Exercise Sheet 3

Topic: Feature Detectors, Descriptors, Epipolar Geometry, RANSAC

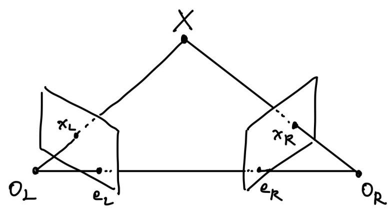
Yun-Jin Li

November 18, 2022

Par1: ORB Descriptors

See implementation on GitLab.

Part 2: Epipolar constraint



We know that the unprojected vector x_L, x_R are normalized vectors

Let's say the real distance $\overline{O_L X}$, $\overline{O_R X}$ are λ_L and λ_R

We first assume X is represented in O_R camera fr

$\lambda_R x_R = X \quad - (1)$

$\lambda_L x_L = R_{LR} X + T_{LR} \quad - (2)$

Take (1) into (2)

$\Rightarrow \lambda_L x_L = R_{LR}(\lambda_R x_R) + T_{LR}$

We try to multiply $\hat{T}_{LR}$ from both side to eliminate T_{LR}

$\Rightarrow \lambda_L \hat{T}_{LR} x_L = \lambda_R \hat{T}_{LR} R_{LR} x_R$

From the epipolar constraint, $O_L x_L$, T_{LR}, $O_R x_R$ should lie on the same plane, since $\hat{T}_{LR} x_L = T_{LR} \times x_L \perp x_L$

If we try to compute the inner product of x_L and $T_{LR} \times x_L$, we would get zero since they are perpendicular.

So, by multiplying both side with x_L^T, we have:

$\Rightarrow \lambda_L \underbrace{x_L^T (T_{LR} \times x_L)}_{=0,\ \text{since } x_L \perp (T_{LR} \times x_L)} = \lambda_R x_L^T \hat{T}_{LR} R_{LR} x_R$

$\Rightarrow x_L^T \hat{T}_{LR} R_{LR} x_R = 0$

And we define our essential matrix $E = \hat{T}_{LR} R_{LR}$

$\Rightarrow x_L^T E x_R = 0 \quad \#$

Part 3: Five-point algorithm and RANSAC

See implementation on GitLab.

Part 4: Bag-of-Words for Place Recognition

The main difference between **match_all()** and **match_bow()** is how they determine the vector **ids_to_match**. The method match_all() basically adds all the image pairs when two images are not in the same frame. However, the method match_bow() iterates through all the pairs of (FrameCamId, KeypointsData), and for each frame (image), it transforms all the corner descriptors into bag-of-words vector v, and then uses query() to find a number of candidate frames (images) from the database. In the end, it only adds pair of current frame (image) ID and its candidate frames (images) IDs into vector **ids_to_match**, which could reduce lots of unnecessary image pairs. The number of candidate frames (images) is controlled by the parameter **num_bow_candidates**.

After successfully implementing the BoW matching method, we now compare the number of candidate pairs and inliers using two different method **match_all()** and **match_bow()**.

match_all() has 13284 candidate pairs and about 44336 inliers features, while match_bow has 3649 candidate pairs and about 23938 inliers features (when parameter num_bow_candidates is set to 25). If now we have 2 x 1000 images, then match_all() would have 1998000 candidate pairs and match_bow would have candidates at some number below 50000 (when parameter num_bow_candidates is set to 25). It's obvious to see that we can reduce a lot of computational cost when use match_bow() method when dealing with a large number of images.

References

As suggested in the exercise sheet.