Structure from Motion

CSE 6367: Computer Vision

Instructor: William J. Beksi

Introduction

- We've seen how 2D and 3D point sets can be aligned and how such alignments could be used to estimate both a camera's pose and its internal calibration parameters

Introduction

- We've seen how 2D and 3D point sets can be aligned and how such alignments could be used to estimate both a camera's pose and its internal calibration parameters
- We now look at the converse problem of estimating the locations of 3D points from multiple images given only a sparse set of correspondences between image features

Introduction

- We've seen how 2D and 3D point sets can be aligned and how such alignments could be used to estimate both a camera's pose and its internal calibration parameters
- We now look at the converse problem of estimating the locations of 3D points from multiple images given only a sparse set of correspondences between image features
- This process often involves simultaneously estimating both 3D geometry (structure) and camera pose (motion) and is commonly known as **structure from motion** (SfM)

Introduction



- 3D reconstruction using structure from motion

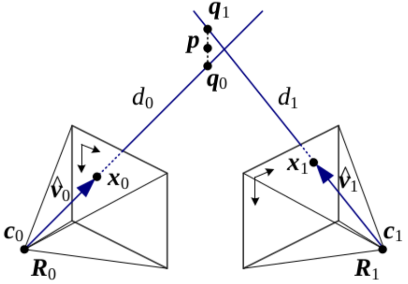
Solving the Triangulation Problem

- The problem of determining a point's 3D position from a set of corresponding image locations and known camera positions is known as **triangulation**

Solving the Triangulation Problem

- The problem of determining a point's 3D position from a set of corresponding image locations and known camera positions is known as **triangulation**
- One of the simplest ways to solve this problem is to find the 3D point $\mathbf{p}$ that lies closest to all of the 3D rays corresponding to the 2D matching feature locations $\{\mathbf{x_j}\}$ observed by cameras $\{P_j = K_j[R_j \mid \mathbf{t}_j]\}$, where $\mathbf{t}_j = -R_j\mathbf{c}_j$ and $\mathbf{c}_j$ is the jth camera center

Solving the Triangulation Problem



- 3D point triangulation by finding the point $\mathbf{p}$ that lies nearest to all of the optical rays $\mathbf{c}_j + d_j\hat{\mathbf{v}}_j$

Solving the Triangulation Problem

- These rays originate at $\mathbf{c}_j$ in a direction $\hat{\mathbf{v}}_j = \mathcal{N}(R_j^{-1}K_j^{-1}\mathbf{x}_j)$

Solving the Triangulation Problem

- These rays originate at $\mathbf{c}_j$ in a direction $\hat{\mathbf{v}}_j = \mathcal{N}(R_j^{-1} K_j^{-1} \mathbf{x}_j)$
- The nearest point to $\mathbf{p}$ on this ray, which we denote by $\mathbf{q}_j$, minimizes the distance

$$||\mathbf{c}_j + d_j \hat{\mathbf{v}}_j - \mathbf{p}||^2$$

which has a minimum at $d_j = \hat{\mathbf{v}}_j \cdot (\mathbf{p} - \mathbf{c}_j)$

Solving the Triangulation Problem

- Therefore,

$$\mathbf{q}_j = \mathbf{c}_j + (\hat{\mathbf{v}}_j \hat{\mathbf{v}}_j^T)(\mathbf{p} - \mathbf{c}_j) = \mathbf{c}_j + (\mathbf{p} - \mathbf{c}_j)_{||}$$

and the squared distance between **p** and **q** is

$$r_j^2 = ||(\mathbf{I} - \hat{\mathbf{v}}_j \hat{\mathbf{v}}_j^T)(\mathbf{p} - \mathbf{c_j})||^2 = ||(\mathbf{p} - \mathbf{c}_j)_\perp||^2$$

Solving the Triangulation Problem

- The optimal value for $\mathbf{p}$, which lies closest to all of the rays, can be computed as a regular least squares problem by summing over all the r_j^2 and finding the optimal value of $\mathbf{p}$,

$$\mathbf{p} = \left[\sum_j (\mathbf{I} - \hat{\mathbf{v}}_j \hat{\mathbf{v}}_j^T) \right]^{-1} \left[\sum_j (\mathbf{I} - \hat{\mathbf{v}}_j \hat{\mathbf{v}}_j^T) \mathbf{c}_j \right]$$

Solving the Triangulation Problem

- An alternative formulation, which is more statistically optimal and which can produce significantly better estimates if some of the cameras are closer to the 3D point than others, is to minimize the residual in the measurement equations

$$x_j = \frac{p_{00}^{(j)}X + p_{01}^{(j)}Y + p_{02}^{(j)}Z + p_{03}^{(j)}W}{p_{20}^{(j)}X + p_{21}^{(j)}Y + p_{22}^{(j)}Z + p_{23}^{(j)}W}$$

$$y_j = \frac{p_{10}^{(j)}X + p_{11}^{(j)}Y + p_{12}^{(j)}Z + p_{13}^{(j)}W}{p_{20}^{(j)}X + p_{21}^{(j)}Y + p_{22}^{(j)}Z + p_{23}^{(j)}W},$$

where (x_j, y_j) are the measured 2D feature locations and $\{p_{00}^{(j)} \dots p_{23}^{(j)}\}$ are the known entries in camera matrix P_j

Solving the Triangulation Problem

- This set of nonlinear equations can be converted into a linear least squares problem by multiplying both sides of the denominator

Solving the Triangulation Problem

- This set of nonlinear equations can be converted into a linear least squares problem by multiplying both sides of the denominator
- Note that if we use homogeneous coordinates $\mathbf{p} = [X, Y, Z, W]^T$, the resulting set of equations is homogeneous and is best solved using SVD

Solving the Triangulation Problem

- This set of nonlinear equations can be converted into a linear least squares problem by multiplying both sides of the denominator
- Note that if we use homogeneous coordinates $\mathbf{p} = [X, Y, Z, W]^T$, the resulting set of equations is homogeneous and is best solved using SVD
- If we set $W = 1$, we can use regular linear least squares, but the resulting system may be singular or poorly conditioned, i.e. if all of the viewing rays are parallel, as occurs for points far away from the camera

Solving the Triangulation Problem

- For this reason, it is generally preferable to parameterize 3D points using homogeneous coordinates, especially if we know that there are likely to be points at greatly varying distances from the cameras

Solving the Triangulation Problem

- For this reason, it is generally preferable to parameterize 3D points using homogeneous coordinates, especially if we know that there are likely to be points at greatly varying distances from the cameras
- For the case of two observations, it turns out that the location of the point $\mathbf{p}$ that exactly minimizes the true reprojection error can be computed using the solution of degree six equations

Solving the Triangulation Problem

- Another problem to watch out for with triangulation is the issue of *chirality*, i.e. ensuring that the reconstructed points lie in front of all of the cameras

Solving the Triangulation Problem

- Another problem to watch out for with triangulation is the issue of *chirality*, i.e. ensuring that the reconstructed points lie in front of all of the cameras
- While this cannot always guaranteed, a useful heuristic is to take the points that lie behind the cameras because their rays are diverging and to place them on the plane at infinity by setting their W values to 0

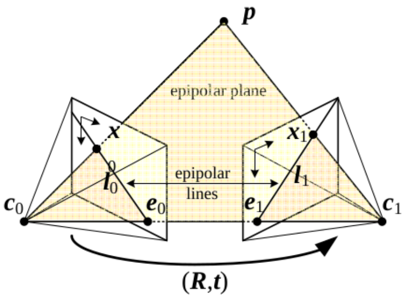
Epipolar Constraint Derivation

- Consider a 3D point **p** being viewed from two cameras whose relative position can be encoded by a rotation R and a translation **t**

Epipolar Constraint Derivation

- Consider a 3D point $\mathbf{p}$ being viewed from two cameras whose relative position can be encoded by a rotation R and a translation $\mathbf{t}$
- Since we do not know anything about the camera positions, without loss of generality, we can set the first camera at the origin $\mathbf{c}_0 = \mathbf{0}$ and at a canonical orientation $R_0 = I$

Epipolar Constraint Derivation



- The vectors $\mathbf{t} = \mathbf{c}_1 - \mathbf{c}_0$, $\mathbf{p} - \mathbf{c}_0$, and $\mathbf{p} - \mathbf{c}_1$ are co-planar and define the basic epipolar constraint expressed in terms of the pixel measurements $\mathbf{x}_0$ and $\mathbf{x}_1$

Epipolar Constraint Derivation

- Now notice that the observed location of $\mathbf{p}$ in the first image, $\mathbf{p}_0 = d_0\hat{\mathbf{x}}_0$ is mapped into the second image by the transformation

$$d_1\hat{\mathbf{x}}_1 = \mathbf{p}_1 = R\mathbf{p}_0 + \mathbf{t} = R(d_0\hat{\mathbf{x}}_0) + \mathbf{t}$$

where $\hat{\mathbf{x}}_j = K_j^{-1}\mathbf{x}_j$ are the (local) ray direction vectors

Epipolar Constraint Derivation

- Taking the cross product of both sides with $\mathbf{t}$ in order to annihilate it on the right hand side yields

$$d_1[\mathbf{t}]_\times \hat{\mathbf{x}}_1 = d_0[\mathbf{t}]_\times R\hat{\mathbf{x}}_0$$

Epipolar Constraint Derivation

- Taking the cross product of both sides with $\mathbf{t}$ in order to annihilate it on the right hand side yields

$$d_1[\mathbf{t}]_\times \hat{\mathbf{x}}_1 = d_0[\mathbf{t}]_\times R\hat{\mathbf{x}}_0$$

- Taking the dot product of both sides with $\hat{\mathbf{x}}_1$ yields

$$d_0\hat{\mathbf{x}}_1^T([\mathbf{t}]_\times R)\hat{\mathbf{x}}_0 = d_1\hat{\mathbf{x}}_1^T[\mathbf{t}]_\times \hat{\mathbf{x}}_1 = 0$$

since the right hand side is a triple product with two identical entries

Epipolar Constraint Derivation

- Taking the cross product of both sides with $\mathbf{t}$ in order to annihilate it on the right hand side yields

$$d_1[\mathbf{t}]_\times \hat{\mathbf{x}}_1 = d_0[\mathbf{t}]_\times R\hat{\mathbf{x}}_0$$

- Taking the dot product of both sides with $\hat{\mathbf{x}}_1$ yields

$$d_0\hat{\mathbf{x}}_1^T([\mathbf{t}]_\times R)\hat{\mathbf{x}}_0 = d_1\hat{\mathbf{x}}_1^T[\mathbf{t}]_\times \hat{\mathbf{x}}_1 = 0$$

  since the right hand side is a triple product with two identical entries

- Note that the cross product matrix $[\mathbf{t}]_\times$ is skew symmetric and returns 0 when pre- and post-multiplied by the same vector

Epipolar Constraint Derivation

- We therefore arrive at the basic **epipolar constraint**

$$\hat{\mathbf{x}}_1^T E \hat{\mathbf{x}}_0 = 0 \tag{1}$$

where

$$E = [\mathbf{t}]_\times R$$

is called the **essential matrix**

Epipolar Constraint Derivation

- The essential matrix is the specialization of the fundamental matrix to the case of normalized image coordinates

Epipolar Constraint Derivation

- The essential matrix is the specialization of the fundamental matrix to the case of normalized image coordinates
- The fundamental matrix may thought of as the generalization of the essential matrix in which the (inessential) assumption of calibrated cameras is removed

Epipolar Constraint Derivation

- The essential matrix is the specialization of the fundamental matrix to the case of normalized image coordinates
- The fundamental matrix may thought of as the generalization of the essential matrix in which the (inessential) assumption of calibrated cameras is removed
- Compared to the fundamental matrix, the essential matrix has fewer degrees of freedom and additional properties

Epipolar Constraint Derivation

- Notice that the essential matrix E maps a point $\hat{\mathbf{x}}_0$ in image 0 into a line $\mathbf{l}_1 = E\hat{\mathbf{x}}_0$ in image 1 since $\hat{\mathbf{x}}_1^T \mathbf{l}_1 = 0$

Epipolar Constraint Derivation

- Notice that the essential matrix E maps a point $\hat{\mathbf{x}}_0$ in image 0 into a line $\mathbf{l}_1 = E\hat{\mathbf{x}}_0$ in image 1 since $\hat{\mathbf{x}}_1^T \mathbf{l}_1 = 0$
- All such lines must pass through the second epipole $\mathbf{e}_1$ which is defined as the left singular vector of E with a 0 singular value (or equivalently the projection of $\mathbf{t}$ into image 1)

Epipolar Constraint Derivation

- Notice that the essential matrix E maps a point $\hat{\mathbf{x}}_0$ in image 0 into a line $\mathbf{l}_1 = E\hat{\mathbf{x}}_0$ in image 1 since $\hat{\mathbf{x}}_1^T \mathbf{l}_1 = 0$
- All such lines must pass through the second epipole $\mathbf{e}_1$ which is defined as the left singular vector of E with a 0 singular value (or equivalently the projection of $\mathbf{t}$ into image 1)
- The dual (transpose) of these relationships gives us the epipolar line in the first image as $\mathbf{l}_0 = E^T\hat{\mathbf{x}}_1$ and $\mathbf{e}_0$ as the zero value right singular vector of E

Recovering Camera Motion

- Given this fundamental relationship (Equation (1)), how can we use it to **recover the camera motion** encoded in the essential matrix E?

Recovering Camera Motion

- Given this fundamental relationship (Equation (1)), how can we use it to **recover the camera motion** encoded in the essential matrix E?
- If we have N corresponding measurements $\{(\mathbf{x}_{i0}, \mathbf{x}_{i1})\}$, we can form N homogeneous equations in the nine elements of $E = \{e_{00} \ldots e_{22}\}$

$$\begin{aligned} x_{i0}x_{i1}e_{00} + y_{i0}x_{i1}e_{01} + x_{i1}e_{02} + \\ x_{i0}y_{i1}e_{00} + y_{i0}y_{i1}e_{11} + y_{i1}e_{12} + \\ x_{i0}e_{20} + y_{i0}e_{21} + e_{22} = 0 \end{aligned} \tag{2}$$

where $\mathbf{x}_{ij} = (x_{ij}, y_{ij}, 1)$

Recovering Camera Motion

- This can written more compactly as

$$[\mathbf{x}_{i1} x_{i0}^T] \otimes E = Z_i \otimes E = \mathbf{z}_i \cdot \mathbf{f} = 0 \tag{3}$$

where $\otimes$ indicates an element-wise multiplication and summation of matrix elements, and $\mathbf{z}_i$ and $\mathbf{f}$ are the rasterized (vector) forms of the $Z_i = \hat{\mathbf{x}}_{i1}\hat{\mathbf{x}}_{i0}^T$ and E matrices

Recovering Camera Motion

- This can written more compactly as

$$[\mathbf{x}_{i1} x_{i0}^T] \otimes E = Z_i \otimes E = \mathbf{z}_i \cdot \mathbf{f} = 0 \tag{3}$$

  where $\otimes$ indicates an element-wise multiplication and summation of matrix elements, and $\mathbf{z}_i$ and $\mathbf{f}$ are the rasterized (vector) forms of the $Z_i = \hat{\mathbf{x}}_{i1} \hat{\mathbf{x}}_{i0}^T$ and E matrices
- Given $N \geq 8$ such equations, we can compute an estimate (up to scale) for the entries in E using an SVD

Recovering Camera Motion

- In the presence of noisy measurements, how close is this estimate to being statistically optimal?

Recovering Camera Motion

- In the presence of noisy measurements, how close is this estimate to being statistically optimal?
- If we look at the entries in Equation (2), we see that some are the products of image measurements such as $x_{i0}y_{i1}$ and others are direct image measurements (or even the identity)

Recovering Camera Motion

- In the presence of noisy measurements, how close is this estimate to being statistically optimal?
- If we look at the entries in Equation (2), we see that some are the products of image measurements such as $x_{i0}y_{i1}$ and others are direct image measurements (or even the identity)
- If the measurements have comparable noise, the terms that are products of measurements have their noise amplified by the other element in the product

Recovering Camera Motion

- In the presence of noisy measurements, how close is this estimate to being statistically optimal?
- If we look at the entries in Equation (2), we see that some are the products of image measurements such as $x_{i0}y_{i1}$ and others are direct image measurements (or even the identity)
- If the measurements have comparable noise, the terms that are products of measurements have their noise amplified by the other element in the product
- This can lead to very poor scaling, e.g. a large influence of points with large coordinates (far away from the image center)

Recovering Camera Motion

- To counteract this, point coordinates can be translated and scaled so that their centroid lies at the origin and their variance is unity, i.e.

$$\tilde{x}_i = s(x_i - \mu_x)$$
$$\tilde{y}_i = s(x_i - \mu_y)$$

such that $\sum_i \tilde{x}_i = \sum_i \tilde{y}_i = 0$ and $\sum_i \tilde{x}_i^2 + \sum_i \tilde{y}_i^2 = 2n$, where n is the number of points

Recovering Camera Motion

- To counteract this, point coordinates can be translated and scaled so that their centroid lies at the origin and their variance is unity, i.e.

$$\tilde{x}_i = s(x_i - \mu_x)$$
$$\tilde{y}_i = s(x_i - \mu_y)$$

such that $\sum_i \tilde{x}_i = \sum_i \tilde{y}_i = 0$ and $\sum_i \tilde{x}_i^2 + \sum_i \tilde{y}_i^2 = 2n$, where n is the number of points

- Once $\tilde{E}$ has been computed from the transformed coordinates $\{(\tilde{\mathbf{x}}_{i0}, \tilde{\mathbf{x}}_{ij}\}$, where $\tilde{\mathbf{x}}_{ij} = T_j \hat{\mathbf{x}}_{ij}$, the original E can be recovered as

$$E = T_1 \tilde{E} T_0$$

Recovering Camera Motion

- Once an estimate for the essential matrix E has been recovered, the direction of the translation vector $\mathbf{t}$ can be estimated

Recovering Camera Motion

- Once an estimate for the essential matrix E has been recovered, the direction of the translation vector $\mathbf{t}$ can be estimated
- Note that the absolute distance between the two cameras can never be recovered from pure image measurements alone, regardless of how many cameras or points are used

Recovering Camera Motion

- Once an estimate for the essential matrix E has been recovered, the direction of the translation vector $\mathbf{t}$ can be estimated
- Note that the absolute distance between the two cameras can never be recovered from pure image measurements alone, regardless of how many cameras or points are used
- Knowledge about absolute camera and point positions or distances, often called *ground control points* in photogrammetry, is always required to establish the final scale, position, and orientation

Recovering Camera Motion

- To estimate this direction, $\hat{\mathbf{t}}$, observe that under ideal noise-free conditions the essential matrix E is singular, i.e. $\hat{\mathbf{t}}E = 0$

Recovering Camera Motion

- To estimate this direction, $\hat{\mathbf{t}}$, observe that under ideal noise-free conditions the essential matrix E is singular, i.e. $\hat{\mathbf{t}}E = 0$
- This singularity shows up as a singular value of 0 when an SVD of E is performed

$$E = [\hat{\mathbf{t}}]_\times R = U\Sigma V^T = \begin{bmatrix} u_0 & u_1 & \hat{\mathbf{t}} \end{bmatrix} \begin{bmatrix} 1 & & \\ & 1 & \\ & & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}_0^T \\ \mathbf{v}_1^T \\ \mathbf{v}_2^T \end{bmatrix} \quad (4)$$

Recovering Camera Motion

- To estimate this direction, $\hat{\mathbf{t}}$, observe that under ideal noise-free conditions the essential matrix E is singular, i.e. $\hat{\mathbf{t}}E = 0$
- This singularity shows up as a singular value of 0 when an SVD of E is performed

$$E = [\hat{\mathbf{t}}]_\times R = U\Sigma V^T = \begin{bmatrix} u_0 & u_1 & \hat{\mathbf{t}} \end{bmatrix} \begin{bmatrix} 1 & & \\ & 1 & \\ & & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}_0^T \\ \mathbf{v}_1^T \\ \mathbf{v}_2^T \end{bmatrix} \quad (4)$$

- When E is computed from noisy measurements, the singular vector associated with the smallest singular value gives us $\hat{\mathbf{t}}$

Recovering Camera Motion

- Since E is rank deficient, it turns out that we actually only need 7 correspondences of the form of Equation (3) instead of 8 to estimate this matrix

Recovering Camera Motion

- Since E is rank deficient, it turns out that we actually only need 7 correspondences of the form of Equation (3) instead of 8 to estimate this matrix
- From this set of 7 homogeneous equations (which we can stack into a 7×9 matrix for SVD analysis) we can find two independent vectors say, $\mathbf{f}_0$ and $\mathbf{f}_1$, such that $\mathbf{z}_i \cdot \mathbf{f}_j = 0$

Recovering Camera Motion

- These two vectors can be converted back into 3×3 matrices E_0 and E_1, which span the solution space for

$$E = \alpha E_0 + (1 - \alpha) E_1$$

Recovering Camera Motion

- These two vectors can be converted back into 3×3 matrices E_0 and E_1, which span the solution space for

$$E = \alpha E_0 + (1 - \alpha) E_1$$

- To find the correct value of α, we observe that E has a zero determinant, since it is rank deficient, and thus

$$\det|\alpha E_0 + (1 - \alpha) E_1| = 0$$

Recovering Camera Motion

- This gives us a cubic equation in α, which has either one or three solutions (roots)

Recovering Camera Motion

- This gives us a cubic equation in α, which has either one or three solutions (roots)
- Substituting these values into Equation (4) to obtain E, we can test this essential matrix against other unused feature correspondences to select the correct one

Recovering Camera Motion

- Once $\hat{\mathbf{t}}$ has been recovered, how can we estimate the corresponding rotation matrix R?

Recovering Camera Motion

- Once $\hat{\mathbf{t}}$ has been recovered, how can we estimate the corresponding rotation matrix R?
- Recall that the cross product operator $[\hat{\mathbf{t}}]$ zeros out the $\hat{\mathbf{t}}$ component, and rotates the other two by 90°,

$$[\hat{\mathbf{t}}]_\times = SZR_{90^\circ}S^T = \begin{bmatrix}\mathbf{s}_0 & \mathbf{s}_1 & \hat{\mathbf{t}}\end{bmatrix} \begin{bmatrix}1 & & \\ & 1 & \\ & & 0\end{bmatrix} \begin{bmatrix}0 & -1 & \\ 1 & 0 & \\ & & 1\end{bmatrix} \begin{bmatrix}\mathbf{s}_0^T \\ \mathbf{s}_1^T \\ \hat{\mathbf{t}}^T\end{bmatrix} \quad (5)$$

where $\hat{\mathbf{t}} = \mathbf{s}_0 \times \mathbf{s}_1$

Recovering Camera Motion

- From Equations (4) and (5) we get

$$E = [\hat{\mathbf{t}}]_\times R = SZR_{90^\circ}S^T R = U\Sigma V^T$$

from which we can conclude that $S = U$

Recovering Camera Motion

- From Equations (4) and (5) we get

$$E = [\hat{\mathbf{t}}]_\times R = SZR_{90^\circ}S^T R = U\Sigma V^T$$

from which we can conclude that $S = U$

- Recall that for a noise-free essential matrix, $\Sigma = Z$, and hence

$$R_{90^\circ}U^T R = V^T$$

and

$$R = UR_{90^\circ}^T V^T$$

Recovering Camera Motion

- Unfortunately, we only know both E and $\hat{\mathbf{t}}$ up to a sign

Recovering Camera Motion

- Unfortunately, we only know both E and $\hat{\mathbf{t}}$ up to a sign
- Furthermore, the matrices U and V are not guaranteed to be rotations (you can flip both their signs and still get a valid SVD)

Recovering Camera Motion

- For this reason, we have to generate all four possible rotation matrices
$$R = \pm U R_{\pm 90^\circ}^T V^T$$
and keep the two whose determinant $|R| = 1$

Recovering Camera Motion

- For this reason, we have to generate all four possible rotation matrices
$$R = \pm U R_{\pm 90^\circ}^T V^T$$
and keep the two whose determinant $|R| = 1$

- To disambiguate between the remaining pair of potential rotations, which form a *twisted pair*, we need to pair them with both possible signs of the translation direction $\pm\hat{\mathbf{t}}$ and select the combination in which the largest number of points is seen in front of the both cameras

Recovering Camera Motion

- Points must lie in front of the camera, i.e. at a positive distance along the viewing rays emanating from the camera (chirality)

Recovering Camera Motion

- Points must lie in front of the camera, i.e. at a positive distance along the viewing rays emanating from the camera (chirality)
- The chirality (sign of the distances) of the points in a reconstruction can be used inside a RANSAC procedure (along with the reprojection errors) to distinguish between likely and unlikely configurations

Pure Translation (Known Rotation)

- In the case where we know the rotation, we can pre-rotate the points in the second image to match the viewing direction of the first

Pure Translation (Known Rotation)

- In the case where we know the rotation, we can pre-rotate the points in the second image to match the viewing direction of the first
- The resulting set of 3D points all move towards (or away from) the **focus of expansion** (FOE)

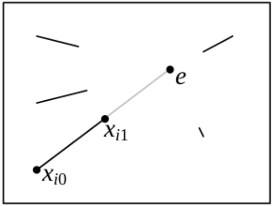
Pure Translation (Known Rotation)

- In the case where we know the rotation, we can pre-rotate the points in the second image to match the viewing direction of the first
- The resulting set of 3D points all move towards (or away from) the **focus of expansion** (FOE)
- The resulting essential matrix E is skew symmetric and so can be estimated more directly by setting $e_{ij} = -e_{ji}$ and $e_{ii} = 0$

Pure Translation (Known Rotation)

- In the case where we know the rotation, we can pre-rotate the points in the second image to match the viewing direction of the first
- The resulting set of 3D points all move towards (or away from) the **focus of expansion** (FOE)
- The resulting essential matrix E is skew symmetric and so can be estimated more directly by setting $e_{ij} = -e_{ji}$ and $e_{ii} = 0$
- Two points with non-zero parallax now suffice to estimate the FOE

Pure Translation (Known Rotation)



- **Pure translational** camera motion results in visual motion where all the points move towards (or away from) a common FOE $\mathbf{e}$ and therefore satisfy the triple product condition $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{e}) = \mathbf{e} \cdot (\mathbf{x}_0 \times \mathbf{x}_1) = 0$

Pure Translation (Known Rotation)

- A more direct derivation of the FOE estimate can be obtained by minimizing the triple product

$$\sum_i (\mathbf{x}_{i0}, \mathbf{x}_{i1}, \mathbf{e})^2 = \sum_i ((\mathbf{x}_{i0} \times \mathbf{x}_{i1}) \cdot \mathbf{e})^2$$

which is equivalent to finding the null space for the set of equations

$$(y_{i0} - y_{i1})e_0 + (x_{i1} - x_{i0})e_1 + (x_{i0}y_{i1} - y_{i0}x_{i1})e_2 = 0 \qquad (6)$$

Pure Translation (Known Rotation)

- In situations where a large number of points at infinity are available (e.g. outdoor scenes or camera motion is small compared to distant objects) this suggests an alternative RANSAC strategy for estimating the camera motion

Pure Translation (Known Rotation)

- In situations where a large number of points at infinity are available (e.g. outdoor scenes or camera motion is small compared to distant objects) this suggests an alternative RANSAC strategy for estimating the camera motion
- First, pick a pair of points to estimate a rotation, then compute the FOE and check whether the residual error is small and whether the motions towards or away from the epipole (FOE) are all in the same direction

Pure Rotation

- The case of **pure rotation** results in a degenerate estimate of the essential matrix E and of the translation direction $\hat{\mathbf{t}}$

Pure Rotation

- The case of **pure rotation** results in a degenerate estimate of the essential matrix E and of the translation direction $\hat{\mathbf{t}}$
- Consider the first case of the rotation matrix being known, the estimates for the FOE will be degenerate, since $\mathbf{x}_{i0} \approx \mathbf{x}_{i1}$, and hence Equation (6) is degenerate

Pure Rotation

- A similar argument shows that the equations for the essential matrix (Equation (2)) are also rank deficient

Pure Rotation

- A similar argument shows that the equations for the essential matrix (Equation (2)) are also rank deficient
- This suggests that it may be prudent before computing a full essential matrix to first compute a rotation estimate R, potentially with just a small number of points, and then compute the residuals after rotating the points before proceeding with a full E computation

Projective (Uncalibrated) Reconstruction

- In many cases we do not know ahead of time the intrinsic calibration parameters associated with the input images

Projective (Uncalibrated) Reconstruction

- In many cases we do not know ahead of time the intrinsic calibration parameters associated with the input images
- In such situations, we can still estimate a two-frame reconstruction although the true metric structure may not be available (e.g. orthogonal lines or planes in the world may not end up being reconstructed as orthogonal)

Projective (Uncalibrated) Reconstruction

- Consider the derivations used to estimate E, in the uncalibrated case we do not know the calibration matrices K_j, so we cannot use the normalized ray directions $\hat{\mathbf{x}}_j = K_j^{-1}\mathbf{x}_j$

Projective (Uncalibrated) Reconstruction

- Consider the derivations used to estimate E, in the uncalibrated case we do not know the calibration matrices K_j, so we cannot use the normalized ray directions $\hat{\mathbf{x}}_j = K_j^{-1}\mathbf{x}_j$
- Instead, we have access only to the image coordinates $\mathbf{x}_j$, and so the essential matrix becomes

$$\hat{\mathbf{x}}_1^T E \hat{\mathbf{x}}_1 = \mathbf{x}_1^T K_1^{-T} E K_0^{-1} \mathbf{x}_0 = \mathbf{x}_1^T F \mathbf{x}_0 = 0$$

where

$$F = K_1^{-T} E K_0^{-1} = [\mathbf{e}]_\times \tilde{H} \tag{7}$$

is the fundamental matrix

Projective (Uncalibrated) Reconstruction

- Similar to the essential matrix, F is (in principle) rank two

$$F = [\mathbf{e}]_\times \tilde{H} = U\Sigma V^T = \begin{bmatrix} \mathbf{u}_0 & \mathbf{u}_1 & \mathbf{e}_1 \end{bmatrix} \begin{bmatrix} \sigma_0 & & \\ & \sigma_1 & \\ & & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}_0^T \\ \mathbf{v}_1^T \\ \mathbf{e}_0^T \end{bmatrix}$$

Projective (Uncalibrated) Reconstruction

- Similar to the essential matrix, F is (in principle) rank two

$$F = [\mathbf{e}]_\times \tilde{H} = U\Sigma V^T = \begin{bmatrix} \mathbf{u}_0 & \mathbf{u}_1 & \mathbf{e}_1 \end{bmatrix} \begin{bmatrix} \sigma_0 & & \\ & \sigma_1 & \\ & & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}_0^T \\ \mathbf{v}_1^T \\ \mathbf{e}_0^T \end{bmatrix}$$

- Its smallest left singular vector indicates the epipole $\mathbf{e}_1$ in the image 1 and its right singular vector is $\mathbf{e}_0$

Projective (Uncalibrated) Reconstruction

- The homography $\tilde{H}$ which in principle should equal

$$\tilde{H} = K_1^{-T} R K_0^{-1}$$

cannot be uniquely recovered from F since any homography of the form $\tilde{H}' = \tilde{H} + \mathbf{e}\mathbf{v}^T$ results in the same F matrix

Projective (Uncalibrated) Reconstruction

- The homography $\tilde{H}$ which in principle should equal

$$\tilde{H} = K_1^{-T} R K_0^{-1}$$

  cannot be uniquely recovered from F since any homography of the form $\tilde{H}' = \tilde{H} + \mathbf{e}\mathbf{v}^T$ results in the same F matrix

- Any one of these valid $\tilde{H}$ maps some plane in the scene from one image to the other

Projective (Uncalibrated) Reconstruction

- The homography $\tilde{H}$ which in principle should equal

$$\tilde{H} = K_1^{-T} R K_0^{-1}$$

  cannot be uniquely recovered from F since any homography of the form $\tilde{H}' = \tilde{H} + \mathbf{e}\mathbf{v}^T$ results in the same F matrix

- Any one of these valid $\tilde{H}$ maps some plane in the scene from one image to the other

- It is not possible to tell in advance which one it is without either selecting four or more co-planar correspondences to compute $\tilde{H}$ as part of the F estimation process or mapping all points in one image through $\tilde{H}$ and seeing which ones line up with their corresponding locations in the other

Projective (Uncalibrated) Reconstruction

- In order to create a **projective reconstruction** of the scene, we can pick any valid homography $\tilde{H}$ that satisfies Equation (7)

Projective (Uncalibrated) Reconstruction

- In order to create a **projective reconstruction** of the scene, we can pick any valid homography $\tilde{H}$ that satisfies Equation (7)
- Following an analogous technique we get

$$F = [\mathbf{e}]_\times \tilde{H} = SZR_{90^\circ} S^T \tilde{H} = U\Sigma V^T$$

and hence

$$\tilde{H} = URK_{90^\circ}^T \hat{\Sigma} V^T$$

where $\hat{\Sigma}$ is the singular value matrix with the smallest value replaced by a reasonable alternative (e.g. the middle value)

Projective (Uncalibrated) Reconstruction

- We can then form a pair of camera matrices

$$P_0 = [I \,|\, \mathbf{0}] \quad \text{and} \quad P_1 = [\tilde{H} \,|\, \mathbf{e}]$$

from which a projective reconstruction of the scene can be computed using triangulation

Projective (Uncalibrated) Reconstruction

- While the projective transformation may not be useful in practice, it can often be *upgraded* to an affine or metric reconstruction

Projective (Uncalibrated) Reconstruction

- While the projective transformation may not be useful in practice, it can often be *upgraded* to an affine or metric reconstruction
- Even without this step, the fundamental matrix F can be very useful in finding additional correspondences as they must all lie on corresponding epipolar lines

Projective (Uncalibrated) Reconstruction

- While the projective transformation may not be useful in practice, it can often be *upgraded* to an affine or metric reconstruction
- Even without this step, the fundamental matrix F can be very useful in finding additional correspondences as they must all lie on corresponding epipolar lines
- For example, any feature $\mathbf{x}_0$ in image 0 must have its correspondence lying on the associated epipolar line $\mathbf{l}_1 = F\mathbf{x}_0$ in image 1, assuming that the point motions are due to rigid transformations

Self-Calibration

- The results of the SfM computation are much more useful (and intelligible) if a *metric* is obtained

Self-Calibration

- The results of the SfM computation are much more useful (and intelligible) if a *metric* is obtained
- This metric could be one in which parallel lines are parallel, orthogonal walls are at right angles, and the reconstructed model is a scaled version of reality

Self-Calibration

- Over the years, a large number of **self-calibration** (or **auto-calibration**) techniques have been developed for converting a projective reconstruction into a metric one

Self-Calibration

- Over the years, a large number of **self-calibration** (or **auto-calibration**) techniques have been developed for converting a projective reconstruction into a metric one
- This is equivalent to recovering the unknown calibration matrices K_j associated with each image

Self-Calibration

- In situations where certain additional information is known about the scene, different methods may be employed

Self-Calibration

- In situations where certain additional information is known about the scene, different methods may be employed
- For example, if there are parallel lines in the scene (several lines converge on the same vanishing point) three or more vanishing points can be used to establish the homography for the plane at infinity from which focal lengths and rotations can be recovered

Self-Calibration

- In situations where certain additional information is known about the scene, different methods may be employed
- For example, if there are parallel lines in the scene (several lines converge on the same vanishing point) three or more vanishing points can be used to establish the homography for the plane at infinity from which focal lengths and rotations can be recovered
- If two or more finite *orthogonal* vanishing points have been observed, a single-image calibration method based on vanishing points can be used instead

Self-Calibration

- In the absence of such external information, it is not possible to recover a fully parameterized independent calibration matrix K_j for each image from correspondences alone

Self-Calibration

- In the absence of such external information, it is not possible to recover a fully parameterized independent calibration matrix K_j for each image from correspondences alone
- To see this, consider the set of all camera matrices $P_j = K_j[R_j \mid \mathbf{t}_j]$ projecting world coordinates $\mathbf{p}_i = [X_i, Y_i, Z_i, W_i]^T$ into screen coordinates $\mathbf{x}_{ij} \sim P_j \mathbf{p}_i$

Self-Calibration

- Now consider transforming the 3D scene $\{\mathbf{p}_i\}$ through an arbitrary 4×4 projective transformation $\tilde{H}$, yielding a new model consisting of points $\mathbf{p}'_i = \tilde{H}\mathbf{p}_i$

Self-Calibration

- Now consider transforming the 3D scene $\{\mathbf{p}_i\}$ through an arbitrary 4×4 projective transformation $\tilde{H}$, yielding a new model consisting of points $\mathbf{p'}_i = \tilde{H}\mathbf{p}_i$
- Post-multiplying each P_j by $\tilde{H}^{-1}$ still produces the same screen coordinates and a new set of calibration matrices can be computed by applying RQ decomposition to the new camera matrix $P'_j = P_j\tilde{H}^{-1}$

Self-Calibration

- For this reason, all self-calibration methods assume some restricted form of the calibration matrix, either by setting or equating some of their elements or by assuming that they do not vary over time

Self-Calibration

- For this reason, all self-calibration methods assume some restricted form of the calibration matrix, either by setting or equating some of their elements or by assuming that they do not vary over time
- We'll consider a simple technique that can recover the focal lengths (f_0, f_1) of both images from the fundamental matrix F in a two-frame reconstruction

Self-Calibration

- To accomplish this, we assume that the camera has zero skew, a known aspect ratio (usually set to 1), and a known optical center

Self-Calibration

- To accomplish this, we assume that the camera has zero skew, a known aspect ratio (usually set to 1), and a known optical center
- How reasonable is this assumption in practice?

Self-Calibration

- To accomplish this, we assume that the camera has zero skew, a known aspect ratio (usually set to 1), and a known optical center
- How reasonable is this assumption in practice?
  - The answer is "it depends"

Self-Calibration

- If absolute metric accuracy is required, it is imperative to pre-calibrate the cameras and to use ground control points to pin down the reconstruction

Self-Calibration

- If absolute metric accuracy is required, it is imperative to pre-calibrate the cameras and to use ground control points to pin down the reconstruction
- If instead we simply wish to reconstruct the world for visualization or image-based rendering applications, then this assumption is quite reasonable in practice

Self-Calibration

- Most cameras today have square pixels and an optical center near the middle of the image, and are much more likely to deviate from a simple camera model due to radial distortion

Self-Calibration

- Most cameras today have square pixels and an optical center near the middle of the image, and are much more likely to deviate from a simple camera model due to radial distortion
- The biggest problems occur when images have been cropped off-center, in which case the optical center will no longer be in the middle, or when perspective pictures have been taken of a different picture in which case a general camera matrix becomes necessary

Self-Calibration

- Given these caveats, a two-frame focal length estimation algorithm can be used as follows

Self-Calibration

- Given these caveats, a two-frame focal length estimation algorithm can be used as follows
- First, take the left and right singular vectors $\{\mathbf{u}_0, \mathbf{u}_1, \mathbf{v}_0, \mathbf{v}_1\}$ of the fundamental matrix F and their associated singular values $\{\sigma_0, \sigma_1\}$

Self-Calibration

- Next, form the following set of equations:

$$\frac{\mathbf{u}_1^T D_0 \mathbf{u}_1}{\sigma_0^2 \mathbf{v}_0^T D_1 \mathbf{v}_0} = -\frac{\mathbf{u}_0^T D_0 \mathbf{u}_0}{\sigma_0 \sigma_1 \mathbf{v}_0^T D_1 \mathbf{v}_1} = \frac{\mathbf{u}_0^T D_0 \mathbf{u}_0}{\sigma_1^2 \mathbf{v}_1^T D_1 \mathbf{v}_1}$$

where the two matrices

$$D_j = K_j K_j^T = \mathrm{diag}(f_j^2, f_j^2, 1) = \begin{bmatrix} f_j^2 & & \\ & f_j^2 & \\ & & 1 \end{bmatrix}$$

encode the unknown focal lengths

Self-Calibration

- For simplicity, we rewrite each of the numerators and denominators as

$$e_{ij0}(f_0^2) = \mathbf{u}_i^T D_0 \mathbf{u}_j = a_{ij} + b_{ij} f_0^2$$
$$e_{ij1}(f_1^2) = \sigma_i \sigma_j \mathbf{v}_i^T D_1 \mathbf{v}_j = c_{ij} + d_{ij} f_1^2$$

Self-Calibration

- For simplicity, we rewrite each of the numerators and denominators as

$$e_{ij0}(f_0^2) = \mathbf{u}_i^T D_0 \mathbf{u}_j = a_{ij} + b_{ij} f_0^2$$
$$e_{ij1}(f_1^2) = \sigma_i \sigma_j \mathbf{v}_i^T D_1 \mathbf{v}_j = c_{ij} + d_{ij} f_1^2$$

- Notice that each of these is affine (linear plus constant) in either f_0^2 or f_1^2

Self-Calibration

- For simplicity, we rewrite each of the numerators and denominators as

$$e_{ij0}(f_0^2) = \mathbf{u}_i^T D_0 \mathbf{u}_j = a_{ij} + b_{ij} f_0^2$$
$$e_{ij1}(f_1^2) = \sigma_i \sigma_j \mathbf{v}_i^T D_1 \mathbf{v}_j = c_{ij} + d_{ij} f_1^2$$

- Notice that each of these is affine (linear plus constant) in either f_0^2 or f_1^2
- Thus, we can cross-multiply these equations to obtain quadratic equations in f_j^2 which can be readily solved

Recovering Structure and Motion using Factorization

- When processing video sequences, we often get extended **feature tracks** from which it is possible to recover the structure and motion using a process called **factorization**

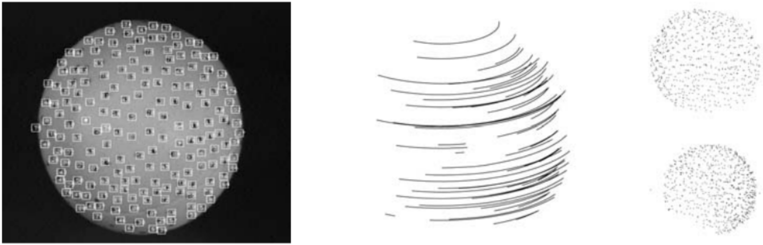
Recovering Structure and Motion using Factorization

- When processing video sequences, we often get extended **feature tracks** from which it is possible to recover the structure and motion using a process called **factorization**
- Consider the tracks generated by rotating a sphere-like object which has been marked with dots to make its shape and motion more discernible

Recovering Structure and Motion using Factorization

- When processing video sequences, we often get extended **feature tracks** from which it is possible to recover the structure and motion using a process called **factorization**
- Consider the tracks generated by rotating a sphere-like object which has been marked with dots to make its shape and motion more discernible
- We can see from the shape of tracks that the moving object must be a sphere, but how can we infer this mathematically?

Recovering Structure and Motion using Factorization



(a) (b) (c)

- 3D reconstruction of a rotating ping pong ball using factorization: (a) sample image with tracked features overlaid; (b) sub-sampled feature motion stream; (c) two views of the reconstructed 3D model

Recovering Structure and Motion using Factorization

- It turns out that shape and motion can be recovered simultaneously using SVD, e.g. consider the orthographic and weak perspective projection models

Recovering Structure and Motion using Factorization

- It turns out that shape and motion can be recovered simultaneously using SVD, e.g. consider the orthographic and weak perspective projection models
- Since the last row is always $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$, there is no perspective division and we can write

$$\mathbf{x}_{ji} = \tilde{P}_j \bar{\mathbf{p}}_i$$

where $\mathbf{x}_{ji}$ is the location of the ith point in the jth frame, $\tilde{P}_j$ is the upper 2×4 portion of the projection matrix P_j, and $\bar{\mathbf{p}}_i = [X_i, Y_i, Z_i, 1]^T$ is the augmented 3D point position

Recovering Structure and Motion using Factorization

- Let us assume that every point i is visible in every frame j

Recovering Structure and Motion using Factorization

- Let us assume that every point i is visible in every frame j
- We can take the **centroid** (average) of the projected point locations $\mathbf{x}_{ji}$ in frame j

$$\bar{\mathbf{x}}_j = \frac{1}{N}\sum_i \mathbf{x}_{ji} = \tilde{P}_j \frac{1}{N}\sum_i \bar{\mathbf{p}}_i = \tilde{P}_j \bar{\mathbf{c}}$$

where $\bar{\mathbf{c}} = [\bar{X}, \bar{Y}, \bar{Z}, 1]^T$ is the augmented 3D centroid of the point cloud

Recovering Structure and Motion using Factorization

- World coordinate frames in SfM are always arbitrary, i.e. we cannot recover the true 3D locations without ground control points (known measurements)

Recovering Structure and Motion using Factorization

- World coordinate frames in SfM are always arbitrary, i.e. we cannot recover the true 3D locations without ground control points (known measurements)
- Therefore, we can place the origin of the world at the centroid of the points, i.e. $\bar{X} = \bar{Y} = \bar{Z} = 0$, so that $\bar{\mathbf{c}} = [0, 0, 0, 1]^T$

Recovering Structure and Motion using Factorization

- World coordinate frames in SfM are always arbitrary, i.e. we cannot recover the true 3D locations without ground control points (known measurements)
- Therefore, we can place the origin of the world at the centroid of the points, i.e. $\bar{X} = \bar{Y} = \bar{Z} = 0$, so that $\bar{\mathbf{c}} = [0, 0, 0, 1]^T$
- We see from this that the centroid of the 2D points in each frame $\bar{\mathbf{x}}_j$ directly gives us the last element of $\tilde{P}_j$

Recovering Structure and Motion using Factorization

- Let $\tilde{\mathbf{x}}_{ji} = \mathbf{x}_{ji} - \bar{\mathbf{x}}_j$ be the 2D point locations after their image centroid has been subtracted

Recovering Structure and Motion using Factorization

- Let $\tilde{\mathbf{x}}_{ji} = \mathbf{x}_{ji} - \bar{\mathbf{x}}_j$ be the 2D point locations after their image centroid has been subtracted
- We can now write
$$\tilde{\mathbf{x}}_{ji} = M_j \mathbf{p}_j$$
where M_j is the upper 2×3 portion of the projection matrix P_j and $\mathbf{p}_i = [X_i, Y_i, Z_i]^T$

Recovering Structure and Motion using Factorization

- We concatenate all of these measurement equations into one large matrix

$$\hat{X} = \begin{bmatrix} \tilde{\mathbf{x}}_{11} & \cdots & \tilde{\mathbf{x}}_{1i} & \cdots & \tilde{\mathbf{x}}_{1N} \\ \vdots & & \vdots & & \vdots \\ \tilde{\mathbf{x}}_{j1} & \cdots & \tilde{\mathbf{x}}_{ji} & \cdots & \tilde{\mathbf{x}}_{jN} \\ \vdots & & \vdots & & \vdots \\ \tilde{\mathbf{x}}_{M1} & \cdots & \tilde{\mathbf{x}}_{Mi} & \cdots & \tilde{\mathbf{x}}_{MN} \end{bmatrix} = \begin{bmatrix} M_1 \\ \vdots \\ M_j \\ \vdots \\ M_M \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 & \cdots & \mathbf{p}_i & \cdots & \mathbf{p}_N \end{bmatrix} = \hat{M}\hat{S}$$

where $\hat{X}$ is the **measurement matrix**, $\hat{M}$ is the **motion matrix**, and $\hat{S}$ is the **structure matrix**

Recovering Structure and Motion using Factorization

- Since $\hat{M}$ is $2M \times 3$ and $\hat{S}$ is $3 \times N$, an SVD applied to $\hat{X}$ has only three non-zero singular values

Recovering Structure and Motion using Factorization

- Since $\hat{M}$ is $2M \times 3$ and $\hat{S}$ is $3 \times N$, an SVD applied to $\hat{X}$ has only three non-zero singular values
- In the case where the measurements in $\hat{X}$ are noisy, SVD returns the rank-three factorization of $\hat{X}$ that is closest to $\hat{X}$ in a least squares sense

Recovering Structure and Motion using Factorization

- It would be nice if the SVD of $\hat{X} = U\Sigma V^T$ directly returned the matrices $\hat{M}$ and $\hat{S}$, but it does not

Recovering Structure and Motion using Factorization

- It would be nice if the SVD of $\hat{X} = U\Sigma V^T$ directly returned the matrices $\hat{M}$ and $\hat{S}$, but it does not
- Instead, we can write the relationship

$$\hat{X} = U\Sigma V^T = [UQ][Q^{-1}\Sigma V^T]$$

and set $\hat{M} = UQ$ and $\hat{S} = Q^{-1}\Sigma V^T$

Recovering Structure and Motion using Factorization

- How can we recover the values of the 3×3 matrix Q?

Recovering Structure and Motion using Factorization

- How can we recover the values of the 3×3 matrix Q?
- This depends on the motion model being used, in the case of orthographic projection the entries in M_j are the first two rows of rotation matrices R_j, thus we have

$$\mathbf{m}_{j0} \cdot \mathbf{m}_{j0} = \mathbf{u}_{2j} Q Q^T \mathbf{u}_{2j}^T = 1$$
$$\mathbf{m}_{j0} \cdot \mathbf{m}_{j1} = \mathbf{u}_{2j} Q Q^T \mathbf{u}_{2j+1}^T = 0$$
$$\mathbf{m}_{j1} \cdot \mathbf{m}_{j1} = \mathbf{u}_{2j+1} Q Q^T \mathbf{u}_{2j+1}^T = 1$$

where $\mathbf{u}_k$ are the 3×1 rows of U

Recovering Structure and Motion using Factorization

- This gives us a large set of equations for the entries in QQ^T from which Q can be recovered using a matrix square root

Recovering Structure and Motion using Factorization

- This gives us a large set of equations for the entries in QQ^T from which Q can be recovered using a matrix square root
- If we have scaled orthography, i.e. $M_j = s_j R_j$, the first and third equations are equal to s_j and can be set equal to each other

Recovering Structure and Motion using Factorization

- Note that even once Q has been recovered, we can never be sure if the object is rotating left or right or if its depth reversed version is moving the other way

Recovering Structure and Motion using Factorization

- Note that even once Q has been recovered, we can never be sure if the object is rotating left or right or if its depth reversed version is moving the other way
- Additional cues such as the appearance and disappearance of points, or perspective effects, can be used to remove this ambiguity

Recovering Structure and Motion using Factorization

- A disadvantage of factorization approaches is that they require a complete set of tracks, i.e. each point must be visible in each frame, in order for the factorization approach to work

Recovering Structure and Motion using Factorization

- A disadvantage of factorization approaches is that they require a complete set of tracks, i.e. each point must be visible in each frame, in order for the factorization approach to work
- This problem can be dealt with by first applying factorization to smaller denser subsets and then using known camera (motion) or point (structure) estimates to *hallucinate* additional missing values

Perspective and Projective Factorization

- Another disadvantage of regular factorization is that it cannot deal with perspective cameras

Perspective and Projective Factorization

- Another disadvantage of regular factorization is that it cannot deal with perspective cameras
- One way to get around this problem is to perform an initial affine (e.g. orthographic) reconstruction and to then correct for the perspective effects in a iterative manner

Perspective and Projective Factorization

- Observe that the object-centered projection model

$$x_{ji} = s_j \frac{\mathbf{r}_{xj} \cdot \mathbf{p}_i + t_{xj}}{1 + \eta_j \mathbf{r}_{zj} \cdot \mathbf{p}_i}$$
$$y_{ji} = s_j \frac{\mathbf{r}_{yj} \cdot \mathbf{p}_i + t_{yj}}{1 + \eta_j \mathbf{r}_{zj} \cdot \mathbf{p}_i}$$

differs from the scaled orthographic projection model by the inclusion of the denominator terms $(1 + \eta_j \mathbf{r}_z j \cdot \mathbf{p}_i)$

Perspective and Projective Factorization

- Observe that the object-centered projection model

$$x_{ji} = s_j \frac{\mathbf{r}_{xj} \cdot \mathbf{p}_i + t_{xj}}{1 + \eta_j \mathbf{r}_{zj} \cdot \mathbf{p}_i}$$
$$y_{ji} = s_j \frac{\mathbf{r}_{yj} \cdot \mathbf{p}_i + t_{yj}}{1 + \eta_j \mathbf{r}_{zj} \cdot \mathbf{p}_i}$$

  differs from the scaled orthographic projection model by the inclusion of the denominator terms $(1 + \eta_j \mathbf{r}_z j \cdot \mathbf{p}_i)$

- If we knew the correct values of $\eta_j = t_{zj}^{-1}$ along with R_j and $\mathbf{p}_i$, we could cross-multiply the lhs by the denominator and get corrected values for which the bilinear model is exact

Perspective and Projective Factorization

- In practice, after an initial reconstruction, the values of η_j can be estimated independently for each frame by comparing reconstructed and sensed point positions

Perspective and Projective Factorization

- In practice, after an initial reconstruction, the values of η_j can be estimated independently for each frame by comparing reconstructed and sensed point positions
- Note that since the η_j are determined from the image measurements, the cameras do not have to be pre-calibrated, i.e. their focal lengths can be recovered from $f_j = s_j/\eta_j$

Perspective and Projective Factorization

- Once the η_j have been estimated, the feature locations can then be corrected before applying another round of factorization

Perspective and Projective Factorization

- Once the η_j have been estimated, the feature locations can then be corrected before applying another round of factorization
- Due to the initial depth reversal ambiguity, both reconstructions have to be tried while calculating η_j (the incorrect reconstruction will result in negative η_j which is not physically meaningful)

Application: Sparse 3D Model Extraction

- Once a multi-view 3D reconstruction of the scene has been estimated, it then becomes possible to create a texture-mapped 3D model of the object and look at it from new directions

Application: Sparse 3D Model Extraction

- Once a multi-view 3D reconstruction of the scene has been estimated, it then becomes possible to create a texture-mapped 3D model of the object and look at it from new directions
- The first step is to create a denser 3D model than the sparse point cloud that SfM produces using a technique such as 3D triangulation

Application: Sparse 3D Model Extraction

- Once a multi-view 3D reconstruction of the scene has been estimated, it then becomes possible to create a texture-mapped 3D model of the object and look at it from new directions
- The first step is to create a denser 3D model than the sparse point cloud that SfM produces using a technique such as 3D triangulation
- Then, the triangulated points can then be used to produce a surface mesh

Application: Sparse 3D Model Extraction

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

- 3D teacup model reconstructed from a 240-frame video sequence: (a) first frame of video; (b) last frame of video; (c) side view of 3D model; (d) top view of 3D model

Application: Sparse 3D Model Extraction

- In order to create a more realistic model, a **texture map** can be extracted for each triangle face

Application: Sparse 3D Model Extraction

- In order to create a more realistic model, a **texture map** can be extracted for each triangle face
- The equations to map points on the surface of a 3D triangle to a 2D image are straightforward: just pass the local 2D coordinates on the triangle through the 3×4 camera projection matrix to obtain a 3×3 homography

Application: Sparse 3D Model Extraction

- In order to create a more realistic model, a **texture map** can be extracted for each triangle face
- The equations to map points on the surface of a 3D triangle to a 2D image are straightforward: just pass the local 2D coordinates on the triangle through the 3×4 camera projection matrix to obtain a 3×3 homography
- When multiple source images are available, either the closest and most fronto-parallel image can be used or multiple images can be blended in to deal with view-dependent foreshortening

Robust Non-Linear Minimization

- The most accurate way to recover structure and motion is to perform robust non-linear minimization of the measurement (re-projection) errors

Robust Non-Linear Minimization

- The most accurate way to recover structure and motion is to perform robust non-linear minimization of the measurement (re-projection) errors
- This technique, commonly used in the photogrammetry (and now computer vision) communities, is known as **bundle adjustment**

Robust Non-Linear Minimization

- Using bundle adjustment, our feature location measurements $\mathbf{x}_{ij}$ now depend not only on the point (track index) i but also on the camera pose index j

$$\mathbf{x}_{ij} = \mathbf{f}(\mathbf{p}_i, R_j, \mathbf{c}_j, K_j)$$

  and that the 3D point positions $\mathbf{p}_i$ are also being simultaneously updated

Robust Non-Linear Minimization

- Using bundle adjustment, our feature location measurements $\mathbf{x}_{ij}$ now depend not only on the point (track index) i but also on the camera pose index j

$$\mathbf{x}_{ij} = \mathbf{f}(\mathbf{p}_i, R_j, \mathbf{c}_j, K_j)$$

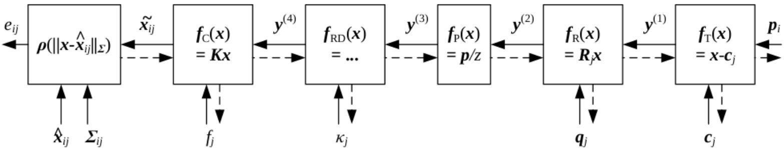
  and that the 3D point positions $\mathbf{p}_i$ are also being simultaneously updated

- In addition, it is common to add a stage for radial distortion parameter estimation

$$\mathbf{f}_{RD}(\mathbf{x}) = (1 + \kappa_1 r^2 + \kappa_2 r^4)\mathbf{x}$$

  if the cameras have not been pre-calibrated

Robust Non-Linear Minimization



- A set of chained transforms for projecting a 3D point $\mathbf{p}_i$ into a 2D measurement $\mathbf{x}_{ij}$ through a series of transformations $f^{(k)}$, each of which is controlled by its own set of parameters, the dashed lines indicate the flow of information as partial derivatives are computed during a backward pass

Robust Non-Linear Minimization

- The leftmost box (transform) in the previous figure performs a robust comparison of the predicted and measured 2D locations $\hat{\mathbf{x}}_{ij}$ and $\tilde{\mathbf{x}}_{ij}$ after re-scaling by the measurement noise covariance Σ_{ij}

Robust Non-Linear Minimization

- The leftmost box (transform) in the previous figure performs a robust comparison of the predicted and measured 2D locations $\hat{\mathbf{x}}_{ij}$ and $\tilde{\mathbf{x}}_{ij}$ after re-scaling by the measurement noise covariance Σ_{ij}
- In more detail, this operation can be written as

$$\mathbf{r}_{ij} = \tilde{\mathbf{x}}_{ij} - \hat{\mathbf{x}}_{ij}$$
$$s_{ij}^2 = \mathbf{r}_{ij}^T \Sigma_{ij}^{-1} \mathbf{r}_{ij}$$
$$e_{ij} = \hat{\rho}(s_{ij}^2)$$

where $\hat{\rho}(r^2) = \rho(r)$

Robust Non-Linear Minimization

- The corresponding Jacobians (partial derivatives) can be written as

$$\frac{\partial e_{ij}}{\partial s_{ij}^2} = \hat{\rho}'(s_{ij}^2)$$

$$\frac{\partial s_{ij}^2}{\partial \tilde{\mathbf{x}}_{ij}^2} = \Sigma_{ij}^{-1}\mathbf{r}_{ij}$$

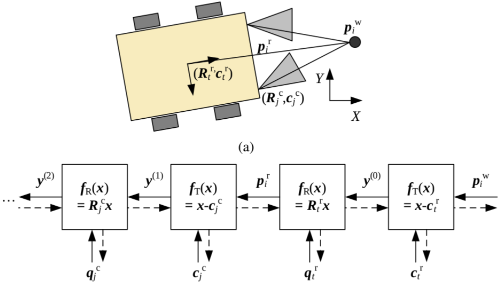
Robust Non-Linear Minimization

- The corresponding Jacobians (partial derivatives) can be written as

$$\frac{\partial e_{ij}}{\partial s_{ij}^2} = \hat{\rho}'(s_{ij}^2)$$
$$\frac{\partial s_{ij}^2}{\partial \tilde{\mathbf{x}}_{ij}^2} = \Sigma_{ij}^{-1}\mathbf{r}_{ij}$$

- The advantage of the chained representation is that it not only makes the computations of the partial derivatives and Jacobians simpler, but it can also be adapted to any camera configuration

Robust Non-Linear Minimization



- A camera rig and its associated transform chain: (a) as the mobile rig (robot) moves around in the world, its pose wrt the world at time t is captured by (R_t^r, c_t^r), each camera's pose with respect to the rig is captured by (R_j^c, c_j^c); (b) a 3D point with world coordinates $\mathbf{p}_i^w$ is first transformed into rig coordinates $\mathbf{p}_i^r$, and then through the rest of the camera-specific chain

Exploiting Sparsity

- Large bundle adjustment problems can require solving non-linear least squares problems with millions of measurements (feature matches) and tens of thousands of unknown parameters (3D point positions and camera poses)

Exploiting Sparsity

- Large bundle adjustment problems can require solving non-linear least squares problems with millions of measurements (feature matches) and tens of thousands of unknown parameters (3D point positions and camera poses)
- Unless some care is taken, these kinds of problems can become intractable since the (direct) solution of dense least squares problems is cubic in the number of unknowns

Exploiting Sparsity

- Fortunately, SfM is a *bipartite* problem in structure and motion

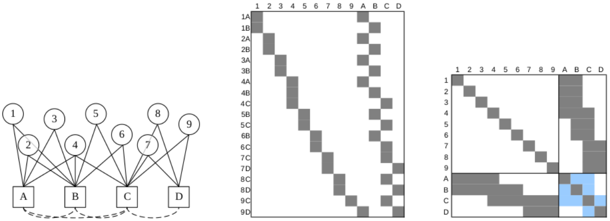
Exploiting Sparsity

- Fortunately, SfM is a *bipartite* problem in structure and motion
- Each feature point $\mathbf{x}_{ij}$ in a given image depends on one 3D point position $\mathbf{p}_i$ and one 3D camera pose $(R_j, \mathbf{c}_j)$

Exploiting Sparsity

- Fortunately, SfM is a *bipartite* problem in structure and motion
- Each feature point $\mathbf{x}_{ij}$ in a given image depends on one 3D point position $\mathbf{p}_i$ and one 3D camera pose $(R_j, \mathbf{c}_j)$
- If the values for all the points are known or fixed, the equations for all the cameras become independent, and vice versa

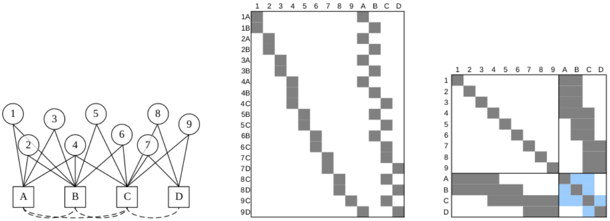
Exploiting Sparsity



(a) (b) (c)

- (a) Bipartite graph for a toy SfM problem and (b) its associated Jacobian J and (c) Hessian A

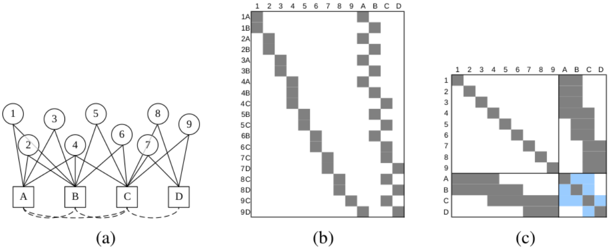
Exploiting Sparsity



(a) (b) (c)

- (a) Bipartite graph for a toy SfM problem and (b) its associated Jacobian J and (c) Hessian A
- Numbers indicate 3D points and letters indicate cameras

Exploiting Sparsity



- (a) Bipartite graph for a toy SfM problem and (b) its associated Jacobian J and (c) Hessian A
- Numbers indicate 3D points and letters indicate cameras
- The dashed arcs and light blue squares indicate the fill-in that occurs when the structure (point) variables are eliminated

Exploiting Sparsity

- If we order the structure variables before the motion variables in the Hessian matrix A we obtain a structure for the Hessian

Exploiting Sparsity

- If we order the structure variables before the motion variables in the Hessian matrix A we obtain a structure for the Hessian
- When such a system is solved using sparse Cholesky factorization, the *fill-in* occurs in the smaller motion Hessian A_{cc}

Exploiting Sparsity

- In more detail, the *reduced* motion Hessian is computed using the *Schur complement*

$$A'_{cc} = A_{cc} - A_{pc}^T A_{pp}^{-1} A_{pc}$$

where A_{pp} is the point (structure) Hessian (top left block), A_{pc} is the point-camera Hessian (top right block), and A_{cc} and A'_{cc} are the motion Hessians before and after the point variable elimination (bottom right block)

Exploiting Sparsity

- In more detail, the *reduced* motion Hessian is computed using the *Schur complement*

$$A'_{cc} = A_{cc} - A_{pc}^T A_{pp}^{-1} A_{pc}$$

where A_{pp} is the point (structure) Hessian (top left block), A_{pc} is the point-camera Hessian (top right block), and A_{cc} and A'_{cc} are the motion Hessians before and after the point variable elimination (bottom right block)

- Notice that A'_{cc} has a non-zero entry between two cameras if they see any 3D point in common

Exploiting Sparsity

- Bundle adjustment is now the standard method of choice for must SfM problems and is commonly applied to problems with hundreds of weakly calibrated images and tens of thousands of points

Exploiting Sparsity

- Bundle adjustment is now the standard method of choice for must SfM problems and is commonly applied to problems with hundreds of weakly calibrated images and tens of thousands of points
- However, as the problems become larger it becomes impractical to re-solve full bundle adjustment problems at each iteration

Exploiting Sparsity

- One approach to dealing with this problem is to use an extended Kalman filter which linearizes measurement and update equations around the current estimate

Exploiting Sparsity

- One approach to dealing with this problem is to use an extended Kalman filter which linearizes measurement and update equations around the current estimate
- Since points disappear from view (and old cameras become irrelevant), a variable state dimension filter (VSDF) can be used to adjust the set of state variables over time, e.g. by keeping only cameras and point tracks seen in the last k frames

Exploiting Sparsity

- While bundle adjustment and other robust non-linear least squares techniques are the methods of choice for most SfM problems, they suffer from initialization problems, i.e. they can get stuck in local energy minima if not started sufficiently close to the global optimum

Exploiting Sparsity

- While bundle adjustment and other robust non-linear least squares techniques are the methods of choice for most SfM problems, they suffer from initialization problems, i.e. they can get stuck in local energy minima if not started sufficiently close to the global optimum
- Many systems try to mitigate this by being conservative in what reconstruction they perform early on and which cameras and points they add to the solution

Application: Match and Move Augmented Reality

- One of the neatest applications of SfM is to estimate the 3D motion of a video or film camera, along with the geometry of a 3D scene, in order to superimpose 3D graphics or computer-generated images (CGI) on the scene

Application: Match and Move Augmented Reality

- One of the neatest applications of SfM is to estimate the 3D motion of a video or film camera, along with the geometry of a 3D scene, in order to superimpose 3D graphics or computer-generated images (CGI) on the scene
- In the visual effects industry, this is known as the *match move* problem since the motion of the synthetic 3D camera is used to render the graphics must be *matched* to that of the real-world camera

Application: Match and Move Augmented Reality

(a) (b)

- 3D augmented reality: (a) Darth Vader and a horde of Ewoks battle it out on a table-top recovered using real-time, keyframe-based structure from motion (b) a virtual teapot is fixed to the top of a real-world coffee cup, whose pose is re-recognized at each time frame

Application: Match and Move Augmented Reality

- For very small motions, or motion involving pure camera rotations, one or two tracked points can suffice to compute the necessary visual motion

Application: Match and Move Augmented Reality

- For very small motions, or motion involving pure camera rotations, one or two tracked points can suffice to compute the necessary visual motion
- For planar surfaces moving in 3D, four points are needed to compute the homography which can then be used to insert planar overlays (e.g. to replace the contents of advertising billboards during sporting events)

Application: Match and Move Augmented Reality

- The general version of this problem requires the estimation of the full 3D camera pose along with the focal length (zoom) of the lens, and potentially its radial distortion parameters

Application: Match and Move Augmented Reality

- The general version of this problem requires the estimation of the full 3D camera pose along with the focal length (zoom) of the lens, and potentially its radial distortion parameters
- When the 3D structure of the scene is known ahead of time, pose estimation techniques such as *view correlation* or *through-the-lens camera control* can be used

Application: Match and Move Augmented Reality

- For more complex scenes, it is usually preferable to recover the 3D structure simultaneously with the camera motion using SfM techniques

Application: Match and Move Augmented Reality

- For more complex scenes, it is usually preferable to recover the 3D structure simultaneously with the camera motion using SfM techniques
- The trick with using such techniques is that in order to prevent any visible jitter between the synthetic graphics and the actual scene, features must be tracked to very high accuracy and ample feature tracks must be available in the vicinity of the insertion locations

Application: Match and Move Augmented Reality

- Closely related to the match move problem is robotics navigation, where a robot must estimate its location relative to its environment while simultaneously avoiding any obstacles

Application: Match and Move Augmented Reality

- Closely related to the match move problem is robotics navigation, where a robot must estimate its location relative to its environment while simultaneously avoiding any obstacles
- This problem is often known as *simultaneous localization and mapping* (SLAM) or *visual odometry*

Application: Match and Move Augmented Reality

- Early versions of such algorithms used range-sensing techniques, such as ultrasound, laser range finders, or stereo matching, to estimate local 3D geometry which could then be fused into a 3D model

Application: Match and Move Augmented Reality

- Early versions of such algorithms used range-sensing techniques, such as ultrasound, laser range finders, or stereo matching, to estimate local 3D geometry which could then be fused into a 3D model
- Newer techniques can perform the same task based purely on visual feature tracking, sometimes not even requiring a stereo camera rig

Uncertainty and Ambiguities

- Since SfM involves the estimation of so many highly coupled parameters, often with no known ground truth, the estimates produced can exhibit large amounts of uncertainty

Uncertainty and Ambiguities

- Since SfM involves the estimation of so many highly coupled parameters, often with no known ground truth, the estimates produced can exhibit large amounts of uncertainty
- An example of this is the classic *bas-relief ambiguity*, which makes it hard to simultaneously estimate the 3D depth of a scene and the amount of camera motion

Uncertainty and Ambiguities

- A unique coordinate frame and scale for a reconstructed scene cannot be recovered from monocular visual measurements alone (when a stereo rig is used, the scale can be recovered if we know the distance (baseline) between the cameras)

Uncertainty and Ambiguities

- A unique coordinate frame and scale for a reconstructed scene cannot be recovered from monocular visual measurements alone (when a stereo rig is used, the scale can be recovered if we know the distance (baseline) between the cameras)
- This seven-degree-of-freedom *gauge ambiguity* makes it tricky to compute the covariance matrix associated with a 3D reconstruction

Uncertainty and Ambiguities

- A simple way to compute a covariance matrix that ignores the gauge freedom (indeterminacy) is to throw away the seven smallest eigenvalues of the information matrix (inverse covariance) whose values are equivalent to the problem Hessian A up to noise scaling

Uncertainty and Ambiguities

- A simple way to compute a covariance matrix that ignores the gauge freedom (indeterminacy) is to throw away the seven smallest eigenvalues of the information matrix (inverse covariance) whose values are equivalent to the problem Hessian A up to noise scaling
- After we do this, the resulting matrix can be inverted to obtain an estimate of the parameter covariance

Uncertainty and Ambiguities

- The other way in which gauge ambiguities affect SfM and, in particular, bundle adjustment is that they make the system Hessian matrix A rank-deficient and thus impossible to invert

Uncertainty and Ambiguities

- The other way in which gauge ambiguities affect SfM and, in particular, bundle adjustment is that they make the system Hessian matrix A rank-deficient and thus impossible to invert
- A number of techniques have been proposed to mitigate this problem, however in practice simply adding a small amount of the diagonal $\lambda \text{diag}(A)$ to the Hessian A itself usually works well

Application: Reconstruction from Internet Photos

- The most widely used application from SfM is the reconstruction of 3D objects from scenes and video sequences and collections of images

Application: Reconstruction from Internet Photos

- The most widely used application from SfM is the reconstruction of 3D objects from scenes and video sequences and collections of images
- There are many techniques for performing this task automatically without the need for any manual correspondence or pre-surveyed ground control points

Application: Reconstruction from Internet Photos

- A lot of these techniques assume that the scene is taken with the same camera and hence the images all have the same intrinsics

Application: Reconstruction from Internet Photos

- A lot of these techniques assume that the scene is taken with the same camera and hence the images all have the same intrinsics
- Many of these techniques take the results of sparse feature matching and the SfM computation and then compute dense 3D surface models using multiview stereo techniques

Application: Reconstruction from Internet Photos

- One application has been the use of SfM and multi-view stereo techniques on thousands of images taken from the Internet where very little is known about the cameras taking the photographs

Application: Reconstruction from Internet Photos

- One application has been the use of SfM and multi-view stereo techniques on thousands of images taken from the Internet where very little is known about the cameras taking the photographs
- Before the SfM computation can begin, it is first necessary to establish sparse correspondences between different pairs of images and then link such correspondences into feature tracks which associate individual 2D image features with global 3D points

Application: Reconstruction from Internet Photos

- Incremental structure from motion: Starting with an initial two-frame reconstruction of Trevi Fountain, batches of images are added using pose estimation, and their positions (along with the 3D model) are refined using bundle adjustment

Application: Reconstruction from Internet Photos

- To begin the reconstruction process, it is important to select a good pair of images where there are both a large number of consistent matches (to lower the likelihood of incorrect correspondences) and a significant amount of out-of-plane parallax (to ensure that a stable reconstruction can be obtained)

Application: Reconstruction from Internet Photos

- To begin the reconstruction process, it is important to select a good pair of images where there are both a large number of consistent matches (to lower the likelihood of incorrect correspondences) and a significant amount of out-of-plane parallax (to ensure that a stable reconstruction can be obtained)
- The EXIF tags associated with the images can be used to get good initial estimates for camera focal lengths, although this is not always necessary since these parameters are re-adjusted as part of the bundle adjustment process

Application: Reconstruction from Internet Photos

- Once an initial pair has been reconstructed, the pose of cameras that see a sufficient number of the resulting 3D points can be estimated and the complete set of cameras and feature correspondences can be used to perform another round of bundle adjustment

Application: Reconstruction from Internet Photos

- Once an initial pair has been reconstructed, the pose of cameras that see a sufficient number of the resulting 3D points can be estimated and the complete set of cameras and feature correspondences can be used to perform another round of bundle adjustment
- Unfortunately, as the incremental SfM continues to add more cameras and points it can become extremely slow

Application: Reconstruction from Internet Photos

- The direct solution of a dense system of $O(N)$ equations for the camera pose updates can take $O(N^3)$ times

Application: Reconstruction from Internet Photos

- The direct solution of a dense system of $O(N)$ equations for the camera pose updates can take $O(N^3)$ times
- While SfM problems are rarely dense, scenes such as city squares, have a high percentage of cameras that see points in common

Application: Reconstruction from Internet Photos

- Rerunning bundle adjustment after every few camera additions results in a quadratic scaling of the run time with the number of images in the dataset

Application: Reconstruction from Internet Photos

- Rerunning bundle adjustment after every few camera additions results in a quadratic scaling of the run time with the number of images in the dataset
- One approach to solving this problem is to select a smaller number of images for the original scene reconstruction and to fold in the remaining images at the very end

Finding Complementary Scene Information

- The most general algorithms for SfM make no prior assumptions about the objects or scenes that they are recovering

Finding Complementary Scene Information

- The most general algorithms for SfM make no prior assumptions about the objects or scenes that they are recovering
- However, in many cases the scene contains higher-level geometric primitives such as lines and planes

Finding Complementary Scene Information

- The most general algorithms for SfM make no prior assumptions about the objects or scenes that they are recovering
- However, in many cases the scene contains higher-level geometric primitives such as lines and planes
- These can provide information complementary to interest points and also serve as useful building blocks for 3D modeling and visualization

Finding Complementary Scene Information

- Sometimes, instead of exploiting regularity in the scene structure, it is possible to take advantage of a constrained motion model

Finding Complementary Scene Information

- Sometimes, instead of exploiting regularity in the scene structure, it is possible to take advantage of a constrained motion model
- For example, if the object of interest is rotating on a turntable (i.e. around a fixed but unknown axis) specialized techniques can be used to recover this motion

Finding Complementary Scene Information

- In other situations, the camera itself may be moving in a fixed arc around some center of rotation

Finding Complementary Scene Information

- In other situations, the camera itself may be moving in a fixed arc around some center of rotation
- Specialized capture setups, such as mobile stereo camera rigs or moving vehicles equipped with multiple fixed cameras, can also take advantage of the knowledge that individual cameras are (mostly) fixed w.r.t to the capture rig

Line-Based Techniques

- It is well known that pairwise epipolar geometry cannot be recovered from line matches alone, even if the cameras are calibrated

Line-Based Techniques

- It is well known that pairwise epipolar geometry cannot be recovered from line matches alone, even if the cameras are calibrated
- To see this, think of projecting the set of lines in each image into a set of 3D planes in space

Line-Based Techniques

- It is well known that pairwise epipolar geometry cannot be recovered from line matches alone, even if the cameras are calibrated
- To see this, think of projecting the set of lines in each image into a set of 3D planes in space
- You can move the two cameras around into any configuration you like and still obtain a valid reconstruction for 3D lines

Line-Based Techniques

- A widely used technique for matching 2D lines is based on the average of 15×15 pixel correlation scores evaluated at all pixels along their common line segment intersection

Plane-based Techniques

- In scenes that are rich with planar structures, e.g. in architecture and certain kinds of manufactured objects such as furniture, it is possible to directly estimate homographies between different planes using feature-based or intensity-based methods

Plane-based Techniques

- In scenes that are rich with planar structures, e.g. in architecture and certain kinds of manufactured objects such as furniture, it is possible to directly estimate homographies between different planes using feature-based or intensity-based methods
- In principle, this information can be used to simultaneously infer the camera poses and the plane equations, i.e. to compute plane-based SfM

Summary

- 2D and 3D point sets can be aligned and used to estimate both a camera's pose and its internal parameters

Summary

- 2D and 3D point sets can be aligned and used to estimate both a camera's pose and its internal parameters
- The converse problem of estimating the locations of 3D points from multiple images given only a sparse set of correspondences between images features is know as the structure from motion problem

Summary

- 2D and 3D point sets can be aligned and used to estimate both a camera's pose and its internal parameters
- The converse problem of estimating the locations of 3D points from multiple images given only a sparse set of correspondences between images features is know as the structure from motion problem
- Bundle adjustment is a general and useful approach to solving structure from motion which simultaneously updates all of the camera and 3D structure parameters