2010 IEEE Intelligent Vehicles Symposium
University of California, San Diego, CA, USA
June 21-24, 2010

A Voting Strategy for Visual Ego-Motion from Stereo

Štěpán Obdržálek and Jiří Matas
Center for Machine Perception, Czech Technical University Prague
xobdrzal@fel.cvut.cz, matas@fel.cvut.cz

Abstract— **We present a procedure for egomotion estimation from visual input of a stereo pair of video cameras. The 3D egomotion problem, which has six degrees of freedom in general, is simplified to four dimensions and further decomposed to two two-dimensional subproblems. The decomposition allows us to use a voting strategy to identify the most probable solution, avoiding the random sampling (RANSAC) or other approximation techniques.**

The input constitutes of image correspondences between consecutive stereo pairs, *i.e.* feature points do not need to be tracked over time. The experiments show that even if a trajectory is put together as a simple concatenation of frame-to-frame increments, it comes out reliable and precise.

I. Introduction

This paper concerns estimation of egomotion of a vehicle carrying a stereo pair of video cameras. The problem is well studied in the literature [1], [2], [3], [4]. Our situation differs from the majority of the published work in two key aspects. First, the intended use is in urban scenes with the possibility of a heavy traffic. A large part, even a majority, of the field of view can be covered by moving objects, which distract the egomotion estimate. Second, the vehicle moves in an open space, where the distance to the observed objects is large compared to the baseline of the stereo pair. This results in very imprecise 3D triangulation, with spatial uncertainty of triangulated points in tens of meters. This is different to navigation in small closed environments, *e.g.* in laboratories or corridors, where the triangulation errors are smaller.

Our intended application is a detection of moving objects around the vehicle, respectively estimation of motion of the objects in the world coordinate frame. We are thus interested in reliable and precise egomotion estimation, but only locally, within a short time span. The problem of obtaining a globally correct trajectory is not dealt with, nor is the problem of detecting a repeated visit of a location (drift removal, or loop closing). Solutions to these are found in the literature (*e.g.* [5], [6], [7]). We assume that a GPS system solves these problems in practice.

A 3D motion has six degrees of freedom, three for rotation (change in orientation) and three for translation (change in position). A standard approach to recover the six unknowns is to align three pairs of corresponding triangulated 3D points [8], which however becomes tricky once the triangulation errors are large. Or, if only a single camera is available, the motion can be computed from a correspondence of at least five points, *e.g.* [9].

*The authors were supported by Toyota Motor Corporation and by Czech Government under the reseach program MSM6840770038.

In our task, however, it is not necessary to recover all six parameters. We are interested only in horizontal projection of the motion, as if seen on a map. In that case the motion has only three unknowns. Two for 2D location on the ground plane and one for orientation – the heading (or yaw) angle. We also estimate the pitch angle for a total of four unknowns. Pitch is the vertical angle between the optical axis and the ground plane and is used to compute elevation of observed objects, relative to our vehicle. The remaining two unknowns, computation of which we avoid, are the rotation around the camera optical axis (roll) and the absolute elevation of our vehicle. The rolling we assume being negligible for ground vehicles under normal driving conditions. And the vehicle's absolute elevation is not useful to us.

The egomotion estimation is based on a voting scheme. The four-dimensional problem is decomposed into two two-dimensional subproblems, which makes the voting feasible. The rotation angles (yaw and pitch) are estimated first. The 2D translation is computed in a second voting step, in which each vote explicitly reflects the triangulation imprecision.

II. Egomotion Estimation

The egomotion is computed in the form of increments from one stereo image pair to another. Therefore, only four images are involved in the computation at a time - the current pair and the immediately preceding one. The situation is illustrated in Fig. 1.

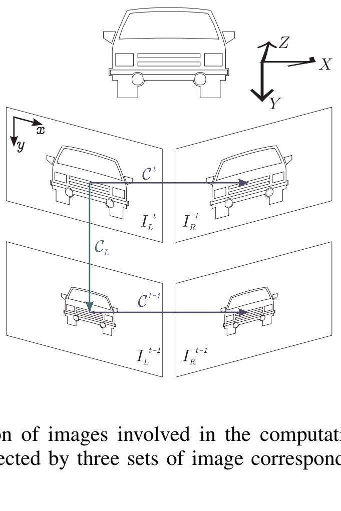

Fig. 1. Illustration of images involved in the computation – two stereo pairs that are connected by three sets of image correspondences.

The figure depict the four images: a current stereo pair taken at time t (images I_L^t and I_R^t) and a preceding one taken at time $t-1$ (images I_L^{t-1} and I_R^{t-1}). Three sets of pixel correspondences are computed. Two sets ($\mathcal{C}^t$ and $\mathcal{C}^{t-1}$: $\{c_i = (x_L, y_L, x_R, y_R)\}$) link pixels in the stereo pairs, the third one ($\mathcal{C}_L$: $\{c_i = (x_L^t, y_L^t, x_L^{t-1}, y_L^{t-1})\}$) connects the two images of the left camera. A 3D scene point X is at time t projected to I_L^t and I_R^t at locations (x_L^t, y_L^t) and (x_R^t, y_R^t). At time $t-1$, it was projected to the preceding image pair to pixels (x_L^{t-1}, y_L^{t-1}) and (x_R^{t-1}, y_R^{t-1}).

Image correspondences are computed with the approach described in [10]. This method gives a semi-dense correspondence map, typically tens of thousands of correspondences are found for a pair of 640×480 images. The camera pair is calibrated, whence the two stereo correspondence sets $\mathcal{C}^t$ and $\mathcal{C}^{t-1}$ can be triangulated, yielding two sets of 3D points $\mathcal{X}^t$ and $\mathcal{X}^{t-1}$ in camera-centric coordinates. The two 3D point sets are connected together by the correspondence set $\mathcal{C}_L$, forming a set of 3D vectors.

A 3D camera motion can be decomposed into two independent components, alignment of directions of the camera axes (camera rotation) and alignment of the camera centers (camera translation). The decomposition is commutative.

A. Estimation of Rotation

The rotation is computed using the 'in-time' correspondence set $\mathcal{C}_L$, containing motion vectors $c_L = (x_L^t, y_L^t, x_L^{t-1}, y_L^{t-1}) \in \mathcal{C}_L$. Let us inspect the effect of the rotation on the motion vectors, assuming for now that the vehicle position is not changing and that the scene is static. Fig. 2 illustrates image motion vectors caused by pitch, yaw and roll components of the 3D rotation (for a camera with spherical projection, which well approximates a perspective camera for narrow fields of view). Assuming zero roll for ground vehicles, the motion vectors caused by rotation are *linear segments that are identical across the image*, independent of object distance.

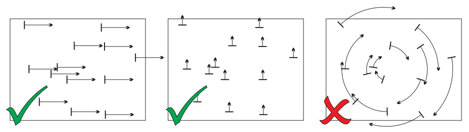
Fig. 2. Image motion vectors due to individual components of 3D rotation. We are interested only in the yaw and pitch, the roll is ignored.

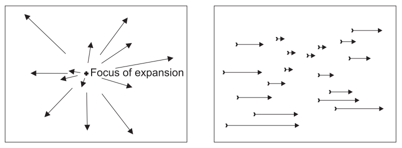

Fig. 3. Image motion vectors due to vehicle translation. Forward motion on the left, sidewise on the right.

If the camera also moves, in addition to rotating, the observed motion field is affected differently. Fig. 3 illustrates the effect. Forward motion produces motion vectors oriented in the direction of the so called focus of expansion, *i.e.* image of the scene point towards which the camera moves. A sidewise motion produces parallel motion vectors similar to the rotation. What is important, in both types of the translation the *length of the motion vectors decreases with distance* to the observed scene point.

The observed motion field is a combination of the translation and rotation components, where the effect of the translation decreases with distance – motion vectors of points at infinity are affected only by the rotation. And the distances are known from the stereo triangulation. This leads us to a very simple voting algorithm for rotation estimation. It is estimated by adding votes to an accumulator, as in the Hough transform. Votes are cast by motion vectors $c_L \in \mathcal{C}_L$ with the weight of the vote being proportional to c_L's 3D distance to the camera. The accumulator domain is in image pixels, its resolution is set to one pixel and its range to $(-\Theta_x, \Theta_x)$ on the x-axis and $(-\Theta_y, \Theta_y)$ on the y-axis. The resolution is given by the precision with which are the image correspondences computed. The bounds on the maximal rotation are set empirically and depend on maximal angular speed and framerate and resolution of the cameras. In our setup we have $\Theta_x = 100$ and $\Theta_y = 50$ pixels, which cover all realistic situations with a large margin.

The procedure is summarised in Algorithm 1. At the end we identify the rotation vector $r = (r_x, r_y)$, in pixels, which has the largest support by the motion vectors. The precision of the estimate is further improved (on the x-axis only) by fitting a quadratic curve to its neighbouring support values in the accumulator, *i.e.* to $\mathcal{A}(r_x - 1, r_y)$, $\mathcal{A}(r_x, r_y)$ and $\mathcal{A}(r_x + 1, r_y)$. Position of the maximum on the parabola is found in a closed form, and it gives us the rotation vector r with a sub-pixel precision.

A final step is to convert the pixel-based vector r to yaw (ψ) and pitch (θ) angles. As shown in Fig. 4, the angle is the inverse tangent of the vector length multiplied by the pixel size p and divided by the focal length f:

$$\psi = \tan^{-1}(\frac{r_x p_x}{f}), \qquad \theta = \tan^{-1}(\frac{r_y p_y}{f}),$$

where p_x and p_y are horizontal and vertical pixel dimensions, in millimeters. Conveniently, in the standard representation of intrinsic camera parameters by an upper triangular 3×3 matrix K [11], the $\frac{f}{p_x}$ and $\frac{f}{p_y}$ ratios are found on its first two diagonal elements. We can therefore write

$$\psi = \tan^{-1}(\frac{r_x}{K_{1,1}}), \qquad \theta = \tan^{-1}(\frac{r_y}{K_{2,2}}).$$

Naturally, the system would be fooled if the field of view is obstructed by a large moving object, *e.g.* a truck pasing close in front of the vehicle. These situations can be detected, as the 3D depths are known, and failure in the estimation can be reported. We have not implemented such a detection though, and, as shown in the experiments, failures of this kind occur.

B. Estimation of Translation

We had two sets of triangulated 3D points, $\mathcal{X}^t$ and $\mathcal{X}^{t-1}$, observed in two consecutive views. The points were in camera-centric coordinates, *i.e.* the origins of coordinate systems coincided with the cameras.

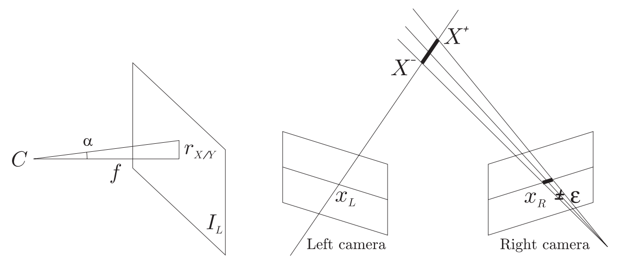

Fig. 4. Left: Relation between a rotation vector r, in image coordinates, and the angle of rotation. Right: Estimation of 3D reconstruction tolerance.

Algorithm 1 Rotation by voting

```
Input: $\mathcal{C}_L$: correspondences between points of two consecutive
    images from one of the cameras
Input: $\mathcal{C}^t$: correspondences between points of the stereo image pair
Output: $r$: vector of rotation, in pixels
/*Initialise the accumulator*/
    $\mathcal{A}_{\Delta x,\Delta y} := 0,\ \Delta x \in (-\Theta_x, \Theta_x),\ \Delta y \in (-\Theta_y, \Theta_y)$
foreach $c_i^t := (x_{i,L}^t, y_{i,L}^t, x_{i,R}^t, y_{i,R}^t) \in \mathcal{C}^t$,
        $c_{j,L} := (x_{j,L}^t, y_{j,L}^t, x_{j,L}^{t-1}, y_{j,L}^{t-1}) \in \mathcal{C}_L$
where $x_{i,L}^t = x_{j,L}^t$ and $y_{i,L}^t = y_{j,L}^t$ do
        /*X: a 3D point in camera-centric coordinates*/
        $X := \mathrm{triangulate}(x_{i,L}^t, y_{i,L}^t, x_{i,R}^t, y_{i,R}^t)$
        /*d: distance between X and the camera*/
        $d := ||X, \bar{0}||$
        /*vote for the rotation, weighted by distance d*/
        $\Delta x := x_{j,L}^{t-1} - x_{j,L}^t$
        $\Delta y := y_{j,L}^{t-1} - y_{j,L}^t$
        $\mathcal{A}_{\Delta x,\Delta y} := \mathcal{A}_{\Delta x,\Delta y} + d$
end
/*find where the maximum is*/
$r := (r_x, r_y) := \operatorname*{argmax}_{\Delta x \in (-\Theta_x, \Theta_x),\, \Delta y \in (-\Theta_y, \Theta_y)} \mathcal{A}_{\Delta x,\Delta y}$
```

We are looking for a coordinate system transformation that will align the scene points while moving the cameras accordingly. Knowing the rotation between the two views, $\mathcal{X}^t$ is rotated around the origin (camera) by θ and ψ. See Fig. 5 for an illustration. After that, the transformation from $\mathcal{X}^{t-1}$ to $\mathcal{X}^t$ is a translation s.

What complicates its identification are great imprecisions in the triangulated coordinates and errors in the established correspondences. In the following we search for a translation vector s that would best explain the difference between the two point sets, given the triangulation errors. Again, a voting scheme is adopted in order to be robust to mismatches in the correspondence sets.

C. Triangulation Uncertainty

A 3D point X is a measurement given by a stereo correspondence $c_i^t = (x_{i,L}^t, y_{i,L}^t, x_{i,R}^t, y_{i,R}^t) \in \mathcal{C}^t$, with uncertainty increasing with distance to the object. The uncertainty is a function of imprecision in the correspondence (in pixels), of camera resolution and calibration, of the disparity, pixel's position in image, and generally of the image content (*e.g.* there may be a smaller uncertainty in higher contrast areas).

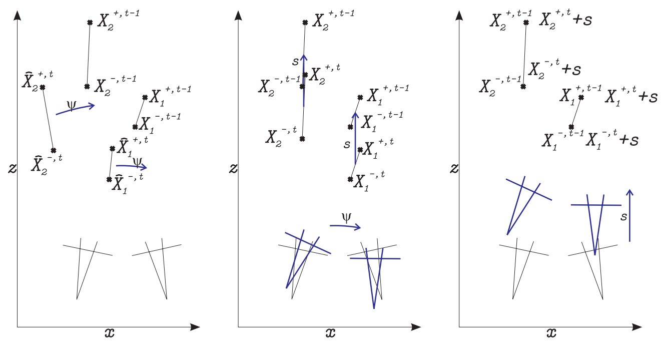

Fig. 5. Two-stage egomotion estimation. The rotation (yaw and pitch) is computed first, followed by the translation.

Let us assume that the images are rectified, *i.e.* that for any stereo correspondence $c_i^t = (x_{i,L}^t, y_{i,L}^t, x_{i,R}^t, y_{i,R}^t) \in \mathcal{C}^t$ it holds that $y_{i,L}^t = y_{i,R}^t$. The correspondence of $(x_{i,L}^t, y_{i,L}^t)$ is then given by a single number, the disparity $d_i = x_{i,R}^t - x_{i,L}^t$. Let the disparities be computed with a tolerance of ϵ, say $\epsilon = 1$ pixel. *I.e.* if a correspondence with a disparity $\hat{d}$ was established, the actual disparity is considered to be $d \in (\hat{d} - \epsilon, \hat{d} + \epsilon)$ with a uniform distribution over the interval.

The pixel-wise tolerance is transformed to the 3D by triangulating both ends of the interval, *i.e.* both $(x_{i,L}^t, y_{i,L}^t, x_{i,R}^t - \epsilon, y_{i,R}^t)$ and $(x_{i,L}^t, y_{i,L}^t, x_{i,R}^t + \epsilon, y_{i,R}^t)$. See Fig. 4 for an illustration. This gives us two endpoints of a 3D line segment on which the scene point X_i is located with distribution approximately again uniform (the distribution is in fact a piece of a quadratic function, since the triangulation errors grow quadratically with the disparity).

The segment goes in the direction to the reference (left) camera and its length increases with the distance, reflecting the higher uncertainty of more distant depth measurements. Table I shows the uncertainty of our stereo configuration, tabulated for some typical distances.

There are other forms of triangulation imprecisions, coming from imprecise calibration of the stereo pair, but their magnitude is significantly smaller. They are all together modelled as Gaussian and we treat them later.

Distance to the object	Uncertainty of 3D triangulation
5 m	±6 cm
10 m	±22 cm
15 m	±49 cm
20 m	±88 cm
30 m	±200 cm
50 m	±555 cm
80 m	±1600 cm
100 m	±2500 cm

TABLE I
3D TRIANGULATION UNCERTAINTY FOR IMAGE CORRESPONDENCES WITH TOLERANCE ±1 PIXEL.

D. Translation by voting

Fig. 5 illustrates the two-step egomotion recovery from a top view. Two scene points, X_1 and X_2, are shown as their respective tolerance segments. We denote as X^- the closer end of the tolerance segment, obtained as triangulation of $(x_L, y_L, x_R - \epsilon, y_L^t)$, and as X^+ the farther end, of $(x_L, y_L, x_R + \epsilon, y_L^t)$.

In the figure on the left, the points are in camera-centric coordinates, as triangulated from the two stereo pairs. The middle figure shows the situation after rotation by the estimated yaw ψ was applied to the points from the current t-th frame. Finally, on the right, the points from the t-th frame are aligned with their counterparts from the $(t-1)$-th frame by the translation vector s, yet unknown.

Fig. 6 shows what the vector s can be, *i.e.* how can we move from X^{t-1} in the previous frame, represented by tolerance segment $\overline{X^{-,t-1}X^{+,t-1}}$, to X^t, represented by segment $\overline{X^{-,t}X^{+,t}}$, in the current frame. All possible translation vectors form (in the 2D top view projection) a tetragon shown in the middle of the figure. The coordinates of its vertices are the differences between the tolerance segment endpoints: $X^{-,t-1} - X^{+,t}$, $X^{-,t-1} - X^{-,t}$, $X^{+,t-1} - X^{+,t}$ and $X^{+,t-1} - X^{-,t}$. This tetragon represents the vote that X casts into the accumulator.

Under the assumption that the distance to the point X does not change much between the frames, *i.e.* that it is relatively larger than the length of the translation vector s, the tolerance segments do not change significantly. We can assume them identical, *i.e.* $X^{+,t-1} - X^{-,t-1} = X^{+,t} - X^{-,t}$. In that case, the vote degenerates to a line segment $\overline{(X^{-,t-1} - X^{+,t})\,(X^{+,t-1} - X^{-,t})}$, as shown on the right side of Fig. 6.

The voting procedure is resumed in Algorithm 2. An accumulator of votes is initialised first, its domain being the translations in world coordinates. We have its resolution set to 1 mm, its range for left-right offset $\Theta_X^{\min} = -200$ mm and $\Theta_X^{\max} = 200$ mm and its backward-forward range $\Theta_Z^{\min} = -500$ mm and $\Theta_Z^{\max} = 1500$ mm. Then, each point X that was successfully triangulated in both t-th and $(t-1)$-th frame adds a vote in the form of the top-view projected 2D line segment.

As a final step, the accumulator is convolved with a kernel of 2D normal distribution, with deviation σ appropriate to cover all the other imprecisions in the triangulation. We have $\sigma = 5$ mm. A position of the maximum in the convolved accumulator is then found as the translation vector s. Fig. 7 shows examples of the accumulated votes. Note that a typical length of a vote is, in the world coordinates, *in the order of meters* or tens of meters. Yet, as shown in the experiments, the maximum can be *localised with a precision of few millimeters*.

The computational cost of the procedure is low once the correspondences were obtained. Since the correspondences are discretised in the pixel domain, the triangulation in camera-centric coordinates can be implemented as a table look-up. The voting itself requires rendering of line segments, which, if implemented on graphics hardware, is almost instantaneous. The only remaining non-trivial operations relate to the accumulator management – initialisation, convolution with a Gaussian kernel and the maximum search – which are all fast and easily parallelisable.

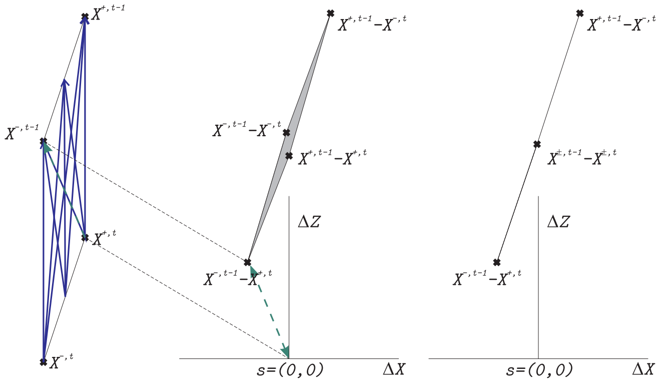

Fig. 6. Shape of the translation vote. See text for explanation.

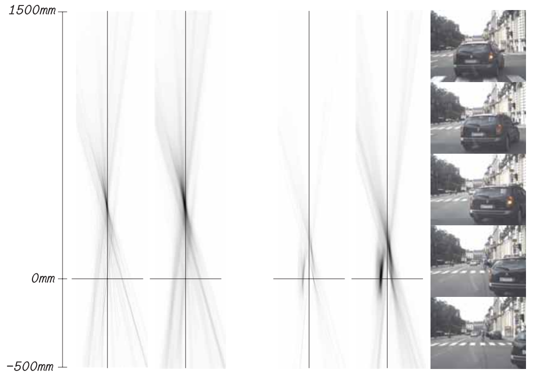

Fig. 7. Estimation of translation: two examples of the voting accumulators, each showing the accumulator $\mathcal{A}$ and its $\mathcal{N}(0, \sigma)$ smoothed variant $\overline{\mathcal{A}}$. Both cases represent an almost forward motion. The right one is at a lower speed and there was another motion candidate, caused by a car in front of our vehicle going at about the same speed and turning to the right. The illusory motion is therefore to the left, with no forward component. The coordinate lines intersect at $s = (0, 0)$.

III. Experiments

The approach was tested on sequences that were taken with a stereo camera pair mounted on a vehicle driven through a city. The sequences, each several thousands of images long, represent real-world scenarios. They include sections of high-speed driving on an expressway as well as traffic congestions and a drive through a city centre with pedestrian-crowded alleyways. Sample images are shown in Fig. 8.

Fig. 8. Sample frames from sequences used to test the egomotion estimation. From open areas to crowded alleyways.

The egomotion was computed on frame-to-frame basis. An update to the orientation and location was calculated from one frame to the immediately following one, never considering preceding images. Therefore, the trajectories presented here are *concatenations of thousands of increments*.

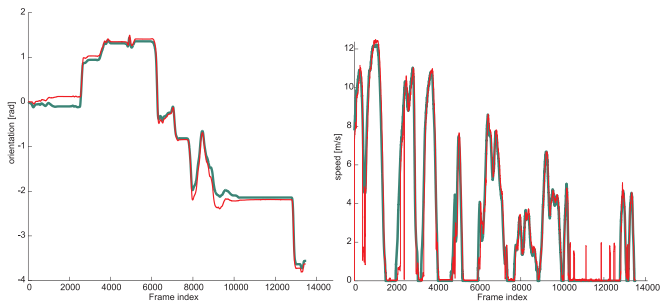

Fig. 10. Comparision of computed (red) and CANbus (green) estimates on the sequence from the left side of Fig. 9. Left: progression of orientation (heading) of the test vehicle. The computed orientation (red) is the cumulative sum of about 14000 yaw angle increments (ψ). Right: Speed of the vehicle. The computed speed (red) is the length of the translation vector s.

Algorithm 2 Translation by voting

```
Input: $\mathcal{C}_L$: correspondences between points of two consecutive
  images from one of the cameras
Input: $\mathcal{C}^t, \mathcal{C}^{t-1}$: correspondences between points of stereo image
  pairs, current and previous frames
Output: $s$: vector of translation, in world coordinates

/*Initialise the accumulator*/
  $\mathcal{A}_{\Delta X, \Delta Z} := 0,$
  $\Delta X \in (\Theta_X^{\min}, \Theta_X^{\max}),\ \Delta Z \in (\Theta_Z^{min}, \Theta_Z^{\max})$
foreach $c_i^t := (x_{i,L}^t, y_{i,L}^t, x_{i,R}^t, y_{i,R}^t) \in \mathcal{C}^t,$
        $c_j^{t-1} := (x_{j,L}^{t-1}, y_{j,L}^{t-1}, x_{j,R}^{t-1}, y_{j,R}^{t-1}) \in \mathcal{C}^{t-1}$
        $c_{k,L} := (x_{k,L}^t, y_{k,L}^t, x_{k,L}^{t-1}, y_{k,L}^{t-1}) \in \mathcal{C}_L$
 where $x_{i,L}^t = x_{k,L}^t$ and $y_{i,L}^t = y_{k,L}^t$ and $x_{j,L}^{t-1} = x_{k,L}^{t-1}$ and
  $y_{j,L}^{t-1} = y_{k,L}^{t-1}$ do
      /*$\hat{X}^{\pm,t}, X^{\pm,t-1}$: endpoints of 3D tolerance segments in
  camera-centric coordinates*/
      $\hat{X}^{-,t} := \mathrm{triangulate}(\mathrm{x}_{\mathrm{i,L}}^{\mathrm{t}}, \mathrm{y}_{\mathrm{i,L}}^{\mathrm{t}}, \mathrm{x}_{\mathrm{i,R}}^{\mathrm{t}} - \epsilon, \mathrm{y}_{\mathrm{i,R}}^{\mathrm{t}})$
      $\hat{X}^{+,t} := \mathrm{triangulate}(\mathrm{x}_{\mathrm{i,L}}^{\mathrm{t}}, \mathrm{y}_{\mathrm{i,L}}^{\mathrm{t}}, \mathrm{x}_{\mathrm{i,R}}^{\mathrm{t}} + \epsilon, \mathrm{y}_{\mathrm{i,R}}^{\mathrm{t}})$
      $X^{-,t-1} := \mathrm{triangulate}(\mathrm{x}_{\mathrm{j,L}}^{\mathrm{t-1}}, \mathrm{y}_{\mathrm{j,L}}^{\mathrm{t-1}}, \mathrm{x}_{\mathrm{j,R}}^{\mathrm{t-1}} - \epsilon, \mathrm{y}_{\mathrm{j,R}}^{\mathrm{t-1}})$
      $X^{+,t-1} := \mathrm{triangulate}(\mathrm{x}_{\mathrm{j,L}}^{\mathrm{t-1}}, \mathrm{y}_{\mathrm{j,L}}^{\mathrm{t-1}}, \mathrm{x}_{\mathrm{j,R}}^{\mathrm{t-1}} + \epsilon, \mathrm{y}_{\mathrm{j,R}}^{\mathrm{t-1}})$

      /*Rotate $\hat{X}^{\pm,t}$ by $\theta$ and $\psi$*/
      $X^{\pm,t} := \mathbf{R}_{\theta,\psi} \cdot \hat{X}^{\pm,t}$

      /*vote for the translation $s$ with a line segment $\overline{uv}$*/
      $u := (X_X^{-,t-1} - X_X^{+,t}, X_Z^{-,t-1} - X_Z^{+,t})$
      $v := (X_X^{+,t-1} - X_X^{-,t}, X_Z^{+,t-1} - X_Z^{-,t})$
      $\mathrm{addLineSegment}(\mathcal{A}, \overline{uv})$
end

/*add tolerance to other forms of noise*/
 $\overline{\mathcal{A}} := \mathrm{convolve}(\mathcal{A}, \mathcal{N}(0,\sigma))$

/*find where the maximum is*/
 $s := (s_X, s_Z) := \operatorname*{argmax}_{\Delta X \in (\Theta_X^{\min}, \Theta_X^{\max}),\, \Delta Z \in (\Theta_Z^{min}, \Theta_Z^{\max})} \overline{\mathcal{A}}_{\Delta X, \Delta Z}$
```

If an error was made in the computation of an increment, it was not compensated later. Nonetheless, the trajectories are precise, indicating that there were only few mistakes and that no significant errors accumulated over time.

Fig. 9 shows top view of the sequences. The sequence on the left lasted about 8 minutes and consists of about 14000 image pairs taken at 30 frames per second. The figure shows our reconstructed trajectory (yellow) overlaid on a satellite map. The actual path, hand-drawn, is shown in red. For this sequence we also have a record of the in-car data from the CANbus, with speed and turning angle readings. The trajectory restored from the CANbus data is shown in green.

Using the CANbus data as ground-truth, we can separately evaluate rotation and translation estimates. The rotations are shown in left part of Fig. 10. By summing the incremental changes in yaw (ψ) computed at each frame, we obtain the cummulative orientation drawn in red. The green line is for the CANbus orientation, which is again a cumulative sum of per-frame readings. The differences in the graphs are within the precision of camera calibration, which indicates that there is no systematic error in the computation accumulating over time.

Right side of Fig. 10 shows progression of vehicle's speed. At each frame, the actual speed is the length of the translation vector s. Again, our measurements are shown in red while the CANbus data are in green. The graphs correspond well, but there are some mistakes to be seen. Mostly they concern acceleration from a stop at a crossing when there is another car immediatelly in front of us accelerating concurrently. In such cases the visually perceived speed is lower than actual. The most pronounced case can be seen at frames around 2200. Yet the mistakes are only few and their overall effect on the trajectory shown in Fig. 9 is small. In numbers, the difference between vision and CANbus speeds is in 92.5% of the meassurements less than $1m/s$ ($33mm$ for $30fps$), in 79.5% less than $10mm$ and in 55% less than $5mm$.

The second sequence shown in Fig. 9 is longer, lasting almost half an hour, and consisting of about 50000 stereo image pairs. Although the trajectory looks rather like a mess, it is in fact mostly correct at local scale. We start at the bottom right corner and until we reach the topmost part, about 25000 video frames later, the differences are small. There we fail to get the orientation correctly, bending the trajectory by about 45 degrees. The same happens in the leftmost part, resulting in a total difference in orientation of about 90 degrees at the end of the sequence. In both cases, the failure was due to other vehicles passing from left to right very close in front of our car, obscuring most of the field of view (see Fig. 12).

Figure 11 shows in detail parts of the sequences. Trajectory segments are accompanied with representative images from the on-board cameras. The first segment is an over a minute long passage through a detour, with presence of multiple distracting moving objects, but none of them dominant. The second one shows a turnabout maneuver that includes reversing. The orthomap backgrounds under the trajectories were aligned manually.

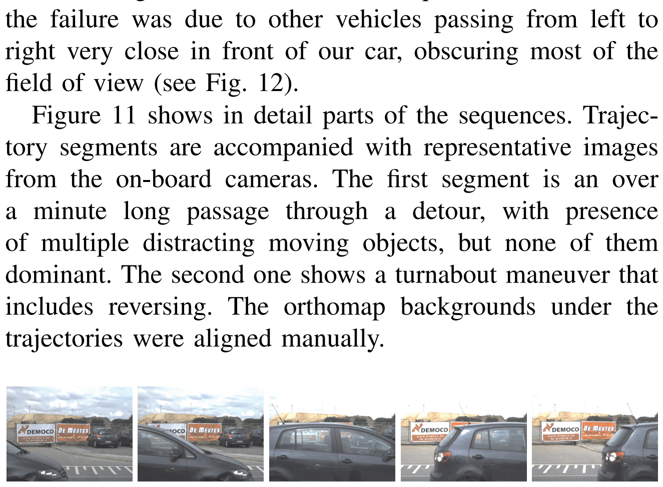

Fig. 12. A situation where we fail to recover the rotation correctly. The car passing in front of us makes for a phantom rotation to the left.

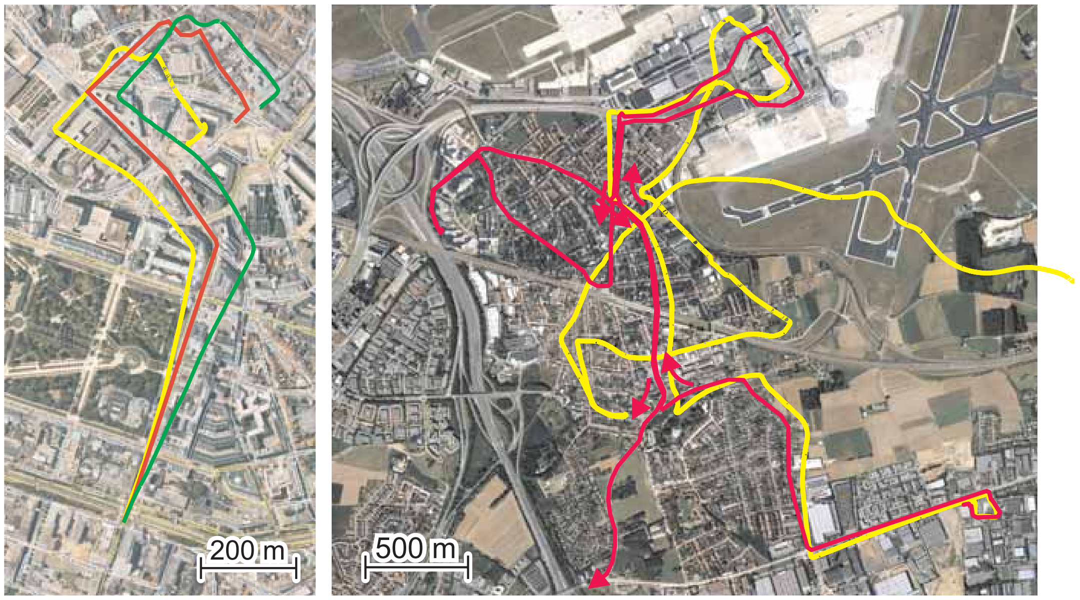

Fig. 9. Comparison of reconstructed trajectories (yellow) with hand-drawn ground-truth (red). For the sequence on the left a trajectory obtained from the vehicle's CANbus data is also shown (green).

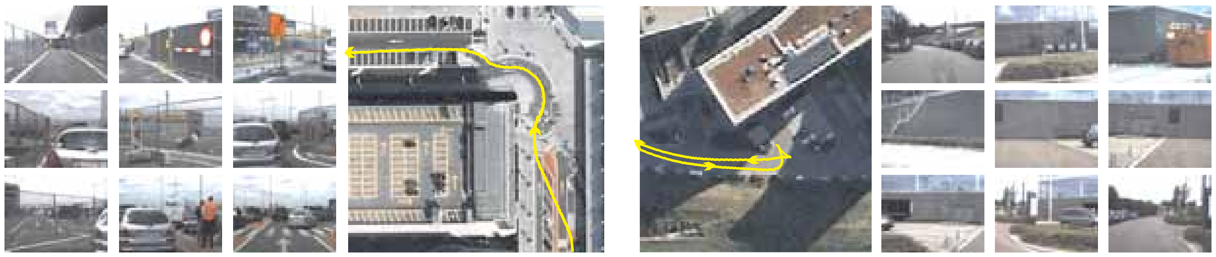

Fig. 11. Two segments of the computed trajectory with corresponding scene images. Left: repeated structures on the fences interfere with the correspondence search process, and other moving objects in the surroundings that create illusions of false egomotion. Right: turnabout maneuver which includes reversing.

IV. Conclusions

We have proposed a solution to the problem of estimation of egomotion from a visual input, if it is acquired with a stereo pair of video cameras. The general 3D motion problem with six unknowns was simplified to four dimensions and further decomposed to two two-dimensional subproblems. The decomposition allowed us to use a voting scheme to reliably identify the most representative egomotion, even when the input data – image correspondences – were noisy.

Experimental evaluation on real-world sequences has shown that although the egomotion was computed in the form of differences between consecutive video frames, the method provides reliable and precise output. The occasional mistakes occur when the visual input is dominated by another object moving in the scene.

A complex egomotion estimation system can be build on top of the proposed procedure. Results of the visual estimator should be combined with other sensors available, *e.g.* accelerometers or the CANbus car controls. Restrictions from a vehicle motion model should be considered, *e.g.* reflecting the minimal turning radius. And corrections at global scale should be obtained using a positioning system (GPS) and/or by any of the vision methods for the long-term drift removal.

References

[1] C. F. Olson, L. H. Matthies, M. Schoppers, and M. W. Maimone, "Rover navigation using stereo ego-motion," *Robotics and Autonomous Systems*, vol. 43, no. 4, pp. 215 – 229, 2003.

[2] T. Lemaire, C. Berger, I.-K. Jung, and S. Lacroix, "Vision-based slam: Stereo and monocular approaches," *Int. J. Comput. Vision*, vol. 74, no. 3, pp. 343–364, 2007.

[3] A. Howard, "Real-time stereo visual odometry for autonomous ground vehicles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 3946–3952.

[4] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry for ground vehicle applications," *Journal of Field Robotics*, vol. 23, 2006.

[5] K. Cornelis, F. Verbiest, and L. Van Gool, "Drift detection and removal for sequential structure from motion algorithms," *IEEE PAMI*, vol. 26, no. 10, pp. 1249–1259, 2004.

[6] T. Thormählen, N. Hasler, M. Wand, and H.-P. Seidel, "Merging of feature tracks for camera motion estimation from video," in *Conference on Visual Media Production*, 2008.

[7] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE PAMI*, vol. 29, 2007.

[8] R. M. Haralick, C.-N. Lee, K. Ottenberg, and M. Nölle, "Review and analysis of solutions of the three point perspective pose estimation problem," *Int. J. Comput. Vision*, vol. 13, no. 3, pp. 331–356, 1994.

[9] D. Nistér, "An efficient solution to the five-point relative pose problem," *IEEE PAMI*, vol. 26, no. 6, pp. 756–777, 2004.

[10] Š. Obdržálek, M. Perďoch, and J. Matas, "Dense linear-time correspondences for tracking," in *Workshop on Visual Localization for Mobile Platforms, CVPR 2008*, June 2008.

[11] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.