


FACULTY OF MATHEMATICS,
PHYSICS AND INFORMATICS
Comenius University
Bratislava

3D Vision

Lecture 5: Structure Computation and Point Correspondences

Ing. Viktor Kocur, PhD.

21.3.2023

Contents

Structure Computation

Consider a set of correspondences $\boldsymbol{x}_i \leftrightarrow \boldsymbol{x}_i'$. Our goal is to find both camera matrices P and P' as well as the original points $\boldsymbol{X}_i$ in the real scene such that for all i:

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

To obtain the camera matrices we can use the fundamental/essential matrix estimation. Since the relative positions of the cameras are known we can perform triangulation in the real space to find the position of $\boldsymbol{X}$.

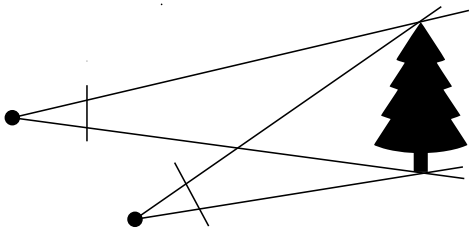
Ambiguities

There are some limitations as to what what information can we obtain from reconstruction. It is clear that we have a choice of the coordinate system, so that any reconstruction is defined only up to an Euclidean transformation.

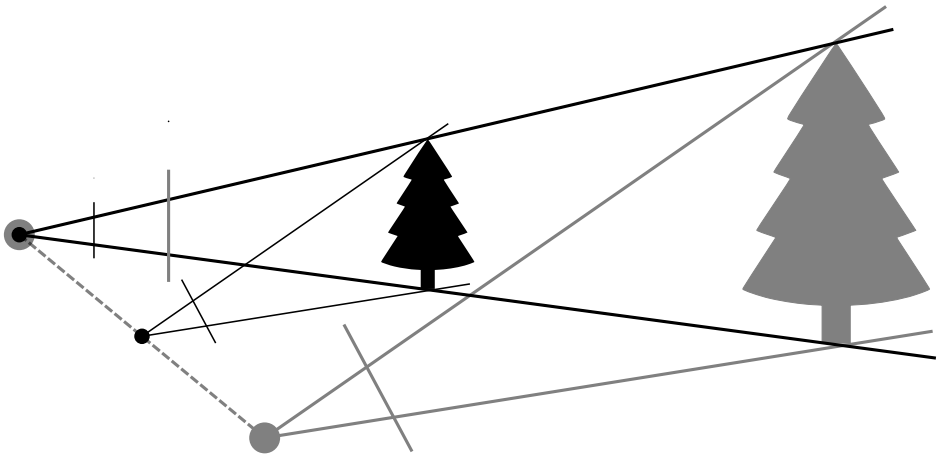
We have also shown in the last lecture, that the scene can be only reconstructed up to a scale. This means that any scene can also be defined only up to a similarity transform.

These ambiguities can be resolved if we have additional information. For example we can determine the latitude and longitude of $\boldsymbol{X}_i$ in the reconstruction if we have enough pairs of image points to such coordinates. Similarly if an object of known dimensions is present in the scene we can then measure distances in the scene in real-world units.

Scale Ambiguity



Scale Ambiguity



Other Ambiguities

When we are dealing with uncalibrated cameras (unknown K, K') and we do not know any prior information about their relative placement $(R, \boldsymbol{t})$ then the reconstructed scene is ambiguous also w.r.t. a projective transformation. This means that we could perform transformation $\boldsymbol{X}_i \mapsto \hat{\boldsymbol{X}}_i$ in the form

$$\hat{\boldsymbol{X}}_i \sim H\boldsymbol{X}_i, \tag{2}$$

where H is 4×4 regular matrix, also a homography, but this time in 3-space. This type of reconstruction is called **projective reconstruction**

Projective Ambiguity

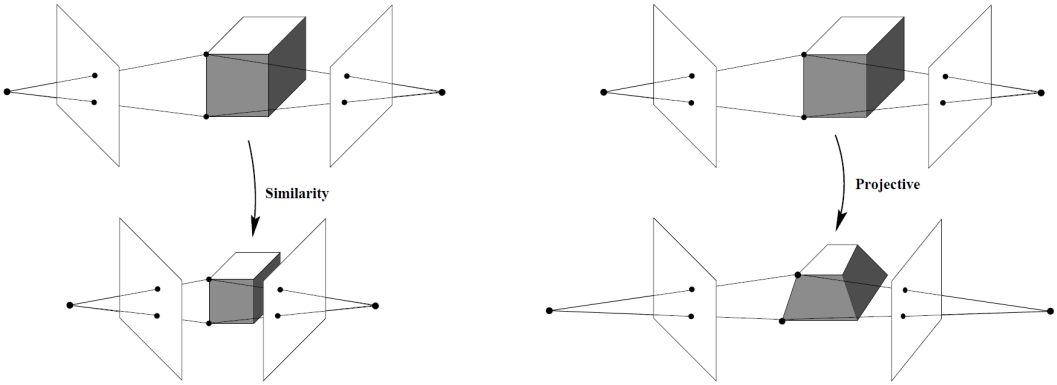


Image adopted from: Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003

Dealing with Projective Ambiguity

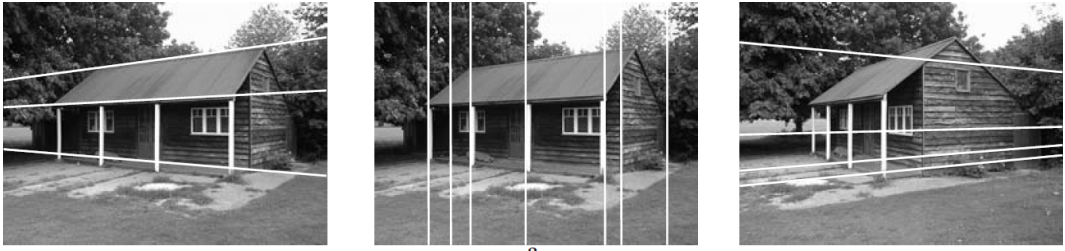
We can deal with the projective ambiguity using additional information about the scene or the cameras. When considering cameras there are two situations:

- If the cameras only undergo pure translation without change of calibration then the scene can be reconstructed up to affine transformation - this is also known as **affine reconstruction**.
- If only the focal lengths of the cameras are unknown then the scene can be reconstructed up to a similarity transformation (scale) - this is also known as **metric reconstruction**.

Projective Reconstruction

Image adopted from: Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003

Using Parallel Lines



We can reduce projective reconstruction to affine reconstruction by knowing that some lines in the scene are parallel in the real world.

Image adopted from: Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003

Constraining K, K'

We can reduce projective and affine reconstructions to metric ones using constraints on the internal parameters of the cameras. For instance we could use knowledge of orthogonal vanishing points/horizon. We can also make assumptions about the cameras such that $f_x = f_y$, skew is zero or principal point is in the center of the image. We could also utilize an assumption that the intrinsics do not change between view.

A combination of these assumptions may enable us to determine K and K' from the fundamental matrix and thus reduce the problem to the calibrated case leading to metric reconstruction.

Obtaining $\boldsymbol{X}$ - Triangulation

Considering the calibrated case we know P and P' as well as the correspondences $\boldsymbol{x}_i \leftrightarrow \boldsymbol{x}'_i$. Our task is then to find $\boldsymbol{X}_i$ for each correspondence. We can achieve this using triangulation. In an ideal scenario where $\boldsymbol{x}_i'^T F \boldsymbol{x}_i = 0$ we could construct use a linear system in $\boldsymbol{X}$ from equations

$$\boldsymbol{x}_i \sim P\boldsymbol{X}_i \quad \boldsymbol{x}'_i \sim P'\boldsymbol{X}_i. \tag{3}$$

The problem is that in reality, due to noise and other factors $\boldsymbol{x}_i'^T F \boldsymbol{x}_i \neq 0$. We will still use (3), but we will not solve the linear system exactly.

Triangulation

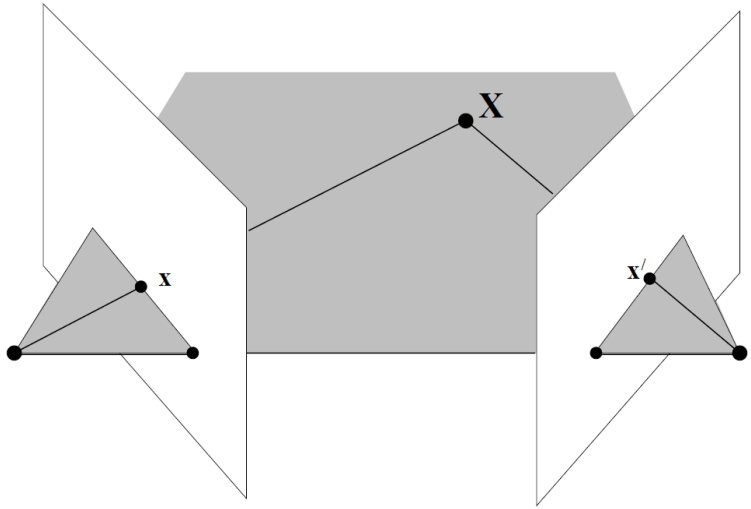


Image adopted from: Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003

Linear System

We can obtain the linear system by considering equations $\boldsymbol{x} \times P\boldsymbol{X} = \boldsymbol{0}$ to form:

$$A\boldsymbol{X} = \begin{pmatrix} x\boldsymbol{p}^T_{(3,:)} - \boldsymbol{p}^T_{(1,:)} \\ y\boldsymbol{p}^T_{(3,:)} - \boldsymbol{p}^T_{(2,:)} \\ x\boldsymbol{p'}^T_{(3,:)} - \boldsymbol{p'}^T_{(1,:)} \\ y'\boldsymbol{p'}^T_{(3,:)} - \boldsymbol{p'}^T_{(2,:)} \end{pmatrix} \boldsymbol{X} = \boldsymbol{0}, \tag{4}$$

where $\boldsymbol{x} = (x, y, 1)^T, \boldsymbol{x'} = (x', y', 1)^T, \boldsymbol{p}_{(i,:)}$ is the i-th row of P and analogously for P'. Note that the crossproduct yields three equations, but only two are linearly independent hence A has 4 rows. Note that the system is overdetermined as we are looking for a homogenous 4-vector. We can therefore use DLT and obtain the solution as the eigenvector corresponding to the smallest eigenvalue using SVD.

Inhomogeneous Solution

We can also solve the system in another way by assuming that $\boldsymbol{X} = (X, Y, Z, 1)$. This will transform the system $A\boldsymbol{X} = \boldsymbol{0}$ into

$$M \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \boldsymbol{b}. \tag{5}$$

The system will be overdetermined as we will get 4 equations in 3 unknown. We can solve this system in least-squares manner again using SVD by calculating the pseudoinverse and finding $M^{\dagger}\boldsymbol{b}$. Note that this method may get unstable with points close to infinity (far from the cameras).

Optimizing for Geometric Error

In both cases we optimize for algebraic error. As also mentioned in previous lectures this is not ideal. We could therefore find a better solution by minizing geometric error. For correspondence $\boldsymbol{x} \leftrightarrow \boldsymbol{x'}$ we will find $\hat{\boldsymbol{x}} \leftrightarrow \hat{\boldsymbol{x'}}$ such that $\hat{\boldsymbol{x'}}^T F \hat{\boldsymbol{x}} = 0$. We therefore have optimization problem:

$$\underset{\hat{\boldsymbol{x'}}^T F \hat{\boldsymbol{x}} = 0}{\arg\min} \left(d(\boldsymbol{x}, \hat{\boldsymbol{x}})^2 + d(\boldsymbol{x'}, \hat{\boldsymbol{x'}})^2 \right), \tag{6}$$

where d is a metric. Note that this optimization can be solved using a direct method, but we will not cover this in lecture. In practice we will use methods based on reducing algebraic error as initialization and perform further optimization later.

Optimizing for Geometric Error

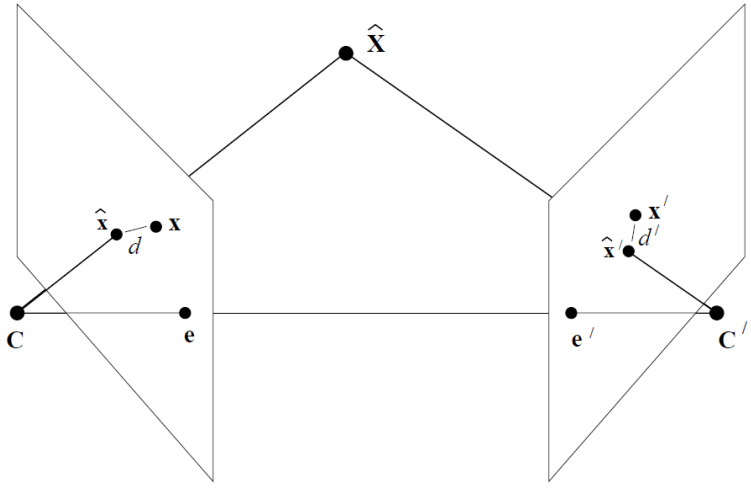


Image adopted from: Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003

Point Correspondences

So far we have assumed a knowledge of some point correspondences $\boldsymbol{x} \leftrightarrow \boldsymbol{x}'$. These could be obtained manually, but a much more practical scenario is obtaining them automatically using computer vision techniques. We will now cover how that could be achieved using a representative classical method and a deep learning method.

We could also use some other form of correspondence such as correspondence between two lines or affine correspondences which also include information about the the transformation of a local neighborhood of the point. These are more advanced topics and will not be covered in lecture.

Keypoint Detection and Matching

Finding correspondences can be split into two distinct tasks:

- Keypoint detection - Task of finding interesting points in each image. These points should be found such that it is possible to accurately match them.
- Keypoint matching - This is the task of pairing two sets of keypoints found in two images.

SIFT

Scale Invariant Feature Transform is a classical method for detection of local features (keypoints). The method first finds keypoints and then assigns them a descriptor vector which is later used for matching.

Keypoints are detecting using following procedure:

1. Find local minima/maxima in difference of gaussian pyramid of the image - the keypoint has a position and also a scale from the pyramid at which it was found
2. Refine keypoint to subpixel position using Taylor expansion
3. Eliminate low-contrast points
4. Eliminate keypoints on edges

SIFT - descriptor

In the next step we create a 128 dimensional descriptor for each point. This is performed in two steps:

1. Determine the dominant orientation in the neighborhood of the keypoint using local gradients
2. Obtain histograms of gradients in local neighborhoods around the keypoint. These are calculated for the given scale and orientation of the detected keypoint. The values in histograms are then concatenated into a single descriptor vector.

The descriptors should be invariant w.r.t. rotation, scale and to some extent also illumination.

SIFT

Image adopted from: *OpenCV Documentation*.

Matching

We can match keypoints from two different images using a metric on the 128-dimensional feature space. We could perform greedy assignment, but this may not be optimal. Therefore we find the 2 nearest neighbors for each keypoint in one image. If the closest neighbor's distance is less than 75% of the distance to the second neighbor we accept the match other.

Other Local Feature Methods

SIFT was the preferred state-of-the-art method for many years, but it was patented. The patent has now expired so you could use it commercially for free.

There are also other similar methods such as ORB, KAZE, SURF etc. Similar methods also exist to match other types of objects such as blobs using MSER.

Keypoints - SuperPoint

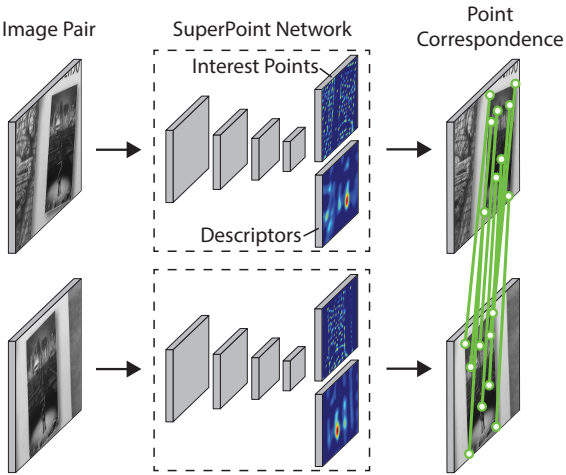


Image adopted from: Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. “Superpoint: Self-supervised interest point detection and description.” In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2018, pp. 224–236

SuperPoint - Training

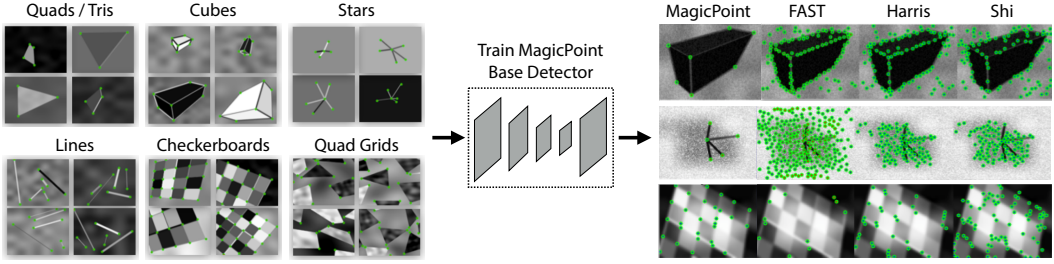


Image adopted from: Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. “Superpoint: Self-supervised interest point detection and description.” In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2018, pp. 224–236

SuperPoint - Training

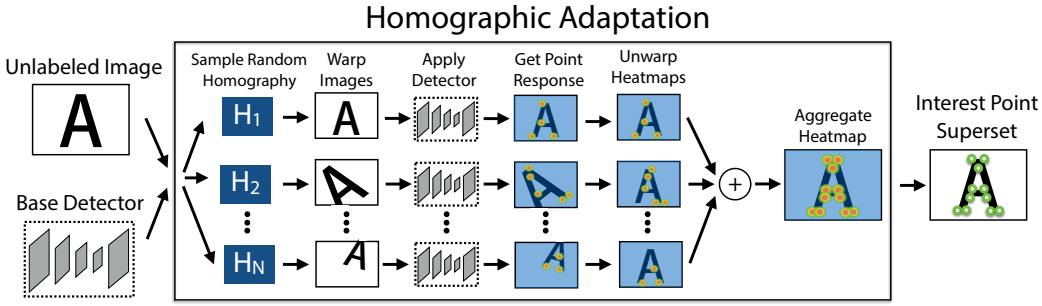


Image adopted from: Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. “Superpoint: Self-supervised interest point detection and description.” In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2018, pp. 224–236

SuperPoint Comparison

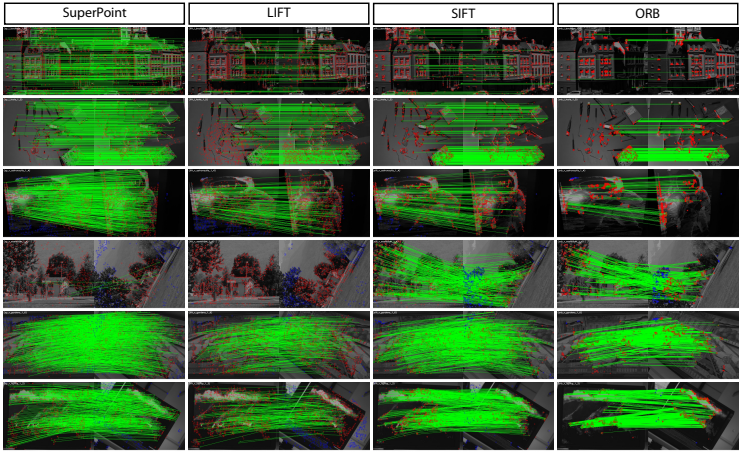


Image adopted from: Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. “Superpoint: Self-supervised interest point detection and description.” In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2018, pp. 224–236

Improved Matching - SuperGlue

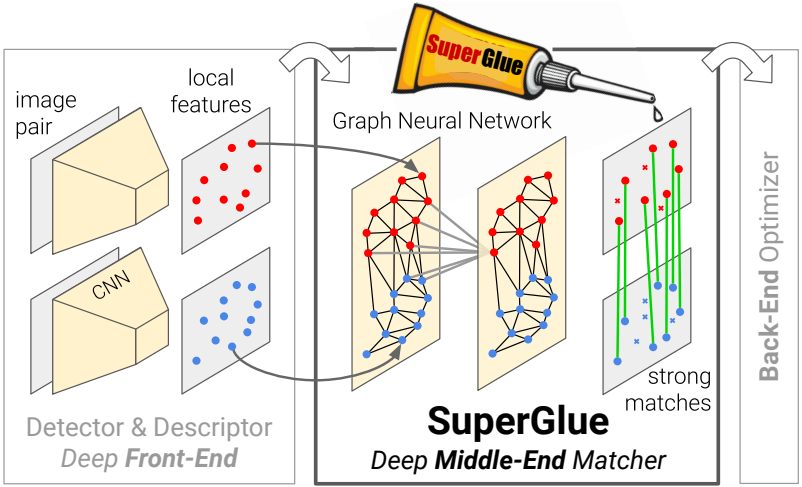


Image adopted from: Paul-Edouard Sarlin et al. “Superglue: Learning feature matching with graph neural networks.” In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 4938–4947

SuperGlue

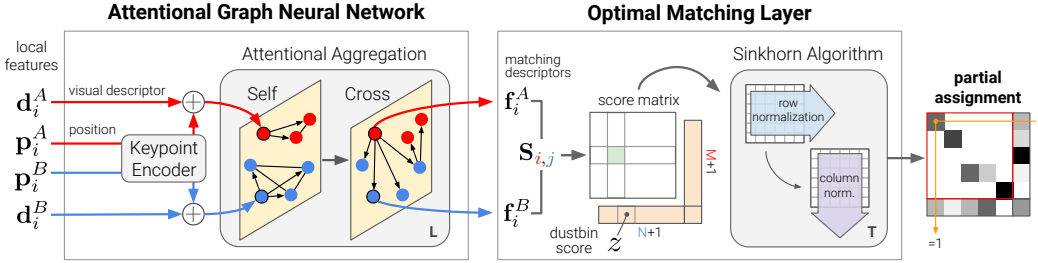
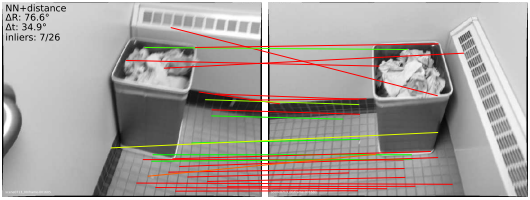
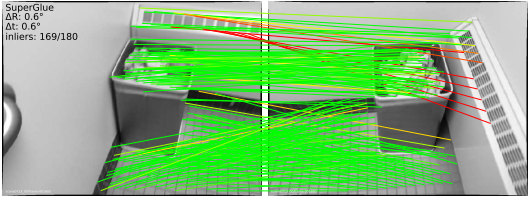


Image adopted from: Paul-Edouard Sarlin et al. “Superglue: Learning feature matching with graph neural networks.” In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 4938–4947

SuperGlue



SuperPoint + NN + Dist



SuperPoint + SuperGlue

Image adopted from: Paul-Edouard Sarlin et al. “Superglue: Learning feature matching with graph neural networks.” In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 4938–4947

Sparse Optical Flow

Image adopted from: *OpenCV Documentation*.

Dense Optical Flow

Image adopted from: *OpenCV Documentation*.

Stereo Rigs

If we consider stereo rigs we can make the problem of detecting correspondences simpler. Let us consider a stereo rig with $P = K[I|\mathbf{0}], P' = K[I|\boldsymbol{t}]$, where $s = 0, \boldsymbol{t} = (1, 0, 0)^T$. In this case we could calculate the epipole $\boldsymbol{e'}$ as the vanishing point corresponding to the direction $\boldsymbol{t}$:

$$\boldsymbol{e'} = K\boldsymbol{t} \sim \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}. \tag{7}$$

This means that the epipole is at infinity. We can still obtain epipolar lines for all points in either image. These lines will be horizontal in both images.

Correspondences

Horizontal epipolar lines mean that for every point $\boldsymbol{x} = (x, y, 1)^T$ its correspondence has to be $\boldsymbol{x'} = (x', y, 1)^T$. This means that for each point we only have one degree of freedom to find the correspondence. Additionally, the ordering of points on a horizontal line from left to right has to be consistent with the ordering on the corresponding line in the second view.

We can leverage this property to find use correlation to find a disparity map. The disparity map $d(x, y)$ can be defined as:

$$x + d(x, y) = x', \tag{8}$$

for a given row y.

Disparity Map

The disparity map can be obtained by a variety of ways. A common approach is to calculate the normalized cross-correlation of segments of rows in one image with segments in the other and finding the maxima. The maxima will be located at the point which determines the distance $d(x, y)$. We can then obtain the depth information using:

$$z(x, y) = \frac{d_b f}{d(x, y)}, \tag{9}$$

where f is the focal length, d_b is the baseline distance.

Other Approaches

We can also obtain depth information using methods other than stereovision:

- LiDAR/Time of Flight - based on the time it takes light to reflect from surface
- Structured Light - projection of light patterns on objects

Different imaging systems can be combined in a structure reconstruction pipeline.

IR Projection

Image adopted from: *Intel RealSense Documentation*.

Structured Light

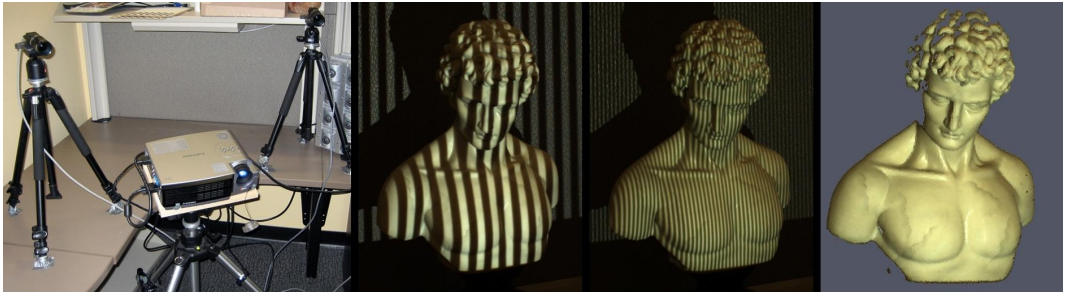


Image adopted from: Douglas Lanman and Gabriel Taubin. URL: `http://mesh.brown.edu/byo3d/source.html`