


FACULTY OF MATHEMATICS,
PHYSICS AND INFORMATICS
Comenius University
Bratislava

3D Vision

Lecture 6: Multiple Views and SfM

Ing. Viktor Kocur, PhD.

28.3.2023

Contents

Reconstruction from Two Views

Recall that in last lectures we used a set of correspondences $\boldsymbol{x}_i \leftrightarrow \boldsymbol{x}_i'$. We then found P and P' as well as the original points $\boldsymbol{X}_i$ in the real scene such that for all i:

$$\boldsymbol{x}_i \sim P\boldsymbol{X}_i \quad \boldsymbol{x}_i' \sim P'\boldsymbol{X}_i. \tag{1}$$

Knowing P and P' we now also know $K, K', R, \boldsymbol{t}$ such that $P = K[I|\boldsymbol{0}], P' = K[R, \boldsymbol{t}]$. The coordinates of $\boldsymbol{X}$ are known in the same coordinate system as the camera.

New Notation

Since we will be adding new cameras we will introduce new notation. We will consider cameras $P_0 = K_0[I|\mathbf{0}], P_1 = K_1[R_1|\boldsymbol{t}_1],\ ...,\ K_n[R_n|\boldsymbol{t}_n]$. We will also consider real points $\boldsymbol{X}_i$ in the scene.

We will also denote images of $\boldsymbol{X}_i$ in j-th view as $\boldsymbol{x}_i^j \sim P_j \boldsymbol{X}_i$. For most of this lecture we will utilize point simple correspondences of the form $\boldsymbol{x}_i^j \leftrightarrow \boldsymbol{x}_i^k$. However we will briefly discuss the case when we may consider triplets of correspondences.

Three-View Geometry

If we consider a set of triplets of corresponding points in the form $\boldsymbol{x}_i^0 \leftrightarrow \boldsymbol{x}_i^1 \leftrightarrow \boldsymbol{x}_i^2$ we could treat them simply as separate correspondences $\boldsymbol{x}_i^0 \leftrightarrow \boldsymbol{x}_i^1, \boldsymbol{x}_i^1 \leftrightarrow \boldsymbol{x}_i^2, \boldsymbol{x}_i^0 \leftrightarrow \boldsymbol{x}_i^2$ and calculate their respective fundamental or essential matrices.

However, given the whole triplet we can use fewer correspondences than we would need when considering pairwise views. The triplet introduces constraints leading to only 18 degrees of freedom for the 3 fundamental matrices. This means that we only need 6 triplets to obtain them.

N-View Geometry

We can also extend this to n views with multiple correspondences. In general given n views leaves $11n - 15$ degrees of freedom for the structure computation (fundamental matrices and points). Consider that we have m correspondences which provide $2mn$ measurements in total (2 coordinates in each view). The system has $11n - 15 + 3m$ degrees of freedom. In order to establish structure we can derive

$$2nm \geq 11n - 15 + 3m \tag{2}$$

$$m \geq \frac{11n - 15}{2n - 3} = 5 + \frac{n}{2n - 3} \tag{3}$$

which needs to hold in order for us to determine the structure. Note that in case of equality there may be multiple solutions and in case of inequality we have an overdetermined system usually with a single solution. This might be complicated by degenerate cases (e.g. points on a plane or 3+ camera centres on a line).

N-View Geometry

In order to grasp the mathematics behind n-view geometry additional objects such as the multiview matrix and tri-/quadri-/multifocal tensors are used. We will not cover them in this lecture.

We will not attempt to compute the structure directly using these mathematical objects, but instead we will consider the problem in incremental fashion by iteratively adding additional views. In between the steps we will consider the problem holistically via non-linear optimization known as bundle adjustment. This is not the only feasible approach, but the most straightforward one.

Sparse 3D Reconstruction Taxonomy

There are several names used for the problem of sparse 3D reconstruction:

- **Structure from Motion** (SfM) - usually assumes that images may come from multiple calibrated or uncalibrated cameras and the images may be processed in any order
- **Simultaneous Localization and Mapping** (SLAM) - usually assumes that images come from a single camera or a given set of calibrated cameras (e.g. mounted on a vehicle) and have to be processed online - in the sequence they are obtained and in real-time
- **Visual Odometry** - this is similar to SLAM, but usually does not require to output the reconstructed pointcloud and does not require consistency along the whole path

These distinctions are not so clear. The term (Visual) SLAM originated in robotics and SfM in vision, but in essence they are both very similar and may be used interchangeably as they produce similar data on output.

Dense Reconstruction

On the other hand if we want to consider a problem of obtaining a dense pointcloud (or even a mesh or textured 3D model) we need dense reconstruction which is called **Multiple View Stereo** (MVS).

Adding the third view

Let us consider that we have three views with K_0, K_1, K_2 known as well as R_1 and $\boldsymbol{t}_1$. We also used correspondences $\boldsymbol{x}_i^0 \leftrightarrow \boldsymbol{x}_i^1$ to obtain points $\boldsymbol{X}_i$. We then want to utilize correspondences $\boldsymbol{x}_i^0 \leftrightarrow \boldsymbol{x}_i^2$ and $\boldsymbol{x}_i^1 \leftrightarrow \boldsymbol{x}_i^2$ to obtain $R_2, \boldsymbol{t}_2$ as well as find additional points $\boldsymbol{X}_i$.

We know that for some $\boldsymbol{x}_i^0 \leftrightarrow \boldsymbol{X}_i$ and $\boldsymbol{x}_i^1 \leftrightarrow \boldsymbol{X}_i$ we can use this to obtain correspondences of the form $\boldsymbol{x}_i^2 \leftrightarrow \boldsymbol{X}_i$. This enables us to consider correspondences between all of the images at the same time to obtain R_2 and $\boldsymbol{t}_2$. Afterwards we can triangulate new points from those correspondences which haven't been associated with a 3D point $\boldsymbol{X}_i$ yet.

Absolute Pose Estimation

We therefore have to solve the following task: given K and set of correspondences $\boldsymbol{x}_i \leftrightarrow \boldsymbol{X}_i$ find $R, \boldsymbol{t}$ such that

$$\boldsymbol{x}_i \approx P\boldsymbol{X}_i = K[R|\boldsymbol{t}]\boldsymbol{X}_i. \tag{4}$$

This problem is known as the **absolute pose estimation** or **perspective-n-point** (PnP). The problem we have solved previously using the essential matrix from pairs of image correspondences is known as **relative pose estimation**.

Minimal Solution to PnP - P3P

We can solve the PnP problem with using at least three correspondences. The solution to the problem has been known since 1841. We will show a solution using distances between points. We will use the camera center $\boldsymbol{C} = -R^T\boldsymbol{t}$. Let us also define:

$$a = |\boldsymbol{X}_2 - \boldsymbol{X}_3|, \tag{5}$$

$$b = |\boldsymbol{X}_1 - \boldsymbol{X}_3|, \tag{6}$$

$$c = |\boldsymbol{X}_1 - \boldsymbol{X}_2]. \tag{7}$$

Let us also consider unprojected points $\hat{\boldsymbol{x}}_i \sim K^{-1}\boldsymbol{x}_i \sim (u_i, v_i, 1)^T$. We can then represent the points $\boldsymbol{X}_i$ as:

$$\boldsymbol{X}_i = s_i \boldsymbol{j}_i = \frac{s_i}{\sqrt{u_i^2 + v_i^2 + 1}} \begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix}, \tag{8}$$

where s_i is a scalar representing and $\boldsymbol{j}_i$ is a unit vector representing direction. We also define the following angles

$$\cos(\alpha) = \langle \boldsymbol{j}_2, \boldsymbol{j}_3 \rangle, \tag{9}$$
$$\cos(\beta) = \langle \boldsymbol{j}_1, \boldsymbol{j}_3 \rangle, \tag{10}$$
$$\cos(\gamma) = \langle \boldsymbol{j}_1, \boldsymbol{j}_2 \rangle. \tag{11}$$

P3P

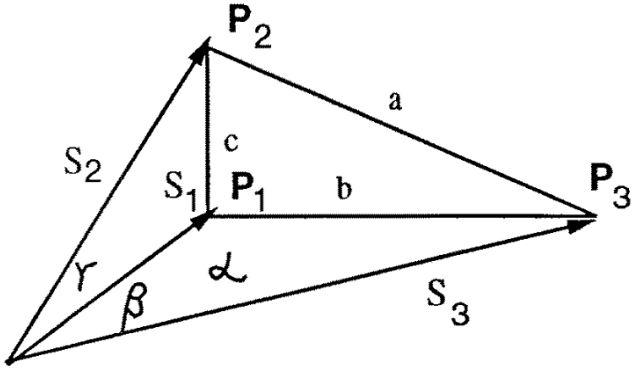


Image adopted from: Bert M Haralick et al. “Review and analysis of solutions of the three point perspective pose estimation problem.” In: *International journal of computer vision* 13 (1994), pp. 331–356

Gunert's solution

Given the geometry of the problem Gunert has (in 1841) used the law of cosines in the form:

$$s_1^2 + s_2^2 - 2s_1 s_2 \cos(\gamma) = c^2, \tag{12}$$
$$s_1^2 + s_3^2 - 2s_1 s_3 \cos(\beta) = b^2, \tag{13}$$
$$s_2^2 + s_3^2 - 2s_2 s_3 \cos(\alpha) = a^2. \tag{14}$$

We can then employ a simple substitution $s_2 = us_1$ and $s_3 = vs_2$ we can then obtain:

$$s_1^2 = \frac{c^2}{1 + u^2 - 2u\cos(\gamma)} \quad s_1^2 = \frac{b^2}{1 + v^2 - 2v\cos(\beta)} \quad s_1^2 = \frac{a^2}{u^2 + v^2 - 2uv\cos(\alpha)}. \tag{15}$$

Gunert's Solution

The equations (15) can be combined into a single polynomial such that we can express u in terms of v and obtain a 4th order polynomial in v.

The polynomial can have up to four real solutions from which the rest of the unkowns can be determined. Note that there are many other approaches to this problem. Different approaches may have different issues when used in numerical calculations.

EPnP and PnPf

We can also use $n \geq 4$ points for a non-minimal solution. EPnP expresses the points as weighted sums of so-called control points. This algorithm has an $O(n)$ time complexity and handles both planar and non-planar points.

The PnP problem can also be extended to estimate the intrinsic camera parameters. For instance the PnPf problem can also find the focal length of the camera.

In practice we will use algorithms implemented in libraries such as OpenCV and PoseLib. Similarly to relative pose estimation we can use these algorithms within RANSAC to obtain the best fit to the given correspondences.

Triangulation

So far we have used the points that already had corresponding 3D points $\boldsymbol{X}_i$. Now we can also use the remaining correspondences $\boldsymbol{x}_i^0 \leftrightarrow \boldsymbol{x}_i^2$ and $\boldsymbol{x}_i^1 \leftrightarrow \boldsymbol{x}_i^2$ to obtain more $\boldsymbol{X}_i$.

Note that we have to keep track of which image points in which views correspond to which 3D points $\boldsymbol{X}_i$. We need this information to perform further steps in the pipeline.

Bundle Adjustment

After triangulation we perform **bundle adjustment** by optimizing the camera parameters and positions of the 3D points to minimize the re-projection error:

$$\sum_i \sum_j \alpha_j^i d(P_i \boldsymbol{X}_j, \boldsymbol{x}_j^i)^2, \tag{16}$$

where i goes over the views and j goes over the points and α_j^i is 1 if $\boldsymbol{X}_j$ is visible in the i-th view and 0 otherwise.

Numerical Optimization

For some problems it is possible to find global optima exactly. However, this is often not the case. Under some assumptions we can use various numerical methods to find

$$\boldsymbol{x}_{min} = \arg\min F(\boldsymbol{x}). \tag{17}$$

We will now cover some optimization methods to derive the most commonly used approach.

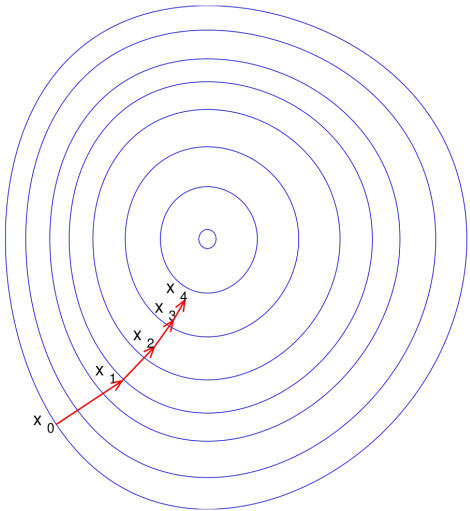
Gradient Descent

We can find a local minimum using gradient descent. This can be performed in an iterative manner given by:

$$\boldsymbol{x}^{n+1} = \boldsymbol{x}^{n} - \eta \frac{\partial F}{\partial \boldsymbol{x}}(\boldsymbol{x}), \tag{18}$$

where $\boldsymbol{x}^{n}$ is the vector at n-th iteration and η is the step size. This type of optimization needs to be initialized with a good initial estimate $\boldsymbol{x}^{0}$. If the function F is convex and has some properties which make it possible to select good η then this algorithm will converge to global minimum. However, this make take many iterations and the path in the space of vectors $\boldsymbol{x}$ can be quite suboptimal.

Gradient Descent



The Newton Method

We can use the Taylor expansion of F to obtain

$$F(\boldsymbol{x}) \approx F(\boldsymbol{x}^n) + \boldsymbol{g}^T(\boldsymbol{x} - \boldsymbol{x}^n) + (\boldsymbol{x} - \boldsymbol{x}^n)^T H(\boldsymbol{x} - \boldsymbol{x}^n), \tag{19}$$

where $\boldsymbol{g}$ is the gradient of F evaluated at $\boldsymbol{x}^n$:

$$\boldsymbol{g} = \left(\frac{\partial F}{\partial x_1}(\boldsymbol{x}^n), \frac{\partial F}{\partial x_2}(\boldsymbol{x}^n), \ldots \frac{\partial F}{\partial x_k}(\boldsymbol{x}^n) \right)^T, \tag{20}$$

and H is the Hessian matrix of F evaluated at $\boldsymbol{x}^n$:

$$H_{i,j} = \frac{\partial^2 F}{\partial x_i \partial x_j}(\boldsymbol{x}^n). \tag{21}$$

The Newton Method

To find a local optimum we may be interested in solving

$$\frac{\partial F}{\partial \boldsymbol{x}} = 0 = \boldsymbol{g} + H(\boldsymbol{x} - \boldsymbol{x}^n). \tag{22}$$

This leads to a step:

$$\boldsymbol{x}^{n+1} = \boldsymbol{x}^n - H^{-1}\boldsymbol{g}. \tag{23}$$

In practice we may again add a multiplicative factor η in front of $H^{-1}\boldsymbol{g}$. This is a second-order method (because it uses H). It usually converges significantly faster than standard gradient descent. The problem is that this method requires the inversion of H which might be quite costly when dealing with large optimization problems.

The Gauss-Newton Method

If F has a specific form we can approximate H^{-1}. Consider an F of the form

$$F = \sum_i r_i^2, \tag{24}$$

where is r_i is called the i-th residual. In this case the the gradient is

$$g_j = 2 \sum_i r_i \frac{\partial r_i}{\partial x_j}. \tag{25}$$

and the Hessian which can be approximated

$$H_{j,k} = 2 \sum_i \left(\frac{\partial r_i}{\partial x_j} \frac{\partial r_i}{\partial x_k} + r_i \frac{\partial^2 r_i}{\partial x_j \partial x_k} \right) \approx 2 \sum_i J_{i,k} J_{i,j}, \tag{26}$$

where $J_{i,j} = \frac{\partial r_i}{\partial x_j}$ is the Jacobi matrix.

The Gauss-Newton Method

The approximation $H \approx 2J^T J$ together with $\boldsymbol{g} = 2J^T \boldsymbol{r}$ leads to the Gauss-Newton algorithm:

$$x^{n+1} = \boldsymbol{x}^n - (J^T J)^{-1} J^T \boldsymbol{r}. \tag{27}$$

This method does not require the calculation of the Hessian matrix, but in order for the approximation to be correct the following has to hold:

$$\left| r_i \frac{\partial^2 r_i}{\partial x_j \partial x_k} \right| \ll \left| \frac{\partial r_i}{\partial x_j} \frac{\partial r_i}{\partial x_k} \right| \tag{28}$$

The Levenberg-Marquardt Method

The Newton method can be modified to

$$\boldsymbol{x}^{n+1} = \boldsymbol{x}^n - (H + \lambda I)^{-1}\boldsymbol{g}. \tag{29}$$

To create a hybrid between gradient descent with step size $\frac{1}{\lambda}$ for $\lambda \mapsto +\infty$ and the Newton method for $\lambda = 0$. In the same manner it is possible to modify the Gauss-Newton method:

$$x^{n+1} = \boldsymbol{x}^n - (J^T J + \lambda I)^{-1} J^T \boldsymbol{r}, \tag{30}$$

or in a more adaptive way:

$$x^{n+1} = \boldsymbol{x}^n - \left(J^T J + \lambda \mathrm{diag}(J^T J)\right)^{-1} J^T \boldsymbol{r}. \tag{31}$$

Thanks to the introduced damping this method is more robust than the standard Gauss-Newton algorithm.

Bundle Adjustment

We can use the non-linear iterative optimization algorithms such as Levenberg-Marquardt to iteratively improve the estimate of the local minimum for

$$\sum_i \sum_j \alpha_j^i d(P_i \boldsymbol{X}_j, \boldsymbol{x}_j^i)^2 = \sum_k r_k. \tag{32}$$

The iterative algorithm usually runs until $|\boldsymbol{x}^{n+1} - \boldsymbol{x}^n|$ falls below a preselected threshold. In some pipelines optimized for speed only a predetermined number of iterations may be performed. Sometimes we may optimize for the positions of points separately from the camera intrinsics and extrinsics.

Retriangulation

After performing the optimization we obtain new values for the camera intrinsics and extrinsics a the point positions. We may then reconsider whether a correspondence $\boldsymbol{x}_i^j \leftrightarrow \boldsymbol{X}_i$ is still valid. This is known as **retriangulation**.

Incremental SfM Pipeline

1. Solve the relative pose problem for the first two views
2. Triangulate the points based on the first two views
3. Perform bundle adjustment
4. Solve the absolute pose problem for an additional view
5. Triangulate points from the additional view
6. Peform bundle adjustment
7. Perform retriangulation
8. Repeat from 4 until all views are added

Incremental SfM Order

In case of a small scene fully captured in all views we can start with any two views and add them in any order. However, this is not the case when larger scenes are considered. In that case it is important to add the views in an order which works well. We can use additional information, correspondence graphs and heuristics for this.

In Visual Odometry and SLAM the views are temporally ordered. However, it is still meaningful to select which views are used for registration. This is known as keyframe selection.

Alternatives to Incremental Approaches

- Hierarchical SfM - This builds models from pairs of images, then pairs the pairs and so on. We also have to select the proper order usually based on proximity or correspondence graphs. Good for parallelization.
- Global SfM - This often consists of steps of rotation averaging, relative translation estimation, translation registration, triangulation and bundle adjustment. This can be parallelized better and result in accurate reconstruction, but the step of rotation averaging may prove to not be robust enough in some cases.

Indirect vs Direct SfM

So far we have shown an **indirect** SfM method. Such methods rely on detected keypoints. Another approach is to consider all pixels in images and minimize

$$\sum_{i=1}^{N} d(I^j(\boldsymbol{x}_i^j), I^k(x(\boldsymbol{x}_i^j, R^{jk}, \boldsymbol{t}^{jk}, K^j, K^k)), \tag{33}$$

which represents the error of pixels matching given the estimated structure. This can be done on all pixels, or on a subset, e.g. only edges or other interesting regions. These types of approaches are known as **direct** methods.