(T_{LR}) $O_L O_R$ is known

R_{LR} is known : $\underbrace{X_L, X_R}$: bearing vectors obtained by unprojecting
deected 2D image points

$\therefore$ ~~A~~ccording to rigid body transformation

$$X_L = R_{LR} X_R + T_{LR}$$

$$\Big\downarrow \text{ crossproduct } T_{LR}$$

$$T_{LR} \times X_L = T_{LR} \times (R_{LR} X_R)$$

$$\Big\downarrow \text{ dot Product } X_L$$

$$0 = \underbrace{X_L^T (T_{LR} \times X_L)} = X_L^T \hat{T}_{LR} R_{LR} X_R$$

skew symmetric

$$\therefore \begin{cases} X_L^T \hat{T}_{LR} R_{LR} \cancel{X}_R = 0 \\ X_L^T E X_R = 0 \end{cases} \Rightarrow E = \hat{T}_{LR} R_{LR} = \widehat{O_L O_R} R_{LR}$$

Part 4:

1° main difference between match_all, match_bow:

match_all use brute force to find all matches. e.g. {frame2, frame1}
{frame3, frame2}. {frame3, frame1} ....

However, match_bow will first read the bow vocabulary, then for each frame to find its bow-vector, then query that in bow-db to find the matches.

2° num_bow_candidates: restrict how many matches it will add at most 25

in code k < min(result_raw.size(), num_bow_candidates)

3° After implementing. in our case we have 2×82 images.
then for 2×1000 images: set N = average corner of one image

2×82 images: ∴ 2×1000×N×25