# Visual Odometry

Course 3, Module 2, Lesson 5

UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE & ENGINEERING

# Learning Objectives

- Learn why visual odometry is useful for self-driving cars
- Learn how to perform visual odometry using image features in consecutive frames, along with their 3D position in the world coordinate frame

# Visual Odometry

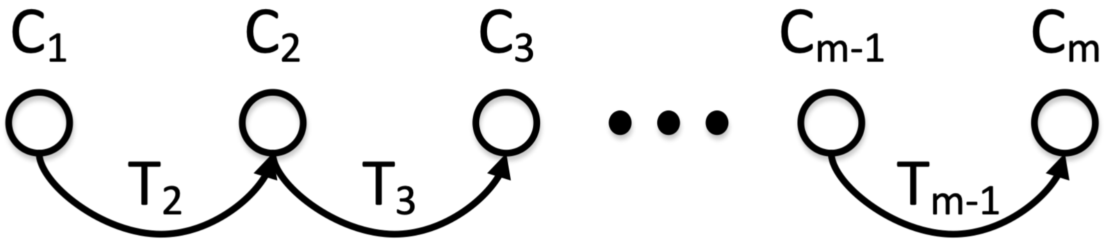
- **Visual Odometry (VO):** is the process of incrementally estimating the pose of the vehicle by examining the changes that motion induces on the images of its onboard cameras
- **VO Pros:**
  - Not affected by wheel slip in uneven terrain, rainy/snowy weather, or other adverse conditions.
  - More accurate trajectory estimates compared to wheel odometry.
- **VO Cons:**
  - Usually need an external sensor to estimate **absolute scale**
  - Camera is a **passive sensor**, might not be very robust against weather conditions and illumination changes
  - Any form of odometry (incremental state estimation) drifts over time, as seen in Course 2

# Problem Formulation:

- Estimate the camera motion $T_k$ between consecutive images $I_{k-1}$and $I_k$

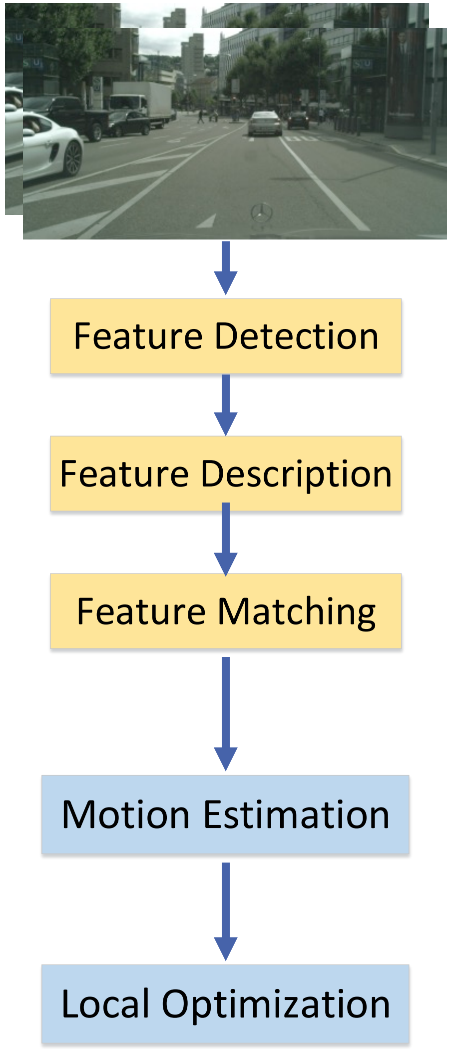
$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix}$$

- Concatenating these single movements allows the recovery of the full trajectory of the camera, given frames $C_1, \dots \dots, C_m$



# Visual Odometry:

- **Given:** two consecutive image frames $\boldsymbol{I_{k-1}}$ and $\boldsymbol{I_k}$
- **Extract and match features** $\boldsymbol{f_{k-1}}$ and $\boldsymbol{f_k}$ between two frames $\boldsymbol{I_{k-1}}$ and $\boldsymbol{I_k}$
- **Estimate motion** between frames to get $\boldsymbol{T_k}$
- **Local optimization** within multiple frame pairs (optional)




$T_n = \begin{bmatrix} R_n & t_n \\ 0 & 1 \end{bmatrix}$

n 0 1 2 3 . . .

same P: $(P)_n = T_n (P)_{n-1}$ wrt frame n-1

center of camera at 1: $C_1$

$(C_1)_1 = T_1 (C_1)_0$  current $C_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

$(C_1)_0 = T_1^{-1} (C_1)_1$ w.r.t frame 1

# Motion Estimation

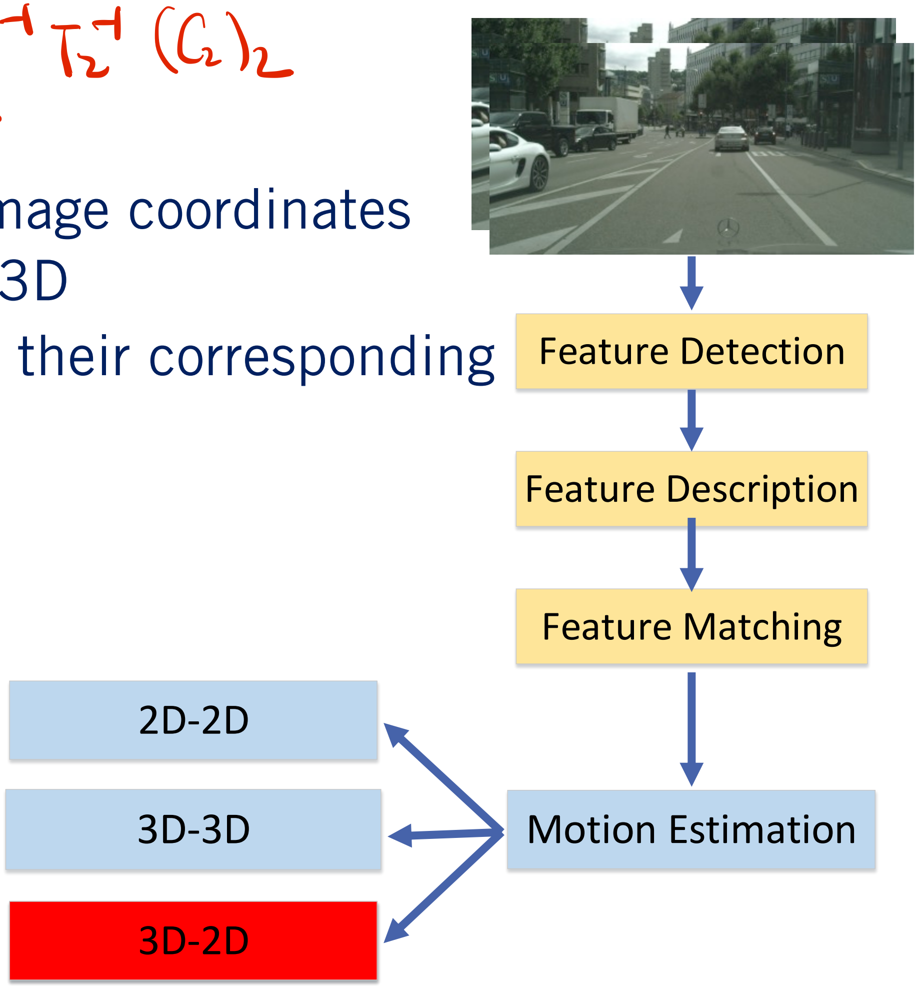
current : frame 2 $\quad C_2 = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$

$(C_2)_2 = T_2 (C_2)_1 = T_2 T_1 (C_2)_0$

$(C_2)_0 = T_1^{-1} T_2^{-1} (C_2)_2$

...

- **Correspondence types:**
  - **2D-2D:** both $\boldsymbol{f}_{k-1}$ and $\boldsymbol{f}_k$ are defined in Image coordinates
  - **3D-3D:** both $\boldsymbol{f}_{k-1}$ and $\boldsymbol{f}_k$ are specified in 3D
  - **3D-2D:** $\boldsymbol{f}_{k-1}$ is specified in 3D and $\boldsymbol{f}_k$ are their corresponding projection on 2D



# 3D – 2D motion estimation

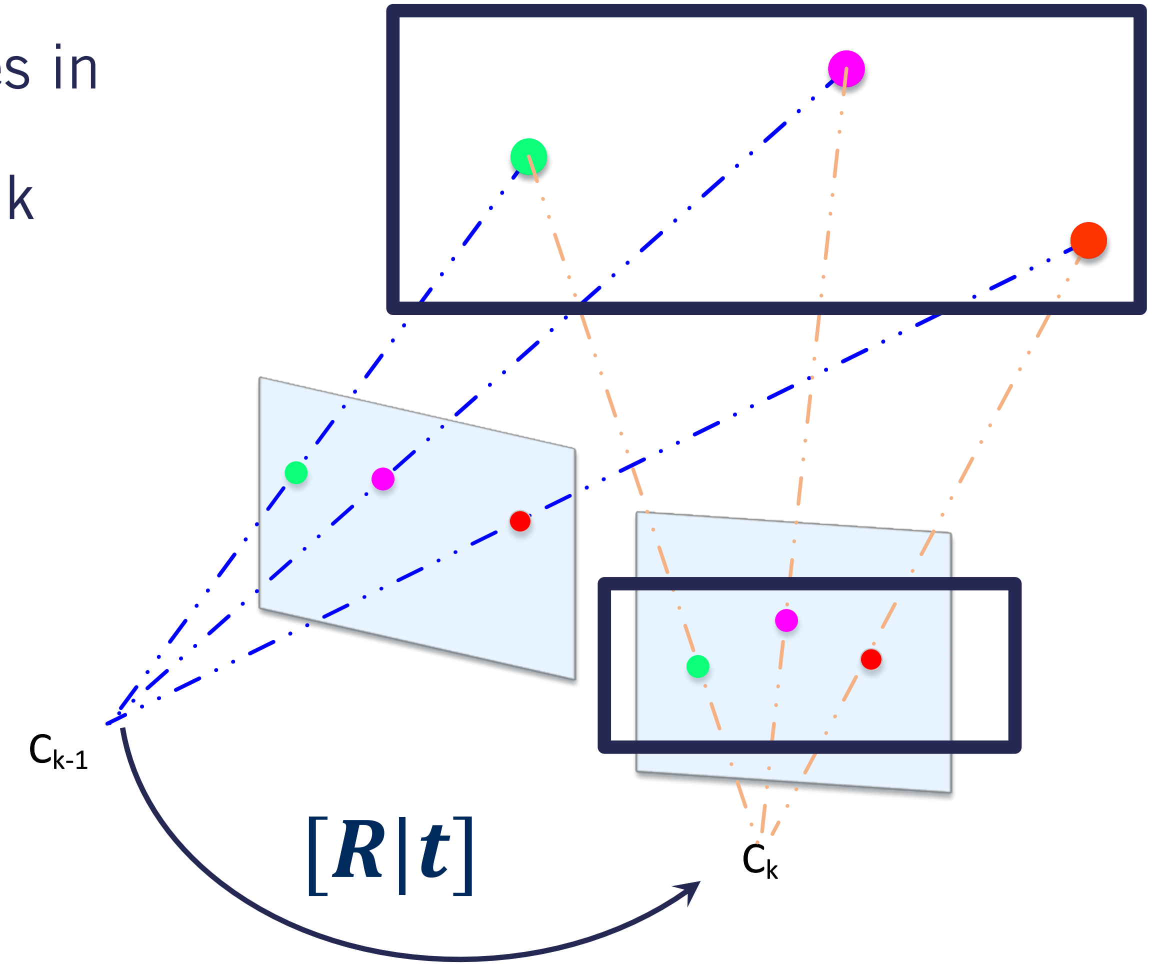
- **Given**:
  - 3D world coordinates of features in frame k-1
  - 2D image coordinates in frame k

feature matching

- **Camera Projection:**

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = K[R|t] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- $K$ is **known** from calibration
- Estimate $[\boldsymbol{R}|\boldsymbol{t}]$



# Perspective N Point (PNP)

- **Camera Projection:**

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = K[R|t]\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \forall f_i$$

- **PnP:**
  - Solve for initial guess of $[\boldsymbol{R}|\boldsymbol{t}]$ using **Direct Linear Transform (DLT)**
    - Forms a linear model and solves for $[\boldsymbol{R}|\boldsymbol{t}]$, with methods such as singular value decomposition (SVD)
  - Improve solution using **Levenberg-Marquardt algorithm (LM)**
  - Need at **least 3 points to solve (P3P)**, 4 if we don't want ambiguous solutions. However, the more features we have, the better!
  - Use **RANSAC** if needed to handle outliers

# Perspective N Point (PNP)

- **OpenCV** has an implementation of the PnP algorithm !

- **cv2.solvePnP():** Solves for camera position given 3D points in frame k-1, their projection in frame k, and the camera intrinsic calibration matrix

- **cv2.solvePnPRansac():** Same as above, but uses RANSAC to handle outliers

# Summary

- Visual odometry can be used to provide accurate trajectory estimate for a self-driving car without suffering from wheel slipping effects due to adverse weather conditions

- Visual odometry can be performed using 2D-3D feature correspondences and the PnP algorithm

- **Next: Deep Neural Networks**