ORB SLAM:- → Efficient, Robust, real-time, low-power Hardware.

- ORiented FAST and BRIEF Simultaneous Localization and Mapping.
- Computer Vision + Robotics + Autonomous vehicles.
- Based on two main components:
  - Features detection and matching (ORiented FAST) (Rotated BRIEF)
  - Pose estimation.

ORiented FAST → Used for feature detection

- very efficient algorithm for finding Key Points

Rotated BRIEF →

- Provides binary descriptor for each Key Point
- find correspondences between Key Points in different frames.

Triangularization or iterative optimization methodes (Bundle adjustment)

- these methods are used for Pose estimation using the Information coming from the camera.

Limitations:

- Poor lighting.
- Repetitive texture.

How does its build its map?

→ Initialization:

- Initialize the map with the first frame it recieves
- then it detects the keypoints using Oriented fast algo.
- then it computes the descriptors for these keypoints using Rotated BRIEF.
- these keypoints and descriptors form the initial map.

→ Tracking :-

- As the camera moves :-
  - It keeps tracking features in the current frame and matches them with features in previous frames using their descriptors.
- this tracking process provides information about how the camera has moved since the last frame.

→ Mapping :-

- When a new frame is processed, ORB SLAM triangulate the matched keypoints to estimate the 3D positions of points in the enviroment relative to camera.
- these 3D points along with their descriptors are added to the map.

→ loop closure
it enhances the accuracy by detecting revisited places.

How does our map looks like?

→ the map consists of two main components { Key frames / Map Points

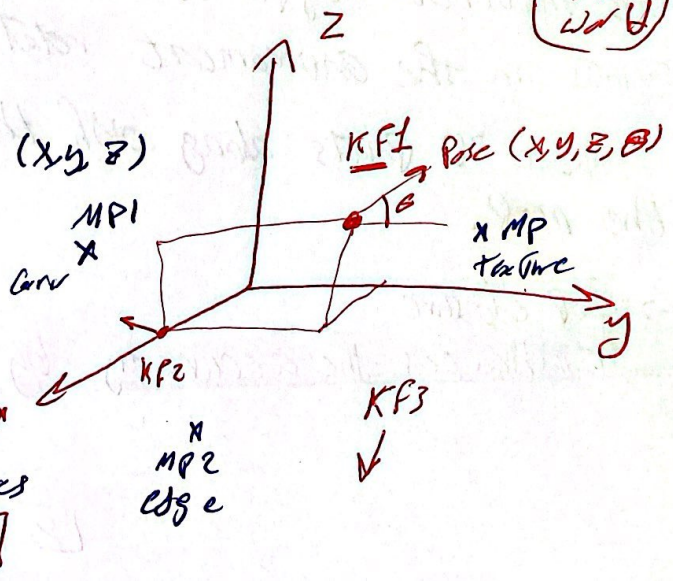
- Key frames :-
  - these are individual frames from the camera feed.
  - the ORB-SLAM selects them as key frames for mapping.
  - they typically represents significant view points
- Each key frame contains information about the camera pose relative to the environment when the frame was captured.

Map Points:

- these are 3D points in the environment.
- observed and triangulated across multiple key frames.
- they represent features or landmarks in the environment.
- ie: walls edges, corners of objects, distinct textures.
- ORB SLAM estimates the 3D positions of these map points relative to the camera
- Each map point is associated with one or more key frames

- Each Key Frame
  - represented by its pose in 3D & how the camera was oriented when the frame was captured

- Map Points
  - points scattered on 3D space represents the locations of the features [3]



- Each map point is associated with the key frames where it is was observed, forming connections or lines between the key frames.

Applying localization:-

- It should be easy task now!
- we need to match the descriptors and map points of the current frame with the map points & key frames in the constructed map.

- Steps:-
  - Feature Detection & Description → ORB
  - Feature Matching
  - Pose estimation
    - ↳ Position & orientation
    - ↳ can be done using approaches such as → Prespective-n-Points (PnP)
      - → Random Sample Consensus (RANSAC)
  - localization update.
  - Repeat

Path planning:-

- this is done after applying localization and constructing the map for the environment.
- localization → provides us with the current state.
- Mapping → contains the desired destination
  - → contains locations of obstacles, land marks & other features.

→ while navigating we must keep ~~stepping~~ obstacle avoidance

→ Usual algorithms for Path planning is A* or Dijkstra

→ there must be a trade off between reaching the goal & avoiding obstacles, this cost function is passed to the Path Planning.

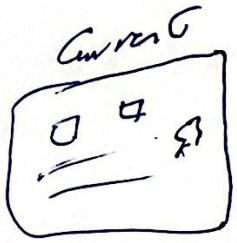
→ we need to take care of the Dynamic environment problem.

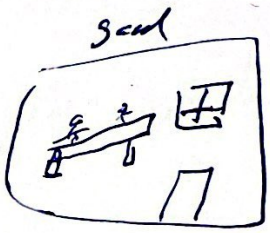
How can we do this when we do not have coordinates?

→ this is one of the most challenging problems, but we can try to solve it using the following steps.

- Graph representation:
  - we represent the environment as a graph.
  - each node corresponds to key frame or map point.
  - the edges represents the connection between them
  - to estimate the distance we need to use a heuristic function, one approach is to use # of key points traversed.
  - or we can estimate the Euclidean distance between the descriptors of the key points in the current frame & the goal

Frame





→ Path search

→ Path execution