Multiview Matching

November 17, 2021

1 Introduction

The usual way to estimate the position of an object imaged by multiple cameras with overlapping field of view is via triangulation. This technique implies that the position of the object in the images is known so as the intrinsic and extrinsic parameter for each one of the cameras. If only one object is present in the scene, the number of detections/positions in each image will be at most one, which means that the correspondences between the detections are given already. If the scene is populated of a number of objects and that we only dispose of a detector that is incapable of distinguish one object from another, finding the correct correspondences, that is finding the detections that correspond to the same physical object, is not trivial.

In this document we describe how this problem can be solved optimally by exploiting the epipolar geometry and Integer Linear Programming (ILP).

2 Background

2.1 Epipolar Geometry

Epipolar geometry describe the geometry between two views. It is independent of the scene structure and only depends on the camera internal parameters and the relative extrinsic ones. The epipolar geometry is usually used to solve the problem of two-views matching and searching for corresponding points.

Supposing that we dispose of two views with camera centers $\mathbf{C}$ and $\mathbf{C}'$ and an image point $\mathbf{x}$ for the first and $\mathbf{x}'$ for the second view where both correspond to the same objects $\mathbf{X}$ as illustrated in Fig. 1. The line/ray that originates from $\mathbf{C}$ and passing from $\mathbf{x}$ correspond to a line in the second view passing from the epipole e' and $\mathbf{x}'$, $\mathbf{x} \mapsto \mathbf{l}'$. The epipole is the image position of the other camera. The geometrical relationship between the two views can be expressed in the form of a matrix called *fundamental matrix*. The fundamental matrix F can be estimated from a set of image points in both views that correspond to the same object in the scene to satisfy $\mathbf{x}F\mathbf{x}' = 0$. It can be shown then that give F and a point in one image $\mathbf{x}$, the corresponding epiline in the second view can be compute using $\mathbf{l}' = F\mathbf{x}$.

3 Blind matching using multiple views constraints

3.1 Problem formulation

In this section we formulate the problem of finding the set of correspondences across views of a set of noisy image locations supposed to correspond to detections of objects in the scene. We define one image location as a point in a 2D space $\mathbf{x} \in \mathcal{R}^2$. This can either correspond to a correct detection or a false positive one. Let $\{\mathbf{x}_i^j\}_{i=1}^{N^j}$ be a set of image locations in view $j \in J$ where the size of the set N^j can vary from one view to another. Assuming that the cameras have overlapping field of view and that all the camera parameters are known, the goal is to find the set of correct correspondences for the same object (if they exist) such that at most one point is selected per view. Note that there can be views where the object is not visible or not detected. Formally, it can be expressed as finding one set of corresponding indexes per object $\{(i^j, i^k) | j, k \in K, j \neq k\}$ where i^j is the index of the selected point in view j and $K \subseteq J$ is a subset that can contain all or part of the views. As an example, if we dispose of four views, the number of correspondences in a set of correspondences for one object would be at most six.

Since the camera parameters are given, to find these correspondences we rely on the epipolar geometry. As we have shown in the background section, if we are given the projection of the same object in two views $\mathbf{x}$ and $\mathbf{x}'$, the epiline $\mathbf{l}' = F\mathbf{x}$ has to cross $\mathbf{x}'$. The same holds for epiline $\mathbf{l}$. It become trivial at this point to find correspondences

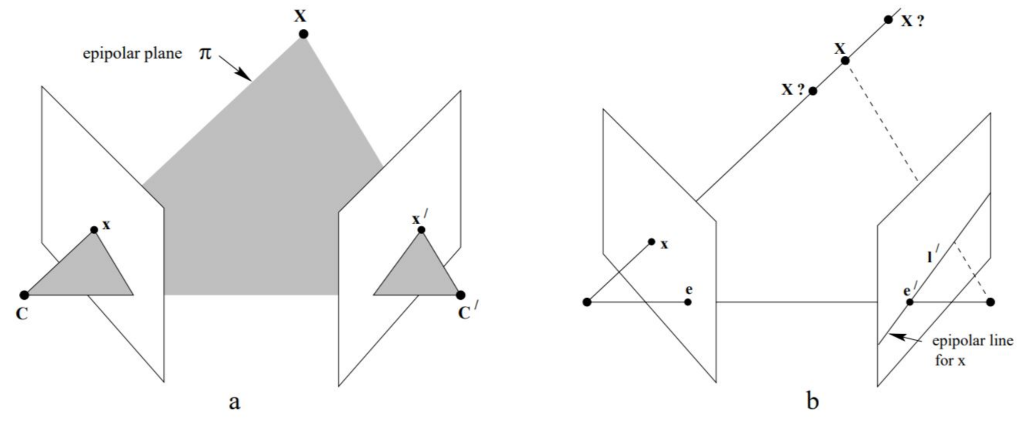

Figure 1: **Point correspondence geometry.** (a) The two cameras are indicated by their centres $\mathbf{C}$ and $\mathbf{C}'$ and image planes. The camera centres, 3-space point $\mathbf{X}$, and its images $\mathbf{x}$ and $\mathbf{x}'$ lie in a common plane π. (b) An image point $\mathbf{x}$ back-projects to a ray in 3-space defined by the first camera centre, $\mathbf{C}$, and $\mathbf{x}$. This ray is imaged as a line $\mathbf{l}'$ in the second view. The 3-space point $\mathbf{X}$ which projects to $\mathbf{x}$ must lie on this ray, so the image of $\mathbf{X}$ in the second view must lie on $\mathbf{l}'$.

if the image location are noise free. Since this is not the case, in order to estimate how likely a candidate correspondence is to be correct we have to measure the distance between the point and the line $\mathbf{l}'$.

At this point, one way to proceed could be to evaluate each candidate correspondence, that is between pair of views, and then pick the most likely one. This solution is however not viable for two reasons. First of all, there can be other points (wrong ones) that are closer to the line then the correct one resulting in wrong assignments. Then, this method does not take into consideration that if a set of image points (one for each view) are the projection of the same object, there must be one correspondence between all possible pairs.

We will show in the next paragraph how this problem can be formulated efficiently as Integer Linear Programming (ILP) problem. The constraints will allow us to form correct sets of correspondences while finding the optimal solution that minimize the distances between the epilines and the image locations.

3.2 Optimal method

We use here the same problem formulation we saw in the previous section, with the only difference that we now define one extra detection for each view $\{\mathbf{x_d^i}\}_{i=1}^{J}$ that we name "dummy detection". This will allow us to formulate this problem as ILP. The set of correspondences for one object can now be expressed more easily as $\{(i^j, i^k)|j,k \in J, j \neq k\}$ where i^j can now be also the index of the dummy detection. The difference with the previous formulation is that we now have one correspondence for all possible pairs of views. This was not the case before due to missing detections.

If we assume now that each detection is a node in a graph where edges represent possible correspondences, the set of correspondences associated to one physical object can be defined as one clique $C = \{i^j\}_{i=1}^{J}$ of size $|J|$ in this graph. If we define that only one image location can correspond to one object in one view, there cannot be two cliques sharing the same detection. Examples of such disjoint cliques are shown on the right side of Fig. 2.

In order to find a solution, which is a set of cliques in this graph that obey to the constraints we just defined, we have to define the cost of a clique. The edges encode the two-way distance (d_1, d_2) between the epilines and the image locations. To translate this distance to a weight we simply use $w = \exp((\frac{d_1+d_2}{2})^2/\sigma^2)$ where sigma is a parameter of this algorithm. If one or both detections are dummy ones, we assign a fixed weight $w_d = const$. We then define the cost of a clique as the sum of all edge weights composing the clique $W = \sum^E w_e$ where E is the set of all edges composing it.

The problem of finding the correspondences has now become finding a set of cliques that minimize the overall distance/weight and that satisfy the constraints we defined. This can be formulate an solved efficiently using integer

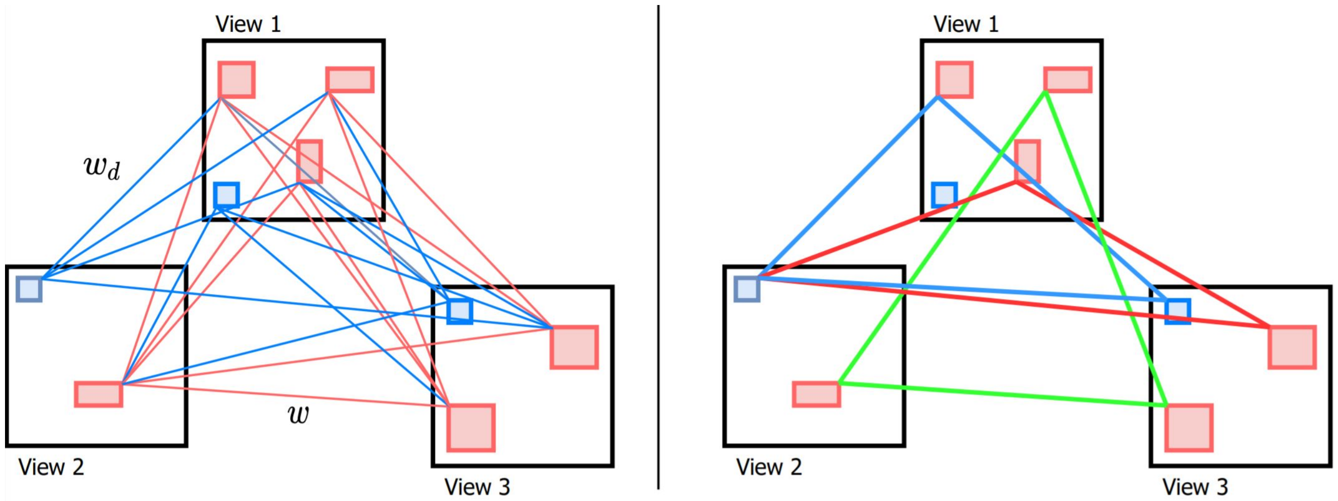

Figure 2: (Left) Graph defining all possible correspondences between detections in different views. Detections are represented by red rectangles. The blue rectangles are dummy detections that serve the purpose of closing the cliques/loops. In this example cliques are triangles. For four views, cliques would have six edges. The red edges represent possible correspondences between detections and the weight associated to them is the likelihood, which is computed using epipolar geometry. The edges in blue connect the dummy nodes to the other detections. These have a fixed weight. (Right) The solution, that is N cliques, that minimizes the cost function.

linear programming as follow:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{M} W_i x_i \\ \text{subject to} & \{x_i + x_j \leq 1 \mid C'_i \cap C'_j \neq \emptyset\} \end{array}$$

where x_i is a binary variable associated to a clique C_i and M is the set of all possible cliques in the graph which can be found prior to executing ILP. With a slightly abused notation, if C' is a clique in which we excluded the dummy nodes $C' = C \backslash \{i_d\}$ then $C'_i \cap C'_j \neq \emptyset$ is the condition defining when two cliques share one or more detections. Therefore, $x_i + x_j \leq 1$ is the constraint we used to impose that only one clique among two with shared nodes can be active at a time.

The solution to the ILP problem is a set of cliques defining the correspondences of image locations for one object. The cliques composed of many dummy nodes are in general discarded. The position of the object in the scene can then be readily computed using triangulation.

4 Conclusion

With this document we explained one way to find sets of correspondences between detections across views that represent objects in the scene. With them, it is possible to estimate the position of the objects through triangulation.

Future Work. Even though finding the solution to this ILP formulation is fast, we noticed that finding all possible candidate cliques, which are computed prior to executing the solver, is the bottleneck of this algorithm. The reason is that as the number of objects increases, the number of possible cliques in the graph increases exponentially making this search costly. To have an idea, it could take around 60 milliseconds to find the solution for a scene with 6 objects and 7 views. This is still reasonable but it can be improved.
One way to overcome this is to include the search of the cliques in the ILP formulation. To do so we have to define the ILP variable x as to be one edge in the graph, then define a set of constraints in addition to the one we have so far that force the solver to find cliques of size J. [1] gives an example of the constraints needed to define a clique.

References

[1] Afshin Dehghan, Shayan Modiri Assari, and Mubarak Shah. Gmmcp tracker: Globally optimal generalized maximum multi clique problem for multiple object tracking.