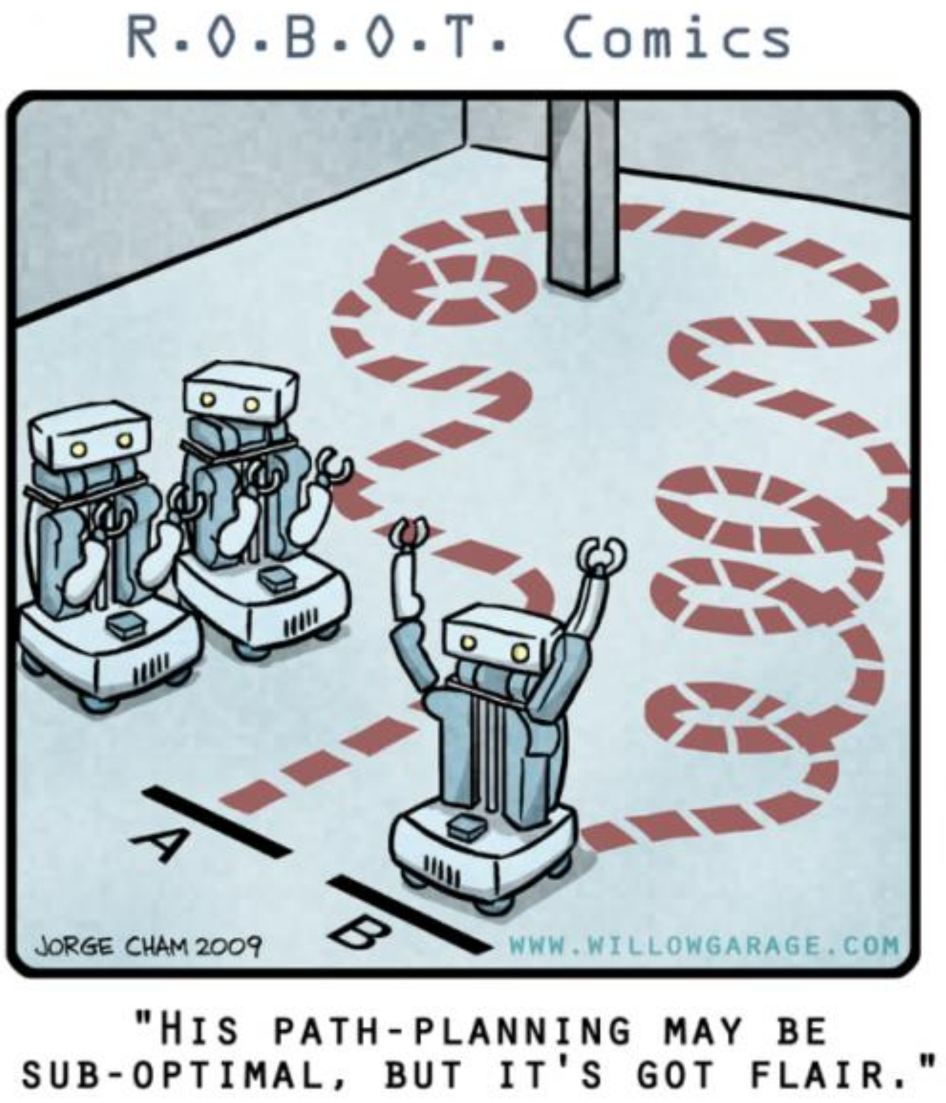
Visual SLAM for Mobile Robotics

Rashik Shrestha
Research Assistant @ NAAMII



Mobile robotics

Refers to the use of robots that can **move** and operate in a variety of environments

Mobile robotics

Mars Rover

Service Robots

Drones

Automatic Vacuum Cleaner

Localization

Process of determining the position and orientation of a robot in its environment

Localization

Crucial for the robot to be able to navigate through the environment and perform tasks effectively

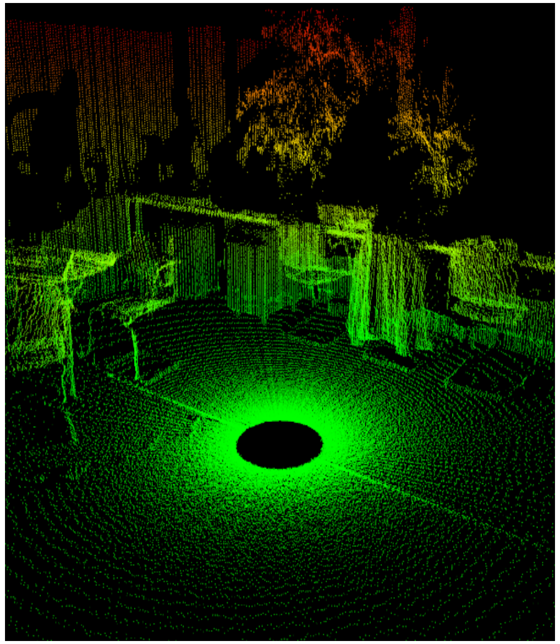
Mapping

Process of creating a map of an environment

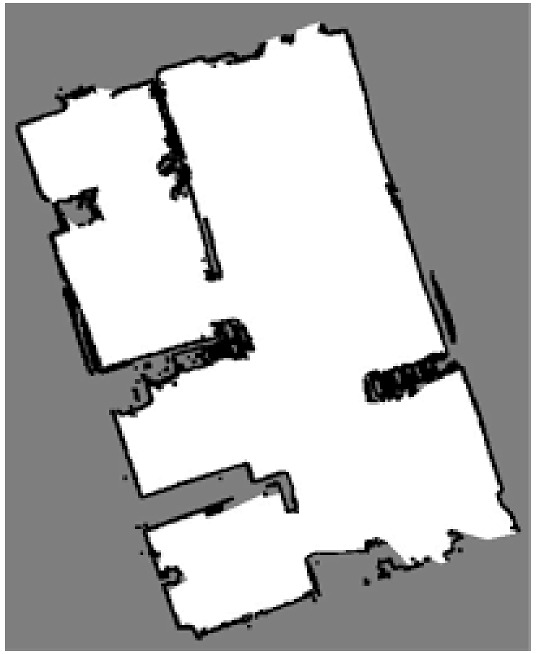
Mapping

In the context of mobile robotics, mapping refers to creating a **"representation"** of the robot's surroundings

Mapping



Point Cloud



Occupancy Grid
(Floor Plan)

3D
reconstruction

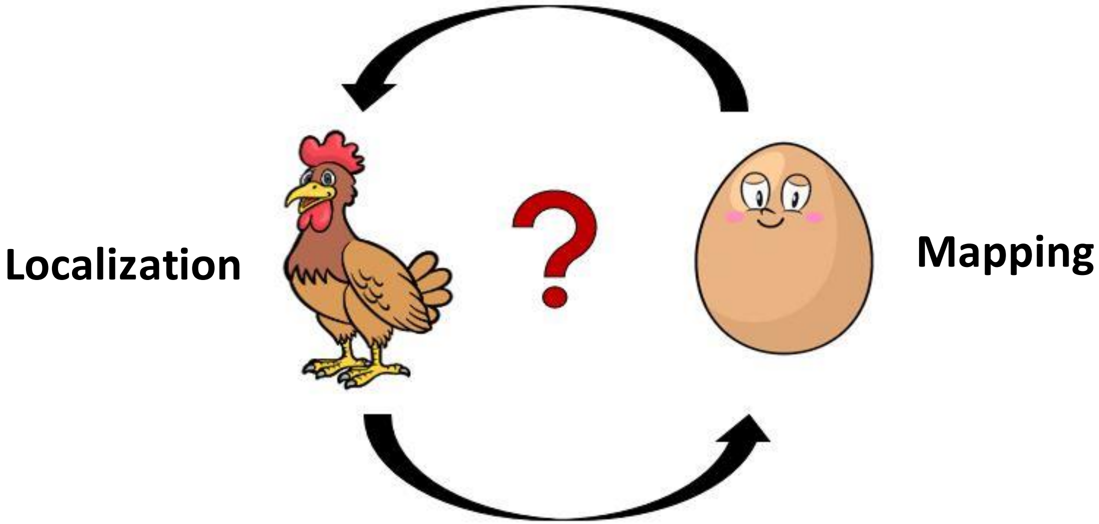
Conundrum

Localization is the process of determining the position within a **KNOWN** environment/map

Robot needs to know its position to accurately align the data it collects and create accurate map

Conundrum

Localization and Mapping are "**Chicken and Egg**" problem



Conundrum

But there is an algorithm which can do both localization and mapping **simultaneously**

Simultaneous Localization and Mapping (SLAM)

Technique used to create a map while simultaneously determining its own position

Sensors used for SLAM

Global Positioning System (GPS) is one of the sensors that can be used for localization

Sensors used for SLAM

Other sensors like camera, lasers, infra-red, ultrasonic can also be used

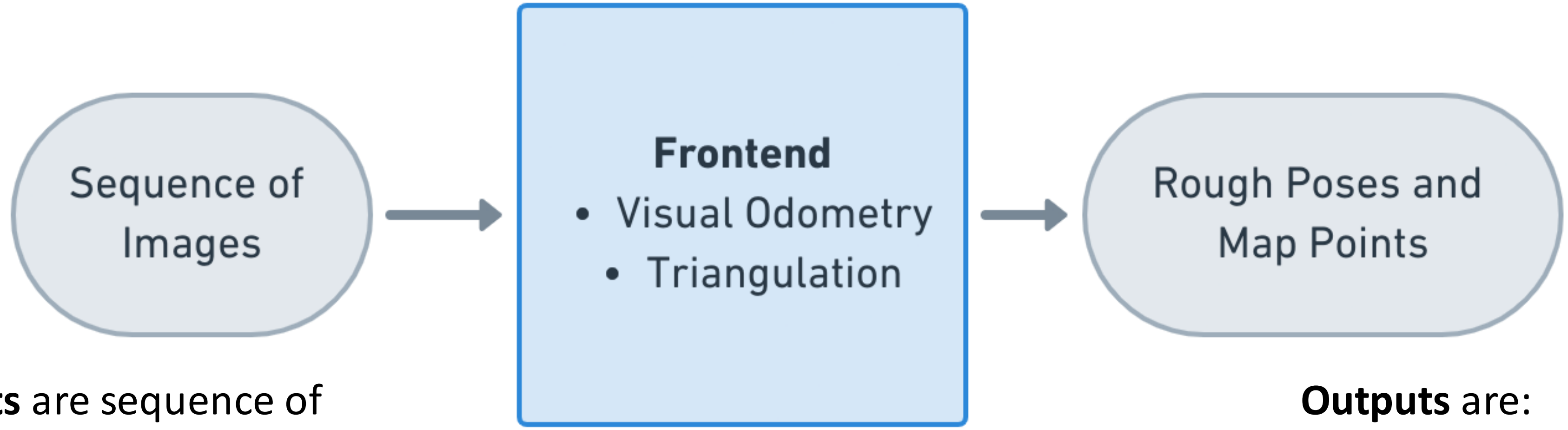
Visual SLAM

SLAM algorithm implemented using **Cameras** as sensor

Visual SLAM

Common process used for Visual SLAM

Frontend



Inputs are sequence of images captured by camera for **visual** SLAM

Outputs are:

1. Pose (position + orientation)
2. Map

Because we are dealing with **SLAM**

Frontend: Visual Odometry

Odometry is a technique used to determine the position and orientation of a robot relative to its starting point

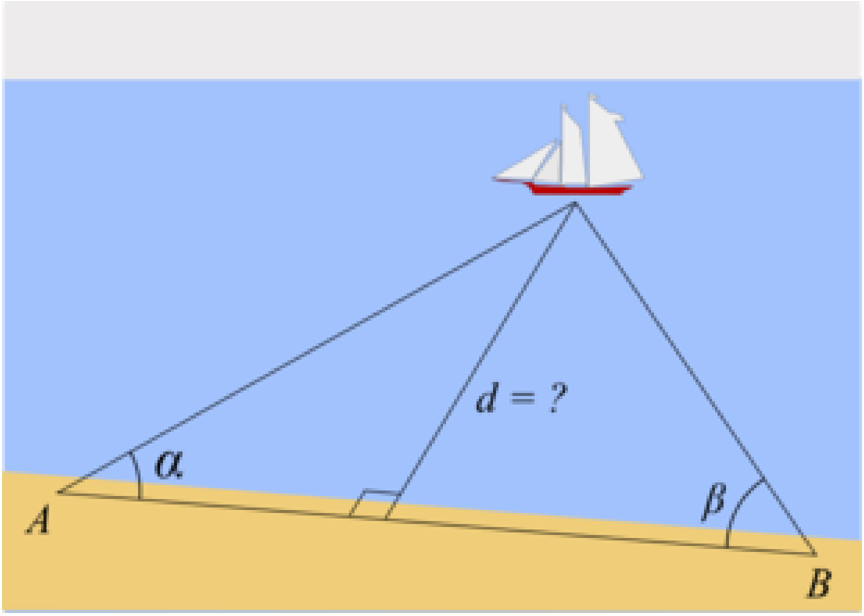
Wheel Odometer used on Vehicles

Frontend: Visual Odometry

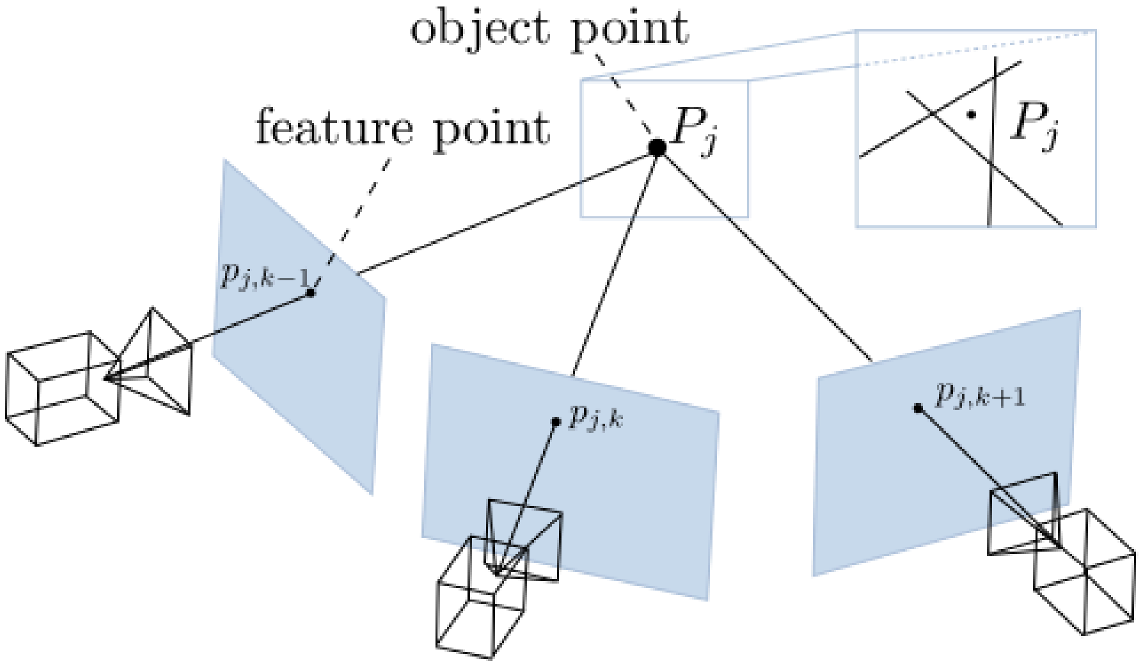
Odometry can also be achieved using Cameras only, termed as Visual Odometry

Frontend: Triangulation

Technique used to determine the position of a point in space relative to two or more other points



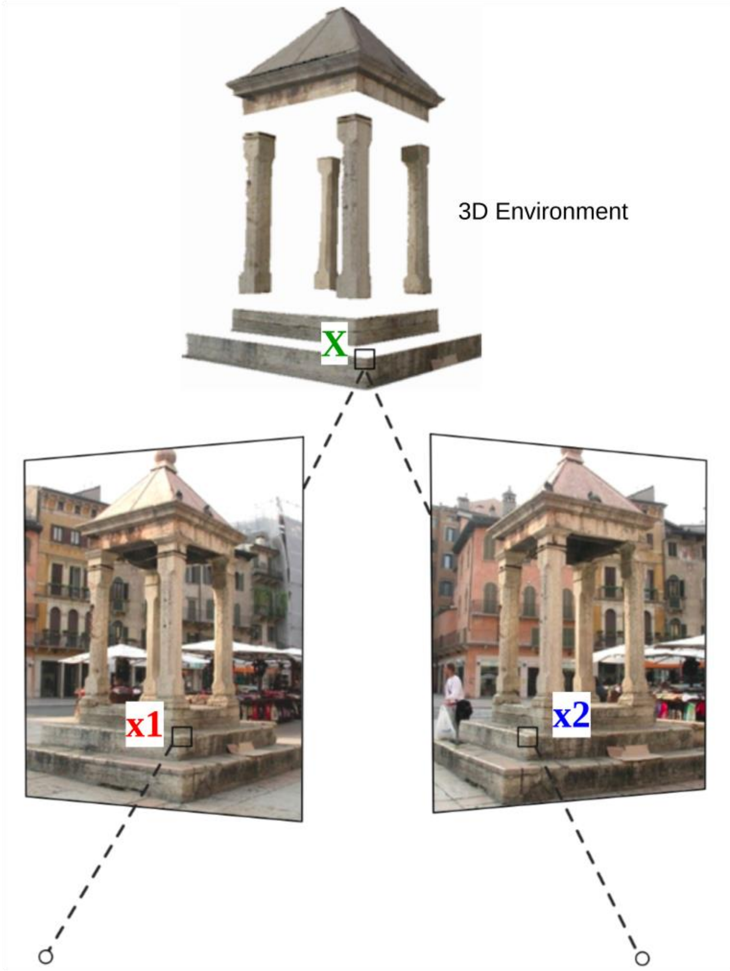
Frontend: Triangulation



Frontend details

Frontend

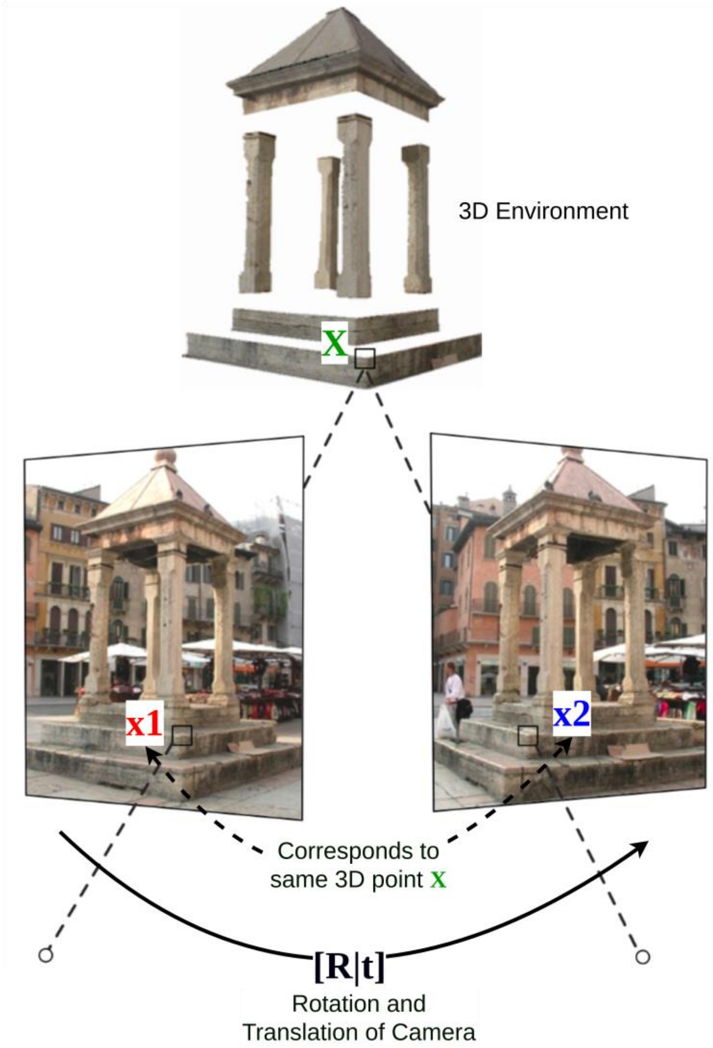
- Visual Odometry
- Triangulation



Frontend details

Frontend

- Visual Odometry
- Triangulation



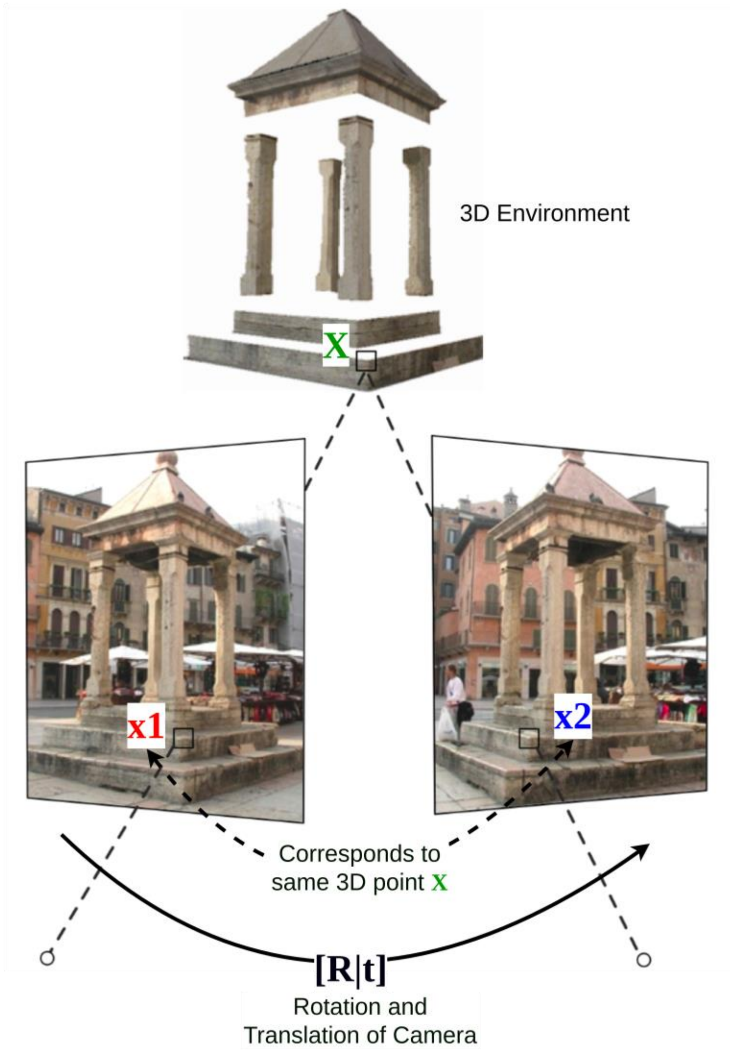
Visual Odometry Problem

X (Map point) known

x1 known

x2 known

Find [R|t]



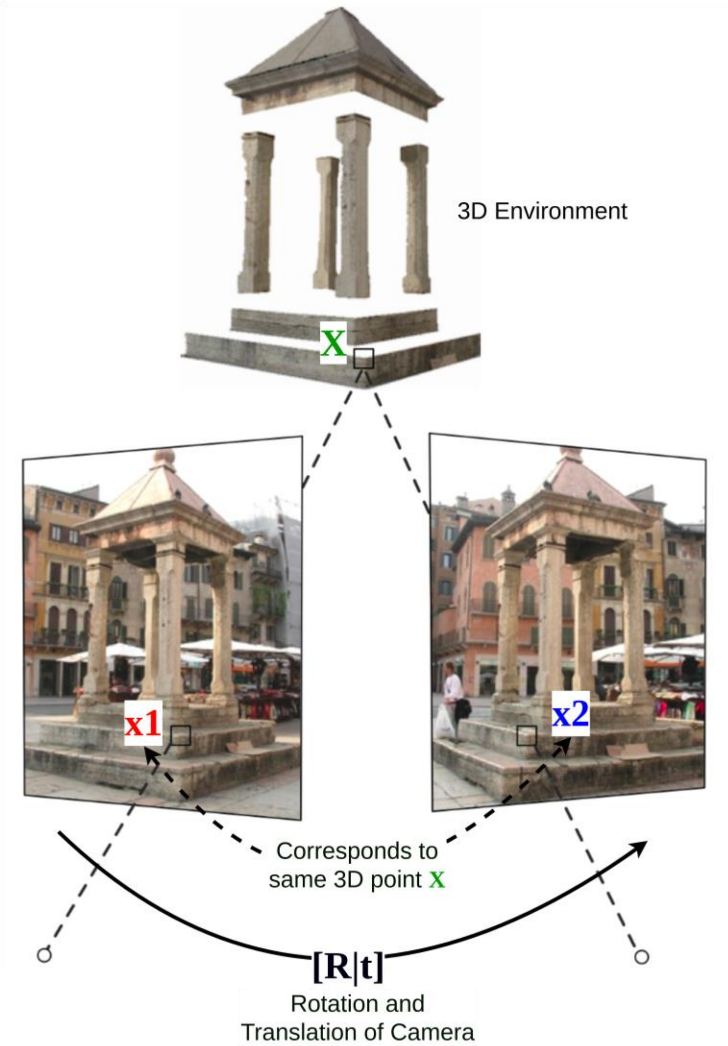
Triangulation Problem

x1 known

x2 known

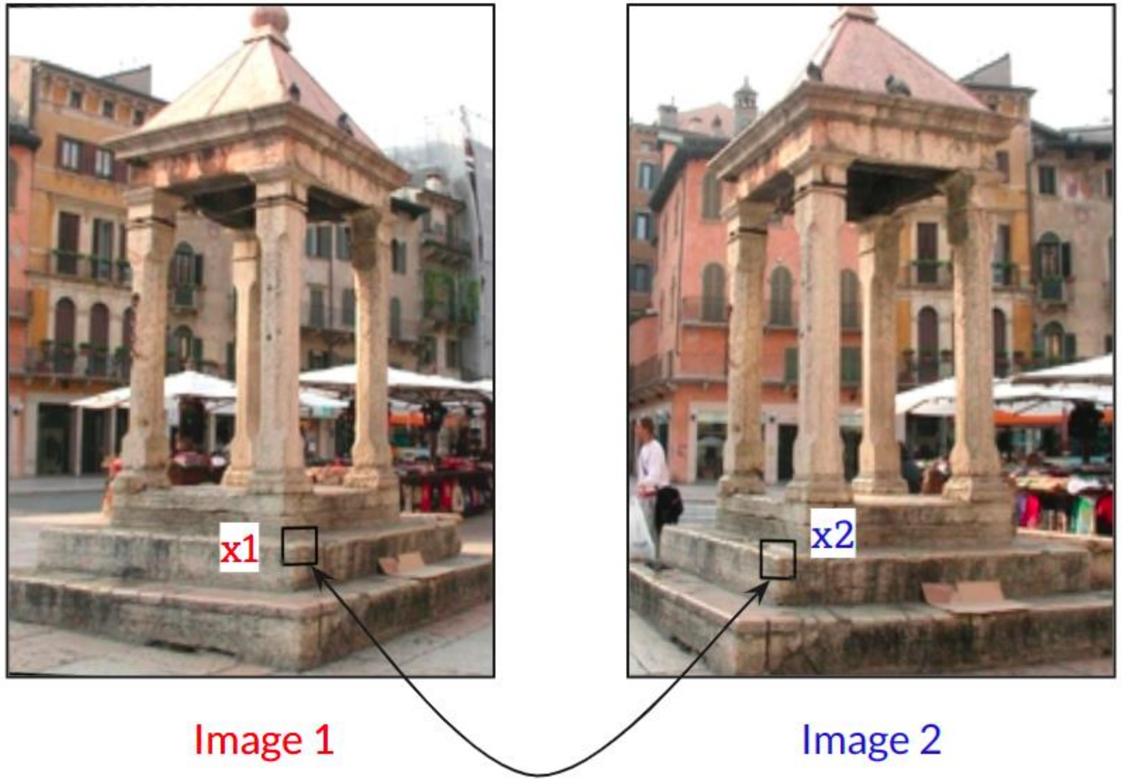
[R|t] known

Find X (Map point)



How to find the corresponding points?

Traditionally this problem has been tackled by a field of study called **"Photogrammetry"**



How do we know if these two points correspond to the same 3D point **?**

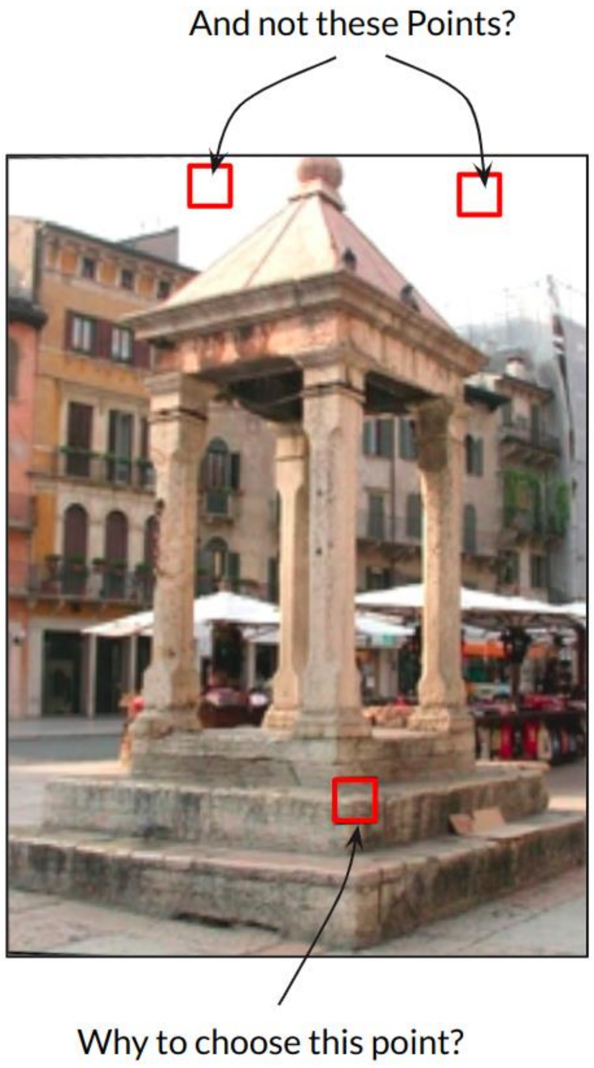
Why we took these specific points **?** There are thousands of other points in the image

Photogrammetry

1. Feature Detection
2. Feature Description
3. Feature Matching

Photogrammetry: Feature Detection

- Impractical to match every points!
- Find distinct points in image
- Called "**Interest Point**" or "**Feature Point**"
- Ought to be unique in across all images
- Methods: FAST, SIFT, SURF, SuperPoint

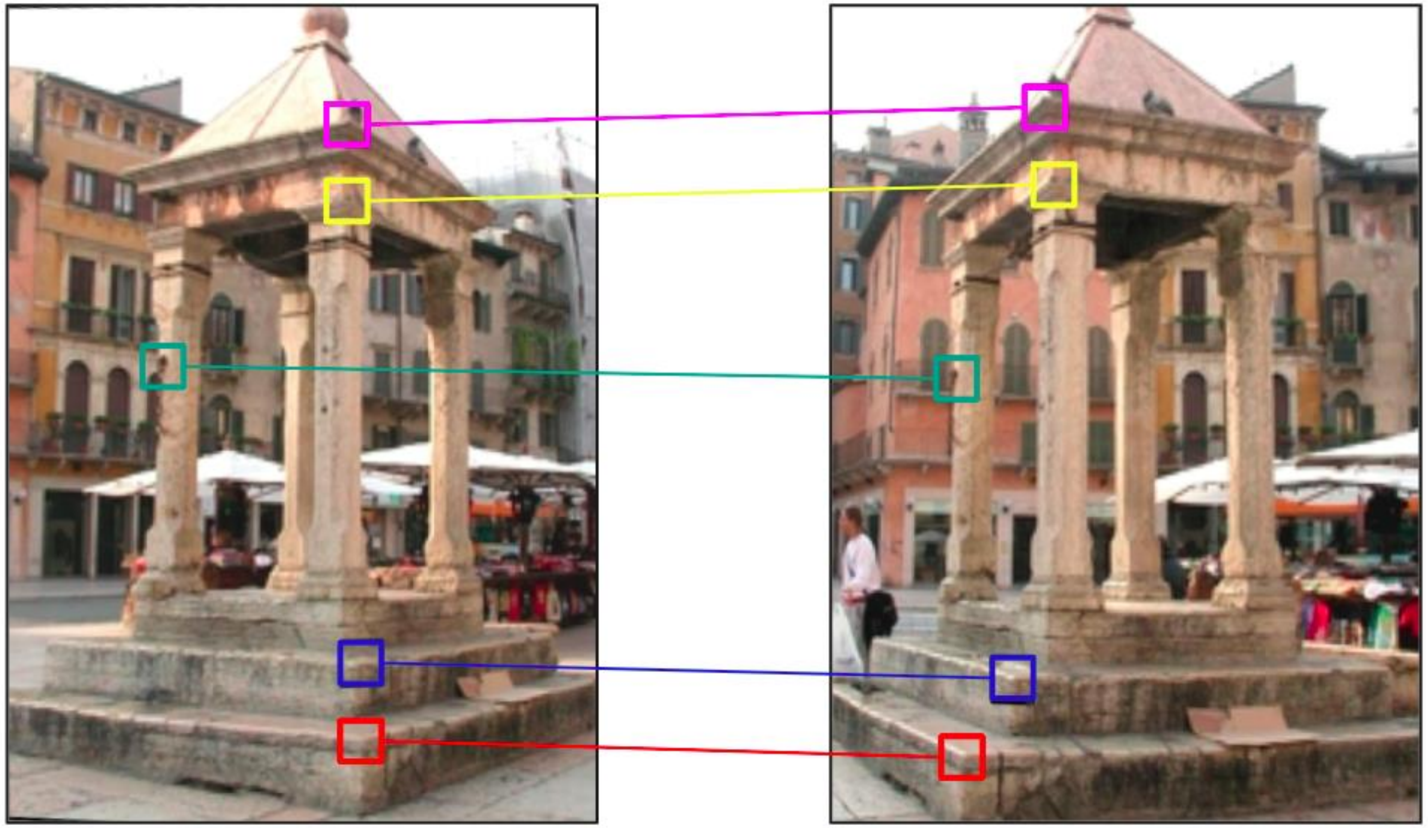


Photogrammetry: Feature Description

- Uniquely Represent each feature point mathematically
- Normally each feature points are represented as a vector.
- Methods: ORB, SIFT, SURF, BRIEF, Superpoint

Different Points should have different Representations

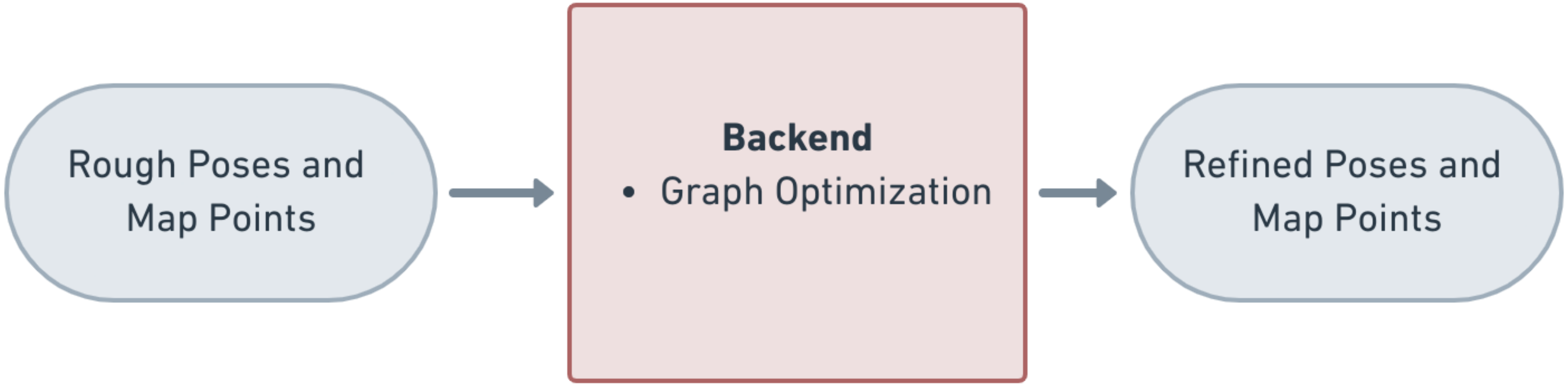
Photogrammetry: Feature Matching



Find the corresponding points based on their descriptors and location

Better feature detection, description and matching can significantly increase the visual SLAM accuracy

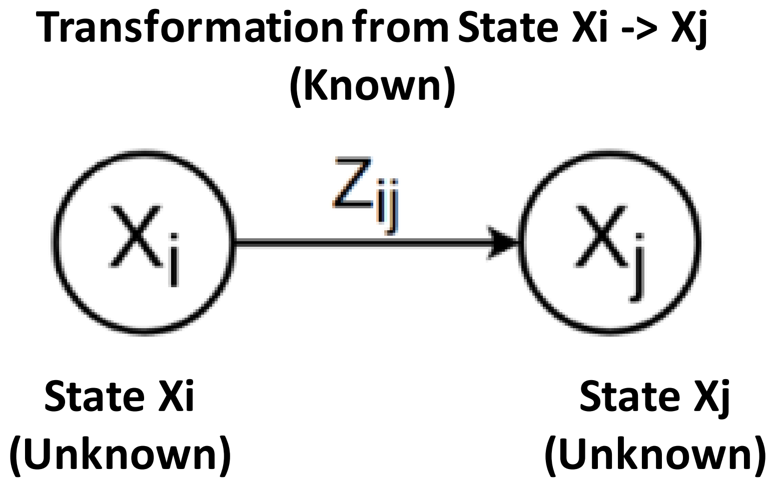
Backend



Backend Refines the rough poses and map

Backend: Graph Optimization

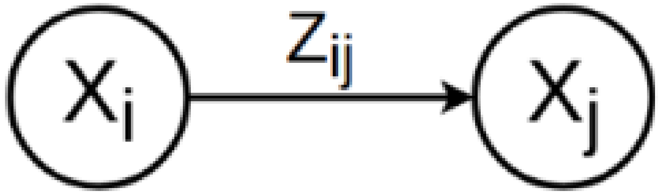
Graph Based Representation:



Backend: Graph Optimization

Graph Based Representation:

Add 3 liters water



Bucket has 2 liters water

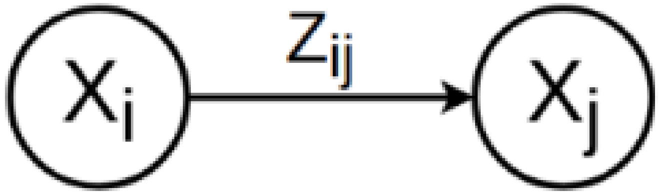
Bucket has 5 liters water

Example

Backend: Graph Optimization

Graph Based Representation:

Add 3 liters water



Bucket has 2 liters water

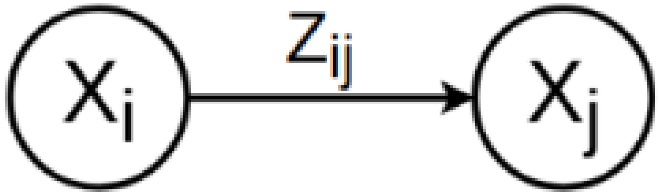
Bucket has 5 liters water

Example

Backend: Graph Optimization

Graph Based Representation:

Add 3 liters water
(Known)



Bucket has
x liters water
(Unknown)

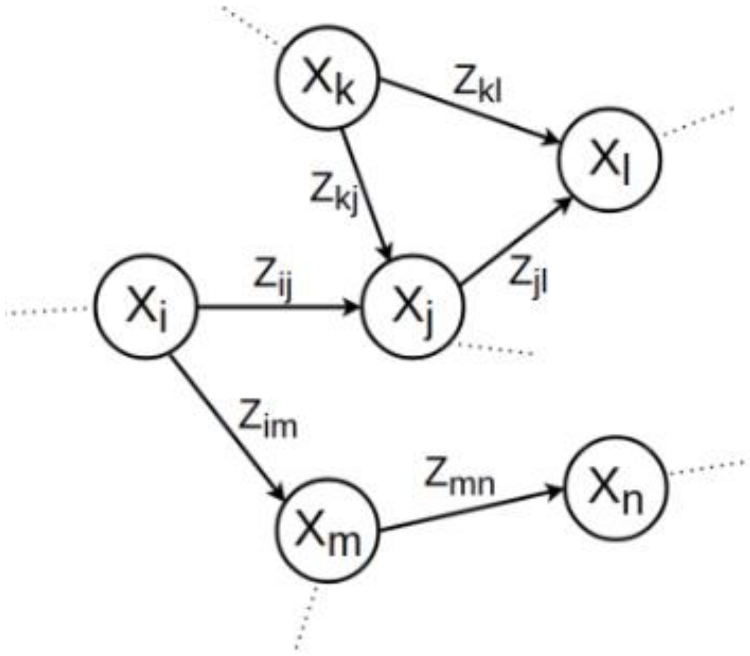
Bucket has
y liters water
(Unknown)

Example

From Zij, we know:
y= x+3

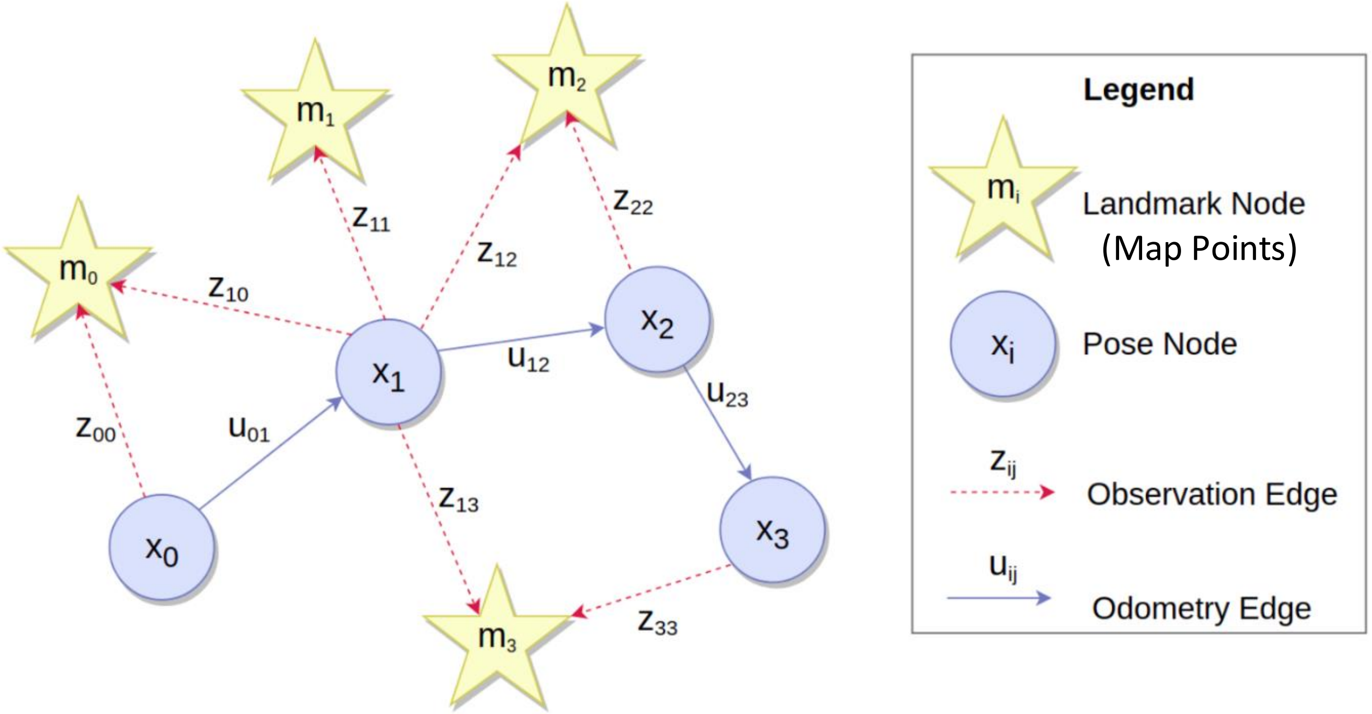
Backend: Graph Optimization

Graph Based Representation:



Backend: Graph Optimization

u= Given by Visual Odometry
z= Given by Triangulation



Graph based representation of Visual SLAM

Backend: Graph Optimization

Find best values for each state (nodes) that best explains the given observation (edges)

Challenges

Change in lighting

Dynamic Objects

Seasonal Change