


FACULTY OF MATHEMATICS,
PHYSICS AND INFORMATICS
Comenius University
Bratislava

3D Vision

Lecture 7: Dense Reconstruction, Advanced RANSAC

Ing. Viktor Kocur, PhD.

4.4.2023

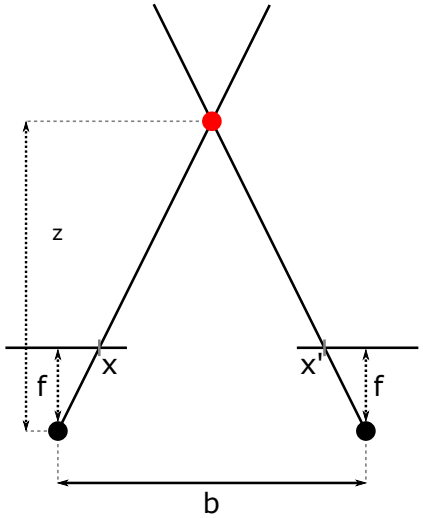
Contents

Depth in Aligned Stereo

We can calculate the depth

$$Z(x, x') = \frac{bf}{|x - x'|}, \qquad (1)$$

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



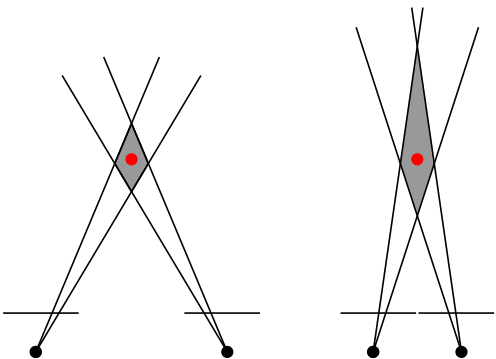
Small vs. Large Baseline

Large baseline:

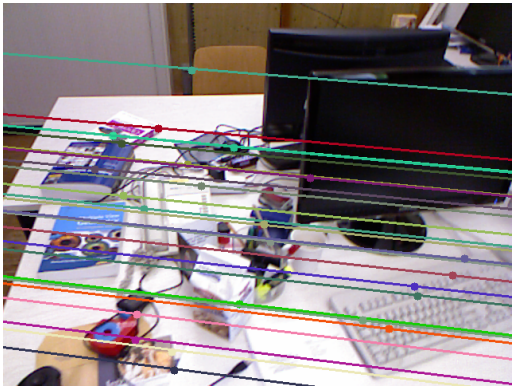
- Small depth error
- Minimum measurable depth increases
- Difficult search problem for close objects

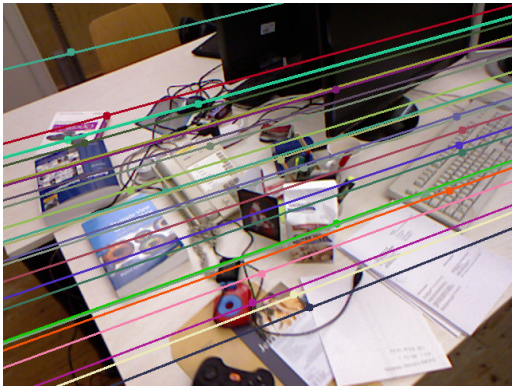
Small baseline:

- Large depth error
- Minimum measurable depth decreases
- Easier search problem for close objects

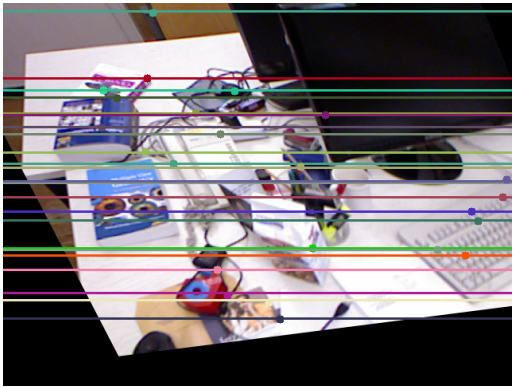


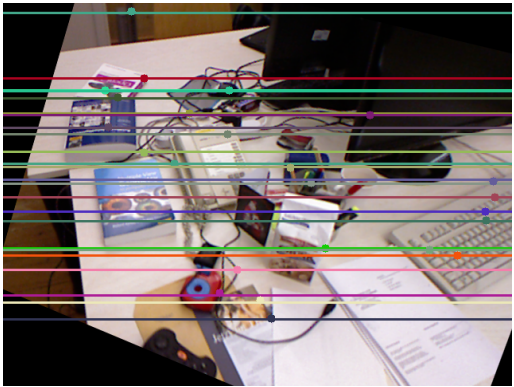
Stereo Rectification





Stereo Rectification





Stereo Rectification

We want to rotate the views in order to obtain the desired images where the epipolar lines are horizontal. Recall that we can do this with a homography to change view from K to $\hat{K}$ and rotation from R to $\hat{R}$:

$$\boldsymbol{x}' \sim P'\boldsymbol{X} \sim (\hat{K}\hat{R})(KR)^{-1}P\boldsymbol{X} \sim (\hat{K}\hat{R})(KR)^{-1}\boldsymbol{x} = H\boldsymbol{x}. \tag{2}$$

We want to do find a homography such that the epipole will be at infinity after the transformation

$$\boldsymbol{e} \mapsto H\boldsymbol{e} \sim \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}. \tag{3}$$

How to choose $\hat{K}$

Do note that we have two views $P = K[I|\mathbf{0}]$ and $P' = K'[R'|\boldsymbol{t}]$. We want to perform transformation on both images such we retain the most information. First we need to determine the new intrinsic camera matrix. It will be the same for both views. We denote is as $\hat{K}$ and we can use a simple calculation to obtain it:

$$\hat{K} = \frac{K + K'}{2}. \tag{4}$$

If the matrices have $s \neq 0$ then we may also wish to set $s = 0$.

How to choose $\hat{R}$.

We also need to select the new rotations $\hat{R}$ and $\hat{R}'$. Note that we want both views to be oriented in the same way so $\hat{R} = \hat{R}'$. To derive the desired $\hat{R}$ let us first denote the camera centers $\boldsymbol{C}$ and $\boldsymbol{C'}$:

$$\boldsymbol{C'} = -R'^{T}\boldsymbol{t}, \tag{5}$$

this would be analogous for non-primed version with $\boldsymbol{C} = \boldsymbol{0}$. Our goal is now to find a rotation such that the baseline $\boldsymbol{C'} - \boldsymbol{C} = \boldsymbol{C'}$ is aligned with the x-axis of the image. This by itself is not enough to define the transformation. We therefore have some variability. Generally speaking we want to select the rotation such that it causes the smallest amount of perspective distortion.

Choosing $\hat{R}$

One popular choice can be defined using the columns of $\hat{R} = (\hat{\boldsymbol{r}}_{:,1}, \hat{\boldsymbol{r}}_{:,2}, \hat{\boldsymbol{r}}_{:,3})$. We then select:

$$\hat{\boldsymbol{r}}_{:,1} = \frac{\boldsymbol{C}' - \boldsymbol{C}}{|\boldsymbol{C}' - \boldsymbol{C}|} = \frac{\boldsymbol{C}'}{|\boldsymbol{C}''|}, \tag{6}$$

$$\hat{\boldsymbol{r}}_{:,2} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \times \hat{\boldsymbol{r}}_{:,1}, \tag{7}$$

$$\hat{\boldsymbol{r}}_{:,3} = \hat{\boldsymbol{r}}_{:,1} \times \hat{\boldsymbol{r}}_{:,2}. \tag{8}$$

Note that in (7) we select the third column of I. If we didn't consider $P = K[I|\mathbf{0}]$, but say $P = K[R'|\boldsymbol{t}']$ we would use the third column of R'.

Disparity map

After rectifying both images. We can calculate the depth by calculating the disparity map. Recall that we defined it as map $d(x, y)$ such that in row y we try to find

$$x + d(x, y) = x', \tag{9}$$

such that $(x, y)^T$ and $(x', y)^T$ are correspondences. If we succeed we can use

$$Z(x, y) = \frac{bf}{|x - x'|} = \frac{bf}{|d(x, y)|}. \tag{10}$$

Then we can obtain the 3D point $\boldsymbol{X} = (X, Y, Z)$ as:

$$\boldsymbol{X} = ZK^{-1} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}. \tag{11}$$

We can use the normalized cross correlation (NCC)

$$\mathrm{NCC}(i, i', x, y, d, n) = \frac{\sum_{x' \in N(x,n)} \left[i(x', y) - \mu_{x'' \in N(x,n)} \left(i(x'', y) \right) \right] \cdot \left[i'(x' + d, y) - \mu_{x'' \in N(x)} \left(i'(x'' + d, y) \right) \right]}{\sigma_{x' \in N(x,n)} \left(i(x', y) \right)^2 \sigma_{x' \in N(x,n)} \left(i'(x' + d, y) \right)^2}, \tag{12}$$

where $i(x, y)$ and $i'(x, y)$ are the intensity values at pixel coordinates x and y, μ is the mean σ is the standard deviation, and $N(x, n)$ is a discrete n-neighborhood of x, e.g. $\{x - n, x - n + 1, ..., x + n\}$.

Finding d from NCC

We can use NCC to obtain the disparity map:

$$d(x, y) = \arg\max_{d} \mathrm{NCC}(i, i', x, y, d, n). \tag{13}$$

Note that n determines how large is the neighborhood around points we consider for matching. We have to select n carefully. If we use n too small we can get more detail, but also more noise. With larger n we get smoother disparity maps, but less detail. Note that these types of approaches are known as **block matching** or **window-based matching**.

Block Matching Alternatives

The algorithm may fail with occlusions, specularities and textureless surfaces. We can improve upon it by considering some other constraints:

- Uniqueness - only one match per image point in i
- Ordering - points on the same surface will be in the same order in both views
- Disparity gradient - $\frac{\partial d}{\partial x}$ is small for points on the same surface

Recently some deep learning methods can also be used to leverage more (domain-specific) information.

Adding More Views

We select a reference image i_R and calculate the depths $z_j(x, y)$ from pairs with with subsequent images i_j. Thus we gain different hypotheses for the values of $z(x, y)$ in the reference image. We can then get a **disparity space image** (DSI). We denote it as

$$C(x, y, Z) = \sum_{j \in \{R+1, \ldots R+n\}} \rho(i_R(x, y), i_j(x', y', Z)), \tag{14}$$

where x', y' are calculated from the relative poses of the views and z and ρ is a metric.

Finding the depth

For each pixel x, y we then select the the depth z which minimizes $C(x, y, z)$:

$$Z = \arg\min_{z} C(x, y, Z(x, y)). \tag{15}$$

We can express this globally if we consider z to be a function:

$$Z(x, y) = \arg\min_{z} \sum_{x,y} C(x, y, Z(x, y)). \tag{16}$$

Regularization

We may force the image to have some global property such as smoothness by instead finding:

$$Z(x,y) = \arg\min_{Z} C(x,y,Z) + \lambda S(Z) \tag{17}$$

where S is a regularization function which may depend on all points in the image thus introducing global information into the optimization, λ determines how strong the regularization is. For examples we can select S to make the surfaces smooth:

$$S = \sum_{x,y} \left(\frac{\partial z}{\partial x}\right)^2 + \left(\frac{\partial z}{\partial y}\right)^2. \tag{18}$$

This term smooths the map and fills the holes. We may ignore the term S for $z(x,y)$ with high accuracy and certainty. Several deep learning approaches also exist.

RANSAC

Let us recall the standard RANSAC algorithm:

1. Sample n data points at random.
2. Calculate a model M using the n data points.
3. Calculate how many of the remaining data points have low errors w.r.t. model M.
4. Perform steps 1-3 multiple times and then pick M with the most inliers (low-error data points).

Usually, RANSAC is used in conjunction with the so-called minimal models. E.g. the methods that use the minimal number of data points (correspondences, points, etc.) to obtain a model (E, F, H, line, plane etc.).

RANSAC iterations

The standard RANSAC parameters are the number of iterations n and usually some threshold t for considering a point an inlier w.r.t the model. We can also use a threshold for inliers of a model in which case we terminate before hitting n. Let us now consider the probability μ that RANSAC will find a good model considering that we assume that our data has w ratio of inliers to total number of data points and we sample m data points to find the model. We can easily see that for the probability p of getting only inliers in n iterations is

$$1 - \mu = (1 - w^m)^n. \tag{19}$$

From this we can derive that

$$n = \frac{\log(1 - \mu)}{\log(1 - w^m)}. \tag{20}$$

Modifications of RANSAC

RANSAC is conceptually simple, but it can be modified in many ways

- Non-uniform sampling
- Local optimization
- Degenerate configuration
- Randomized verification
- Methods without threshold
- Differentiable methods

NAPSAC

Inliers are often grouped in clusters together. A simple idea is to sample the datapoints which are close together. For example in NAPSAC for line we select a first point uniformly and then we select the second point in the local neighborhood.

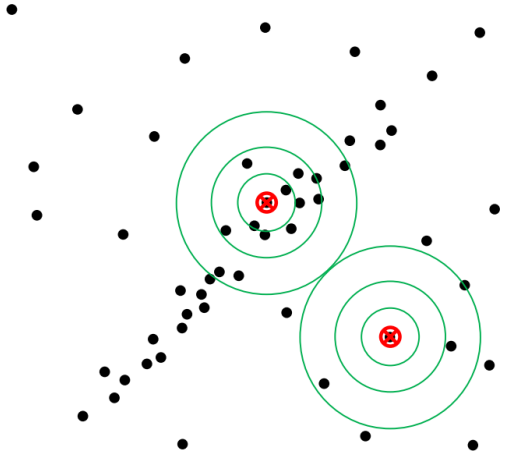


Image adopted from: Ondřej Chum. *RANSAC in 2020 tutorial: Traditional approaches.*

Using Datapoint Quality

Note that not all data points are always equal. We can very often compare their quality (e.g. using scores from matching algorithms etc.). We could leverage this fact by simply sampling all of the samples at once and then evaluating them in the order given by the quality of the worst datapoint in a sample.

We may find a good fit earlier, but this is inefficient. We can use the quality before sampling!

PROSAC

We order all of the datapoints by their quality obtaining a sequence $(p_1, p_2, ..., p_N)$. We then perform the following algorithm:

1. Let $n := m, t := 0$.
2. $t := t + 1$, if $t = T'_n$ then $n := n + 1$.
3. If $T'_n < t$
   - Then: use p_n in the sample and choose the remaining $m - 1$ points from $(p_1, p_2, ...p_{n-1})$.
   - Else: select m points from $(p_1, p_2, ...p_n)$.
4. Find the model M using the sample
5. Repeat from 2 until stopping criterion is reached.

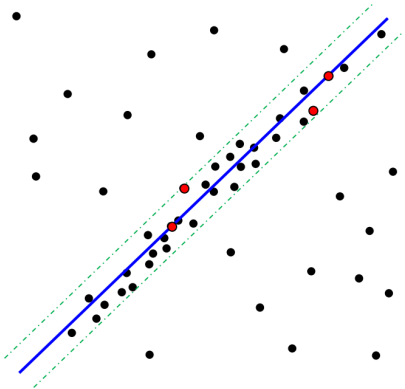
T'_n in this case determines how many new samples can be sampled from $(p_1, p_2, ...p_{n-1})$ as opposed to $(p_1, p_2, ...p_n)$.

PROSAC - Stopping Criterion

The stopping criterion has two conditions.

- **Non-random solution** - This means that for a model to be considered correct, it must have more inliers than what would randomly happen for an incorrect model. This is computed by estimating the probability that an incorrect model (fitted on a sample) is supported by a given point, not in the sample.
- **Maximality** - We want to stop sampling when the chance of getting a model that fit more points is lower than a certain threshold. This is computed by looking at the odds of missing a set of inliers bigger than previously found after a number of draws. If this probability falls under a certain threshold (usually 5%), it is not worth continuing drawing and we terminate.

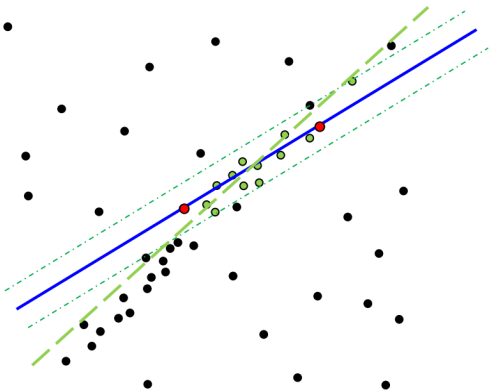
RANSAC All-inlier termination



At the end RANSAC terminates with an all-inlier sample.

Image adopted from: Ondřej Chum. *RANSAC in 2020 tutorial: Traditional approaches*.

RANSAC All-inlier models



But not all all-inlier models would get the same number of inliers.

Image adopted from: Ondřej Chum. *RANSAC in 2020 tutorial: Traditional approaches*.

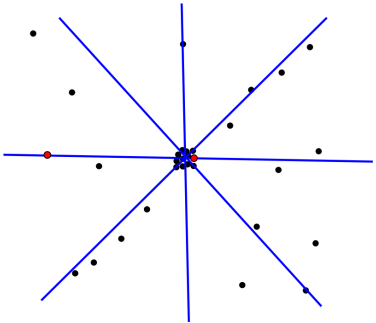
Local Optimization

If we already find an all-inlier sample we may perform optimization w.r.t its inliers to get a better model. Optimization is usually expensive so we do this only when we hit the new best model. The odds of finding a new model are logarithmic w.r.t the number of iterations so we do not have to do this so often.

To find a locally optimal model we may use:

- Least squares fit
- Iteratively re-weighted least squares
- Graph-cut

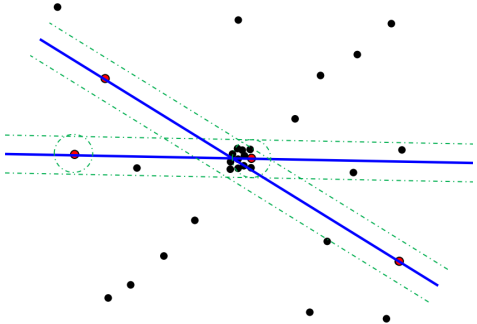
Degenerate Configurations



Consider that we have a degenerate configuration in our data. The problem is that we may get lot of inliers from this degenerate configuration, but these the degenerate configuration leads to infinite number of correct solutions.

Image adopted from: Ondřej Chum. *RANSAC in 2020 tutorial: Traditional approaches*.

Exploiting Degenerate Configurations



If we find a degenerate configuration we can exploit it by assuming that points from such configuration are likely to come from inliers.

Image adopted from: Ondřej Chum. *RANSAC in 2020 tutorial: Traditional approaches*.

F with Coplanar Points - DEGENSAC

If a dominant plane is present it is very likely to sample 4 or more coplanar points. If we sample 6 or 7 such points we will hit a degenerate configuration for computation of F by having an infinite solutions for F. On the other hand we may easily estimate a homography H from such points. Now consider that if we have a valid homography H we can parametrize

$$F = [\boldsymbol{e}']_{\times} H. \tag{21}$$

Thus if we have H we need only two more points to find the fundamental matrix. Sampling 2 inliers is orders of magnitude more likely than sampling 7. The algorithm may therefore terminate earlier.

Randomized Verification

We calculate the quality of the model using all data points. This may be unnecessary for bad models. We can perform pre-verification on a randomly sampled subset of the data points.

This introduces some issues. We may get false positives resulting in inefficiencies. We may also get false negatives which results in decreased probability of finding a correct model after n iterations.

Setting the Threshold

We still needed to select a threshold for the previous approaches. This may sometimes be undesirable as the optimal threshold depends on the noise level, which may differ across various data.

We may want to estimate the model simultaneously with the threshold. We can do this by consider multiple noise scales (thresholds) during the RANSAC loop. To consider this we may need some assumptions on the distributions of the inliers and outliers.

With the assumption of the ξ^2 distribution for the squared inlier residuals (errors) and uniform distribution for the outliers we can calculate the model likelihood:

$$L(\theta|\sigma) = \frac{1}{L^{N-|I(\sigma)|}} \prod_{x \in I(\sigma)} \left[2C(p)\sigma^{-p} D^{p-1}(\theta, x) e^{\frac{-D^2(\theta,x)}{2\sigma^2}} \right], \tag{22}$$

where θ are the model parameters, σ is the noise level, L is the range of sensor (e.g. image diagonal in pixels), N is the total number of data points $I(\sigma)$ is the set of inliers implied by σ, C is the distribution constant, p are the problem degrees of freedom and D is the distance function.

We then use a new quality function (as opposed to inlier count) to select the best model:

$$Q^*(\theta) = \frac{1}{\sigma_{max}} \int_0^{\sigma_{max}} \ln L(\theta|\sigma) \mathrm{d}\sigma, \tag{23}$$

where σ_{max} is the expected maximum noise level. Using Q^* does not depend on the level σ and thus selects the model without the need to choose a threshold.

Note that using this strategy we do not know which data points are inliers for final refinement, but we can calculate probabilities that they are inliers and use those as weights for final refinement.

MAGSAC Termination

For standard RANSAC we can rewrite the termination criterion (20):

$$n = (\theta, \sigma) = \frac{\ln(1-\mu)}{\ln\left(1 - \left(\frac{|I(\theta,\sigma)|}{N}\right)^m\right)}, \tag{24}$$

where μ is the desired confidence and m is the model size. In MAGSAC we can then use

$$n^*(\theta, \sigma) = \frac{1}{\sigma_{max}} \int_0^{\sigma_{max}} n(\theta, \sigma) \mathrm{d}\sigma. \tag{25}$$

Note that integration over σ is known as marginalization thus MAGSAC - Marginalized Sample Consensus.

Differentiable RANSAC

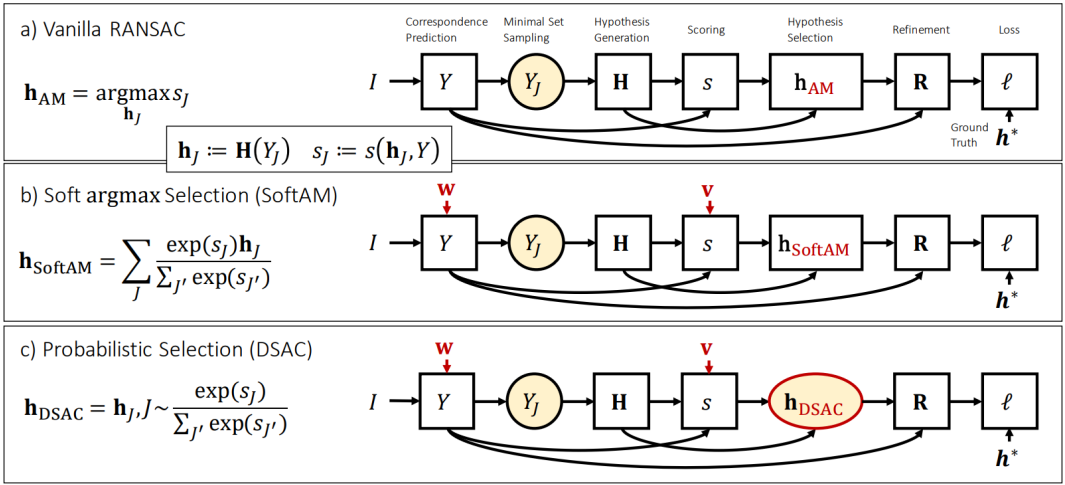


Image adopted from: Eric Brachmann et al. "Dsac-differentiable ransac for camera localization." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 6684–6692