A Vision-Based Approach for Tramway Rail Extraction

Matthijs H. Zwemer, Dennis W.J.M. van de Wouw, Egbert Jaspers, Sveta Zinger and Peter H.N. de With

a ViNotion B.V., Horsten 1, 5612 AX, Eindhoven, The Netherlands;
b Eindhoven University of Technology, Den Dolech 2, 5612 AZ, Eindhoven, The Netherlands

ABSTRACT

The growing traffic density in cities fuels the desire for collision assessment systems on public transportation. For this application, video analysis is broadly accepted as a cornerstone. For trams, the localization of tramway tracks is an essential ingredient of such a system, in order to estimate a safety margin for crossing traffic participants. Tramway-track detection is a challenging task due to the urban environment with clutter, sharp curves and occlusions of the track. In this paper, we present a novel and generic system to detect the tramway track in advance of the tram position. The system incorporates an inverse perspective mapping and a-priori geometry knowledge of the rails to find possible track segments. The contribution of this paper involves the creation of a new track reconstruction algorithm which is based on graph theory. To this end, we define track segments as vertices in a graph, in which edges represent feasible connections. This graph is then converted to a max-cost arborescence graph, and the best path is selected according to its location and additional temporal information based on a maximum a-posteriori estimate. The proposed system clearly outperforms a railway-track detector. Furthermore, the system performance is validated on 3,600 manually annotated frames. The obtained results are promising, where straight tracks are found in more than 90% of the images and complete curves are still detected in 35% of the cases.

Keywords: Railway accidents, Railway safety, Tramway-track detection, Graph theory

1. INTRODUCTION

With a growing number of urban traffic participants, the desire grows for automatic collision assessment systems. Such a system can reduce the number of casualties by alerting the driver when a collision is imminent. Although driver support systems for cars and railways are widely investigated, current systems are insufficiently reliable for use in trams, due to the challenging conditions of an urban environment. Moreover, the lack of manoeuvring possibilities and the long stopping distance of a tram require the system to look farther ahead to safely avoid collisions. For a reliable collision assessment, the driving trajectory of the tram is required, which is obtained by detecting the tramway track. Therefore, this paper proposes a vision-based approach for tramway-track extraction for trams operating in an urban environment. This is a challenging task for a vision-based system, due to different weather conditions, varying textures of the road surface as well as the complex shape and (partial) occlusions of the tramway track caused by other traffic participants (Fig. 1).

A recently proposed railway-track detection system by Ross, models the railway track as the sum of two Gaussians, one for each rail. The track is found in the camera image by scanning from the bottom upwards, using a Kalman filter to estimate the track geometry. This approach fails to detect the track in sharp curves, as the horizontal distance between the rails is assumed to be linearly decreasing towards zero in the horizon. To make the system more reliable, Ross has also presented a method using probabilistic spline curves to estimate the tracks. A bird’s-eye view image is used to determine multiple spline control points which are tracked over time. The system reliably detects branches in the track, but occlusions are not handled.

Collision prevention systems for trains typically search for objects on the track by looking ahead as far as possible. However, these methods are not robust against occlusions, since they stop searching for more distant tracks when parts of the tracks are occluded. Gschwandtner et al. propose to detect the railway track not as an entity, but rather in parts. They use Difference of Gaussians (DoG) filters with large kernel sizes for edge detection and fit straight lines to the detected edges. The lines are then paired to form track candidates, which are combined by fitting a second-order polynomial to consecutive parts. This allows the track to be detected up
Figure 1. Three challenging scenarios for tramway-track detection in an urban environment. Several challenges can be observed, such as additional line structures, clutter and multiple track possibilities.

...to a distance of 40 meters, while small occlusions are dealt with (e.g. rain drops on the camera lens). However, in an urban environment occlusions are typically much larger and the shape geometry of the rails is much more complex at larger distances.

Weichselbaum et al.\textsuperscript{11} record the track in advance on which the train is driving, using a laser scanner and GPS positions. During operation, the system uses a GPS receiver for self-localization, such that the trajectory of the track ahead can be predicted. However, this projected track proves to be inaccurate, due to the elastic suspension causing pitch and roll movements of the vehicle. Furthermore, GPS reception is often poor in an urban environment.

Our proposed tramway-track detection system uses an approach similar to Gschwandtner et al.\textsuperscript{10} in which a bird’s-eye view is applied to find the track in segments. The novelty in this paper lies in a graph-based reconstruction of the track, such that the complex shape and (partially) occluded track segments can be handled. The presented work is extensively validated on a real-life dataset acquired from a tram.

The remainder of this paper is organized as follows. First, a description of the system architecture is given in Section 2. Tramway detection is explained in Section 3. Section 4 discusses our novel track reconstruction method. An extensive validation is presented in Section 5 including a comparison with the system of Ross. Finally, conclusions and recommendations are discussed in Section 6.

2. SYSTEM ARCHITECTURE

The proposed tramway-track detection is part of a larger collision-avoidance system as shown in Figure 2. In this paper, the tramway-track detection is discussed only. The latter can be divided into two stages. In the first stage, the camera image is transformed to a bird’s-eye-view to remove perspective distortion. Track detection then becomes the problem of detecting a pair of parallel, fixed set of rails having a mutual predefined distance (i.e. track candidates). The latter is performed in parallel for non-overlapping horizontal image slices, reducing the effect of occlusions and improving computations efficiency.
In the next stage, the track is reconstructed from the detected track candidates by creating an arborescence graph (Fig. 4d). In this graph the most feasible driving trajectory is selected based on temporal information.

3. TRACK DETECTION

This section first summarizes the track-detection system and then presents specific details of the algorithm with respect to filtering, line finding and track construction.

3.1 Overview of the track-detection stage

This subsection provides an overview of the track-detection stage, which is activated to find track candidates in the image. First, perspective distortion is removed by transforming the camera image to a bird’s-eye view by Inverse Perspective Mapping (IPM), based on the procedure described by Muad et al.

This bird’s-eye view image is divided into 16 horizontal slices (Fig. 3b), such that the rail can be approximated by a straight line in each slice. A dedicated rail filter is applied to each slice to enhance the visibility of the present rail segments. Next, thresholding results in a binary image providing the rail pixels. A line-detection algorithm then finds rail structures in this binary image. Track candidates are now constructed by employing a line-pair constraint, i.e. searching for parallel lines having a predefined mutual distance. The filtering, line finding and track-candidate construction steps are discussed in the following subsections.

3.2 Edge extraction for rail detection

Besides the rails, there are many local (well-defined) edges in the bird’s-eye view caused by the pavement, grass or other small structures (clutter). To increase the detection accuracy of the rails, a specific filter is introduced which emphasizes the width of the rails. This width $w_r$ is specifically employed in the following equation:

$$f_R(x, y) = \begin{cases} 
1 & |x| < \frac{w_r}{2}, \\
-1 & \frac{w_r}{2} \leq |x| < w_r, \\
0 & x \geq w_r, |y| > h \ (elsewhere),
\end{cases}$$

where $h$ is the height of the filter in pixels. To compensate for the signal variation (e.g. contrast), the filter response is normalized to the signal energy. For each pixel, the energy is computed over the complete region.
around this pixel, as defined by the filter kernel (Eqn. (1)). Because rails can be lighter or darker compared to their surroundings (see Fig. 3a where the color of the track changes w.r.t. the distance), the absolute value of the filter response is employed. The final filter response of filter $f_R(x, y)$ applied to image $I(x, y)$ is defined by:

$$h_R(x, y) = \frac{|f_R(x, y) \otimes I(x, y)|}{\sqrt{\sum_{x,y} I(x, y)^2}}.$$  \hspace{1cm} (2)

The resulting normalized filter response $h_R(x, y)$ has a bi-modal histogram containing two distinct classes: rail-like pixels and background pixels. A threshold is determined and applied to the filter response by minimizing the intra-class variance (combined spread) using Otsu’s method. The resulting binary image after applying the threshold is shown in Fig. 3c.

### 3.3 Detecting Rail Candidates

Rail candidates are straight-line segments extracted from the binary map resulting from the rail-edge extraction, as discussed in the previous subsection. Line-segment selection is commonly based on using the Hough transform or the RANSAC algorithm. In this paper we use RANSAC, as it was shown by Farin that both achieve similar results, where the RANSAC implementation is computationally more efficient.

RANSAC is a randomized algorithm that proposes a set of model parameters. The quality of these model parameters is evaluated and the best hypothesis for the line is selected. In our implementation, the RANSAC algorithm creates a line hypothesis by selecting two random pixels $p(x_1, y_1), q(x_2, y_2)$ from the set of rail pixels $\mathcal{P}$, where the pixel positions satisfy a line-model constraint, in the form of $x = ay + b$, according to Eq. (4). First, the line-model parameters are described by

$$a = (x_2 - x_1)/(y_2 - y_1),$$
$$b = (x_1y_2 - y_1x_2)/(y_2 - y_1).$$  \hspace{1cm} (3)

These parameters are subject to a constraint that the tramway is dominantly visible in the vertical direction ($\alpha_r \approx 0$) and can only slightly bend to left or right. The constraint on parameter $\alpha_r$ speeds up the algorithm, because rails with a larger angle than $\alpha_r$ are not taken into account. This constraint is described by

$$a \leq \tan \alpha_r.$$  \hspace{1cm} (4)

The test that a pixel belongs to the line segment is implemented in a simple computation, which evaluates the distance $d(x', y')$ to the previous line model. This distance is given by

$$d(x', y') = |ay' + b - x'|.$$  \hspace{1cm} (5)

The distance is calculated for all line pixels in a hypothesis and used to compute a normalized score $s_{rail}$ by

$$s_{rail} = \frac{\sum_{(x',y') \in \mathcal{P}} \max(\tau - d(x', y'), 0)}{N\tau^2},$$  \hspace{1cm} (6)

where $\omega_r = 2\tau - 1$ defines the typical rail width in the binary image and $N$ denotes the number of pixels involved. The score $s_{rail}$ represents the support of a line, given by the number of white pixels close to that line, weighted by their distance to the line and normalized between $0 \leq s_{rail} \leq 1$. The process is repeated until a sufficiently high amount of line hypotheses are generated. The hypothesis with the highest score is then selected. The number of hypotheses necessary for a reliable result is determined in Section 5.

The described RANSAC algorithm detects the most dominant line in the dataset $\mathcal{P}$. The pixels supporting this hypothesis are removed and the algorithm starts from the beginning to find another line model until sufficient models are found. The RANSAC procedure ends as soon as a line has a score below $s_{min}$, or if a maximum of $M$ line models are found. The minimum score ensures that the algorithm terminates when only low scoring line models are left in the dataset. The minimum score $s_{min}$ and the maximum number of line models $M$ are determined in Section 5.

Parts of the obtained rail candidates correspond to non-rail pixels, such as lane markings, curbs, pedestrian crossings or other line-like structures. Such false rail candidates are suppressed by the line-pair constraint, which is discussed in the next subsection.
3.4 Track candidates

A track candidate consists of a left and right rail candidate and represents a piece of a possible tramway track. In the normal situation, the rails have approximately the same orientation, so that the orientation of the tramway track is found by

$$\alpha_{\text{track}} = \frac{\alpha_{\text{rail},\text{left}} + \alpha_{\text{rail},\text{right}}}{2}. \quad (7)$$

The difference in orientation between two rail candidates is used as a measure for parallelism and is constrained by $\alpha_{\text{max}}$, which defines the maximum allowed difference in angle, defined by $|\alpha_{\text{rail},\text{left}} - \alpha_{\text{rail},\text{right}}| < \alpha_{\text{max}}$. Its value is determined in Section 5. The a-priori known track width between the rails and the parallelism of the rails are exploited to remove false rail candidates and create track candidates. These candidates $P_{\text{rail}}$ are found by evaluating that a set of two candidates with similar orientation $\alpha_{\text{track}}$ lead to a track width $w_{\text{track}}$, specified by

$$w_{\text{track}} = \left| (P_{\text{rail},\text{left}} - P_{\text{rail},\text{right}}) \cos (\alpha_{\text{track}}) \right|, \quad (8)$$

while satisfying the constraint $|w_{\text{track}} - w_{t}| \leq w_{\text{max}}$. This parameter $w_{\text{max}}$ defines the maximum deviation from the actual track width $w_{t}$, and is determined in Section 5. Finally, multiple instances of the same line are merged.

4. TRACK RECONSTRUCTION

Now that track candidates are available, the complete track needs to be reconstructed. We define the track reconstruction as a graph problem of which the properties are explained in the following subsection. In the subsection thereafter, the derived graphs are filtered with respect to inappropriate connections. Finally, in the third subsection the correct graph is selected.

4.1 Connection graph

We consider the track reconstruction as a generic graph problem, in which weights are assigned to edges between data points (vertices) that possibly form a path. This results in a so-called directed acyclic graph. Each track candidate is a vertex, and the weights of the edges are defined by the angle and position of the track candidate. The graph is reduced to an arborescence graph, in which the most feasible path is found, i.e. the track on which the tram is driving.

Figure 4. Our track reconstruction (best viewed in color). (a) The bird’s-eye view of a frame. (b) All detected track candidates denoted by red dots, where the lines represent their directions. (c) The created graph showing all feasible connections. All track candidates are also connected to a starting point, however, these connections are left out for clarity. (d) The resulting arborescence graph in which only the best connections are kept. (e) The selected track.
A path in a graph representing the complete track can be found by connecting the track candidates. This set still contains false candidates (although they are satisfying the constraints of Section 3) and those belonging to the incorrect track, i.e. the track(s) not driven on. Also, there is a possibility that no track candidates are found in a slice, or even multiple adjacent slices, e.g. when occlusions occur. To cope with this, a weighted directed graph \( G(V,E) \) is defined with root \( r = V_0 \), in which track candidates are represented by vertices \( V \). Edges \( E_{ij} \) between vertices \( V_i \) and \( V_j \) are created if the corresponding track candidates form a feasible connection. Furthermore, all vertices are connected to root \( r \). The root can be seen as a track candidate, which is located underneath the tram. This results in a connection graph, of which an example is shown in Fig. 4c.

An edge \( E_{ij} \) between vertices \( V_i \) and \( V_j \) is a feasible connection if the following two properties hold. First, corresponding track candidates can be connected if the curvature \( \max(K_{ij}) \) is smaller than a predefined threshold \( K_{\text{thres}} \). Second, for track candidates to be connected, they should not lie farther apart than \( O_{\text{max}} \) slices. The threshold \( K_{\text{thres}} \) is set to the a-priori known maximum curvature of the track. The maximum distance between track candidates is chosen to be \( O_{\text{max}} = 6 \) slices, which bounds the maximum size of an occlusion in the bird’s-eye view image. All segments leading to exceeding curvature and occlusion constraints are not used for reconstruction.

Let us now explain how the curvature is computed, in order to elaborate on the curvature constraint. A cubic Bezier curve, determined by control points \( B_{ij} \), is used to compute the maximum curvature between vertices of track candidate \( V_i \) and \( V_j \), located at \( P_i = (x_i,y_i) \) and \( P_j = (x_j,y_j) \) with angle \( \alpha_{\text{track},i} \) and \( \alpha_{\text{track},j} \), respectively. Control points lie at the coordinates of both track candidates \( P_i \) and \( P_j \). The other two control points are defined at \( \frac{1}{3} \) of the distance between the track candidates in the direction of each track candidate (see the figure below). This ensures that the Bezier curvature starts and ends at a track candidate in the direction where the track candidate is detected, and therefore describes a smooth fit. The above control points are defined as follows:

\[
B_{ij} = \begin{bmatrix}
B_0 \\
B_1 \\
B_2 \\
B_3
\end{bmatrix} = \begin{bmatrix}
P_i \\
P_i \\
P_j \\
P_j
\end{bmatrix} + \frac{1}{3}(y_j - y_i) \begin{bmatrix}
0 \\
\alpha_i \\
-\alpha_j \\
0
\end{bmatrix},
\]

where \( B_{ij} \) contains the Bezier control points for the cubic Bezier curve \( C_{ij}(t) \) from candidate \( P_i \) to \( P_j \). The curvature \( K_{ij}(t) \) of an arbitrary-speed curve \( C_{ij}(t) \) is defined as (see ref. [13] Section II.4) follows

\[
K_{ij}(t) = \frac{|x'(t)y''(t) - y'(t)x''(t)|}{(x'(t)^2 + y'(t)^2)^{\frac{3}{2}}},
\]

where \( C'_{ij}(t) = [x'(t), y'(t)] \) and \( C''_{ij}(t) = [x''(t), y''(t)] \) are the first and second derivatives of \( C_{ij}(t) = [x(t), y(t)] \).

Besides the decision if an edge \( E_{ij} \) forms a feasible connection, the maximum curvature is also used to define a weight \( w_{ij} \) for each edge in combination with the vertical distance between the track candidates. The vertical distance between the track candidates is taken into account, because longer curves tend to have a smoother fit, hence their maximum curvature is smaller. This vertical distance in incorporated in the weight \( w_{ij} \) between track candidate \( V_i \) in slice \( k \) and track candidate \( V_j \) in slice \( l > k \), and is given by

\[
w_{i,j} = 1 - \frac{l - k}{O_{\text{max}}} - \max(K_{ij}(t)).
\]

A higher weight represents a better connection. Track candidates which lie close to each other always have higher weights than track candidates farther apart.

### 4.2 Maximum-cost arborescence graph

A max-cost arborescence graph of a general directed graph \( G(V,E) \) with root \( r \) should be a spanning tree of \( G \) in which all points are connected to a neighboring point and where the direction of the edges is ignored. The conditions for such a spanning tree are that (1) it contains a directed path from \( r \) to each vertex in \( V \setminus \{r\} \) and (2) the sum of all weights is maximal.
Since each existing path in the graph represents a feasible combination of track candidates, it is not trivial to directly extract a single track of interest. Many duplicate paths exist in which track candidates are skipped. For example, a considered path $P_{ij}$ between vertex $V_i$ and vertex $V_j$ can be removed if it has an accumulated weight $w_{i,j}$ that is smaller than any other path $P = \{P_{ik}, P_{kl}, P_{lj}\}$ of which the sum of the individual weights along the path $P_{ij}$ is larger than the accumulation of $w_{i,j}$. This way, a max-cost arborescence graph is obtained.

This definition of a max-cost arborescence graph is similar to the minimum spanning tree problem which concerns undirected graphs. However, our graph contains directed weighted edges as the track always grows forward from the root of the graph. Here, the Chu-Liu/Edmonds' Algorithm is used to find a max-cost arborescence graph. An example of such a max-cost arborescence graph is shown in Fig. 4d.

4.3 Selection of the most feasible track

This section describes the method for selecting the most likely driving trajectory, i.e. the most likely path in the graph. The correct path in the max-cost arborescence graph is found by computing the maximum a-posteriori (MAP) estimate. Here, the likelihood that a path is the path of interest is combined with prior knowledge about the location of this path, as defined by

$$\text{MAP} = \arg \max_i P(\text{track} | i) P(i).$$

Here, the likelihood of a path is based on the the number of track candidates in a path and the distance that this path covers. A path which contains more track candidates is more likely to be the path of interest. The same holds for the distance spanned by the path, which is more robust to occlusions. Since the image is divided in 16 slices, both the number of track candidates $n$ in a path with index $i$ and the distance $d$ spanned by the path range from 1 to 16. The likelihood $P(i)$ that path $i$ is the path of interest is defined by

$$P(i) = \beta \frac{n}{16} + (\beta - 1) \frac{d}{16},$$

in which $\beta$ is a weighting parameter.

The prior $P(\text{track} | i)$, the initial probability that $i$ is the correct track, is defined by a normal distribution $N(\sigma, \mu)$, where $\sigma$ and $\mu$ are derived from the positions of all the annotated tracks. This distribution describes the possibility that a correct track is located at a horizontal position $x$ in the first slice. The prior is updated by using temporal information. Here, the last 2 track positions are used to create an estimate about the next starting position of the track. This way, the track of interest can be found in branches and curves where the prior, which assumes driving straight ahead, is not satisfied. The horizontal speed of the track at the bottom of the image is assumed constant. Let us denote the last two track positions by $x_{-1}$ and $x_{-2}$, then the estimate for the track position in the next frame $\hat{x}_0$ and the prior are computed by

$$\hat{x}_0 = x_{-1} + (x_{-1} - x_{-2}),$$

$$P(\text{track} | i) = \gamma N(\sigma, \mu) + (1 - \gamma) N(\sigma_{\text{est}}, \hat{x}_0)$$

in which $0 \leq \gamma \leq 1$ determines the weight of the temporal information. Furthermore, $\sigma_{\text{est}}$ is derived from the ground truth.

To avoid the situation where an incorrectly detected track propagates through the prior, e.g. when two tracks are in parallel and the wrong track is favored by the updated prior, the prior is reset to the default distribution (see Eq. (14)) when the following conditions hold. If the horizontal position of the found track is constant over 5 frames and does not satisfy the ground-truth prior denoting driving straight ahead, it is assumed that an incorrect parallel track is found. In such a case, the prior is reset to go back to the track of interest. This occurs most frequently when leaving a curved track section.
Table 1. Dataset size and description used for validation

<table>
<thead>
<tr>
<th>Description</th>
<th>Straight track</th>
<th>Curved track</th>
<th>Occlusions</th>
<th>Complete recordings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset Frames</td>
<td>2,044</td>
<td>362</td>
<td>124</td>
<td>3,685</td>
</tr>
</tbody>
</table>

5. EXPERIMENTS AND RESULTS

5.1 Annotations & datasets

Multiple videos are acquired using a camera mounted on top of a tram, while driving through the city center of Amsterdam, the Netherlands. These videos contain various scenarios where accidents are likely to occur. All videos have a resolution of 1920 × 1440 pixels at 2 fps. Two videos have been manually annotated for validation of the system, where in each frame the left and right rails of the track, on which the tram is driving, are marked. Each annotated rail is represented by a spline by manually specifying several control points on the rail. These annotations, which implicitly contain the orientation of the track, are used to divide the videos into 4 datasets.

The first dataset contains images with straight tracks, in which no occlusions are present. A straight track is defined as the complete track at an angle of 10 degrees or less with respect to the vertical axis. This angle can be retrieved from the annotations. The second dataset contains the images with curved tracks. These are all images without occlusions, which are not in the first dataset and in which the curved track is actually visible, e.g. the track lies inside the field of view of the camera. The third dataset only contains images in which occlusions are present, meaning that a part of either left, right or both rails is not visible. The last dataset represents the complete videos with all difficult aspects, to validate the proposed system for practical cases. A short summary of the datasets is also shown in Table 1.

5.2 Performance criteria

In order to evaluate the detected track candidates, the rail annotations are transformed to ground-truth track candidates in the bird’s-eye view. This ensures that the distance between detected and annotated tracks (in pixels) is directly proportional to the actual distance, and not influenced by the perspective. First, the annotated rails are mapped to the bird’s-eye view. Next, the center line of the annotated track is found. The orientation and position in each of the 16 slices is then determined by least-mean-squares line fitting. The resulting positions and orientations form the annotated track candidates and can be compared directly to the track candidates found in Section 3.

A detection is considered a true positive (TP) if it is located within 15 cm and its orientation is within 5 degrees of an annotated track candidate. These values are based on empirical experiments and on the desire of a sufficiently accurate outcome. On the other hand, if no detections are found that satisfy these criteria, the track candidate is counted as a false negative (FN), hence a missed track candidate. False positives (FPs) are defined as detections that are not corresponding to any annotation.

The performance of the system is expressed by the well-known recall and precision metrics. Recall denotes the percentage of annotated track candidates found, while precision denotes the percentage of the found detections that are correct. More formally, these metrics are expressed as

$$\text{Recall} = \frac{TP}{TP + FN}, \quad \text{Precision} = \frac{TP}{TP + FP}. \quad (15)$$

Here, it should be noted that only the track on which the tram is driving is annotated. Therefore, detections corresponding to parallel tracks are counted as FPs, which causes a low precision (this approach is a worst-case consideration and also practical for annotation).

Apart from the above metrics to evaluate track candidates, another measure is used to validate the complete track (e.g. a path of interest in the max-cost arborescence graph as described in Section 4.3). An evaluated track is classified as Completely Found (CF), Partially Found (PF) or Not Found (NF). We have defined that CF
means that only 2 out of 16 track candidates can have a distance larger than 15 cm from the annotated track. This means that CF is chosen as an outcome when 87.5% of the track candidates corresponding to a track are classified as TP and the track spans a distance of at least 75% of the distance spanned by the annotated track. A track is classified as PF, if at least 30% of the annotated track candidates are correctly detected. If a track is neither classified as CF nor as PF, then it is classified as NF, meaning that less than 30% of the track is found.

5.3 Types of experiments
The experiments summarized below are split in three tests to incrementally measure the system performance.

5.3.1 Parameter estimation
In this experiment, the RANSAC parameters are evaluated using the Recall and Precision of the track candidates. These are used to find a trade-off between speed and performance, e.g. the number of iterations $I$, the minimum score for a line $s_{min}$ and the number of models $M$. Here, the maximum deviation in track width is set to $w_{max} = 100$ pixels (i.e. 30 cm) and the maximum deviation in angle between rail candidates is set to $\alpha_{max} = 10$ degrees. Furthermore, in a second experiment, the maximum deviation $w_{max}$ from the track width $w_t$ and the maximum deviation in angle $\alpha_{max}$ are evaluated using a Recall-Precision graph for different combinations of these two parameters. Both experiments are only carried out on Dataset 1.

5.3.2 Comparison
The system is then compared to the system proposed by Ross. Since the latter system is not able to find the track of interest, we only evaluate the system up to the track reconstruction. Therefore, for both systems, we evaluate if the annotated track per frame is correctly found, leading to the CF, PF and NF scores.

Minor modifications have been made to the method of Ross, such that it can cope with both light and dark lines, representing the rail in an urban environment, sometimes positioned somewhat lower than the road surface. This is implemented by applying the filter of Ross to both the image and the negative image.

5.3.3 Complete system
Finally, the complete system performance is measured, including the selection of the correct track. Track reconstruction, selection of the correct track and the temporal tracking are applied to all datasets, using the parameters for track detection found in the first experiment. All results are shown in the form of the average scores per frame, yielding the CF, PF and NF scores.

5.4 Experimental results
This section presents the results of the experiments described in Section 5.3.

5.4.1 Parameter estimation
Fig. 5a shows the Recall for different RANSAC iterations. Configurations with 64 models perform slightly better than configurations with 32 models. However, these configurations with 64 models have a lower Precision as shown in Fig. 5b. On the other hand, configurations with 16 models show a significantly lower Recall than other configurations (a 10% drop). Increasing the minimum score $s_{min}$ positively affects the Precision, while deteriorating the Recall, especially for a lower number of iterations.

Given the high Recall score, the detection of possible track candidates proves to be reliable. The configuration which searches for $M = 32$ models, with $I = 16$ iterations per model with a minimum score $s_{min} = 0.3$ is chosen as a suitable trade-off between computational complexity and a high Recall (as denoted by the diamond marker in Fig. 5a). The Precision of this configuration is low, however, this is inevitable as all (parallel) tracks are detected and we are interested in only one. Moreover, the Precision is further increased in the track reconstruction step, where the proper track is selected.

Fig. 6 shows the Recall-Precision plot for different values of $w_{max}$ and $\alpha_{max}$, using the trade-off parameters as mentioned previously. The Recall increases with a higher maximum track width, although it also reduces the Precision. The same holds for $\alpha_{max}$. However, when $\alpha_{max}$ exceeds 4 degrees and/or the maximum width exceeds 90 pixels (27 cm), the Recall only slightly improves while the Precision significantly decreases. Therefore, $w_{max} = 90$ and $\alpha_{max} = 4$ are chosen.
Table 2 shows the results obtained for both the proposed system and the system of Ross. It can be observed that the proposed system performs significantly better. In the proposed system, the annotated track is found in 96% of the images containing straight tracks, compared to 47% by the system of Ross. Moreover, in the proposed system, even the remaining 4% is partially found. A decrease in performance is shown for the ‘curved track’ dataset, in which 57% of the curved tracks are completely found and 28% of the curved tracks are partially found. The system of Ross shows a similar drop in the detection rate for curved tracks. In the dataset containing occlusions, only a small percentage of the tracks are not found by the proposed system. In general, an annotated track is not detected in only 9% of the cases. This compares favorably to Ross, where 23% is not found.

It should be mentioned that in the dataset containing occlusions, a horse carriage riding in front of the tram for over 15 frames causes the track to be partially found or even not found (see Fig. 7k).
Table 2. Track reconstruction rate comparison

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Proposed system</th>
<th>Ros[@]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CF</td>
<td>PF</td>
</tr>
<tr>
<td>1</td>
<td>0.96</td>
<td>0.04</td>
</tr>
<tr>
<td>2</td>
<td>0.57</td>
<td>0.28</td>
</tr>
<tr>
<td>3</td>
<td>0.71</td>
<td>0.15</td>
</tr>
<tr>
<td>4</td>
<td>0.78</td>
<td>0.13</td>
</tr>
</tbody>
</table>

Table 3. Track selection rates

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Proposed system</th>
<th>Ros[@]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CF</td>
<td>PF</td>
</tr>
<tr>
<td>1</td>
<td>0.91</td>
<td>0.07</td>
</tr>
<tr>
<td>2</td>
<td>0.35</td>
<td>0.27</td>
</tr>
<tr>
<td>3</td>
<td>0.55</td>
<td>0.19</td>
</tr>
<tr>
<td>4</td>
<td>0.67</td>
<td>0.14</td>
</tr>
</tbody>
</table>

5.4.3 Complete system

Table 3 shows the final results of the complete system, indicating whether the system is able to select the correct track on a frame-by-frame basis.

In the dataset containing straight tracks, the correct track is found and automatically selected by the system in approximately 91% of the images, while in around 96% of the images the completely found track is available in the max-cost arborescence graph (see Table 2). The 4% that is not present in the max-cost arborescence graph is most likely caused by a poor contrast of the rail, or by strong rail-like structures in the close vicinity of the track (Fig. 7d).

The track selection rate for curved track segments is significantly lower compared to straight track segments. Here, approximately 35% of the curved tracks is completely found and about 27% is partially found, compared to 57% and 28% in the max-cost arborescence graph (see Table 2), respectively. Curved tracks are more difficult to detect for three reasons. First, curved rails are less visible at farther distance, as the rails are embedded in the road surface. Second, the field of view of the camera is limited, which causes the first 8 meters of the track to be invisible. Hence, the tracks are only partially visible in a curve, or not at all. Third, the rail filter is optimized for straight rails, hence curved rails lead to a lower response (Fig. 7h). A significant part of incorrect detections is caused by tracks moving out of the camera field of view. The reason for this is the implementation of the tracking in the proposed system, which assumes that there is always a track present. Ideally, the algorithm detects when the track of interest leaves the field of view. This aspect is not considered in this paper.

In 55% of the frames with occlusions, the track can still be completely found. Occlusions are handled reliably under ideal circumstances, e.g. only small or no curvature in the track occur and only small occlusions appear, such as pedestrians or cyclists (Fig. 7j). Larger occlusions like vehicles pose a problem as they occlude a significant part of the track (Fig. 7l), which makes track reconstruction impossible in some cases. In general, occlusions form a challenging problem, because they mainly occur at crossings and crosswalks, which are typically accompanied by clutter and additional line structures (Fig. 7).

5.5 Discussion & future work

The work of Ross is normally applied to railway tracks. The comparison with the proposed system shows that some assumptions made for railway-track detection are not valid in an urban environment in which trams operate. While occlusions rarely occur on railway tracks, they are common for trams that typically share the city road with other traffic participants. Although not yet perfect, the proposed system can handle the complex shapes of the tramway track in a better way, both with and without occlusions.

Experiments have show that the majority of partially found tracks are caused by a poor contrast between the rails and the background. Although the rails are present after applying our custom rail filter, the filter response is typically much lower than that of lane markings. This sometimes causes the adaptive thresholding to reject the rails. The lower response for curved tracks can be improved by rotating the rail filter to the correct angle to increase the filter response. This decision can be made based on previous detections, the track location at the bottom part of the image and/or prior information about the tram network. Occlusion handling can also be improved by using prior information about the tram network, such that an estimation can be made about the position of the track. Another option is to apply obstacle detection, e.g. using a stereo camera set-up or laser scanner, such that occluded track positions are known in advance and can be compensated.
Figure 7. Images in which straight (a-d), curved (e-h) and occluded (i-l) track is completely found (CF) (second column) and not found (NF) (fourth column). The corresponding original images are shown in the first and third column.

Finally, the current implementation of the system supports track detection up to a distance of 45 meters. This range will be extended in future work, for example by adaptively changing the number of parts in the bird’s-eye view, according to the shape of the track. Another shortcoming is that our system cannot handle the track moving out of the camera view in sharp curves. This can be solved by aiming the camera lower, such that the track is always visible at the bottom of the camera frame, albeit that this will prevent the system to look farther away.

6. CONCLUSIONS
We have proposed a vision-based approach for tramway-track detection using a monocular camera. This is a challenging task due to the surrounding busy environment involving clutter, sharp curves in the track and appearing occlusions. We have found that existing techniques and algorithms, such as applied to railway detection for trains, are not suited for such challenging conditions.

We have designed a new algorithm, featuring the following important elements. First, our system transforms the image to a bird’s-eye view to remove perspective distortion and to obtain a normalized view of the track configuration. Second, the normalized view is split in 16 horizontal slices, so that occlusions have only a local influence. Third, the application of graphs enables a smooth track reconstruction and helps in reducing the influence of occlusions. The vertices and edges of the connection graph are used to evaluate feasible connections. This graph is converted to a max-cost arborescence graph, which contains the most feasible reconstructed tracks. The maximum a-posteriori estimate is used to select the best path.

The system is extensively evaluated on more than 3,600 frames, where each frame was manually annotated. The system performance shows promising results on detecting straight track segments, where over 90% of the
annotated tracks are completely found and 7% partially. Only in 4% of the images, the wrong track is chosen, e.g. a parallel track. Curved tracks still pose some challenges, where 60% of the curved track segments are completely or partially found. Occlusion handling is useful but is limited to straight tracks and only to occlusions caused by medium-sized objects, such as pedestrians/cyclists.

On the basis of our research, we conclude that the proposed method for tramway-track detection offers an attractive solution for use in an urban environment, but its robustness needs to be further improved, especially in curves, by for example further exploiting the existing tram-network layout.

REFERENCES