Vision par ordinateur: Homographie et Calibration

Jean-Philippe Tardif
Sébastien Roy

Département d'Informatique et de recherche opérationnelle
Université de Montreal

Hiver 2006

Au programme

1. Homographie
   - Homographie
   - Calcul d'homographie
2. Calibration planaire
   - Outil
   - Calibration
   - Optimisation
3. Calibration par Rotation pure

Sommaire

Définition : homographie 2D

Il s'agit d'une transformation linéaire entre deux plans projectifs

C'est-à-dire qu'un ensemble de points 2D projectifs $\mathbf{q}_i$ sur un plan π_1 (dans son système de coordonnée) peuvent être projetés sur un deuxième plan π_2 en des points $\mathbf{p}_i$ donnés par

$$\mathbf{p}_i \propto \mathsf{H}\,\mathbf{q}_i, \quad \text{avec} \quad \mathbf{p}_i, \mathbf{q}_i \in \mathbb{P}^2$$

où

$$\mathsf{H} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

Homographie appliquée aux lignes 2D

Dans sa forme implicite, l'équation d'une ligne 2D peut être représentée par un vecteur $\mathbf{u} = (u_1, u_2, u_3)^\top \in \mathbb{P}^2$. Caractéristique de $\mathbf{u}$:

- Les points $\mathbf{x} \in \mathbb{P}^2$ se trouvant sur la ligne vérifient que $\mathbf{u}^\top \mathbf{x} = 0$
- Une ligne paramétrique $\lambda(a_1, a_2) + (1 - \lambda)(b_1, b_2)$ est donnée en forme vectorielle implicite par $(a_1, a_2, 1) \times (b_1, b_2, 1)$

La transformation d'une ligne dans π_1 vers π_2 est donnée par

$$\mathbf{u}_i \propto \mathsf{H}^{-\top} \mathbf{v}_i$$

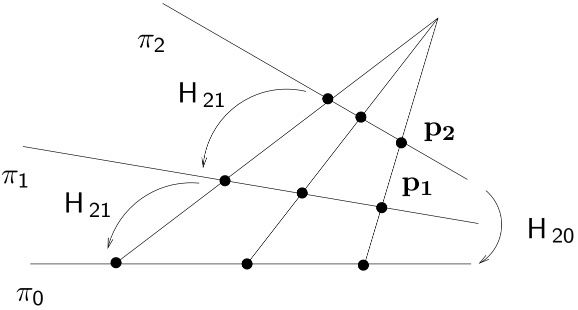
On peut vérifier que :

$$\mathsf{H}\,\mathbf{p} \times \mathsf{H}\,\mathbf{q} \propto \mathsf{H}^{-\top}(\mathbf{p} \times \mathbf{q}) \propto \mathsf{H}^{-\top}(\mathbf{q} \times \mathbf{p})$$

En général H :

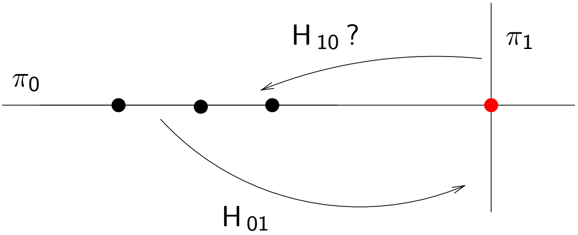
- 8 degrés de liberté, car $\mathrm{H} \equiv \alpha \mathrm{H}$
- Rang 3 (inversible)
- pour deux relations $\mathrm{H}_{01} : \pi_0 \longrightarrow \pi_1$ et $\mathrm{H}_{12} : \pi_1 \longrightarrow \pi_2$, on peut construire $\mathrm{H}_{02} \propto \mathrm{H}_{01}\mathrm{H}_{12} : \pi_0 \longrightarrow \pi_2$

Exemple 1D :



Cas dégénérés

Exemple 1D :



Exercice

- Imaginez les situations dégénérées en 2D.
- Pour chacune d'elles, que se passe-t-il avec H_{01}
- Quelle sont les conséquences sur H_{10} ?

Sommaire

Première étape

Entre deux vues :

- Établir des correspondances entre les images
- Deux solutions :
  - "Tracking" de points saillants au long d'une séquence
  - Mise en correspondances automatique des points (descripteur SIFT [6], invariant affines[7]) (voir aussi http ://www.robots.ox.ac.uk/ vgg/research/affine/index.html)

Avec une grille de calibration :

- Trouver des points saillants
- Retrouver automatiquement la configuration des points (i.e. leur coordonnée)

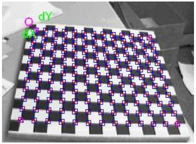


FIG.: Image de [1]

On peut maintenant passer aux calculs

Calcul d'homographie : système linéaire

Nous avons donc une relation d'une plan à l'autre :

$$\mathbf{p} \propto \mathsf{H}\,\mathbf{q}$$

ou bien

$$\alpha\,\mathbf{p} = \mathsf{H}\,\mathbf{q},\ \forall\alpha$$

Une façon commode de comparer deux vecteurs identiques à un facteur d'échelle est de faire :

$$\mathbf{p} \times \alpha\mathbf{p} = \mathbf{0}$$

ce qui nous permet d'utiliser la relation :

$$\mathbf{p} \times \mathsf{H}\,\mathbf{q} = \mathbf{0}$$

Solution simple : système linéaire (suite)

On a en fait un système à trois équations :

$$\begin{bmatrix} 0 & 0 & 0 & -p_3q_1 & -p_3q_2 & -p_3q_3 & p_2q_1 & p_2q_2 & p_2q_3 \\ p_3q_1 & p_3q_2 & p_3q_3 & 0 & 0 & 0 & -p_1q_1 & -p_1q_2 & -p_1q_3 \\ -p_2q_1 & -p_2q_2 & -p_2q_3 & p_1q_1 & p_1q_2 & p_1q_3 & 0 & 0 & 0 \end{bmatrix} \mathbf{H} = \mathbf{0}$$

$$\mathbf{H} = (h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9)^\top$$

(avec la troisième équation redondante.)
Avec plusieurs correspondances, ceci est un système linéaire classique :

$$\mathsf{A}\,\mathbf{x} = \mathbf{0}$$

et l'erreur est $||\epsilon||^2$, avec $\epsilon = \mathsf{A}\,\mathbf{x}$

ATTENTION ceci est une minimisation algébrique ! Il est essentiel de normaliser les données. (centrage et mise à l'échelle)

Solution plus correcte (erreur symétrique)

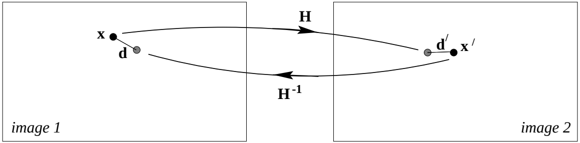


FIG.: Figure de [4].

Idéalement, on aimerait trouver une homographie en minimisant une erreur ayant un sens géométrique plus clair. On voudrait

minimiser la distance entre les points originaux et les points transformés.

Solution plus correcte (suite)

Pour n points, on veut minimiser :

$$\arg\min_{\mathsf{H}} \sum_i dist(\mathbf{p}_i, \mathsf{H}^{-1}\mathbf{p}'_i)^2 + dist(\mathsf{H}\,\mathbf{p}_i, \mathbf{p}'_i)^2$$

où la fonction d est une fonction de distance géométrique des points

$$dist(\mathbf{p}, \mathbf{p}') = ||\frac{(p_1, p_2)}{p_3} - \frac{(p'_1, p'_2)}{p'_3}||$$

Au long, par exemple

$$dist(\mathbf{p}, \mathsf{H}\,\mathbf{p}')^2 = \left(\frac{p_1}{p_3} - \frac{h_1 p'_1 + h_2 p'_2 + h_3 p'_3}{h_7 p'_1 + h_8 p'_2 + h_9 p'_3}\right)^2 + \left(\frac{p_2}{p_3} - \frac{h_4 p'_1 + h_5 p'_2 + h_6 p'_3}{h_7 p'_1 + h_8 p'_2 + h_9 p'_3}\right)^2$$

ce qui n'est pas linéaire !

Une autre façon (erreur de reprojection symétrique)

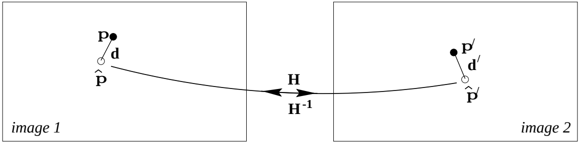


FIG.: Figure de [4].

En plus d'estimer l'homographie, on cherche les points corrigés $\mathbf{p}_i \rightarrow \hat{\mathbf{p}}_i, \mathbf{p}'_i \rightarrow \hat{\mathbf{p}}'_i$ qui vérifie parfaitement H Pour n points, on cherche :

$$\arg \min_{\mathrm{H}, \hat{\mathbf{p}}_i, \hat{\mathbf{p}}'_i} \sum_i^n dist(\mathbf{p}_i, \hat{\mathbf{p}}_i)^2 + dist(\hat{\mathbf{p}}'_i, \mathbf{p}'_i)^2, \quad \text{sujet à} \quad \hat{\mathbf{p}}'_i = \mathrm{H}\,\hat{\mathbf{p}}_i$$

Il faut donc minimiser 4n+9 paramètres. On note que le nombre d'inconnues est très grand, ce qui est un gros inconvénient. Par contre, ceci est un problème **sparse** et on peut résoudre assez efficacement.

Note : Dans le cas des homographies, cette méthode n'améliore pas significativement. Elle est surtout donnée à titre d'exemple, puisqu'elle est utile dans certaines circonstances (matrice fondamentale, calcul de conique...).

Optimisation d'une telle fonction ? Excellente référence : [10]

On pourrait en parler longtemps...
Plusieurs approche pour ce genre de problèmes :

- Méthode Gauss-Newton (voir Levenberg-Marquardt)
- Méthode itérative avec approximation (nous y reviendrons)

Généralement, il faut un estimé des paramètres du problème. Ils sont données par l'algorithme algébrique précédent.

Dans **Mathematica**, il y à la commande *FindMinimum* (attention, ce n'est pas la même chose que *Minimize*, qui est une fonction beaucoup plus complexe).

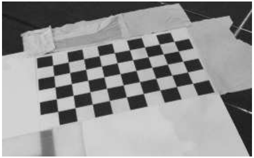
Idée — Références utiles [13, 8, 9, 5, 2, 3, 1]

Nous voulons utiliser l'image d'un plan a géométrie euclidienne connue pour calculer les paramètres internes de notre caméra, ainsi que sa position par rapport au plan pour chaque photo.

Plan euclidien

Cela veut tout simplement dire qu'on connaît un système d'axe euclidien dans ce plan. On connaît par exemple :

- l'angle entre deux lignes
- la distance entre les points
- le fait que l'échiquier est constitué de "carrés"



Sommaire

Modèle de projection → transformation

Jusqu'à maintenant, nous avons utilisé une projection :

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathsf{P}\,\mathbf{X} = \mathsf{K}\,\mathsf{R}_{3\times 3}[\mathtt{I}_{3\times 3}|\mathbf{t}] \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Pour la calibration planaire, on suppose que notre plan de calibration est en $Z = 0$, ce qui veut dire que nos points 3D sont de la forme $\propto (X, Y, 0, 1)^\top$. On peut donc utiliser une homographie, plutôt qu'une matrice de projection. En effet :

$$\mathsf{K}\,\mathsf{R}\,[\mathtt{I}_{3\times 3}|\mathbf{t}] \begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix} = \mathsf{K}\,\mathsf{R} \begin{bmatrix} 1 & 0 & \\ 0 & 1 & \mathbf{t} \\ 0 & 0 & \end{bmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix} = \mathsf{H} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

Voyons quelques outils qui nous seront utiles...

Conique absolue Ω_∞

Définition

Il s'agit d'une conique de points complexes situés dans la plan infini $\pi_\infty = (0, 0, 0, 1)^\top$ (encore une fois, sous forme vectorielle implicite, i.e. un points $\mathbf{x}$ sur π_∞ vérifie que $\mathbf{x}^\top \pi_\infty = 0$ Donc $\mathbf{x}$ est de la forme $(x, y, z, 0)^\top$ et

$$(x, y, z) \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = 0$$

Projection de π_∞

Définition

Il s'agit de la projection de Ω_∞

Pour montrer exactement sa forme explicite, nous devons trouver la transformation entre les points se trouvant sur π_∞ vers l'image de notre caméra. Nous avons :

$$\mathbf{p} = \mathsf{K}\,\mathsf{R}\,[\mathtt{I}|\mathbf{t}] \begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix} = \mathsf{K}\,\mathsf{R} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Qu'est-ce que ça veut dire ?

Eh bien que seul l'orientation de la caméra et ses paramètres internes influencent la projection de ces points.

Analogie : La position du soleil ou des étoiles ne changent pas si on bouge, seulement si on change de direction.

Projection de Ω_∞

Comme Ω_∞ est dans π_∞ on peut travailler avec les plans projectifs

- π_∞
- notre image de caméra

Supposons une conique C dans un plan. Lorsqu'on projette (transforme) le plan avec H, C sera transformée telle que :

$$\mathrm{C}' \propto \mathrm{H}^{-\top} \mathrm{C} \mathrm{H}^{-1}$$

Avec ce résultat, on peut appliquer $\mathrm{H} = \mathrm{K}\mathrm{R}$ à Ω_∞ :

$$(\mathrm{K}\mathrm{R})^{-\top} \mathrm{I} (\mathrm{K}\mathrm{R})^{-1} = \mathrm{K}^{-\top} \mathrm{R} \mathrm{I} \mathrm{R}^{-1} \mathrm{K}^{-1} = \mathrm{K}^{-\top} \mathrm{K}^{-1}$$

Définition : Image of the Absolute Conic IAC

$$\omega = \mathrm{K}^{-\top} \mathrm{K}^{-1}, \quad \text{avec} \quad \mathrm{K}^{-\top} = (\mathrm{K}^{-1})^{\top}$$

Détails sur ω

Au long,

$$\omega \propto \begin{bmatrix} \alpha^2 & 0 & -\alpha^2 c_x \\ 0 & 1 & -c_y \\ -\alpha^2 c_x & -c_y & f^2\alpha^2 + c_x^2\alpha^2 + c_y^2 \end{bmatrix}$$

- On voit qu'une fois que les éléments de la matrice sont connus, on peut facilement retrouver les paramètres.
- On peut aussi le faire automatiquement par décomposition de Cholesky.
- Évidemment, pour le moment, on ne connaît pas ces paramètres, ω est donc encore inconnue :

$$\hat{\omega} \propto \begin{bmatrix} w_1 & 0 & w_2 \\ 0 & w_3 & w_4 \\ w_2 & w_4 & w_5 \end{bmatrix}$$

Sommaire

Revenons à notre transformation H

On cherche donc a calculer les éléments de la matrice $\hat{\omega}$ à partir d'une homographie. On peut vérifier que pour $\mathsf{H} = \mathsf{K}\,\mathsf{R}\,\mathsf{T}$

$$\mathbf{h}_1^\top \omega \mathbf{h}_2 = 0, \text{ et } \mathbf{h}_1^\top \omega \mathbf{h}_1 - \mathbf{h}_2^\top \omega \mathbf{h}_2 = 0$$

où $\mathbf{h}_i$ est la $i^{\text{ème}}$ colonne de H . Nous avons donc des contraintes linéaires :

$$\begin{bmatrix} h_1 h_2 & h_4 h_5 & h_7 h_8 & h_5 h_7 + h_4 h_8 & h_7 h_8 \\ h_1^2 - h_2^2 & h_4^2 - h_5^2 & h_7^2 - h_8^2 & 2h_4 h_7 - 2h_5 h_8 & h_7^2 - h_8^2 \end{bmatrix} \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{pmatrix} = \mathbf{0}$$

Nous pouvons alors retrouver l'IAC et ensuite calculer les paramètres internes.

Exercice

- Combien faut-il d'homographie(s) pour calibrer complètement la caméra.
- Si le ratio d'aspect est 1 et qu'on connaît le point principal, combien en faut-il ?

Calcul de pose

Une fois K connue, on voudrait la position de la caméra. Le problème consiste a décomposer $\mathsf{M} = \mathsf{K}^{-1}\mathsf{H} = \mathsf{R}\,\mathsf{T}$, pour R et T
Deux façons :

- Retrouver la translation avec $\mathsf{H}^\top \omega \mathsf{H} \propto \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ t_x & t_y & t_x^2 + t_y^2 + t_z^2 \end{bmatrix}$ et ensuite retrouver R avec $\mathsf{K}, \mathsf{T}, \mathsf{H}$ en main.
- Retrouver R avec $\left[\mathsf{M}^1 \quad \mathsf{M}^2 \quad \mathsf{M}^1 \times \mathsf{M}^2\right]$, forcer $|\mathsf{R}| = 1$, puis retrouver la translation

Exercice :Notez l'ambiguïté sur le signe t_z

- À quoi correspond chaque signe ?
- Si vous retrouvez deux position de caméras ? Peut-on avoir deux signes différents ?
  - Si oui, pourquoi ?
  - Si non, comment corriger ça ?

Sommaire

Optimisation de l'erreur de reprojection

Un peu comme le calcul d'homographie, la solution linéaire n'est pas optimale au niveau des erreurs de reprojection. On aimerait trouver :

$$\arg \min_{\mathsf{R},\mathsf{T},\mathsf{K}} \sum_{i,n} dist(\mathbf{p}_{i,n}, \mathsf{K}\,\mathsf{R}_i \mathsf{T}_i \mathbf{q}_n)^2$$

où les q_n sont les points dans le plan de calibration (notez qu'il n'y a pas d'index i)

Normalement, on voudrait optimiser un peu comme on le fait pour une homographie ordinaire. Mais il y a un problème ! Il faut imposer que R soit un matrice de rotation. Exemple de paramétrisation :

- $\mathsf{R} = \mathsf{R}_x(\theta)\mathsf{R}_y(\phi)\mathsf{R}_z(\rho)$
- Équation de Rodrigues [12] (que nous avons vue précédemment, axe + norme)

Problème : Cela donne une fonction complexe (surtout pour les dérivés partielles) et donc difficile à optimiser.

Meilleure solution : optimisation itérative

À tout le moins, cette solution est plus simple. Il s'agit de considérer le caractère itératif de l'optimisation.
Par rapport à l'étape précédente, la nouvelle rotation est très **semblable** Cela permet de simplifier notre fonction à chaque itération. Pour la rotation, nous avons :

$$\mathsf{R}^{n+1} = \mathsf{R}'\mathsf{R}^{n}, \quad \text{ou} \quad \mathsf{R}' = \mathsf{R}_x(\theta)\mathsf{R}_y(\phi)\mathsf{R}_z(\rho)$$

avec θ, ϕ, ρ **petits**. On peut alors approximer (mais surtout simplifier) R' avec les transformations :

$$\begin{aligned} \cos x &= 1 \\ \sin x &= x \end{aligned}$$

ce qui donne

$$\mathsf{R}' \approx \begin{bmatrix} 1 & \rho & -\phi \\ \theta\phi - \rho & \theta\rho\phi + 1 & \theta \\ \theta\rho + \phi & \rho\phi - \theta & 1 \end{bmatrix}$$

Meilleure solution : optimisation itérative (suite)

Et si on veut, nous pouvons ensuite approximer encore plus avec

$$\begin{aligned}\theta\phi &\approx 0\\ \theta\rho &\approx 0\\ \phi\rho &\approx 0\end{aligned}$$

ce qui donne

$$\mathrm{R}' \approx \begin{bmatrix} 1 & \rho & -\phi \\ -\rho & 1 & \theta \\ \phi & -\theta & 1 \end{bmatrix}$$

La matrice R^n est donc fixe, et lorsqu'une itération est terminée on met à jour $\mathrm{R}^{n+1} = \mathrm{R}'\mathrm{R}^n$ avec la formule de rotation originale (avec les cos et sin).

Exercice (plutôt technique)

- Chercher un façon d'implémenter cela en **Mathematica**
- Faire une version itérative de **FindMinimum**
- Comprendre pourquoi une telle fonction est moins efficace

Les deux derniers points sont plutôt difficile

Méthode sans grille de calibration (auto-calibration)

Définition : Caméra en rotation pure

Il s'agit d'une caméra qui tourne parfaitement autour de son centre optique $(\mathbf{t} = \mathbf{0})$

Dans cette situation, deux images de caméra peuvent être reliée par une homographie. Comme il n'y a pas d'effet de profondeur, c'est comme si tout ce trouvait sur un plan.

Il est possible de calibrer une caméra dans une telle circonstance. Supposons deux caméras avec $\mathsf{H}_1 = \mathsf{K}_1$ (sans rotation, ni translation) et $\mathsf{H}_2 = \mathsf{K}_2\mathsf{R}$. Ce sont des transformation Monde $\rightarrow$ Image.

On peut alors les relier ensemble en faisant Image$_1$ $\rightarrow$ Monde $\rightarrow$ Image$_2$: $\mathsf{H}_12 = \mathsf{K}_2\mathsf{R}\,\mathsf{K}_1^{-1}$

Construction de mosaïque

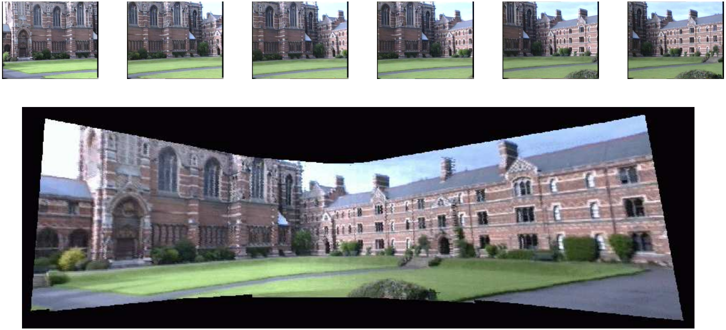


Fig.: Images de [4]

Petit algorithme simple

- Choisir une vue de référence
- Calculer les homographies entre celle-ci et les autres images
- Reprojeter ces images dans la vue de référence

Retrouver la rotation

Rotation conjuguée

Une matrice $\mathsf{H} = \mathsf{T}\,\mathsf{R}\,\mathsf{T}^{-1}$, où R est une matrice orthogonale et T est une transformation projective.

Pour une matrice de rotation R, nous avons

- valeurs propres de la forme $\lambda(1, e^{i\theta}, e^{-i\theta})$, i.e. une réelle et deux complexes conjuguées.
- vecteurs propre correspondants $(\mathbf{a}, \mathbf{I}, \mathbf{J})$

$\mathbf{a}$ donne directement l'axe de rotation de la matrice et θ est l'angle

Pour une rotation conjuguée, ces propriétés sont préservées.

Retrouver la calibration [4, 11]

Un cas simple (ratio d'aspect à 1, point principal connu et mis à l'origine)

Supposons K $= \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}$. Dans ce cas H $= \begin{bmatrix} r_1 & r_2 & fr_3 \\ r_4 & r_5 & fr_6 \\ \frac{r_7}{f} & \frac{r_8}{f} & r_9 \end{bmatrix}$

À partir des propriétés de H (qui est presqu'une matrice de rotation) on peut alors retrouver f avec différentes solutions analytiques. Par ex :

$$h_1^2 + h_2^2 + h_3^2/f^2 = h_4^2 + h_5^2 + h_6^2/f^2$$

ou bien

$$h_1 h_4 + h_2 h_5 + \frac{h_3 h_6}{f^2} = 0$$

Jean-Yves Bouguet
Camera Calibration Toolbox for Matlab
URL http ://www.vision.caltech.edu/bouguetj/calib_doc/index.html

P.Gurdjos et R.Payrissat.
Plane-based Calibration of a Camera with Varying Focal Length : the Centre Line Constraint.
BMVC 2001.

P. Gurdjos, A. Crouzil et R. Payrissat.
Another Way of Looking at Plane-Based Calibration : the Centre Circle Constraint.
ECCV 2002.

R. Hartley and A. Zisserman
Multiple View Geometry in Computer Vision
Cambridge University Press 2000.

Intel Open Source Computer Vision Library.
URL http :// www.intel.com/research/mrl/research/opencv/.

David G. Lowe.
Distinctive image features from scale-invariant keypoints,
International Journal of Computer Vision, 60, 2 (2004), pp. 91-110.

K. Mikolajczyk and C. Schmid,
Scale and Affine invariant interest point detectors.
In IJCV 1(60) :63-86, 2004.

P. Sturm.
Algorithms for plane-based pose estimation.
CVPR 2000.

P. Sturm, S. Maybank.
On Plane-Based Camera Calibration.
CVPR 1999.

B. Triggs, P. McLauchlan, R. Hartley, A. Fitzgibbon.

Bundle Adjustment, A Modern Synthesis.
Vision Algorithms 1999.

Error Analysis of Pure Rotation-Based Self-Calibration
Lei Wang, Sing Bing Kang, Heung-Yeung Shum and Guangyou Xu
PAMI 2004.

Eric W. Weisstein et al.
"Rodrigues' Rotation Formula." From MathWorld–A Wolfram Web Resource.
http ://mathworld.wolfram.com/RodriguesRotationFormula.html

Z. Zhang.
A Flexible New Technique for Camera Calibration.
PAMI, 22(11), 1330–1334, 2000.