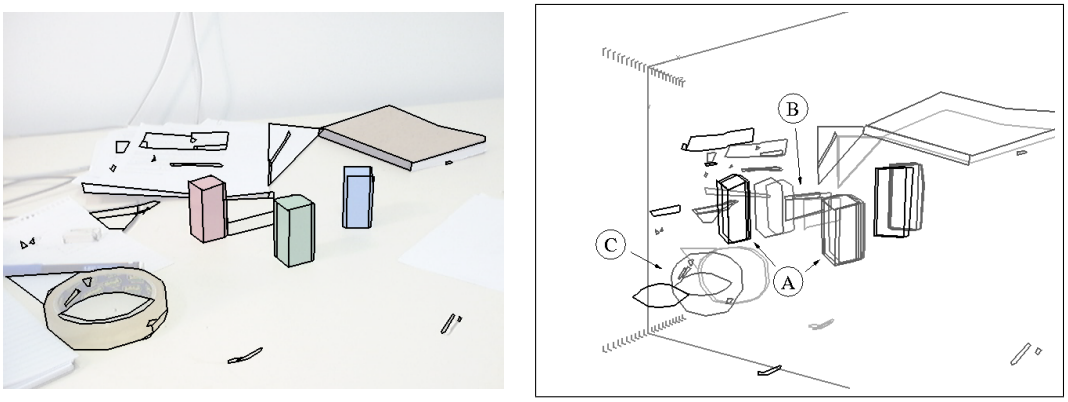
of the scene for clarity. We can see that the outline forms the farthest surface and the various surface markings are perceived as covering this surface.



(a) Detected closures. (b) Surfaces in depth.

Figure 7.7: CD cover.

Figure 7.8 shows the three coloured blocks already encountered in Figure 6.18 after 0.294 s. We see that the faces of the blocks are perceived as lying in front of the contour (A). Furthermore the shadow caused by the left (red) block is perceived behind it and also behind the centre (green) block (B). The roll of sticky tape on the left (C) does not quite make sense: our depth reasoning does not correctly perceive holes.

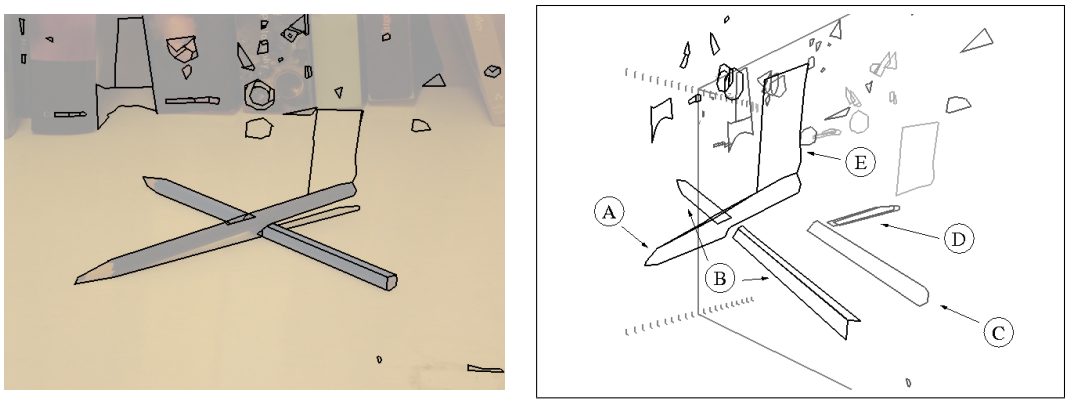


(a) Detected closures after 0.294 s. (b) Surfaces in depth.

Figure 7.8: Three coloured blocks.

Figure 7.9 shows two pencils, one occluding the other. The front pencil (A) is correctly perceived as occluding the left and right part of the back pencil (B). The right part of (B) is detected in some more detail (two faces of the pencil which is a hexagonal prism) and (C) is the coarser outline of that part and accordingly

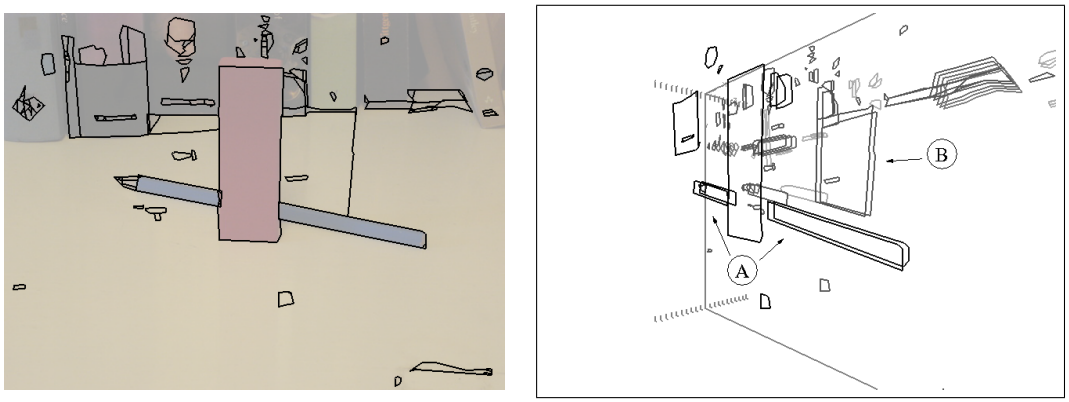
perceived below (B). The shadow (D) of the front pencil lies correctly behind both pencils. Only the reflection (E) of one of the books in the background on the table is perceived incorrectly, the reason being that while it does form a T-junction on the left side, it also (accidentally) aligns rather smoothly on the other side.



(a) Detected closures after 0.400 s (b) Surfaces in depth.

Figure 7.9: Two pencils.

Figure 7.10 shows the occluded pencil of Figure 6.16 (before amodal completion) after 0.831 s. The pencil parts are perceived as behind the block (A). Also the reflection of one of the books in the background is perceived as behind the block and pencil (B).

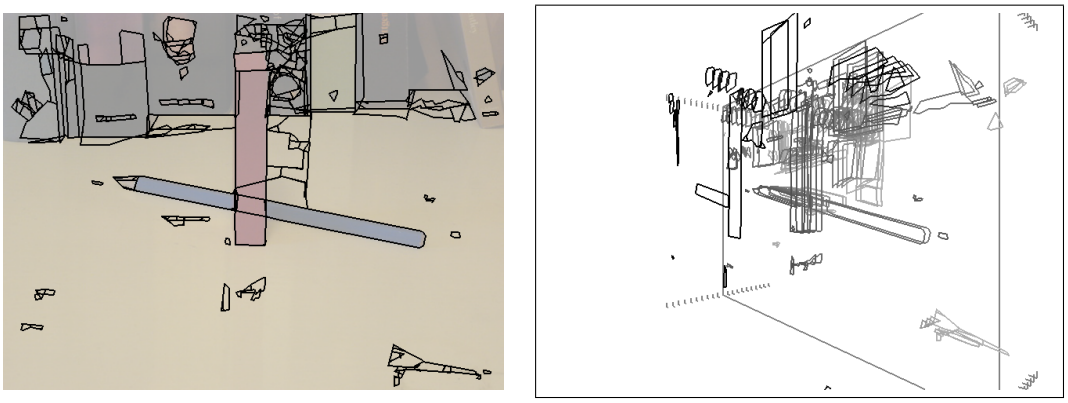


(a) Detected closures after 0.831 s. (b) Surfaces in depth.

Figure 7.10: Occluded pencil.

Figure 7.11 shows again the occluded pencil this time after the contour was amodally completed. We see that by and large the depth ordering is still correct (also the completed pencil contour is correctly placed behind the occluding block, although barely visible in all the clutter). However we needed to wait 21 s for

the occluded contour to be completed! The reason is that while waiting for the gap to close we accumulated in the order of 350 surfaces. The depth ordering now tries in vain to fit these into a consistent labelling with only the 21 (from -10 to 10) depth labels we originally defined. As long as there are few sites, consistent labelling is easy and found quickly. A new surface will quickly find a suitable label and only "move"a few neighbouring surfaces. As complexity and neighbourhoods grow, it is increasingly difficult to find a consistent labelling and surfaces tend to be "pushed" against the borders of minimum and maximum label, not able to move further although they still violate depth constraints. Increasing the number of labels would postpone this problem, but also slow down the HCF algorithm, which for each site tries all labels in order to find the one resulting in minimal energy.



(a) Detected closures after 21(!) s.

(b) Surfaces in depth.

Figure 7.11: Occluded pencil, amodally completed.

7.6 Discussion

The aim of this chapter was to put the closures detected in the previous chapter into a more global context. Furthermore to provide a purpose for perceptual grouping. Perceptual grouping is not meaningful in and on itself. (Local) perceptual grouping should eventually lead to a global interpretation of the scene, a goal which is however still elusive.

Although the proposed MRF based method manages to get correct depth ordering in simple cases, it does not quite tackle the perceptual confusion of multiple hypotheses. Lots of loose clutter can be seen "flying around". More severely, the method runs out of depth labels with increasing complexity, i.e. when run longer.

A possible solution to that could be to determine the necessary number of labels automatically. Equations 7.12 and 7.13 iterate over all labels in the label set $\mathcal{L}$ to find the optimum. One could start with a minimal label set $\mathcal{L} = \{l_d\}$

containing only the default label and in the above equations tentatively extend the label set to $\mathcal{L}' = \mathcal{L} \cup \{l_{min} - 1\} \cup \{l_{max} + 1\}$. If the optimum label would be outside the original label set $\mathcal{L}$ we permanently extend it to incorporate that label. It shows however that the number of labels then just tends to grow arbitrarily and large "gaps" of unused labels form. So we would require a "repacking" of the label set, removing unused labels and adjusting the bounds accordingly. It is not entirely clear however how all these ad-hoc modifications would affect the original minimisation algorithm, and a deeper look into the theory is needed.

Another solution could be to stop incremental processing as soon as the label set is exhausted. I.e. use the limited capability of the depth ordering as a stopping criterion: We can stop finding more structure as soon as we can no longer handle it.

Both of the above proposals are not entirely satisfactory however. All in all the proposed MRF based method seems not powerful enough to provide the sort of global reasoning we are after in all but rather simple cases. Moreover we would actually like a global reasoning system to be able to represent mutually exclusive but internally stable and consistent interpretations (such as Necker cubes), which an MRF can not do. These would correspond to strong local minima which are placed far apart, and the local HCF minimisation can not easily move between these. Other combinatorial optimisation methods such as genetic algorithms might be better suited. For example a genetic algorithm could represent mutually exclusive interpretations as different populations. Our initial choice for a MRF with HCF minimisation however was motivated by ease of implementation and the fact that it is a deterministic algorithm without the need for "black magic".

Another desireable property of a global reasoning system is dealing with cases which are actually globally inconsistent, such as a Penrose triangle or Escher pictures. Such pictures contain locally consistent parts (e.g. the corners of the Penrose triangle), which are perceived as such by human observers, which however do not fit globally. So we would like our reasoning system to keep the local consistencies and realise the global inconsistency. For the MRF approach it is not guaranteed that the configuration into which it would settle in such a case would actually preserve the local consistencies.

Right now neighbourhoods are only formed between surfaces touching each other (forming a common contour or a T-junction). Same depth could however also be inferred from similarity or alignment of surfaces (think of the windows on a building front). In fact it would be desireable to incorporate many more Gestalt principles in the global reasoning stage. This would however typically mean larger cliques. The depth relations we have now are binary relations resulting in size-two cliques. Similarity between n surfaces means a size-n clique. Unfortunately however performance of HCF degrades with larger cliques (also Schlueter et al. [SWSP00] only use size-two cliques). Given that the MRF approach as it stands now only handles fairly simple cases, it seems too early to think about such

ambitious extensions.

Chapter 8

Summary and Discussion

This work started with rather abstract, sometimes philosophical considerations concerning a possible theory of vision. We felt the need for such explorations because much work in computer vision seems to address the wrong questions. We particularly argued against approaches simply based on the recognition of a few particular image patterns and showed an example why in Chapter 3.

We reviewed past work, sometimes dating back to the early days of the field, which did address the right questions. Sometimes owing to the limited available processing power at the times, but typically due to a failure to recognise the importance of tackling complexity, much of this work was based on overly optimistic assumptions and did therefore not scale to the real world (and sadly was subsequently abandoned).

Chapter 4 introduced quantitative measures for non-accidentalness, based on probabilistic considerations. We aimed for a general formulation of the significance of a visual event. These significances are used in grouping and reasoning processes in later chapters.

Chapters 5 and 6 presented a way to tackle complexity, without resorting to thresholding it away. Tuning parameters and early pruning of hypotheses are the bane of many (otherwise ingenious) approaches. Incremental processing, from the identification of junctions, to the formation of closed contours and finally reasoning about depth provides a means to avoid parameters and also leads to the desireable property of anytimeness. We could show that scene complexity (represented by the number of lines/arcs in an image) poses no problem as Incremental Image Space Indexing allows grouping in linear runtime. With increasing complexity the system degrades gracefully. Different types of scenes were used for evaluation without the necessity of tuning any parameters. Finally we indicated how attentional mechanisms can easily be incorporated into the incremental grouping procedure.

Chapter 7 finally put the local perceptual grouping processes of earlier chapters into a more global context. Finding a globally consistent interpretation of a scene is still well beyond reach. We could however show, using a Markov Random Field based approach that a limited notion of three-dimensionality, namely a relative

depth ordering of surfaces is feasible, at least for scenes of limited complexity.

What about the sought-after theory of vision? What can be learned from the presented work? Although some choices along the way were rather ad-hoc (such as the choice of Canny edge detection, of Dijkstra shortest path search in a graph or the choice of a Markov Random Field with the Highest Confidence First algorithm to solve a combinatorial search problem) and mainly guided by computational considerations, a general lesson learned seems to be the importance of parameter-free methods and anytimeness.

Dealing with real-world (cluttered, noisy, ambiguous, locally incorrect, ...) data from the start is a necessary prerequisite for any successful approach. Assuming clean edge segmentation and "for the time being" concentrating on higher level aspects will typically lead to unrealistically optimistic systems which can not easily be adapted later to realistic images. Our own work, starting with rather high level ideas of vision and meaning of objects, and later forced to concentrate on efficient grouping methods first, reflects this lesson.

Although we talked a lot about Gestalt principles generally, we only implemented relatively few, namely those leading to surfaces, i.e. good continuation, similarity (of colour) and closure. We are looking at closures because they typically correspond to surfaces (or surface patches) of physical objects. And it is these that we are ultimately interested in. An extension to incorporate more Gestalt principles is tempting. However we need a more powerful integration scheme than the Markov Random Field presented in Chapter 7 first.

The hope of finding a general clean probabilistic framework for perceptual grouping was only partially fulfilled. We managed to use a consistent notion of significance based on non-accidentalness and probability distributions and used these probabilities in the clique potentials of a Markov Random Field. It is not clear however how this choice of significance measure makes a significant (pun intended) difference versus more ad-hoc saliency measures. Moreover our Markov Random Field approach does not seem to be powerful enough to provide the sort of global consistency reasoning we are after and we do not go so far as to claim it to be a general formulation of perceptual grouping.

Finally, the main open (and challenging) question remains how to establish a link to the notion of *object*. As we pointed out in the introduction the vision system itself should not be concerned with objects and the notion of object only emerges at the intersection of different modalities, where a specific intention of the user of these modalities links certain percepts from different modalities to achieve a certain goal (such as grasping). The different modalities therefore must be able to link up, and it remains to be seen how perceptual groups represent such a link.

Appendix A

Data Set for Ellipse Detection

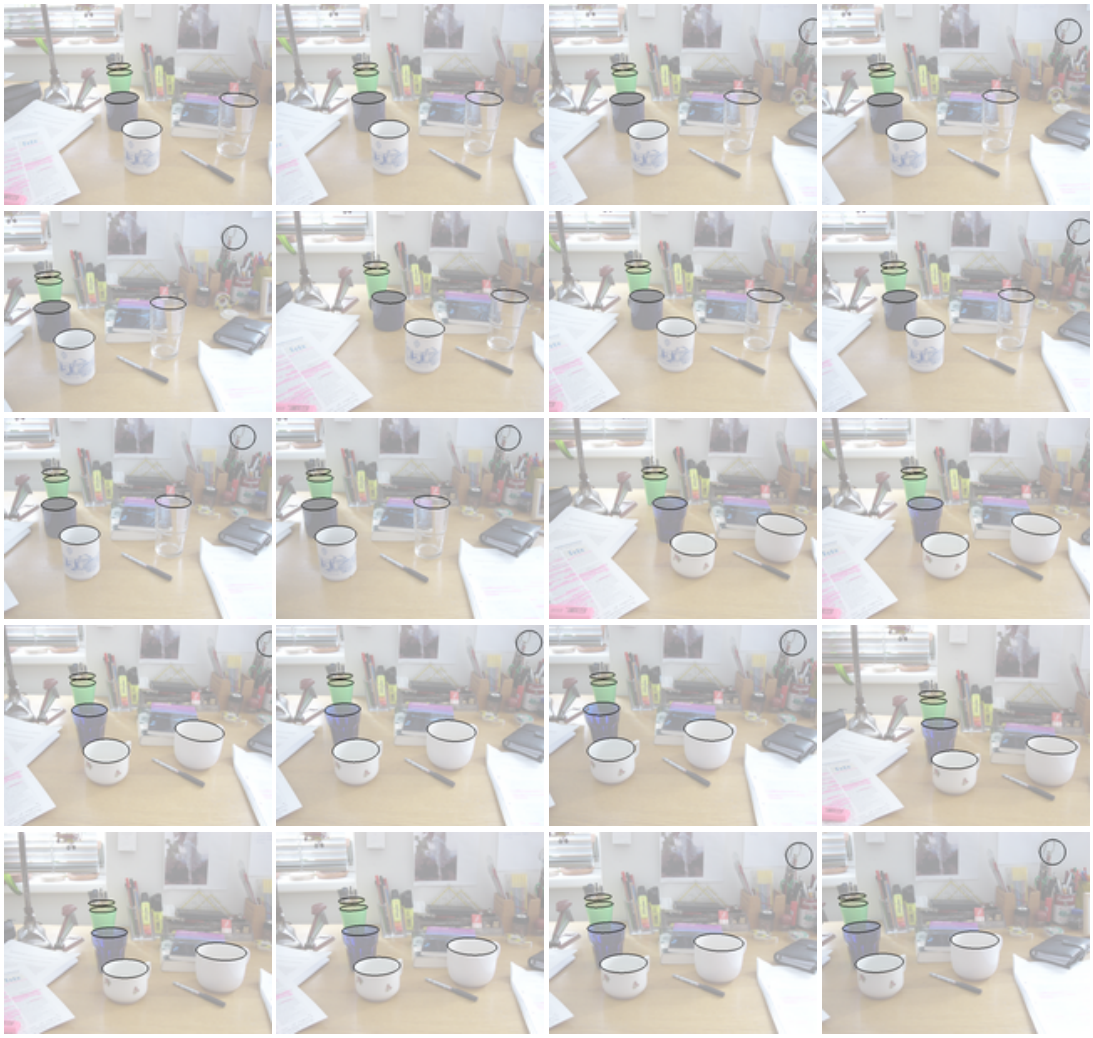


Figure A.1: Office scenes with ground truth ellipses, 40 images in 5 sizes: 320×240, 480, 640×480, 960×720 and 1280×960 (only 20 images shown here).

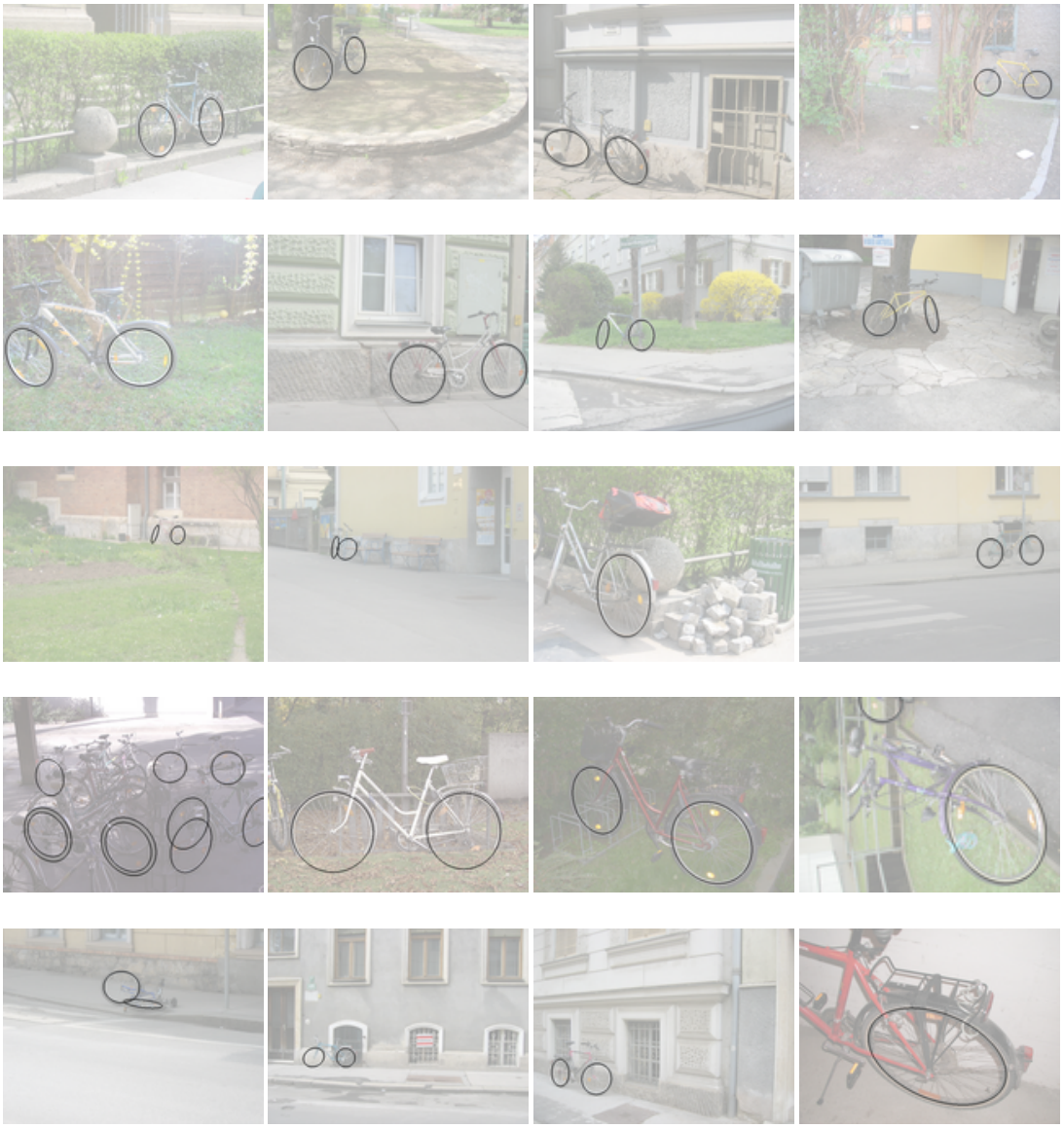


Figure A.2: Bike scenes with ground truth ellipses. Images taken from GRAZ02 database [OP02], 20 images.

Figure A.3: Car scenes with ground truth ellipses. Images taken from GRAZ02 database [OP02], 20 images.

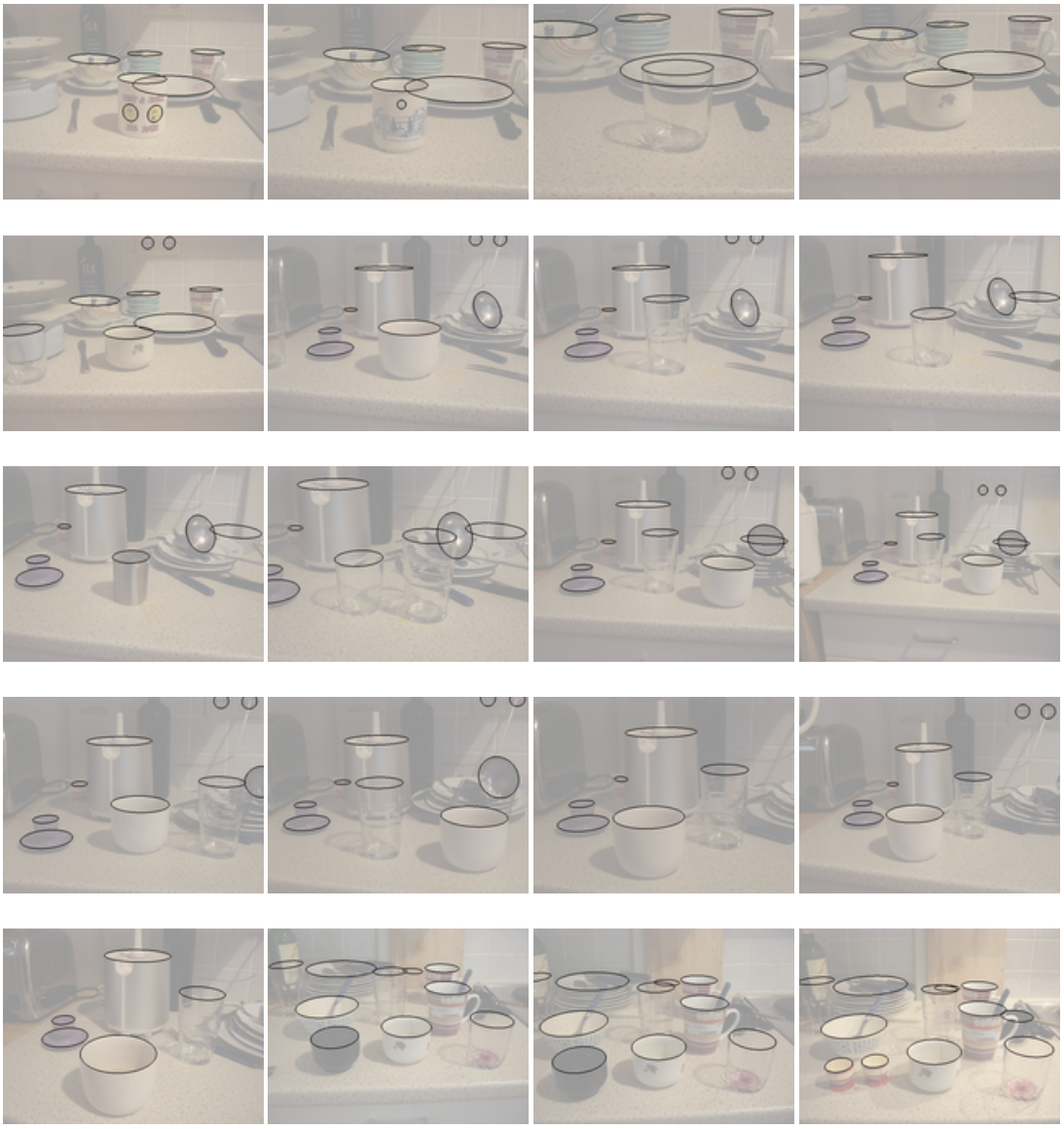


Figure A.4: Kitchen scenes with ground truth ellipses, 20 images.

Figure A.5: Traffic sign scenes with ground truth ellipses, 20 images.

Appendix B

Detecting Closures

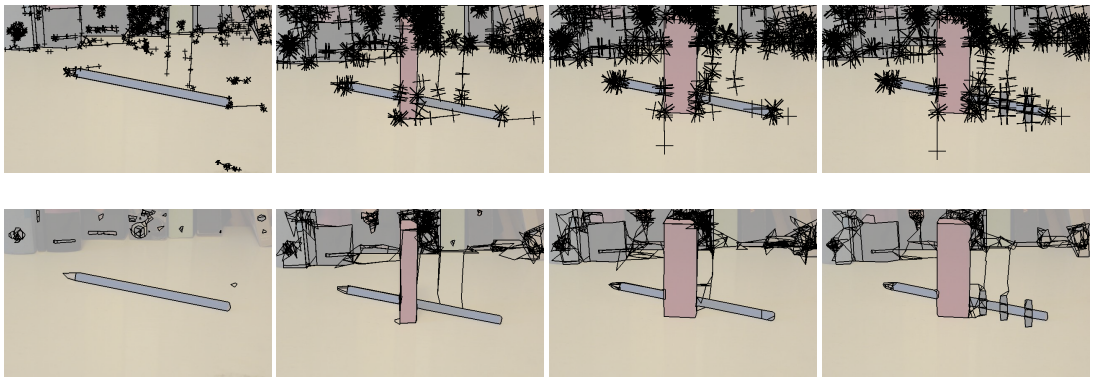


Figure B.1: Pencil with different amounts of occlusion: EQUAL search lines (top row) and detected closures (bottom row).

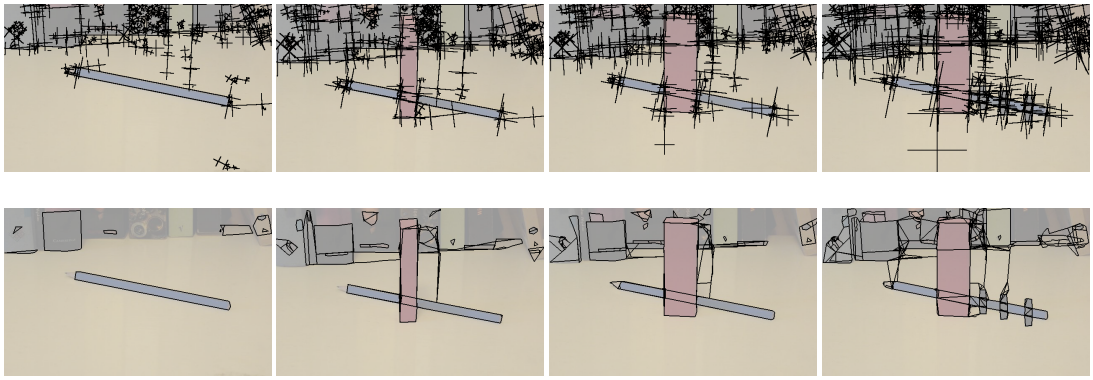


Figure B.2: Pencil with different amounts of occlusion: WEIGHTED search lines (top row) and detected closures (bottom row).

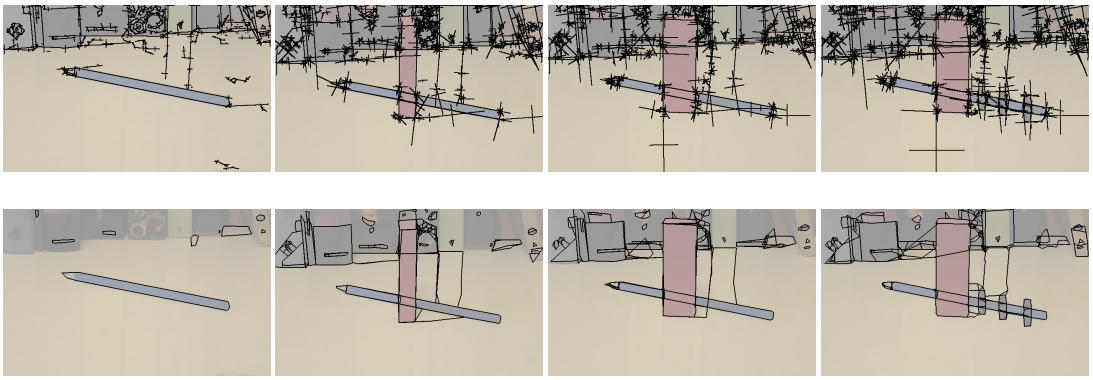


Figure B.3: Pencil with different amounts of occlusion: SMART search lines (top row) and detected closures (bottom row).

Bibliography

[Alb01] Marc K. Albert. Surface perception and the generic view principle. *Trends in Cognitive Sciences*, 5(5):197–203, 2001.

[AMP+97] F. Ackermann, A. Maßmann, S. Posch, G. Sagerer, and D. Schlüter. Perceptual grouping of contour segments using markov random fields. *Pattern Recognition and Image Analysis*, 7(1):11–17, 1997.

[BC70] Colin Blakemore and Graham Cooper. Development of the brain depends on the visual environment. *Nature*, 228:471–478, 1970.

[BGV92] B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *5th Ann. Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, 1992. ACM.

[Bie87] I. Biederman. Recognition by components: A theory of human image understanding. *Psychological Review*, 94(2):115–147, 1987.

[Boo79] F. L. Bookstein. Fitting Conic Sections to Scattered Data. *Computer Graphics and Image Processing*, 9:56–71, 1979.

[BS93] Kim L. Boyer and Sudeep Sarkar. Perceptual organization in computer vision: A review and a proposal for a classificatory structure. *IEEE Transactions on Systems, Man and Cybernetics*, 23(2):382–399, March 1993.

[BS97] Anthony J. Bell and Terrence J. Sejnowski. The 'independent components' of natural scenes are edge filters. *Vision Research*, 37(23):3327–3338, 1997.

[BS99] Kim L. Boyer and Sudeep Sarkar. Perceptual organisation in computer vision: Status, challenges and potential. *Guest Editorial in Computer Vision and Image Understanding*, 76(1):1–5, October 1999.

[BT78] H. G. Barrow and J. M. Tenenbaum. Recovering Intrinsic Scene Characteristics from Images. In A. Hanson and E. Riseman, editors, *Computer Vision Systems*, pages 2–26. Academic Press, New York, 1978.

[BT81] Harry G. Barrow and Jay M. Tenenbaum. Interpreting line drawings as three-dimensional surfaces. *Artificial Intelligence*, 17:75–116, 1981.

[Can86] F. John Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[CB90] P. B. Chou and C. M. Brown. The theory and practice of Bayesian image labeling. *International Journal of Computer Vision*, 4:185–210, 1990.

[CCS+93] P. B. Chou, P. R. Cooper, M. J. Swain, C. M. Brown, and L. E. Wixson. *Markov Random Fields: Theory and Applications*, chapter Probabilistic network inference for cooperative high and low level vision, pages 211–243. Academic Press, Boston, 1993.

[CK99] Justin C. Crowley and Lawrence C. Katz. Development of ocular dominance columns in the absence of retinal input. *Nature Neuroscience*, 2(12):1125–1130, December 1999.

[CM98] Stefano Casadei and Sanjoy Mitter. A perceptual organization approach to contour estimation via composition, compression and pruning of contour hypotheses. Technical Report LIDS-P2415, Lab. for Information and Decision Systems, MIT, April 1998. Presented at the IEEE Workshop on Perceptual Organization in Computer Vision, Santa Barbara, California, June 26, 1998.

[DBB+97] S. Dickinson, R. Bergevin, I. Biederman, J.-O. Eklundh, A. Jain, R. Munck-Fairwod, and A. Pentland. Panel report: The potential of geons for generic 3-d object recognition. In *Image and Vision Computing, Vol. 15, No. 4*, pages 277–292, April 1997.

[Dij59] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[DMM00] Agnes Desolneux, Lionel Moisan, and Jean-Michel Morel. Meaningful alignments. *International Journal of Computer Vision*, 40(1):7–23, 2000.

[DMM02] Agnès Desolneux, Lionel Moisan, and Jean-Michel Morel. Gestalt theory and computer vision. In A. Carsetti, editor, *Seeing, thinking and knowing*, pages 71–101. Kluwer Academic, 2002.

[DMM03] Agnes Desolneux, Lionel Moisan, and Jean-Michel Morel. A Grouping Principle and Four Applications. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 25(4):508–513, April 2003.

[EAB92] T. Ellis, A. Abbood, and B. Brillaut. Ellipse Detection and Matching with Uncertainty. *Image and Vision Computing*, 10(2):271–276, 1992.

[FB81] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Comm. of the ACM*, 24:381–395, 1981.

[FBL06] Sanja Fidler, Berginc Gregor, and Leonardis Ales. Hierarchical Statistical Learning of Generic Parts of Object Structure. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 182–189, 2006.

[FF95] A.W. Fitzgibbon and R.B. Fisher. A buyer's guide to conic fitting, 1995. BMVC95.

[FPF99] Andrew W. Fitzgibbon, Maurizio Pilu, and Robert B. Fisher. Direct least square fitting of ellipses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):476–480, 1999.

[GGS94] W. Gander, G. H. Golub, and R. Strebel. Least-Square Fitting of Circles and Ellipses. *BIT*, 43:558–578, 1994.

[GM96] 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.

[GM04] Gösta H. Granlund and Anders Moe. Unrestricted recognition of 3-D objects for robotics using multi-level triplet invariants. *Artificial Intelligence Magazine*, 25(2):51–67, 2004.

[GPSG01] W. S. Geisler, J. S. Perry, B. J. Super, and D. P. Gallogly. Edge co-occurrence in natural images predicts contour grouping performance. *Vision Research*, 41:711–724, 2001.

[GT93] Fred Glover and Eric Taillard. A user's guide to tabu search. *Annals of Operations Research*, 41(1):1–28, March 1993.

[Hel67] Hermann von Helmholtz. *Handbuch der Physiologischen Optik*. L. Voss, Leipzig, 1867. online version available at Max-Planck-Institute for the History of Science http://vlp.mpiwg-berlin.mpg.de.

[HH63] Richard Held and Alan Hein. Movement-produced stimulation in the development of visually guided behaviour. *Journal of Comparative and Physiological Psychology*, 56(5):872–876, 1963.

[Hou62] P. V. C. Hough. A Method for Recognizing Complex Patterns. US Patent 3,069,654, 1962.

[Huf71] David Huffman. Impossible Objects as Nonsense Surfaces. *Machine Intelligence*, 6, 1971.

[HW91] D.P. Huttenlocher and P.C. Wayner. Finding convex edge groupings in an image. In *IEEE Converence onf Computer Vision and Pattern Recognition*, pages 406–412, June 1991.

[IK87] J. Illingworth and J Kittler. The adaptive Hough transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):690 – 698, September 1987.

[Jac95] D.W. Jacobs. Robust and efficient detection of convex groups. *CVPR*, 93:770–771, 1995.

[JDP97] A. Jacot-Descombes and T. Pun. Asynchronous Perceptual Grouping: From Contours to Relevant 2-D Structures. *Computer Vision and Image Understanding*, 66(1):1–24, April 1997.

[Kan94] Ken-Ichi Kanatani. Statistical Bias of Conic Fitting and Renormalisation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 16(3):320–326, 1994.

[KB01] Timor Kadir and Michael Brady. Scale, Saliency and Image Description. *International Journal of Computer Vision*, 45(2):83–105, November 2001.

[KC02] Lawrence C. Katz and Justin C. Crowley. Development of cortical circuits: lessons from ocular dominance columns. *Nature Reviews Neuroscience*, 3:34–42, January 2002.

[KN98] Tsuyoshi Kawaguchi and Ryo-Ichi Nagata. Ellipse detection using a genetic algorithm. In *Fourteenth International Conference on Pattern Recognition*, volume 1, pages 141–145, 1998.

[KO04] Ken-Ichi Kanatani and Naoya Ohta. Automatic Detection Of Circular Objects By Ellipse Growing. *International Journal of Image and Graphics*, 4(1):35–50, 2004.

[KZB04] Timor Kadir, Andrew Zisserman, and Michael Brady. An Affine Invariant Salient Region Detector. In *European Conference on Computer Vision*, pages 228 – 241, 2004.

[Li95] Stan Z. Li. *Markov Random Field Modeling in Computer Vision*. Springer-Verlag, 1995.

[Low85] D. G. Lowe. *Perceptual Grouping and Visual Recognition*. Kluwer Academic, Boston, 1985.

[Low87] David G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31(3):355–395, 1987.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[Mal87] Jitendra Malik. Interpreting line drawings of curved objects. *International Journal of Computer Vision*, 1:73–103, 1987.

[Mar82] David Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. W. H. Freeman, 1982.

[MBLS01] Jitendra Malik, Serge Belongie, Thomas K. Leung, and Jianbo Shi. Contour and texture analysis for image segmentation. *International Journal of Computer Vision*, 43(1):7–27, 2001.

[McL98] Robert A. McLaughlin. Randomized Hough transform: improved ellipse detection with comparison. *Pattern Recognition Letters*, 19(3-4):299 – 305, March 1998.

[MM89] Jitendra Malik and Dror Maydan. Recovering Three-Dimensional Shape from a Single Image of Curved Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6):555–566, June 1989.

[MWTX03] Shyjan Mahamud, Lance R. Williams, Karvel K. Thornber, and Kanglin Xu. Segmentation of Multiple Salient Closed Contours from Real Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(4), April 2003.

[NNM96] S. A. Nene, S. K. Nayar, and H. Murase. Columbia Object Image Library (COIL-100). Technical Report CUCS-006-96, Dept. Comp. Science, Columbia University, 1996.

[NR79] Y. Nakagawa and A. Rosenfeld. A Note on Polygonal and elliptical approximation of mechanical parts. *Pattern Recognition*, 11:133–142, 1979.

[OF96] B. A. Olshausen and D. J. Field. Emergence of Simple-Cell Receptive Field Properties by Learning a Sparse Code for Natural Images. *Nature*, 381:607–609, 1996.

[OP02] Andreas Opelt and Axel Pinz. GRAZ-02 database. `http://www.emt.tugraz.at/~pinz/data/GRAZ_02/`, 2002.

[Pal99] Stephen Palmer. *Vision Science: Photons to Phenomenology*. MIT Press, 1999.

[Por90] J. Porrill. Fitting Ellipses and Predicting Confidence Envelopes Using a Bias Corrected Kalman Filter. *Image and Vision Computing*, 8(1):37–41, February 1990.

[Pyl00] Zenon W. Pylyshyn. Situating vision in the world. *Trends in Cognitive Sciences*, 4(5):197–207, 2000.

[Pyl01] Zenon W. Pylyshyn. Visual indexes, preconceptual objects, and situated vision. *Trends in Cognitive Sciences*, 80(1-2):127–158, 2001.

[QH99] Ji Qiang and R.M. Haralick. A statistically efficient method for ellipse detection. In *1999 International Conference on Image Processing (ICIP 99)*, volume 2, pages 730–734, 1999.

[RNE00] D. Roobaert, P. Nillius, and J.-O. Eklundh. Comparison of learning approaches to appearance-based 3D object recognition with and without cluttered background. In *Proc. 4th Asian Conference on Computer Vision*, pages 443–448, Taipei, Taiwan, January 2000.

[Rob65] L. G. Roberts. Machine perception of three-dimensional solids. In J. T. Tippett, editor, *Optical and Electro-Optical Information Processing*, pages 159–197. MIT Press, Cambridge, MA, 1965.

[Roc97] Irvin Rock. *Indirect Perception*. Bradford Books Series in Cognitive Psychology. MIT Press, 1997.

[Roo99] D. Roobaert. Improving the Generalisation of Linear Support Vector Machines: an Application to 3D Object Recognition with Cluttered Background. In *Proc. Workshop Support Vector Machines at the Int. Joint Conf. on Artificial Intelligence*, pages 29–33, Stockholm, August 1999.

[Roo00] D. Roobaert. DirectSVM: a fast and simple Support Vector Machine perceptron. In *Proc. IEEE Int. Workshop Neural Networks for Signal Processing*, Sydney, Australia, December 2000.

[Ros93] P. L. Rosin. A Note on the Least Squares Fitting of Ellipses. *Pattern Recognition Letters*, 14:799–808, October 1993.

[Ros96] Paul L. Rosin. Assessing error of fit functions for ellipses. *Graphical models and image processing: GMIP*, 58(5):494–502, 1996.

[RVH99] Roobaert D. and M. M. Van Hulle. View-based 3D Object recognition with Support Vector Machines. In *Proc. IEEE International Workshop on Neural Networks for Signal Processing*, pages 77–84, Wisconsin, USA, August 1999.

[RW95] Paul L. Rosin and G.A.W. West. Non-parametric segmentation of curves into variuos representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(12):1140–1153, 1995.

[RZE01] Danny Roobaert, Michael Zillich, and Jan-Olof Eklundh. A pure learning approach to background-invariant object recognition using pedagogical support vector learning. In *Proceedings of the 2001 IEEE Conference on Computer Vision and Pattern Recognition*, pages 351–357, 2001.

[Sam82] P. D. Sampson. Fitting Conic Sections to Very Scattered Data: An Iterative Refinement of the Bookstein Algorithm. *Computer Graphics and Image Processing*, 9:97–108, 1982.

[SAS00] Jitendra Sharma, Alessandra Angelucci, and Mriganka Sur. Induction of visual orientation modules in auditory cortex. *Nature*, 404:841–847, April 2000.

[SB93] Sudeep Sarkar and Kim L. Boyer. Integration, Inference, and Management of Spatial Information Using Bayesian Networks: Perceptual Organization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(3):256–274, 1993.

[SB94] 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.

[SB95] S. Sarkar and K. L. Boyer. Using Perceptual Inference Networks to Manage Vision Processes. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 62(1):27–46, July 1995.

[Sch01] Daniel Schlüter. *Hierarchisches Perzeptives Gruppieren mit Integration dualer Bildbeschreibungen*. PhD thesis, Bielefeld University, 2001.

[SGR88] M. Sur, P.E. Garraghty, and A.W. Roe. Experimentally induced visual projections into auditory thalamus and cortex. *Science*, 242:1437–1441, 1988.

[SL95] Markus Stricker and Ales Leonardis. Exsel++: A general framework to extract parametric models. In *Computer Analysis of Images and Patterns*, pages 90–97, 1995.

[SP98] Kah Kay Sung and Tomaso Poggio. Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):39–51, 1998.

[SSW+98] C. Saunders, M. O. Stitson, J. Weston, L. Bottou, B. Schoelkopf, and A. Smola. Support Vector Machine - Reference Manual. Technical Report CSD-TR-98-03, Royal Holloway, 1998.

[SU88] Amnon Sha'ashua and Shimon Ullman. Structural Saliency: The Detection of Globally Salient Structures Using a Locally Connected Network. In *IEEE International Conference on Computer Vision*, pages 321–327, 1988.

[SWSP00] D. Schlüter, S. Wachsmuth, G. Sagerer, and S. Posch. Towards an integrated framework for contour–based grouping and object recognition using markov random fiels. In *Proc. International Conference on Image Processing*, volume II, pages 100–103, Vancouver, Sep. 2000. IEEE.

[SX01] B. J. Scholl and Y. Xu. The magical number 4 in vision. *Behavioral and Brain Sciences*, 24(1):145–146, 2001.

[Tau91] G. Taubin. Estimation of Planar Curves, Surfaces and Non-Planar Space Curves Defined by Implicit Equations, WIth Applications to Edge and Range Image Segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(11):1115–1138, November 1991.

[US88] Shimon Ullman and Amnon Sha'ashua. Structural Saliency: The Detection of Globally Salient Structures Using a Locally Connected Network. A. I. Memo 1061, Massachusetts Institute of Technology, July 1988.

[Vap98] V. N. Vapnik. *Statistical Learning Theory.* Adaptive and Learning Systems for Signal Processing. John Wiley & Sons, New-York, 1998.

[vMPS00] Laurie von Melchner, Sarah L. Pallas, and Mriganka Sur. Visual behaviour mediated by retinal projections directed to the auditory pathway. *Nature*, 404:871–876, April 2000.

[Wal75] David Waltz. Understanding line drawings of scenes with shadows. In Patrick Henry Winston, editor, *The Psychology of Computer Vision.* McGraw-Hill, New York, 1975.

[XO93] Lei Xu and E. Oja. Randomized Hough Transform (RHT): Basic Mechanisms, Algorithms and Complexities. *Computer Vision, Graphics, and Image Processing : Image Understanding*, 57(2):131 – 154, March 1993.

[XOK90] Lei Xu, E. Oja, and P. Kultanen. A New Curve Detection Method: Randomized Hough Transform (RHT). *Pattern Recognition Letters*, 11:331 – 338, 1990.

[YIK89] H. K. Yuen, J Illingworth, and J Kittler. Detecting partially occluded ellipses using the Hough transform. *Image and Vision Computing*, 7(1):31–37, February 1989.

[Zha97] Zhengyou Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. Technical Report RR-2676, INRIA, 1997.