Computer Vision
N-View Reconstruction

dsai.asia

Asia Data Science and Artificial Intelligence Master's Program

Co-funded by the Erasmus+ Programme of the European Union

Readings

Readings for these lecture notes:

- Hartley, R., and Zisserman, A. (2004), *Multiple View Geometry in Computer Vision*, Cambridge University Press, Chapters 18–19.
- Triggs, B., McLauchlan, P., Hartley, R., and Fitzgibbon, A. (1999), Bundle adjustment — A modern synthesis, *Vision Algorithms: Theory and Practice*, Springer-Verlag.
- Tomasi and Kanade, Sturm and Triggs, Pollefeys et al., Davison et al., Klein and Murray, Lourakis and Argyros, Kerl, Sturm, and Cremers.

Outline

We have seen two-view projective and metric reconstruction techniques. As we move to many views, however, what can we do?

In this part we consider how to obtain a sparse reconstruction given a sequence of images.

This is where Hartley and Zisserman's book becomes a little obsolete, as there have been many new developments since 2004.

A brief tour of the history:

1. 1970's: The term bundle adjustment emerges in the photogrammetry literature, referring to simultaneous optimization of parameters of a set of cameras and a set of points observed by those cameras.
2. 1992: Tomasi and Kanade show how to use SVD to factor the observation matrix to estimate a sequence of cameras and collection of 3D points. Limited to affine cameras with no missing points.
3. 1996: Sturm and Triggs show how to use iterative factorization to obtain a projective reconstruction. Other factorization methods refine the technique.
4. 2004: Pollefeys et al. combine keyframe selection, SfM, BA, resectioning, loop closure, autocalibration, and 3D mesh techniques to obtain textured 3D models from videos obtained with hand-held cameras.

Most offline 3D reconstruction methods use a pipeline similar to Pollefeys' approach, with manual intervention.

More recently, SfM (computer vision) and SLAM (robotics) techniques are starting to converge.

1. 2006: Davison et al. introduce the first real-time monocular SLAM method, called MonoSLAM.
2. 2007: Klein and Murray introduce PTAM (Parallel Tracking and Mapping) aimed at augmented realilty applications.
3. 2009: Lourakis and Argyros introduce SBA (Sparse BA), an efficient open source bundle adjustment library, bringing fast BA to the masses.
4. 2013: Kerl, Sturm, and Cremers introduce DVO SLAM (Dense Visual SLAM) for RBG-D cameras.

As compute power increases, we are seeing more incremental real time methods with excellent results.

1. 2014: Engel, Schöps, and Cremers introduce SVO, a semi-direct method for monocular visual odometry.
2. 2014: Engel, Stückler, and Cremers (2015) introduce LSD SLAM (Large-Scale Direct Monocular SLAM), the first dense monoSLAM method.
3. 2015: Mur-Artal, Montiel, and Tardós introduce ORB-SLAM, the most robust feature-based monoSLAM method today, combining the basic approach of PTAM with ORB features, BA, and loop closure techniques.

LSD SLAM brings the idea of dense photometric alignment from RGBD to RGB, providing richer maps than ORB-SLAM at higher computational cost.

Today, work is continuing on improving the robustness and efficiency of visual SLAM systems.

Several methods now obtain real time results on smartphones or embedded platforms like the Odroid XU4 or NVidia TX2.

However, it is still very difficult for state-of-the-art methods to keep track of a set of features during rotations and fast relative motion.

Visual-inertial SLAM systems attempt to combine IMU readings (linear acceleration and rotational velocity) with vision.

Knowing approximately how the camera has moved since the last keyframe gives us a better idea of where to look for features in the next frame.

Some visual-inertial SLAM systems:

- Christian Forster's Ph.D. thesis (2016) demonstrates combination of SVO with IMU preintegration to achieve very accurate VISLAM. No open source implementation.
- 2016: Forster, Zhang, Gassner, Welberger, and Scaramuzza introduce SVO-2, a faster, more accurate version of SVO incorporating Forster's IMU priors. Implementation is commercial (no open source).
- Raúl Mur-Artal, and Juan D. Tardós (2017) introduce VI-ORB, a visual-inertial version of ORB-SLAM using Forster's IMU preintegration. The authors did not release an open source version, but there is a community developed version on github.
- Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart and Paul Timothy Furgale (2015) introduce OKVIS, "Keyframe-based visual–inertial odometry using nonlinear optimization." Open-source version maintained by the author available on github.

Interesting project for this class: VI-ORB or OKVIS experiments on Pixhawk or Android...

Outline

Bundle adjustment

The idea

Given a set of unknown 3D points X_j viewed by a set of cameras with unknown projection matrices $\mathtt{P}^i$ at image points x_j^i, we seek to find the camera matrices $\mathtt{P}^i$ and 3D points X_j minimizing the reprojection error

$$\min_{\hat{\mathtt{P}}^i, \hat{\mathrm{X}}_j} \sum_{ij} d(\hat{\mathtt{P}}^i \hat{\mathrm{X}}_j, \mathrm{x}_j^i)^2$$

Iterative minimization of this cost function is called bundle adjustment because it involves adjusting the bundle of rays between each camera center and the set of 3D points.

As formulated above, we need outlier removal prior to nonlinear least squares optimization. However, Triggs, McLauchlan, Hartley, and Fitzgibbon (1999) argue for a formulation with a more general cost function allowing robust estimation with the outliers included.

There are two big problems with bundle adjustment:

- It needs a good starting point to begin optimization.
- There can be a large number of parameters involved in the minimization. For n points viewed by m cameras we have $3n + 11m$ parameters. This makes the matrices used by Levenberg-Marquardt prohibitively large.[1]

[1] Remember the linear system we have to solve on each iteration of LM?

$$\left[\mathrm{J}_{\mathrm{f}}^{T}(\mathrm{P})\mathrm{J}_{\mathrm{f}}(\mathrm{P}) + \mu\mathrm{I}\right]\delta\mathrm{P} = -\mathrm{J}_{\mathrm{f}}^{T}(\mathrm{P})\mathrm{f}(\mathrm{P}).$$

Solutions of the first problem (the initial solution) generally involve linear methods such as factorization.

Solutions to the second problem (large parameter matrix):

- Reduce n and/or m, by using a subset of the points or partitioning the views [suboptimal].
- Interleave estimation of camera matrices with estimation of 3D points [guaranteed to converge but slow].
- Use a sparse minimization routine:
  - Download the `sba` (sparse bundle adjustment) open source software from `http://www.ics.forth.gr/~lourakis/sba/`
  - (For Debian) install the `liblapack-dev` package, and add `-llapack` to your gcc command line.
  - Call `sba_motstr_levmar()` to estimate motion ($\mathtt{P}_i$'s) and structure ($\mathtt{X}$).

Outline

Tomasi and Kanade's algorithm is a maximum likelihood reconstruction in the case of affine cameras.

It requires that all points be seen in all views.

For affine cameras we write $\mathrm{x} = (x, y)^T$ and $\mathrm{X} = (X, Y, Z)^T$. Then we have the projection equation

$$\mathrm{x} = \mathtt{M}\mathrm{X} + \mathrm{t}$$

We seek cameras $\{\mathtt{M}^i, \mathrm{t}^i\}$ and 3D points X_j such that the distance between estimated and predicted image points is minimized:

$$\min_{\mathtt{M}^i, \mathrm{t}^i, \mathrm{X}_j} \sum_{ij} (\mathrm{x}_j^i - \hat{\mathrm{x}}_j^i)^2 = \min_{\mathtt{M}^i, \mathrm{t}^i, \mathrm{X}_j} \sum_{ij} \left(\mathrm{x}_j^i - (\mathtt{M}^i \mathrm{X}_j + \mathrm{t}^i)\right)^2.$$

If we choose the centroid of the points to be the origin of the coordinate system, then we can estimate the translation vectors t^i easily.

Under affine projection, the origin of the coordinate system is projected to $(0,0)^T$ in the image. This means that t needs to translate the projection of the origin to the mean of the observed image points:

$$\mathrm{t}^i = \frac{1}{n}\sum_j \mathrm{x}_j^i.$$

To simplify the remaining calculations, we set $\mathrm{t}^i = 0$ and adjust the points x_i^j accordingly.

Factorization

Affine factorization

Now we arrange the adjusted image points in the $2m \times n$ measurement matrix

$$\mathtt{W} = \begin{bmatrix} \mathrm{x}_1^1 & \mathrm{x}_2^1 & \cdots & \mathrm{x}_n^1 \\ \mathrm{x}_1^2 & \mathrm{x}_2^2 & \cdots & \mathrm{x}_n^2 \\ \dots & \dots & \ddots & \dots \\ \mathrm{x}_1^m & \mathrm{x}_2^m & \cdots & \mathrm{x}_n^m \end{bmatrix}.$$

We want to find $\mathtt{M}^i$ and X_j such that

$$\mathtt{W} = \begin{bmatrix} \mathtt{M}^1 \\ \mathtt{M}^2 \\ \dots \\ \mathtt{M}^m \end{bmatrix} \begin{bmatrix} \mathrm{X}_1 & \mathrm{X}_2 & \cdots \mathrm{X}_n \end{bmatrix}.$$

Since $\mathtt{M}$ and $\mathtt{X}$ are rank 3, their product is rank 3.

Since in general $\mathtt{W}$ will not be rank 3 due to measurement noise, we replace it with the best possible reconstruction, the matrix $\hat{\mathtt{W}}$ which is rank 3 and closest to $\mathtt{W}$ in Frobenius norm.

Such a matrix $\hat{\mathtt{W}}$ can be computed easily using the SVD $\mathtt{UDV}^T = \mathtt{W}$ by truncating $\mathtt{U}$ and $\mathtt{V}$ to three columns to get $\hat{\mathtt{U}}$ and $\hat{\mathtt{V}}$, then truncating $\mathtt{D}$ to a 3×3 matrix $\hat{\mathtt{D}}$, then finally letting $\hat{\mathtt{W}} = \hat{\mathtt{U}}\hat{\mathtt{D}}\hat{\mathtt{V}}^T$.

Since $\hat{\mathtt{W}}$ minimizes $\|\mathtt{W} - \hat{\mathtt{W}}\|_F$, $\hat{\mathtt{M}} = \hat{\mathtt{U}}$ and $\hat{\mathtt{X}} = \hat{\mathtt{V}}^T$ is a maximum likelihood reconstruction.

So we see that a straightforward application of the SVD gives us an optimal reconstruction in the case of affine cameras.

The affine factorization method doesn't work for projective reconstruction, so any serious projective distortion will introduce error into the reconstruction.

Sturm and Triggs (1996), however, pointed out that if we knew the projective depth of each point, then the structure and motion (camera matrices) could be estimated correctly by factorization.

We have $\mathrm{x}_j^i = \mathtt{P}^i \mathrm{X}_j$. With a projective camera we have homogeneous points and there is an implicit constant factor which we can make explicit as $\lambda_j^i \mathrm{x}_j^i = \mathtt{P}^i \mathrm{X}_j$ with $\mathrm{x}_j^i = (x_j^i, y_j^i, 1)^T$.

If we write the depths explicitly and all points are visible in all images, we can write the problem in terms of a scaled measurement matrix:

$$\mathtt{W} = \begin{bmatrix} \lambda_1^1 \mathbf{x}_1^1 & \lambda_2^1 \mathbf{x}_2^1 & \cdots & \lambda_n^1 \mathbf{x}_n^1 \\ \lambda_1^2 \mathbf{x}_1^2 & \lambda_2^2 \mathbf{x}_2^2 & \cdots & \lambda_n^2 \mathbf{x}_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_1^m \mathbf{x}_1^m & \lambda_2^m \mathbf{x}_2^m & \cdots & \lambda_n^m \mathbf{x}_n^m \end{bmatrix} = \begin{bmatrix} \mathtt{P}^1 \\ \mathtt{P}^2 \\ \vdots \\ \mathtt{P}^m \end{bmatrix} \begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \mathbf{X}_n \end{bmatrix}$$

Since $\mathtt{P}$ and $\mathtt{X}$ are each rank 4, $\mathtt{W}$ will be rank 4 and the rank 4 factorization from the SVD will give a valid $\mathtt{P}$ and $\mathtt{X}$.

The key to projective factorization is how to choose the projective depths[2] λ^i_j?

- We could use another reconstruction method to estimate λ^i_j.
- We can also initialize them arbitrarily, e.g. $\lambda^i_j = 1$, then interleave factorization with depth estimation.

Though there is no guarantee that the iterative method will converge to a global minimum, it is widely used in practice.

[2] The λ^i_j are called projective depths because in a Euclidean frame they would be the lengths of the projections of the scene points onto the cameras' principal axes.

The algorithm works well in practice, but it is important to know what cost function is it minimizing.

By using rank 4 decomposition, it turns out the algorithm is minimizing

$$\|\mathtt{W} - \hat{\mathtt{W}}\|_F^2 = \sum_{ij} \|\lambda_j^i \mathrm{x}_j^i - \hat{\lambda}_j^i \hat{\mathrm{x}}_j^i\|^2 = \sum_{ij} (\lambda_j^i x_j^i - \hat{\lambda}_j^i \hat{x}_j^i)^2 + (\lambda_j^i y_j^i - \hat{\lambda}_j^i \hat{y}_j^i)^2 + (\lambda_j^i - \hat{\lambda}_j^i)^2$$

When the λ_j^i are close to each other, we see that the cost function is an approximation to a scaled geometric distance.

Because the projective reconstruction cost function involves λ^i_j, and the projective factorization method minimizes geometric error when $\forall i, j, \lambda^i_j = 1$, we would like to keep the λ^i_j as close to 1 as possible.

If we scale $\mathtt{P}$ and $\mathtt{X}$, we obtain

$$(\alpha^i \beta_j \lambda^i_j)\mathrm{x}^i_j = (\alpha^i \mathtt{P}^i)(\beta_j \mathrm{X}_j)$$

which means we can replace the projective depths by multiplying the ith row or jth column of $\mathtt{W}$ by an arbitrary factor.

One normalization method producing good results in practice is to renormalize, on every iteration, the rows and columns of $\mathtt{W}$ so they have unit norm.

As always, the image points should be normalized by isotropic scaling before beginning.

Projective factorization: Objective

Given a set of n image points seen in m views:

$$\mathrm{x}_j^i;\, i = 1, \ldots, m, \quad j = 1, \ldots, n$$

compute a projective reconstruction.

Projective factorization: Algorithm

(i) Normalize the image data using isotropic scaling.

(ii) Set projective depths $\lambda_j^i = 1$ or use some other method to estimate them.

(iii) Normalize the depths λ_j^i by multiplying rows and columns by constant factors. One way is to do a pass setting the norm of each row to 1 then a pass setting the norm of each column to 1.

(iv) Form the $3m \times n$ scaled measurement matrix $\mathtt{W}$ and obtain $\mathtt{P}^i$ and $\mathtt{X}_j$ from the rank 4 factorization of $\mathtt{W}$.

(v) Reproject the points into each image to obtain new estimates of the depths and repeat from step (iii) until convergence.

Factorization methods for reconstruction have received a great deal of attention over the last 10 years.

See Tang and Hung (2006) for a method based on the projective factorization idea that allows missing points and is also guaranteed to converge to a local minimum.

Outline

The factorization methods just outlined are batch methods.

They cannot be used to perform on-line 3D estimation without modification.

The resectioning method first gets an initial 3D reconstruction using, for example, the SVD of $\mathtt{E}$, then repeats the following steps for frame i:

- Get correspondences between already-estimated 3D points $\{\mathrm{X}_j\}_{j \in 1 \ldots n}$ and new 2D points $\{\mathrm{x}_k\}_{k \in 1 \ldots m}$.
- Use the correspondences to estimate $\mathtt{P}^i$. This is called resectioning.
- Use triangulation to estimate new 3D points using correspondences between frame i and frame $i-1$.

The main challenges in resectioning are accumulated error, outliers, and degenerate conditions.

To reduce accumulated error, we can periodically use bundle adjustment to find a globally consistent, minimum error configuration of the cameras and points.

To mitigate the effect of outliers, we use RANSAC or other robust estimators and outlier rejection.

To avoid degenerate conditions, we apply keyframe selection, in which we choose the images we use for reconstruction and resectioning specifically to avoid degenerate configurations of the cameras and points.

This leads to the general algorithm, on the next page.

Resectioning method

The algorithm

Reconstruct from an image sequence: Algorithm II

(i) Compute interest points in each image using, e.g., SIFT.

(ii) Extract keyframes from the image sequence in which significant motion separates successive keyframes.

(iii) Obtain 2-frame reconstruction from keyframes 1 and 2.

(iv) Perform resectioning on remaining images.

(v) Bundle adjust the cameras and 3D structure for the complete keyframe sequence to obtain a maximum likelihood projective reconstruction.

Resectioning method

Keyframe selection

Keyframe selection improves the runtime performance of 3D reconstruction, helps improve accuracy, and helps avoid degeneracy.

Ahmed, Dailey, Landabaso, and Herrero (2010)[3] experimented with a variety of criteria for robust key frame extraction.

Given the first keyframe, we apply correspondence ratio and degeneracy avoidance constraints to eliminate inappropriate candidate keyframes.

Then we select the best successor keyframe from the remaining candidate keyframe set by maximizing an objective function.

[3] Ahmed, M.T., Dailey, M.N., Landabaso, J.L., and Herrero, N. (2010), Robust key frame extraction for 3D reconstruction from video streams. In *International Conference on Computer Vision Theory and Applications (VISAPP)*.

Resectioning method

Keyframe selection

The correspondence ratio R_c is a proxy for the baseline:

$$R_c = \frac{T_c}{T_f},$$

where T_c is the number of inlier correspondences and T_f is total number of features found.

Low values of R_c indicate low overlap between two frames, in turn indicating long baselines.

Long baselines are desired for triangulation accuracy, but if the number of corresponding points is insufficient, camera motion estimation accuracy suffers.

Thus, the correspondence ratio is constrained to lie between an upper threshold T_1 and a lower threshold T_2.

Resectioning method

Keyframe selection

To avoid degenerate cases, the geometric robust information criterion (GRIC) score is used to measure the goodness of fit for a homography or the fundamental matrix.

$$GRIC = \sum_i \rho(e_i^2) + \lambda_1 dn + \lambda_2 k,$$

where $i = 1 \ldots n$, $\rho(e_i^2)$ is a robust function

$$\rho(e_i^2) = min(\frac{e_i^2}{\sigma^2}, \lambda_3(r - d))$$

of the residual e_i over the n correspondences, σ is the assumed standard deviation of the error, d is the model dimension, k is the model degrees of freedom, r is the dimension of the data, $\lambda_1 = log(r)$, $\lambda_2 = log(rn)$, and λ_3 limits the residual error.

The GRIC constraint is to reject any candidate keyframes for which the homography model has a lower GRIC score than the fundamental matrix.

Resectioning method

Keyframe selection

After filtering by correspondence ratio and degeneracy constraints, we select as the next keyframe the frame that maximizes certain selection criteria incorporating

- Normalized GRIC difference:

$$f_G(i,j) = \frac{GRIC_H(i,j) - GRIC_F(i,j)}{GRIC_H}.$$

  The difference is maximized when the fundamental matrix model is much better than the homography model.

- Point-to-epipolar line cost (PELC):

$$f_{GP}(i,j) = w_G f_G(i,j) + w_P(\sigma - PELC(i,j)),$$

  where i is the current keyframe and j is a candidate next keyframe. PELC tends to be high when correspondences are not accurate.

The camera matrix $\mathtt{P}^i$ is computed from 3D-2D correspondences and the DLT algorithm.

In practice, 3D reconstruction using long video sequence suffers from accumulated error. Some issues are:

- Matching existing 3D to 2D points in an additional frame is not perfect. Sometimes, already-existing features are treated as new points. This arises when the camera moves back and forth or when a point becomes occluded and then visible again.
- Resectioning is highly sensitive to outliers. The camera estimate can be completely wrong if we estimate $\mathtt{P}^i$ in the presence of outliers.

Resectioning method

Example from AIT Vision and Graphics Lab

Atima Tharatipyakul worked on solving these problems by finding 2D-3D point pairs using multiple frames then using RANSAC in camera estimation to discard 2D-3D correspondences outliers.[4]

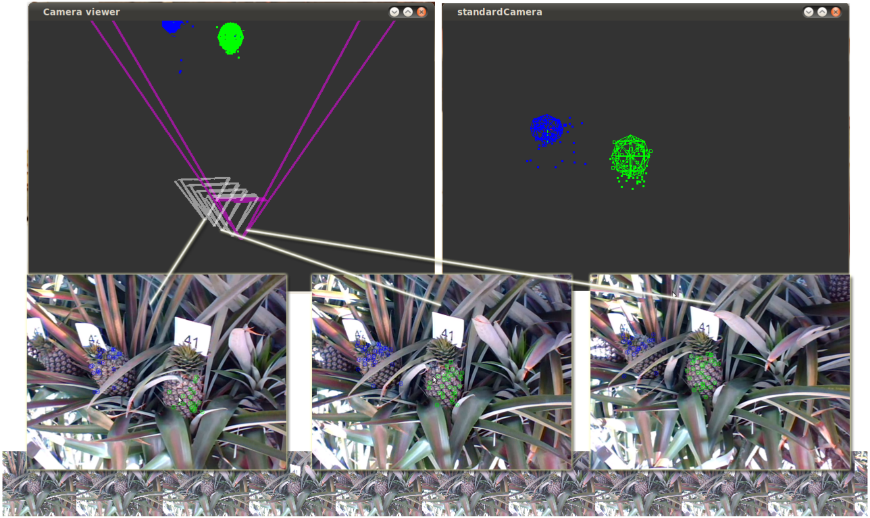
Next slide: example results from resectioning method in Atima's work, with 5 keyframes selected from a sequence of images containing pineapples.

3 of the keyframes are shown, along with camera estimates, point cloud estimates, and fruit ellipsoid estimates.

[4] Tharatipyakul, A,, *3D Visualization from Video Sequence for Agricultural Field Inspection Robot*, Master's thesis, Asian Institute of Technology, 2011.

Resectioning method

Example from AIT Vision and Graphics Lab



Outline

Metric upgrade and auto-calibration

Introduction

As we already know, if $\mathtt{K}$ is known we can directly obtain metric reconstructions. For two frames:

- For first pair of keyframes, estimate $\mathtt{F}$.
- Calculate $\mathtt{E}$ from $\mathtt{F}$ and $\mathtt{K}$.
- Factor $\mathtt{E}$ to obtain $\mathtt{P}'$ as described in H&Z Section 9.6.

For N frames, we factor $\mathtt{E}$ for the first pair of keyframes, then use incremental resectioning and bundle adjustment.

If $\mathtt{K}$ is unknown or changing during the image sequence, however, we need a method for auto-calibration.

Here we look at some of the techniques for auto-calibration from image correspondences over an image sequence.

The idea of auto-calibration:

- Obtain a projective factorization $\mathtt{W} = \mathtt{P}\mathtt{X}$.
- Estimate a homography $\mathtt{H}$ such that $\mathtt{W} = (\mathtt{P}\mathtt{H})(\mathtt{H}^{-1}\mathtt{X})$ is a metric reconstruction.
- $\mathtt{P}^i\mathtt{H}$ is metric if it can be decomposed as $\mathtt{K}^i[\mathtt{R}^i \mid \mathtt{t}^i]$ where the $\mathtt{K}^i$ are consistent with a priori constraints (the same across the image sequence, only focal length changing, etc.).

There are two main approaches to auto-calibration for general motion:

- Direct estimation of $\mathtt{H}$.
- Stratified reconstruction, beginning with an affine reconstruction (which identifies $\boldsymbol{\pi}_\infty$) followed by metric reconstruction.

Here we only consider direct methods, though stratified methods have some advantages, mainly that there is a linear solution for metric reconstruction once $\boldsymbol{\pi}_\infty$ is known (see text for details).

We seek $\mathtt{H}$ such that $\mathtt{P}^i_M = \mathtt{P}^i\mathtt{H} = \mathtt{K}^i[\mathtt{R}^i \mid \mathtt{t}^i]$ for $i = 1, \ldots, m$.

Since we don't care about the absolute frame, we assume $\mathtt{P}^1 = [\mathtt{I} \mid 0]$ and that therefore $\mathtt{P}^1_M = \mathtt{K}^1[\mathtt{I} \mid 0]$.

In general, $\mathtt{H}$ takes the form

$$\mathtt{H} = \begin{bmatrix} \mathtt{A} & \mathtt{t} \\ \mathtt{v}^T & k \end{bmatrix}.$$

Since $\mathtt{P}^1_M = \mathtt{P}^1\mathtt{H} = [\mathtt{I} \mid 0]$, we can infer that $\mathtt{A} = \mathtt{K}^1$ and $\mathtt{t} = 0$.

Since $\mathtt{H}$ is necessarily non-singular we can assume $k = 1$ to obtain

$$\mathtt{H} = \begin{bmatrix} \mathtt{K}^1 & 0 \\ \mathtt{v}^T & 1 \end{bmatrix}.$$

The plane at infinity $\boldsymbol{\pi}_\infty$ is $(0,0,0,1)^T$ in an affine or metric frame, and $\mathtt{H}^{-T}$ is the point/plane transform from the metric frame to the projective frame. This means we can derive

$$\boldsymbol{\pi}_\infty = \begin{pmatrix} -(\mathtt{K}^1)^{-T}\mathrm{v} \\ 1 \end{pmatrix}.$$

Finally we can write

$$\mathtt{H} = \begin{bmatrix} \mathtt{K} & 0 \\ -\mathrm{p}^T\mathtt{K} & 1 \end{bmatrix}, \boldsymbol{\pi}_\infty = (\mathrm{p}^T, 1)^T.$$

For the rest of the cameras ($i = 2, \ldots, m$), we write $\mathtt{P}^i = [\mathtt{A}^i \mid \mathrm{a}^i]$.

Using $\mathtt{P}^i_M = \mathtt{P}^i\mathtt{H}$, we can obtain

$$\mathtt{K}^i\mathtt{R}^i = (\mathtt{A}^i - \mathrm{a}^i\mathrm{p}^T)\mathtt{K}^1$$

and (since $\mathtt{R}^i$ is orthogonal),

$$\mathtt{K}^i\mathtt{K}^{iT} = (\mathtt{A}^i - \mathrm{a}^i\mathrm{p}^T)\mathtt{K}^1\mathtt{K}^{1T}(\mathtt{A}^i - \mathrm{a}^i\mathrm{p}^T)^T.$$

These are the basic auto-calibration equations. If we know $\mathtt{K}^1$, p, and $\mathtt{P}^i$, we can calculate $\mathtt{P}^i_M$.

The matrix $\mathtt{K}^i\mathtt{K}^{iT}$ is the dual image of the absolute conic (DIAC) ω^{*i}.

The DIAC is the projection of the absolute dual quadric $\mathtt{Q}_\infty^*$:

$$\mathtt{K}^i\mathtt{K}^{iT} = \omega^{*i} = \mathtt{P}^i\mathtt{Q}_\infty^*\mathtt{P}^{iT}$$

If we know $\mathtt{Q}_\infty^*$, we can calculate $\mathtt{K}^i$ directly (by Cholesky decomposition of $\mathtt{P}^i\mathtt{Q}_\infty^*\mathtt{P}^{iT}$.

Important facts:

- The absolute conic Ω_∞ is a conic on π_∞ containing the intersection of all circles and spheres with π_∞.
- The absolute dual quadric $\mathtt{Q}^*_\infty$ is a rank 3 dual quadric whose envelope is the set of planes tangent to Ω_∞.
- The absolute dual quadric is invariant under similarity transforms and is just diag(1,1,1,0) in a metric frame.
- The absolute dual quadric's null space is π_∞.

Auto-calibration based on $\mathtt{Q}^*_\infty$: Objective

Given a set of matched points across several views and constraints on the calibration matrices $\mathtt{K}^i$, compute a metric reconstruction of the points and cameras.

Auto-calibration based on $\mathtt{Q}^*_\infty$: Algorithm

(i) Compute a projective reconstruction from a set of views, resulting in cameras $\mathtt{P}^i$ and points $\mathtt{X}$.

(ii) Use $\omega^{*i} = \mathtt{P}^i \mathtt{Q}^*_\infty \mathtt{P}^{iT}$ along with constraints to estimate $\mathtt{Q}^*_\infty$.

(iii) Decompose $\mathtt{Q}^*_\infty$ as $\mathtt{H}\tilde{\mathtt{I}}\mathtt{H}^T$ where $\tilde{\mathtt{I}}$ =diag(1,1,1,0).

(iv) Apply $\mathtt{H}^{-1}$ to $\mathtt{X}$ and $\mathtt{H}$ to $\mathtt{P}^i$ to obtain a metric reconstruction of the point and cameras.

(v) Use iterative least squares to improve the solution.

As a nice example of linear constraints on $\mathtt{Q}^*_\infty$, see Pollefeys et al. (2004), Visual modeling with a hand-held camera, *IJCV* 59(3).

As a nice example of more sophisticated methods to enforce the rank 3, positive semidefiniteness, and "chirality" contraints on $\mathtt{Q}_\infty$, see Chandraker et al. (2007), Autocalibration via rank-constrained estimation of the absolute dual quadric, In *Proceedings of CVPR*.