Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

Diplomová práce

Jakub Žemlička

Recognition of 3D Objects with Uniform Surface Reflectance

Katedra softwarového inženýrství

Vedoucí diplomové práce: ing. Martin Urban, Phd., Katedra kybernetiky, FEL ČVUT

Studijní program: Informatika, Softwarové systémy - Počítačová grafika

PODĚKOVÁNÍ

Děkuji Ing. Martinovi Urbanovi, Phd. za vedení této diplomové práce. Dále nesmím zapomenout na Doc. Dr. Ing. Jiřího Matase a Ing. Michala Perďocha za jejich cenné připomínky a poznatky. A zvláštní dík firmám Eyedea Recognition, s.r.o. a ZNOVU, s.r.o. za poskytnutí testovacích dat a oporu ve studiu.

Prohlašuji, že jsem svou diplomovou práci napsal(a) samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 10. srpna 2007 Jakub Žemlička

Contents

List of Algorithms

List of Figures

Anotace

Název práce: *Recognition of 3D Objects with Uniform Surface Reflectance*

Autor: *Jakub Žemlička*

Katedra (ústav): *Katedra softwarového inženýrství*

Vedoucí diplomové práce: *Ing. Martin Urban, Phd., Katedra kybernetiky, FEL ČVUT*

e-mail vedoucího: *urbanm@cmp.felk.cvut.cz*

Abstrakt: Jako alternativa k MSERům a Harrisovským bodům byla vyvinuta metoda pro detekci polygonálních struktur v obraze, která je založena na analýze hranových pixelů. Původní motivací bylo rozpoznávání objektů s uniformním povrchem. Detekované polygony lze ovšem také použít jako doplňující příznaky pro rozpoznávání neuniformních objektů. Navrhovaná metoda byla testována na reálné aplikaci týkající se pohledově invariantní detekce SPZ.
Hlavním přínosem našeho postupu je jeho schopnost detekce polygonů bez ohledu na jejich velikost, počet vrcholů nebo orientaci. Polynomiální asymptotická složitost pro nízké stupně hledaných polygonů umožňuje našemu algoritmu jeho nasazení v aplikacích pracujících v reálném čase. Navíc jej lze jednoduše modifikovat pro hledání více typů polygonů najednou.

Klíčová slova: zpracování obrazu, zájmové oblasti, detekce SPZ

Abstract

Title: *Recognition of 3D Objects with Uniform Surface Reflectance*

Author: *Jakub Žemlička*

Department: *Department of Software Engineering*

Supervisor: *Ing. Martin Urban, Phd., Department of Cybernetics, FEE CTU*

Supervisor's e-mail address: *urbanm@cmp.felk.cvut.cz*

Abstract: A new approach based on the edge examination has been developed for the detection of polygonal structures as an alternative to MSERs and Harris points. The recognition of objects with a uniform surface was the initial motivation. The detected polygonal structures can be used as supplementary features for the non-uniform object recognition as well. Finally, the method has been tested on the real application of the viewpoint invariant car license plate detection.
The main contribution of our approach is the ability to detect polygons regardless to their size, number of vertices and orientation. The polynomial time complexity for low polygon degrees allows our method the real-time performance. Moreover, the searching algorithm can be modified to find more types of polygons at a time very easily.

Keywords: image processing, features, regions of interest, license plate detection

Chapter 1

Introduction

1.1 Motivation

The task of the 3D object recognition in a scene captured from an arbitrary viewpoint is a common fundamental problem in the computer vision. Many applications of autonomous recognition systems have found their way into commercial products during last three decades since the computers were capable to process large data sets such as images. With arising demands on vision systems the complexity of a current problem may increase rapidly. Consequently there is no universal method telling how should computer vision problems be solved. Instead, there exists an abundance of methods for solving various well-defined tasks, where the methods are often very specific and seldom can be generalized over a wide range of applications. Many of them are still in the state of basic research.

Recently, a considerable success in solving this task has been addressed by approaches based on matching of local regions. Their descriptors should be invariant to geometric changes which can be modeled by affine transformations under the assumption of local planarity. Thus, a significant viewpoint invariance is achieved even for objects with complicated shapes contrary to global object descriptors considering only their whole appearance. As a result, these approaches are more robust to partial occlusion as well as to cluttered background since it is not required that all local features match.

The first important issue each recognition process should address is an extraction of anchor key points or regions on which local appearance descriptors can be computed. These points and regions should be repeatably detected and localized in images taken over the range of viewpoint and illuminant changes as well as assumed degradation conditions. There exist several methods based on the matching of repeatably distinguished areas which represent the state-of-the-art [32, 31, 27, 30] in the visual recognition. However, the restricted class of objects these methods are applicable to seems to be a main limitation to them - they fail for objects with a uniform surface reflectance. The reason is that they focus on color or intensity cluttered areas because there is a better opportunity to do various independent measurements. Uniform objects are characterized by their shape in the first case whereas their textures provide only supplementary attributes, such as a color, a material or an illumination reflectance. For example, windows on buildings, on cars and other shiny items represent typical objects which can be determined only by their boundaries as their faces reflect the surrounding environment. Objects like chairs, ladders or streetlights form another special class of objects [6] which are not also detectable via common distinguished regions as most of their face is transparent to the background. That's why the texture examination of uniform objects is not suitable for a unique and successful object recognition.

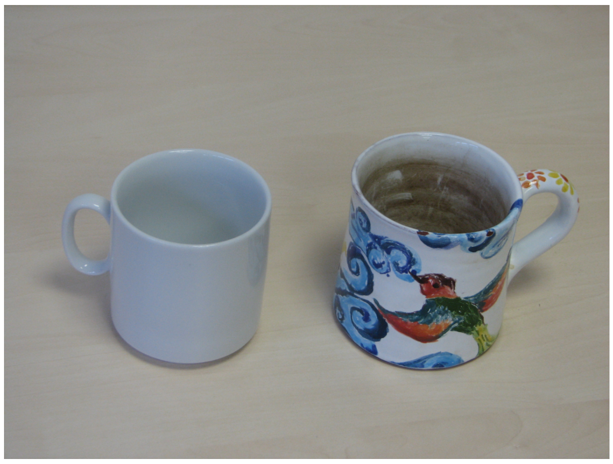


Fig. 1.1: *A difference between the uniform (left) and the non-uniform (right) object.*

The initial goal of this work was to investigate a method dealing with uniform objects while trying to cover this class as much as possible. Edges, corners and boundary pixels carry the most relevant information indicating a presence of these objects in the input image. However, their general recognition based on the contour classification presented itself as a too extensive task to solve within the scope of this thesis and which, in addition, has been also widely described among the literature.

Instead, we attended to the detection of well-defined closed polygonal regions as distinctive features occurring in the single gray-scaled image. It has figured out that looking for polygonal structures was well suitable even for non-uniform objects too. The car license plate localization seemed to be a very interesting real application for our method because license plate occurrences in the image are also conditioned by the presence of their boundary edges. In addition, a fact that a big amount of multi-view traffic images was at our disposal conduced to the decision that the proposed polygon detector was primarily tested on this task. Detected quadrangle hypotheses were compared to manually marked ground-truth license plate positions within the over 3.500 testing images taken from different viewpoints.

Even though our polygon detector was tuned just for the task of the license plate localization, it was executed on building facades to verify a feature repeatability under the given viewing homography. Examples of successfully detected polygons will be shown at the end of this work. Next phases of the object recognition, such as the feature description, feature matching and the object's model verification, have not been investigated in this work so the possible reader concerned in this problem is referred to [34].

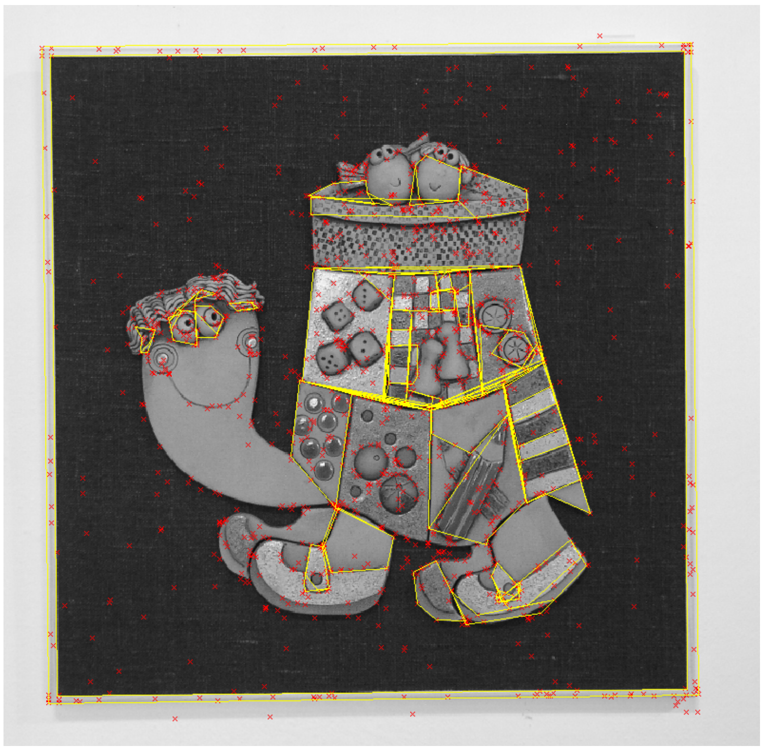


Fig. 1.2: *An example of detected polygonal features (yellow) with junctions (red).*

1.2 Overview of the Proposed Approach

Given a single gray scaled image, our method consists of following steps:

1. *Edge detection using the Canny's operator.*
2. *Extraction of line primitives representing straightest patterns within the binary edge image.*
3. *Finding junctions interpreting probable relationships (intersections) between detected line segments based on their mutual proximity.*
4. *Building a relation graph $G = (V, E)$ where line segments stand for vertices and two of them are connected by an edge if and only if they have a common junction point*

5. *Relation graph optimization by grouping of collinear segments tending to approximate the same straight edge.*
6. *Searching G for well-defined closed cycles corresponding to polygons we wish to detect.*

We also had to challenge several difficulties which occurred during the solving this task:

Edge detector's unreliability accompanied with an object contour disruption. Ordinary edge detectors are not robust enough to identify the same edgel under different homography and illuminant changes. Hence found edges as well as detected line segments are corrupted due to variety of biases. We faced this problem by computing junction points which put detected line segments into their mutual relations and thus help us to bridge occurred gaps. All is performed without any a priori information about their mutual relations.

Efficient cycle evaluation was another problem we had to deal with since the searching of unoriented graphs for all cycles is time consuming. This has been solved by an establishing of special polygon restrictions applied during the graph search which has prevented the algorithm to perform useless computations.

Our polygon detector has been supposed to be robust to geometry as well as to intensity changes providing that relevant edges have been still found. Thus, we do not expect any robustness to blurred images. The noise sensitivity also depends on the edge detector or on any other chosen segmentation method.

1.3 Thesis Outline

Some of the state-of-the-art approaches for the detection of regions of interest are introduced in the next section 1.4. The problematics of polygon detection with other related works are analyzed in the detail in section 2.1. The line detection, junction finding and cycle searching algorithms are presented in sections 2.2, 2.3, 2.4, respectively. Experiment results are contained in chapter 3.

1.4 State-of-the-art of the Local Features Retrieval

A lot of candidate feature types have been described and explored among the literature. A brief review of a recent and past methodology of the feature extraction can be found in [40]. Indeed, many of them are based on corner detectors [5, 31] such as Harris, Harris-Laplace or Hessian-Laplace. Such methods are mostly applicable in the field of the image registration where scenes often have a non-uniform surface. These methods are not usually suitable for the uniform objects recognition because their descriptors don't provide a sufficient determinability in this case.

Lowe in [27] presented a method for the image features retrieval called the Scale Invariant Feature Transform (SIFT). This approach transforms an image into a large collection of local feature vectors (SIFT keys) based on local histograms of oriented gradient changes which are invariant to image translation, scaling and partially to illumination changes. The scale-invariant features are efficiently identified using a staged filtering approach. First, key locations are marked as extremal responses to the Difference of Gaussian filter through the image scale-space. Each point is then used to generate a SIFT vector that describes the local image region sampled relatively to its scale-space coordinate frame.

Tuytelaars et al. in [37] discussed two different methods for the invariant regions extraction. First, a geometry-based, uses Harris corners as seeds for the subsequent extraction of patches as invariant regions. Patches are then attached to their anchor points and adapted to cover the same region in the sense of viewpoint independence. As viewpoint changes, these invariant patches change their shape with regard to extrema of a specific intensity function of the underlying image. Second Tuytelaars's method is intensity-based. Its main purpose is to avoid the inherent unreliability of edge detections and serves as the complement to the first approach. It is directly based on the analysis of image intensity, without an intermediate step involving the extraction of features such as edges or corners. With the local intensity maximum as a center, an affine invariant irregular polygonal region of stable gradient is found to which an ellipse is fitted using moments up to the 2nd order.

A term of the *virtual circle* was first defined by Alhichri [1]. A virtual circle is a circle with maximum radius representing an open space in the image edge map. Circle centers are computed as local maximums of a function resulting in a distance of each pixel to the nearest edge point. The noise sensitivity caused by the nature of edge detectors has been eliminated by the smoothness criterion filtering approach applied on each virtual circle separately. However, a detection of virtual circles is invariant only to scale and shift transforms whereas rotation and other affine similarities were subjects of his consecutive research.

A new type of distinguished regions was presented in [30], the *Maximally Stable Extremal Regions*, MSERs. A MSER is a connected component of pixels which are all brighter (or darker) than all on region's boundary and is maximally stable in a manner of its size changes while the intensity threshold decreases (or increases, respectively). MSERs are characterized by a great covariance to affine and perspective transforms and to monotonic intensity changes. The computational complexity of the evaluating algorithm is almost linear in the number of pixels and thus near real-time performance. Both very fine and coarse image structures are detected since no smoothing is involved but the disadvantage of the sensitivity to corruptions of the region's boundary still remains.

A more detailed comparisons of some of the state-of-the-art feature detectors (also including MSERs and SIFTs) have been performed by Mikolajczyk et. al. in [32].

Chapter 2

Polygon Detection

2.1 Problem Formulation and Related Work

In scenes containing manufactured objects, features often appear as regular polygons, convex polygons, rectangles or skewed parallelograms. For example, triangular, rectangular or octagonal signs display critical information in road scenes. License plates situated on cars are perspectively projected rectangles into the image plane. Buildings and their windows also exhibit features that can be recognized repeatably by the polygon detection. Polygonal shapes and straight structures such as lines appear frequently in aerial images of urbanistic densely populated areas. Indoor scenes may be represented by a large sets of polygons too.

The more is the problem of the polygon detection generalized to cover a wider class of polygons, the more complex this task is. This is the main reason why many algorithms described in a literature are task-specific with various number of restrictions, mostly including a constrained view-range or exact polygon type to be detected. A parallelism with a convexity are usually considered as strong clues reducing a search space. Although the method of the polygon detection presented in this work gives an idea how to detect an arbitrary polygonal shape in an image, its time-complexity does not give a chance to achieve a real time performance without well defined polygon restrictions according to the given task. Next, some of the techniques concerned in the detection of polygonal features will be briefly introduced.

2.1.1 Methods based on Hough Transform

Hough Transform is a standard approach for the detection of different types of geometric primitives, including lines and polygons [9]. In HT based methods, each pixel *votes* for each feature it could possibly be a part of. The efficiency of these methods strongly depends on the parametrization of shapes to be detected, resp. on the dimension of a parametric space to be searched for specific patterns. For example, a generic rectangle has five degrees of freedom: two for center coordinates, and by one for width, height and orientation. Although the mapping into this space is relatively simple for each pixel, the shape generalizations [3] vote into higher dimensions and quickly become computationally expensive. Thus, some methods derive their benefits from the simplifying of a shape representation which results in a constrained capability of an arbitrary shape detection.

Jung and Schramm presented a rectangle detector that uses a Windowed Hough Transform in [18]. Their idea is based on the statement proved by Rosenfeld and Weiss [36] which says that a convex polygon is uniquely determined by the peaks which correspond to its sides and form specific patterns that can be detected directly in its Hough Space. First, line elements are found using HT within the sliding window. Next, several conditions are used to determine if some of them form a rectangle centered in the middle of the sliding window. It can be seen from the nature of the algorithm that it is able to detect only rectangles which do not exceed the size of the sliding window. On the other hand, orientation of such rectangles is not a problem.

Most recently, a novel technique for the regular polygons detection similar to HT has been introduced and applied to detect road signs by Barnes and Loy [28, 4]. Their method extends the concept of the Fast Radial Symmetry Transform [29] to detect regular polygons. The Fast Radial Symmetry Transform is also a method for detecting of points of interest and is similar to circular Hough Transform. Five parameters are needed to represent an arbitrary regular polygon too: Two for center position and orientation, radius and number of sides. However, only 3D space has been examined since the number of sides and polygon scale were fixed. Each pixel then votes in this space for possible centers of a polygon it could be a part of according to its gradient direction. This method is invariant to an in-plane rotation while returning the location and size of the detected shape. A good real-time performance and reliability was claimed by the authors.

2.1.2 Methods based on perceptual grouping

Another techniques reported in the literature and concerned in a detection of polygonal structures often include methods based on the perceptual grouping which organizes detected linear features into the high-level formations. Our approach also belongs to this category.

The common characteristics of methods based on the perceptual grouping is a claimed partial robustness to noise and occlusions because they allow gaps to appear within the line groups. This property is very disputable as it depends on the tolerance of the gap measure, ie. the maximal allowed distance between the given line segment and its possible neighbours. The more are gaps tolerable the more false positive detections occur whereas the zero tolerance can cause that no polygon would be found. Gaps occurrences also expose the problem of detected collinear segments which are probably parts of a single continuous line in the reality. Such line is often corrupted and severed in a number of smaller segments due to the discretization error or the edge detector's inconsistency. Thus the pre-processing which finds collinear sets of lines is usually performed before the self polygonal grouping. However, this optimization takes a lot of processing time since all line pairs have to be examined as their mutual relations are still unknown.

To decide whether some lines form a polygon, their mutual location must be analyzed. Jacobs [17] pointed out that convex sets of line segments with relatively small gaps between neighbouring elements rarely occur at random. This fact enables more efficient finding of such groups and their reliable identification because too distant segments can be ignored. Jacobs presented an exponential algorithm finding first m most salient convex groups in an image having n lines in expected time $O(n^2 \log n + nm)$. The saliency is measured as a percentage of the boundary identified as gaps between end points of neighbouring segments. The line space is recursively searched while evaluating only promising possibilities of convex groups in descending order by their salient factor. Small concativities are also allowed to occur within the convex groups due to sensing errors. However, Jacobs's algorithm failed on groups where one or more line segments can divert the convexity but taking only their parts would have formed strong convex groups (see fig. 2.1).

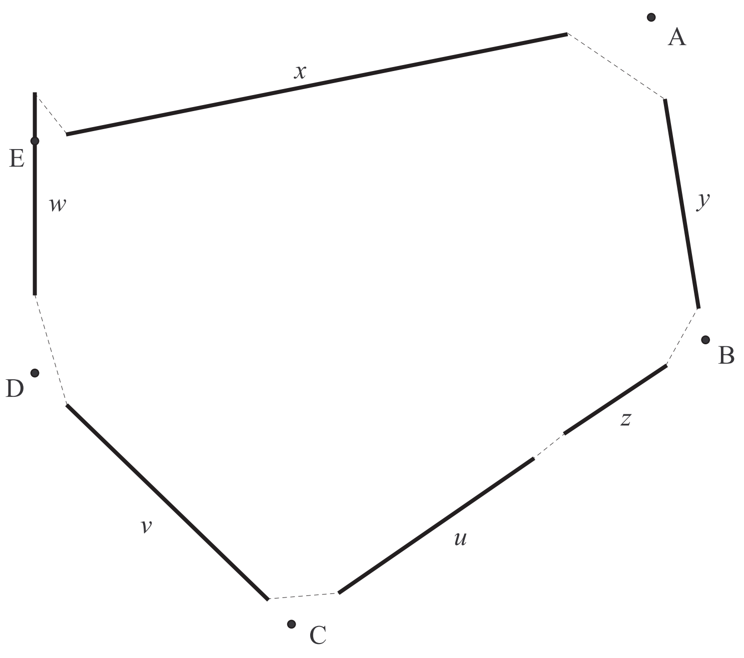


Fig. 2.1: *An example of a group of line segments u,v,w,x,y,z where the Jacob's method would fail. It is obvious that partitioning of w in point E would have formed a strong convex group. Dashed lines represent gaps in the group boundary whereas A,B,C,D,E possible corners of the hypothetical convex polygon.*

Another approach based on the perceptual grouping was studied by Lin and Nevatia [26]. They focused on the detection of flat-roofed buildings in aerial images which may appear as parallelograms that could be only projections of rectangles. Another constrains were derived from the viewing geometry and the orientation of the parallelogram. The process starts with the extraction of line segments within a range of values determined by maximum and minimum building sizes. Neighbouring parallel segments are grouped into a single segment whose length and orientation is derived from the contributing elements. This step helps to partially overcome the problem of fragmented lines generated by low-level vision processes. Next, all L-junctions and T-junctions as probable rectangle corners are found among detected and infinitely extended lines. Then the parallelogram hypotheses can be evaluated. Given an initial line segment, its anti-parallel lines (parallel, but with opposite directions) are found. They together define a region where remaining two sides of the rectangle are searched. As this procedure evaluates an enormous number of false positive hypotheses, they must satisfy several topological conditions in order to proceed to the next stage of the algorithm. The filtering relies on the mutual constellation of junction points telling which line segments really incidents to others.

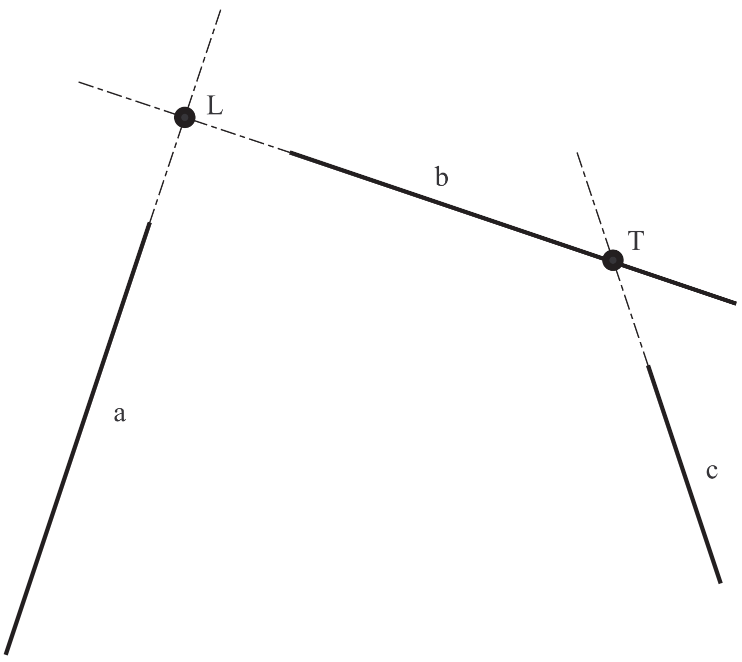


Fig. 2.2: *Points L and T represent an L-junction and a T-junction, respectively.*

Lagunovsky and Ablameyko [23] proposed a similar method that has been intended to detect rectangles in remote sensing images and in industrial images of integrated circuits, ie. in the images where rectangles don't undergo almost any perspective deformation. Their technique differs from [26] by the way of detection of line pairs within the parallelism tolerance. As they claimed, the direct evaluation of all possible line combinations has relatively high computational cost and the number of combinations is enormous. To reduce the number of operations, a 3D parameter space was constructed in such a manner that almost parallel lines which can possibly form opposite sides of a rectangle will be situated close to each other. A rectangle is evaluated by corners found as cross-points of chosen lines by letting them grow until intersection. The mutual relationship of detected pairs is directly analyzed by computing their *rectangularity* value to decide whether the quadrangle is within a confidence level. This value is computed as the maximum distance between the intersection point and its corresponding lines. If it exceeds an allowed threshold, the quadrangle is rejected.

2.1.3 Comparison of our approach with others

Two last mentioned methods ([23, 26]) use junction points as a tool for the consecutive filtering of generated groups of line segments. Contrary to them, we find all feasible junctions first. It can be done in the linear time with respect to the number of pixels all line segments are composed of. This enables to perform the grouping faster and, of course, brings the advantage for the direct cycle evaluation according to our demands. Hence we don't need to compute any gap measure for polygon hypotheses as our junctions follow this condition automatically.

Another limitation of methods described above is the restricted class of polygons they are able to found because the generalization of their grouping processes is not trivial. In our approach, we transformed an input image into a planar graph representation whose vertices correspond to detected line segments. Two vertices are connected with an edge if and only if there exist their appropriate junction. Every closed loop in this graph corresponds to a polygon from the input image. This way is in principle general and allows to detect an arbitrary polygonal shape regardless to the number of its sides, scale, orientation and other properties. It only depends on the description of polygons we wish to detect. Each such a definition is a set of rules which are applied during a cycle search and help the algorithm to reduce the number of its decisions made at each vertex. As mentioned above, the request of the polygon convexity is a very strong clue for real images. Unfortunately, with an increasing degree of freedom of the polygon definition one must count with the higher time consumption, especially when the input image is too complex. The reason is that the finding of all cycles in a graph is NP-hard as it includes the Hamiltonian problem. We will attend to this problem in the section 2.4 more precisely.

2.2 Line Primitives Extraction

At first, the input image is processed by the Canny's edge detector. This step significantly reduces the number of pixels to be examined which is very advantageous especially in the light of the algorithm speed as well as of the fact that non-edge pixels are not interesting for the polygon detection purposes. The gained binary edge image is represented by a system of mutually disjoint and connected sets of edgels (so-called *strings*) such each its element has at most two neighbours in the manner of the standard pixel 8-connectivity. In our approach we exploited the advantage that points in the string are connected and ordered.

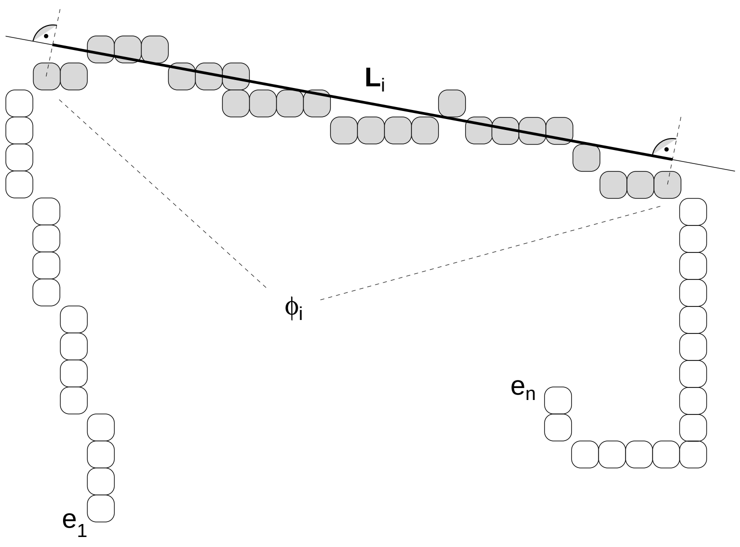


Fig. 2.3: *Illustration scheme of the string with highlighted line segment support (gray pixels).*

The next task is to find such line segments which may constitute parts of desired polygons. Line segments never appear ideally due to the rasterization during the image acquisition. Because we are interested in their detection as accurate as possible, the effort is to identify only the straightest and longest blocks of input strings as the line segment supports. The real problem comes at corners where edgels are usually displaced and form local fuzzy structures. Adding these inappropriate pixels to any neighbouring line support may cause the overall fit inaccuracy (fig. 2.4). So we perceive edgels with too high local curvature as noise and exclude them from further computations. Thus we do not strictly insist on the request that the entire string must be approximated by a single connected poly-line at all cost by which we differ from another algorithms approximating discrete curves because they tend to consider every pixel of their input.

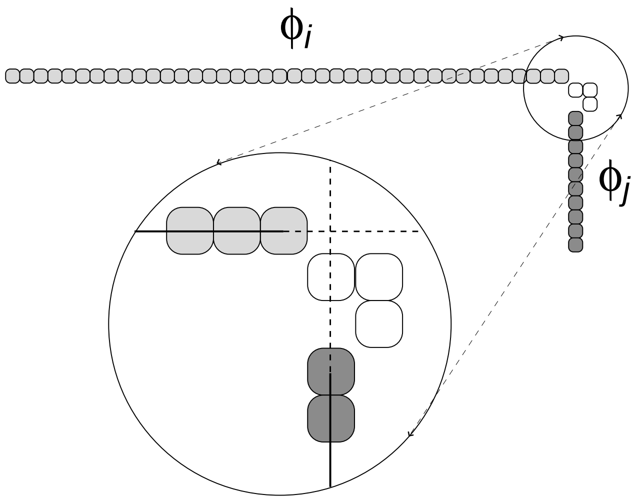


Fig. 2.4: *It is unwished if white pixels would contribute to either of neighbouring supports ϕ_i or ϕ_j which would deflect both line segments from their ideal positions.*

Let $S = \{e_1, e_2, \ldots, e_n\}$ be an arbitrary string gained by the Canny's operator. Let's say that an edgel e_i is on the left of e_j for any $1 \leq i < j \leq n$. Analogically, e_j is on the right of e_i. We wish to approximate the given string S by a set of line segments such that we are looking for an optimal partitioning of S into k mutually exclusive subsets $\phi_1, \phi_2, \ldots, \phi_k$, which form separate supports for the consecutive line fitting. Given a ϕ_i, the problem of identification of line segment L_i approximating ϕ_i is addressed using a classical orthogonal regression technique [2]. Thus, if we mind a ϕ_i as the *support* for the infinite line $\overleftrightarrow{l_i}$ under the given error tolerance, line endings of the segment L_i are found as orthogonal projections of ϕ_i's endpoints on the $\overleftrightarrow{l_i}$. It is obvious that $\bigcup_{i=1}^{k} \phi_i$ is not necessarily equal to S in the manner of our line detection philosophy.

2.2.1 The Fit-Split algorithm

Our Fit-Split algorithm has been inspired by the one presented in [25]. It takes a set of self non-crossing, non-cyclic and continuous strings as its input. The output is the list of all detected line segments. Without loss of generality, we will consider only a single string S as the algorithm's input. Remaining strings are examined in the same way independently. The Fit-Split algorithm works in $O(kn)$ where k is the average number of line segments found within the S of length n.

Let ω_0 be a specific initial set of $5-7$ adjacent pixels from S which we call the *seed*. The seed also forms the support for the initial line $\overleftrightarrow{l_0}$ whose direction specifies an axis of two imaginary infinite bands of pre-defined widths b_{in} and b_{out}, $b_{in} \leq b_{out}$ (fig. 2.5). Let $\delta(x, \overleftrightarrow{l})$ be the orthogonal distance of point x from the line $\overleftrightarrow{l}$ whereas $\delta'(x, L)$ denotes the distance between x and the nearest point lying on the segment L. During the first stage, adjoining edgels are added to ω_0 as long as they lie within the outer band along the foregoing initial line, ie. as long as it holds that $\delta(x, \overleftrightarrow{l_0}) < \frac{b_{out}}{2}, \forall x \in \omega_0$. We say that the seed ω_0 grows while forming the new line support $\phi_0 = \{x : x \in \omega_0 \wedge \delta(x, \overleftrightarrow{l_0}) \leq \frac{b_{in}}{2}\}$. Next, the line $\overleftrightarrow{l_0}$ is re-fitted using only pixels from the ϕ_0. The grow and re-fit stages repeat until the seed is no longer updated. Afterwards, whole process repeats on remaining two (the *left* and *right*) substrings recursively.

Dividing ω_0 in two its separate subsets has two reasons. This allows the algorithm a better accuracy near line endings providing that casual corner edgels can deflect whole line from its exact location. Both line ends consisting of last edgels tending to leave the outer band are cut off due to this little trick so they do not contribute to the line support as long as they don't belong to ϕ_0. However, they possibly will be considered in the next iteration step if the line can benefit from their presence in its support. The second advantage is a controlled robustness to outlayers which also shouldn't fall in ϕ_0, depending on the chosen values of b_{in} and b_{out}. The tighter is the outer band the less it tolerates outlayers. It results in the unreasonable line fragmentation whereas wider inner band yields to greater fitting error.

Assuming that a good-class line should only grow from a high-quality seed, an input string should be searched for the best candidate initially. As we are interested in the most straight sections of the string, we are looking for seeds with maximum weight (ie. straightness) defined as

$$w_i = \frac{1}{1+\lambda_i} \tag{2.1}$$

where λ_i stands for the smallest non-negative eigenvalue of the appropriate covariance matrix related to the seed with center in e_i. It is strongly advisable to begin line growing from the max-weighted seeds.

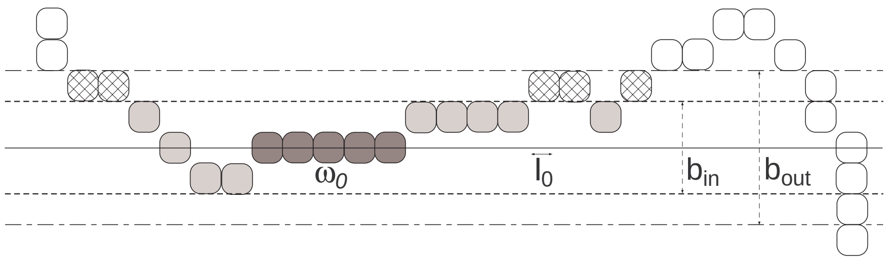
Average fitting error of the line segment L_k is equal to

$$\Lambda_{L_k} = \frac{\sum_{e_i \in \phi_k} \lambda_i}{|\phi_k|} \tag{2.2}$$

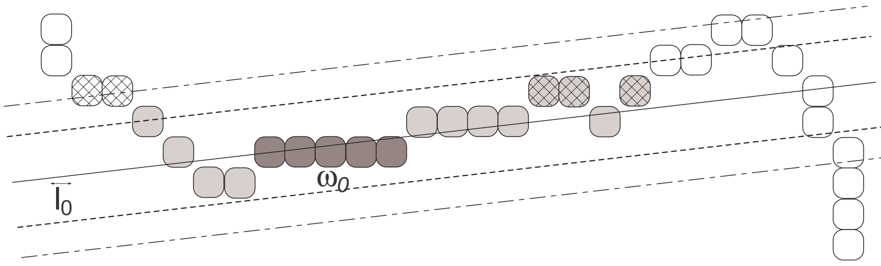
and it represents the line's overall quality measure. It is clear from previous definitions that $\Lambda_{L_k} \geq 0$ and the less it is the more probable is that L_k corresponds to a real one in the image.

2.2.2 Discussion

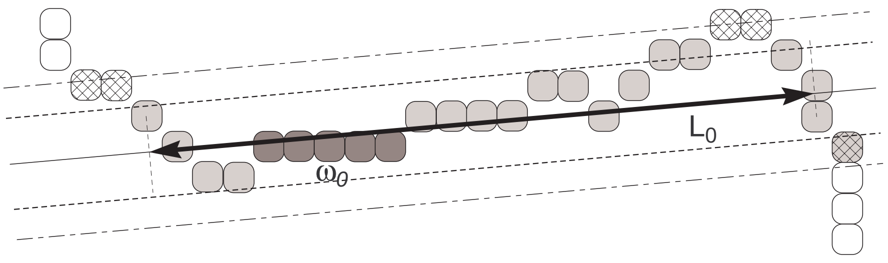
The described algorithm for the line detection performs in $O(kn)$. The fitting stage itself takes $O(n)$ as pixels join the line support and the searching for best and worst pivots also takes at most $O(n)$ in our implementation, both executed k times in average. The worst case is for $k = n$ which would mean that each pixel represents a single degenerated line segment. The Canny's operator has been used for the string extraction.



(a) Seed initialization



(b) Line refit



(c) Final step

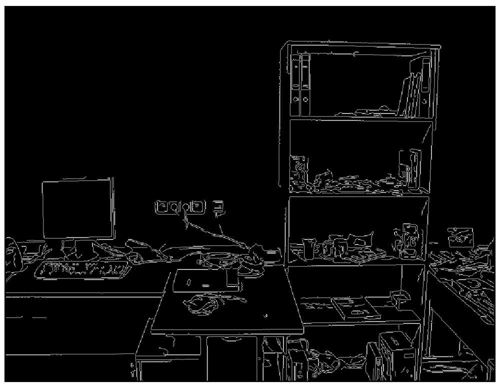
Fig. 2.5: *This figure demonstrates particular stages of the Fit-Split algorithm. Dark pixels form the initial seed ω_0 (fig. 2.5(a)). Pixels marked by the light gray fall into the inner band of the current line $\overleftrightarrow{l_0}$ whereas shaded pixels are always perceived as outlayers. Fig. 2.5(b) shows the refit stage when only dark and light gray pixels are used for the $\overleftrightarrow{l_0}$ recalculation. Note that three shaded pixels on the right fall into the inner band in the next iteration step (fig. 2.5(c)). Finally, the complete line segment L_0 is established while all white pixels are ignored.*

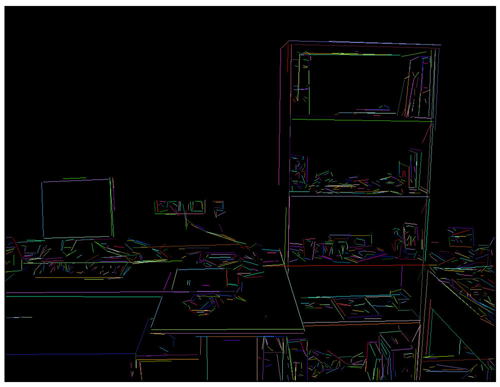
The inner and outer band widths have been chosen very carefully. The fact, that these choices might affect the outlayer sensibility to diagonal lines more than to verticals or horizontals, has also been considered. Thus the inner half-band must be wide at least $\sqrt{2}$ of pixel size in order to allow neighbouring diagonal edgels to join the line support. However, as it was mentioned above, too big values are also inadvisable. Experiments have proved that values of 1.5 and 2.5 for $\frac{b_{in}}{2}$ and $\frac{b_{out}}{2}$, respectively, appear as sufficient estimations for most images regardless to their resolution and the scene. As no images with pre-defined edges and line segments were available, the quality of the whole detection process has been checked visually from the series of real images as well as from synthetic pictures showing geometric primitives. Hence there is no statistical analysis available. Nevertheless, very satisfactory results were achieved by this method.

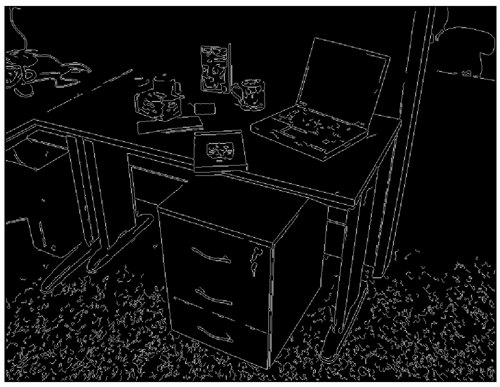
Advantages of the Fit-Split algorithm have outweighed its drawbacks. Specially we highlight the user controlled preciseness of the line fit while outlayers are being omitted. An interesting feature of this algorithm is also its ability to dynamically adjust the direction of bands to increase the number of enclosed pixels and thus to enlarge the line support. As the main disadvantage we see the strong dependency on the edge detector. The Fit-Split algorithm assumes that chains of edgels are not severed. If so, gaps along the line occur, and thus the original line segment appears to be consisting of several disconnected collinear parts. The solution comes with the estimation of so-called junction points to which is the attention given in the next section.

There are a lot of methods reported in the literature over the last 30 years concerned in the polygonal representation of discrete curves and are still in state of the research.

Kumar et al. [22] presented a polygonal approximation of closed curves invariant to projective transformations. Furthermore, let's refer to the work of Feschet et al. [12, 11] as an interesting example of the recent methodology. They proposed a purely discrete $O(n)$ algorithm for the tangent estimation for each of n points of the discrete curve as the improvement of the Vialard's [38] one resulting from Debled's [8]. Each tangent corresponds to largest linear segment of pixels centered at the given point. Although the optimal layout of line segments along the curve has not been investigated in this method, it might show the way of the possible future improvement of our line detector. A similar algorithm to Fechet's can be found in Lagunovsky's [23], where straight linear primitives are grouped to form line segments during the edge detection. However, the cluster analysis of purely discrete methods has a higher sensitivity to corruption of line straightness than algorithms based on approximation. Therefore we combined the efficiency of discrete methods with the analytical accuracy. Although the time complexity of our approach is not optimal with respect to number of found edgels, experiments have proved that it performs fast enough to not become a crucial stumbling block of our polygon detector.


(a)


(b)


(c)

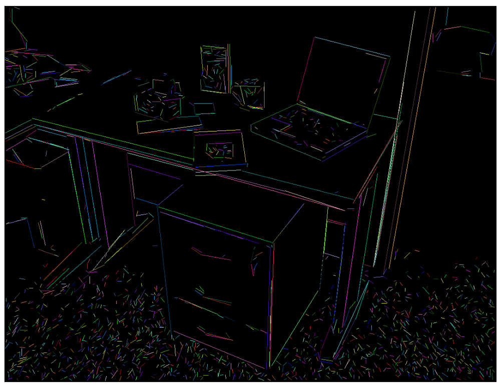

(d)

Fig. 2.6: *Edge maps (left) and detected line segments (right).*

Algorithm 1 The Fit-Split algorithm

```
INPUT: A continuous, non-cyclic edge string $S = \{e_1, e_2, \ldots, e_n\}$
INPUT: Indices $first$ and $last$ such as $1 \leq first < last \leq n$
INPUT: Initial seed size $z$
INPUT: Inner and outer band widths $b_{in}$, $b_{out}$ in pixels
OUTPUT: The list of line segments along the $S$ between $e_{first}$ and $e_{last}$
    /* Initial call is FitSplit($S$,1,$n$,$z$,$b_{in}$,$b_{out}$) */
 1: if $last - first < z$ or $first < 1$ or $last > n$ then
 2:     return $\emptyset$
 3: end if
 4: find $k : w_k = max\{w_i : first \leq i \leq last\}$
 5: set $left := k - \lfloor \frac{z}{2} \rfloor$
 6: set $right := k + \lfloor \frac{z}{2} \rfloor$
 7: set $\phi := \{e_{left}, \ldots e_k, \ldots e_{right}\}$
 8: repeat
 9:     fit line $\overleftrightarrow{l}$ to $\phi$
        /* Grow line support to the left */
10:     while $\delta(e_{left-1}, \overleftrightarrow{l}) < \frac{b_{out}}{2}$ and $left > first$ do
11:         if $\delta(e_{left-1}, \overleftrightarrow{l}) < \frac{b_{in}}{2}$ then
12:             $\phi := \phi \cup e_{left-1}$
13:         end if
14:         $left := left - 1$
15:     end while
        /* Grow line support to the right */
16:     while $\delta(e_{right+1}, \overleftrightarrow{l}) < \frac{b_{out}}{2}$ and $right < last$ do
17:         if $\delta(e_{right+1}, \overleftrightarrow{l}) < \frac{b_{in}}{2}$ then
18:             $\phi := \phi \cup e_{right+1}$
19:         end if
20:         $right := right + 1$
21:     end while
22: until $\phi$ is no longer updated
23: compute $L$ as orthogonal projections of $e_{left}$ and $e_{right}$ on the $\overleftrightarrow{l}$
24: set $right := max(i : e_i \in \phi_k)$
25: set $left := min(i : e_i \in \phi_k)$
26: set $LeftList :=$ FitSplit($S$, $first$, $left - 1$, $z$, $b_{in}$, $b_{out}$)
27: set $RightList :=$ FitSplit($S$, $right + 1$, $last$, $z$, $b_{in}$, $b_{out}$)
28: return $LeftList \cup L \cup RightList$
```

2.3 Finding Junctions

The next objective is to find out which lines or their parts probably form sides of polygons occurring in the image. That's why we are interested in their mutual planar topology and thus in finding their junctions. In general, junctions are perceived more as adjoining relations between line segments rather than their possible intersections if they were extended. The reason is that we also wish to connect collinear segments whose extensions would never intersect or would intersect out of the image boundaries but obviously approximate the same straight edge. That is the reason why no method for finding analytical intersections between lines come in useful to our purposes. Another reason is that we work with discrete and mostly displaced line segments and thus it proved to our advantage to analyze their mutual relations in the image bitmap directly by the use of the Bresenham's algorithm. So, by the junction point we understand a representative of the most probable area where two or more line segments reach the other's proximity. Two line segments are considered proximal if any part of one segment appears close to any part of the other one.

2.3.1 Tolerance Areas

Due to noise in feature detection, we assume that every discrete detected line segment present itself as an entity with uncertain position, length and orientation with respect to the reality. Thus the line attributes provide only a probabilistic information. Thereby we define the so-called *Tolerance Area* (TA) to model this uncertainty associated with each line segment. The TA of a given line segment is a non-empty bounded set of pixels from its immediate vicinity where the original line probably occurs. The junction points are found as means of non-empty intersections between appropriate TAs. Summarized, we demand that each TA

- *is evaluated easily in order to compute their intersections efficiently*
- *model possible occurrences of a given line segment enough to reach TAs of its probable neighbours*
- *do not produce any false junctions*

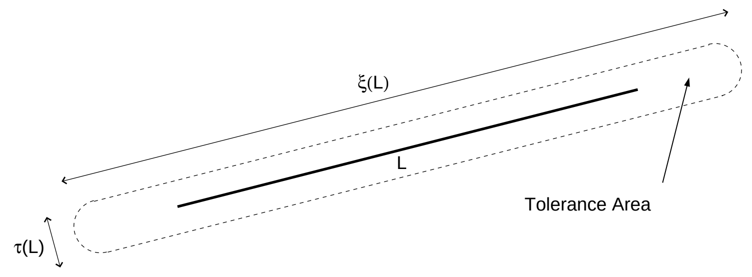


Fig. 2.7: *Line Tolerance Area.*

The definition of each *Tolerance Area* can be any function of the given line segment returning a (weighted) set of pixels from the line segment's vicinity. In our approach, each TA was simply constructed as thickened and to both directions extended original line segment which could be evaluated by the Bresenham's algorithm very easily. Let's call $\tau(L)$ the *acceptable thickness* and $\xi(L)$ the *length* of the TA associated with the line segment L. The thickness $\tau(L)$ has been set to $3-5$ pixels by default for each tolerance area which allowed relations between two parallels approximating the same edge to be established. Obviously, if it had only one pixel width we wouldn't be also sure that two crossing lines (resp. their TAs) had a non-zero intersection due to the pixel discretization (fig. 2.9).

The question arises how much should the original line be extended to satisfy conditions mentioned above? Following facts have been observed from both the real and simulated experiences:

Due to disturbances near corners line segments miss their real endpoints. The effort was to prevent line segments to be overgrown while giving them a sufficient power to compensate leaks and gaps within polygon boundaries. It has figured out that a constant extension of each line segment in both its directions wasn't sufficient enough to cover corrupted areas. To be robust to polygon occlusions, each line segment should be extended much more, even at the cost of the increased count of junctions. Summarized, a length proportional estimation of the line extension has

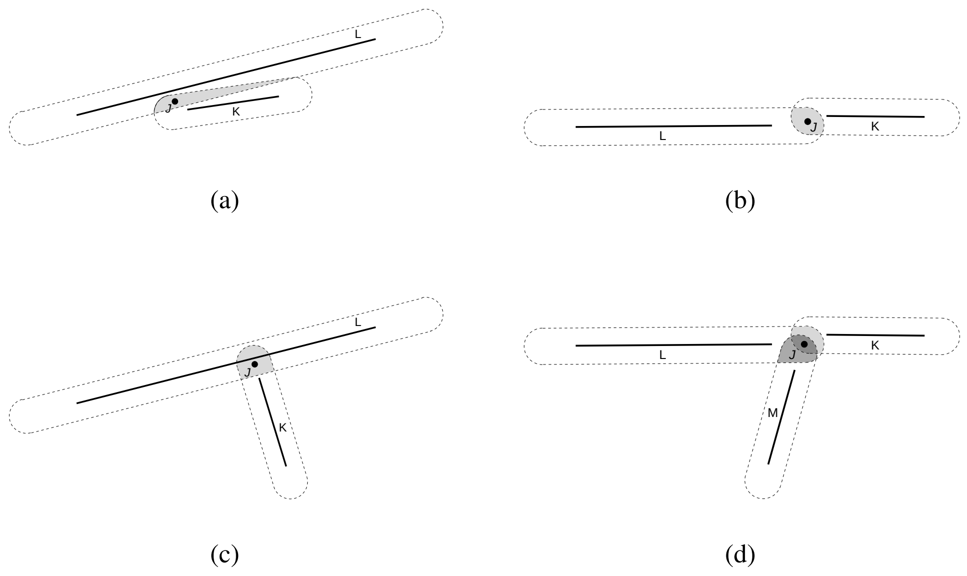


Fig. 2.8: *Examples of junctions between two, resp. three, line segments.*

seemed to be the most reasonable solution. Moreover, the line's average fitting error should also be taken into account. The effort is not to extend line segments with greater fitting error too much in comparison with those whose supports are almost ideal. The goal of this effort is to eliminate too short line segments which originated from displaced or noisy pixels and which would yield into unwanted junctions.

Let $\rho(L)$ be a relative expansion function returning a coefficient of the relative extension for a given line segment L with respect to its length. Therefore, an absolute line extension in one direction is

$$\sigma(L) = |L| \cdot \rho(L) \tag{2.3}$$

where $|L|$ denotes the length of the line segment L. Thus, the total length of an appropriate tolerance area of L is equal to

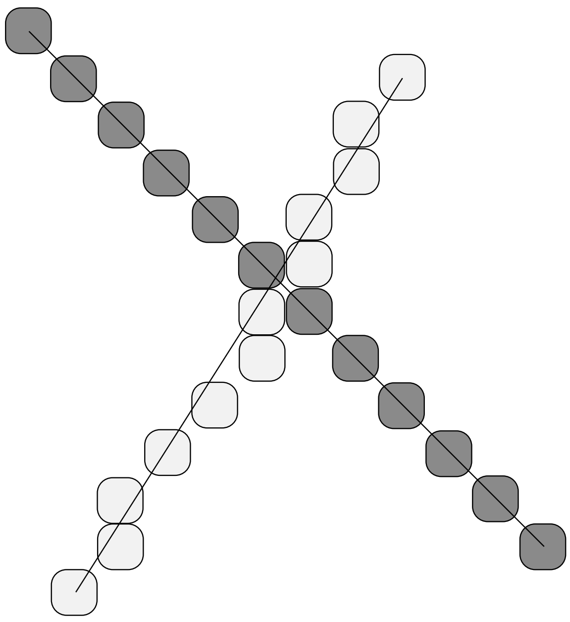


Fig. 2.9: *Example of two line segments whose TAs wouldn't intersect if they had one pixel width.*

$$\begin{aligned} \xi(L) &= |L| + 2\sigma(L) \\ &= |L|(2\rho(L)+1) \end{aligned} \tag{2.4}$$

A $\rho(L)$ can be any function designed for the current application's purposes according to assumed image degradations. The behavior of the polygon detector strongly depends on the choice of this function. We set the $\rho(L)$ as

$$\begin{aligned} &\rho(L) = \frac{1}{\frac{|L|-m}{k}(\Lambda_L+1)+c} \quad for \quad |L| \geq m \\ &\quad else \\ &\rho(L) = 0 \end{aligned} \tag{2.5}$$

for any fixed parameters $k > m > 0$, $c > 0$. The interpretation of this function is that any given line segment L of length at least m is about to be extended by $p = \frac{100}{c}\%$ to both its directions but no more than by k pixels. Good-class segments have been made advantageous by the Λ-parameter which can be optionally multiplied by a constant to influence its effect.

2.3.2 The algorithm

Analytical finding of intersections between sets of pixels may not be trivial. Instead, the non-analytical bitmap-based algorithm has been developed working in the bounded image domain which plays into its hands. Its time complexity is linear with respect to number of pixels all tolerance areas consist of which never can be worse than the image size.

Every tolerance area is printed into the helper bitmap of the same size as the original image using the Bresenham's line draw algorithm (even for thick lines). Each pixel holds the list of all line identifiers passing through in order of their appearance. Thus the intersections between TAs (resp. junctions) are found as local maximas (2) of the discrete 3D function where x and y coordinates denote pixel's position while the z axis its intensity, resp. the total number of lines drawn over the given pixel. From the real image experience, we do not expect that more than four or five lines meet at one point, so the expected memory requirement for this helper data structure corresponds to the size of an ordinary RGBA image.

Every time a pixel is touched during the printing operation its intensity is raised by one, thus it denotes the total number of line segments (resp. their TAs) meeting at its location. Next, all affected pixels are sorted in descendant order in the list $\mathcal{P}$ by their intensity after all segments have been drawn into the helper bitmap. This list can be built in the linear time and helps the algorithm to extract local extremas using a non-maximum suppression.

Let's suppose that each pixel constitutes a potential junction point. Let's denote the set of line identifiers meeting at pixel $[x,y]$ as $\mathcal{S}_{xy}$ while $I(x,y) = |\mathcal{S}_{xy}|$ stands for

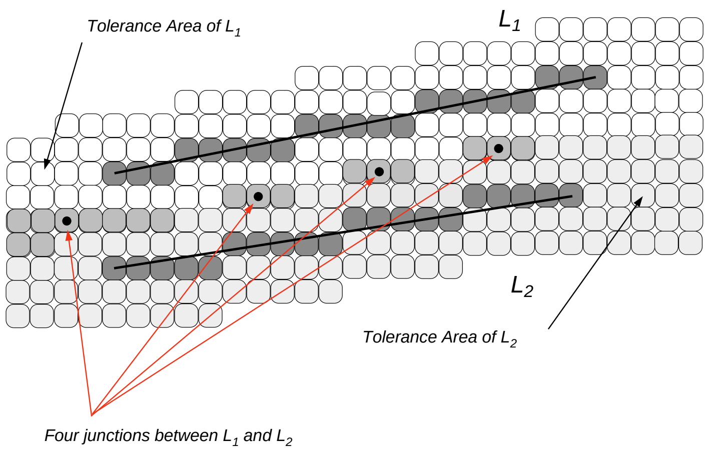


Fig. 2.10: *A side-effect may occur where mutual position of two parallels is examined. As can be seen at this figure, several junctions originated between two parallels L_1 and L_2.*

its intensity. Each junction point, besides its position, is also characterized by a set of incident lines which are common for all neighbouring pixels having the same intensity. Without loss of generalization, let's assume that i-th pixel $[x,y] \in \mathcal{P}$ is being examined and not visited yet. Thus, it creates a basis of a new junction point J whose position J_{pos} is centered to $[x,y]$ initially. The J inherits the $J_{list} = \mathcal{S}_{xy}$ automatically. The J_{pos} is found by an iterative neighbour examination as the mean of all pixels $[x',y']$ reachable from J_{pos} having all $\mathcal{S}_{x'y'} = J_{list}$. Simultaneously, any pixel from the J_{pos} vicinity with $\mathcal{S}_{x'y'} \subseteq J_{list}$ is marked as visited. This, together with the descendant character of $\mathcal{P}$, ensures that no pixel which is not a maximum won't serve as a new junction basis.

It often happens that several junctions are found as multiple responses of two parallel lines (fig. 2.10). It is caused by the discrete error when the intersection of two appropriate tolerance areas is not continuous. In final all such junctions are represented by a single edge in the relation graph.

Algorithm 2 Non-max suppression algorithm

INPUT: $\mathcal{P}$ - descendant sorted list of pixels by their intensity

OUTPUT: $\mathcal{J}$ - list of detected junctions

```
1:  set $\mathcal{J} := \emptyset$
2:  while $\mathcal{P}$ is not empty do
3:      pick up first pixel $[x,y]$ from $\mathcal{P}$
4:      if $[x,y]$ not visited then
5:          set $\mathcal{A} := \emptyset$
6:          set $queue := [x,y]$
7:          while $queue$ is not empty do
8:              pick up first pixel $[x',y']$ from $queue$
9:              if $\mathcal{S}_{x'y'} \subseteq \mathcal{S}_{xy}$ then
10:                 set $[x',y']$ as visited
11:             end if
12:             if $\mathcal{S}_{x'y'} == \mathcal{S}_{xy}$ then
13:                 $\mathcal{A} \leftarrow [x',y']$
14:             end if
15:             $queue \leftarrow$ all neighbouring pixels of $[x',y']$
16:         end while
17:         create new junction $J$
18:         set $J_{list} := \mathcal{S}_{xy}$
19:         set $J_{pos} := \frac{1}{|\mathcal{A}|}\sum_{[x',y']\in\mathcal{A}}[x',y']$
20:         $\mathcal{J} \leftarrow J$
21:     end if
22: end while
```

2.3.3 Relation graph construction

Let $V = \{L_1, L_2, \dots, L_n\}$ be the set of detected line segments denoting vertices in the unoriented relation graph $G = (V, E)$. Let an unique ID to be assigned to each junction such $\mathcal{J} = \{J^1, J^2, \dots, J^m\}$. Two line segments are connected by an edge if and only if they have a common junction. Formally,

$$(e_{ji} = e_{ij} = \{L_i, L_j\} \in E, \quad i \neq j) \Leftrightarrow (\exists J \in \mathcal{J} : L_i \in J_{list} \wedge L_j \in J_{list}) \tag{2.6}$$

If L_i and L_j are not collinear or parallel, the edge e_{ij} represents their crosspoint whose position is inherited from J or can be analytically refined from L_i and L_j if necessary.

In result, pairwise relations between line segments have been found. This has brought a small disadvantage of false cycles when at least three lines are about to meet close to one point. As can be seen at fig. 2.11, lines L_h, L_i and L_j meet at junction indexed by 7. Thus the derived graph contains a cycle $L_h \rightarrow L_i \rightarrow L_j$ whose edges e_{hi}, e_{ij}, e_{jh} define corner points of a possible polygon. Because all these edges originate from the same junction, it implies that this polygon (triangle) would degenerate to one point. So let each junction be uniquely indexed and all edges referring to the same junction to share its index. In our example, edges e_{hi}, e_{ij}, e_{jh} would have been indexed by 7 providing that their common junction has also been previously indexed by 7. This feature helps us to eliminate unwanted false cycles if we made a statement that no pair of edges carrying the same index may occur in any cycle. In the following text, indexed edges and junctions will be intuitively denoted as e^{γ}_{xy} and J^{γ}, respectively.

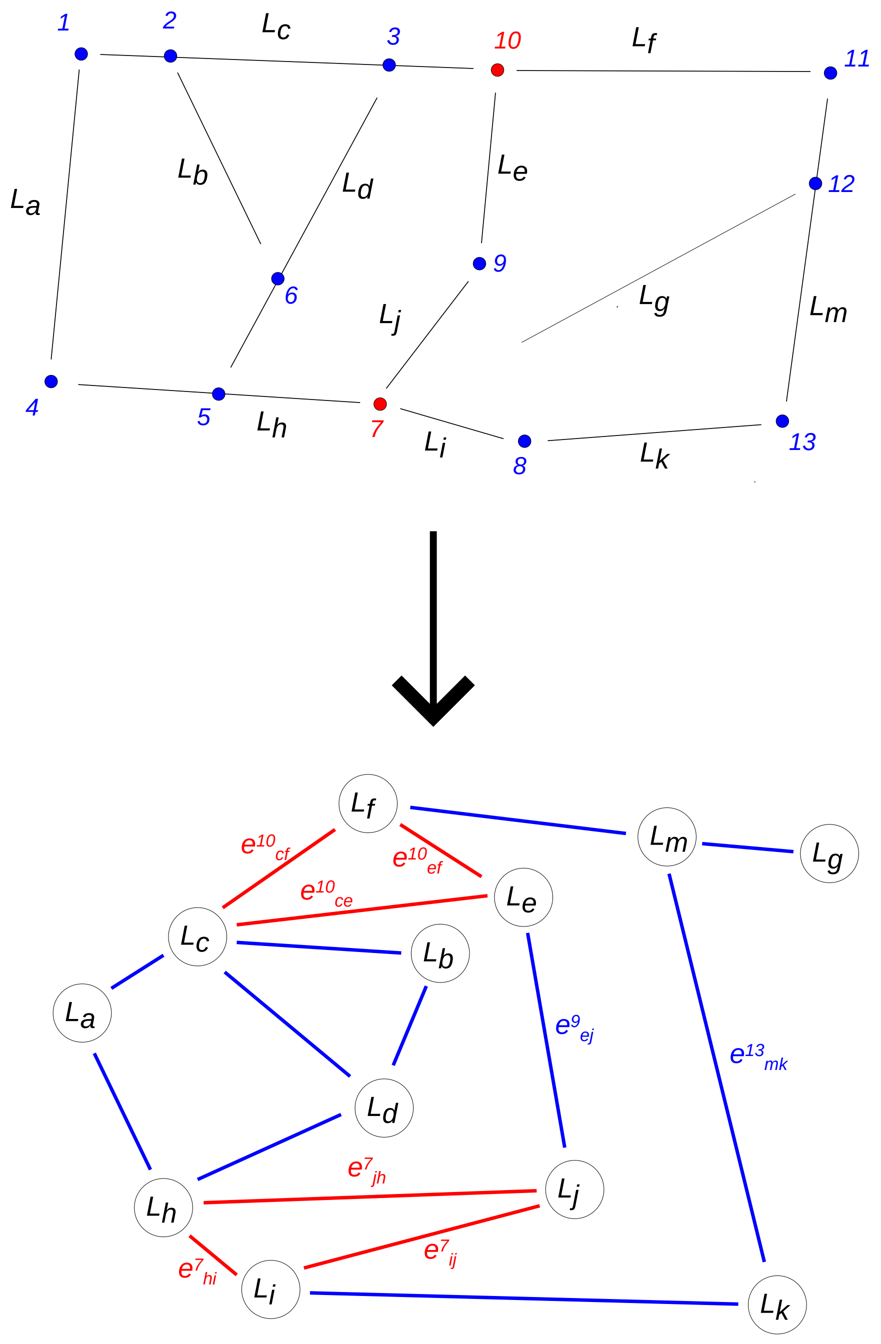


Fig. 2.11: *Relation graph construction. Above: Line segments with colored junctions. Below: Constructed graph.*

2.3.4 Relation graph optimization

It is very convenient to join collinear segments into their single representative which is equivalent to merging vertices of the relation graph. This grouping process optimizes the graph and ensures that searched k-polygons really correspond to cycles of same lengths. In other words, collinear segments constituting parts of one straight edge are eliminated and replaced by their common representative. A knowledge of their mutual relations implies the efficient implementation because it reduces the search space of collinear segments.

Due to noise in feature detection, we must always allow for some deviation from the pure form of any non-accidental property. Hence grouping systems based on the parallelism or the collinearity must always consider lines that are bit non-parallel. Therefore, clues like parallelism in practice provide only probabilistic information. There is a non-zero probability that random orientations will lead to nearly parallel lines so it is likely that two coterminous collinear segments form parts of the same object. These facts yield into following definition:

Definition 1 *Let a_i, b_i and c_i be the normalized parameters of line $\overleftrightarrow{l_i}$ relating to segment L_i such as $a_ix + b_iy + c_i = 0$ and $a_i^2 + b_i^2 = 1$. Two line segments L_i, L_j are parallel under the given tolerance κ, $0 < \kappa \leq 1$ if and only if*

$$\kappa < |\cos \alpha_{ij}| = |[a_i, b_i] \cdot [a_j, b_j]| \leq 1$$

where κ is the inferior threshold for a cosine of angle between the pair of incident lines and the symbol $\cdot$ denotes the dot product between two vectors.

$\square$

If the previous condition is fulfilled for any L_i and L_j we mark this fact as $L_i \parallel_\kappa L_j$. Clearly, if $\alpha = 0$, both lines are purely parallel and for $i = j$ the inequation holds automatically.

Having known the relation graph $G = (V, E)$, clusters of collinear segments are identified as maximal mutually disjoin connected components of G such that all segments within each cluster are *parallel* to their common representative. This central segment also adopts all neighbours of contributing elements while the originals can be discarded safely. The result of this clustering is an optimized graph where all collinear segments are replaced by a single one. Additionally, we demand that no line segment is from its representative orthogonally further than the value of acceptable thickness τ. This imposes more natural requirement and prevent too distant parallel lines to join a common cluster. In other words, we do not let any two line segments to merge if the sum of their distances from their common junction exceeds τ, ie.

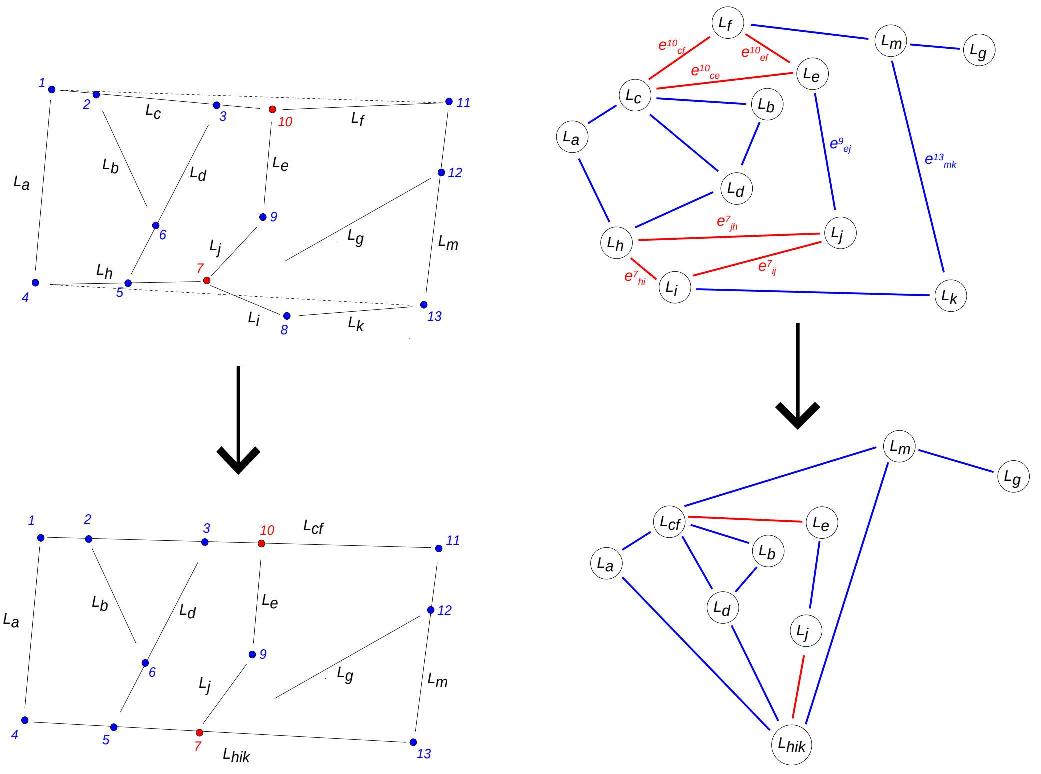
$$\delta(J^{\gamma}_{pos}, \overleftrightarrow{l_i}) + \delta(J^{\gamma}_{pos}, \overleftrightarrow{l_j}) \leq \tau \tag{2.7}$$

for any appropriate L_i and L_j connected by the edge e^{γ}_{ij}. Summarized, the task is to find a complete division of graph's vertices into separate equivalence classes (clusters) according to their mutual coincidence and collinearity:

Definition 2 *A set $C \subseteq V$ is a cluster of collinear segments represented by a segment L_C if and only if*
$(\forall \{L_i, L_j\} \in C, i \neq j) \Rightarrow (\exists$ *connected path P from L_i to L_j, $P \subseteq V : \forall L_P \in P_{ij} \Rightarrow L \parallel_{\kappa} L_C$*$)$

$\square$

The algorithm performing the collinear grouping (alg. 3) combines both the principles of the mean-shift clustering as well as of the graph BFS. Line segments (graph vertices) are grouped (merged) together on basis of their normal vectors similarities. However, only this property is not enough because it yields itself in clusters of all parallels regardless of their mutual locations. Thereby each such cluster has to be divided into separate equivalence classes just in accordance to their mutual coincidences. This is achieved by the BFS part of the algorithm which identifies connected components of the relation graph in a manner of the line segment collinearity. The cluster representative is computed from the two farthest endpoints of segments within the current cluster (fig. 2.12) and are iteratively updated from the principle of the mean-shift procedure.



(a) Line segments (b) Relation graphs

Fig. 2.12: *Line segments before and after collinear grouping (left) and their appropriate relation graphs (right).*

Algorithm 3 Collinear grouping

```
INPUT: Relation $G = (V, E)$
OUTPUT: Optimized $G = (V, E)$
 1: for all unvisited $L_i \in V$ do
 2:     set $L_i$ as visited
 3:     set $C_i := \{L_i\}$
 4:     repeat
 5:         $C_i := \{L_j : e^{\gamma}_{ij} = (L_i, L_j) \in E \wedge L_i \parallel_{\kappa} L_j \wedge \delta(J^{\gamma}_{pos}, \overleftrightarrow{l_i}) + \delta(J^{\gamma}_{pos}, \overleftrightarrow{l_j}) \leq \tau\}$
 6:         update parameters of $L_i$ with respect to $\forall L_k \in C_i$
            /* let vertex $L_i$ to adopt all neighbours of $\forall L_k \in C_i, k \neq i$ */
 7:         $E \leftarrow E \cup \{(L_i, L_n) : (\forall L_k \in C_i, k \neq i \neq n) \Rightarrow (L_k, L_n) \in E\}$
            /* delete $\forall L_k \in C_i, k \neq i$ from $G$ */
 8:         $E \leftarrow E \setminus \{(L_k, L_n) : (L_k, L_n) \in E \wedge L_k \in C_i, k \neq i \neq n\}$
 9:         $V \leftarrow V \setminus L_k : L_k \in C_i, k \neq i$
10:     until $C_i$ is no longer updated
11: end for
```

2.4 Searching a Graph for Specific Cycles

Throughout the following text, we have in mind an unoriented graph $G = (V,E)$ constructed as in the previous section. With the apparent intuition, we always mean by the term *path of length n* the connected sequence of n vertices without any repetitions whereas the *cycle* refers to its closed version. Let's denote path and cycle of n vertices as C_n and $P_n$respectively. Note that $C_n = P_{n+1}$ if and only if first and last vertices of P_{n+1} are equal.

Every cycle C_n, $n \geq 3$, found in the graph substitutes a polygon whose n sides are formed by line segments corresponding to vertices of C_n. It is clear that the number of all possible cycles within the graph can be enormous. Their complete evaluation can grow exponentially with increasing n coming close to $|V|$. According to this fact, we have decided to look only for well-specified cycles designing only a small subset of all of them. Especially, it includes triangles, convex quadrangles, parallelograms, hexagons and octagons as features pointing to man-made objects so the searching for polygons of higher degrees does not seem to make a sense.

It is obvious that finding all cycles first and their consecutive filtering is not feasible for our purposes. Instead, demanded cycles are evaluated directly during the search process. The fact that the collinear grouping has been just performed brings a big advantage to us: The graph became simpler while segments approximating the same edge have been grouped so each C_n really substitutes a unique polygon of the same degree. The only problem is the efficient evaluation of such cycles.

There are many ways of finding cycles in both oriented and unoriented graphs. Mostly, one of fundamental cycle bases [13, 15, 10] is established from which all possible cycle combinations are generated to complete a finite-dimensional vector space of all cycles for a given graph. Every cycle in the graph can be found as a linear combination of basis elements. But the construction of all suitable polygons can not be done without the evaluation of the whole vector space because even a cycle of demanded properties can be composed from those which does not seem to be useful. Contrary, the algorithm of Yuster and Zwick [39] can verify for a constant n if a given graph contains a C_{2n} and finds one if such exists. This algorithm is based on the BFS search executed from each vertex and thus it runs in $O(|V|^2)$. However, the way of evaluation of all cycles has not been investigated. Instead, we have decided for the $O(|V|^{n+1})$ straightforward algorithm which is ease to understand and implement, despite of Monien [33] claims that the decision of existence of P_n between each pair of vertices can be made in $O(n!.|V|.|E|)$. Other methods for the cycle evaluation have not been studied.

Our effort was aimed at minimization of number of decisions to be made during a graph searching which would be impossible in Moniens's method based on incidence matrix multiplications. The constrained polygon degree, parallelism and the convexity are the most useful clues indicating whether a given path is worth to be grown further. Applying these rules does not improve the asymptotic complexity of the algorithm but the set of all possible paths will be reduced very significantly. In result, an almost real time performance has been achieved.

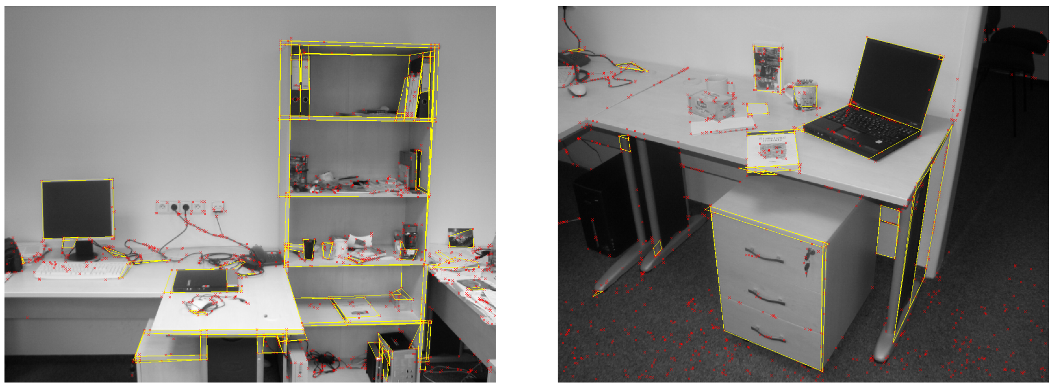

(a) (b)

Fig. 2.13: *Detected polygons (yellow) with junctions (red).*

2.4.1 The algorithm

It holds that no cycle can consist of vertices from different biconnected components of G which can be identified in $O(|E|)$ running a DFS from arbitrary vertex. So, without loss of generalization, let's assume the input graph $G = (V,E)$ to be biconnected. The algorithm for finding all cycles of the given length n is very simple: It runs a brute-force search for all salient paths of length $n+1$. Each path is then tested if its endpoints are equal and if so the cycle C_n is identified. Running a search algorithm from each vertex allows to find all cycles a given vertex is a part of. Such one can be safely discarded so the graph becomes even simpler for the next iteration until no vertex remained in $|V|$. It is also advisable to force the graph to be still biconnected to keep it as small as possible. It can be achieved easily by the recursive vertex removal which, besides the given vertex, also deletes all vertices with at most one neighbour because such vertices cannot be parts of any cycle.

Let us recall that vertices represent line segments and edges their uniquely indexed cross-points since no pair of parallels connected by an edge can occur in the graph due to just performed collinear grouping. To not be confused with previous symbolism, let's follow from now the L_1 to be the first vertex of a path, L_2 the second one and so on. Simultaneously, e_{ij} stands for the crosspoint between L_i and L_j. For convex parallelograms, a vertex L_{i+1} is allowed to join the $P_i = \{L_1, L_2, \ldots, L_i\}$ if conditions in the following order are satisfied to avoid non-promising computations:

1. *Edge $e^{\gamma}_{i,i+1} = \{L_i, L_{i+1}\} \notin P_i$ while the index γ has not occurred in P_i yet.*
2. *The new path length must not exceed $n+1$: $i \leq n$*
3. $\delta'(e^{\gamma}_{i-1,i}, L_i) + \delta'(e^{\gamma}_{i,i+1}, L_i) < |e_{i,i+1} - e_{i-1,i}|$
4. *A convexity must not be broken so vector products $(e^{\gamma}_{i,i+1} - e^{\gamma}_{i-1,i}) \times (e^{\gamma}_{i-1,i} - e^{\gamma}_{i-2,i-1})$ must have the same sign (let's say positive) for $\forall i > 2$. In the other words, it says that the path can turn only to the right (or to the left, depending on the sign). A convexity for remaining edges has also to be checked when the path is about to close.*
5. *Opposite sides of P_{i+1} must remain parallel under the given tolerance: $L_i \parallel_{\kappa} L_{i-\frac{n}{2}}$ for $\forall i > \frac{n}{2}$*

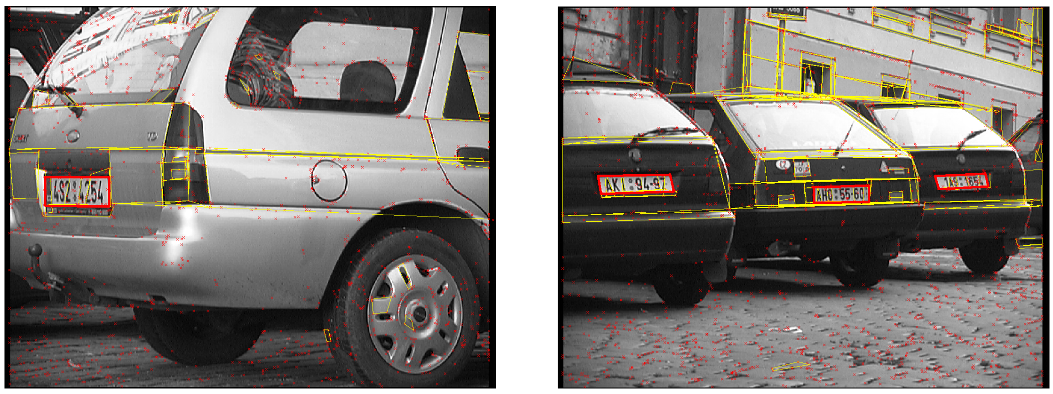
First condition guarantees that no vertex is repeated while preventing from false cycles whereas the second one ensures the proper length of the evaluated path. The fourth and fifth conditions are optional and the need of their fulfilling depends on the given task. If we wished to detect specific cycles such as regular polygons, size constrained polygons or polygons under a specific orientation, another rules had to be established to be applied during a path evaluation. If the convexity is not demanded one must count with the fact that each non-convex cycle would have been found just twice for each its direction as the initial turn has not been specified contrary to the convex case.

Chapter 3

Experiments

3.1 The License Plate Detection

The main experiment was aimed at the detection of car license plates (LP) from multi-view gray-scaled images. This kind of real application represented a great and interesting opportunity to test abilities of our polygon detector. Although license plates can not be considered to be uniform objects, their occurrences in the image are conditioned by the presence of their boundary edgels. Detected quadrangle hypotheses were compared to manually marked ground-truth license plate positions within the set counting over 3.500 testing images taken from various viewpoints and distances under different illuminant conditions. The detector's run time, number of LP hypotheses per image and a probability of searched LPs occurrence among generated hypotheses were the main quality measures.



(a) (b)

Fig. 3.1: *Detected license plates with polygon hypotheses.*

There exist many ways of the car's license plate detection developed during the past 20 years. In general, they mostly include approaches based on template matching [21], intensity thresholding [35], edge statistics [7, 14, 16], local intensity and gradient histogram analysis [24, 20] or adaboost classification [19]. Such methods usually consider the camera's front view which is the most frequent property requested by commercial surveillance systems. They are built on the fact that license plate's background together with its characters constitute high contrast areas in the image. But a problem arises when it is needed to estimate geometric deformations of the license plate precisely for character segmentation purposes. Mentioned systems are claimed to work reliable enough because they are specialized for their specific task, so it wasn't our effort to overcome any of them. Our intention was to find out if our polygon detector could be worth to be considered as a complement of any other LP detector to increase its overall efficiency.

3.1.1 LP detector's brief description

Our experimental LP detector works in following steps:

Step 1: *First, our polygon detector is used to generate all LP hypotheses. This implies that its parameters must be customized according to this task while considering certain projective distortions. In general, license plates can occur as perspectively transformed rectangles in images. This means that we are looking for all convex quadrangles whose opposite sides are almost parallel, ie. they are parallel under the tolerance specified for both the vertical* ($\kappa_V = 0.9$) *and horizontal* ($\kappa_H = 0.94$) *directions separately. The* κ *coefficient has been set to* 0.96. *This settings corresponds to approx.* 25° *to be tolerable deviation for two horizontal parallels,* 20° *to vertical ones and* 16° *for two collinear segments.*

Step 2: *An amount of generated quadrangles can be still high, so it is necessary to perform some filtering. Only polygons with absolute size in pixels greater than* 1.500*px and a dominant horizontal axis can be passed to the next stage of the LP detector. The specified minimal size results from the fact that each LP should have at least* 15*px in height and* 100*px in width unless the characters are not readable. Over a half of hypotheses were eliminated for each image by this step which drastically increased the speed of our LP detector.*

Step 3: *Each polygon hypothesis is normalized to fixed proportions using an estimated affine homography and passed to the SVM module which verifies whether the queried polygon contains a license plate or not. The demonstrational video sequences on the enclosed CD were generated frame by frame with the help of the adopted non-linear SVM classifier whose design wasn't part of this work.*

All the detector's parameters were determined to give optimal results experimentally and are summarized in tab. 3.2. For example, we could have reached better ratio of false negatives if we allowed line segments to be more extended during the junction detection phase or if we tolerated more line inaccuracy, but in the cost of the increased number of hypotheses to be analyzed.

Detection phase	**Parameters**		
Line fitting	$z = 7px$	$b_{in} = 3.6px$	$b_{out} = 5px$
Junction det.	$k = 20px$	$m = 10px$	$c = 2$
Cycle det.	$\kappa = 0.96$	$\kappa_H = 0.94$	$\kappa_V = 0.9$

Fig. 3.2: *Parameters used for the LP detection.*

3.1.2 Experiment results

The testing set consisted of 3.688 gray scaled images containing 3.329 readable license plates. The resolution of testing images varied between $740x287x8$ and $720x576x8$. We have encountered many images with various problems, including

- *a poor ambient lighting*
- *diverse plate locations*
- *a tilt of the plate*
- *a strong influence of sun beams*
- *partially occluded LPs and*
- *blur and noise*

The efficiency of our polygon detector was obtained by comparisons of detected hypotheses to manually defined ground-truth license plate positions defined for each sample image within the testing set. A quadrangle hypothesis was considered as convenient if it contained all the characters of the searched license plate and has the same size and orientation under the given tolerances. Our polygon detector missed about 16% of license plates while generating in average 44 hypotheses per image after the filtering. With the 97% efficiency of the SVM we reached the overall effectiveness of 82% for our LP detector. The computation time spent on each image by our C++ implementation, including the edge detection and hypotheses generation, varied between 2 – 10 FPS on a standard Pentium 4 2.6GHz PC. Unfortunately, we used a 3rd party implementation of Canny's detector. As came out later, its execution took the overwhelming majority of the computation time which influenced the overall performance negatively (fig.3.4). In spite of this, our part of the algorithm (line fitting, junction detection and cycle evaluation) usually took less than 100ms. However, there exist numerous ways in which the code (including ours) could be improved to reduce the required amount of computation time. We believe that the polygon detector could be faster by up to 50% if this were our primary goal. The data gained during our experiment follow in tab.3.3:

Testing set	**# img**	**# lp**	**hyp/img**	**hypf/img**	**# FN**	**err. rate**	**% ok**
IMAGES	2609	2251	66.4	28.5	437	0.194	80.6%
VIDEO1	726	725	366.0	86.7	83	0.115	88.5%
VIDEO2	353	353	174.8	73.5	4	0.011	98.9%
TOTAL	3688	3329	135.7	44.3	524	0.157	84.3%

Fig. 3.3: *Result data measured on the testing set of traffic images.*
From left: No. of images, No. of LPs, No. of detected LP hypotheses, No. of hypotheses after filtering, No. of False Negatives, Error rate and Overall efficiency.

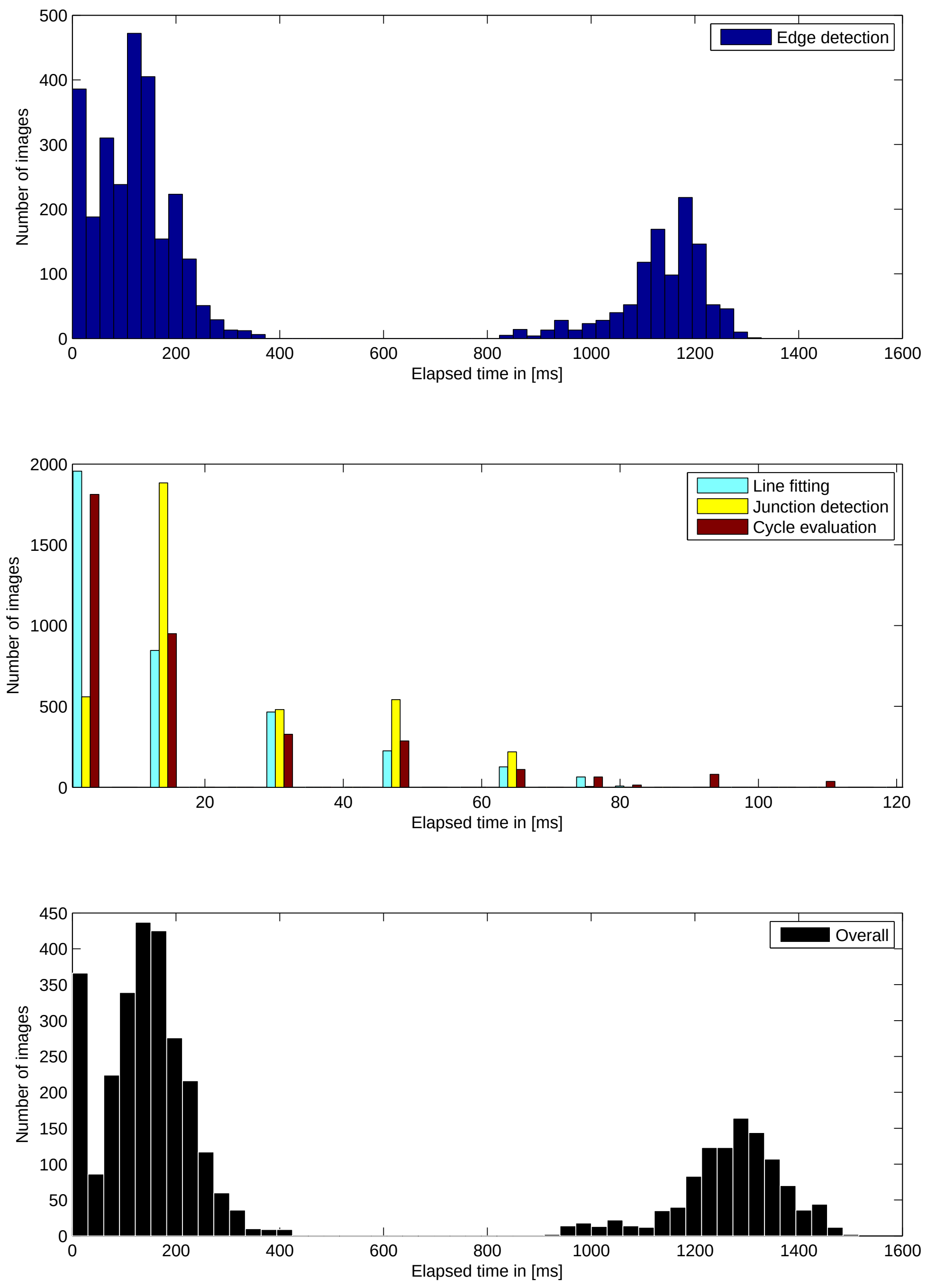
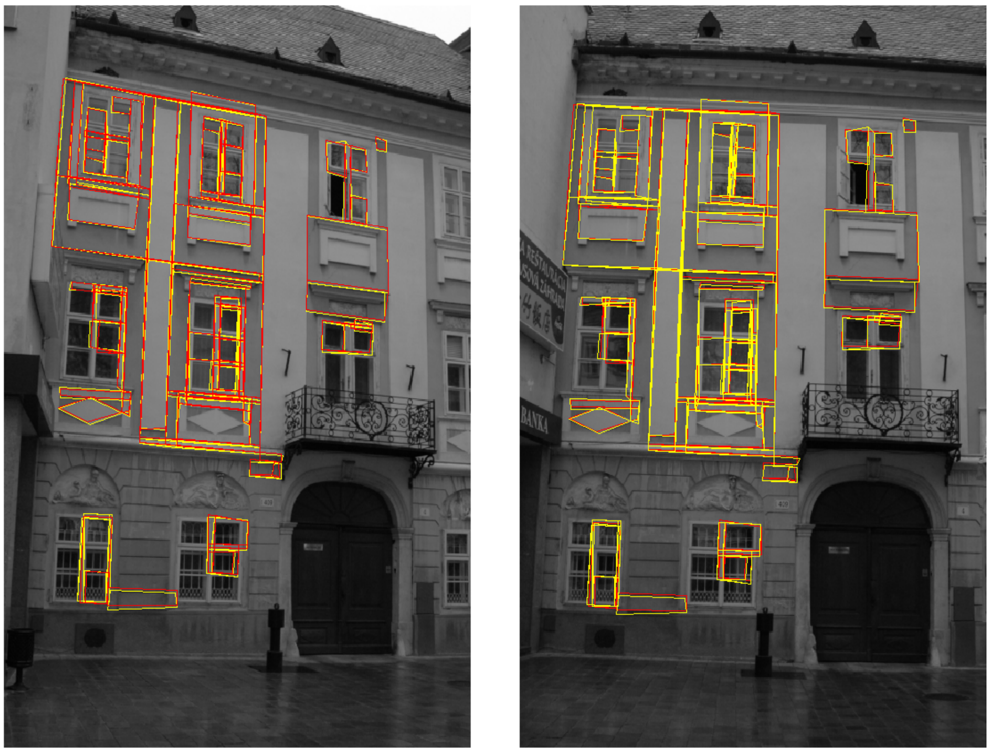


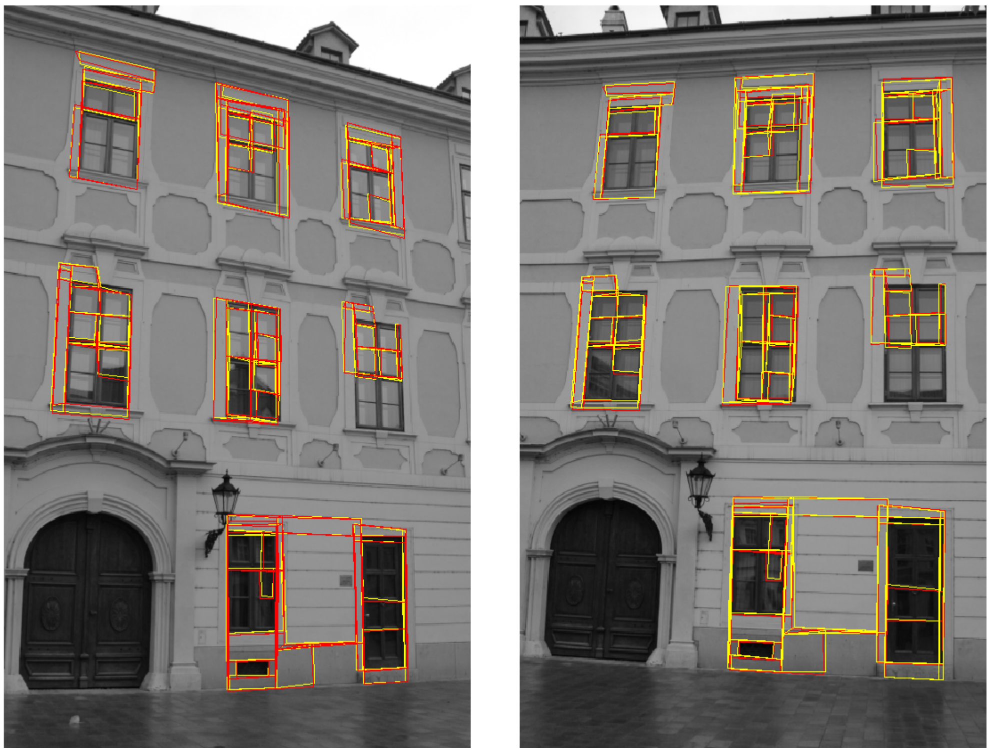
Fig. 3.4: *Time histograms of measured durations on the test set.*

3.2 Feature Repeatability

The second, very short and optional experiment was aimed on the repeated detection of polygonal features within two radially undistorted stereo images of the same resolution of $480x720$ as it would be acquired by an ordinary portable video camera. Testing images depicted building facades on which we estimated their mutual affine homography. We were interested in the total number of successful polygon matches. Following examples show a few hundreds of corresponding features detected in both the left and right stereo images.



(a) 339 matches



(b) 263 matches

Fig. 3.5: *Feature correspondences.*

Chapter 4

Concluding Remarks

We have developed a new polygon detector which is also robust to partial occlusions. The motivation was to recognize objects with a uniform surface reflectance. However the detected polygonal features can serve as the supplementary basis for the recognition of non-uniform objects as well. The detector was called to test its ability in scenes containing both uniform and non-uniform objects. In spite of its polynomial time complexity dependent on the degree of desired polygons, our C++ implementation allows its assignment in real-time vision systems. It has shown off that finding of polygons different from triangles and convex quadrangles, pentagons, hexagons and octagons is worthless because they do not occur in real images in practice.

Our feature detector has been tested on the real application of the viewpoint invariant car license plate (LP) detection. Our detector has been meant as the supplementary alternative to another commercial license plate detector based on MSERs in order to improve its overall efficiency. We hoped that our detector would be able to find some license plates the second detector failed on due to missing or occluded extremas. We achieved the 84% effectiveness only by our approach on the set of over 3.500 testing images of a traffic. It is more optimistic result than we have expected in spite of the strong dependency on the edge detector. The integration with the commercial MSER based LP system decreased the rate of undetected license plates to one third.

Other similar methods ([26, 23]) for the polygon detection often consider only one type of geometric primitives with very constrained properties to be found at a time while the approaches based on Hough Transform ([18, 4]) are limited by their Hough space searching capability. Our method is able to detect numerous types of polygons of any size, scale and orientation simultaneously.

Summarized, our detector of polygonal structures occurring in the image has been found useful as the supplementary feature for several wide base line stereo applications as well as for license plate surveillance systems.

Appendix A

Content of the CD-ROM

A.1 Directory structure

/data samples of testing images

/demo matlab demo application for the polygon detection

- **config.m** default configuration file
- **run.m** main demo script
- **polydetect.mexw32** WIN32 mex file compatible with Matlab7

/source source code of the demo application

/video processed traffic video sequences

thesis.pdf pdf version of this document

A.2 Demo application usage

1. Run Matlab7 under the Windows operating system.
2. Execute the *run.m* script which takes the full image path as the input.

Bibliography

[1] ALHICHRI, H. S., AND KAMEL, M. Virtual circles: a new set of features for fast image registration. *Pattern Recogn. Lett. 24*, 9-10 (2003), 1181–1190.

[2] ANDĚL, J. *Statistické metody.* MATFYZPRESS, MFF UK, Praha, 2003.

[3] BALLARD, D. H. Generalizing the hough transform to detect arbitrary shapes. 714–725.

[4] BARNES, N., LOY, G., SHAW, D., AND ROBLES-KELLY, A. Regular polygon detection. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 778–785.

[5] BAY, H., TUYTELAARS, T., AND VAN GOOL, L. Surf: Speeded up robust features. In *9th European Conference on Computer Vision* (Graz Austria, May 2006).

[6] CARMICHAEL, O., AND HEBERT, M. Shape-based recognition of wiry objects. *IEEE Trans. Pattern Anal. Mach. Intell. 26*, 12 (2004), 1537–1552.

[7] CASTELLO, P., COELHO, C., NINNO, E. D., OTTAVIANI, E., ZANINI, M., AND P. A., E. S. Traffic monitoring in motorways by real-time number plate recognition. In *ICIAP '99: Proceedings of the 10th International Conference on Image Analysis and Processing* (Washington, DC, USA, 1999), IEEE Computer Society, p. 1128.

[8] DEBLED-RENNESSON, I., TABBONE, S., AND WENDLING, L. Fast polygonal approximation of digital curves. In *ICPR '04: Proceedings of the Pattern*

Recognition, 17th International Conference on (ICPR'04) Volume 1 (Washington, DC, USA, 2004), IEEE Computer Society, pp. 465–468.

[9] DUDA, R. O., AND HART, P. E. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM 15*, 1 (1972), 11–15.

[10] FERREIRA, A., FONSECA, M., AND JORGE, J. Polygon detection from a set of lines. In *In Proceedings of 12 Encontro Portugu es de Computac ao Gr afica (12th EPCG)* (Porto, Portugal, 2003), pp. 159–162.

[11] FESCHET, F. Canonical representations of discrete curves. *Pattern Anal. Appl. 8*, 1 (2005), 84–94.

[12] FESCHET, F., AND TOUGNE, L. Optimal time computation of the tangent of a discrete curve: Application to the curvature. In *DCGI '99: Proceedings of the 8th International Conference on Discrete Geometry for Computer Imagery* (London, UK, 1999), Springer-Verlag, pp. 31–40.

[13] GIBBS, N. E. A cycle generation algorithm for finite undirected linear graphs. *J. ACM 16*, 4 (1969), 564–568.

[14] HONGLIANG, B., AND CHANGPING, L. A hybrid license plate extraction method based on edge statistics and morphology. In *ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 2* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 831–834.

[15] HONKANEN, P. A. Circuit enumeration in an undirected graph. In *ACM-SE 16: Proceedings of the 16th annual Southeast regional conference* (New York, NY, USA, 1978), ACM Press, pp. 49–53.

[16] HSIEH, J.-W., YU, S.-H., AND CHEN, Y.-S. Morphology-based license plate detection from complex scenes. In *ICPR '02: Proceedings of the 16 th International Conference on Pattern Recognition (ICPR'02) Volume 3* (Washington, DC, USA, 2002), IEEE Computer Society, p. 30176.

[17] JACOBS, D. W. Robust and efficient detection of salient convex groups. *IEEE Trans. Pattern Anal. Mach. Intell. 18*, 1 (1996), 23–37.

[18] JUNG, C. R., AND SCHRAMM, R. Rectangle detection based on a windowed hough transform. In *SIBGRAPI '04: Proceedings of the Computer Graphics*

and Image Processing, XVII Brazilian Symposium on (SIBGRAPI'04) (Washington, DC, USA, 2004), IEEE Computer Society, pp. 113–120.

[19] KHAMMARI, A., NASHASHIBI, F., ABRAMSON, Y., AND LAURGEAU, C. Vehicle detection combining gradient analysis and adaboost classification. In *Intelligent Transportation Systems* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 66–71.

[20] KIM, S., KIM, D., RYU, Y., AND KIM, G. A robust license-plate extraction method under complex image conditions. In *ICPR '02: Proceedings of the 16 th International Conference on Pattern Recognition (ICPR'02) Volume 3* (Washington, DC, USA, 2002), IEEE Computer Society, p. 30216.

[21] KO, M.-A., AND KIM, Y.-M. License plate surveillance system using weighted template matching. In *32nd Applied Imagery Pattern Recognition Workshop* (Washington, DC, USA, 2003), IEEE Computer Society, pp. 269–274.

[22] KUMAR, M., GOYAL, S., JAWAHAR, C., AND NARAYANAN, P. Polygonal approximation of closed curves across multiple views. In *Indian Conference on Computer Vision, Graphics and Image Processing (2002)*, pp. 317 – 322.

[23] LAGUNOVSKY, D., AND ABLAMEYKO, S. Straight-line-based primitive extraction in grey-scale object recognition. *Pattern Recogn. Lett. 20*, 10 (1999), 1005–1014.

[24] LEE, H.-J., CHEN, S.-Y., AND WANG, S.-Z. Extraction and recognition of license plates of motorcycles and vehicles on highways. In *ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 4* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 356–359.

[25] LEUNG, M. K., AND YANG, Y.-H. Dynamic strip algorithm in curve fitting. *Comput. Vision Graph. Image Process. 51*, 2 (1990), 146–165.

[26] LIN, C., AND NEVATIA, R. Building detection and description from a single intensity image. *Comput. Vis. Image Underst. 72*, 2 (1998), 101–121.

[27] LOWE, D. G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision 60*, 2 (2004), 91–110.

[28] LOY, G., AND BARNES, N. Fast shape-based road sign detection for a driver assistance system. In *IEEE/RSJ Interational Conference on Intelligent Robots and Systems (IROS2004), Sept 28 - Oct 2, 2004* (Sendai, Japan, 2004), IEEE Computer Society, pp. 70–75.

[29] LOY, G., AND ZELINSKY, A. A fast radial symmetry transform for detecting points of interest. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part I* (London, UK, 2002), Springer-Verlag, pp. 358–368.

[30] MATAS, J., CHUM, O., URBAN, M., AND PAJDLA, T. Robust wide baseline stereo from maximally stable extremal regions. In *Proceedings of the British Machine Vision Conference* (London, UK, September 2002), P. L. Rosin and D. Marshall, Eds., vol. 1, BMVA, pp. 384–393.

[31] MIKOLAJCZYK, K., AND SCHMID, C. Scale and affine invariant interest point detectors. *International Journal of Computer Vision 60*, 1 (2004), 63–86.

[32] MIKOLAJCZYK, K., TUYTELAARS, T., SCHMID, C., ZISSERMAN, A., MATAS, J., SCHAFFALITZKY, F., KADIR, T., AND VAN GOOL, L. A comparison of affine region detectors. *International Journal of Computer Vision 65*, 1/2 (2005), 43–72.

[33] MONIEN, B. How to find long paths efficiently. *Annals of Discrete Mathematics 25* (1985), 239–254.

[34] OBDRŽÁLEK, Š. Object recognition using local affine frames. *PhD thesis* (2006).

[35] OZBAY, S., AND ERCELEBI, E. Automatic vehicle identification by plate recognition. In *Transactions On Engineering, Computing And Technology* (Tokyo, Japan, 2005), vol. 9, WORLD ENFORMATIKA SOCIETY, pp. 564–568.

[36] ROSENFELD, A., AND WEISS, I. A convex polygon is determined by its hough transform. *Pattern Recogn. Lett. 16*, 3 (1995), 305–306.

[37] TUYTELAARS, T., AND VAN GOOL, L. Matching widely separated views based on affine invariant regions. *Int. J. Comput. Vision 59*, 1 (2004), 61–85.

[38] VIALARD, A. Geometrical parameters extraction from discrete paths. In *DCGA '96: Proceedings of the 6th International Workshop on Discrete Geometry for Computer Imagery* (London, UK, 1996), Springer-Verlag, pp. 24–35.

[39] YUSTER, R., AND ZWICK, U. Finding even cycles even faster. In *Automata, Languages and Programming* (1994), pp. 532–543.

[40] ZITOVÁ, B., AND FLUSSER, J. Image registration methods: a survey. *Image and Vision Computing 21*, 11 (October 2003), 977–1000.