Odométrie Visuelle Monoculaire

SOBUCKI Stéphane

***Abstract*— Ce document introduit l'odométrie visuelle monoculaire, qui est une technique pour estimer la pose de la caméra à chaque instant et estimer sa trajectoire. Nous nous intéresserons plus particulièrement à une méthode pour les problèmes plans.**

***Index Terms*— Odométrie visuelle, homographies, calibration**

I. Introduction

L'odométrie visuelle [1][2] est une technique qui permet de retrouver, à partir d'une séquence d'images ou d'une vidéo, la position de la caméra associée à chaque prise de vue et d'estimer la trajectoire associée. C'est une technique qui a fait son apparition pour répondre à une problématique que l'odométrie "classique". En effet, l'odométrie "classique" est basée sur le mouvement des roues et ne convient donc pas à toutes les surfaces et est sujette aux problèmes de glissement [3]. Nous allons dans ce papier nous intéresser à l'odométrie visuelle monoculaire qui consiste à utiliser une seule caméra. Nous dresserons l'état de l'art de cette technique et nous allons également présenter quelques domaines d'applications. Enfin, nous proposerons une implémentation d'une méthode et analyserons les résultats obtenus.

II. État de l'art et applications

L'odométrie visuelle monoculaire est utilisée dans des domaines tels que l'automobile pour les véhicules autonomes [4][5] ou en robotique d'exploration spatiale [6][7]

Les principales étapes de l'odométrie visuelle sont énumérées ci-dessous[1] :

1) Acquisition des images
2) Détection des paramètres/points d'intérêt
3) Suivi des paramètres/points d'intérêt (tracking)
4) Mise en correspondance des paramètres/points d'intérêt entre deux images (matching)
5) Estimation de la pose
6) Estimation de la trajectoire

Il y a plusieurs méthodes différentes pour réaliser un système d'odométrie visuelle monoculaire. Tout d'abord, sur les paramètres d'intérêt peuvent être géométriques ou non. Il existe différents choix pour les paramètres d'intérêt parmi lesquels : SURF [8], SIFT [9], Harris [10] etc. Ensuite, le fait que la caméra soit calibrée ou non mène à différents algorithmes. Si la caméra est calibrée, l'estimation de la pose se fait à partir de la matrice essentielle sinon on utilise la matrice fondamentale. Enfin, si les points d'intérêt sont coplanaires nous utiliserons l'homographie, qui est un cas particulier de la matrice essentielle, pour estimer la pose.

Il existe également des méthodes basées sur du Deep Learning non supervisé [11] qui n'ont besoin que de l'acquisition d'images.

III. Implémentation

La méthode que nous allons implémenté est celle proposée par Zhang [12] qui permet de calibrer une caméra avec une mire plan (ici un échiquier). Pour les paramètres d'intérêt on choisit les intersections des droites de l'échiquier, que l'on trouve avec le détecteur de Harris.

A. Acquisition d'images

Pour l'acquisition d'images, nous avons utilisé un iPad pro 2018. On va placer un échiquier au sol et prendre une série d'images en se déplaçant avec la tablette. Nous allons travaillé sur un set de 15 images d'échiquier.

B. Détection des points d'intérêt

Pour la détection des points d'intérêt, on veut récupérer les coins qui se trouvent dans l'image. Pour cela, on va utiliser la fonction *detectCheckerboardPoints* sous Matlab.

Les points en rouge (Fig. 1) sont les points d'intérêt que nous allons utilisé pour calculer les homographies et calibrer notre caméra.

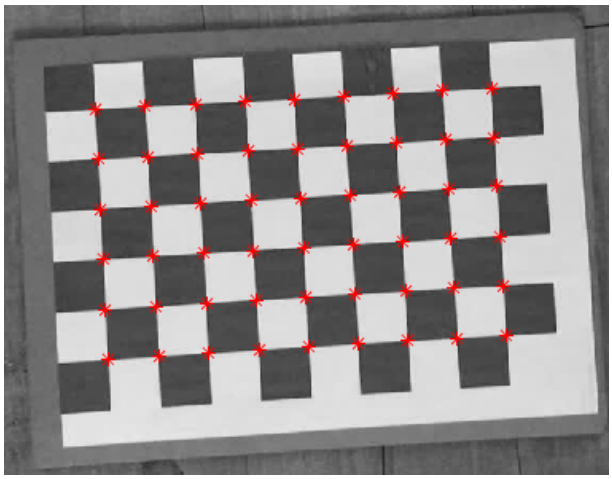

Fig. 1: Détections de coins

C. Homographie

Nous nous plaçons dans le cas particulier d'une mire plan (un échiquier). Nous allons sélectionner des couples (M_i, m_i) pour estimer P. Les M_i sont considérés coplanaires et définissent le plan $z = 0$. Une transformation qui transforme un plan en un autre est une homographie H. H est un cas particulier de P.

$$P\begin{pmatrix} X_i \\ Y_i \\ Z_i = 0 \\ 1 \end{pmatrix} = K\begin{pmatrix} R & T \end{pmatrix} = K\begin{pmatrix} r_1 & r_2 & \cancel{r_3} & T \end{pmatrix}\begin{pmatrix} X_i \\ Y_i \\ \cancel{Z_i} \\ 1 \end{pmatrix}$$

$$H_i\begin{pmatrix} X_i \\ Y_i \\ 1 \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \text{ avec H = } K\begin{pmatrix} r_1 & r_2 & T \end{pmatrix}$$

que l'on peut réécrire sous la forme suivante :

$$H_i\tilde{M}_i \times \tilde{m}_i = \begin{pmatrix} O_3^t & -\tilde{M}_i^t & y_i\tilde{M}_i^t \\ \tilde{M}_i^t & O_3^t & -x_i\tilde{M}_i^t \\ -y_i\tilde{M}_i^t & x_i\tilde{M}_i^t & O_3^t \end{pmatrix}\begin{pmatrix} H^{1^t} \\ H^{2^t} \\ H^{3^t} \end{pmatrix}$$

Cette matrice est de rang 2 à cause du préproduit vectoriel. On ne garde donc que 2 lignes sur 3 pour calculer H. Nous avons 9 inconnues et 2 équations par couple $(\tilde{M}_i, \tilde{m}_i)$. Nous pouvons donc estimer H avec 4 couples connus. On construit la matrice A_i constituée des couples connus et le vecteur des inconnues : $X = \begin{pmatrix} h_{11} \\ h_{12} \\ \vdots \\ h_{33} \end{pmatrix}$ pour définir le système à résoudre $A_iX = O_{2n}$ Nous pouvons résoudre ce système par la décomposition en valeurs singulières (ou SVD). $A = U\Sigma V^t$ et la solution de notre équation correspond à la dernière colonne de V : $X = V$, on en déduit l'homographie H. On définit l'erreur de reprojection telle que :

$$e = \sum_{i=1}^{n}(H\tilde{M}_i - \tilde{m}_i)^2$$

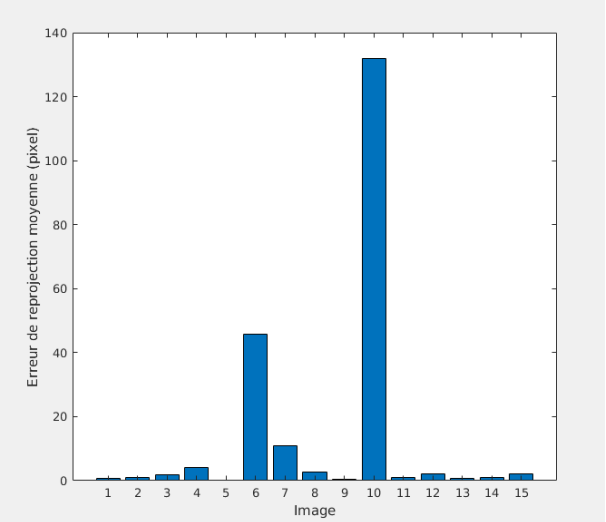

Fig. 2: Erreurs de reprojection

Si on se contente de choisir 4 couples aléatoires pour trouver H, on peut avoir une très grande erreur de reprojection (Fig. 2). Nous pouvons utiliser l'algorithme RANSAC [13] qui permet en prenant comme critère cette erreur de trouver les meilleurs couples pour notre modèle. Pour N itérations, on choisit aléatoirement 4 couples, on calcule H et on regarde l'erreur de reprojection. On choisit les 4 couples qui minimisent l'erreur de reprojection sur l'ensemble de nos points.

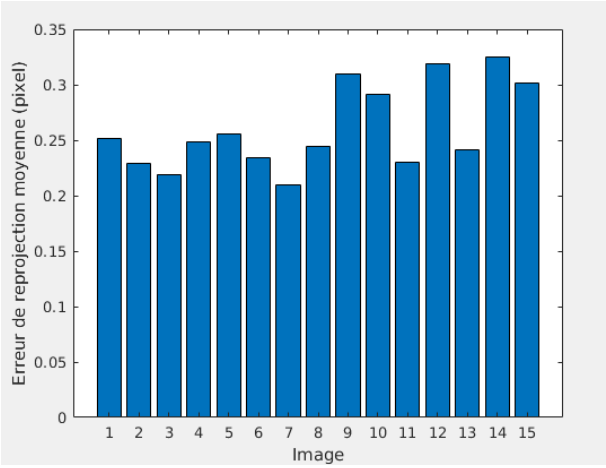

Fig. 3: Erreurs de reprojection avec RANSAC : N = 5000

L'algorithme est beaucoup plus robuste et l'erreur de reprojection étant inférieure à 0.35 pixels en moyenne (Fig. 3), on peut dire qu'elle est très satisfaisante.

D. Calibration

Pour calibrer notre caméra, nous utiliserons les propriétés d'orthogonalité de la matrice de rotation et que chacun de ces vecteurs colonnes est à norme unitaire. $r_1 = K^{-1}h_1$ et $r_2 = K^{-1}h_2$ définis à un facteur d'échelle λ près.
D'après les propriétés d'une matrice orthogonale le produit scalaire de $r_1 \cdot r_2 = 0$. Nous avons donc le système d'équation suivant :

$$\begin{cases} h_1^tK^{-t}K^{-1}h_2 = 0 \\ h_1^tK^{-t}K^{-1}h_1 = h_2^tK^{-t}K^{-1}h_2 \end{cases}$$

On note $B = K^{-t}K^{-1}$ la matrice définie ci-dessous :

$$\begin{pmatrix} \frac{1}{f_x^2} & \frac{-\gamma}{f_x^2 f_y} & \frac{\gamma c_y - c_x f_y}{f_x^2 f_y} \\ \frac{-\gamma}{f_x^2 f_y} & \frac{\gamma^2 + f_x^2}{(f_x f_y)^2} & \frac{-\gamma^2 c_y + \gamma c_x f_y - c_y f_x^2}{(f_x f_y)^2} \\ \frac{\gamma c_y - c_x f_y}{f_x^2 f_y} & \frac{-\gamma^2 c_y + \gamma c_x f_y - c_y f_x^2}{(f_x f_y)^2} & \frac{(\gamma c_y)^2 + (c_y f_x)^2 + (f_x f_y)^2}{(f_x f_y)^2} \end{pmatrix}$$

On remarque que la matrice B est symétrique, on peut donc représenter la matrice B comme un vecteur b défini tel que :

$$b = \begin{pmatrix} B_{11} & B_{12} & B_{13} & B_{22} & B_{23} & B_{33} \end{pmatrix}^T$$

En reprenant les équations ci-dessus et en remplaçant $K^{-t}K^{-1}$ par B, on remarque que l'on peut réécrire l'équation 1 de cette façon :
$h_i^tBh_j = v_{ij}^tb$

Avec v, un vecteur de taille 1x6 tel que :

$$v_{ij} = \begin{pmatrix} h_{1i}h_{1j} & h_{1j}h_{2i} + h_{1i}h_{2j} & h_{1j}h_{3i} + h_{1i}h_{3j} & h_{2i}h_{2j} & h_{2i}h_{3j} + h_{3i}h_{2j} & h_{3i}h_{3j} \end{pmatrix}^t$$

Nous pouvons donc réécrire le système d'équations de la façon suivante :

$$\begin{pmatrix} v_{12} \\ v_{11} - v_{22} \end{pmatrix} b = 0$$

Il y a 6 inconnues à déterminer et chaque homographie nous donne deux équations, il faut donc avoir au préalable calculer au moins 3 homographies pour pouvoir résoudre ce système et déterminer b. On note A la matrice contenant les équations associées à chaque homographies et nous avons donc le système suivant à résoudre : $Ab = 0$. Nous pouvons donc trouver b en faisant une décomposition en valeurs singulières $A = U\Sigma V^t$ La solution correspond à la dernière colonne de V. En se souvenant que la solution est vraie à un coefficient près, on note $B = \lambda K^{-t}K^{-1}$. Enfin, on peut retrouver la matrice des paramètres intrinsèques avec :

$$\begin{cases} c_y = \frac{B_{12}B_{13} - B_{11}B_{23}}{B_{11}B_{22} - B_{12}^2} \\ \lambda = B_{33} - \frac{B_{13}^2 + c_y(B_{12}B_{13} - B_{11}B_{23})}{B_{11}} \\ f_x = \sqrt{\frac{\lambda}{B_{11}}} \\ f_y = \sqrt{\frac{\lambda B_{11}}{B_{11}B_{22} - B_{12}^2}} \\ \gamma = \frac{-B_{12}f_x^2 f_y}{\lambda} \\ c_x = \frac{\gamma c_y}{f_y} - \frac{B_{13}f_x^2}{\lambda} \end{cases}$$

Après avoir calibrer notre caméra, on peut maintenant retrouver sa pose dans le repère monde.

E. Determination de la pose et de la trajectoire

On a pour chaque image, les relations ci-dessous définies à un coefficient multiplicatif près.

$$\begin{cases} r_{i,1} \propto K^{-1}H_{i,1} \\ r_{i,2} \propto K^{-1}H_{i,2} \\ t_i \propto K^{-1}H_{i,3} \end{cases}$$

D'après les propriétés d'une matrice orthogonale, on doit avoir $||r_{i,1}|| = ||r_{i,2}|| = 1$. Pour cela, on doit diviser normalisé le terme en $K^{-1}H$. On définit donc deux coefficients :

$$\begin{cases} \lambda_1 = ||K^{-1}H_{i,1}|| \\ \lambda_2 = ||K^{-1}H_{i,2}|| \end{cases}$$

On a donc :

$$\begin{cases} r_{i,1} = \frac{K^{-1}H_{1,i}}{\lambda_1} \\ r_{i,2} = \frac{K^{-1}H_{2,i}}{\lambda_2} \\ r_{i,3} = \pm(r_{i,1} \times r_{i,2}) \\ t = \frac{2K^{-1}H_{3,i}}{\lambda_1 + \lambda_2} \end{cases}$$

Le signe de $r_{i,3}$ est choisi tel que $\det(R) = 1$.

Cette méthode ne nous permet pas d'obtenir systématiquement une matrice de rotation orthogonale. Il est alors nécessaire de l'orthogonaliser, pour cela il existe différentes méthodes. Nous définissons l'erreur comme le résultat du produit scalaire de r_1 et r_2. Si nous supposons que cette erreur est faible, nous pouvons orthogonalisé la matrice de rotation de cette façon. On a ainsi accès à la position de la caméra pour chaque image et on peut en déduire la trajectoire en reliant chaque position successive de la caméra par une droite.

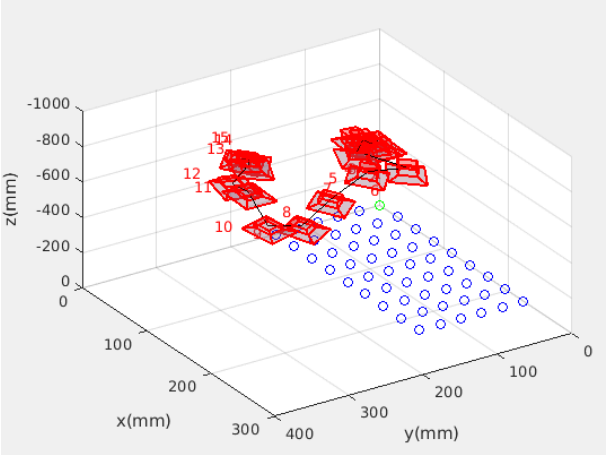

Fig. 4: Estimation de la trajectoire de la caméra

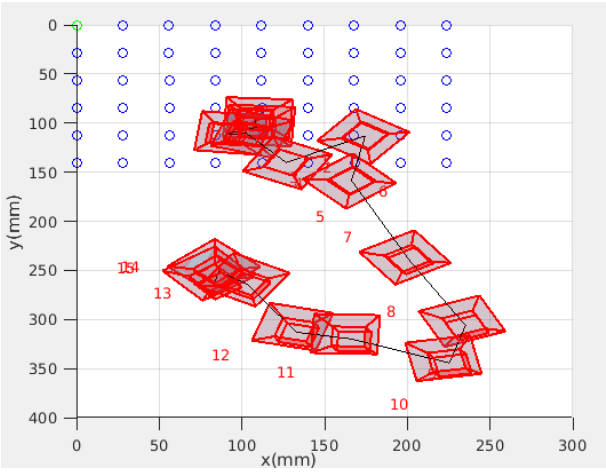

Fig. 5: Estimation de la trajectoire de la caméra dans le plan XY

IV. Analyse des résultats

Nous arrivons à estimer la trajectoire et en déduire la distance parcourue par notre caméra. Notre méthode ne fonctionne que lorsque les points d'intérêt choisis sont coplanaires et n'est donc pas adaptée dans le cas contraire. Elle nécessite également la calibration de la caméra qui peut prendre du temps et durant laquelle on peut rencontrer des erreurs numériques à cause de matrices mal conditionnées. L'utilisation de RANSAC augmente grandement la justesse de nos résultats mais rallonge considérablement le temps de calcul, il existe des variantes de l'algorithme permettant de remédier à ce problème [14]. De plus, la matrice de rotation obtenue n'est pas systématiquement orthogonale et nécessite un post-traitement pour la corriger, quand cela est possible (si la matrice trouvée n'est pas trop loin d'une matrice orthogonale). L'odométrie visuelle en général est également susceptible aux variations rapides de luminosité durant lesquels la caméra a du mal à suivre les points d'intérêt.

V. Conclusion

En conclusion, nous avons vu comment estimer la pose de notre caméra et en déduire la distance parcourue par celle-ci. Néanmoins, on peut vite voir les limitations de cette méthode, elle nécessite d'avoir à tout instant notre mire dans le champ de la caméra. De plus le détecteur de Harris n'est pas invariant au changement d'échelle. Il serait intéressant de choisir d'autres paramètres d'intérêt comme SIFT ou SURF et d'utiliser la matrice essentielle pour généraliser aux points d'intérêt non coplanaires.

References

[1] Davide Scaramuzza and Friedrich Fraundorfer. Visual Odometry [Tutorial]. *IEEE Robotics & Automation Magazine*, 18(4) :80–92, December 2011.

[2] Friedrich Fraundorfer and Davide Scaramuzza. Visual Odometry : Part II : Matching, Robustness, Optimization, and Applications. *IEEE Robotics & Automation Magazine*, 19(2) :78–90, June 2012.

[3] L. Ojeda, D. Cruz, G. Reina, and J. Borenstein. Current-Based Slippage Detection and Odometry Correction for Mobile Robots and Planetary Rovers. *IEEE Transactions on Robotics*, 22(2) :366–378, April 2006.

[4] N. Nourani-Vatani, J. Roberts, and M.V. Srinivasan. Practical visual odometry for car-like vehicles. In *2009 IEEE International Conference on Robotics and Automation*, pages 3551–3557, Kobe, May 2009. IEEE.

[5] D. Scaramuzza and R. Siegwart. Appearance-Guided Monocular Omnidirectional Visual Odometry for Outdoor Ground Vehicles. *IEEE Transactions on Robotics*, 24(5) :1015–1026, October 2008.

[6] Yang Cheng, M. Maimone, and L. Matthies. Visual Odometry on the Mars Exploration Rovers. In *2005 IEEE International Conference on Systems, Man and Cybernetics*, volume 1, pages 903–910, Waikoloa, HI, USA, 2005. IEEE.

[7] Mark Maimone, Yang Cheng, and Larry Matthies. Two years of Visual Odometry on the Mars Exploration Rovers. *Journal of Field Robotics*, 24(3) :169–186, March 2007.

[8] Anqi Xu and Gaurav Namit. *SURF : Speeded-Up Robust Features*.

[9] D.G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, pages 1150–1157 vol.2, Kerkyra, Greece, 1999. IEEE.

[10] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. Fourth Alvey Vision Conference*, pages 147–152, 1988.

[11] Ruihao Li, Sen Wang, Zhiqiang Long, and Dongbing Gu. UnDeepVO : Monocular Visual Odometry through Unsupervised Deep Learning. *arXiv :1709.06841 [cs]*, September 2017.

[12] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11) :1330–1334, Nov./2000.

[13] Random Sample Consensus : A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. page 42.

[14] Rahul Raguram, Jan-Michael Frahm, and Marc Pollefeys. A Comparative Analysis of RANSAC Techniques Leading to Adaptive Real-Time Random Sample Consensus. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *Computer Vision – ECCV 2008*, volume 5303, pages 500–513. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.