Robust Dense Visual Odometry For RGB-D Cameras In A Dynamic Environment

Abdallah Dib, François Charpillet

Robust Dense Visual Odometry For RGB-D Cameras In A Dynamic Environment

Abdallah Dib*, François Charpillet*
*Inria, Villers-lès-Nancy, 54600, France.
*Université de Lorraine, LORIA, UMR 7503, Vandoeuvre-lès-Nancy, 54506, France.
*CNRS, LORIA, UMR 7503, Vandoeuvre-lès-Nancy, 54506, France.
Email: firstname.lastname@inria.fr

***Abstract*—The aim of our work is to estimate the camera motion from RGB-D images in a dynamic scene. Most of the existing methods have a poor localization performance in such environments, which makes them inapplicable in real world conditions. In this paper, we propose a new dense visual odometry method that uses RANSAC to cope with dynamic scenes. We show the efficiency and robustness of the proposed method on a large set of experiments in challenging situations and from publicly available benchmark dataset. Additionally, we compare our approach to another state-of-art method based on M-estimator that is used to deal with dynamic scenes. Our method gives similar results on benchmark sequences and better results on our own dataset.**

I. Introduction

Visual odometry is a fundamental challenge in robotics and computer vision. It consists in localizing a robot using only images coming from an on-board camera sensor. For human-robot interaction and assistance tasks, robot localization is a fundamental problem to solve in order for the robot to interact and assist the person.

There exists a variety of approaches for visual odometry, including sparse and dense methods. Sparse methods (also called feature based) only use a selection of features from the camera images which ease real-time robot localization. In contrast, dense methods (also called direct or global) use on the entire camera image for localization and do not rely on any feature extraction. Most existing approaches for visual odometry assume scenes are static[1]. These methods result in poor robot localization performances in dynamic environments because they can not differentiate between the motion of the robot and that of objects and persons. It is that inability that makes these methods inapplicable in dynamic environments.

A visual odometry method is called 'robust' if it is able to accurately estimate the camera position in a dynamic environment, where outliers pixels must be excluded in order to get accurate estimation. Moving persons, lighting changes generate outliers that must be eliminated when estimating the motion of the camera. To this end, a robust dense visual odometry method is proposed in this paper. The main contribution of this work is the introduction of a new dense visual odometry method that uses RANSAC [4] algorithm to cope with dynamic scenes which is described in section III. The accuracy and robustness of our approach is validated in section IV with an intensive set of experiments in challenging situations with persons moving in front of a moving camera. We also validate our method with benchmark sequences from [5]. The last contribution of this paper is the comparison of our method to an existing state-of-the-art robust function (Huber) that uses M-estimators to cope with outliers.

[1]where there are no changes in lighting conditions, no moving persons and objects in the scene

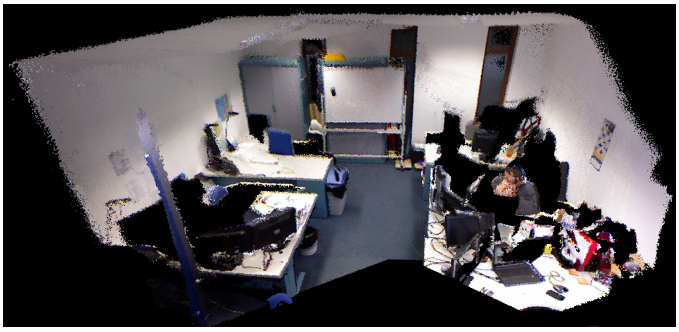


Fig. 1. 3D coloured map produced by our method from a hand-held camera in a static scene.

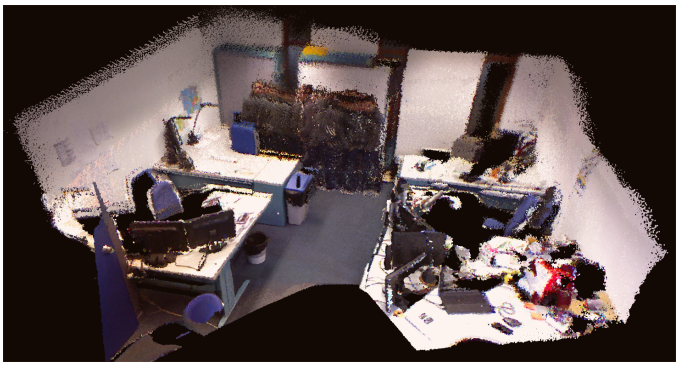


Fig. 2. 3D coloured map produced by our method from a hand-held camera in a dynamic scene. The map displays the static (inliers) and mobile (outliers) vertices. In this image, we aim to show that our method is able to accurately localize the camera when there are moving objects. The accuracy of our method can be inferred from the good quality of the reconstruction of the background (inliers) compared to figure 1.

II. Related Work

Visual odometry estimates the robot position from camera images only. It has been widely investigated in the literature.

There exists a variety of approaches for visual odometry, including sparse and dense methods.

Sparse visual odometry methods use visual features extraction such as Harris [6], FAST [7], SIFT [8] or SURF [9]. These features are tracked and used to estimate the camera motion; To cope with dynamic scenes, RANSAC is used to remove inconsistent features matches [10] [11] [12].

Dense methods use all pixels in the image for registration. The first dense odometry methods were introduced by [13]. These methods use the Lucas-Kanade framework [1] for image alignment, by minimizing the photo-metrical error between two consecutive images. [14] discussed the Lucas-Kanade framework and provides various optimizations of the algorithm. Alternatively, ICP based methods, as introduced by [15], minimize a geometrical error distance. ICP methods require to perform, at each iteration of the algorithm, an expensive nearest neighbour search. [16] uses a KD-Tree to accelerate the nearest neighbour search. [17] uses a cache for accelerating KD-tree based ICP.

Recently after the release of the low cost RGB-D cameras (e.g. Kinect, Asus Xtion), indoor visual odometry has become an active field in the research area of robotics and computer vision. Newcombe et al. [18] introduced KinectFusion developed by Microsoft for the Kinect SDK, the system uses a method derived from the ICP algorithm to align the whole image to the scene model. Microsoft introduced a real-time implementation of the ICP algorithm using GPGPU technology. Whelan et al. [19] proposed an extension to KinectFusion by integrating ICP and dense RGB-D mapping and proposes a least-square solution that minimizes both the RGB-D and ICP cost functions. The authors claims the robustness of their method in dynamic scenes but it has not been shown in their paper. Forster, et al. [20] introduced a robust semi-dense visual odometry algorithm that does not require feature extraction and matching. However this method was implemented for monocular RGB camera only, and hence it requires external sensor or prior scene knowledge to provide metric reconstruction. Tykkala et al. [3] proposed a dense method for RGB-D cameras that uses ICP. Following the same line, Audras et al. [21] proposed a robust dense method that estimates the motion of an RGB-D camera by minimizing the photo-metrical error between two images. To achieve robustness in dynamic environments, these two methods use a weight function based on robust statistics [22]. Kerl et al. [23] compared different robust functions (Huber, Tukey, T-Distribution).

It has been shown in [24] and [25] that dense methods outperform feature-based. In this paper, we use a direct visual odometry method similar to the one used in [21] to estimate the camera motion. However, instead of using robust weight functions to achieve robustness, we use a RANSAC implementation that efficiently eliminates outliers. The robustness of the proposed approach is conducted with a set of experiments in challenging situations and from benchmark sequences. Additionally, we compare our method to Huber robust weight function and we show that our approach gives similar results and in some situations it gives better estimates.

III. Method

In this section we describe the framework for robustly estimating the trajectory of the RGB-D camera in a dynamic environment. The framework is based on the Lucas-Kanade image alignment algorithm adapted to RGB-D cameras and an optimized RANSAC method for outliers rejection.

Visual odometry aims to estimate the motion of the camera between two consecutive images (I_1, I_2) by minimizing the intensity error between them.

We define the non-linear least square cost function that minimizes the intensity error between the two images:

$$E(\xi) = \sum_i (I_2((\omega(\xi, p_i)) - I_1(p_i))^2, \tag{1}$$

where ξ is the camera motion $\in \mathbb{R}^6$ that represents the linear and angular velocity of the camera, and $\omega(\xi, p_i)$ is the warping function that projects each pixel p_i from I_1 to I_2. Equation 1 can be solved using an iterative least square method. The solution to the equation 1 is ξ^* equal to:

$$\xi^* = argmin_\xi(E(\xi)).$$

Equation 1 is linearized and solved iteratively. For more details on how to solve it we refer the reader to the following papers [23] [2]. In the next section we discuss how the warping function is built.

A. Building the Warp function

In this section, the different steps for building the warp function in equation 1 are described briefly. We refer the reader to the following paper [2] for more details.

The warping function ω is constructed by first back-projecting each pixel $p(u, v)$ to a 3D point $P(X, Y, Z, 1)$ in the coordinate frame of I_1. This is possible using the depth image of the camera:

$$\begin{aligned} X &= \frac{(u - c_x)}{f_x} \times Z(p), \\ Y &= \frac{(v - c_y)}{f_y} \times Z(p), \\ Z &= Z(p), \end{aligned}$$

where $Z(p)$ is the depth of the pixel p fetched from the depth image of the camera and f_x, f_y, c_x, c_y are the intrinsics parameters (focal and optical center respectively) of the camera.

Next, the point P is projected to the coordinate frame of I_2 using the rigid body transformation T that includes the rotation and translation:

$$P' = T \times P.$$

The homogeneous representation of T is written as follow:

$$\begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix},$$

with R is a 3×3 matrix that represents the rotation and t is a 3×1 vector that represents the translation. The rotation can be

expressed with 3 Euler angular rotations only. We use the Lie algebra representation of T that represents the transformation T with a twist coordinates ξ with 6 degrees of freedom (rotation + translation):

$$\xi = (w_1, w_2, w_3, v_1, v_2, v_3).$$

where w and v are the angular and linear velocity respectively. The final transformation matrix T is obtained from the exponential map:

$$T = \exp(\hat{\xi}),$$

where $\hat{\xi}$ is defined as follow:

$$\hat{\xi} = \begin{pmatrix} \hat{w} & v \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{4\times4},$$

with $\hat{w}$ a skew-symmetric matrix equal to:

$$\hat{w} = \begin{pmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{pmatrix} \in \mathbb{R}^{3\times3},$$

and:

$$v = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}.$$

Finally, the transformed point P' is projected to screen space to get the final warped pixel coordinates $p'(u', v')$ using the following equations:

$$u' = \frac{f_x \times X'}{Z'} - c_x,$$
$$v' = \frac{f_y \times Y'}{Z'} - c_y.$$

B. Multi-resolution pyramid

Equation 1 is linearized and solved iteratively and this is only valid for small values of ξ. In order to improve the final estimate and to handle large translational and rotational movements, we construct a pyramid of RGB-D images as described in [13] where each image in the pyramid is down-sampled by a factor of 2. We start by the lowest resolution and estimate the motion that will be used as initialization for the next image in the pyramid.

C. Robustification

In this section, we describe our RANSAC implementation to cope with dynamic scenes. In the reminder of the section, we give an overview on the RANSAC algorithm and then we discuss how to implement it for dense visual odometry.

1) Background: RANSAC is an iterative method used to estimate a given model parameters using a set of N data points with outliers. The algorithm consists of two main steps: hypothesis generation and the hypothesis evaluation. In the first phase, n data points are randomly sampled from the whole dataset and an hypothesis of the model is generated using the sampled data. This hypothesis is an estimation of the model parameters. For each generated hypothesis, a score is calculated by counting the number of inliers points that lie within a predefined threshold. The algorithm iterates over these two steps and at the end it keeps the best hypothesis which has the highest number of inliers. The final selected hypothesis is refined by re-estimating the model parameters from its inliers.

The original RANSAC algorithm determines the number of iterations k required to obtain at least one non-contaminated set of n samples. We define p as the probability that only inliers are selected by the algorithm, and w be the probability that a sample is an outlier. Hence, $(1-w)^n$ gives the probability that the algorithm only selects inliers. While $1-(1-w)^n$ is the probability that a least one of the selected n points is outlier. For the all k iterations, $(1-(1-w)^n)^k$ gives the probability that the algorithm never choose a set of n points which all are inliers. Hence, $(1-(1-w)^n)^k$ is equal to $1-p$. This can be written as follow:

$$(1-(1-w)^n)^k = 1-p,$$

The number of iterations k is equal to:

$$k = \frac{log(1-p)}{log(1-(1-w)^n)}. \quad (2)$$

Generally, the best performance and speed are achieved when n is equal to the minimum number of points to generate a hypothesis. For instance, in the line fitting problem, $n = 2$ is the minimum number of points to produce a line. However, Rosten et al. [26] showed that the performance of the RANSAC algorithm can be improved by selecting more than the minimal number of samples when working with noisy data.

2) Implementation: Equation 1 has 6 unknowns (linear and angular velocities). In the Lucas-Kanade framework, In order to estimate the velocity of a pixel, the authors use a patch of of 3×3 pixels around the center one to over-determine the system. Otherwise equation 1 can not be solved. This assumption means that all pixels within that patch have the same velocity, which is not always true.

In our method, which is an extension of the Lucas-Kanade framework, to build a hypothesis, we need to sample, at least 6 pixels (which is the minimal number of pixels to obtain an estimate) and their neighbours[2] to estimate the velocity vector of the camera, which at the end gives a total of 54 pixels

In conclusion, we randomly samples 6 pixels and extract their neighbours from the image and estimates their motion with the Lucas-Kanade method. If the sampled pixels belong to a static object in the scene, the estimated motion corresponds to the camera motion. This process of random sampling is repeated a certain number of iterations and the hypothesis that has the higher number of inliers is selected.

As a reminder, the minimization of the cost function in equation 1 is a non-linear task because pixel intensities $I(p)$ are un-related to the pixel coordinates. As described in [14], there is no direct solution for equation 1. To solve it, Lucas-Kanade algorithm assumes that an initial estimate for the camera motion is known and then proceeds iteratively to find the final transformation that minimizes the cost function. This means, that in our method, each hypothesis is obtained by

[2]This process of sampling is completely different from randomly sampling 54 pixels from the whole image, because, doing the latter, yields to a lower chance that selected pixels have the same motion which means, reducing the chance that the RANSAC algorithm finds a non-contaminated set of samples.

solving the non-linear least square equation 1 for the randomly sampled pixels.

Algorithm 1 describes the different steps of the camera motion estimation algorithm. $threshLum$ and $threshDepth$ are respectively the luminance and depth threshold for a pixel to be considered as an inlier in the current hypothesis. p is the probability that at a certain iteration, the algorithm selects only inliers, and w is the probability that a pixel is an outlier.

Algorithm 1: Camera motion estimation

input : $threshLum$, $threshDepth$, p, w
output: ξ^*

$$k = \frac{log(1-p)}{log(1-(1-w)^n)}$$

foreach *image* I^k *in pyramid* I **do**
  bestConsensusSet $= 0$
  while *iterations* $< k$ **do**
    Randomly select n pixels
    Build hypothesis ξ using n pixels
    consensusSet $= 0$
    foreach *pixel* p_i *in* I^k **do**
      $residualLum = I_2^k((\omega(\xi, p_i)) - I^k{}_1(p_i)$
      $residualDepth = Z_2^k((\omega(\xi, p_i)) - Z^k{}_1(p_i)$
      if *(residualLum* $<$ *threshLum) and (residualDepth* $<$ *threshDepth)* **then**
        increment size of consensusSet
    if *(consensusSet* $>$ *bestConsensusSet)* **then**
      bestConsensusSet $=$ consensusSet
  $\xi^* =$ Build hypothesis using consensusSet
  use ξ^* as initial guess for the next pyramid level
return ξ^*

IV. Evaluation

The first part of the evaluation consists of using the TUM RGB-D benchmark [5] to compare our method to the Huber robust function using the relative pose error (RPE) as metric. The second part of the evaluation consists on testing our method in a real environment. Finally, execution speed of the method is discussed.

A. Experimental setup

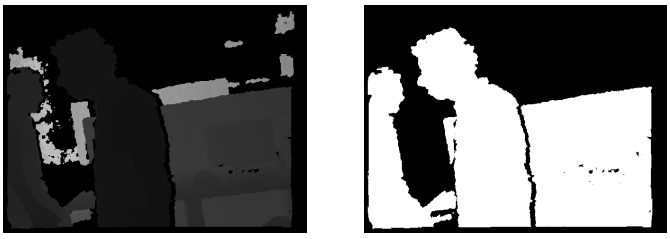
In all experiment, we have fixed $p = 0.99$, $w = 0.3$ which mean that we assume that there is no more than 30% of outliers pixels in the image and $threshLum = 30$. The $threshDepth = 5$ cm. Level 0 of the pyramid has the lowest resolution which is equal to $= 80 \times 60$ pixels. Please note that in the next, classic method refers to the non-linear least square solution without RANSAC.

B. Evaluation with TUM RGB-D benchmark

In this part, our approach is evaluated on static and dynamic sequences using a benchmark dataset [5]. To this goal, we used the relative pose error (RPE) metric, as described in [5], that measures the drift of the trajectory estimated by our method with respect to the ground truth trajectory. Table I shows the root mean square error (RMSE) calculated for each method. For static (*freiburg1_xyz*, *freiburg2_desk* and *freiburg2_xyz*) and dynamic sequences (*freiburg2_desk_with_person* and *freiburg3_sitting_halfsphere*), both RANSAC and Huber methods yield very close RMSE. Our method and the Huber function clearly outperforms the classic method. We also tested both methods in more challenging situations from the benchmark, where there were many persons moving very close to the camera in a an open space (e.g. rgbd_dataset_freiburg3_walking_rpy). Both methods failed on these sequences since the RGB-D sensor has a limited range (up to 7 meters) and it is very sensitive to noise when we surpass half that range. According to Khoshelham and Elberink [27], the accuracy of the Kinect sensor is degraded at larger distances, and thus the range must be reduced at least to 3 or 4 meters for indoor localization applications. Based on this conclusion, in our experiments, we have filtered the input images by pruning the pixels deeper than 4 meters. For instance, in the figure 3(a) most of the image (upper part) is not used because its depth is higher than 4 meters, thus reducing the number of exploitable pixels to almost the half. In figure 3(b), unused pixels are displayed in black. We calculated the percentage of unused pixels and we found that they represent 49.23% of the image. In such challenging conditions, both our method and the Huber method fail because of the extremely high number of outliers in the image and the very low percentage of the inliers due to the reduced camera range. Such conditions are still an open challenge for robot localization.

dataset	Huber	RANSAC	Classic
freiburg1_xyz	0.028044m	**0.026854m**	0.036899m
freiburg2_desk	**0.016415m**	0.018925m	0.020270m
freiburg2_xyz	**0.007809m**	0.012054m	0.036899m
freiburg2_desk_with_person	**0.021134m**	0.027167	0.028489m
freiburg3_sitting_halfsphere	0.027248m	**0.026507m**	0.028831m

TABLE I. Root mean square error (RMSE) of drift in meters per second for different methods for ground truth data.



(a) (b)

Fig. 3. a) Depth image taken from sequence rgbd_dataset_freiburg3_walking_rpy of the benchmark. b) Binary representation showing unused pixels in black (pixels that are far than 4 meters from the camera). Those pixels represent 49.23% of the image.

C. Evaluation with real experiments

We evaluated our approach using multiple experiments in a dynamic environment. Table II summarizes all sequences used in our experiments.

Sequence name	Camera position	Scene description
seq0	Hand-held camera	Static scene
seq1	Fixed camera	One person moving
seq2	Camera on pan-tilt motor	One person moving
seq3	Camera on pan-tilt motor	Two persons moving

TABLE II. SEQUENCES OF DIFFERENT EXPERIMENTS.

We start with a fixed camera and we show that our method is able to robustly eliminate outliers and outperforms the classic method (classic method is the non-linear least square without RANSAC). Next, we evaluate the method with a camera mounted on a pan-tilt motor rotating around vertical axis. Then, we compare the RANSAC method to the Huber robust function and we show that our method gives better estimates. We also show qualitative results from a hand-held camera.

1) Fixed camera in a dynamic environment: The first experiment consists of using a fixed camera in a dynamic environment. A person is continuously moving in front of the camera. This experiment is used to verify that our approach detects moving objects as outliers. With static camera, moving objects are easily obtained by subtracting two consecutive images.

Figure 4(a) and 4(c) show the subtraction of two consecutive images used as ground truth. White pixels are outliers that correspond to the motion of the person. In figures 4(b) and 4(d), green pixels correspond to outliers detected by the algorithm, blue pixels are inliers. Figure 4 shows that our approach efficiently eliminates outliers.

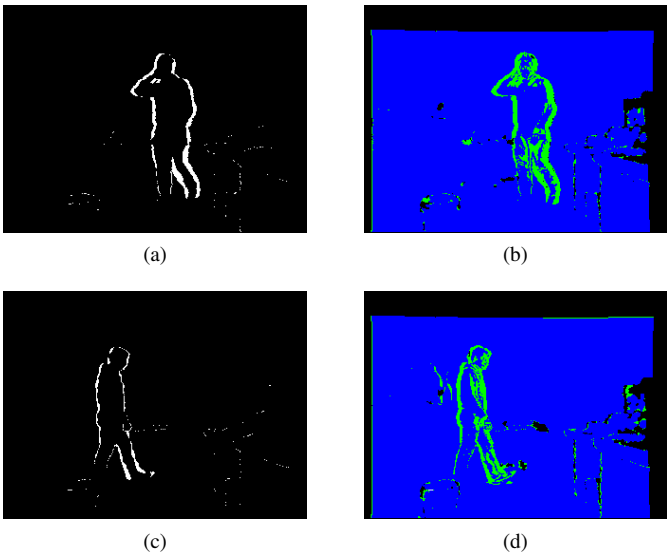
Figure 5(a) measures the distance error between the estimated camera position (x, y, z) and the real position $(0, 0, 0)$. RANSAC has a negligible error while the drift of the classic method is very high and the error reaches 20 cm.

Figures 5(b) and 5(c) show the estimated camera orientation on X and Y axis. The classic method has a very high error that exceeds 30 degrees on vertical Y-axis. The rotation on Z-axis is almost zero for RANSAC and reaches 3 degrees for the classic method.

As shown in figure 5, RANSAC highly improves the tracking stability. The drift of the classic method (without RANSAC) is very high.

2) Camera on a pan-tilt motor: In this experiment, the camera was mounted on a pan-tilt motor rotating around the vertical axis with a step of two degrees per second. One person was moving in front of the camera. Figure 6 shows the estimated trajectory of RANSAC and classic method. Figure 6(a) measures the distance error between the estimated camera position (x, y, z) and the real camera position $(0, 0, 0)$. The error of the classic method is very high and exceeds 45 cm while RANSAC has a very low one close to zero.

Figure 6(b) compares the angular rotation around the vertical axis estimated by both methods. The estimated orientation of RANSAC (red) is very close to the reference (green) and its hard to distinguish between them. For the classic method, the estimated angular rotation has a very high drift. The reference trajectory is obtained from the pan-tilt motor.



(a) (b) (c) (d)

Fig. 4. a) and c): the subtraction of two consecutive images. b) and d): inliers pixels (in blue) and outliers pixels (in green) returned by our method.

3) Hand-held camera: In this section, we show the capacity of the system to efficiently eliminates outliers from a multiple dynamic scenes with a hand-held camera. The scenes consists of a subject moving fast and jumping in front of a moving camera, as shown in Fig. 7(a) and Fig. 7(c) respectively.

Figure 7 shows the outliers detected by the method at different time of the sequence. Figure 7(a) and 7(c) correspond to the current image (I_2) taken at different time steps from the hand-held sequence. Figure 7(b) and 7(d) correspond to the outliers detected by our method.

Additionally, we built a 3D map from a hand-held sequence as shown in figure 2. This map displays both outliers (vertices corresponding to a moving person) and inliers (vertices that belong to the static background) vertices. The reconstructed map obtained from the RGB-D images qualitatively proves that the localization of the camera is accurate.

This map was built to show that our method is able to accurately localize the camera in a dynamic scene. This is inferred by the fact that the background of the scene is correctly mapped (with respect to figure 1).

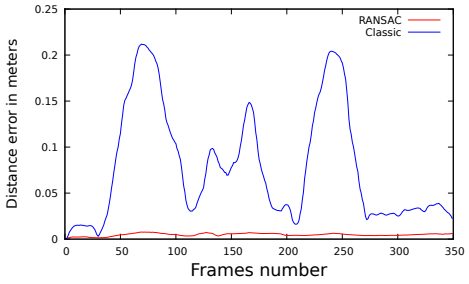
4) Comparison against Huber robust function: In this section we compare our method to the Huber robust function.

Figure 8, 9 and 10 show that our approach has lower error against the Huber robust function.

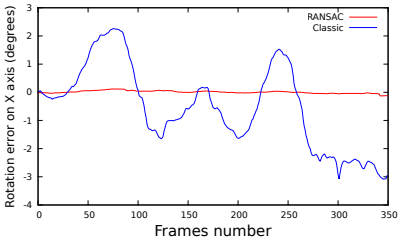
V. EXECUTION SPEED

One of the disadvantages of RANSAC is that the time it takes to find the best model parameters suffers from high variability with respect to the upper bound. Our approach adds little increment in complexity and runs at 2 FPS on a single core processor in dynamic scenes. Many parts of the algorithm can be accelerated on CPU and/or GPU.

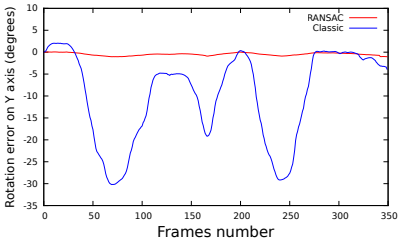
In RANSAC, the processing time for each input image depends on the percentage of outliers. Table III summarizes



(a) Trajectory error ($\sqrt{x^2+y^2+z^2}$).

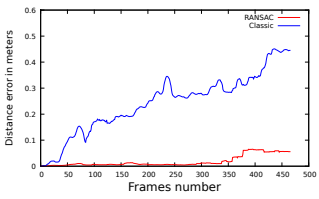


(b) Camera rotation error on x axis.

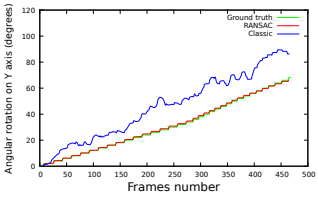


(c) Camera rotation error on y axis.

Fig. 5. Comparison of trajectory error for *seq1* (fixed camera) with RANSAC method in red and the classic method (No RANSAC) in blue. Please note that b) and c) do not have the same Y axis range.



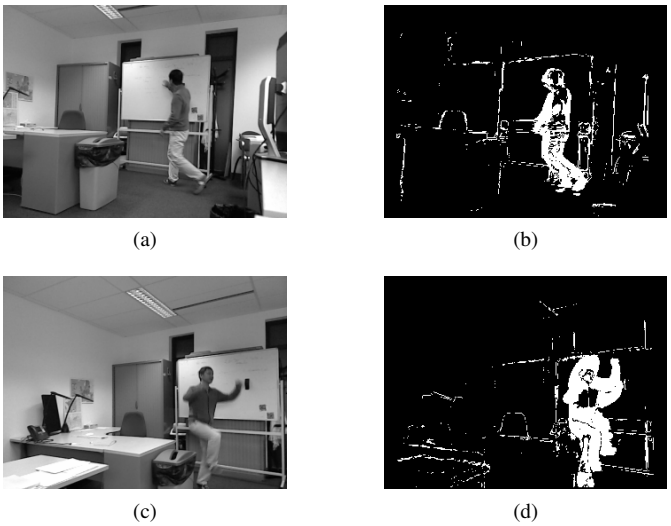
(a) Trajectory error.



(b) Estimated angular rotation on vertical axis.

Fig. 6. Comparison of estimated trajectory for *seq2* with RANSAC in red and classic method in blue. On b) the estimated angular rotation obtained by RANSAC is very close to the ground truth (in green).

the minimum execution time for different experiments. For static scenes (*seq0*), the system runs with a minimum of 14 FPS. This speed-up is explained by the fact that the algorithm stops if it finds an hypothesis with ratio higher than 0.9 which is easily satisfied for static scenes. Our method has a lower bound equal to 2 FPS for all sequences. The classic method (without RANSAC) and Huber function have approximately a constant FPS equal to 25.



(a) (b) (c) (d)

Fig. 7. Images taken at different time steps from a hand-held sequence showing that outliers (white pixels) that correspond to the motion of the moving person are efficiently eliminated by our method.

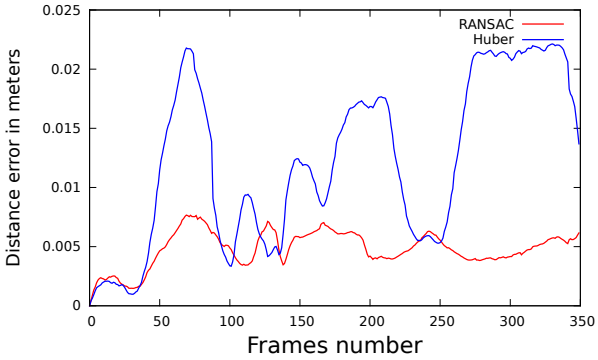


Fig. 8. Camera trajectory error for *seq1* of RANSAC method and Huber function.

VI. Conclusion and Future Work

We proposed a new method to robustly estimates the camera motion in a dynamic environment based on RANSAC algorithm. We showed with an intensive set of experiments and with benchmark sequences the robustness of the approach. We also demonstrated that our method is accurate and robust. We compared our approach with a state-of-the art method for outliers rejection based on M-estimators. In the future, we want to extend our previous method [28] which extracts mobile objects and persons in static scene to a dynamic one.

References

[1] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, April 1981, pp. 674–679.

sequence name	minimum Fps
Seq0	14
Seq1, Seq2, Seq3	2

TABLE III. Minimum execution time measured in Fps for each sequence.

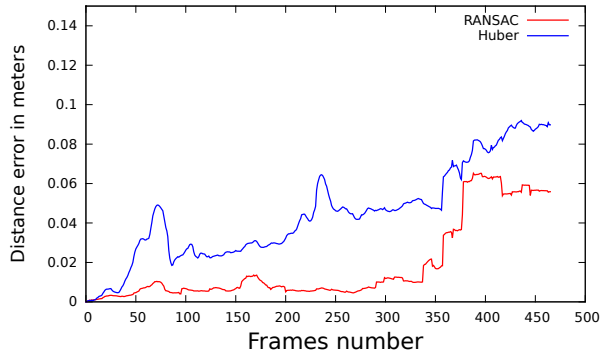


Fig. 9. Camera trajectory error for *seq2* of RANSAC method and Huber function.

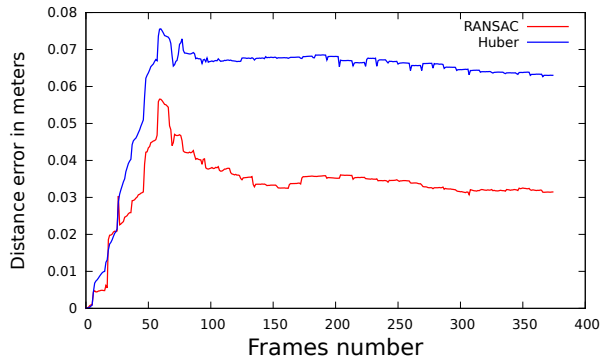


Fig. 10. Camera trajectory error for *seq3* of RANSAC method and Huber function.

[2] F. Steinbruecker, J. Sturm, and D. Cremers, "Real-time visual odometry from dense RGB-D images," in *Workshop on Live Dense Reconstruction with Moving Cameras at the Intl. Conf. on Computer Vision (ICCV)*, 2011.

[3] T. Tykkala, C. Audras, and A. I. Comport, "Direct iterative closest point for real-time visual odometry," in *ICCV Workshops*. IEEE, 2011, pp. 2050–2056.

[4] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981. [Online]. Available: http://doi.acm.org/10.1145/358669.358692

[5] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[6] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. of Fourth Alvey Vision Conference*, 1988, pp. 147–151.

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

[8] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the International Conference on Computer Vision -Volume 2-*, ser. ICCV '99. Washington, DC, USA: IEEE Computer Society, 1999. [Online]. Available: http://dl.acm.org/citation.cfm?id=850924.851523

[9] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features SURF," *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, Jun. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.cviu.2007.09.014

[10] P. H. S. Torr and A. Zisserman, "MLESAC: A new robust estimator with application to estimating image geometry," *Computer Vision and Image Understanding*, vol. 78, 2000.

[11] K. Konolige, M. Agrawal, R. C. Bolles, C. Cowan, M. Fischler, and B. Gerkey, "Outdoor mapping and navigation using stereo vision," in *Proceedings of the International Symposium on Experimental Robotics*, 2006.

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

[13] A. Comport, E. Malis, and P. Rives, "Real-time quadrifocal visual odometry," *Int. J. Rob. Res.*, vol. 29, no. 2-3, pp. 245–266, Feb. 2010. [Online]. Available: http://dx.doi.org/10.1177/0278364909356601

[14] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *International Journal of Computer Vision*, vol. 56, no. 3, pp. 221 – 255, March 2004.

[15] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992. [Online]. Available: http://dx.doi.org/10.1109/34.121791

[16] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Trans. Math. Softw.*, vol. 3, no. 3, pp. 209–226, Sep. 1977. [Online]. Available: http://doi.acm.org/10.1145/355744.355745

[17] A. Nuchter, K. Lingemann, and J. Hertzberg, "Cached k-d tree search for ICP algorithms," *3D Digital Imaging and Modeling, International Conference on*, vol. 0, pp. 419–426, 2007.

[18] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*, ser. ISMAR '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 127–136. [Online]. Available: http://dx.doi.org/10.1109/ISMAR.2011.6092378

[19] T. Whelan, H. Johannsson, M. Kaess, J. Leonard, and J. McDonald, "Robust real-time visual odometry for dense RGB-D mapping," in *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Karlsruhe, Germany, May 2013.

[20] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[21] C. Audras, A. Comport, M. Meilland, and P. Rives, "Real-time dense RGB-D localisation and mapping," in *Australian Conference on Robotics and Automation*, Monash University, Australia, December 7-9 2011.

[22] H. A. David, *Order statistics*, ser. A Wiley series in probability and mathematical statistics. Applied probability and statistics. New York: J. Wiley and sons, 1970. [Online]. Available: http://opac.inria.fr/record=b1078689

[23] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, May 2013.

[24] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *Proceedings of the 2011 International Conference on Computer Vision*, ser. ICCV '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 2320–2327. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2011.6126513

[25] S. Lovegrove, A. J. Davison, and J. I. Guzman, "Accurate visual odometry from a rear parking camera." in *Intelligent Vehicles Symposium*. IEEE, 2011, pp. 788–793.

[26] E. Rosten, G. Reitmayr, and T. Drummond, "Improved RANSAC performance using simple, iterative minimal-set solvers," *CoRR*, vol. abs/1007.1432, 2010. [Online]. Available: http://arxiv.org/abs/1007.1432

[27] K. Khoshelham and S. O. Elberink, "Accuracy and resolution of kinect depth data for indoor mapping applications," *Sensors*, vol. 12, no. 2, pp. 1437–1454, 2012. [Online]. Available: http://www.mdpi.com/1424-8220/12/2/1437

[28] A. Dubois, A. Dib, and F. Charpillet, "Using HMMs for Discriminating Mobile from Static Objects in a 3D Occupancy Grid," in *23rd IEEE International Conference on Tools with Artificial Intelligence - ICTAI 2011*. Boca Raton, Florida, États-Unis: IEEE, Nov. 2011, pp. 170–176. [Online]. Available: http://hal.inria.fr/hal-00654041