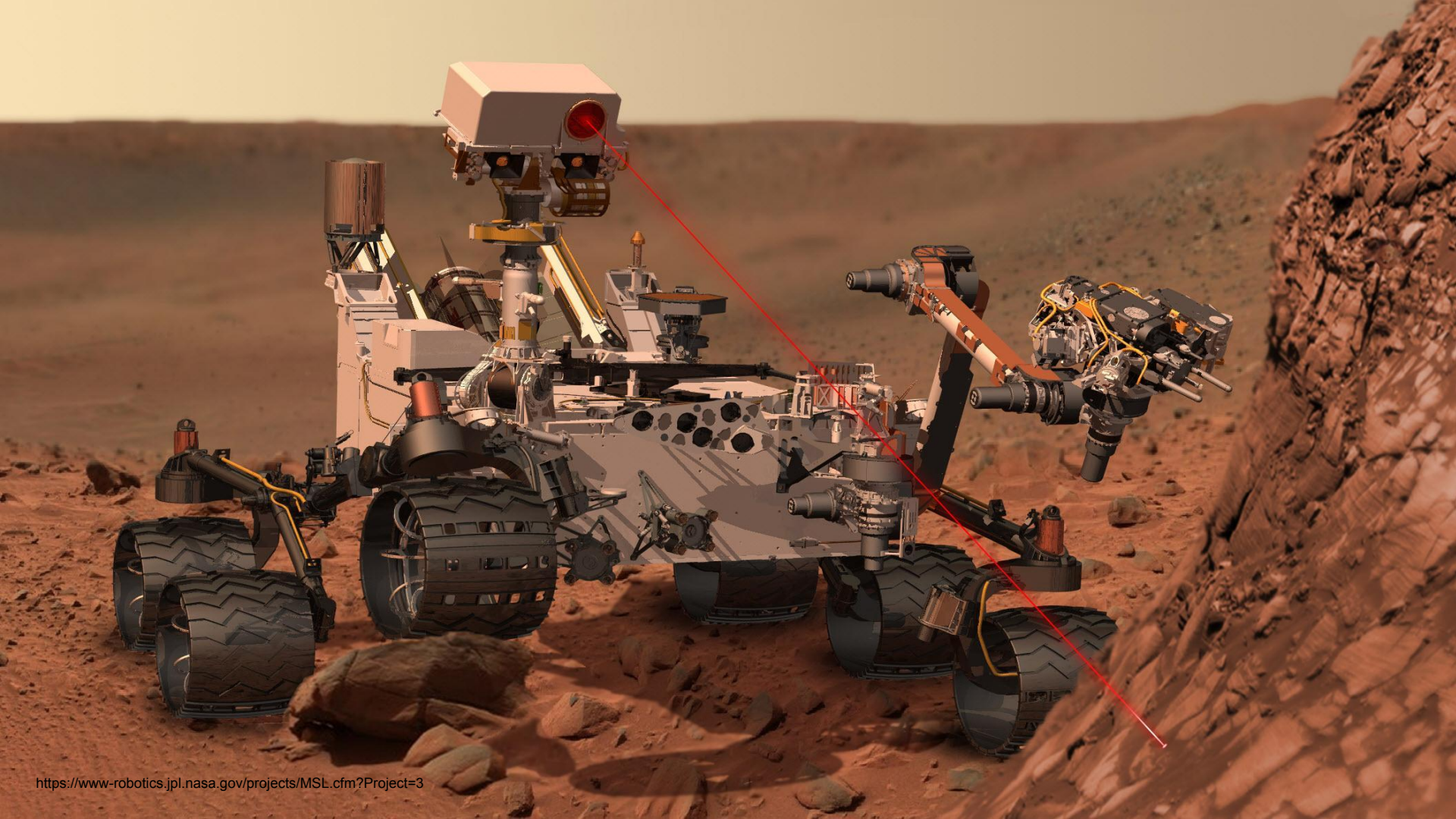
Visual Odometry

Akshay Iyer
Akshay Katpatal
Amey Kulkarni
Bryan Rathos

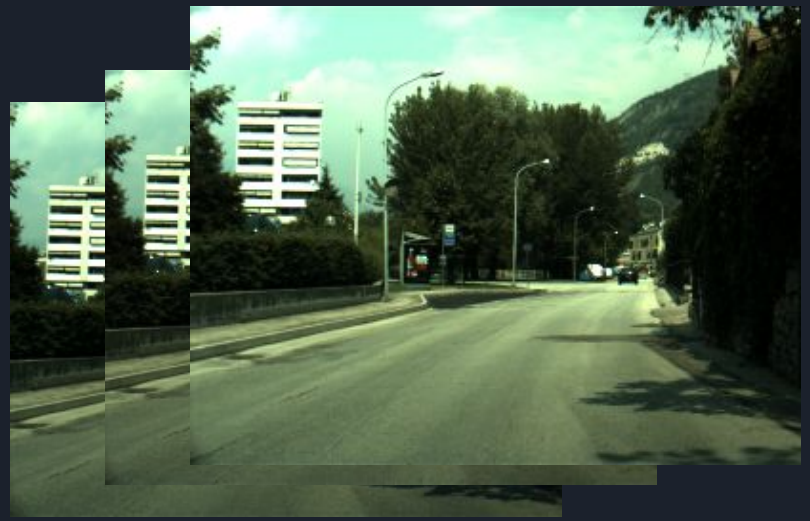


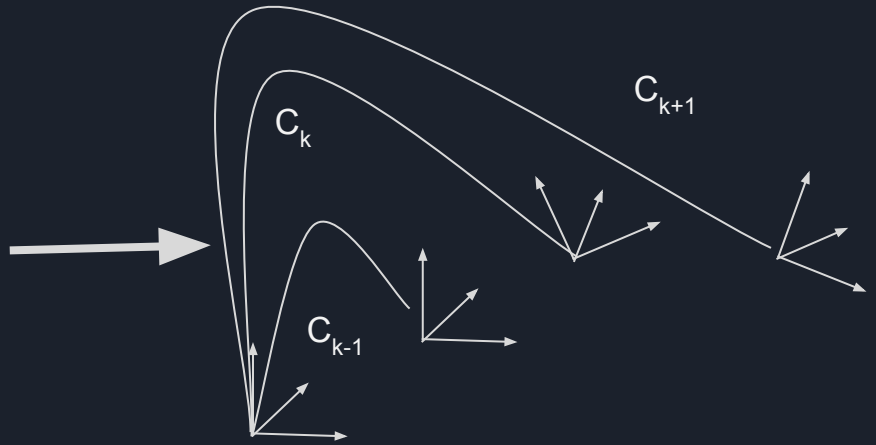
https://www-robotics.jpl.nasa.gov/projects/MSL.cfm?Project=3

What is Visual Odometry?

Visual Odometry

Incrementally estimating the pose of the vehicle using the images obtained from the onboard cameras





Courtesy : https://tev.fbk.eu/databases/diplodoc-road-stereo-sequence

Why Visual Odometry?

GPS

Not always available

- **Indoors**
- **Unknown environments - Mars**

Wheel Odometry

- **Skiddy surfaces / wheel slips**

Inertial Measurement Unit

- **Costly**
- **Noisy**

Why Visual Odometry?

GPS

Not always available

- Indoors
- Unknown environments - Mars

Wheel Odometry

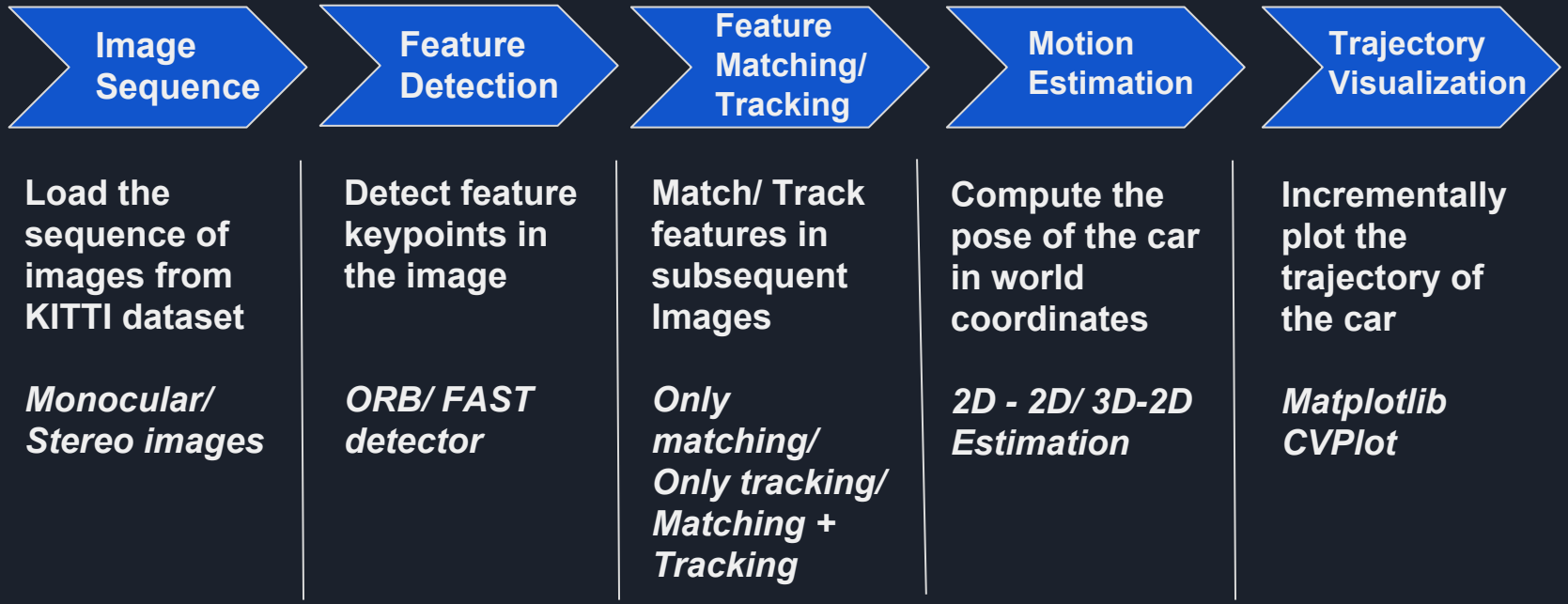
- Skiddy surfaces / wheel slips

Inertial Measurement Unit

- Costly
- Noisy

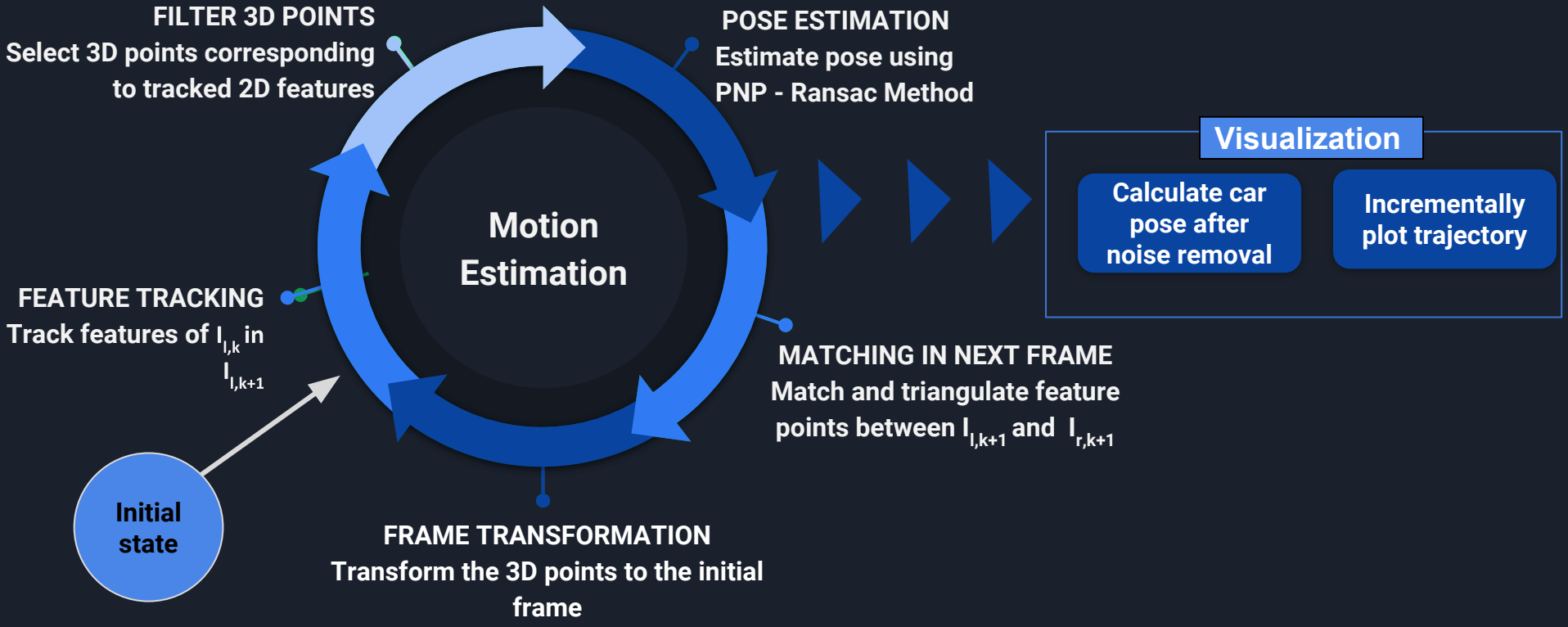
Fun

How do we go about Visual Odometry?

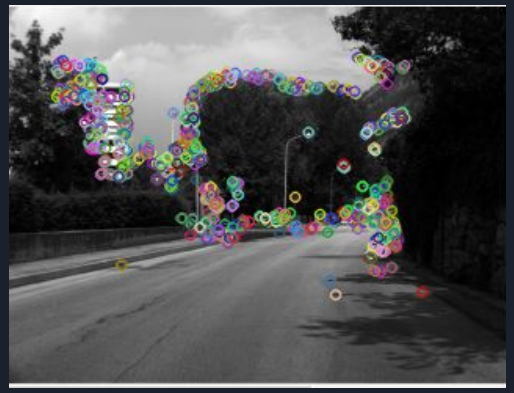


Approach 1 -
Stereo Visual Odometry

Visual Odometry from 3D to 2D correspondences



Feature detection and Matching - ORB and Brute Force

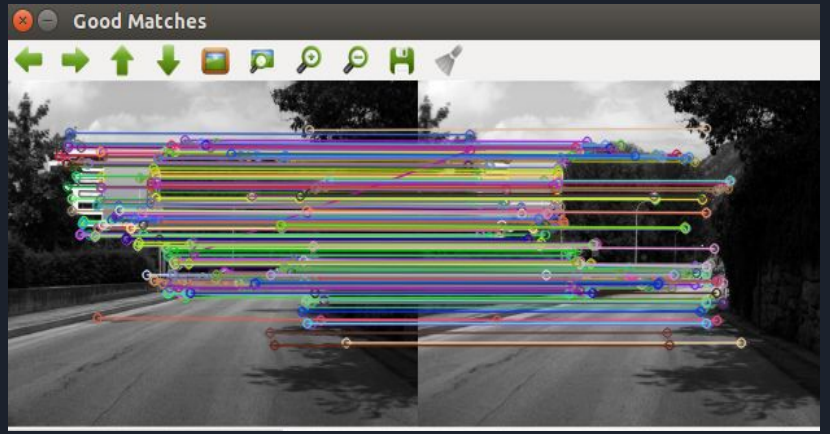


ORB - Oriented Fast and Rotated Brief

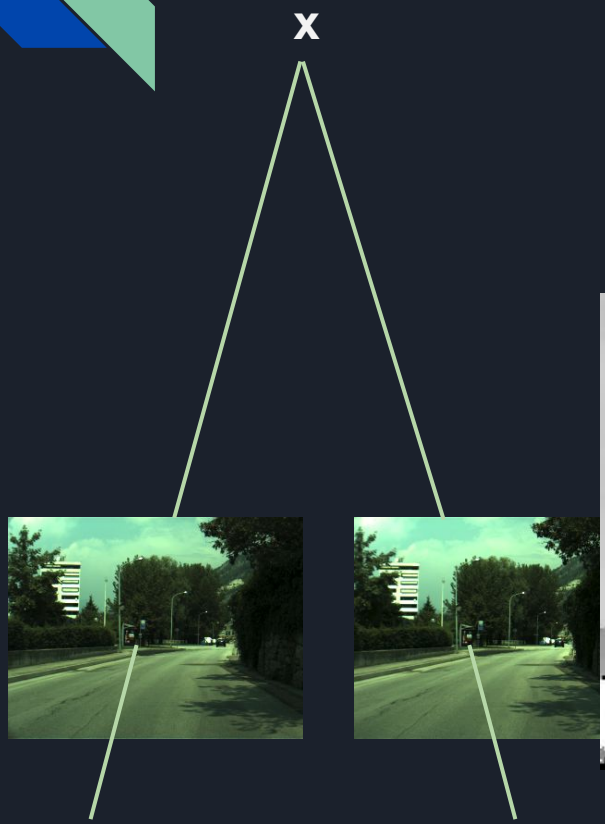
- Fast | Better performance | Open source
- FAST detector ---> keypoints
- Harris detector ---> top keypoints
- Vector from keypoint to centroid of patch ---> direction (solves rotation invariance)
- Orientation of patch ---> rotate the descriptor matrix

Brute Force Matcher

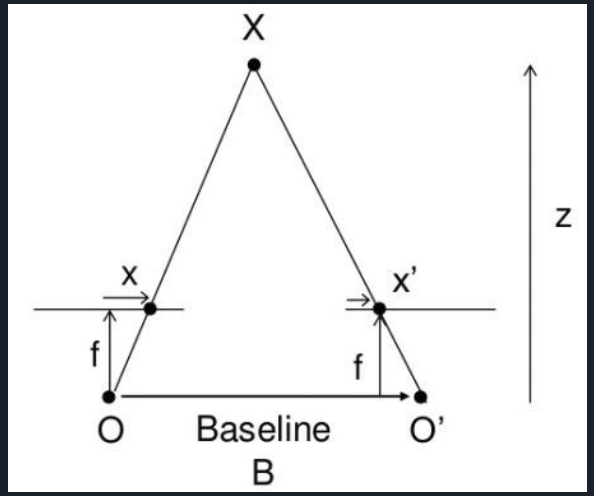
- Compare descriptor of one feature with all other feature descriptors
- Return K Nearest Neighbors
- Apply distance threshold ---> choose best among K matches



Triangulation and 3D reprojection

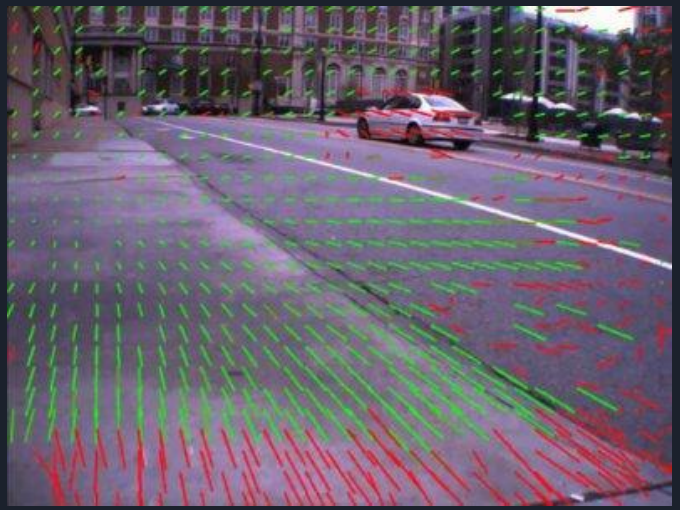


- To determine the 3D coordinates of the matched 2D points
- First disparity is calculated
- Depth is estimated from simple triangle similarity.

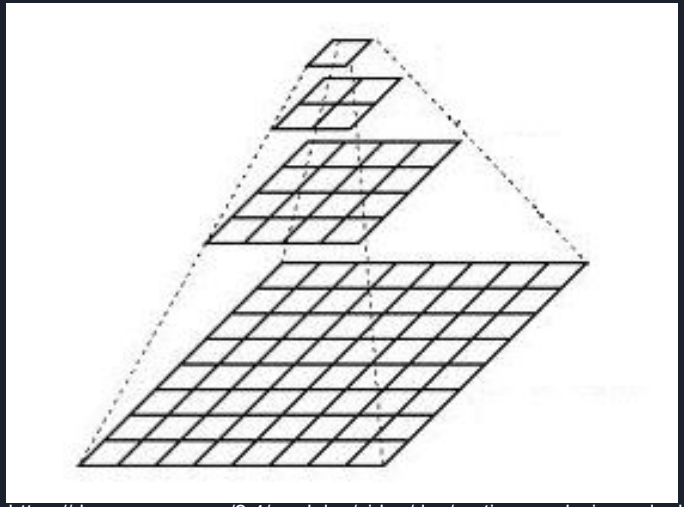


Lucas-Kanade(LK) Optical flow tracking

- Brightness of the feature point is tracked in the next instance frame
- A window is chosen centered at the feature point
- All neighbors in the window undergo the same motion which is assumed to be very small
- System of equations are solved to get the velocity of feature
- Pyramids are used for tracking large motions-Makes tracking robust



http://www.hizook.com/blog/2010/02/16/learning-estimate-robot-motion-and-find-unexpected-objects-optical-flow



https://docs.opencv.org/2.4/modules/video/doc/motion_analysis_and_object_tracking.html

Perspective from N Points(PNP)+RANSAC

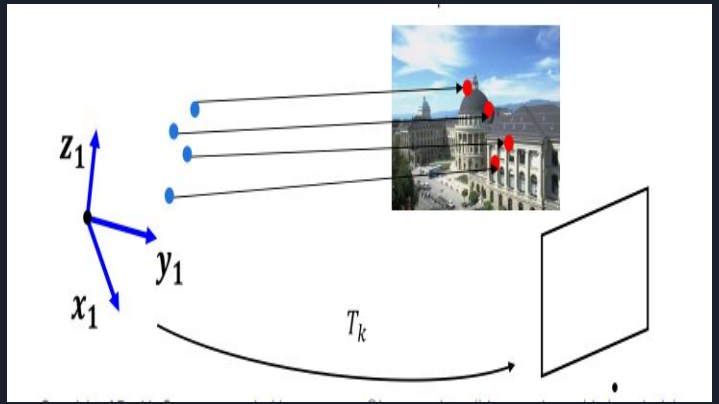
- To determine the pose of the camera

$$sp_c = K\,[R|T]\,p_w$$

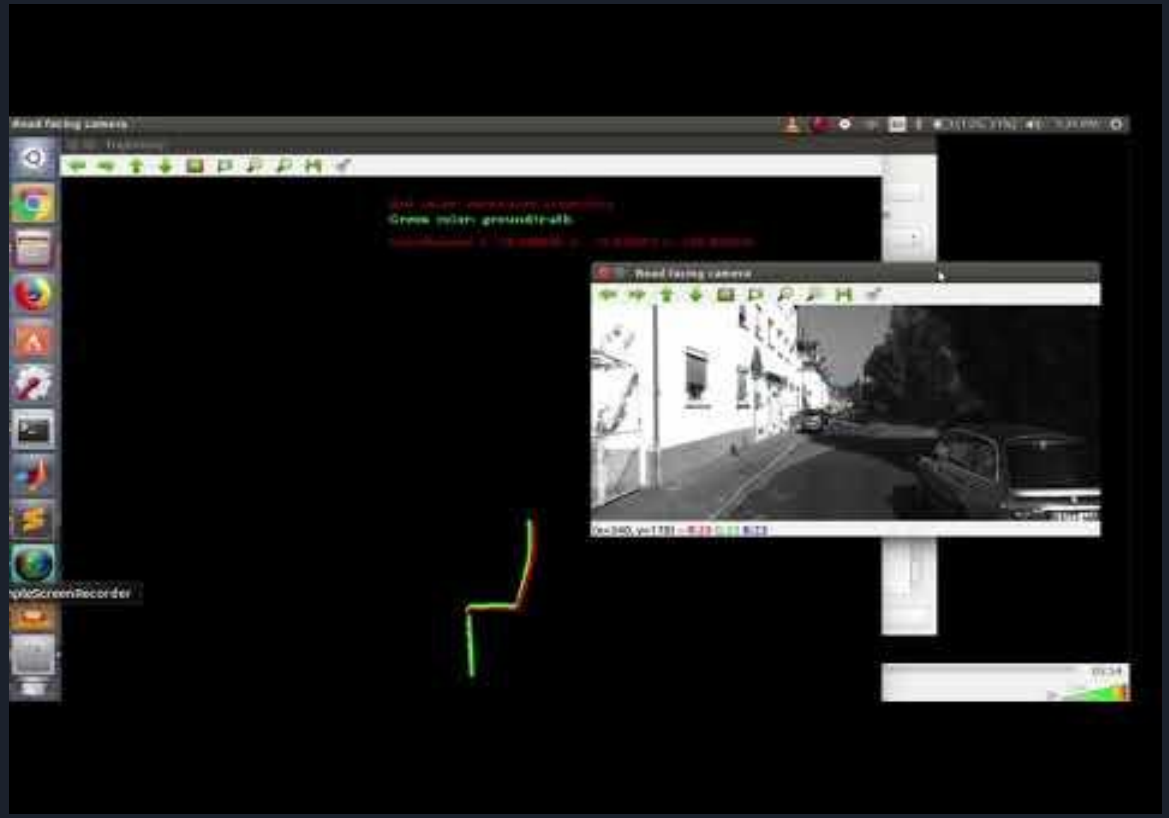
- PNP results noisy- RANSAC!!

- RANSAC removes outliers and fit a model on inliers

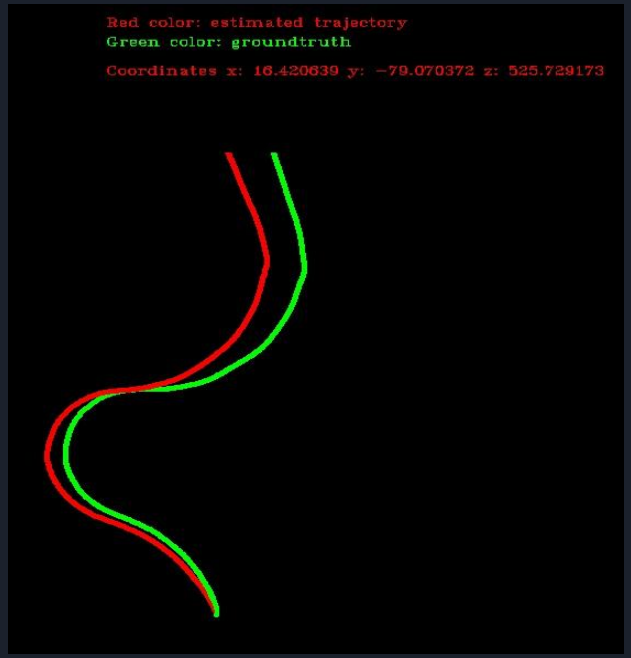
- Best model is selected after numerous iterations

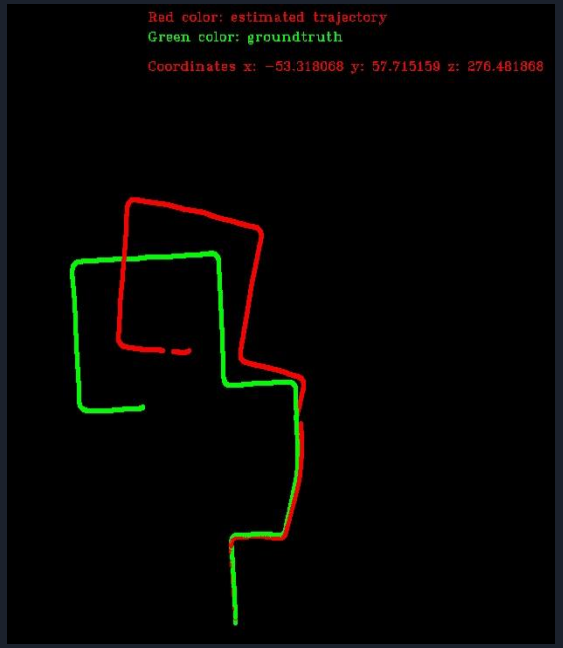


Enjoy the show...!



Some more trajectories





Approach 2 -
Monocular Visual Odometry

Visual Odometry from 2D to 2D correspondences

FEATURE TRACKING
Track features using Lucas Kanade method

POSE ESTIMATION
Recover Pose from Essential Matrix

Motion Estimation

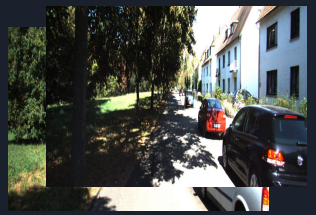
FEATURE DETECTION
FAST algorithm

SCALE COMPENSATION
Calculate Absolute Scale

Visualization

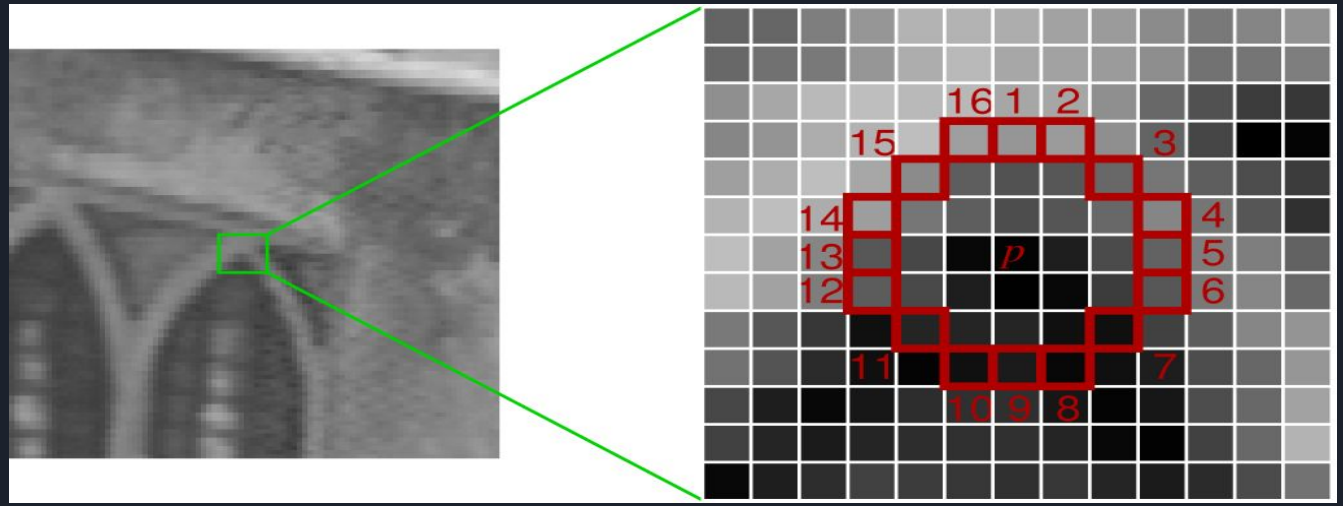
- Calculate car pose in world frame
- Incrementally plot trajectory

FRAME TRANSFORMATION
FInd Pose with respect to world frame



Feature Detection using FAST

- FAST stands for Features from Accelerated Segment Test.
- FAST was chosen as it is suitable for real-time applications due to less computation time.
- A circle circumference containing 16 pixels around candidate point P. (I_p is pixel intensity)
- Pixel P is considered a corner if a set of “N” contiguous pixels in the circle which are all brighter than (I_p + T) , or all darker than (I_p - T).
- High speed test. 1,9,5,13. Three candidates to determine if corner. Full segment test for passed candidates.



High performance detector but with some weaknesses

1. For N < 12, it does not reject many candidates.
2. Pixel choices aren't optimal since efficiency depends on ordering of the questions and distribution of corner appearances.
3. Results of high-speed test are discarded.
4. Multiple adjacent features.

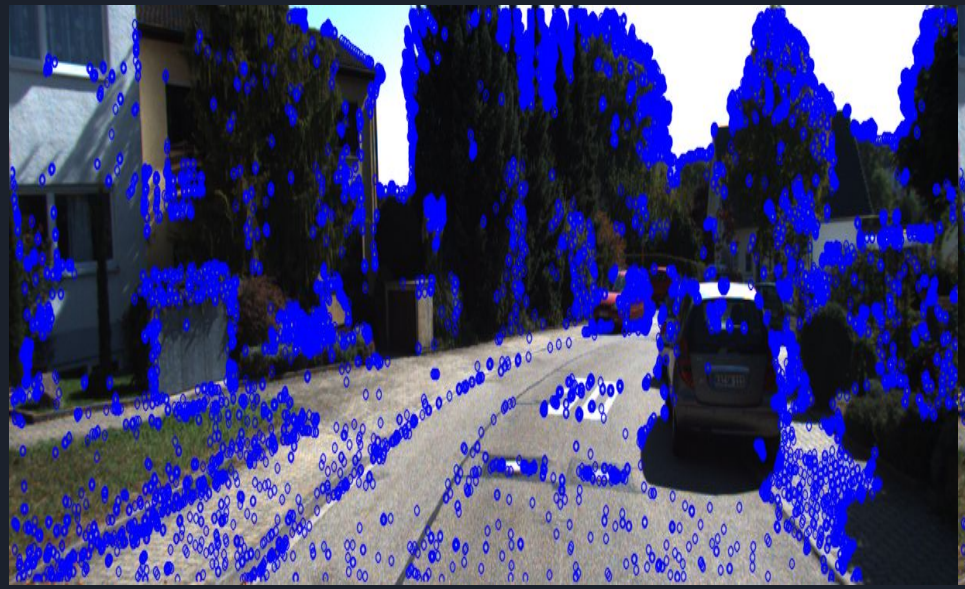
First 3 can be solved by a machine learning approach and last is solved by Non-Maximal Suppression.

We have eliminated multiple adjacent features using Non-Maximal Suppression technique.

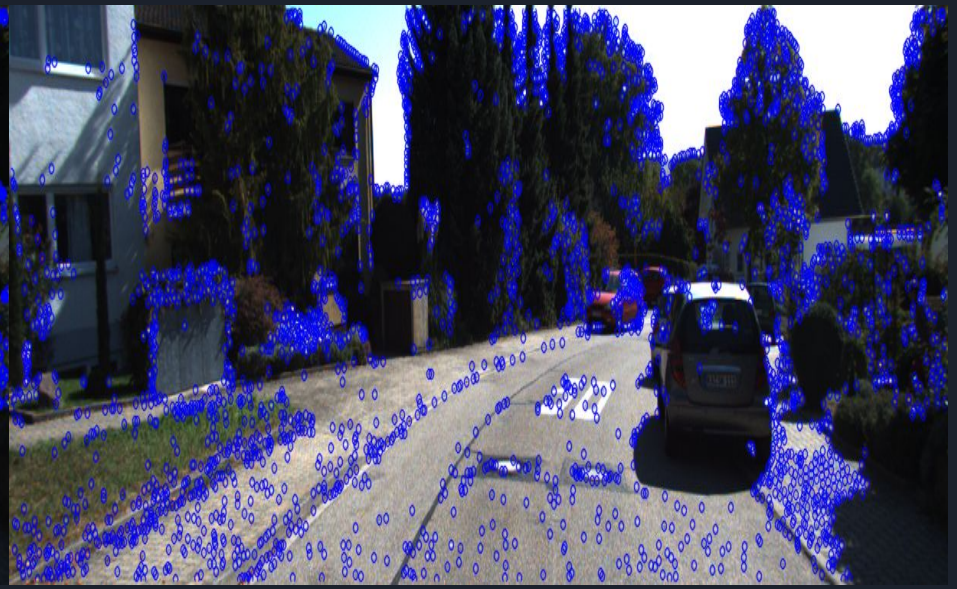
Non Maximal Suppression

- NMS solves the problem of multiple interest point detections in adjacent locations.
- Define a score function “V” for all the detected feature points.
- “V” is the sum of absolute difference between P and 16 pixels. Discard lower score function between two adjacent keypoints.

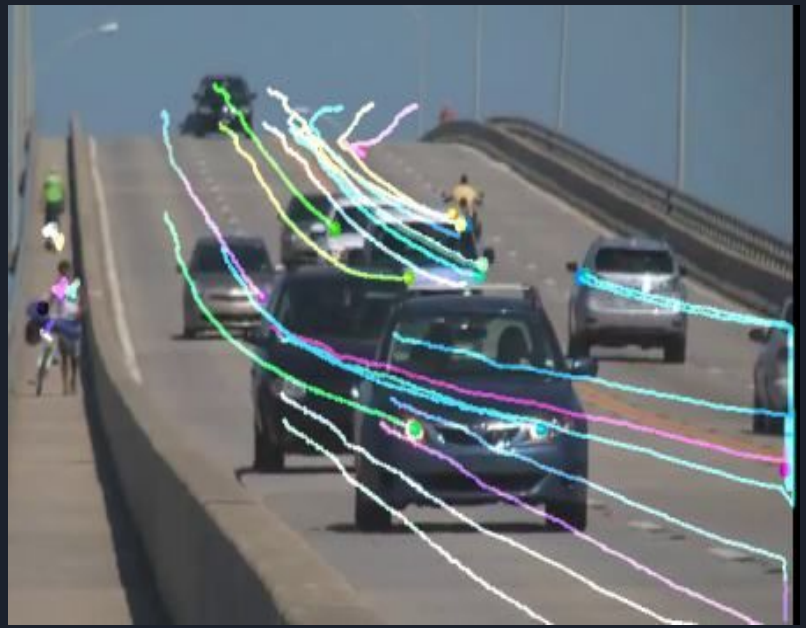
11263 keypoints



3012 keypoints



Feature Tracking



- Feature tracking is done using Lucas Kanade method for Sparse optical flow with pyramids.
- Pyramid structure is used to account for large motions.

Estimating the Essential Matrix

- The main property of 2-D-to 2-D-based motion estimation is the epipolar constraint.
- This constraint can be formulated by

$$(y')^T E y = 0$$

where y’ and y are the normalised image coordinates.

- The geometric relations between two images I_k and I_{k-1} of a calibrated camera are described by the essential matrix E.

- The essential matrix can be computed from 2-D-to-2-D feature correspondences, and rotation and translation can directly be extracted from E.

Finding Outliers

- A standard technique of handling outliers when doing model estimation is RANSAC.
- RANSAC is an iterative algorithm. At every iteration, it randomly samples five points from the set of correspondences, estimates the Essential Matrix, and then checks if the other points are inliers when using this essential matrix.
- The algorithm terminates after a fixed number of iterations, and the Essential matrix with which the maximum number of points agree, is used.

Computing the Rotation and Translation

- E contains the camera motion parameters up to an unknown scale factor for the translation in the following form:

$$E = t_x R$$

Here, t_x = Cross product Representation of the Translation Matrix,

& R= Rotation Matrix

- Now, R and t are calculated by taking the SVD of the essential matrix

$$E = U \sum V^T$$

$$R = UW^{-1}V^T \ \& \ t_x = UW\sum U^T$$

Computing Trajectory

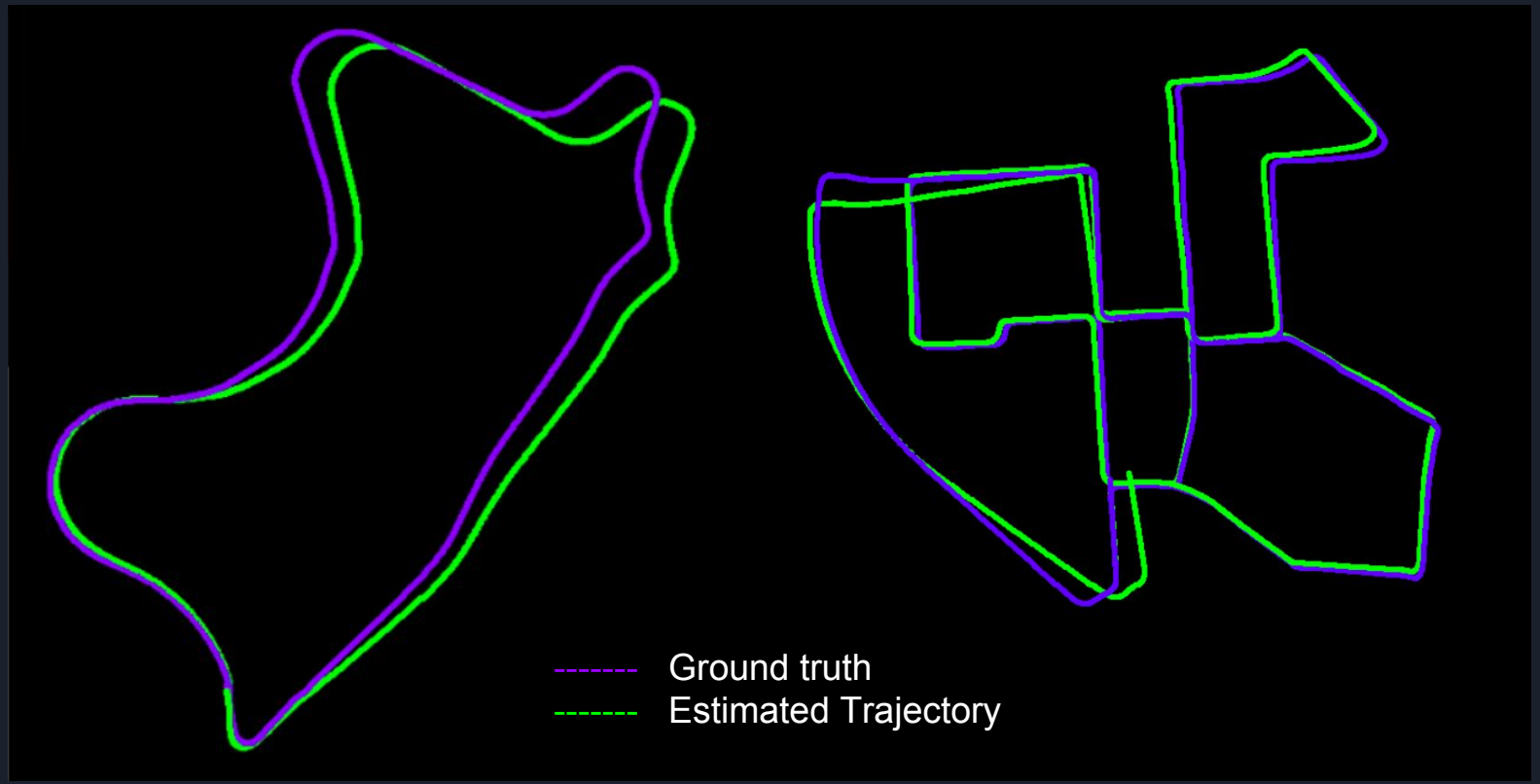
- The pose of the camera be denoted by R_{pos}, t_{pos}.
- The following equation can be used for tracking the trajectory

$$R_{pos} = R R_{pos}$$

$$t_{pos} = t_{pos} + t_x R_{pos}$$

Enjoy the show...!

Some more trajectories!



Way forward...

- **Windowed bundle adjustment**
- **Grid based feature detection to ensure even coverage of feature points**
- **Keyframe selection**
- **Improved outlier removal**
- **Real time tracking window size optimization**
- **Visual Inertial Odometry**

Thank You