Robot Localization using efficient planar features matching

Baptiste Charmette
baptiste.charmette@univ-bpclermont.fr
Éric Royer, Frédéric Chausse and Laurent Lequievre
Clermont Universités, Institut Pascal, BP 10448, F-63000 CLERMONT-FERRAND
CNRS, UMR 6602, Institut Pascal, F-63177 AUBIÈRE

Abstract— **Real-time accurate localization is a key component of an autonomous mobile robot. Visual localization algorithms usually rely on feature matching between the current view and a map using point descriptors. Many descriptors such as SIFT or SURF are designed to recognize features seen from different viewpoint. But in a robotic context, the robot movement can be modeled and bring useful information for the matching problem. In this paper we detail a way of matching features with a local 3D model of the features taking advantage of the motion model of the robot. We describe then methods to describe the motion model. The experimental results show how useful the motion model of robot movement is, and prove that use of other sensors can greatly improve precision and robustness of the localization.**

I. INTRODUCTION

Autonomous navigation of a robot along a trajectory is possible provided that its pose can be computed at any time. Indeed, the robot has to stay on the expected trajectory and correct its movement as soon as necessary. When localization relies on vision, it is usually achieved by matching features between the current view and a map of landmarks. In our case, the map is built during an off-line learning step. Then, the robot has to be localized in this map during the autonomous navigation phase. The main difficulty to achieve this goal consists in matching points seen from different viewpoints.

Some feature descriptors such as SIFT [1] or SURF [2] are designed to be almost invariant to changes of viewpoint. But they need a lot of computation time. To achieve real-time performance, many authors [3], [4], [5] have implemented these methods on a Graphical Processing Unit (GPU). Other matching processes [6], [7] use a 3D modeling of the features to make viewpoint-independent descriptor.

However in a robotic context, the robot movement can generally be modeled, because vehicle is moving continuously — without being teleported from one point to an other. It seems interesting to take part of this fact to improve the matching and the localization. For this reason other matching methods [8], [9], [10] based on 3D models of the features have recently been proposed. These methods, instead of generating a viewpoint independent descriptor, consider landmarks as points lying on locally planar surfaces. Then, using an approximate position of the robot, the plane is projected in the current view, making the matching easier. Contrarily to [9], the present work uses monocular vision instead of stereo. And building the map off-line allows to use a much larger map than in the SLAM approach of [8]. These methods can take as much computation time as the previously cited descriptor based method. For this reason, our algorithm is implement on a GPU with CUDA to achieve real-time performance, as described in [11].

The weakest point in this method is that matching part and consequently, pose computation is highly dependent of the predicted position of the robot. In our work we have defined three prediction models and use this algorithm with every one in real conditions. With this experiment we have determined how the prediction can change the result of localization and how localization can be more precise and robust.

After a summary of the algorithm in section II, the prediction model are detailed in part III. After that, our experiment in the whole localization process is described in section IV.

II. LOCALIZATION ALGORITHM

The algorithm is designed to have first a learning stage. During this stage, the vehicle is manually driven in the test area to find some features. These features are considered as planar features, whose orientation and position are computed as described in part II-A. Then in the second stage the vehicle is automatically driven computing its position with the features compared to the images seen by the camera. The localization part is described in part II-B. Result of the localization are combined in the dynamic model of the robot described in part III.

A. Planar feature

Planar feature generation from the learning sequence use the process described in [10]. Some points are tracked on every images, using the Harris [12] interest point detector, and matched according to the Zero Normalized Cross Correlation (ZNCC). Then the structure from motion algorithm [13] computes 3D coordinates of the features and the camera pose of each view.

After that, the features are considered to be lying on a locally planar surface. Considering 2 poses P_i and P_j from where the feature can be seen, an homography $H_{i\to j}$ induced by the plane can be defined to transform the image of the feature seen from P_i into the image seen from P_j. $H_{i\to j}$ depends on the pose P_i and P_j and on the normal to the plane. $H_{i\to j}$ is used to warp the image taken from

P_i in an other view and compared with the real image as shown if the figure 1. If we note I_i^j the image of the

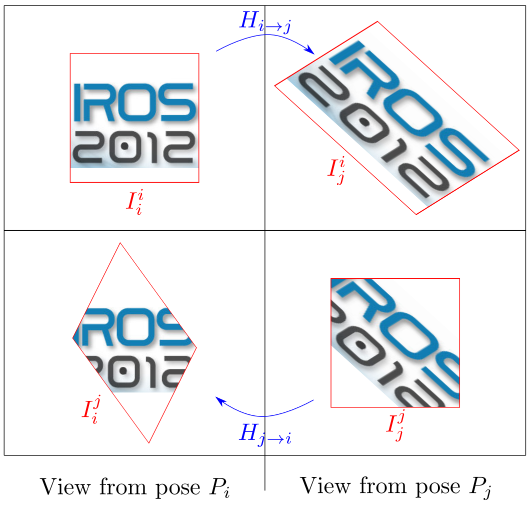


Fig. 1: Summary of the real image and warped image

feature taken from pose P_i and warped by the homography to have the appearance of the feature in the pose P_j. Then for different values of the normal, the image I_i^j is compared with the image I_j^j. An optimization algorithm whose variation parameter is the normal of the surface minimizes the difference between I_i^j and I_j^j to find the value of the normal.

Then a virtual pose P_{ref} is defined to describe the final texture of the patch. For every pose i, the final value of the normal is used to compute $H_{i\to ref}$, homography which transforms image I_i^i in an image I_{ref}^i of the feature as seen from P_{ref}. Then, every image I_{ref}^i is averaged to have the image I_{ref} of the feature. I_{ref} is considered as the appearance of the feature —the texture of the patch— as seen from P_{ref}.

We can note that P_{ref} is not a fronto-parallel pose, but a pose located close to the pose of the image where the feature was seen. This avoids to have an important resampling when the feature was seen from a grazing angle.

After that, the texture I_{ref} is compared to the image I_{ref}^i of the feature to determine the quality of this patch. Indeed, in case of bad matching, or when the feature is not lying on a planar surface, I_{ref} is a very blurred image useless for the localization. Comparison with the image give a quality factor used to remove a feature not usable.

Finally the pose P_i where the feature was observed gives an estimation of the position from where the patch can be seen. The area around these positions is computed to have the observability area. In the localization part, when the camera is not in the observability area, the matching process can be avoided on the feature.

To sum up the patches generated are composed with

- The 3D coordinates of the feature
- The orientation of the normal to the surface
- The planar texture I_{ref}
- The pose P_{ref} associated to the texture
- The observability area

B. Localization

The localization part is done in a second stage, when the vehicle is moving autonomously in the area. The image shot by the camera is compared to the patches and when their positions in the image are found, a triangulation algorithm is applied to compute the camera pose. The chart on figure 2 describe the localization algorithm used to compare features with the image sent by the camera.

1) Pose Prediction: The first step consists in predicting the camera pose P_{pred}, using dynamic model of the robot and the previous position computed. As the robot is moving on the road, the predicted parameters are only its coordinates and the yaw angle. The pitch and roll angle are supposed constant and are not used in the prediction. This prediction is made by the model described in III.

2) Patch Projection: With the pose prediction, the camera position is approximatively known and only patches whose observability area include the camera predicted position are considered. Moreover, as P_{pred} is known with a certain variance, for each patch, a region of interest (*ROI*) is computed defining the area where the patches can be located. The homography induced by the plane is computed and the texture is resampled to obtain image I_{pred} of the patch, as it was observed from the pose P_{pred}. Then a descriptor D_{patch} is computed with the image using for each pixel x the equation 1

$$D_{patch}(x) = \frac{I_{pred}(x) - \overline{I_{pred}}}{\sqrt{\sum_{x\in E_{patch}}(I_{pred}(x) - \overline{I_{pred}})^2 / N_{patch}}} \quad (1)$$

where E_{patch} represent every pixel of I_{pred}, and N_{patch} the number of element of E_{patch}

3) Image processing: At the same time the image from the camera is processed. First, the camera calibration is used to remove any distortion on the image. Then, an interest point detector — the same as the one used for the planar feature generation in section II-A — is applied on the image. After that, for every interest point, a descriptor D_{IP} is computed, using an equation similar to equation 1, using the neighborhood of the point — a window of 16 by 16 pixel in our case — instead of I_{pred}.

4) Matching: Every interest point lying in an uncertainty area of a patch is considered as a candidate to be matched with the patch. To compute their matching score S, the descriptor D_{patch} and the descriptor D_{IP} are combined with the equation 2

$$S = \frac{1}{N}\sum_{x\in E} D_{patch} D_{IP} \quad (2)$$

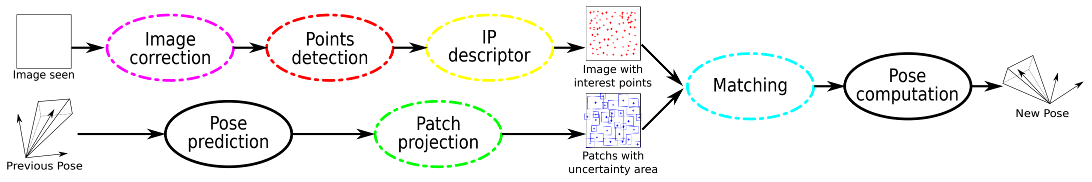


Fig. 2: Chart showing the localization algorithm

where $E = E_{IP} \cap E_{patch}$ is the set of value common with E_{patch} and E_{PI} and N its number of element.

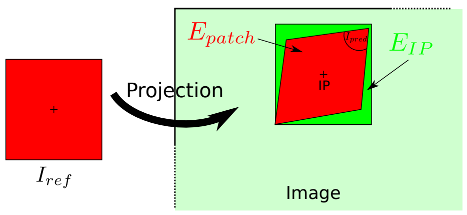


Fig. 3: Patch Projection on an image, where are shown the set E_{PI} neighborhood of an interest point and the set E_{patch}, area where the patch is projected

We can note that, as shown on the figure 3, the reprojection process make the set E_{patch} not fit in the set E_{IP}. For this reason the matching score is not exactly the same as a classic ZNCC score. However, the areas are not very different and the score computed with this method is highly similar to the real correlation computed on the region E.

The matching pairs with a score higher than a threshold — 0.5 in our experiments — are kept. When every candidate matching on every patches have been computed, they are compared and if a patch or IP is present in several different pairs, only the pair with the best score is kept.

After this operation the 2D coordinates of interest point are linked with the 3D coordinates of the patches. With these, the algorithm described in [14] is applied to compute the camera pose and the variance associated.

5) GPU Implementation: The implementation of the algorithm on a classical processor needs a lot of computation time, mainly during the projection of patch. For this reason an implementation on GPU (graphics processing unit) has been done and is described in details in [11]. In particular, the major improvements happen in the resampling part of the algorithm, in the image correction and the patch projection. The resampling is done on the GPU with the texture memory, which is designed to compute hardware interpolation explaining why such an improvement can be done.

An other improvement is due to the massive parallelization of the GPU, which allows to compute several patches simultaneously.

To evaluate the improvement made by the GPU implementation, some measurement of the execution time of the localization have been made. The global time has been decreased to less than 200 ms per image and made it possible to use it in real time for the navigation of a mobile robot. The position is then send to the dynamic model of the robot which will predict the position in the next iteration.

III. Dynamic Model of the Robot

The position computed with the image analysis is integrated in a dynamic model of the robot. This model is then used to compute the prediction of the next position needed in the first part of the localization algorithm. Three models have been developed for our tests.

A. *Simple model*

The first model used considers only that the robot is not moving quickly and so uses the previous camera pose as the prediction. With this model, the uncertainty of the prediction is not changing, and the covariance matrix associated to the position is the sum of the covariance matrix of the previous pose and a constant matrix, big enough to insure the new position of the robot is in the uncertainty area. The main advantage of this model is that no assumption are made on the vehicle movement. But the uncertainty area has to be important to consider any change of position. Moreover, the prediction is not very reliable because generally, the vehicle is moving.

B. *Constant speed model*

The second model uses an extended Kalman filter [15] to predict the evolution of the robot. The model is considered only in 2D, in the ground plane. The main idea of this model is to consider that the linear speed — written $\overline{v}$ — and angular speed — written $\dot{\theta}$ — are constant. The state vector $\underline{X}$ is defined with $\underline{X} = (x, z, \theta, \overline{v}, \dot{\theta})^T$ where θ is the yaw angle. The vehicle is modeled with the tricycle model showed on figure 4. In the discrete time, the value at iteration $k+1$ is defined by equation 3.

$$\begin{pmatrix} x_{k+1} \\ z_{k+1} \\ \theta_{k+1} \\ \overline{v}_{k+1} \\ \dot{\theta}_{k+1} \end{pmatrix} = \begin{pmatrix} x_k - \frac{\overline{v}_k}{\dot{\theta}_k} \left(\cos(\theta_k - \dot{\theta}_k \Delta t_k) - \cos\theta_k\right) \\ z_k - \frac{\overline{v}_k}{\dot{\theta}_k} \left(\sin(\theta_k - \dot{\theta}_k \Delta t_k) - \sin\theta_k\right) \\ \theta_k + \dot{\theta}_k \Delta t_k \\ \overline{v}_k \\ \dot{\theta}_k \end{pmatrix} + \underline{W}_k \quad (3)$$

with Δt_k the elapsed time between iteration k and $k+1$ and $\underline{W}_k$ a noise vector of the same dimension as $\underline{X}$ whose coordinates are supposed uncorrelated zero-mean Gaussian values. In the extended Kalman filter implementation, the covariance matrix Q_k associated to $\underline{W}_k$ is constant. In our

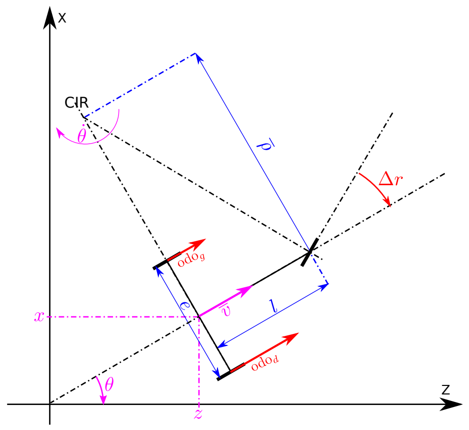


Fig. 4: Model used to describe the vehicle

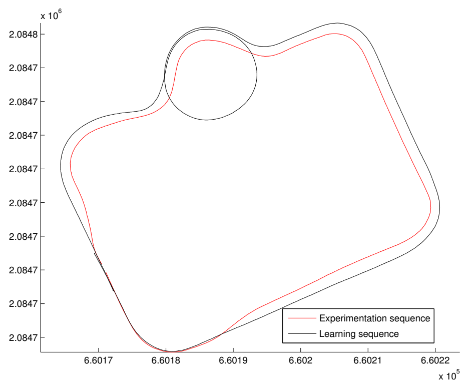


Fig. 5: GPS Track of the learning and test sequence

case, experimental values taken for this matrix is $10^{-5}I$ with I, the identity matrix of size 5.

The vision localization algorithm updates directly the first three values of the state vector using Kalman equation. The noise on the measure is computed in the localization process using the projection error of every features.

This model gives better prediction than the simple model, because vehicle speed is taken in consideration. The prediction errors occur mainly when the vehicle is turning and/or changing its speed.

C. *Data fusion with odometry*

To help the prediction and predict the change of speed, some information from odometry is used in the last model. The same filter as the constant speed model is used but besides the vision, data from odometry are used to update the filter. The sensors embedded in the vehicle measure the linear speed of the left and right wheel and the angular deviation of the front wheel in the tricycle model. These value are respectively written odo_l, odo_r and Δr and symbolized in red on figure 4. To integrate these values in the filter, the observation function is given by equation 4.

$$\underline{Y}_k = \begin{pmatrix} \text{odo}_l \\ \text{odo}_r \\ \Delta r \end{pmatrix} = \begin{pmatrix} \overline{v} + \dot{\theta} e/2 \\ \overline{v} - \dot{\theta} e/2 \\ \arctan\left(\frac{l\dot{\theta}}{\overline{v}}\right) \end{pmatrix} \tag{4}$$

with l the distance between rear and front wheel and e distance between left and right wheel.

This model can give the best information available at every time, mainly because data from odometry is available at high frequency.

The three models can be used to predict the pose in the localization process. But in the case of autonomous navigation, they can be used at any time — even between image acquisition — to evaluate the current position of the vehicle and, for example send correct orders to the robot actuator.

IV. Experiment

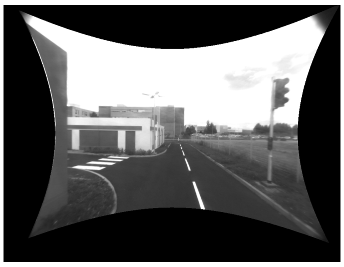
The whole algorithm has been tested to localize a vehicle. First a learning sequence has been acquired with the vehicle moving on the right side of the road. Then the vehicle is manually driven along the left side of the same road, and the algorithm analyzes the camera image to determine the position of the vehicle relatively to the learning sequence. The vehicle has a differential GPS with centimeter precision on board to define a localization reference. Image used have a resolution of 512×384 pixels and the localization was composed of 1232 images. Figure 5 shows the GPS track of the vehicle trajectory while saving learning and localization sequence. Figure 6b shows an image used to generate the patches, 6a shows an image saved in the localization process and 6c the projection of every patch in the pose of this image.

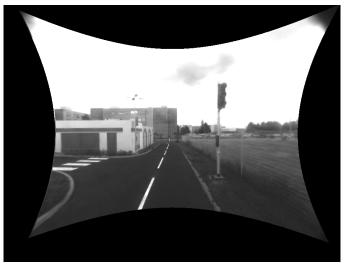
A. *Precision Evaluation*

The localization process used the patches built in the learning sequence. Obviously, the GPS reference is only used a as the ground truth to compare the result and not used for the localization. To compare the vision results with the GPS localization, the learning trajectory is used to find the transformation needed to convert vision localization in a GPS reference. Then the position found with every prediction model can be compared with the reference.

Figure 7 shows the localization results for every method and numerical evaluation of the localization error is summed up in the table I. We can see directly on the top view that with the simple model, localization is failing before the end. The comparison with the reference shown in figure 7b and in the table I are taken on the beginning of the trajectory, when the localization has not failed. Even in this part the first model has less precision than the other. It proves that the simple model gives a quite bad prediction to the algorithm. The fact that localization failed show that prediction is very important in the process. Indeed when the prediction is bad, the patch reprojections are bad and can not be matched with the current view.

For the other localization algorithm, there is no real difference in localization, and no algorithm can be considered more accurate than another. We can conclude that, concern-


(a) Localization image


(b) Learning image

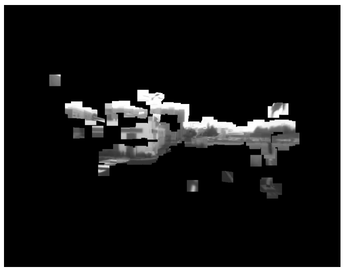
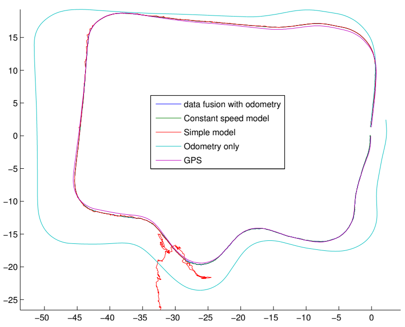
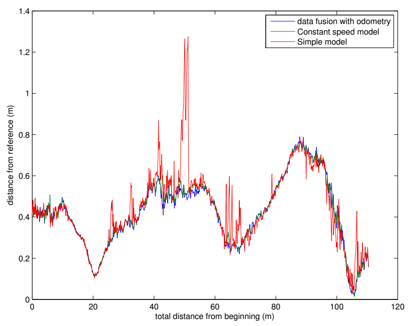

(c) Patches reconstruction

Fig. 6: Extract from the test Sequence



(a) Top view of the localization



(b) Localization error. Only the part where every method are localized is shown.

Fig. 7: Results of the localization

	average error	standard deviation	median error	maximum error
Simple model	0.441543	0.177010	0.427648	1.276092
Constant speed model	0.422777	0.157316	0.422250	0.767080
Data fusion with odometry	0.419063	0.157680	0.422715	0.789980

TABLE I: Result of the localization process on the beginning of the sequence (where no localization has failed)

ing the precision of the localization, when the prediction is quite correct, it has no influence.

B. *Robustness Evaluation*

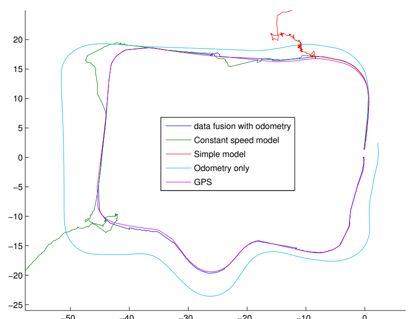
If the precision is the same with or without odometry, the use of an other sensor is not useless. Actually in harder conditions, for exemple if there is some problem with the camera, the use of the odometry improves the localization. To evaluate this point an other experimentation was conducted. The same sequence was used, but several images were removed. The lack of images simulate for example an occultation of the camera or a temporary under or overexposure which is common in the case of indoor-outdoor transition. With this kind of problems, the vehicle can make an important movement between two well exposed images. Results are shown on figure 8. In this experiment, localization using the simple model failed in the first turn with a lack of images. It seems logical because in the case of a turn, the features are moving a lot in the image. If the prediction is still the same pose as before, no projected patch can be matched on the current view.

Using the model without odometry creates several differences and localization failed in the middle of the trajectory. Figure 8b shows that even in the first part, localization is quite bad after every lack of image, mainly when the vehicule is turning. It is confirmed with statiscal analysis of the data in the table II. The use of odometry greatly improves the prediction, having a correct localization in every situation.

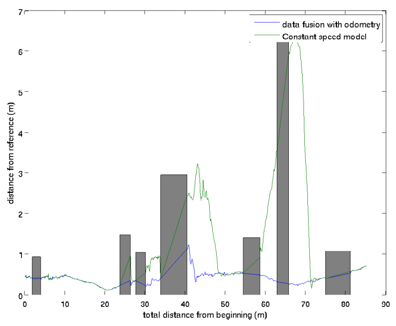
V. CONCLUSIONS AND FUTURE WORKS

A. *Future Works*

To use directly Kalman filter equation in the `Constant speed model`, the covariance on the model and on the measured value have to be known. Measure variance is evaluated with the reprojection error given by the pose computation. The model uncertainty is not so easy to know. In fact, it symbolizes the time variation of the speed, or in other word the acceleration. For the moment, the uncertainity is considered as constant and applied in the same way on every value. A future work could be to model the noise only as an acceleration part on every iteration and combine it in the Kalman filter. This can decrease uncertainty on the prediction method and then decrease computation time of



(a) Top view of the localization



(b) Localization error. Only the part where both methods are localized is shown. Gray rectangles are the parts where no images are available

Fig. 8: Results of the localization after removing images

	average error	standard deviation	median error	maximum error
Constant speed model	1.13	1.42	0.47	6.43
Odometry fusion	0.41	0.15	0.41	1.23

TABLE II: Result of the localization process on the beginning of the sequence (where no localization has failed) where several images have been removed

the localization because less matching candidates would be analysed.

An other way of improvement could be to use the prediction to make a previous selection of the patches before the projection. Although the observability area is filtering several patches, a lot of them are still processed and are not useful, because they are not matching any interest point. A previous analysis could avoid to make too many projection, and prefer to project only the best patches which are sufficient to achieve a good localization.

B. Conclusions

We have shown that a new matching method based on planar feature modelling and reprojection can be used for the robust pose computation of a mobile robot. Thanks to an efficient GPU implementation, the localization algorithm can be used for real time autonomous navigation. The experimental resuts outline that the prediction step is really important for the robustness of the algorithm. the information given by other sensor such as odometry can also improve the algorithm particularly in difficult cases when the position computed with vision momentarily unavailable.

VI. Acknowledgments

This work was supported by the french ANR national agency in the CityVIP project

References

[1] D. G. Lowe, “Object recognition from local scale-invariant features,” in *ICCV*, 1999, pp. 1150–1157.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “Speeded-up robust features (SURF),” *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[3] S. Sinha, J. Frahm, M. Pollefeys, and Y. Genc, “GPU-based video feature tracking and matching,” in *EDGE, Workshop on Edge Computing Using New Commodity Architectures*, vol. 278. Citeseer, 2006.

[4] N. Cornelis and L. Van Gool, “Fast scale invariant feature detection and matching on programmable graphics hardware,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2008. CVPR Workshops 2008*, 2008, pp. 1–8.

[5] M. Schweitzer and H.-J. Wuensche, “Efficient Keypoint Matching for Robot Vision using GPUs,” in *Fifth IEEE Workshop on Embedded Computer Vision (ECV’09)*, 2009.

[6] K. Koser and R. Koch, “Perspectively invariant normal features,” in *ICCV 2007*, 2007, pp. 1–8.

[7] C. Wu, B. Clipp, X. Li, J. Frahm, and M. Pollefeys, “3D model matching with Viewpoint-Invariant Patches (VIP),” in *CVPR 2008*, 2008, pp. 1–8.

[8] N. Molton, A. Davison, and I. Reid, “Locally planar patch features for real-time structure from motion,” in *BMVC*, 2004.

[9] C. Berger and S. Lacroix, “Using planar facets for stereovision SLAM,” in *IROS*, 2008, pp. 1606–1611.

[10] B. Charmette, É. Royer, and F. Chausse, “Matching Planar Features for Robot Localization,” in *International Symposium on Visual Computing*. Springer, 2009, pp. 201–210.

[11] B. Charmette, F. Chausse, and É. Royer, “Efficient planar features matching for robot localization using gpu,” in *Sixth Workshop on Embedded Computer Vision held in conjuction with CVPR 2010*, Jun. 2010.

[12] C. Harris and M. Stephens, “A combined corner and edge detector,” in *Alvey vision conference*, 1988.

[13] E. Royer, M. Lhuillier, M. Dhome, and J. Lavest, “Monocular vision for mobile robot localization and autonomous navigation,” *International Journal of Computer Vision*, vol. 74, pp. 237–260, 2007.

[14] H. Araujo, R. Carceroni, and C. Brown, “A fully projective formulation to improve the accuracy of Lowe’s pose-estimation algorithm,” *Computer vision and image understanding(Print)*, vol. 70, no. 2, pp. 227–238, 1998.

[15] R. Kalman, “A new approach to linear filtering and prediction problems,” *Journal of basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.