Lab 8 – Visual SLAM

First: The project

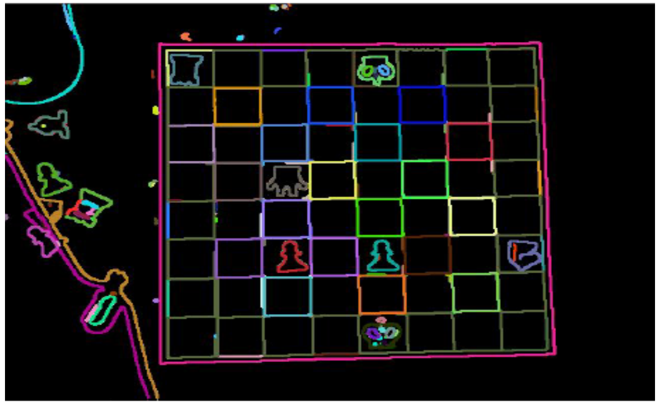
The student project

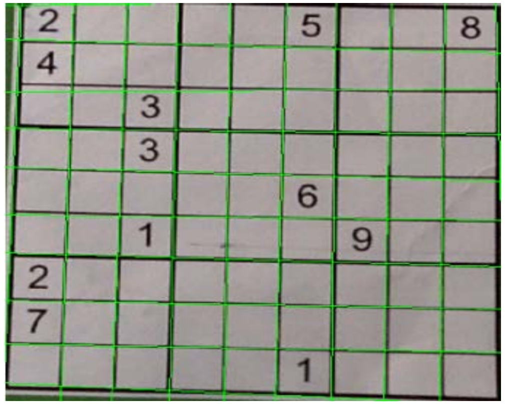
- Develop a functioning computer vision system that does something interesting
  - Large: More than a month
  - Mandatory: 60% of the grade

- Students propose their own projects
- Preferably groups of 2-3 students

The student project

- Develop a functioning computer vision system that does something interesting
  - Large: More than a month
  - Mandatory: 60% of the grade
- Students propose their own projects
- Preferably groups of 2-3 students





The student project

- Freedom of choice
  - Platform, programing language, tools,…
- Important dates
  - **Thursday April 12th**: Hand in written project proposal
  - **Thursday April 19th**: Verbal feedback on the project proposals
  - **Sunday May 27th** : Hand in project report
  - **Thursday May 31st** : Project presentations
- In the project period we will be available for project support here at Kjeller on Thursdays 09:15-12:00
- The lab will in general be available to you (at least within office hours)

The student project

- The project proposal
  - Describes what you want to do
  - Briefly describes how you plan to do it
  - Preferably not more than one page

- The feedback
  - Verbal (here in the lab)
  - Our thoughts regarding difficulty, workload and relevance

- The project report
  - We expect a proper written report
  - Project code

- The project presentation
  - Typically a powerpoint presentation + LIVE demonstration

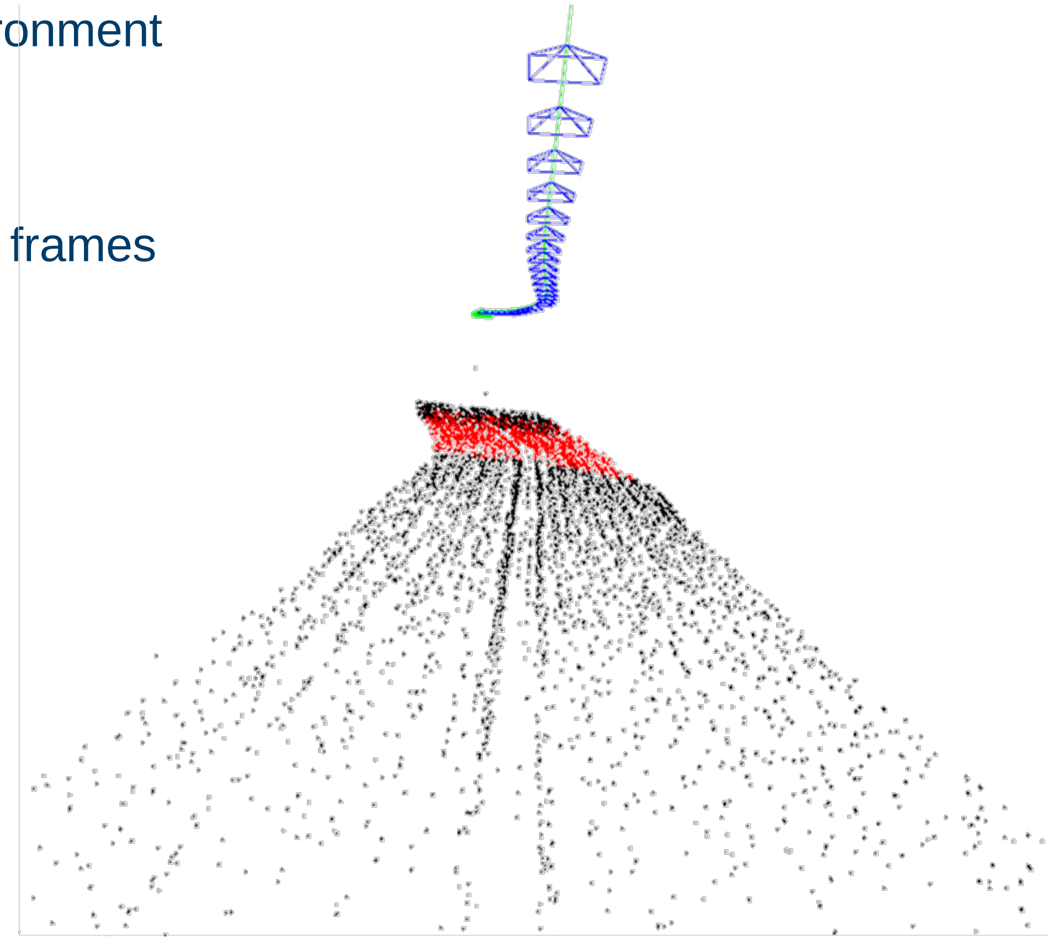
Visual SLAM

Visual simultaneous localization and mapping

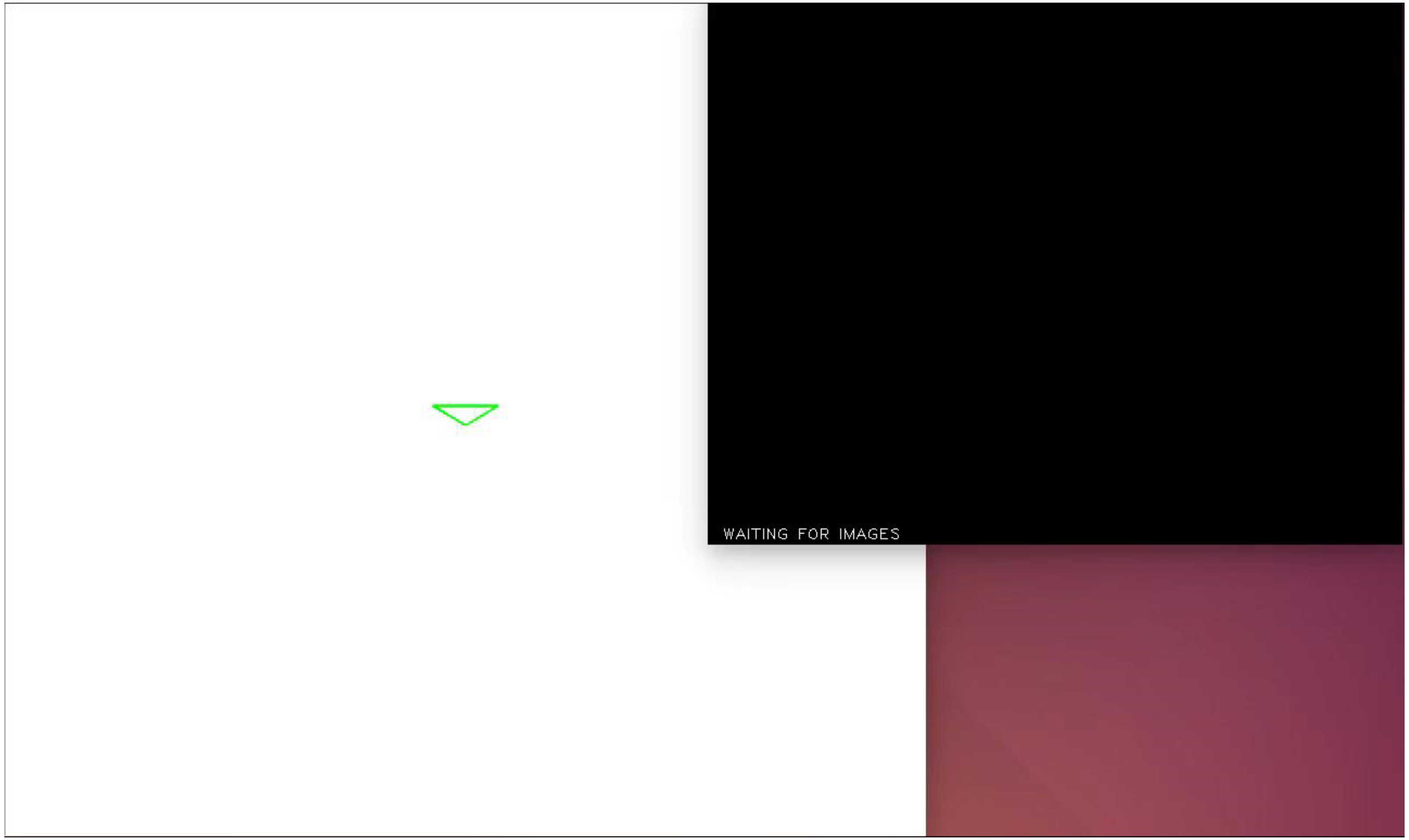
- Mapping
  - Continuously expanding a map while exploring the environment
- Localization (tracking)
  - Localization within the map = tracking the map in image frames
- Loop closure
  - Recognizing areas you have visited before
  - Added as constraints in the map

    ⇒ More consistent map topology

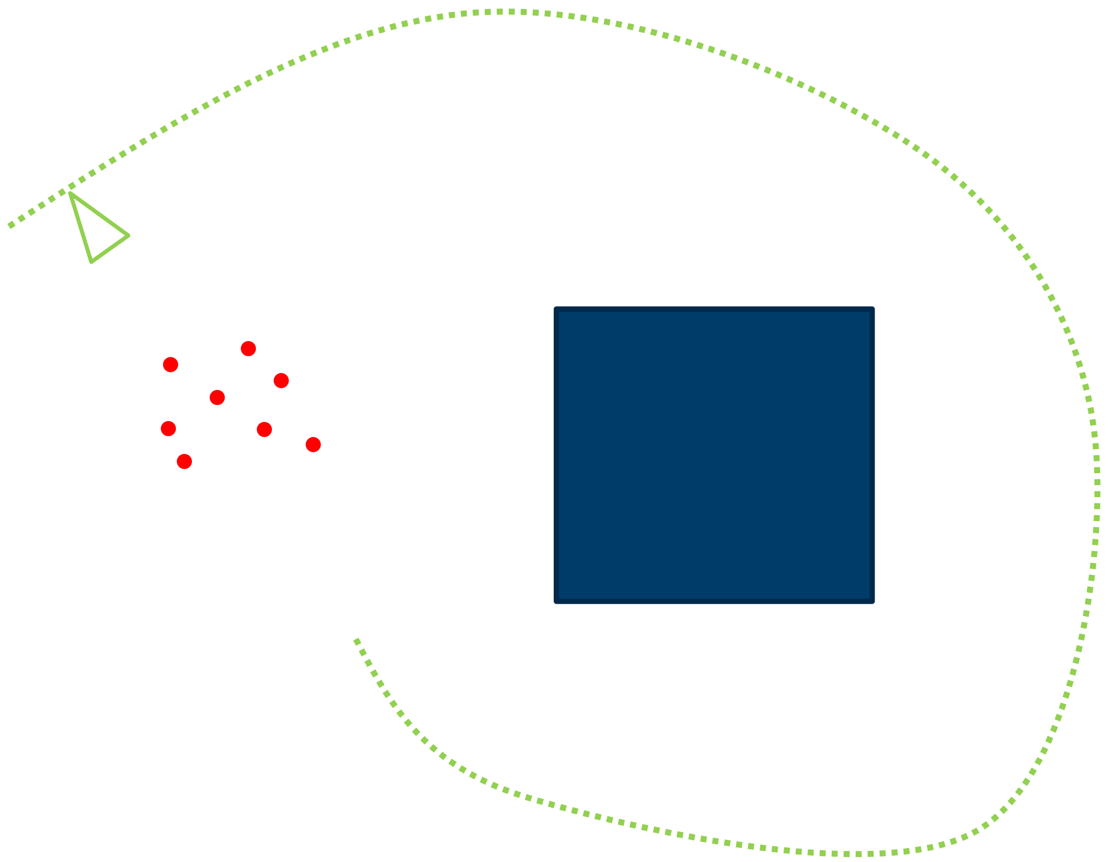
    ⇒ Less drift
- Visual SLAM ≈ “Real-time SfM”
- Visual SLAM ≈ Visual odometry + loop closure



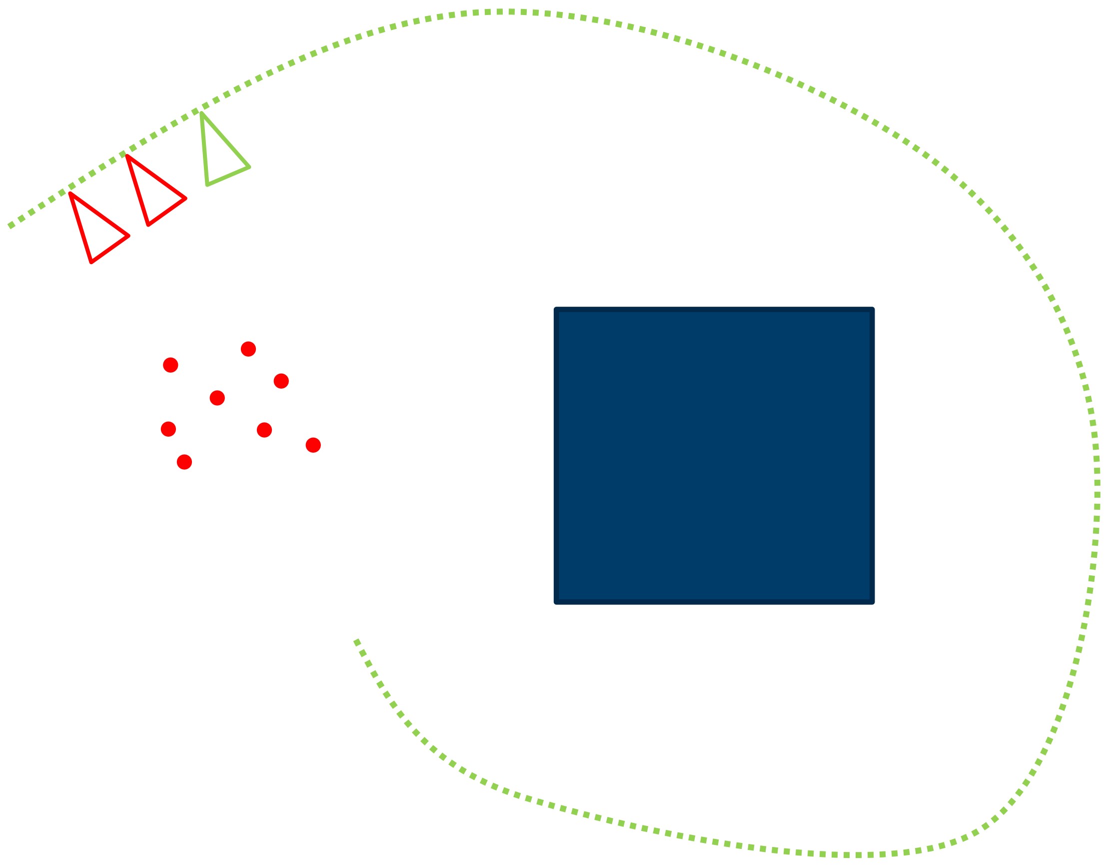
Visual simultaneous localization and mapping



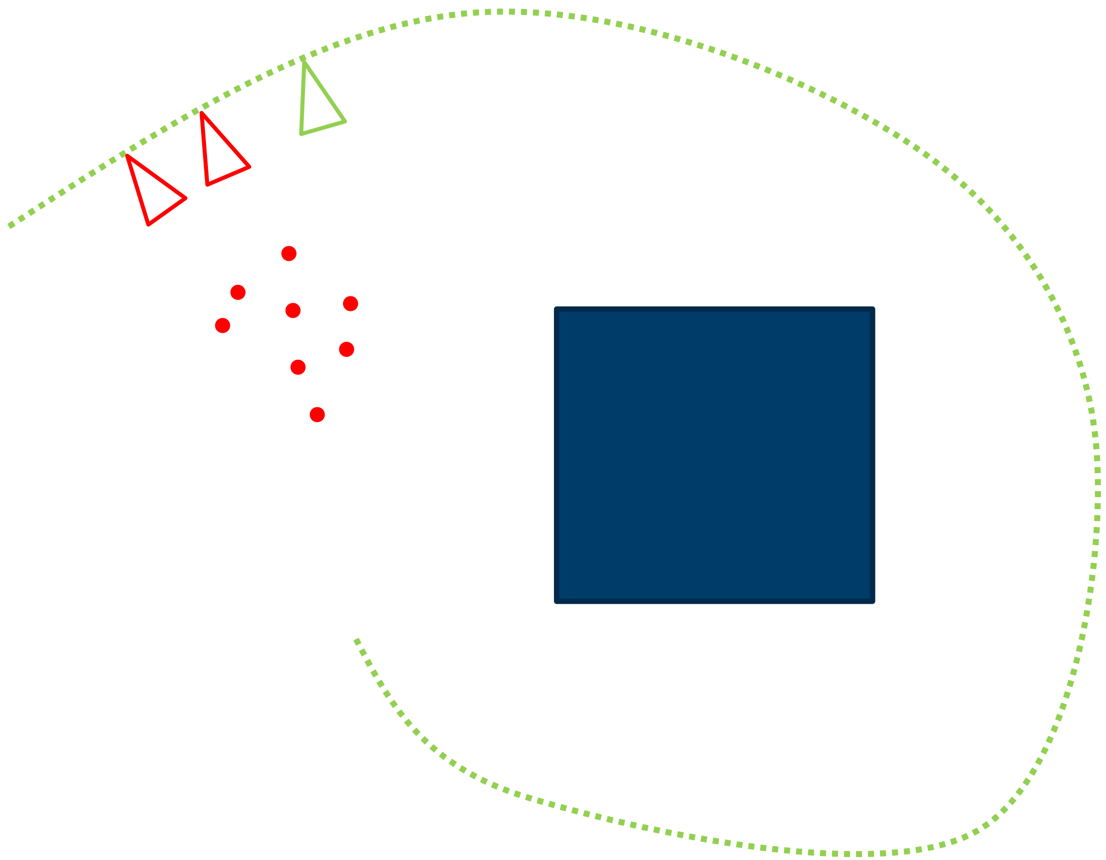
Pose estimation with known 3D points (single-view geometry – lab 5)



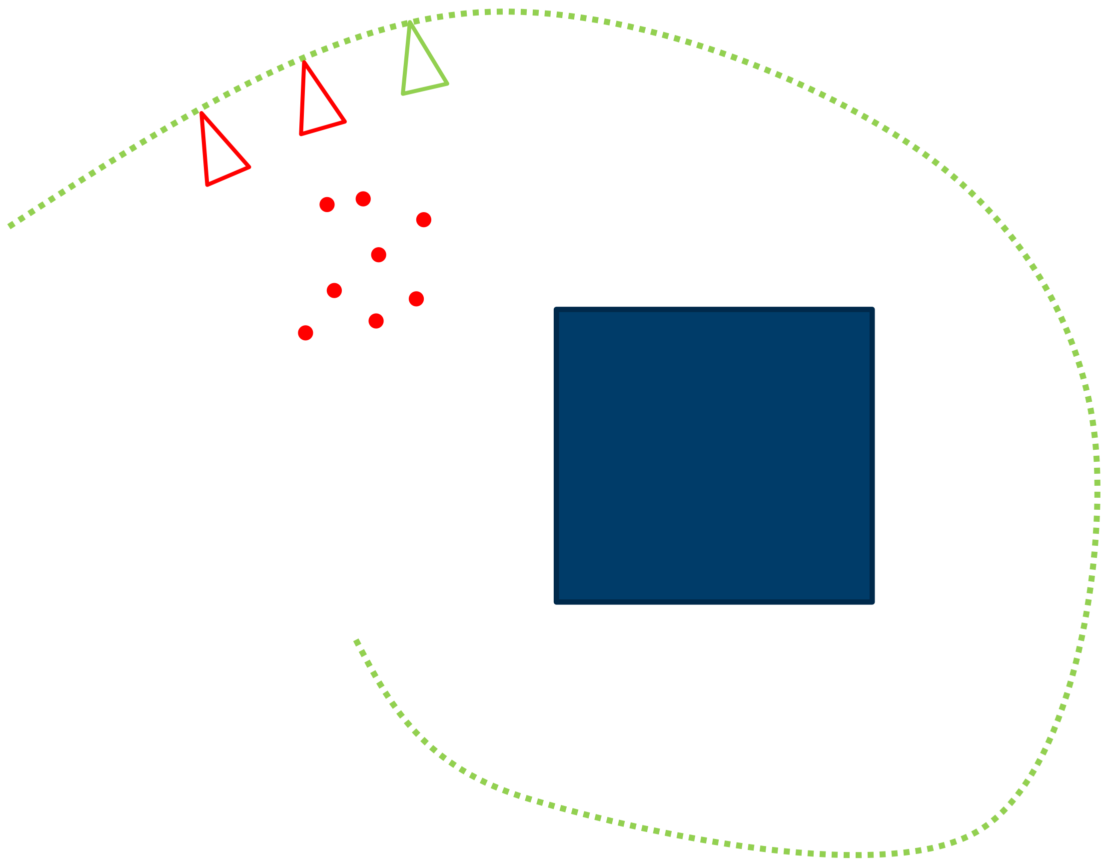
Map initialization and visual odometry (two-view geometry – lab 7)



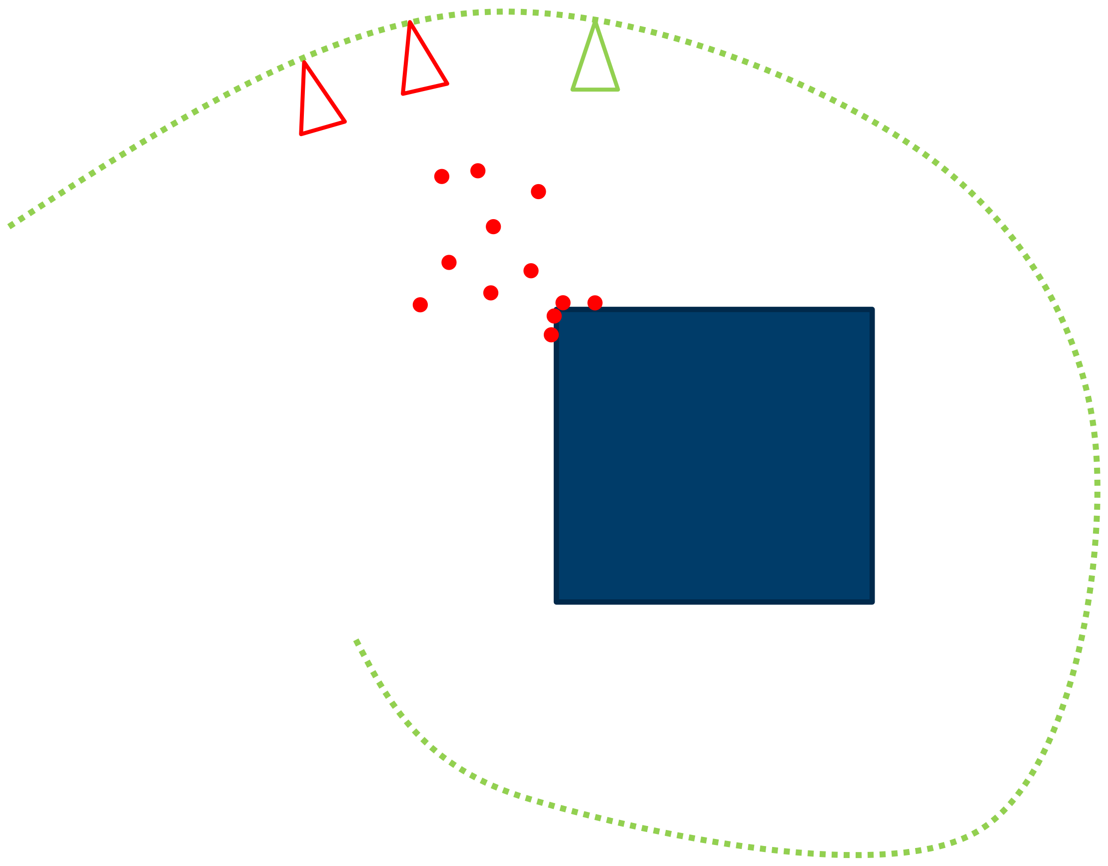
Map initialization and visual odometry (two-view geometry – lab 7)



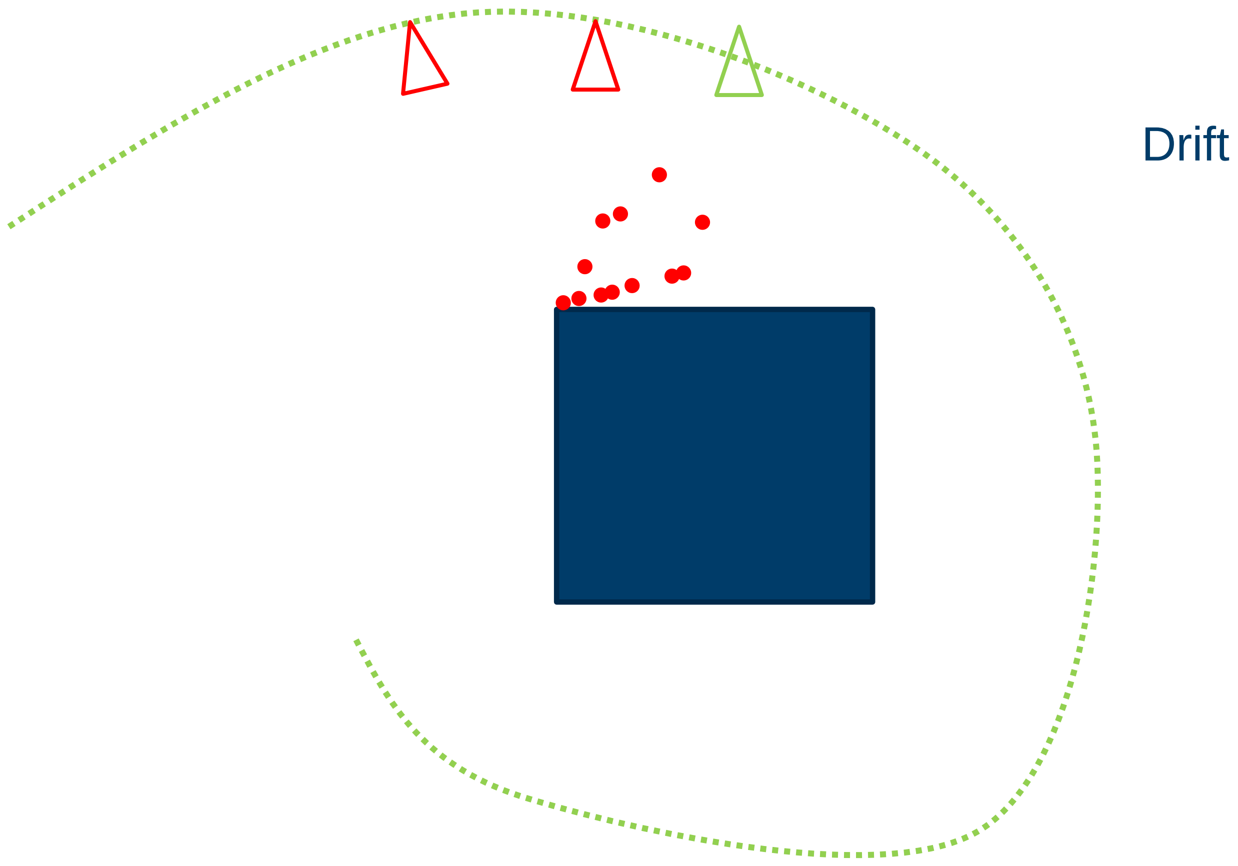
Map initialization and visual odometry (two-view geometry – lab 7)



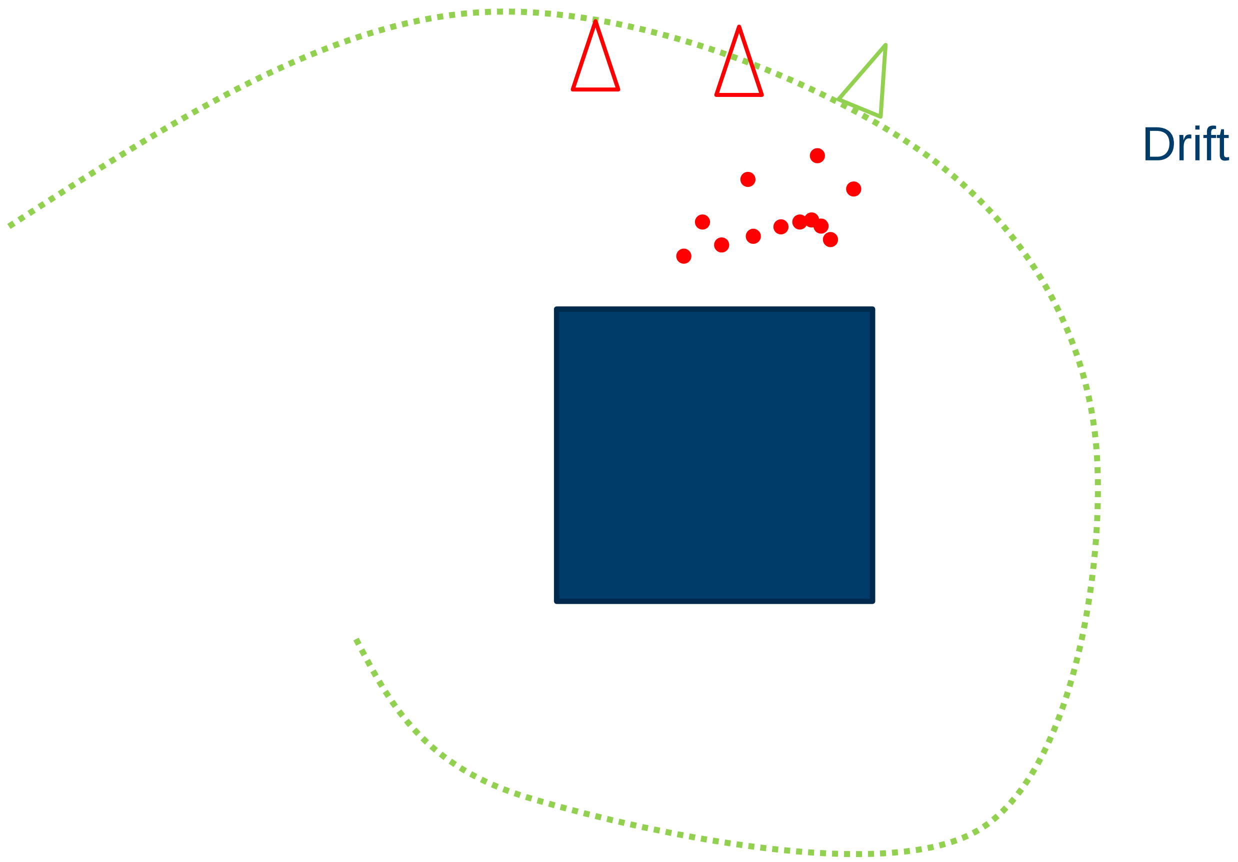
Map initialization and visual odometry
(two-view geometry – lab 7)



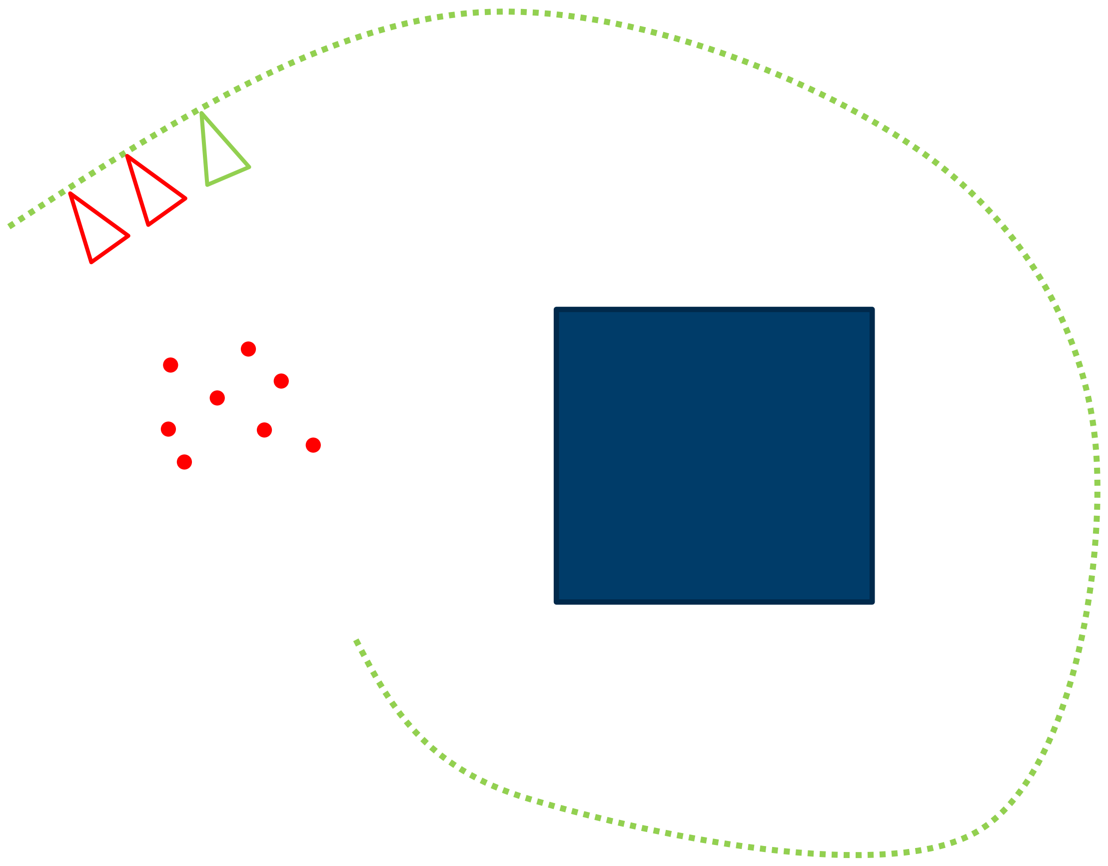
Map initialization and visual odometry (two-view geometry – lab 7)



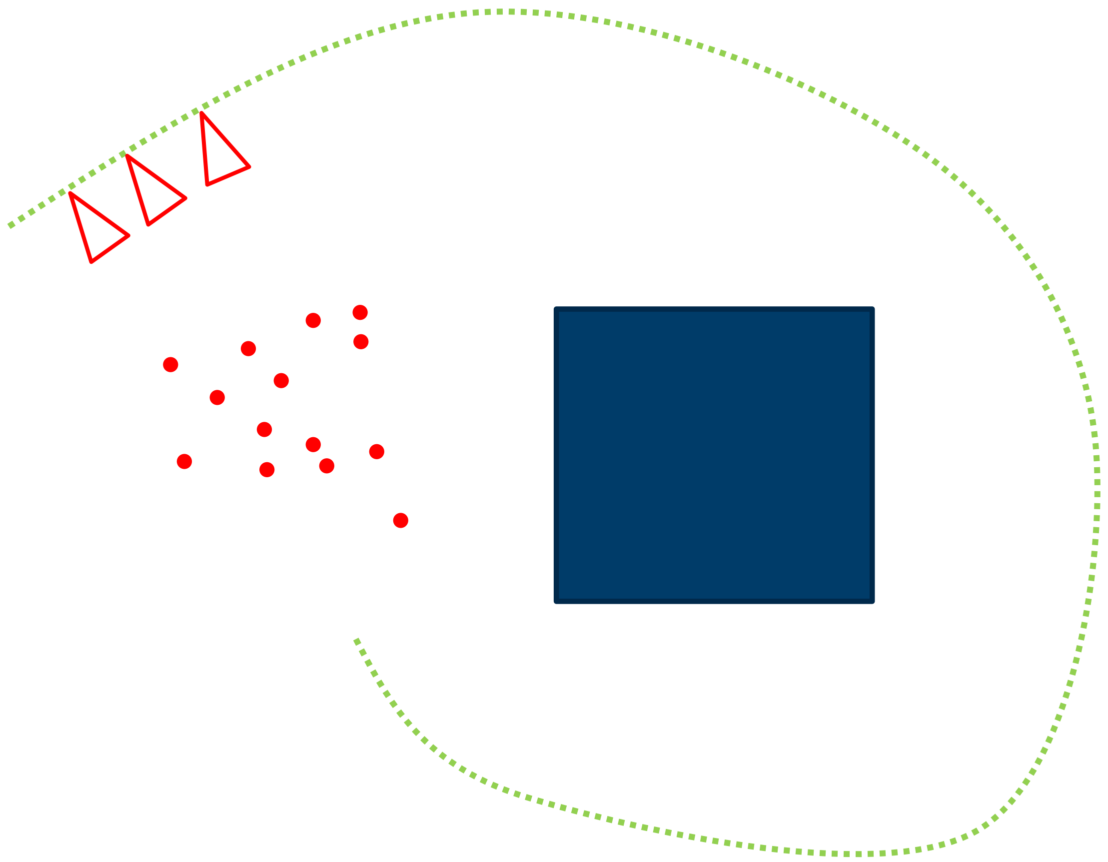
Map initialization and visual odometry (two-view geometry – lab 7)



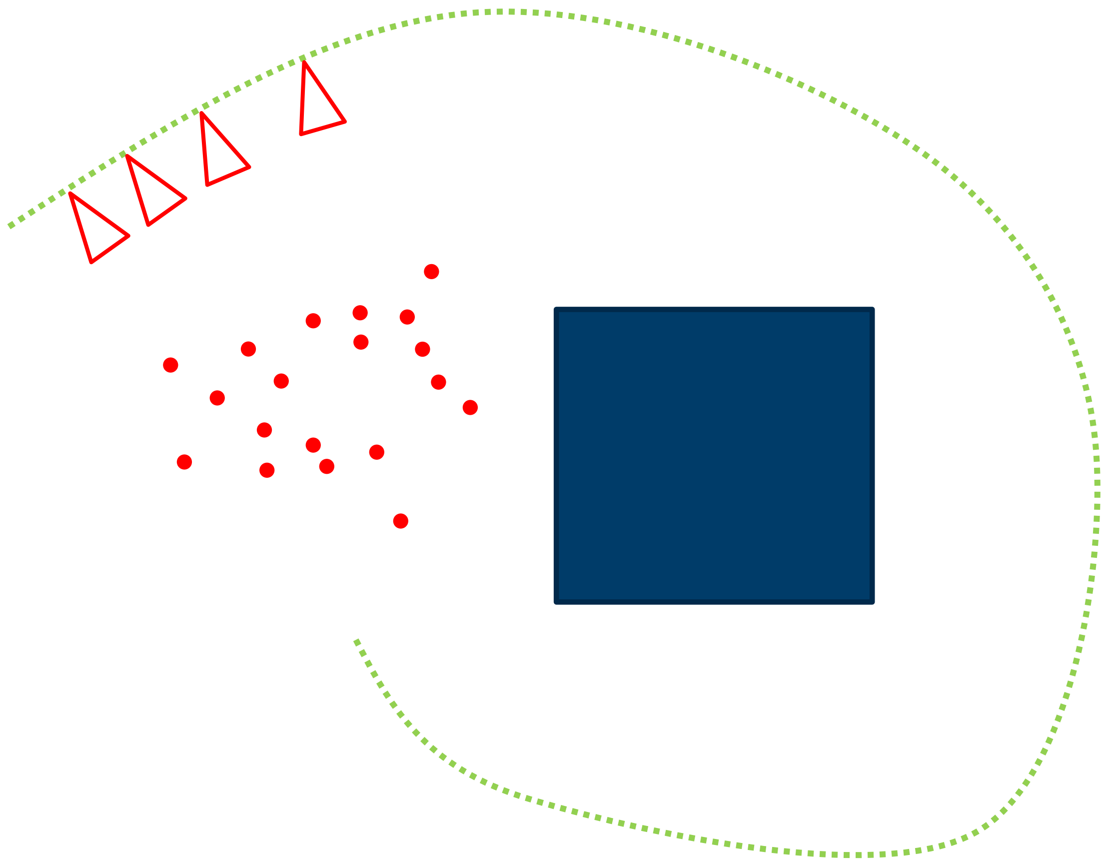
Monocular visual SLAM



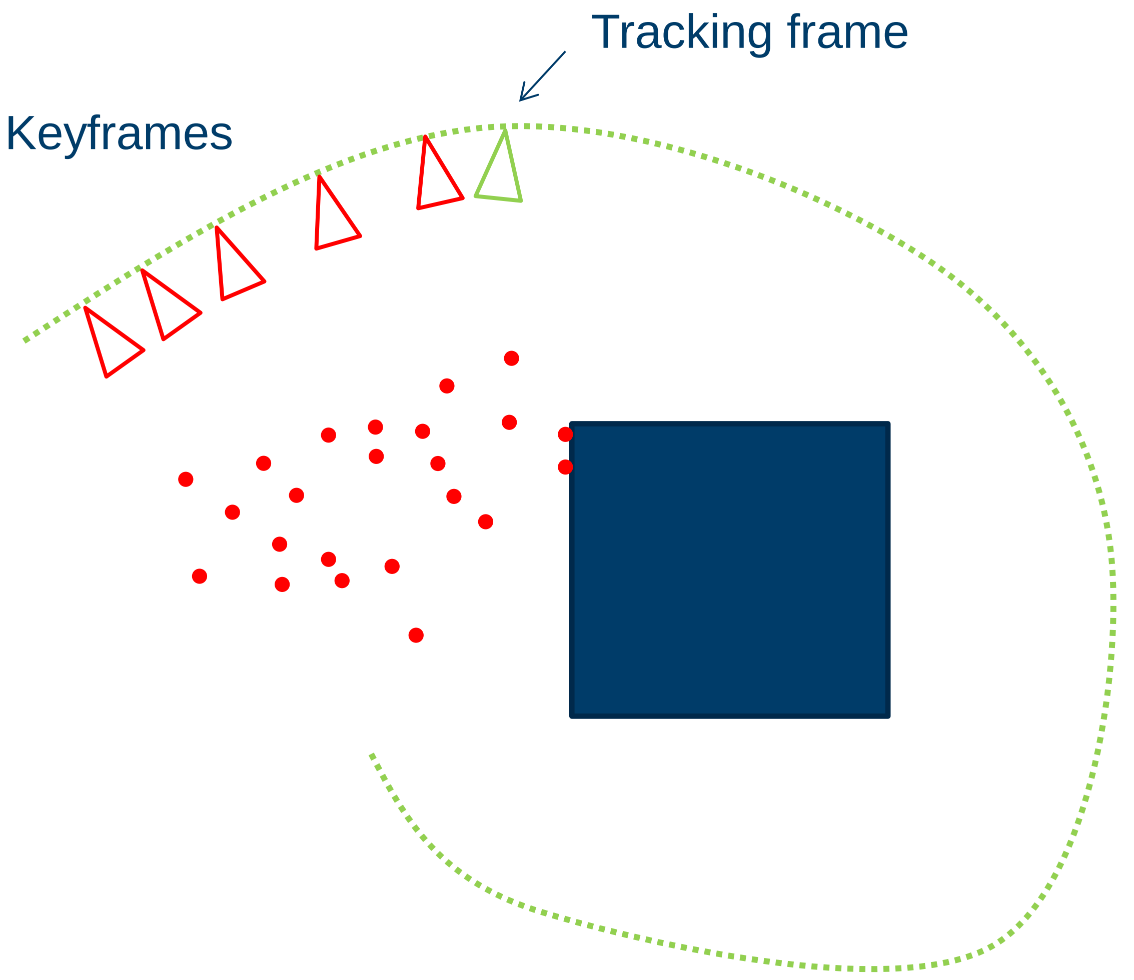
Monocular visual SLAM



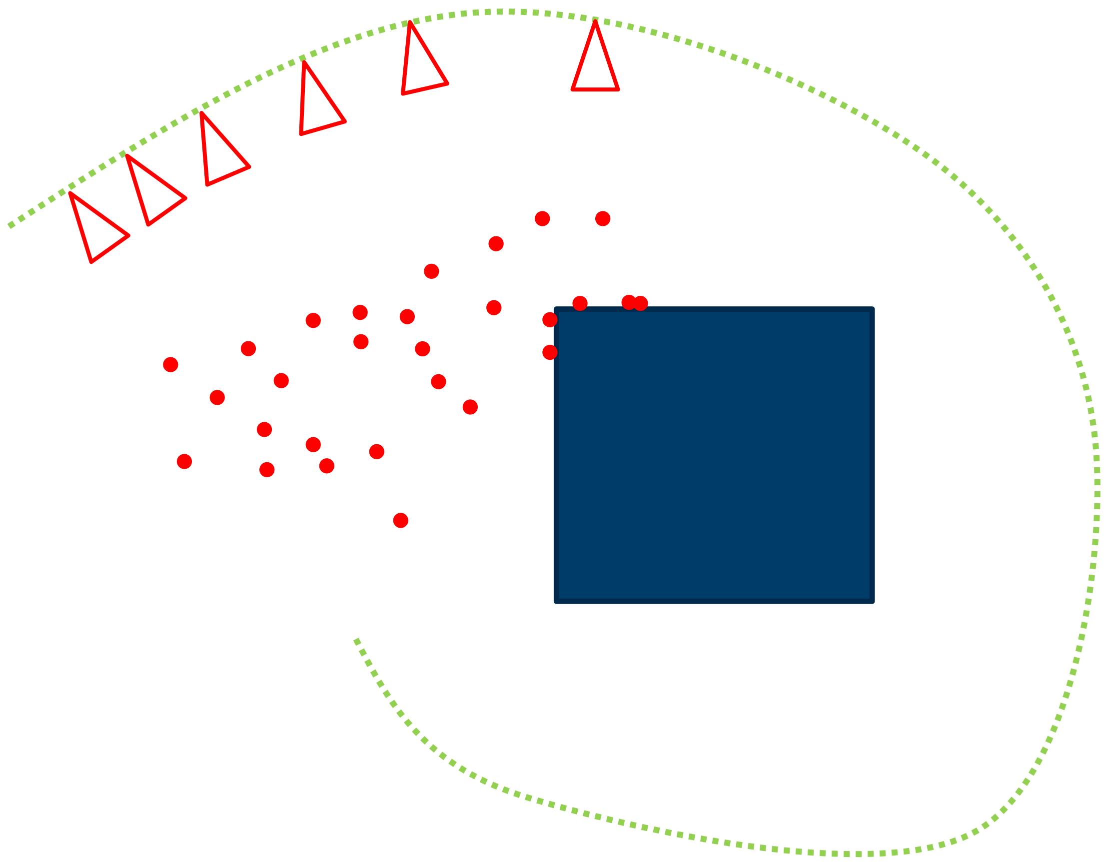
Monocular visual SLAM



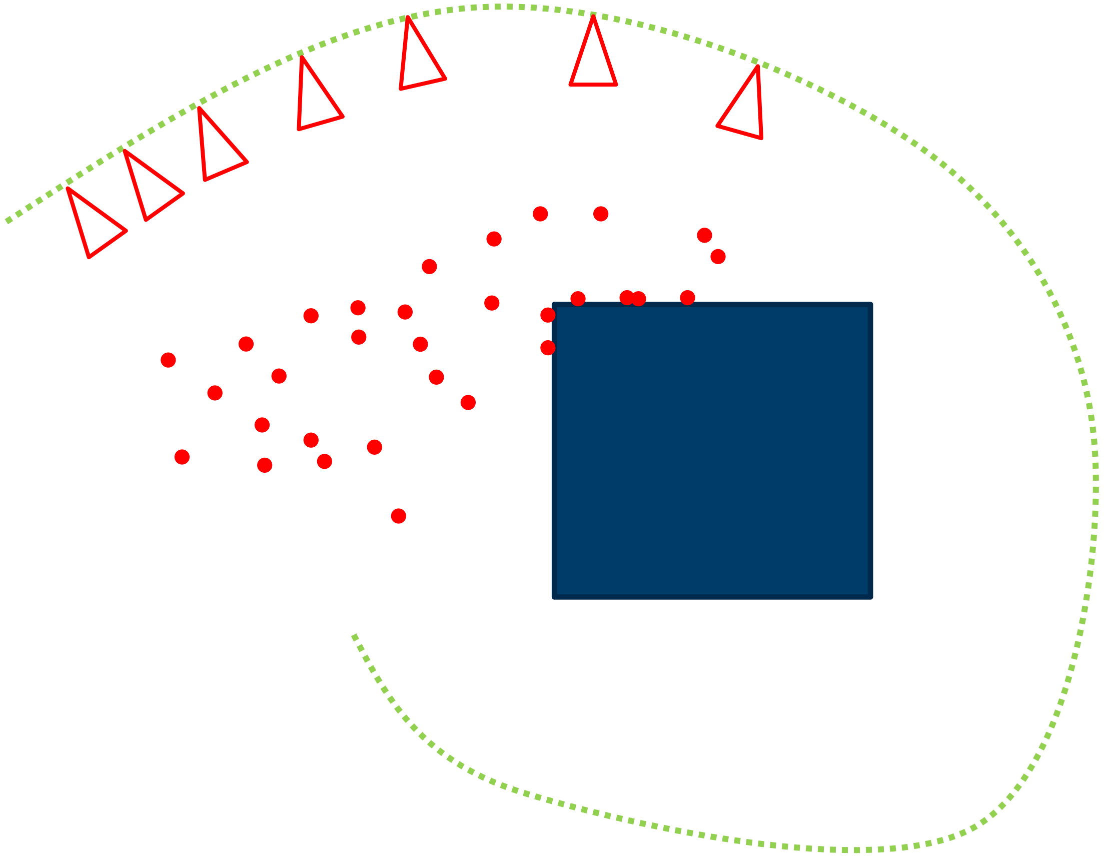
Monocular visual SLAM



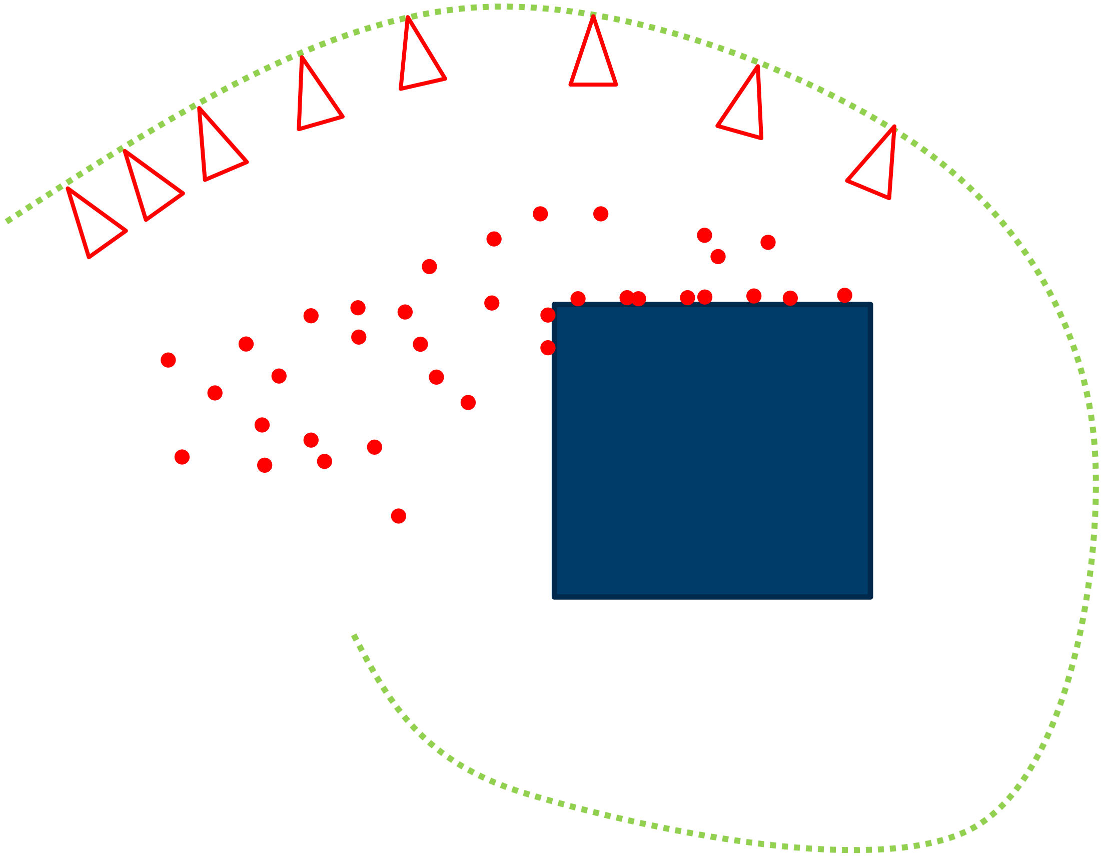
Monocular visual SLAM



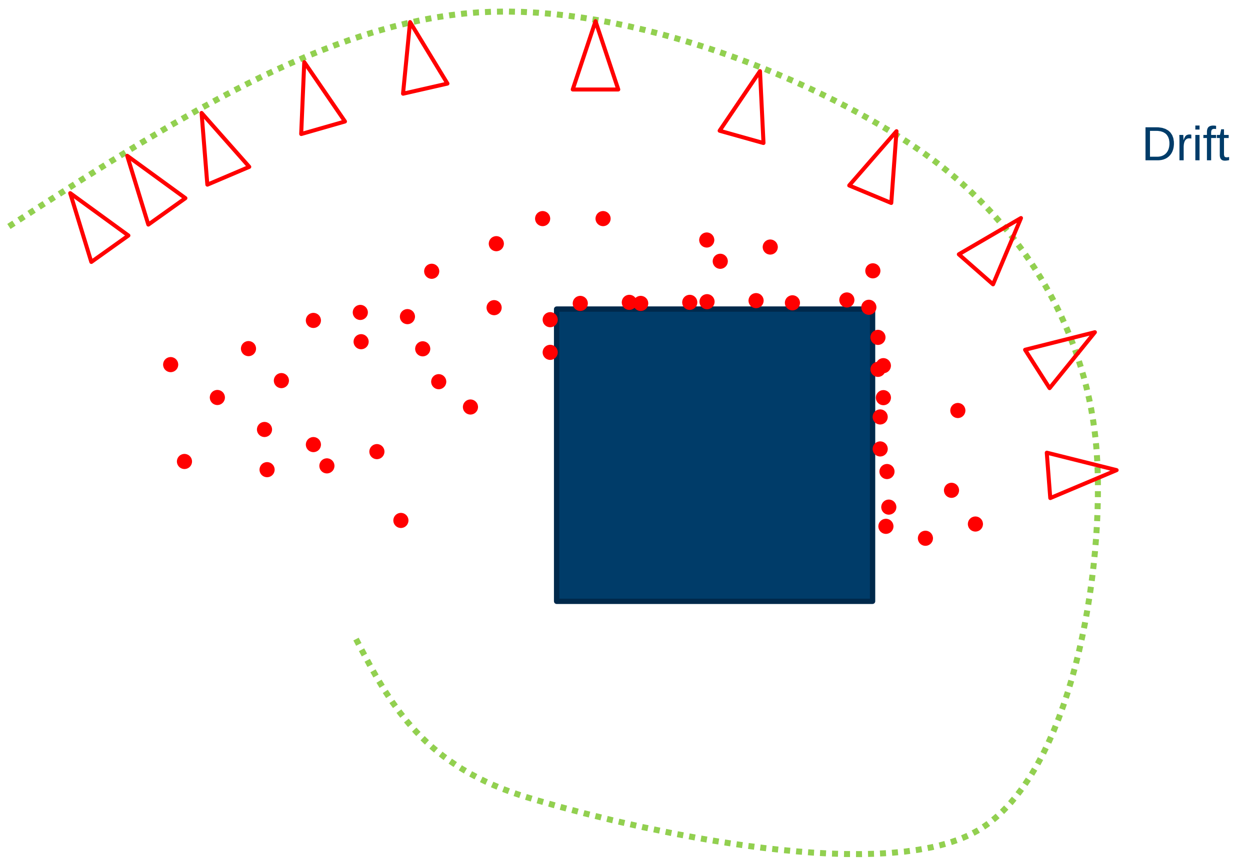
Monocular visual SLAM



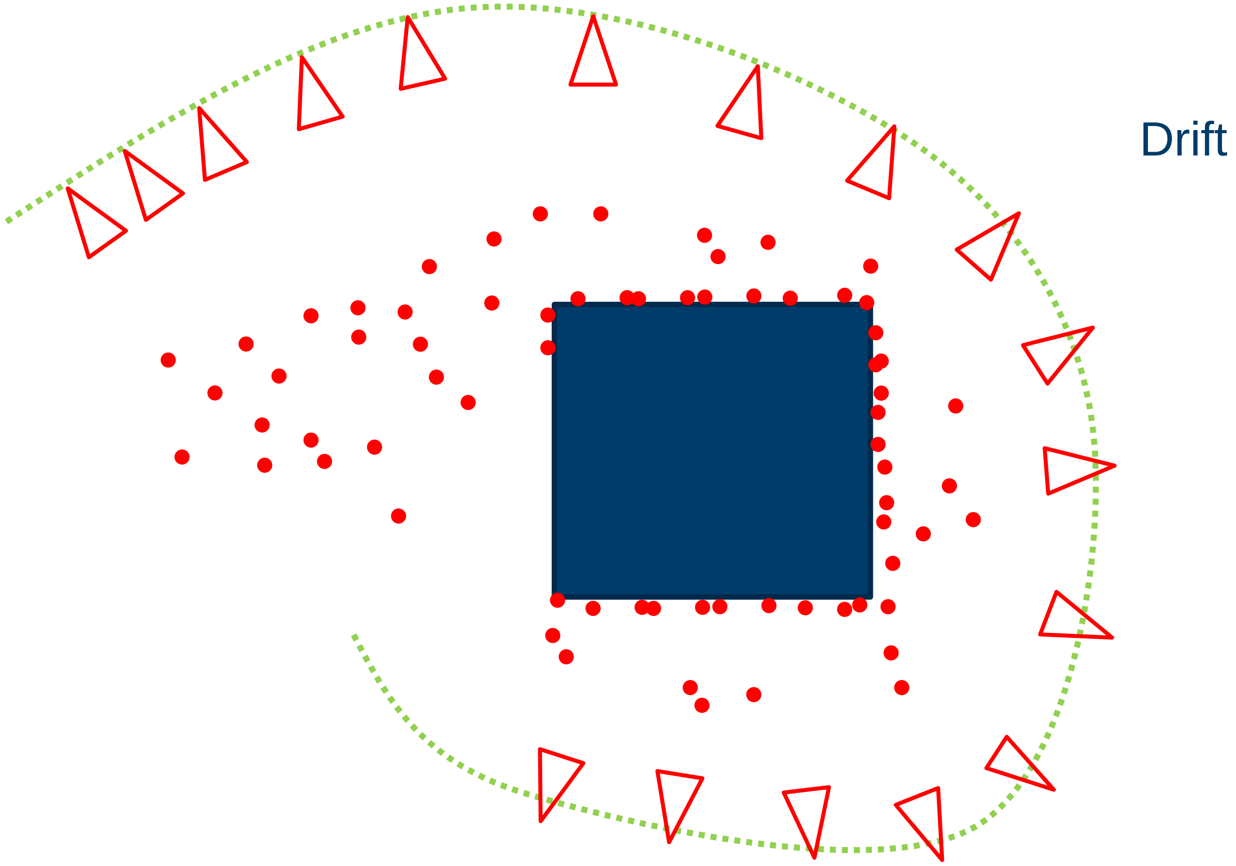
Monocular visual SLAM



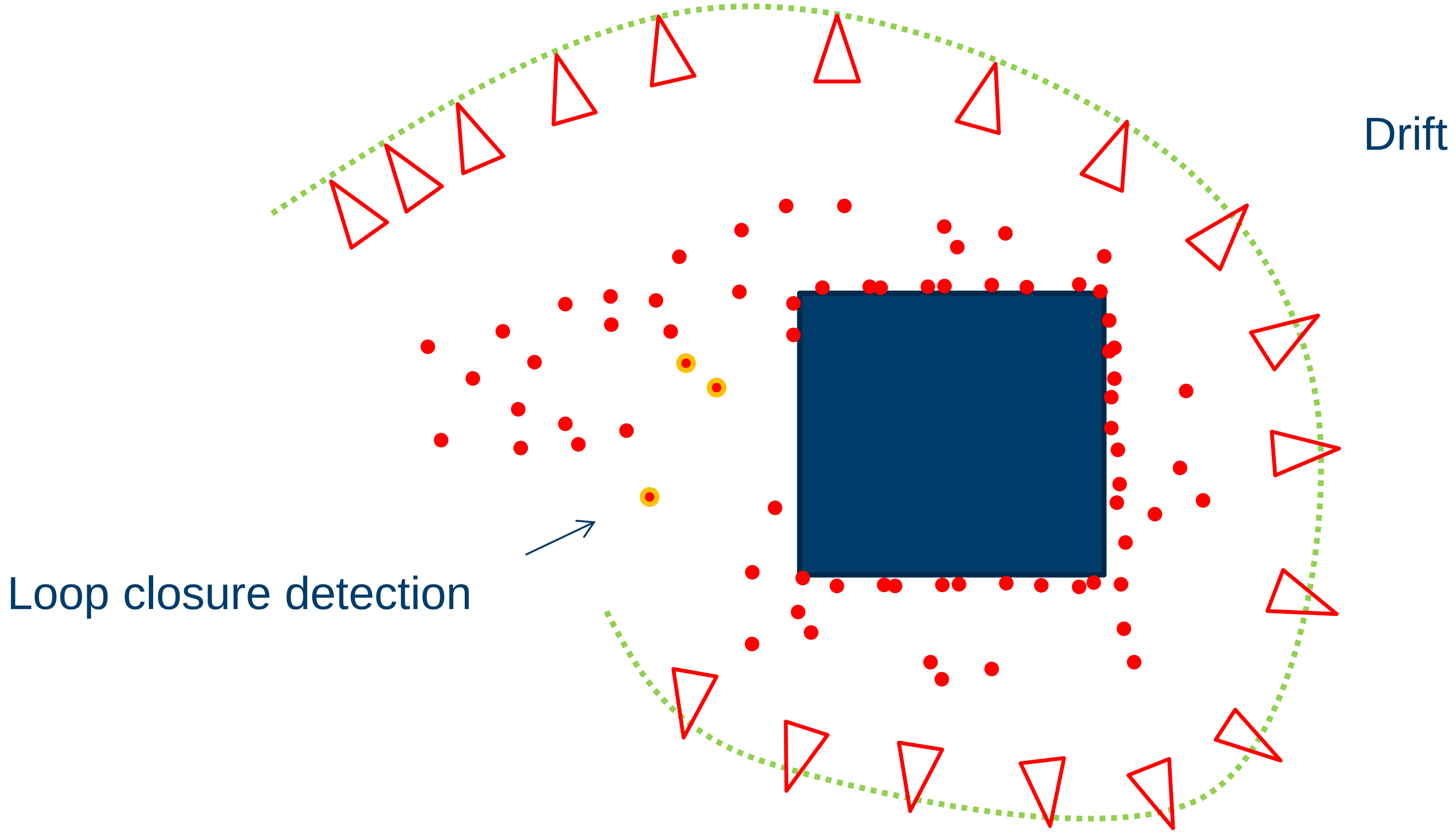
Monocular visual SLAM



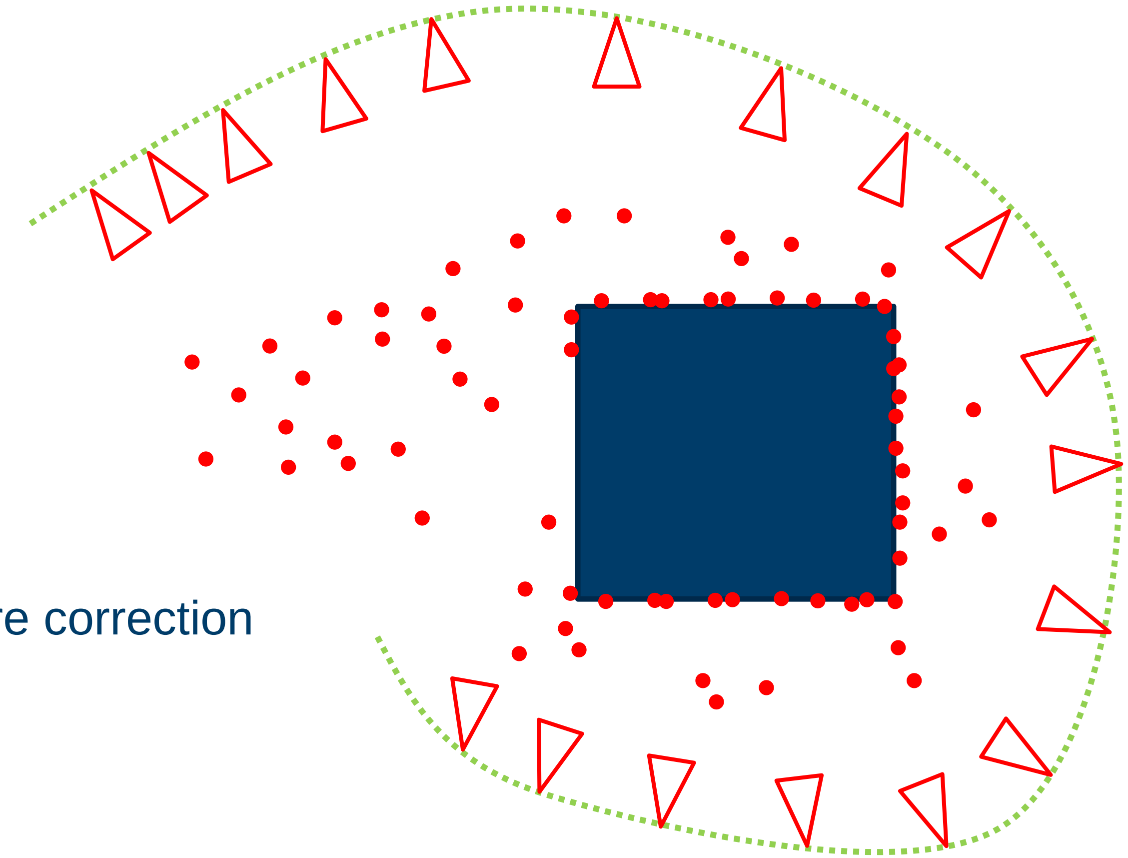
Monocular visual SLAM



Monocular visual SLAM

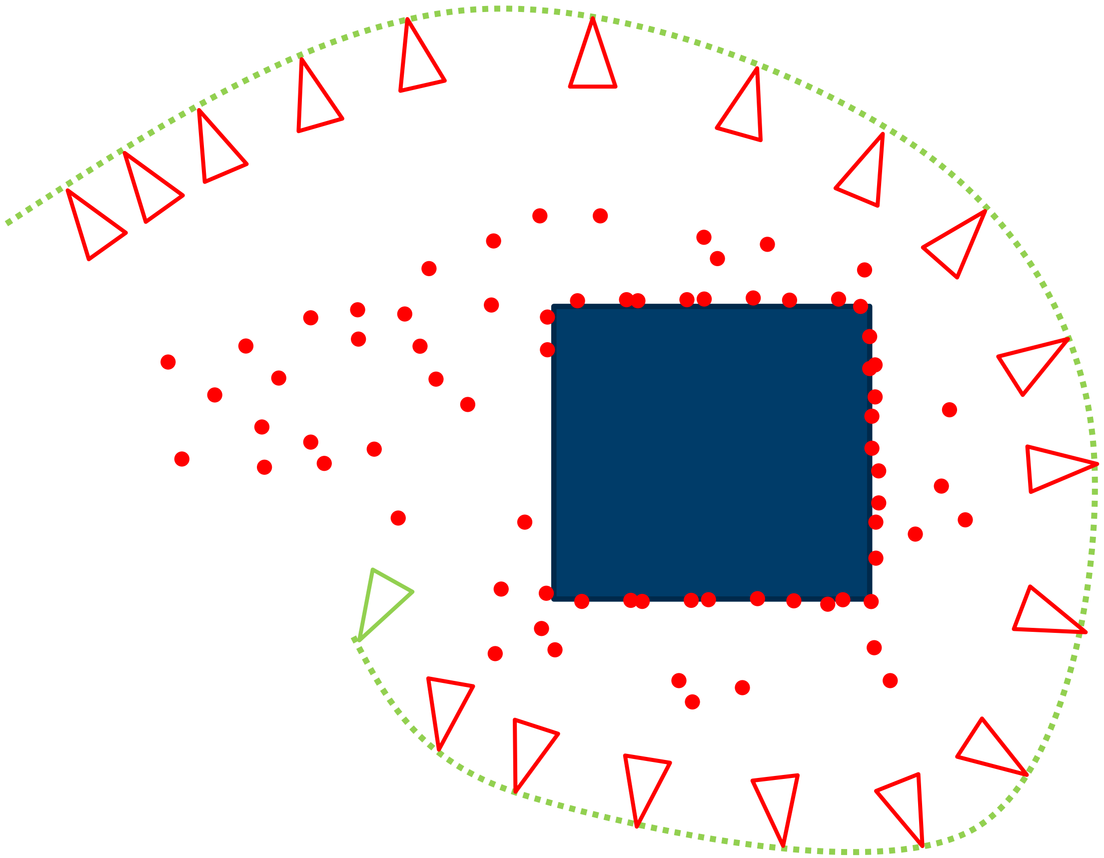


Monocular visual SLAM



Loop closure correction

Monocular visual SLAM



Visual SLAM with ORB-SLAM

Visual SLAM with ORB-SLAM

- https://github.com/raulmur/ORB_SLAM2

- ORB-SLAM is already installed on the lab machines

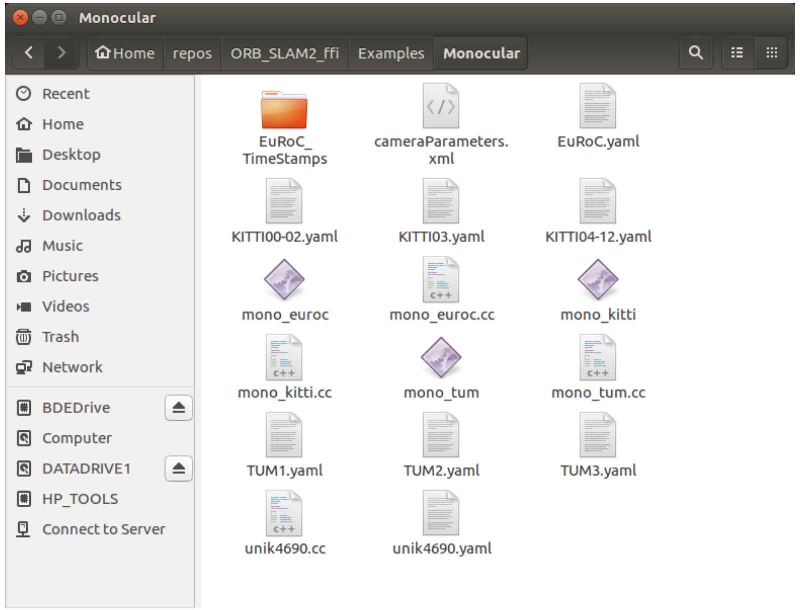
- To use ORB-SLAM with our cameras, we have to
  1. Calibrate the camera
  2. Write a program that captures images from the camera and transfers them to ORB-SLAM
  3. Build the program
  4. Run the program

Visual SLAM with ORB-SLAM

- Step 1: Calibration
  - Use calibration from previous lab
- Or:

```
opencv_interactive-calibration -ci=0 -t=chessboard -sz=30 -w=8 -h=5 -pf=calibSettings.xml
```

  - Fill the calibration into unik4690.yaml, and put it in ORB_SLAM2/Examples/Monocular/:



Visual SLAM with ORB-SLAM

- Step 2: Add the camera capture program
  - Copy unik4690.cc to ORB_SLAM2/Examples/Monocular/
  - Add as a new «target» in ORB_SLAM2/CMakeLists.txt:

```
set(CMAKE_RUNTIME_OUTPUT_DIRECTORY ${PROJECT_SOURCE_DIR}/Examples/Monocular)

add_executable(mono_tum
Examples/Monocular/mono_tum.cc)
target_link_libraries(mono_tum ${PROJECT_NAME})

add_executable(mono_kitti
Examples/Monocular/mono_kitti.cc)
target_link_libraries(mono_kitti ${PROJECT_NAME})

add_executable(mono_euroc
Examples/Monocular/mono_euroc.cc)
target_link_libraries(mono_euroc ${PROJECT_NAME})

add_executable(unik4690
Examples/Monocular/unik4690.cc)
target_link_libraries(unik4690 ${PROJECT_NAME})
```

Visual SLAM with ORB-SLAM

- Step 3: Build
  - In ORB_SLAM2/:
    ```
    ./build.sh
    ```

- Step 4: Run
  - In ORB_SLAM2/Examples/Monocular:
    ```
    ./unik4690 ../../Vocabulary/ORBvoc.txt unik4690.yaml
    ```

- Step 5: Play!

- Step 6: Take a look at the code in ORB_SLAM2!
  - Recognize anything?

3D reconstruction

- Download and try Agisoft Photoscan
  - http://www.agisoft.com/downloads/installer/
- Capture images:
  - Write a program that captures images when you press space
  - Store the images with `cv::imwrite()`
- Or use your phone…