


FACULTY OF MATHEMATICS,
PHYSICS AND INFORMATICS
Comenius University
Bratislava

3D Vision

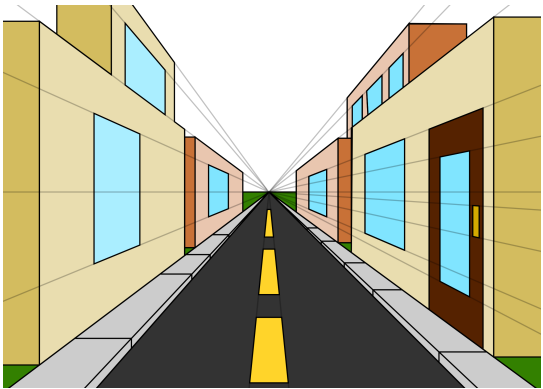
Lecture 3: Vanishing Points and Lines, Homography Estimation

Ing. Viktor Kocur, PhD.

7.3.2023

Contents

Vanishing Points



Lines which are parallel in the real scene are imaged such that they all coincide into a single point called the **vanishing point** (*úbežník*).

Image adopted from: *Wikipedia*.

Vanishing Points

Let us consider a line in the world coordinates defined by a point $\boldsymbol{A}$ in homogeneous coordinates and a direction $\boldsymbol{D} = (\boldsymbol{d}^T, 0)$ with a world coordinate system aligned with the camera (e.g. $R = I$ and $\boldsymbol{t} = \boldsymbol{0}$). Let us parametrize the points on the line as $\boldsymbol{X}(\lambda) \sim \boldsymbol{A} + \lambda \boldsymbol{D}$. Using the camera model we can derive the image of the point:

$$\boldsymbol{x}(\lambda) \sim K\,[I|\boldsymbol{0}]\,\boldsymbol{X}(\lambda) \sim \boldsymbol{a} + \lambda K \boldsymbol{d}, \tag{1}$$

where $\boldsymbol{a}$ is the image of $\boldsymbol{A}$. Then the vanishing point $\boldsymbol{v}$ is obtained as the limit:

$$\boldsymbol{v} \sim \lim_{\lambda \to +\infty} \boldsymbol{x}(\lambda) \sim \lim_{\lambda \to +\infty} \boldsymbol{a} + \lambda K \boldsymbol{d} \sim K \boldsymbol{d}. \tag{2}$$

Note that we can also use (2) to calculate $\boldsymbol{d} \sim K^{-1}\boldsymbol{v}$.

Vanishing Points

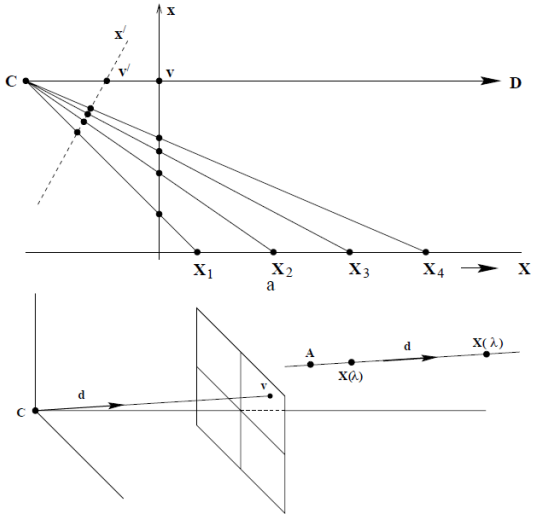


Image adopted from: Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003

Vanishing Points - Angles

Given two vanishing points $\boldsymbol{u}$ and $\boldsymbol{v}$ imaged the same camera we can calculate the angle θ between them:

$$\cos\theta = \frac{\boldsymbol{u}^T(KK^T)^{-1}\boldsymbol{v}}{\sqrt{\boldsymbol{u}^T(KK^T)^{-1}\boldsymbol{u}}\sqrt{\boldsymbol{u}^T(KK^T)^{-1}\boldsymbol{v}}}. \tag{3}$$

Note that this is especially useful when the vanishing points correspond to orthogonal directions. Then we can reduce (3) to:

$$0 = \boldsymbol{u}^T(KK^T)^{-1}\boldsymbol{v}. \tag{4}$$

This is useful as it places one constraint on K which can be used to find the intrinsic camera parameters which is also known as **camera calibration**.

Vanishing Lines

When considering a plane in the real world we may investigate the images of all of the directions parallel to the plane. The vanishing points for these directions form a line in the image. This line is called the **vanishing line** or sometimes also the **horizon** of the plane. Note that all real-world planes which are parallel to one another share the same vanishing line.
When the intrinsic parameters of the camera are known it is possible to use the vanishing line $\boldsymbol{h}$ of a plane to calculate the plane normal $\boldsymbol{n}$ in the real-world camera coordinate system as

$$\boldsymbol{n} = K^T \boldsymbol{h}. \tag{5}$$

Vanishing Lines

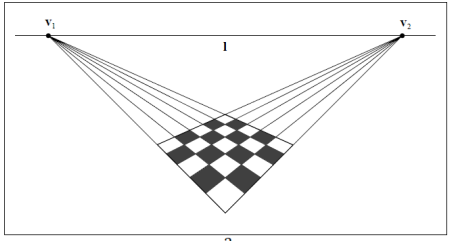
We can also calculate the angle θ between two planes with vanishing lines $\boldsymbol{k}$ and $\boldsymbol{h}$:

$$\cos\theta = \frac{\boldsymbol{k}^T(KK^T)\boldsymbol{h}}{\sqrt{\boldsymbol{k}^T(KK^T)\boldsymbol{k}}\sqrt{\boldsymbol{h}^T(K^TK)\boldsymbol{h}}}. \tag{6}$$

A vanishing point $\boldsymbol{v}$ corresponding to the direction of the normal of a plane which has the vanishing line $\boldsymbol{h}$ impose two constraints on the intrinsic matrix via the equation

$$\boldsymbol{h} = \left((KK^T)^{-1}\boldsymbol{v}\right). \tag{7}$$

Vanishing Lines



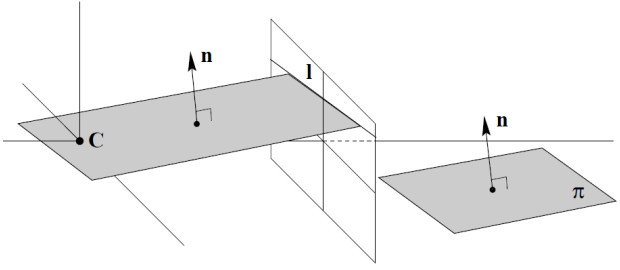


Image adopted from: Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003

Example - Traffic Camera Calibration

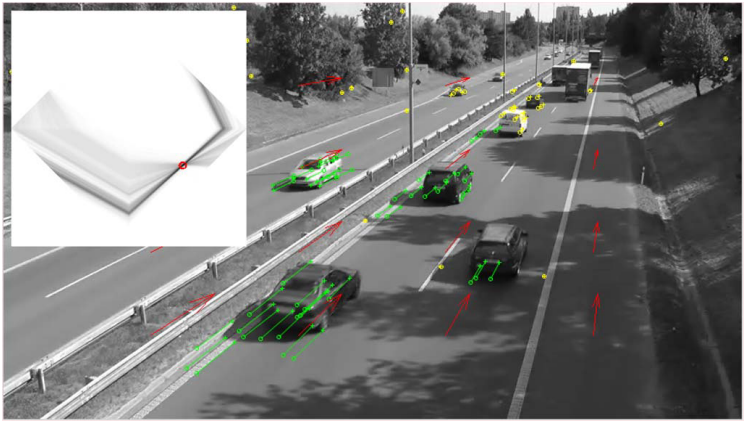


Image adopted from: Markéta Dubská et al. “Fully automatic roadside camera calibration for traffic surveillance.” In: *IEEE Transactions on Intelligent Transportation Systems* 16.3 (2014), pp. 1162–1171

Example - Traffic Camera Calibration

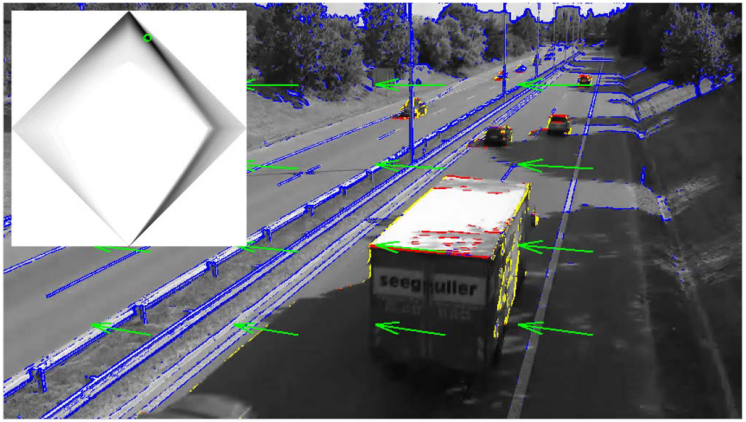


Image adopted from: Markéta Dubská et al. “Fully automatic roadside camera calibration for traffic surveillance.” In: *IEEE Transactions on Intelligent Transportation Systems* 16.3 (2014), pp. 1162–1171

Example - Traffic Camera Calibration

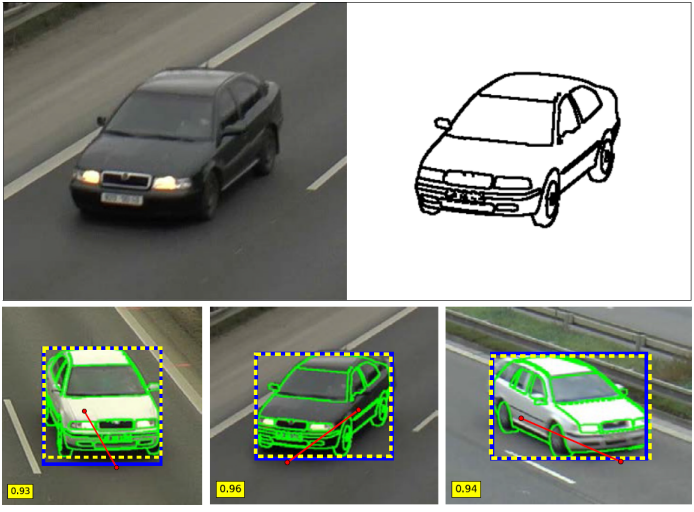


Image adopted from: Jakub Sochor, Roman Juránek, and Adam Herout. “Traffic surveillance camera calibration by 3d model bounding box alignment for accurate vehicle speed measurement.” In: *Computer Vision and Image Understanding* 161 (2017), pp. 87–98

Example - 3D Bounding Box Detection

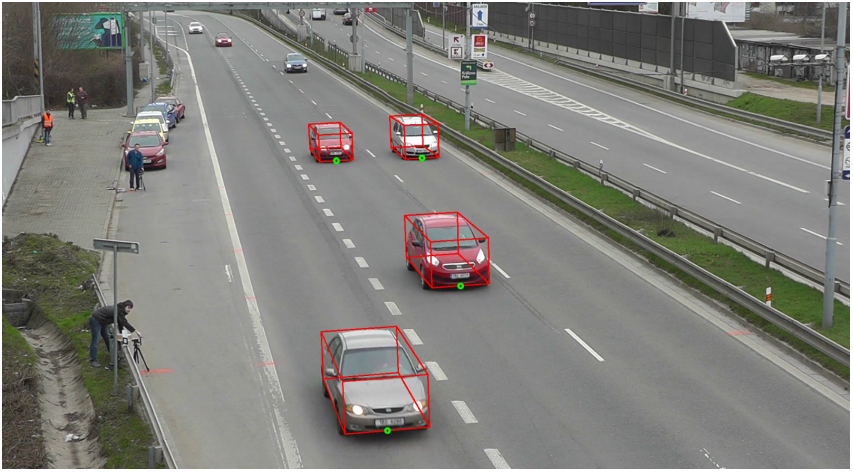


Image adopted from: Viktor Kocur and Milan Ftáčnik. “Detection of 3D bounding boxes of vehicles using perspective transformation for accurate speed measurement.” In: *Machine Vision and Applications* 31.7 (4), pp. 1–15

Homography Estimation

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

Homography Estimation

We may be interested in approximating a Homography given some point correspondences. Suppose that we have correspondences $\boldsymbol{x}_i' \leftrightarrow \boldsymbol{x}_i$. We want to use them to find a homography H such that:

$$\boldsymbol{x}_i' \sim H\boldsymbol{x}_i. \tag{8}$$

The matrix H has 8 degrees of freedom since it has 9 entries, but the scale invariance reduces it by one degree of freedom. Every equation (8) provides further reduction by 2 degrees of freedom. Therefore, barring degenerate configurations, we need 4 such correspondences to find H.

Dealing with Homogeneous Coordinates

Using (8) directly introduces a problem because of a lack of equality between the two vectors. To work with the correspondences we will have to rewrite (8) using the following property of the vector product:

$$\boldsymbol{x} \times \boldsymbol{y} = \mathbf{0} \iff (\exists \alpha \neq 0)(\boldsymbol{x} = \alpha \boldsymbol{y}) \vee \boldsymbol{x} = \mathbf{0} \vee \boldsymbol{y} = \mathbf{0}. \tag{9}$$

We can then transform (8) to the following form:

$$\boldsymbol{x}_i' \times H\boldsymbol{x}_i = \mathbf{0}. \tag{10}$$

Note that (10) can also be considered as three separate equations. However only two of them are linearly independent.

Rewriting H

To proceed further we will rewrite $H\boldsymbol{x}_i$ as:

$$H\boldsymbol{x}_i = \begin{pmatrix} \boldsymbol{h}_{(1,:)}^T \boldsymbol{x}_i \\ \boldsymbol{h}_{(2,:)}^T \boldsymbol{x}_i \\ \boldsymbol{h}_{(3,:)}^T \boldsymbol{x}_i \end{pmatrix}, \tag{11}$$

where $\boldsymbol{h}_{(j,:)}^T$ is j-th row of H. We can then rewrite (10) as:

$$\boldsymbol{x}_i' \times H\boldsymbol{x}_j = \begin{pmatrix} y_i' \boldsymbol{h}_{(3,:)}^T \boldsymbol{x}_i - w_i' \boldsymbol{h}_{(2,:)}^T \boldsymbol{x}_i \\ w_i' \boldsymbol{h}_{(1,:)}^T \boldsymbol{x}_i - x_i' \boldsymbol{h}_{(3,:)}^T \boldsymbol{x}_i \\ x_i' \boldsymbol{h}_{(2,:)}^T \boldsymbol{x}_i - y_i' \boldsymbol{h}_{(1,:)}^T \boldsymbol{x}_i \end{pmatrix}, \tag{12}$$

where $\boldsymbol{x}_i' = (x_i', y_i', w_i')^T$.

Rewriting the system

We can continue with further modifications:

$$\begin{pmatrix} \mathbf{0}^T & -w_i'\boldsymbol{x}_i^T & -y_i'\boldsymbol{x}_i^T \\ w_i'\boldsymbol{x}_i^T & \mathbf{0}^T & -x_i'\boldsymbol{x}_i^T \\ -y_i'\boldsymbol{x}_i^T & x_i'\boldsymbol{x}_i^T & \mathbf{0}^T \end{pmatrix} \begin{pmatrix} \boldsymbol{h}_{(1,:)} \\ \boldsymbol{h}_{(2,:)} \\ \boldsymbol{h}_{(3,:)} \end{pmatrix} = A_i\boldsymbol{h} = \mathbf{0}, \tag{13}$$

where A_i is a 3×9 matrix and $\boldsymbol{h}$ is a vector containing the elements of H such that:

$$H = \begin{pmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{pmatrix}. \tag{14}$$

Leaving out Linearly Dependent Rows

Note that A_i is of rank 2 at best since in the original system of equations was linearly dependent. We can therefore remove one row from the matrix:

$$\begin{pmatrix} \mathbf{0}^T & -w_i'\boldsymbol{x}_i^T & -y_i'\boldsymbol{x}_i^T \\ w_i'\boldsymbol{x}_i^T & \mathbf{0}^T & -x_i'\boldsymbol{x}_i^T \end{pmatrix} \begin{pmatrix} \boldsymbol{h}_{(1,:)} \\ \boldsymbol{h}_{(2,:)} \\ \boldsymbol{h}_{(3,:)} \end{pmatrix} = A_i\boldsymbol{h} = \mathbf{0}, \tag{15}$$

and A_i would thus become a 2×9 matrix. Note that in both cases we obtain a system linear in $\boldsymbol{h}$.

Solution for 4 Correspondences

If we consider 4 point correspondences. We can stack the equations to obtain the system:

$$\begin{pmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \end{pmatrix} \boldsymbol{h} = A\boldsymbol{h} = \mathbf{0}. \tag{16}$$

Note that the matrix A is either 8×9 or 12×9, but in any way its rank is at most 8. We are not interested in the trivial solution where $\boldsymbol{h} = \mathbf{0}$. We can therefore choose $\boldsymbol{h}$ to be any other vector from $\ker(A)$ - the null space of A. We can find such vector with $|\boldsymbol{h}| = 1$ using SVD. The algorithm we have shown is called **direct linear transformation**.

More than 4 Correspondences

We can also consider more than 4 correspondences by stacking matrices. If we get exact coordinates without noise then the resulting matrix A would still have rank at most 8. However this never happens in practice due to noise, sampling of points and numerical accuracy. The resulting matrix would thus be of rank 9.

We will therefore want to find a solution under some constraint such as $|\boldsymbol{h}| = 1$ while minimizing the error $|A\boldsymbol{h}|$.

Yet Another Use of SVD

Consider a $n \times m$ matrix M with $\text{rank}(M) > m$ such that its SVD is $M = USV^T$. Then we can find:

$$\hat{\boldsymbol{x}} = \underset{|\boldsymbol{x}|=1}{\arg\min} |M\boldsymbol{x}| \tag{17}$$

as the column of V which corresponds to the smallest singular value of M. Note that we can use this to find $\boldsymbol{h}$ from the previous slide.

Degenerate Configurations

It is possible to encounter some degenerate configurations of 4 point correspondences. This may occur if three of the points are colinear (e.g. they lie on a single line) in one of the views. In such case there are two possibilities:

- The corresponding points in the other view are also colinear. In such case the rank of A will be lower than 8 and thus its null-space will have dimension greater than 1. This means that we can find different homographies which will satisfy the relationship of the given correspondences.
- The corresponding points in the other view are not colinear. In such case we will find a single matrix H, but it will not be regular and thus not a valid homography.

Note that very often configurations which are close to the degenerate ones may lend themselves to lead to unstable numerical results. It is therefore advisable to keep such configurations in mind.

Homography from Lines

Instead of point correspondences we could also use line correspondences. For every line correspondence $\boldsymbol{k}_i' \leftrightarrow \boldsymbol{k}_i$ we get the following:

$$\boldsymbol{k}_i \sim H\boldsymbol{k}_i'. \tag{18}$$

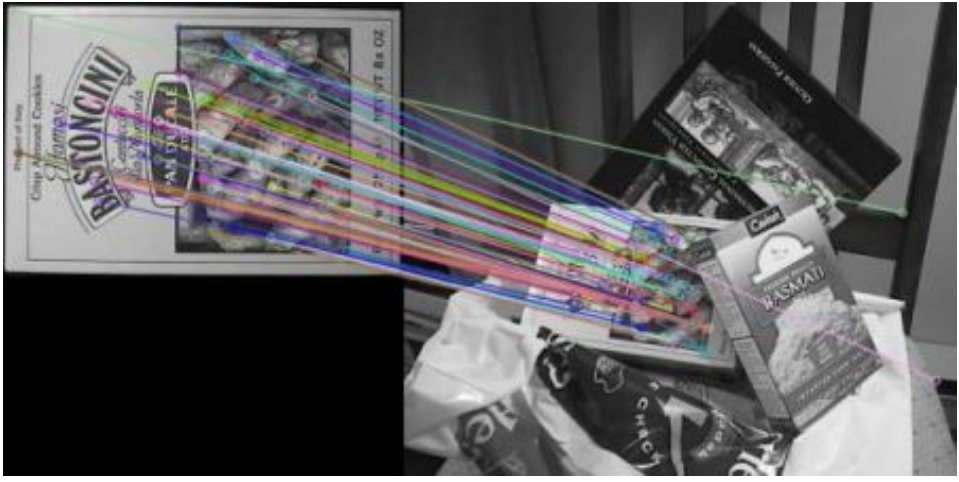
We can use this to create a similar system in the form of $A_i\boldsymbol{h} = \boldsymbol{0}$. Using a line introduces 2 linearly independent equations so we can use 4 line correspondences to obtain H. We can also use a combination of 3 points and a line or 3 lines and a point. Using 2 points and 2 lines leads to a degenerate configuration.

Estimating Homography in Practice

If we want to implement robust homography estimation we will proceed with following steps:

- Find correspondences
- Use RANSAC to find correctly identified correspondences
- Use the inliers and initial estimate to perform non-linear optimization of H

Correspondences



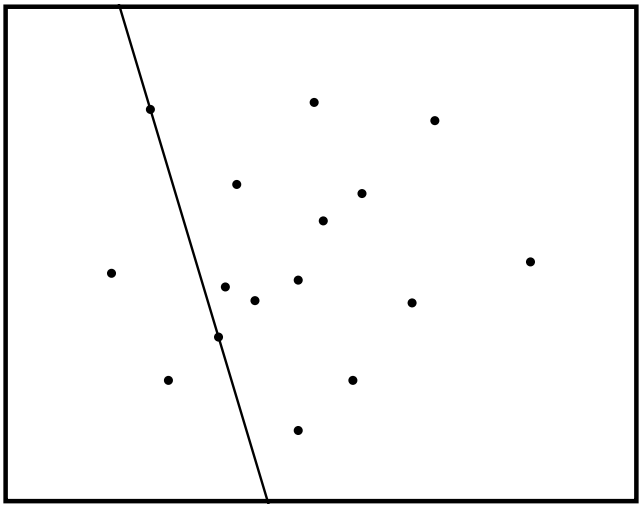
We may use various methods to obtain correspondences, but it is very common for automatic methods to find incorrect correspondences between images!

Dealing with Bad Correspondences

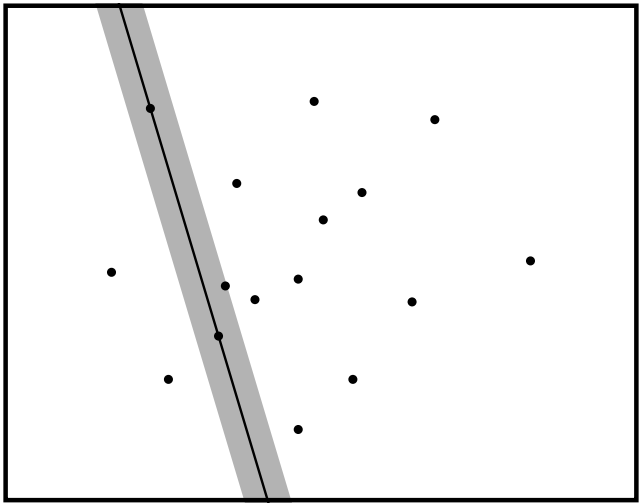
We will almost always have some bad (outlier) correspondences. Fitting a homography on all correspondences (including bad ones) will result in bad results! We will try to find the good (inlier) correspondences using **random sample consensus algorithm** (RANSAC). The algorithm has a following structure.

1. Select 4 correspondences at random.
2. Calculate H using the four correspondences.
3. Calculate how many of the remaining correspondences have low errors w.r.t. matrix H.
4. Perform steps 1-3 multiple times and then pick H with the most inliers.

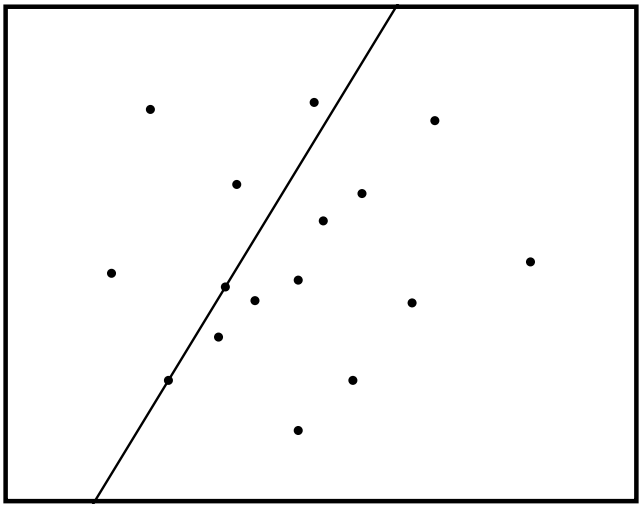
Ransac Example - Line



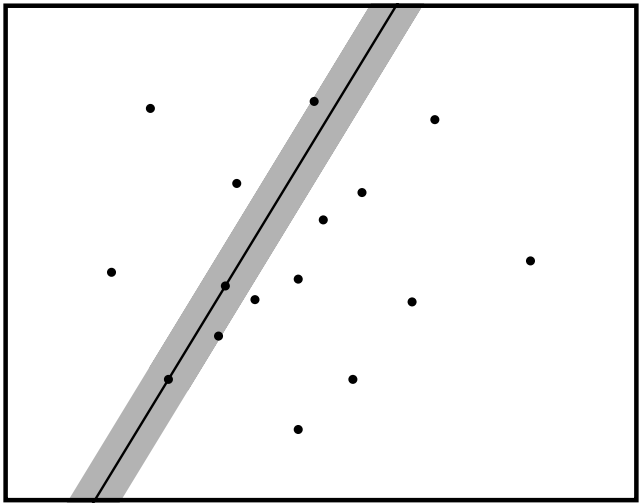
Ransac Example - Line



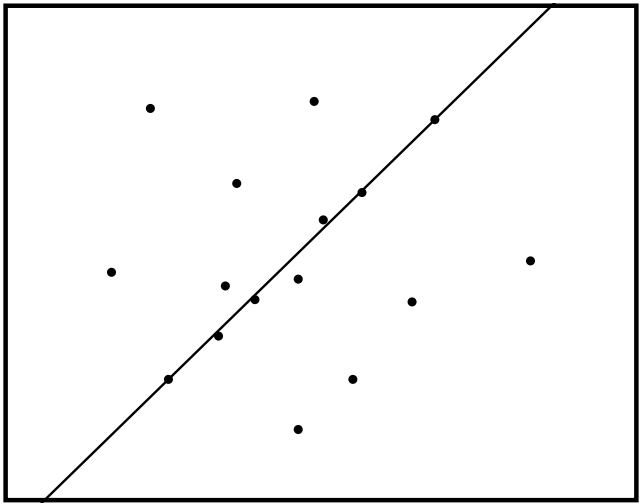
Ransac Example - Line



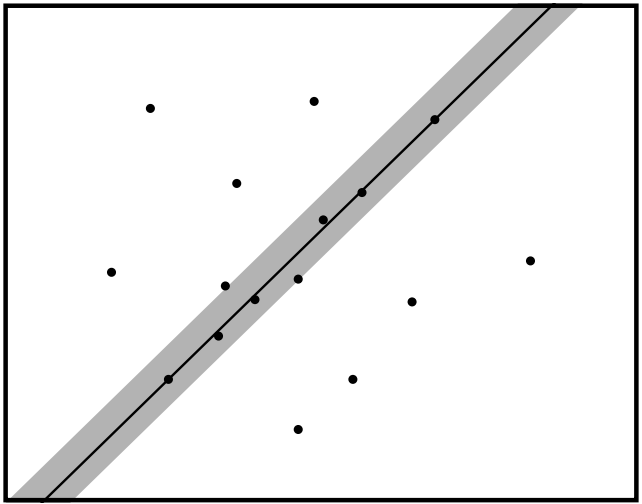
Ransac Example - Line



Ransac Example - Line



Ransac Example - Line



Non-linear optimization

Using RANSAC we have an initial estimate of H and also a set of correspondences $\boldsymbol{x}_i' \leftrightarrow \boldsymbol{x}$ that we consider inliers. To obtain the final homography we may want to find the homography H such that it minimizes:

$$\sum_{i=1}^{N} d(\boldsymbol{x}_i', H\boldsymbol{x}) + d(H^{-1}\boldsymbol{x}_i', \boldsymbol{x}_i), \tag{19}$$

where d is standard Euclidean distance in image coordinates. This can be achieved using an iterative method which starts from our original estimate of H and refines it in a gradient-based optimization procedure. Note that we may also consider using DLT for all of the correspondences, but that minimizes $|Ah|$ which may not correspond to a geometrically optimal homography.