SLAM

Обзор

Что мы уже умеем?

Отображать точки из 3D в 2D и обратно
Преобразовывать 3D точки между камерами
Матчить точки между изображениями

Чему мы хотим научиться?

Находить позы камер
Находить глубины точек

О чем будем говорить?

1. Немного о трансформациях SE(3)
2. Эпиполярная геометрия, существенная матрица
3. Восстановление позы по матчам точек
4. Восстановление 3D точек по найденной позе
5. Восстановление позы по матчам точек с глубиной
6. Шум

SE(3)

Для описания положения камеры
В пространстве могут использоваться:

1. SE(3)
2. Quaternion + translation
3. Euler angles + translation (Gimbal lock)

Отличаются только описанием вращения,
которые могут преобразовываться друг в друга

$$R = R_x(\alpha)R_y(\beta)R_z(\gamma) = \underset{yaw}{\begin{bmatrix}\cos\alpha & -\sin\alpha & 0\\ \sin\alpha & \cos\alpha & 0\\ 0 & 0 & 1\end{bmatrix}}\underset{pitch}{\begin{bmatrix}\cos\beta & 0 & \sin\beta\\ 0 & 1 & 0\\ -\sin\beta & 0 & \cos\beta\end{bmatrix}}\underset{roll}{\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\gamma & -\sin\gamma\\ 0 & \sin\gamma & \cos\gamma\end{bmatrix}} =$$

$$= \begin{bmatrix}\cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma\\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma\\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma\end{bmatrix}$$

SE(3). Из мира в камеру и наоборот.

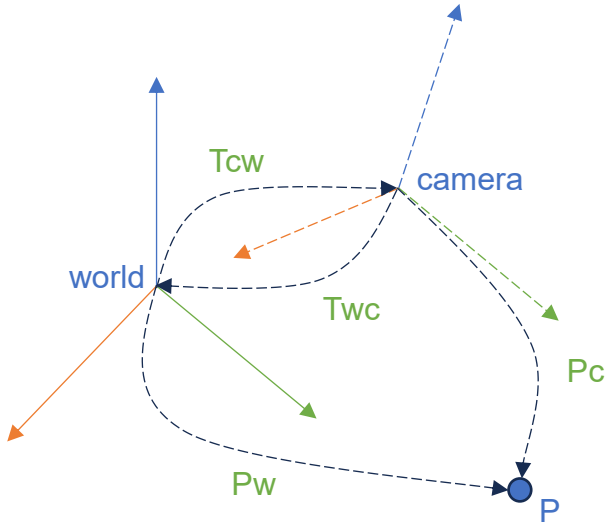
$$P_{world} = P_{world-camera} \cdot P_{camera}$$

$$P_{camera} = T_{camera-world} \cdot P_{world}$$

$$T_{world-camera} = (T_{camera-world})^{-1}$$

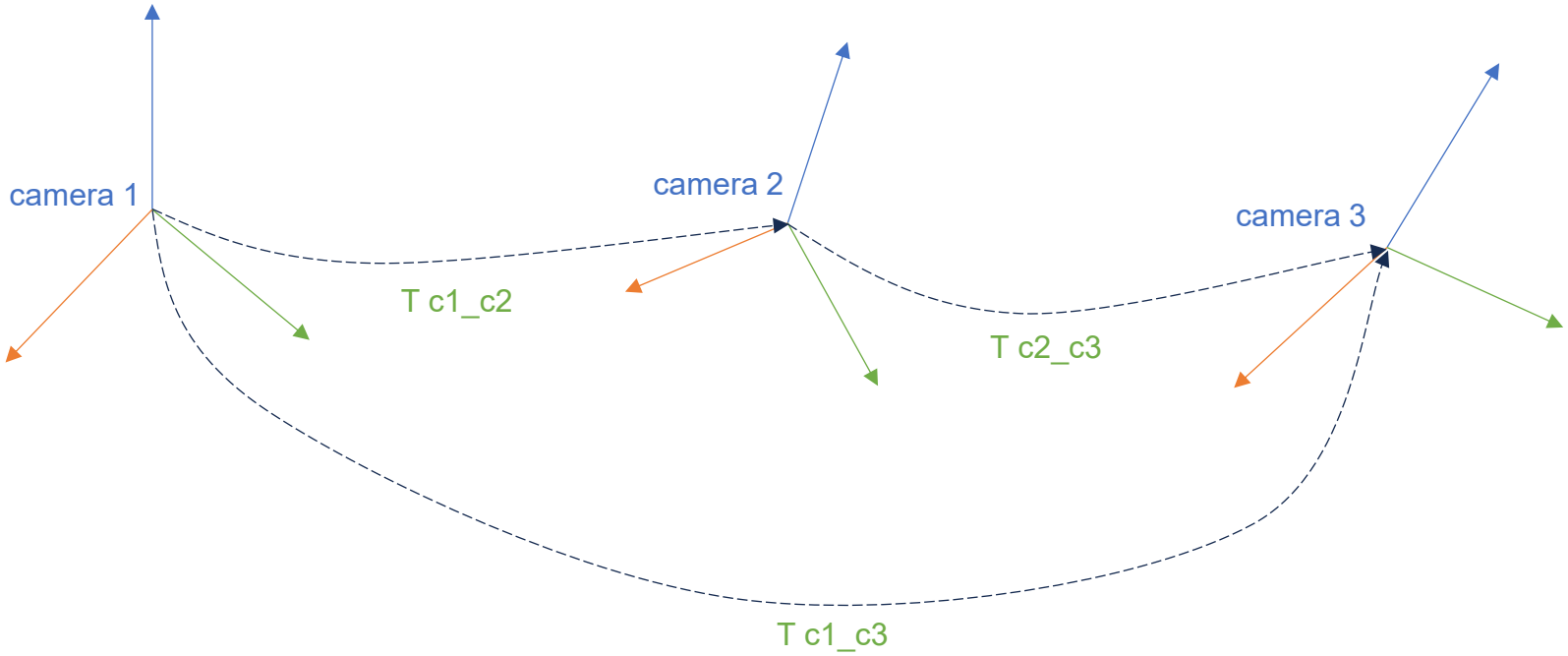
$$T_{camera-world} = \begin{pmatrix} R_{cw} & t_{cw} \\ 0 & 1 \end{pmatrix}$$

$$T_{world-camera} = \begin{pmatrix} R_{cw}^T & -R_{cw}^T \cdot t_{cw} \\ 0 & 1 \end{pmatrix}$$



$R^T R = I \Rightarrow R^T = R^{-1}$ **Ортогональность** – свойство матриц вращения

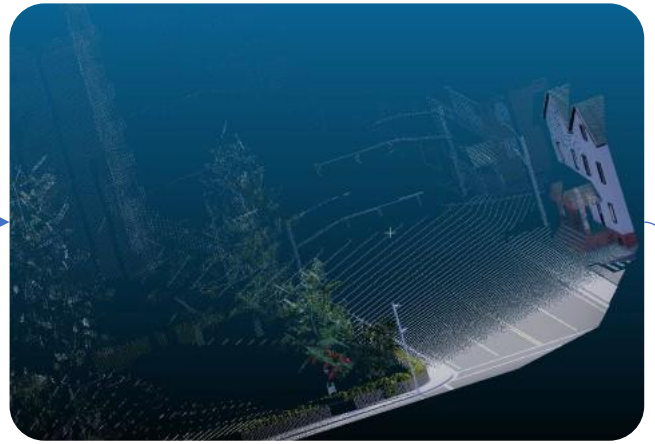
SE(3). Основные операции.



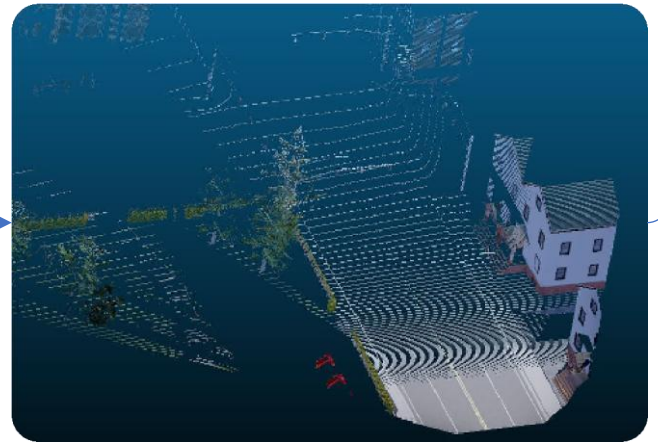
$$T_{camera1-camera3} = T_{camera1-camera2} \cdot T_{camera2-camera3}$$

Формулировка проблемы

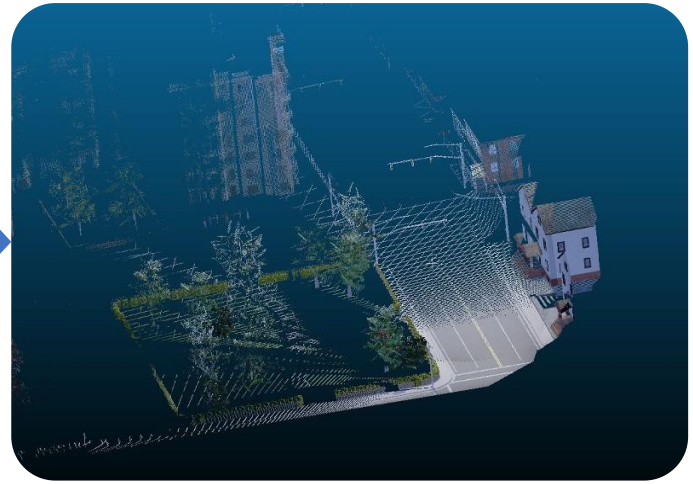
Depth map 1



Depth map 2



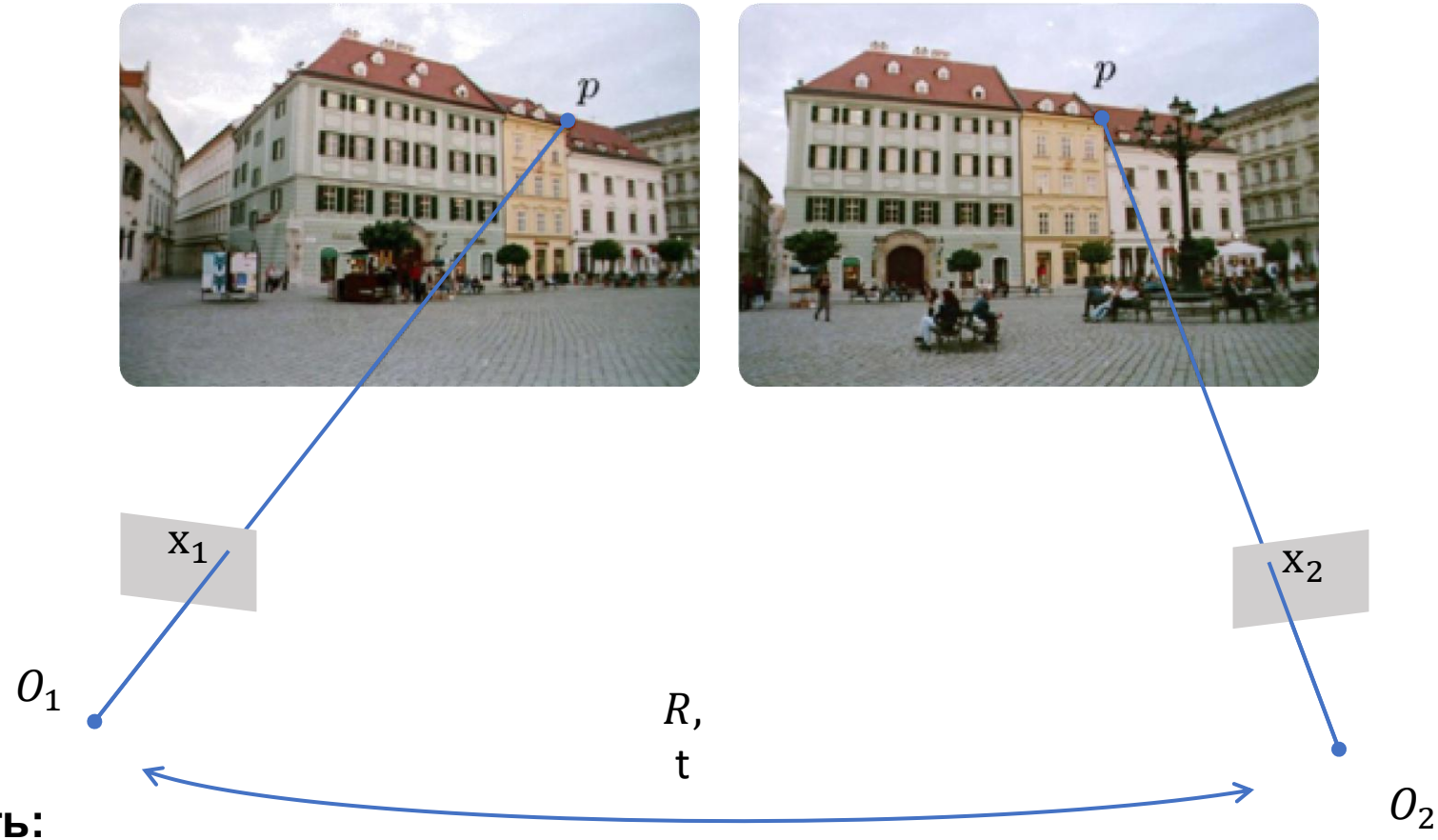
R, t



Формулировка проблемы

В целом 3D-реконструкция представляет собой сложную задачу.

Если нам дано два вида со 100 Характерными точками в каждом из них, то у нас есть 200 координат точек в 2D.



Цель состоит в том, чтобы оценить:

$$E(R,T,X_j,\dots,X_{100}) = \sum_j \left\| \mathrm{x}_1^j - \pi(X_j) \right\|^2 + \left\| \mathrm{x}_2^j - \pi(R,T,X_j) \right\|^2$$

Эпиполярная геометрия

x_1 – луч на первой камере (back-project от фичи 1)

x_2 – луч на второй камере (back-project от фичи 2)

λ_1 – глубина точки на первой камере

λ_2 – глубина точки на второй камере

$\lambda_1 \mathrm{x}_1 = X \qquad \lambda_2 \mathrm{x}_2 = RX + T$

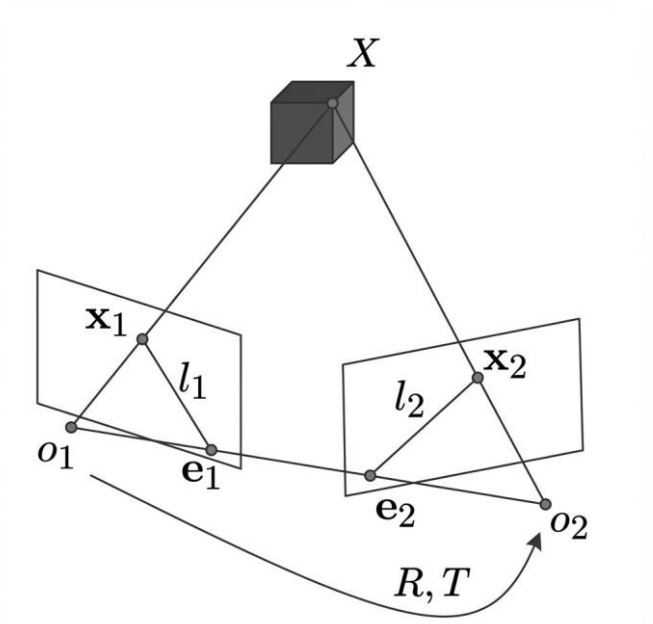
3D точка в первой и второй камерах

$\lambda_2 \mathrm{x}_2 = R(\lambda_1 \mathrm{x}_1) + T$

Подставляем X из первого уравнения во второе

$\lambda_2 \hat{T} \mathrm{x}_2 = \lambda_1 \hat{T} R \mathrm{x}_1 \qquad \hat{T} v \equiv T \times v \qquad \hat{T} = \begin{pmatrix} 0 & T_3 & T_2 \\ T_3 & 0 & -T_1 \\ -T_2 & T_1 & 0 \end{pmatrix}$

$\mathrm{x}_2^T \hat{T} R \mathrm{x}_1 = 0 \longrightarrow$ **epipolar constraint**



Существенная матрица

Epipolar constraint предоставляет отношение между двумя 2D точками. Здесь больше не нужна 3D точка.

$\mathrm{x}_2^T \hat{T} R \mathrm{x}_1 = 0 \longrightarrow$ epipolar constraint

$E = \hat{T}R \quad \in \mathbb{R}^{3\times 3}$

– существенная матрица **(Essential matrix)**

$E = U\Sigma V^T \quad \Sigma = diag\{\sigma, \sigma, 0\}$

Существенная матрица

Свойства существенной матрицы:

Epipolar constraint геометрически означает,
что векторы $\mathrm{x}_1, \mathrm{x}_2, T$ лежат в одной плоскости

Существенная матрица имеет специальный вид разложения SVD

Как получить вращение и перемещение из существенной матрицы?

$$(\hat{T}_1, R_1) = \left(UR_Z\left(+\frac{\pi}{2}\right)\Sigma U^T, UR_Z^T\left(+\frac{\pi}{2}\right)V^T\right) \qquad (\hat{T}_2, R_2) = \left(UR_Z\left(-\frac{\pi}{2}\right)\Sigma U^T, UR_Z^T\left(-\frac{\pi}{2}\right)V^T\right)$$

Базовый алгоритм реконструкции

Восстановить существенную матрицу
из набора фич на левой и правой картинке

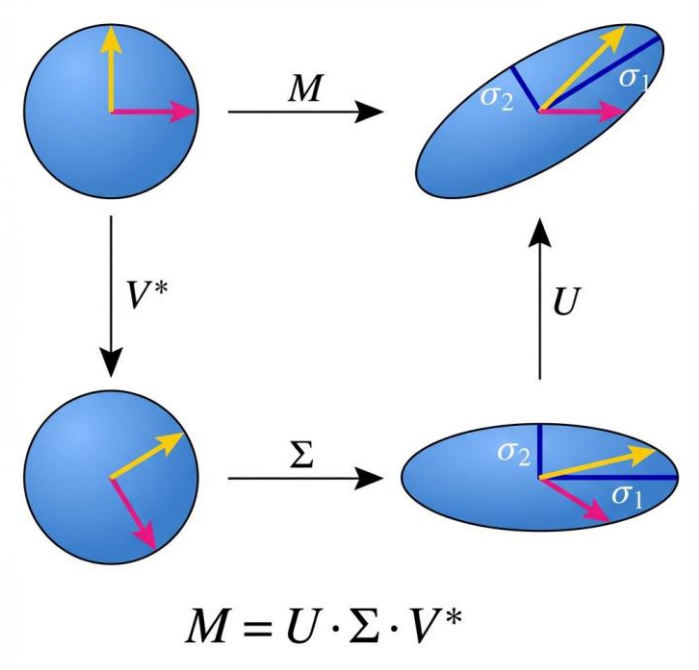
$$(x_2^T)_i E (x_1)_i = 0$$

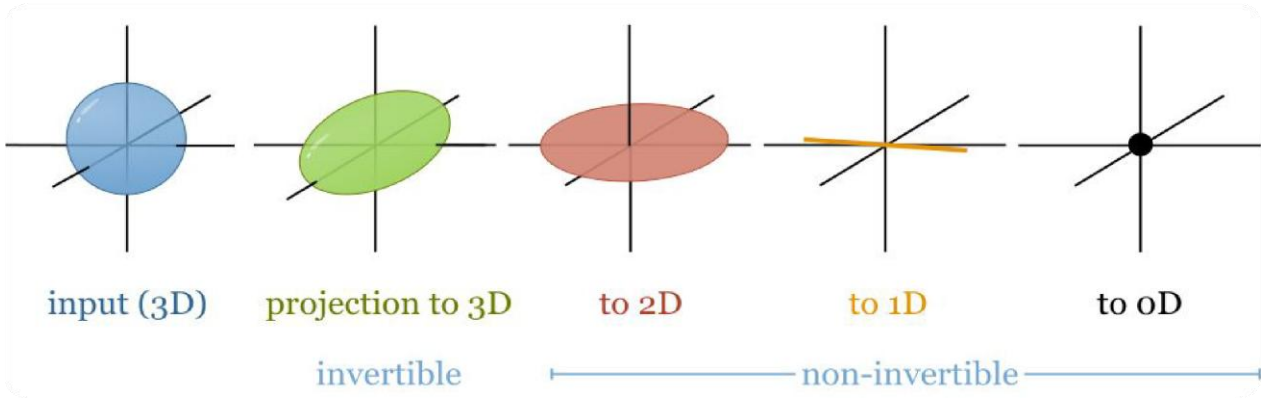
Восстановить R, T из существенной матрицы

$$\left(\hat{T}_1, R_1\right) = \left(UR_Z\left(+\frac{\pi}{2}\right)\Sigma U^T, UR_Z^T\left(+\frac{\pi}{2}\right)V^T\right)$$

$$\left(\hat{T}_2, R_2\right) = \left(UR_Z\left(-\frac{\pi}{2}\right)\Sigma U^T, UR_Z^T\left(-\frac{\pi}{2}\right)V^T\right)$$

Singular Value Decomposition





8-точечный алгоритм (The Eight-Point Linear Algorithm)

Перепишем немного epipolar constraint.
Представим существенную матрицу
E в виде вектора.

$$E^s = (\mathrm{e}_{11}, \mathrm{e}_{21}, \mathrm{e}_{31}, \mathrm{e}_{12}, \mathrm{e}_{22}, \mathrm{e}_{32}, \mathrm{e}_{13}, \mathrm{e}_{23}, \mathrm{e}_{33})^T \in \mathbb{R}^9$$

Так же введем комбинацию
векторов $\mathbf{x}_1$ и $\mathbf{x}_2$

$$a = (\mathrm{x}_1\mathrm{x}_2, \mathrm{x}_1\mathrm{y}_2, \mathrm{x}_1\mathrm{z}_2, \mathrm{y}_1\mathrm{x}_2, \mathrm{y}_1\mathrm{y}_2, \mathrm{y}_1\mathrm{z}_2, \mathrm{z}_1\mathrm{x}_2, \mathrm{z}_1\mathrm{y}_2, \mathrm{z}_1\mathrm{z}_2)^T \in \mathbb{R}^9$$

$$\mathbf{x}_i = (\mathrm{x}_i, \mathrm{y}_i, \mathrm{z}_i)$$

Тогда epipolar constraint можно переписать

$$\mathbf{x}_2^T E \mathbf{x}_1 = a^T E^s = 0$$

$$\chi^T E^s = 0, \quad \chi^T = (a^1, a^2, \ldots, a^n)^T$$

8-точечный алгоритм (The Eight-Point Linear Algorithm)

Полученная матрица E^s в общем случае не является существенной матрицей, так как не удовлетворяет ее свойствам. Самый простой способ – найти ближайшую существенной к найденной E^s

Ближайшая матрица по норме Фробениуса

$\|E - E^s\|_f^2$

$$E^s = U \cdot diag\{\sigma_1, \sigma_2, \sigma_3\} \cdot V^T$$

$$\sigma_1 \geq \sigma_2 \geq \sigma_3$$

$$\longrightarrow$$

$$E^s = U \cdot diag\{\sigma_1, \sigma_2, \sigma_3\} \cdot V^T$$

$$\sigma = \frac{\sigma_1 + \sigma_2}{2}$$

На деле, так как E определена с точностью до умножению на константу, то можно принять $\sigma = 1$

Eight-Point Algorithm (Longuet-Higgins ’81)

Дан набор лучей $\mathbf{x}_1, \mathbf{x}_2, n \geq 8$ Эти 3D лучи получены путем обратной проекции 2D фич

Посчитать приближение существенной матрицы

$a = (\mathrm{x}_1\mathrm{x}_2, \mathrm{x}_1\mathrm{y}_2, \mathrm{x}_1\mathrm{z}_2, \mathrm{y}_1\mathrm{x}_2, \mathrm{y}_1\mathrm{y}_2, \mathrm{y}_1\mathrm{z}_2, \mathrm{z}_1\mathrm{x}_2, \mathrm{z}_1\mathrm{y}_2, \mathrm{z}_1\mathrm{z}_2)^T \in \mathbb{R}^9$

$\chi = (a^1, a^2, \ldots, a^n)^T$

$\chi\, E^s = 0, \quad SVD\chi = U_\chi \Sigma_\chi V_\chi^T \quad E^s$ – последний столбец V_χ

E^s – превращаем из вектор 9 в матрицу 3×3

Ищем ближайшую существенную матрицу

$E^s = U \cdot diag\{\sigma_1, \sigma_2, \sigma_3\} \cdot V^T$

$\sigma_1 \geq \sigma_2 \geq \sigma_3$

$\longrightarrow E = U \cdot diag\{1,1,0\} \cdot V^T$

Восстанавливаем R и T из E

$R = UR_Z^T\left(\pm\frac{\pi}{2}\right)V^T, \quad \hat{T} = UR_Z\left(\pm\frac{\pi}{2}\right)\Sigma U^T \qquad R_Z^T\left(\pm\frac{\pi}{2}\right) = \begin{pmatrix} 0 & \pm 1 & 0 \\ \mp 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Только одно из этих 4х решений даст адекватное положение точек при триангуляции

Минимальные задачи

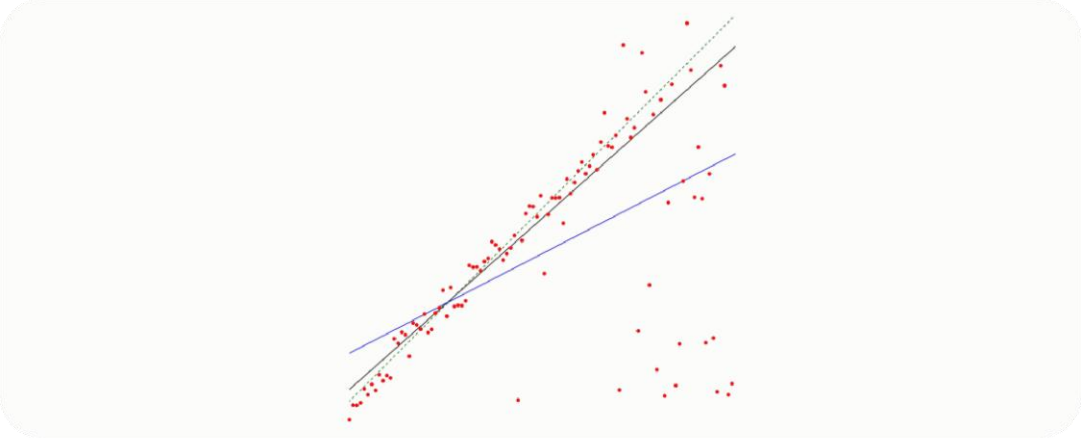
8 точечная задача – не минимальная. У E 5 степеней свободы – 3 степени свободы у R
3 степени свободы у T, -1 степени свободы на $scale$.

Примеры минимальных задач: 5-pt algorithm – relative pose
6-pt algorithm – relative pose and focal length

Много точек для минимальных задач

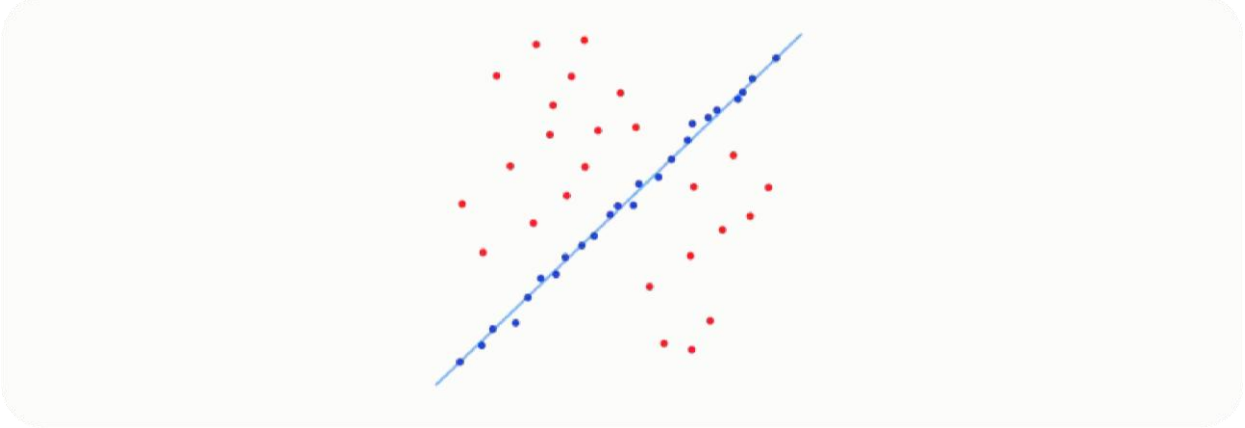
Что если мы нашли много матчей между изображениями,
но хотим использовать 5pt алгоритм

Метод наименьших квадратов



Решить какую-нибудь
оптимизационную задачу
(примерно это мы сделали в 8pt алгоритме)

RANSAC



Взять 5 случайных точек, посчитать R и T,
посчитать количество инлаеров. Продолжать так
до тех пор, пока не найдется максимальное
количество инлаеров.

Восстановление точек

Мы восстановили E, а из него R и T с точностью до скейла. Возвращаемся к набору точек (точнее, лучей).

$\lambda_2^j \mathbf{x}_2^j = \lambda_1^j R \mathbf{x}_1^j + \gamma T, \qquad j = 1, \dots, n,$ добавляем γ как скейл, в остальном это просто трансформация точки j

$\lambda_1^j \widehat{\mathbf{x}_2^j} R x_1^j + \gamma \widehat{\mathbf{x}_2^j} T = 0, \qquad j = 1, \dots, n,$ домножаем все на $\widehat{\mathbf{x}_2^j}$

$\left(\widehat{\mathbf{x}_2^j} R \mathbf{x}_1^j, \widehat{\mathbf{x}_2^j} T\right) \begin{pmatrix} \lambda_1^j \\ \gamma \end{pmatrix} = 0, \qquad j = 1, \dots, n.$ превращаем в линейную систему

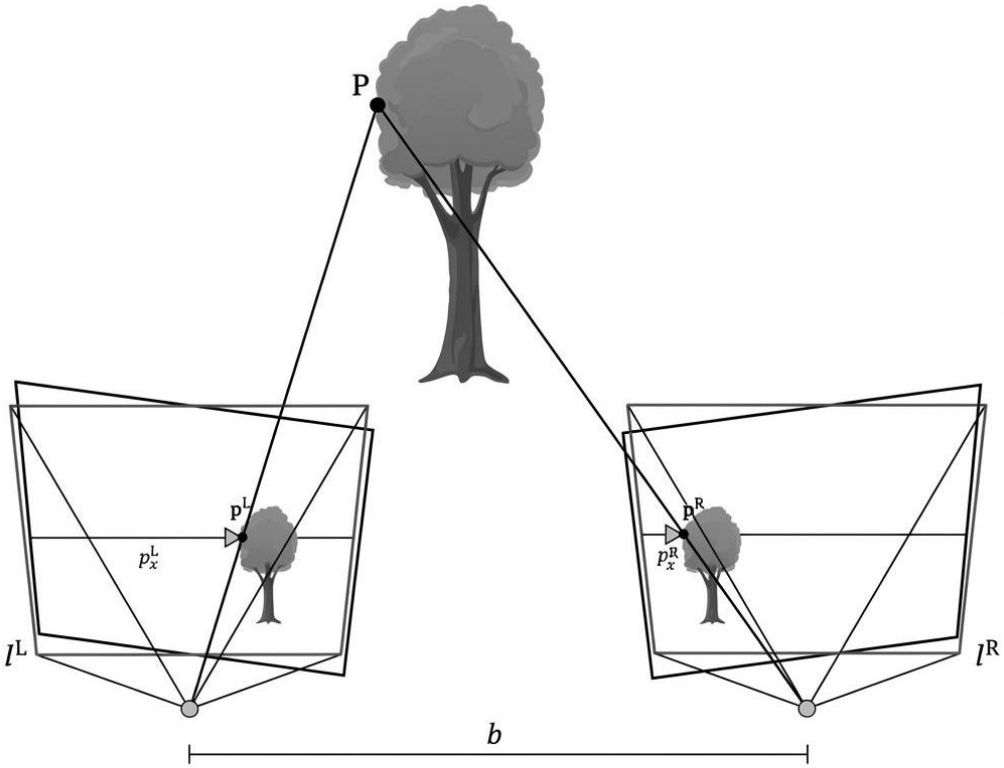
$\vec{\lambda} = (\lambda_1^1, \lambda_1^2, \dots, \lambda_1^n, \gamma)^T \in \mathbb{R}^{n+1}$ вектор λ – это столбец из всех глубин и в конце скейл

$$M\vec{\lambda} = 0 \qquad M \equiv \begin{pmatrix} \widehat{\mathbf{x}_2^1} R \mathbf{x}_1^1 & 0 & 0 & 0 & 0 & \widehat{\mathbf{x}_2^1} T \\ 0 & \widehat{\mathbf{x}_2^2} R \mathbf{x}_1^2 & 0 & 0 & 0 & \widehat{\mathbf{x}_2^2} T \\ 0 & 0 & \ddots & 0 & 0 & \vdots \\ 0 & 0 & 0 & \widehat{\mathbf{x}_2^{n-1}} R \mathbf{x}_1^{n-1} & 0 & \widehat{\mathbf{x}_2^{n-1}} T \\ 0 & 0 & 0 & 0 & \widehat{\mathbf{x}_2^n} R \mathbf{x}_1^n & \widehat{\mathbf{x}_2^n} T \end{pmatrix}$$

Для всех n точек образуется система. Решается она уже знакомым образом, через SVD (λ – последний столбец в матрице V разложения SVD). Вектор λ определен с точностью до глобального скейла.

Пример: стереокамера

$$T_{right-left} = \begin{pmatrix} 1 & 0 & 0 & -b \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



$$x_{left} \to \begin{pmatrix} u_l \\ v_l \\ 1 \end{pmatrix} \to \begin{pmatrix} \frac{1}{f_x} & 0 & -\frac{p_x}{f_x} \\ 0 & \frac{1}{f_y} & -\frac{p_y}{f_y} \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} u_l \\ v_l \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{u_l - p_x}{f_x} \\ \frac{v_l - p_y}{f_y} \\ 1 \end{pmatrix} \to \begin{pmatrix} Z \cdot \frac{u_l - p_x}{f_x} \\ Z \cdot \frac{v_l - p_y}{f_y} \\ Z \end{pmatrix} \to T_{right-left} \cdot \begin{pmatrix} Z \cdot \frac{u_l - p_x}{f_x} \\ Z \cdot \frac{v_l - p_y}{f_y} \\ Z \end{pmatrix} = \begin{pmatrix} Z \cdot \frac{u_l - p_x}{f_x} - b \\ Z \cdot \frac{v_l - p_y}{f_y} \\ Z \end{pmatrix} = \begin{pmatrix} Z \cdot \frac{u_r - p_x}{f_x} \\ Z \cdot \frac{v_r - p_y}{f_y} \\ Z \end{pmatrix}$$

Координаты пикселя слева

Unproject пикселя слева

Луч от пикселя слева

3D точка в левой камере

3D точка в правой камере

3D точка в правой камере через пиксель

$$Z \cdot \frac{u_l - p_x}{f_x} - b = Z \cdot \frac{u_r - p_x}{f_x}$$

$$Z = \frac{b \cdot f_x}{u_l - u_r} = \frac{baseline \cdot focal_x}{disparity}$$

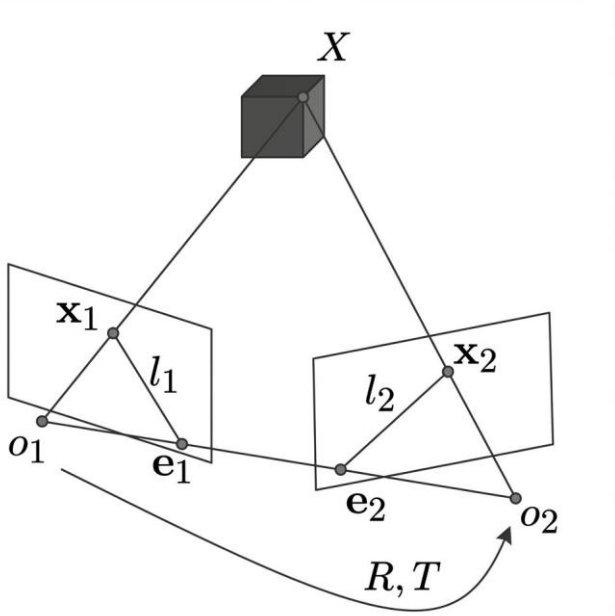
Восстановление глубины из стереопары

Фундаментальная матрица

$$\mathbf{x}_2^T \hat{T} R \mathbf{x}_1 = 0 \quad \Leftrightarrow \quad \mathbf{x'}_2^T K^{-T} \hat{T} R K^{-1} \mathbf{x'}_1 = 0$$

Это луч, back-project от фичи

Это 2D фича, в однородных координатах



$$\mathbf{x'}_2^T F \mathbf{x'}_1 = 0$$

Fundamental matrix

$$F \equiv K^{-T} \hat{T} R K^{-1} = K^{-T} E K^{-1}$$

$$x' = \begin{pmatrix} u \\ v \end{pmatrix} \rightarrow \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} \frac{1}{f_x} & 0 & -\frac{p_x}{f_x} \\ 0 & \frac{1}{f_y} & -\frac{p_x}{f_x} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} r_x \\ r_y \\ 1 \end{pmatrix} = x$$

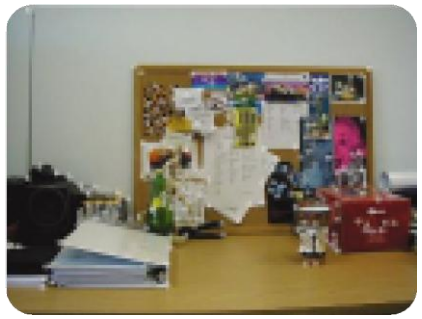
K^{-1}

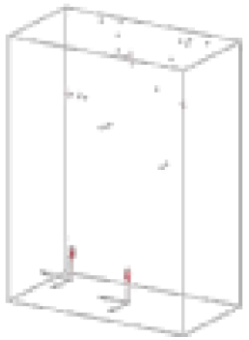
Промежуточные итоги

Подготовка

Калибруем камеру, получаем модель камеры и модель ректификации.

Простейшая реконструкция





Приходит изображение 1, убираем дисторсию, находим кейпоинты.

Приходит изображение 2, убираем дисторсию, находим кейпоинты, матчим их с первым изображением.

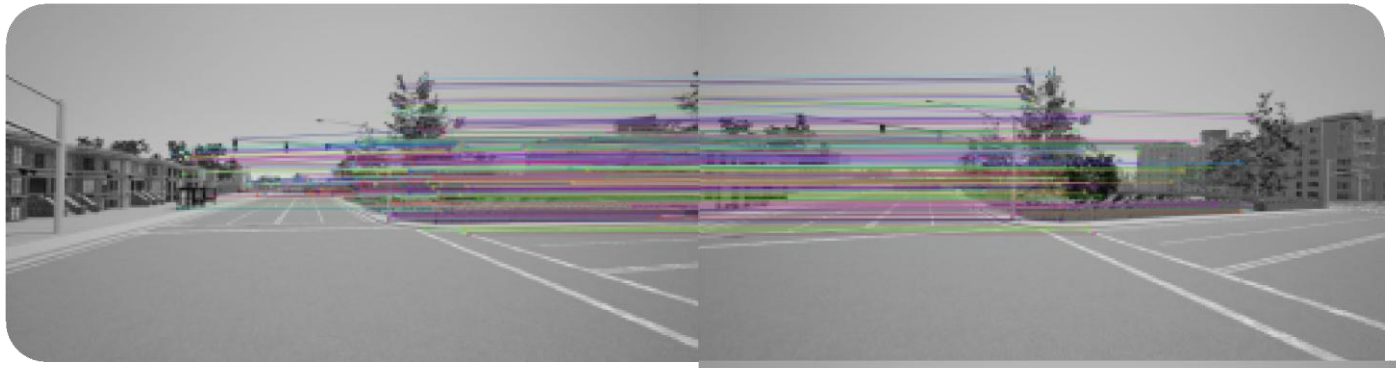
Восстанавливаем R и T между изображениями.

Восстанавливаем глубины для заматченных точек.

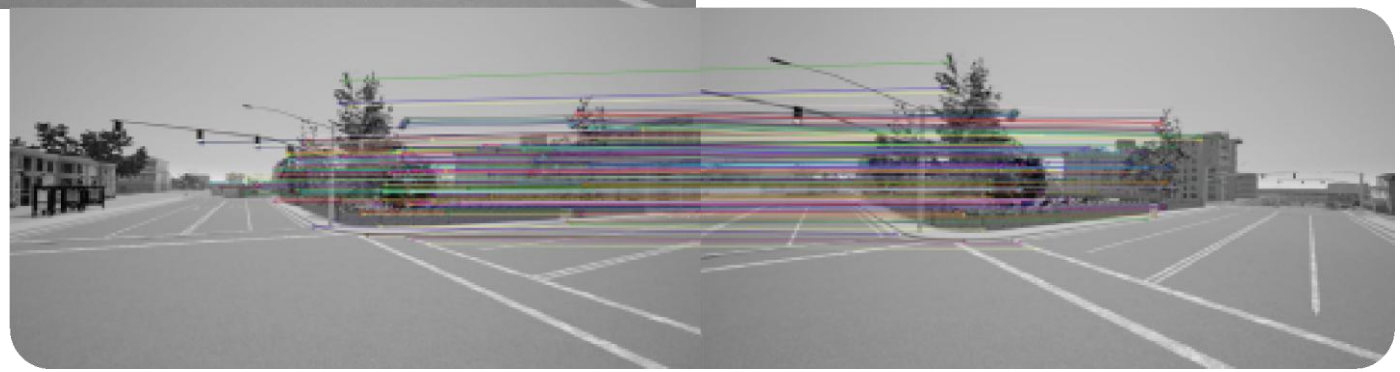
В SLAM это иногда называют инициализация.

А что дальше?

Frame 1 & Frame 2 → R и T между Frame 1 и Frame 2, 3D точки



Frame 1



Frame 2

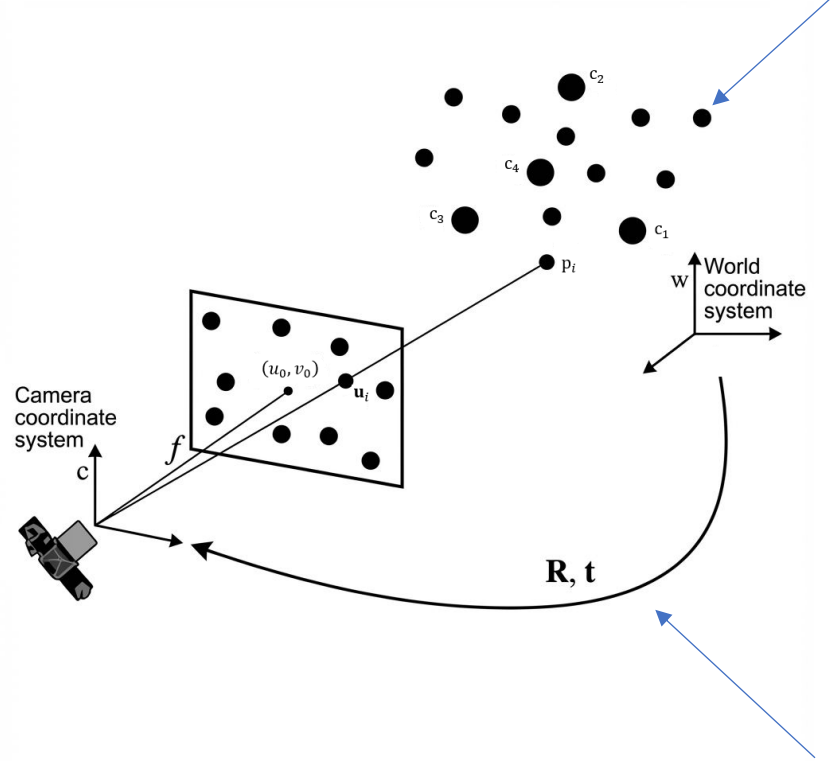
Frame 3

Frame 1 & Frame 2 & Frame 3 → ?

Perspective-n-Point (PnP)

Это новый Frame 3, мы сматчили его 2D фичи с 2D фичами Frame 2. То есть у нас есть информация, куда 3D проецируются на Frame 3.

Эти 3D точки мы восстановили из Frame 2



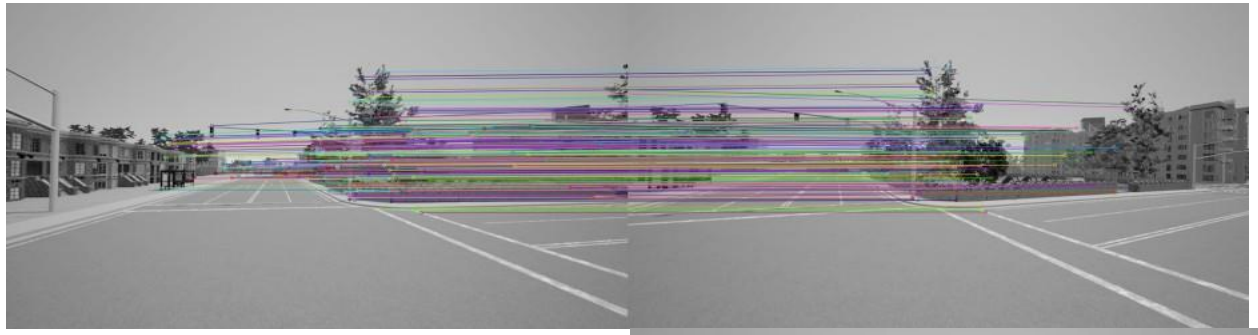
Минимальной задачей PnP является P3P (3 пары фич для нахождения R и T)

Задача – найти R и T

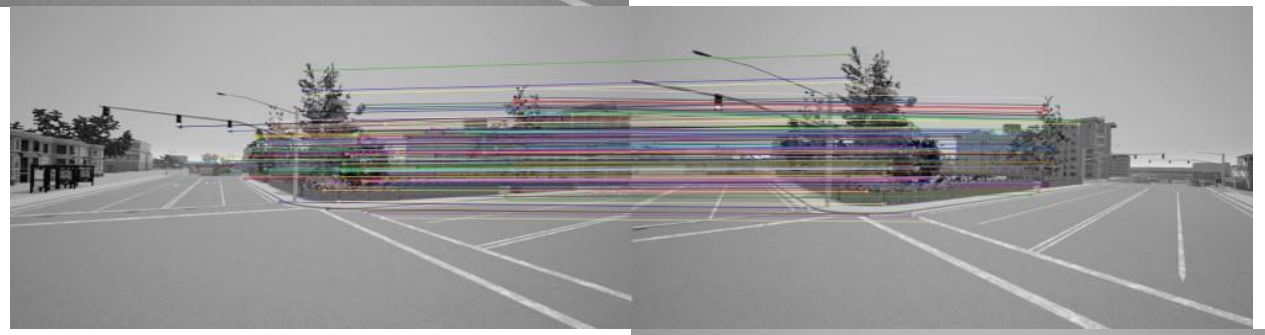
Простейший SLAM

Frame 1 & Frame 2

Матчим точки, находим R, T с помощью 5pt, триангулируем 3D точки



Frame 1



Frame 2

Frame 3

Frame 4

Frame 1 & Frame 2 & Frame 3

Матчим точки, находим R, T с помощью PnP, триангулируем 3D точки

OpenCV

cv2.findEssentialMat(...) – найти существенную матрицу

cv2.recoverPose(...) – разложить существенную матрицу в R и T

cv2.solveP3P(...) – получить R и T после решения P3P

cv2.solvePnP(...) – получить R и T после решения PnP

cv2.triangulatePoints(...) – триангуляция точек

Библиотеки

Линейная алгебра

Eigen

SE(3)

Sophus

manif

Алгоритмы реконструкции

OpenCV, tutorials

OpenGV

Использованные материалы

- Материалы курса 3DCV от DeepSchool,
  автор презентации – Игорь Ильин, разработчик SLAM
- Документация OpenCV
- Материалы курса Computer Vision, Carnegie Mellon

Примеры работ

- A monocular SLAM system based on SIFT features for gastroscope tracking
- Real time SLAM in Endoscopy Applications
- KDD 2019: SLAM Endoscopy Enhanced by Adversarial Depth Prediction - Reconstruction of Colon Phantom
- EndoSLAM dataset and an unsupervised monocular visual odometry and depth estimation approach for endoscopic videos
- CudaSIFT-SLAM: multiple-map visual SLAM for full procedure mapping in real human endoscopy