**MINOS, ORBSLAM Integration Procedure**

The purpose of this integration is to run SLAM over a simulated environment. MINOS : Multimodal Indoor Simulator is a simulator designed for navigation in complex indoor environments. With the help of MINOS, we can experiment over SLAM using a simulated and static environment.

**Method:**

1. We chose ORB SLAM (By Raul Mur et al ) for integration with MINOS.
2. ORB SLAM detects features of images that are published over a ROS Topic.
3. We needed to make sure that the MINOS images were published to this ROS Topic. The problem is that MINOS does not work over the ROS environment.
4. Fortunately, MINOS has an option of saving its frames in a folder. We modified this feature which allowed MINOS to save its frames by over-writing that frame over a single image.
5. A ROS node was created which publishes this image over a topic and that topic is subscribed by ORB SLAM.

Pre-requisites: (Run following command in terminal)

*sudo apt-get install python3.5-dev && sudo apt-get install python3-tk && sudo apt-get install build-essential libxi-dev libglu1-mesa-dev libglew-dev libopencv-dev libvips && sudo apt install git && sudo apt install curl && libboost-all-dev*

**1. Install ROS Kinetic first:**

Follow <http://wiki.ros.org/kinetic/Installation/Ubuntu>

**2. Install ORBSLAM:**

* At home type:

git clone “https://github.com/raulmur/ORB\_SLAM”

* After Download is complete, build Thirdparty packages, build g2o.

Go into Thirdparty/g2o/ and execute:

sh

*mkdir build*

*cd build*

*cmake .. -DCMAKE\_BUILD\_TYPE=Release*

*make*

Build DBoW2. Go into Thirdparty/DBoW2/ and run:

*mkdir build*

*cd build*

*cmake .. -DCMAKE\_BUILD\_TYPE=Release*

*make*

* A few changes need to be done before building ORB\_SLAM. After compiling thirdparty g2o and DBoW2, before building ORB\_SLAM

1. In src/ORBextractor.cc include OpenCV library:

*#include <opencv2/opencv.hpp>*

(b) Remove opencv2 dependency from manifest.xml

1. In CmakeList.txt, add lboost as target link library which can be done by replacing

*target\_link\_libraries(${PROJECT\_NAME}*

*${OpenCV\_LIBS}*

*${EIGEN3\_LIBS}*

*${PROJECT\_SOURCE\_DIR}/Thirdparty/DBoW2/lib/libDBoW2.so*

*${PROJECT\_SOURCE\_DIR}/Thirdparty/g2o/lib/libg2o.so*

*)*

with

*target\_link\_libraries(${PROJECT\_NAME}*

*${OpenCV\_LIBS}*

*${EIGEN3\_LIBS}*

*${PROJECT\_SOURCE\_DIR}/Thirdparty/DBoW2/lib/libDBoW2.so*

*${PROJECT\_SOURCE\_DIR}/Thirdparty/g2o/lib/libg2o.so*

*-lboost\_system*

*)*

(d) Install eigen form here <https://launchpad.net/ubuntu/trusty/amd64/libeigen3-dev/3.2.0-8>

Download the debian file and install using

sudo dpkg -i libeigen3-dev\_3.2.0-8\_all.deb

* Before building ORB\_SLAM run this is terminal, (change PC name accordingly)

export ROS\_PACKAGE\_PATH=${ROS\_PACKAGE\_PATH}:/home/romi

then run:

roslaunch ORB\_SLAM ExampleGroovyOrNewer.launch

IN THE FILE … ROB\_SLAM/SRC/TRACKING.CC

on line 163 use following line:

ros::Subscriber sub = nodeHandler.subscribe("/usb\_cam/image\_raw", 1, &Tracking::GrabImage, this);

**3. MINOS installation procedure:**

**Installation Method**

1. git clone <https://github.com/minosworld/minos>
2. Now change code branch to following

Download this version of repository too

Now go to the minos folder, press Ctrl+h and copy the .git folder

Paste this .git folder to the minos0.7x folder

Delete the minos folder and change the name of minos0.7x folder to minos

1. Open the minos folder and open terminal on it.
2. Install node.js using the Node Version Manager (nvm).

curl -o- <https://raw.githubusercontent.com/creationix/nvm/v0.33.7/install.sh> | bash

source ~/.bashrc

nvm install 8.11.3

nvm alias default 8.11.3

*OR*

*nvm install v10.13.0*

*nvm alias default 10.13.0*

1. Build the MINOS server modules inside the server directory by:

npm install -g yarn

yarn install

*OR (not recommended)*

*npm install*

This process will download and compile all server module dependencies and might take a few minutes.

1. Install the minos Python module by running pip3 install -e . in the root of the repository or pip3 install -e . -r requirements.txt

**RUN**

Before running MINOS copy the Materport3D and SUNCG datasets in the work folder.

Check that everything works by running the interactive client through

* **python3 -m minos.tools.pygame\_client**
* **python3 -m minos.tools.pygame\_client --dataset mp3d --scene\_ids 17DRP5sb8fy --env\_config pointgoal\_mp3d\_s –save\_png --depth**

invoked from the root of the MINOS repository. You should see a live view which you can control with the W/A/S/D keys and the arrow keys. This client can be configured through various command line arguments. Run with the --help argument for an overview and try some of these other examples:

**MINOS Code Changes:**

For merging purpose, we need to save simulator frames in a folder of our choice, go to minos/minos/lib/Simulator.py and

replace lines 98-102 with

*if 'logdir' in params:*

*self.\_logdir = '/home/romi/frames'*

*else:*

*self.\_logdir = '/home/romi/frames'*

and replace line 422-428 with

*if self.params.get(‘save\_png’):*

*if image is None:*

*image = Image.frombytes(mode,(data.shape[0], data.shape[1]),data)*

*time.sleep(0.06)*

*image.save(‘/home/romi/frames/color\_.jpg’)*

and save Simulator.py

**Create ROS Node:**

This ROS node is important, simply paste the attached ROS node (merger) in catkin\_ws/src and do necessary changes to CmakeList.txt by Uncommenting following:

*add\_compile\_options(-std=c++11)*

*add\_executable(${PROJECT\_NAME}\_node src/merger.cpp)*

*target\_link\_libraries(${PROJECT\_NAME}\_node*

*${catkin\_LIBRARIES}*

*)*

*Add find\_package(catkin REQUIRED COMPONENTS*

*roscpp*

*rospy*

*std\_msgs*

*image\_transport*

*std\_msgs*

*cv\_bridge*

*)*

Later, catkin\_make and test by running following command:

rosrun merger merger\_node

Following is our ROS node, we named it “merger.cpp”

**Merger.cpp**

#include <ros/ros.h>

#include <iostream>

#include <image\_transport/image\_transport.h>

#include <opencv2/highgui/highgui.hpp>

#include <cv\_bridge/cv\_bridge.h>

using namespace std;

int main(int argc, char\* argv[])

{

ros::init(argc,argv,"minos\_image\_publisher");

ros::NodeHandle nh;

image\_transport::ImageTransport it(nh);

image\_transport::Publisher pub=it.advertise("/usb\_cam/image\_raw",12);

stringstream ss;

std::string image\_name;

stringstream exception\_text;

ss<<"/home/romi/frames/color\_.jpg";

image\_name=ss.str();

cv::Mat image;

sensor\_msgs::ImagePtr msg;

ros::Rate loop\_rate(5);

while(nh.ok()) {

image=cv::imread(image\_name,CV\_LOAD\_IMAGE\_COLOR);

if(!image.empty()) {

msg=cv\_bridge::CvImage(std\_msgs::Header(),"bgr8",image).toImageMsg();

pub.publish(msg);

}

//ROS\_INFO\_STREAM(image\_name);

ros::spinOnce();

loop\_rate.sleep();

}

}

Change ORB SLAM Camera Calibration Parameters:

In order to obtain features quickly, we need to make sure that the Camera Calibration parameters are set according to MINOS. Usama Muddassar, from ROMI Lab has worked on calculating the Camera Parameters for Matterport3D indoor scenes. These settings must be set in /ORB\_SLAM/Data/Settings.yaml:

%YAML:1.0

# Camera Parameters. Adjust them!

# Camera calibration parameters (OpenCV)

Camera.fx: 890.246939

Camera.fy: 889.082597

Camera.cx: 378.899791

Camera.cy: 210.334985

# Camera distortion parameters (OpenCV) --

Camera.k1: 0.224181

Camera.k2: -1.149847

Camera.p1: 0.007295

Camera.p2: 0.0

# Camera frames per second

Camera.fps: 10

# Color order of the images (0: BGR, 1: RGB. It is ignored if images are grayscale)

Camera.RGB: 1

**Create bash file to run all commands simultaneously (make adjustments accordingly):**

**/usr/bin/gedit ~/.bashrc**

#!/bin/bash

cat /dev/null > abc.txt

cat /dev/null > abc2.txt

rm -ff /home/romi/frames/\*

rm -ff /home/romi/ORB\_SLAM/bin/TrackLost.jpg

sourm -ff /home/romi/ORB\_SLAM/bin/Tracking.jpg

clear

echo "Deleting Images"

rm -ff /home/romi/frames/\*

echo "deleted Images"

echo "Starting ROSCORE"

roscore &

sleep 2

echo "Started ROSCORE"

cd /home/romi/minos #Following command runs MINOS

python3 -m minos.tools.ans --dataset mp3d --scene\_ids JeFG25nYj2p --env\_config pointgoal\_mp3d\_s --save\_png --width 600 --height 400 --agent\_config agent\_gridworld -s map --navmap &

cd /home/romi/ORB\_SLAM #Following command runs ORB\_SLAM after a 10 second delay

sleep 5

export ROS\_PACKAGE\_PATH=${ROS\_PACKAGE\_PATH}:/home/romi

roslaunch ORB\_SLAM ExampleGroovyOrNewer.launch &

sleep 5 #Following command runs the integration algorithm after a 10 second delay

cd /home/romi/catkin\_ws

source devel/setup.bash

rosrun merger merger\_node

rosrun merger\_node merger\_node

**Another issue is that process ORBSLAM doesn’t initialize, this error was also removed by doing some changes in g2o thirdparty dependency.**

Steps:

cd ORB\_SLAM/Thirdparty/g2o/g2o/solvers/linear\_solver\_eigen.h

In line 56, replace

***typedef Eigen::PermutationMatrix<Eigen::Dynamic, Eigen::Dynamic, SparseMatrix::Index>***

with

***typedef Eigen::PermutationMatrix<Eigen::Dynamic, Eigen::Dynamic, int>***

and then build g2o again.

[ORB\_SLAM-3] process has died [pid 3069, exit code -11, cmd /ORB\_SLAM/bin/ORB\_SLAM Data/ORBvoc.txt Data/Settings.yaml \_\_name:=