Computer Vision - 3D reconstruction from cameras

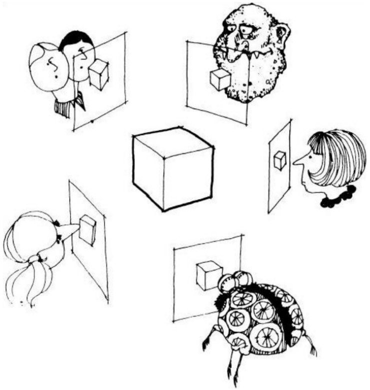
Artur Wilkowski

9 January 2022

Structure From Motion

Structure from Motion

Structure from Motion - estimation of 3D scene structure from a sequence of 2D camera images. Structure from Motion is a *photogrammetrical* method.

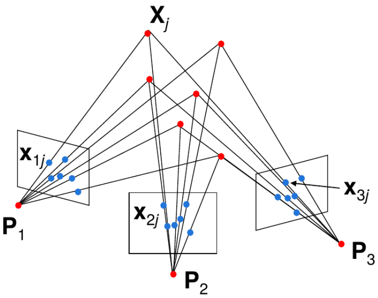


source: Rob Fergus

Structure from Motion

If we operate on points, the problem thus can be reformulated:

- Having given m images of n fixed 3D points...
- establish camera projection matrices P_i for each view...
- and establish 3D point coordinates X_j from a maximum of mn image correspondences x_{ij}



after: Rob Fergus

Structure from Motion

Our assumptions

- We are able to establish correspondences between points in images (by feature matching) - we won't discuss it
- Our camera is calibrated, intrinsic parameters are known, so we establish only $\mathbf{R_i}, \mathbf{t_i}$ not the whole projection matrix P_i

Inherent scale ambiguity

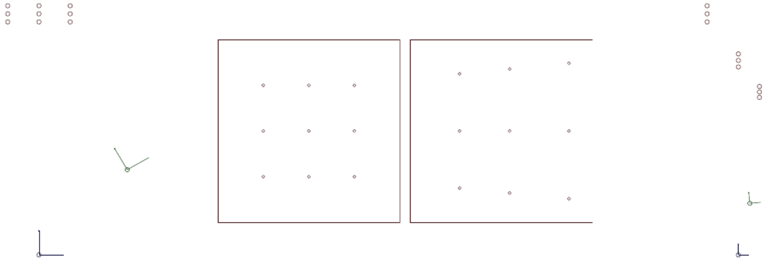
- We cannot establish a true scale of the scene from any number of images (unless we measure sth. in the image and in physical world).

Scene reconstruction from 2 views

Scene reconstruction from two views

- If we knew camera positions we could establish point coordinates by triangulation (stereovision problem)
- How to obtain relative camera positions from point correspondences?

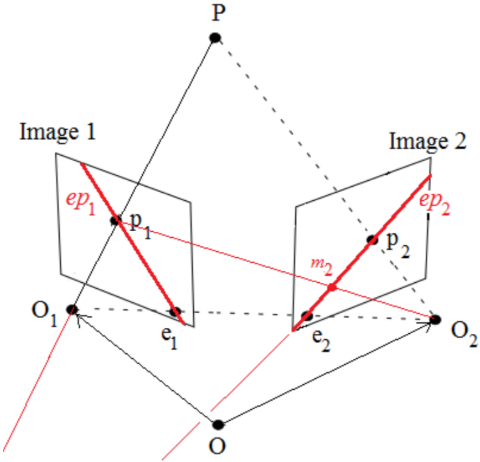
Attention: Scene reconstruction from two views may be non-unique for some point configurations!



Source: Segvic et al. Performance evaluation of the five-point relative pose...

Essential matrix revisited

Epipolar geometry:



Estimation of the essential matrix - 8 point algorithm

- 3D point normalized image coordinates are given as $p = (x, y, 1)^T$ in the first camera and as $p' = (x', y', 1)^T$ in the second camera.
- The coordinates p and p' are related by the essential matrix

$$(p')^T E p = 0$$

- Let us rewrite them using scalar values

$$(x', y', 1) \begin{pmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 0$$

- After expanding the equation we obtain

$$\begin{aligned} x'e_{11}x + x'e_{12}y + x'e_{13} + \\ y'e_{21}x + y'e_{22}y + y'e_{23} + \\ e_{31}x + e_{32}y + e_{33} = 0 \end{aligned}$$

Estimation of the essential matrix - 8 point algorithm

- By stacking at least 8 such equations we obtain a homogenous system of linear equations with 9 variables
- For this system we can solve the *Homogenous Least Squares Problem* to estimate the essential matrix E (the matrix is defined up to a scaling)
- However, the parameters of the essential matrix have only 5 degrees of freedom ...
- ...so in noisy case, the estimated matrix E will not satisfy the internal constraints of the essential matrix

$$\det E = 0,\ 2EE^TE - \mathrm{tr}(EE^T)E = 0$$

Estimation of the essential matrix - 8 point algorithm

In order to normalize the matrix

- Compute SVD of E

$$E = USV^T$$

  with U and V being orthogonal matrices and S - diagonal matrix with singular values

- Assuming that values in the diagonal of S are ordered decreasingly - set s_{33} to 0 and remaining ones to 1

$$S_{corr} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

- Use S_{corr} with U and V to recover the essential matrix

$$E_{corr} = US_{corr}V^T$$

Estimation of the essential matrix - 8 point algorithm

8 point algorithm properties

- Simple formulation and implementation $(+)$
- Requires 8-points to compute E (but remember - E has 5 degrees of freedom) (-)
- Usually requires normalization of E - recovering of internal constraints. (-)
- 8-point algorithm fails in some critical point configuration e.g. planar points

Estimation of the essential matrix - 8 point algorithm

Configuration	6-point algorithm	7-point algorithm	8-point algorithm
all points on surface of type (1) or (2)	$\sigma_9 = 0$	$\sigma_3 = 0$	$s_8 = 0$
all points but one on surface of type (1) or (2)	OK	OK	OK
all points on cylinder	$\sigma_8 = 0$	$\sigma_2 = 0$	$s_8 = 0$
all points but one on cylinder	OK	OK	OK
all points on plane	$\sigma_5 = 0$	$s_7 = 0$	$s_7 = 0$
all points but one on plane	OK	OK	$s_8 = 0$
all points but two on plane	OK	OK	OK

Source: *Johan Phillip. Critical configurations of the 5-,6-,7-, and 8-point algorithms ...*

Estimation of the essential matrix - 5 point algorithm

1. The system of homogenous equations is prepared as in the case of 8-point algorithm
2. The null-space of the above equation system is computed (using QR decomposition but SVD is also possible)
3. Four vectors X, Y, Z, W corresponding to the 4 smallest eigenvalues are selected, and the essential matrix is represented as a linear combination of these vectors

   $$E = xX + yY + zZ + W$$

   where x, y, z are unknown coefficients
4. The above equation is substituted into internal essential matrix constraint equations
5. After some processing a 10th degree (!!) polynomial is obtained
6. Different solutions of this polynomial lead to different hypotheses for the essential matrix

Estimation of the essential matrix - cheirality check

- Cheirality check is performed to reduce the number of hypotheses
  - Cheirality check - test physical feasibility of the solution - check if all recovered points are in front of both cameras
- 5 points are not enough to resolve most ambiguities (for random scenes 2.74 solutions are valid) so. . .
- The cheirality check should be performed on more (all available) points . . .
- However, some degenerate cases will still remain for 2 views. . . .

5 point algorithm properties

- Complex formulation and very complex implementation (-)
- Requires 5-points to compute E (which is the minimum) $(+)$
- Produces many outputs instead of failing in critical point configurations $(+)$
- "The only critical configuration when 5-point algorithm fails is for 4 or 5 points on the same line" (J.Phillip) $(+)$

Recovery of R and t from the essential matrix

Procedure to recover rotation and translation from the essential matrix E

- Decompose the essential matrix using SVD

$$E = USV^T \text{ with } det(U) > 0 \text{ and } det(V) > 0$$

- Define an auxiliary matrix D

$$D = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- Compute two possible rotation matrices

$$R_a = UDV^T \text{ and } R_b = UD^TV^T$$

- Compute a candidate translation vector $t = [u_{13}, u_{23}, u_{33}]^T$
- Compute four possible projection matrices for the second camera

$$P_A = [R_a|t_u], P_B = [R_a, -t_u], P_C = [R_b|t_u], P_D = [R_b| - t_u]$$

- Apply cheirality check to disambiguate solutions

3D reconstruction from 2 views

Assumption - both images were obtained from the same calibrated camera

1. Establish point correspondences in 2 images by feature (e.g. SIFT or SURF) detection and description or feature tracking
2. Compute the essential matrix (matrices) E relating corresponding points using 5-point algorithm, use RanSaC procedure to ingore outlier correspondences
3. Extract the relative camera rotation(s) (R) and translation $(\mathbf{t})$ from the essential matrix (matrices)
4. Perform cheirality check on all inliers to select a single solution for R and t
5. Perform triangulation to recover the 3D structure of the scene

Note:

1. The scale of translation $\mathbf{t}$ cannot be uniquely recovered
2. The whole rigid transformation may not be uniquely recovered (degenerate cases)

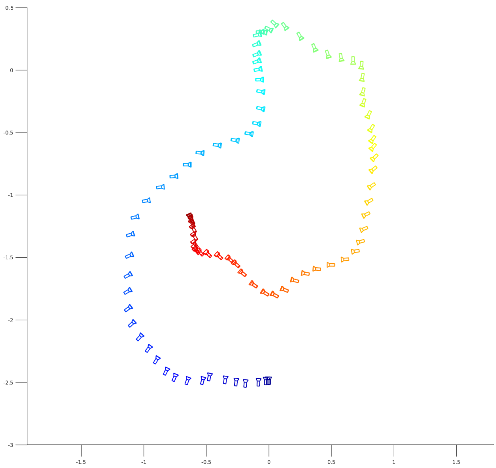
Scene reconstruction from multiple views

Scene reconstruction from a sequence of views

- We are given a sequence of images I_j, $j = 1 \ldots N$ from a single moving calibrated camera.
- We want to obtain a sequence of camera poses $R_i, \mathbf{t}_i$ together with a reconstruction of a scene observed by the camera.

Scene reconstruction from multiple views - approach 1

Approach 1



Source: *Maciej Stefanczyk*

Approach 1

1. Fix the global transformation for the first view to $[R_1^G|\mathbf{t_1^G}] = [I|0]$
2. For each subsequent image pair I_j and I_{j+1} and $j = 1 \dots N-1$ do:
   - Using procedure '3D reconstruction from 2 views', obtain a relative transformation $R_{j+1,j}$ and $\mathbf{t_{j+1,j}}$ from the local camera frame I_{j+1} to I_j.
   - Obtain a cloud $C_{j,j+1}$ by recovering a structure from triangulation of views I_j and I_{j+1}
   - Compute a global transformation for a view I_{j+1}:

$$[R_{j+1}^G|\mathbf{t_{j+1}^G}] = [R_j^G R_{j+1,j}|R_j^G \mathbf{t_{j+1,j}} + \mathbf{t_j^G}]$$

   - Transform points $C_{j,j+1}$ to the global frame (obtaining $C_{j,j+1}^G$ by applying $[R_j^G|\mathbf{t_j^G}]$

Are there any flaws in this approach?

Approach 1

1. Fix the global transformation for the first view to $[R_1^G|\mathbf{t_1^G}] = [I|0]$
2. For each subsequent image pair I_j and I_{j+1} and $j = 1 \dots N-1$ do:
   - Using procedure '3D reconstruction from 2 views', obtain a relative transformation $R_{j+1,j}$ and $\mathbf{t_{j+1,j}}$ from the local camera frame I_{j+1} to I_j.
   - Obtain a cloud $C_{j,j+1}$ by recovering a structure from triangulation of views I_j and I_{j+1}
   - Compute a global transformation for a view I_{j+1}:

$$[R_{j+1}^G|\mathbf{t_{j+1}^G}] = [R_j^G R_{j+1,j}|R_j^G \mathbf{t_{j+1,j}} + \mathbf{t_j^G}]$$

   - Transform points $C_{j,j+1}$ to the global frame (obtaining $C_{j,j+1}^G$) by applying $[R_j^G|\mathbf{t_j^G}]$

Scales of estimated translations $\mathbf{t_{j+1,j}}$, $\mathbf{t_{j+1,j}}$ are unrelated!

Approach 1 - corrected for scale

1. Fix the global transformation for the first view to $[R_1^G|\mathbf{t_1^G}] = [I|0]$
2. For each subsequent image pair I_j and I_{j+1} and $j = 1 \ldots N-1$ do:
   - Using procedure '3D reconstruction from 2 views', obtain a relative transformation $R_{j+1,j}$ and $\mathbf{t_{j+1,j}}$ from the local camera frame I_{j+1} to I_j.
   - Obtain a cloud $C_{j,j+1}$ by recovering a structure from triangulation from views I_j and I_{j+1}
   - For $j \geq 2$ find at least two corresponding point pairs in $C_{j,j+1}$ and $C_{j-1,j}$ - adjust the scale of $\mathbf{t_{j+1,j}}$ according to distances between points in both clouds
   - Compute a global transformation for a view I_{j+1}:

$$[R_{j+1}^G|\mathbf{t_{j+1}^G}] = [R_j^G R_{j+1,j}|R_j^G \mathbf{t_{j+1,j}} + \mathbf{t_j^G}]$$

   - Transform points $C_{j,j+1}$ to the global frame (obtaining $C_{j,j+1}^G$) by applying $[R_j^G|\mathbf{t_j^G}]$

Approach 1 properties:

- Two subsequent views must have at least 5 points in common (and more for robust estimation) - possible problems with marker-based approaches and sparse features (-)
- Two subsequent view pairs must share at least 2 points to determine scale (however we may want to estimate common scale on cycle close) (-)
- There is a significant chance of experiencing scene reconstruction from 2-views ambiguities (-)
- One obtains a graph of poses, with relative transformatin between views, which is suitable for loop closing (e.g. using *g2o*) algorithm (+)

Scene reconstruction from multiple views - approach 2

Perspective n-point algorithm

- Perspective n-point (PnP) algorithm is used to find camera orientation with respect to a known point cloud using n-camera-3D point correspondences
- This is a subproblem of camera calibration, but we assume that camera intrinsic parameters are already known
- The problem can be formulated as: *Find R and t basing on correspondences* $p_i \leftarrow P_i$ *for* $i = 1 \ldots N$

$$s_i p_i = K[R|t]P_i \text{ for } i = 1 \ldots n$$

- Minimum 3 points are required to solve equations and 4 points to obtain a unique solution

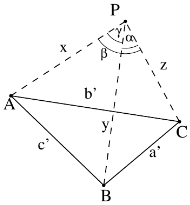
Perspective n-point algorithm

Simple solutions:

- Use a DLT algorithm to solve 'calibration' problem
  - 6 points required
  - in effect a projection matrix P is obtained, which can be decomposed into R and t (and K_int if it is not known. . .)
  - R and t can be further refined e.g. using non-linear optimization
  - the method fails for planar points
- Perform non-linear optimization to searching for the minimum-reprojection-error 'from scratch'

P3P algorithm [Gao]

$$\begin{cases} Y^2 + Z^2 - YZp - a'^2 &= 0 \\ Z^2 + X^2 - XZq - b'^2 &= 0 \\ X^2 + Y^2 - XYr - c'^2 &= 0 \end{cases}$$



- The algorithm uses 3 point correspondences to solve the following system of quadratic equations with unknowns X, Y, Z which is a straightforward application of the cosine theorem.
- $|PA| = X, |PB| = Y, |PC| = Z, \alpha = \angle BPC, \beta = \angle APC, \gamma = \angle APB, p = 2cos\alpha, q = 2cos\beta, r = 2cos\gamma, |AB| = c', |BC| = a', |AC| = b'$
- cosines are computed from the normalized image coordinates (also using cosine theorem)

Perspective n-point algorithm

P3P algorithm [Gao] cont.

- The results need to be disambiguated using the 4th point correspondence
- The algorithm requires arbitrary selection of 3 points (no n-point optimization is performed)

Perspective n-point algorithm

EPnP algorithm

- For any $n \geq 4$ the points in the world coordinate frame (as well as in camera coordinate frame) are expressed as a weighed sum of just 4 control points (chosen arbitrarily)

$$P_i = \sum_{j=1}^{4} \alpha_{ij} c_j^w, \text{ with } \sum_{j=1}^{4} \alpha_{ij} = 1$$

- The algorithm reconstructs control points in camera frame.
- By projecting control points in the camera frame onto the image we obtain a set of homogenous equations, with the control points as variables. HLLSE is used to solve the system.
- Depending on the number of input points (and their configuration), the nullspace can have $N = 1 \dots 4$ dimensions and the solution can be given as

$$\mathbf{x} = \sum_{i=1}^{N} \beta_i \mathbf{v_i}$$

EPnP algorithm

- Thus our solution (parameters of control points in camera frame) is constrained to a linear combination of **at most** 4 known vectors (compare: 5-point algorithm...) - e.g. for six or more point correspondence we have only a single β - and only scale need to be adjusted
- Values of β_i are established taking into account distances between the estimated control points in camera frame and the control points in world frame (which must be identical)
- $\mathbf{R}$ and $\mathbf{t}$ are computed by 3D pose estimation from the computed control points in the camera and world coordinate frames
- β_i **may be** further refined using Gauss-Newton method.

EPnP algorithm properties

- Requires at least 4 points
- Is able to optimize using arbitrary number of correspondences
- Closed-form solution (does not require initalization)
- Works for planar and non-planar configurations
- Offers a very good accuracy of estimation

Perspective n-point algorithm

Practical considerations

- Points in 3D are typically estimated from 2D images ...
- ... so the may have associated feature descriptors like SIFT or SURF
- The features may be matched to obtain image-3D correspondences
- **RanSaC should be employed to remove outliers**

Approach 2

1. Fix the global transformation for the first view to $[R_1^G|\mathbf{t_1^G}] = [I|0]$
2. Performed two view scene reconstruction from the first and the second view. Obtain the transformation R_2^G, $\mathbf{t_2^G}$ and the point cloud C_2^G
3. For each subsequent view j:
   - align the j-th sensor pose by matching projected points with the current scene cloud C_{i-1}^G - obtain sensor transformation R_j^G, $\mathbf{t_j^G}$ (via some verision PnP algorithm)
   - triangulate 3D points from all point correspondences in view j and all views $1 \ldots j-1$
   - merge triangulated points and the point cloud C_{i-1}^G into a new cloud C_i^G

Approach 2

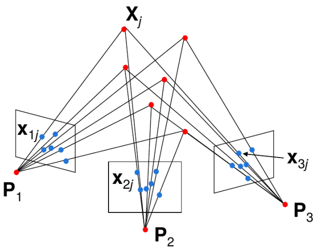
Approach 2 properties:

- Only two first views must have at least 5 points in common (and more for robust estimation) (+)
- Any subsequent view must contain only 4 points common wht the set reconstructed so far (but the points could be reconstructed from different views!). (+)
- Scale is consistent with the first image pair used in reconstruction (+)
- The method is quite robust to different point configurations (+)
- One does not obtain directly a graph of poses, with relative transformation between views, loop closing is more challenging (-)

Bundle Adjustment

Bundle adjustment

- Most of points are visible in more than 2 views
- We can use overplus point correspondences to correct different types of errors:
  - Point detection innacuracies
  - View matching innacuracies
  - Scale, position and angle drift



source: Rob Fergus

Bundle adjustment

Procedure

- Input data (parameters): 2D point coordinates in views, let p_{ij} be the projection of 3D point j in a view i, intrinsic camera parameters
- Optimized variables: extrinsic parameters for all cameras T_i (with a fixed T_1), 3D points coordinates P_i
- Initial approximation - extrinsic camera parameters and points positions obtained from the incremental structure from motion
- Minimized function: reprojection error for all views and all available 3D-2D point pairs

$$rErr = \sum_{(i,j)} w_{ij} \|proj(P_j, T_i) - p_{ij}\|^2$$

with $proj$ being the perspective projection function for view i, w_{ij} - weight adjusted during optimization, (i, j) - available viewpoint pairs

Properties

- Large BA problems may involve thousands of parameters (performed off-line)
- Point correspondence graph i sparse - the sparsity of the Jacobian matrix **may/must** be utilized to speed-up computations
- Some minimum representation of rotations is convenient (e.g. axis-angle)
- Sophisticated weighing schemes can be used - to accommodate outliers
- It is possible to treat intrinsic parameters as variables (so called autocalibration)