Estimation of stereo camera motion

Maciej Polańczyk, Przemysław Barański, Michał Strzelecki

Technical University of Lodz, Institute of Electronics,
e-mail: maciej.polanczyk@gmail.com, przemysław.baranski@p.lodz.pl, mstrzel@p.lodz.pl

The paper presents an algorithm for estimating the motion of a stereovision camera. The movement is calculated on the base of a sequence of depth pictures. While moving the camera can be held in hands without much impact on the prediction accuracy. The algorithm provides real-time data processing when a 2.5 GHz dual core computer is used.

Keywords and phrases: stereovision, motion estimation.

Introduction

The motion estimation algorithms are used in many fields, among others robotics and car industry. GPS readouts are not always available e.g. indoor and underground areas, tunnels and so forth. Moreover, in an urban environment GPS inaccuracies can reach as much as 100 m [1]. Hence, motion estimation is commonly used to supplement poor quality GPS readouts. An accelerometer measures accelerations which can be transformed into displacement by double integration. The errors grow quadratically over time. Encoders mounted on a robot's wheels yield much better distance estimation, providing that the signals from the encoders can be easily accessible, which is not always the case e.g. a car. A wheel can also slip which introduces an error. Therefore these method are combined with other methods like computer vision which has very interesting properties.

The algorithms for estimating the motion of a camera are the subject of many research projects. Subsequent frames are analyzed to determine which picture areas are motionless. These areas are employed to calculated the camera movement. Most algorithms use a stereoscopic camera. Under some additional circumstances, the task can be accomplished by using a monocamer. The problem of calculating a camera motion is complex and so far no algorithm provides good results in the long run. The crux of the problem is to find correspondent areas between the subsequent frames. At times, errors are caused by moving objects.

Related work

Many algorithms for estimating motion of a camera have been already developed. Recently, an interesting class of techniques, SLAM (Simultaneous Localization And Mapping) has arisen. Commonly used EKF-SLAM [2] merges data from inertial sensors and characteristic points in pictures. In a similar way FastSLAM [3] works. It employs additionally particle filtering, yielding superior results. The drawback of this method is the necessity to use inertial sensors.

A related project is shown in [4]. A stereoscopic camera, tri-axial accelerometer and a magnetometer are mounted rigidly on the user. The accelerometer is used to detect step instants, whereupon the camera is triggered, giving a frame rate of 2fps. This eliminates camera swings and results in more stable pictures. The algorithm singles out traceable features. Multi-target tracking is supported by a Kalman lter. The infuence of moving objects is reduced by the RANSAC estimator. Neither the usage of the magnetometer nor the responsiveness to rotations were commented. The methodology was not validated by practical trials.

PTAM [5] is a technique that bases entirely on vision data for motion estimation. The motion is solely estimated from a monocamera. An initialization routine

is required, whereby the monocamera is shifted to capture two pictures. Consequently, a reference 3D map of the surroundings is created. As the camera starts to move, its movement vector is estimated on the base of the reference 3D map. At the same time, the 3D map of a new area is built. When exploring new areas, the current camera state is estimated on the base of the previous one what leads to a fast accumulation of errors. The algorithm performs very well for small, closed areas. The camera movement is estimated by the RANSAC estimator.

The abovementioned techniques impose big computational burden, which confines their usage to fairly simple scenes.

The solution presented in this paper neither requires inertial sensors nor imposes any scene complexity limitations. The camera can be mounted on a stable robot as well as held in hands without much negative influence on the precision of motion estimation. The algorithm was validated by a 2.5 km long path, that exhibited diversity like hanging tree branches, sun exposure, shadow and so on. The obtained preliminary results are quite satisfactory, though some glitches require refinement.

Algorithm description

The motion algorithm is depicted by the block diagram in Fig. 1. Pictures captured by the camera are converted into greyscale. The first frame serves as a key frame. Fast-10 [6], a corner detection algorithm, singles out stable corners. The next frame is compared against the key one in terms of corners. Pyramidal Implementation of the Lucas Kanade Feature Tracker [7] is applied to find corner correspondence. The paramount part of the algorithm is to calculate the motion of the camera. If the camera changes its orientation considerably, the current frame and the associated corners are stored as a reference for the subsequent frames. The algorithm iterates onwards.

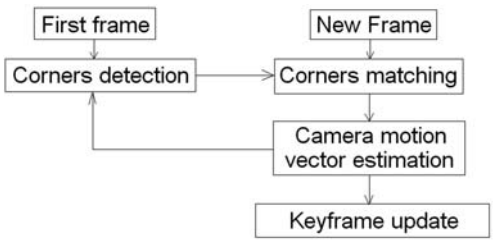


Fig. 1. System diagram.

Corners detection

Stable corners in a picture are detected by the Fast-10 algorithm [6]. Up to 2000 corners are singled out, where upon they are sorted for disparity values — closer points to the camera come first. Corners, whose disparities are not accurate, are eliminated. Only up to 100 points are seleted to the next step of the algorithm. The disparity value plays a crucial role for motion estimation, thus must be carried out carefully. The accuracy of disparity degrades heavily with the distance. Empirically, the algorithm divides selected points in this way, that 90% of them are in the distance not larger than 5m from the camera. Remaining points constitute an separate set. Performed tests showed, that motion estimation based only on the closest points is encumbered with larger errors. The proposed ratio between close and distant points strikes a good trade-off. The disparity for closer points can be estimated more accurately however they are often blurred and may vanish in the next frames, what thwarts to find a good corners correspondence between subsequent frames. Sometimes corners are falsely matched. If these correspondences correlate, then camera movement is misjudged.

Corners matching

Corner correspondence is found by the algorithm presented in [7]. A corner match is evaluated by the SSD method (Sum of Squared Difference). To speed up the corner detection, a pyramidal search is performed. In the first step, pictures of lower resolutions are compared, where coarse matches are detected. A native resolution, with subpixel refinement, is applied to estimate shifts between the corresponding corners. For this purpose, a 10 by 10 pixel window is used. The method often results in corner mismatches (see Fig. 2), whose influence is eliminated by the RANSAC algorithm, presented in the following section.

Fig 2. Corners pairing between two successive images. Some corners are mismatched, though as long as these errors are not strongly correlated, the motion is estimated correctly.

Motion vector estimation

To calculate the camera motion between two successive frames, equation (1) needs to be solved. This equation can be derived on the base of Fig. 3. Three corners are required to satisfy the equation. A larger number of points, however, improves the quality of motion estimation, sine the coordinates of every point are encumbered with an error. Usually, the Least Square method is used to estimate function coefficients from a series of uncertain observations. The function describes a process of interest. LS is vulnerable to outliers that largely bias the final estimation. RANSAC [8] was found to be vastly superior to LS, as the former has the ability to be totally insensitive to the outliers as long as they are not strongly correlated, see Fig. 4.

The RANSAC algorithm was applied in the following way. At the outset, the algorithm picks up randomly three points. Using these points, a motion vector is calculated. The algorithm iterates through the rest of the points and checks if the motion model is satisfied, assuming a given error. Then the motion vector is recalculated on the base of all selected points and the algorithm calculates parameters, that reflect the quality of motion model. The difference between the calculated location (on the base of the motion vector) and the actual location in the picture is calculated for each point. These errors are summed up and divided by the squared number of the points. The error and motion vector values are stored. The algorithm picks up randomly another three points and repeats the above procedure. After a given number of iteration, the algorithm singles out the most accurate motion vector.

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -\frac{f}{Z} & 0 & \frac{x}{Z} & \frac{xy}{f} & -\left(\frac{x^2}{f}+f\right) & y \\ 0 & -\frac{f}{Z} & \frac{y}{Z} & \left(\frac{y^2}{f}+f\right) & -\frac{xy}{f} & -x \end{bmatrix} \begin{bmatrix} U \\ V \\ W \\ \alpha \\ \beta \\ \gamma \end{bmatrix} \quad (1)$$

where: u, v — point velocity in image;
f — focal length;
x, y — point coordinates;
Z — point distance from camera;
U, V, W — camera motion;
α, β, g — camera rotation.

Results

The algorithm is implemented in C++ with OpenCV routines [10]. The application runs on a 2.5 GHz double core processor. Pictures with depth information are captured by the stereovision camera Bumblebee2 [11].

The motion estimation algorithm was tested around the University Campus with two paths, each of them ca. 2.5 km long. The camera was carried in hands and was directed along the walking direction. The acquired sequence of pictures exhibited vertical waving due to placed steps. Also horizontal waving, due to sideways slanting of the body is also present.

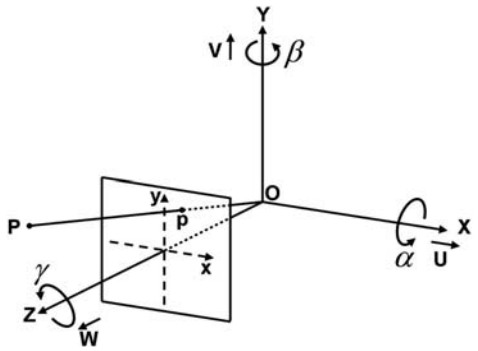


Fig. 3. The relationship between the camera motion and motion of the points in the analysed picture [9].

Fig. 4. An illustration of the RANSAC algorithm. The thick line is the best fitted one. The thin line represents a worse line model. The thick line is not biased by the points that were not fitted in the line model.

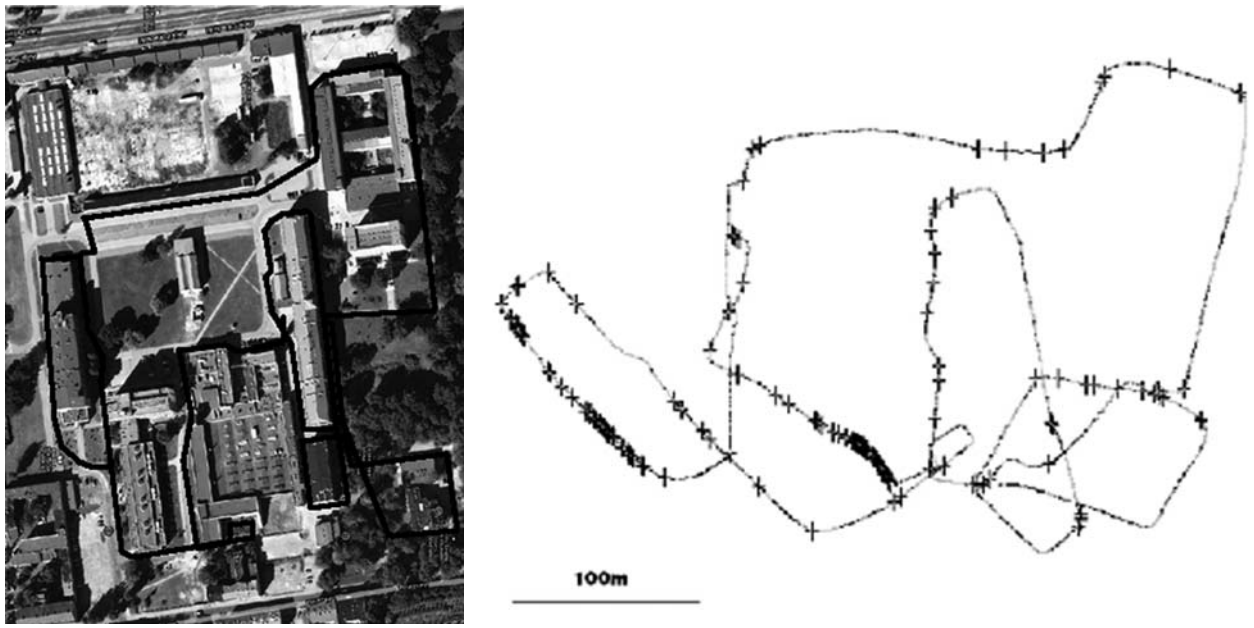
Figures 5a and 5b presents the true and the estimated paths. Darker shade corresponds to larger number of points used for calculation of the motion vector, light for a smaller number of points. The crosses stand for the places, where the algorithm could not estimate the motion. There are two main sources of path estimation inaccuracy. The first problem was with the estimation of an orientation change during turns. The algorithm was unable to find corresponding features in the subsequent pictures due to low frame acquisition rate (around 7 fps), although the person holding the camera walked with a normal pace. The second problem is associated with a direction drift, rendered by hanging down branches of trees. The algorithm mismatched points on trees between successive frames. In the case of correlated errors, the direction was misjudged. The distance was indicated precisely, as shown in Table 1. The distance measurement error is lower than 5%.

Table 1. The measured and the true distance.

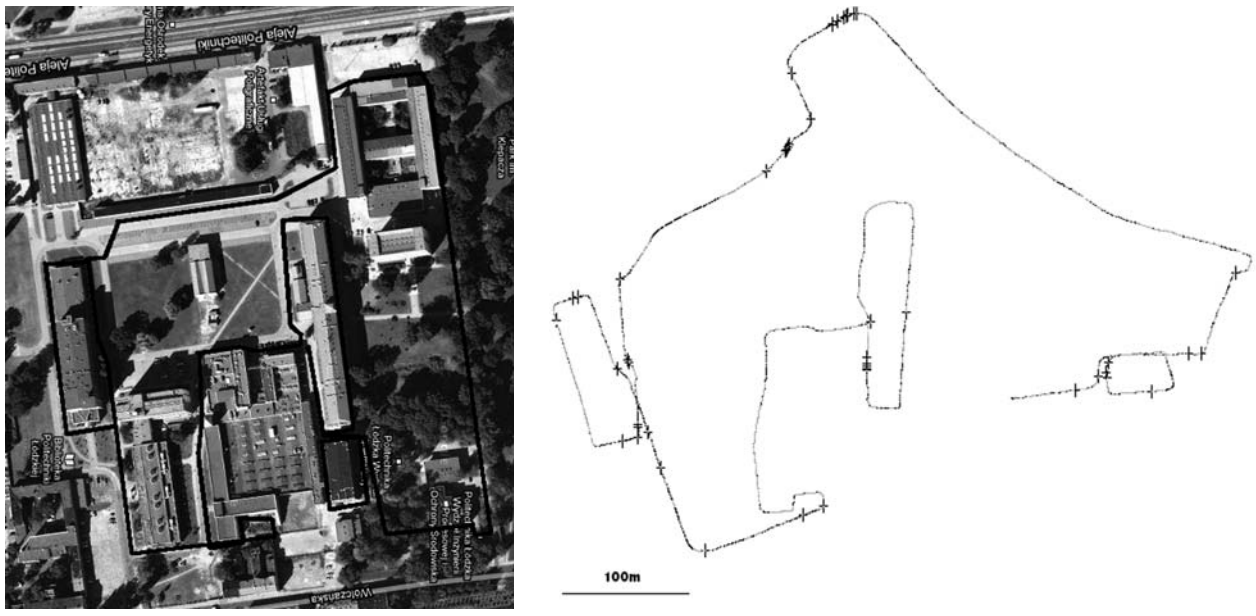
path	real distance (m)	estimated distance (m)
1	ca. 2550	2473
2	ca. 2400	2439

Conclusions

The presented algorithm correctly estimates the motion of the camera in most tested cases. The effectiveness is compromised by the evaluation of the orientation changes. The distance measurement is quite precise, up to 5%. The encountered problems warrant further work. Firstly, the algorithm should be optimized for speed, so that better frame rate can be achieved, which is especially important during turns. Direction drifts can be reduced by detecting and tracing of far located landmarks. Another idea that is worth testing relays on so called detection of loop closure. If the camera revisits a neighbourhood of a given place, then the algorithm calculate a relative location to that place. The path (distance and angles) is then corrected so that these two locations coincide. This idea is extensively used in SLAM systems.



a)



b)

Fig. 5. (a) The real path (on the left) and the estimated path (on the right) — path I. (b) The real path (on the left) and the estimated path (on the right) — path II.

References

[1] Modsching, M., R. Kramer, and K. Hagen. "Field trial on GPS Accuracy in a medium size city: The influence of built-up". Proceeding Of The 3rd Workshop On Positioning, Navigation And Communication (WPNC'06).

[2] Smith, R.C., and P. Cheeseman. "On the representation and estimation of spatial uncertainty". *International Journal of Robotics Research* 5(4), 1986: 56–68.

[3] Montemerlo, M., et al. "FastSLAM2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges". In Proc. International Joint Conference on Artificial Intelligence, 2003: 1151–1156.

[4] Cecelja, F., W. Balachandran, and R. Jirawimut. "A Stereo Vision System for Pedestrian Navigation". 17th International Conference on Applied Electromagnetics and Communications. 1–3 October 2003. Dubrovnik, Croatia.

[5] Klein, G., and D. Murray. "Parallel Tracking and Mapping for Small AR Workspaces". In Proc. International Symposium on Mixed and Augmented Reality (ISMAR'07, Nara).

[6] Rosten, E., and T. Drummond. "Machine learning for high-speed corner detection". *European Conference on Computer Vision* 2006: 430–443.

[7] Lucas, D., T. Kanade, and C. Tomassi. "Kanade Feature Tracker". http://www.ces.clemson.edu/~stb/klt/ (last visited may 2010).

[8] Fischler, M., and R. Bolles. "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography". *Communications of the ACM* 24(6), June 1981: 381–395.

[9] Skulimowski, P. "Segmentation and parametric description of scene objects in image sequences recorded form a mobile stereovision system, with the aim of auditory 3D scene presentation". PhD Thesis, Technical University of Lodz, 2009 (In Polish).

[10] "The OpenCV project". http://opencv.willowgarage.com/wiki/ (last visited may 2010).

[11] "The PointGrey". http://www.ptgrey.com/products/bumblebee2/index.asp (last visited may 2010).