

# **INSTALACIÓN DE ORB\_SLAM2 CON ZED STEREO CAMERA**

## **ÍNDICE**

### **INSTALACIÓN DE ZED STEREO CAMERA**

**CUDA 10.0**

**ZED SDK v2.8.3**

**ZED-ROS-WRAPPER**

### **INSTALACIÓN DE ORB\_SLAM2**

#### **REQUISITOS**

**Pangolin**

**OpenCV v4.2.11**

**Eigen3 v3.2.10**

**INSTALAR ORB\_SLAM2**

**SOLUCIONAR ERRORES (build\_ros.sh)**

### **CONFIGURAR CÁMARA WEB CON ORB\_SLAM 2 (MONO)**

### **PUBLICACIÓN DE POSE DE ORB\_SLAM2 PARA CÁMARA MONO**

### **CALIBRACIÓN CÁMARA ZED**

### **CONFIGURAR CÁMARA ZED CON ORB\_SLAM2 (STEREO)**

### **PUBLICACIÓN DE POSE, GUARDADO Y CARGADO DE MAPAS DE ORB\_SLAM2 PARA CÁMARA STEREO**

### **VERIFICAR FUNCIONAMIENTO**

**ORB\_SLAM - ZED CAMERA**

**GUARDAR MAPA**

**CARGAR MAPA**

**POSE**

**ORB\_SLAM - USB\_CAM**

### **FUENTES BIBLIOGRÁFICAS**

## INSTALACIÓN DE ZED STEREO CAMERA

### CUDA 10.0

Es necesario que sea CUDA 10.0 para el Software Development Kit versión 2.8.3 de ZED Stereolabs.

[https://developer.nvidia.com/cuda-10.0-download-archive?target\\_os=Linux&target\\_arch=x86\\_64&target\\_distro=Ubuntu&target\\_version=1604&target\\_type=deblocal](https://developer.nvidia.com/cuda-10.0-download-archive?target_os=Linux&target_arch=x86_64&target_distro=Ubuntu&target_version=1604&target_type=deblocal)

Verificar que está seleccionado lo siguiente:

- Sistema Operativo: Linux.
- Arquitectura: x86\_64
- Distribución: Ubuntu
- Versión: 16.04
- Tipo de instalador: deb (local)

Se procede a descargar el instalador base y se siguen las instrucciones en la terminal para finalizar la instalación. Es posible que esto deba realizarse desde la carpeta en la que se descargó el archivo.

Al realizar la instalación saldrá un cuadro de diálogo en la terminal en el que se debe de aceptar a los términos. Tiene un botón de Ok que no es posible presionar, pero en cuanto uno presiona tab puede navegar presionando varias veces tab y seleccionar la opción que uno considere pertinente con la tecla enter.

### ZED SDK v2.8.3

Cuando ya se tiene instalado CUDA 10.0, se procede a descargar el archivo de extensión .run ( <https://www.stereolabs.com/developers/release/> ) para instalar el SDK de ZED.

La liga <https://www.stereolabs.com/docs/getting-started/installation/> contiene los pasos a seguir para realizar la instalación. Una vez instalado se pueden seguir tutoriales y hacer pruebas de los demo para empezar a entender la cámara, pero se recomienda proceder con ROS-Wrapper.

### ZED-ROS-WRAPPER

ROS-Wrapper es una herramienta que nos permite obtener información de la ZED. Esto también puede hacerse por medio de un API, pero no se ha revisado a fondo la documentación de este método. Para mayor información de API, visitar <https://www.stereolabs.com/docs/positional-tracking/using-tracking/>

ROS-Wrapper creará un nodo de forma que uno se puede suscribir a los diferentes tópicos que la ZED publica, como la información que da la cámara izquierda, la derecha o la pose de la cámara. RViz es una de las herramientas que ofrece ROS y que utilizamos con ROS-Wrapper. Para dar con esto, se siguen los pasos que hay en estos dos githubs <https://github.com/stereolabs/zed-ros-wrapper> y [https://github.com/stereolabs/zed-ros-wrapper/tree/master/zed\\_wrapper](https://github.com/stereolabs/zed-ros-wrapper/tree/master/zed_wrapper)

En caso de haber error al correr catkin\_make escribir  
catkin\_make\_isolated

## INSTALACIÓN DE ORB\_SLAM2

Vídeo de funcionamiento esperado en la ZED:

[https://www.youtube.com/watch?v=OfrPAUyp1\\_s&fbclid=IwAR1fhwybMMrsjZq-VHMSyuAYEe109GqSTWndluJPMODZktGSvjF03slulRg](https://www.youtube.com/watch?v=OfrPAUyp1_s&fbclid=IwAR1fhwybMMrsjZq-VHMSyuAYEe109GqSTWndluJPMODZktGSvjF03slulRg)

### REQUISITOS:

#### Pangolin

Instalación de Pangolin: <https://github.com/stevenlovegrove/Pangolin>

Requisitos:

- C++11
- OpenGL (Desktop / ES / ES2)
- Glew (opción: deb)
- CMake (for build environment) (opción: deb)

#### OpenCV v2.4.11

Instalación de OpenCV v2.4.11 (<https://youtu.be/2Pbog2LFoal?t=62> a partir del minuto 1:02)

- Descarga de script.sh al cual cambiaremos de nombre a opencv.sh

[http://www.dashub.org/unlv/wiki/doku.php?id=opencv\\_install\\_ubuntu](http://www.dashub.org/unlv/wiki/doku.php?id=opencv_install_ubuntu)

Nota: debemos de cambiar la primera línea de código del archivo a:

version="2.4.11"

- Colocar el archivo anterior donde usted desee, tomando en cuenta que éste generará una carpeta con el nombre OpenCV y dentro de ella una carpeta llamada opencv-2.4.11 con todos los archivos de esta versión de OpenCV. Nota: no es necesario descargar la carpeta que menciona el tutorial dentro del minuto 0:00 al 1:02.
- Continuar con el tutorial mostrado en el vídeo, terminando en el minuto 4:36

Otra alternativa es seguir los pasos del siguiente link:

- <https://gist.github.com/dynamicguy/3d1fce8dae65e765f7c4>

#### Eigen3 v3.2.10

Instalación de Eigen3 v3.2.10: [http://eigen.tuxfamily.org/index.php?title=Main\\_Page](http://eigen.tuxfamily.org/index.php?title=Main_Page)

- Escribir en una terminal: `sudo apt-get install libeigen3-dev`

Descargar archivo tar.gz, descomprimir y mover al src del workspace. seguir los siguientes pasos:

```
mkdir build
cd build
cmake ..
cmake --build .
```

## INSTALAR ORB\_SLAM2

Instalación del GitHub ORB\_SLAM2 modificado por kinglitanxia:

[https://github.com/kinglitanxia/ORB\\_SLAM2\\_MAP](https://github.com/kinglitanxia/ORB_SLAM2_MAP)

Dentro del workspace deseado, instalar el github de ORB\_SLAM2. Escribir en una terminal:  
git clone [https://github.com/kinglinter/ORB\\_SLAM2\\_MAP](https://github.com/kinglinter/ORB_SLAM2_MAP) ORB\_SLAM2

Deberá de descargar la carpeta de Vocabulary del gitHub original de raulmur y ponerlo en su carpeta de ORB\_SLAM2 ([https://github.com/raulmur/ORB\\_SLAM2](https://github.com/raulmur/ORB_SLAM2)). Después en una terminal deberá de ingresar:

```
cd ORB_SLAM2
chmod +x build.sh
./build.sh
```

Esto creará libORB\_SLAM2.so en la carpeta *lib* y creará los ejecutables **mono\_tum**, **mono\_kitti**, **rgbd\_tum**, **stereo\_kitti**, **mono\_euroc** and **stereo\_euroc** en la carpeta correspondiente a mono, rgbd y stereo dentro de *Examples*.

Para poder ejecutar los nodos ros, se debe de agregar el espacio de trabajo ROS (esto sirve para no escribir siempre el comando de export ROS\_PACKAGE\_PATH que se mostrará más adelante) Nota: cambiar **PATH** por la dirección que lleva hasta ORB\_SLAM2, por ejemplo: /home/gabriel/ORB\_SLAM2/Examples/ROS/:

Ingresa en una terminal: gedit ~/.bashrc

Después deberá de escribir hasta el final de la página mostrada lo siguiente:

```
export ROS_PACKAGE_PATH=${ROS_PACKAGE_PATH}:/PATH/ORB_SLAM2/Examples/ROS/
```

Terminando la instalación de ORB\_SLAM2, procedemos a escribir en una terminal lo siguiente. Si surge algún error al ejecutar lo siguiente, deberá de revisar la sección SOLUCIONAR ERRORES (build\_ros.sh). Ejecute el source hasta que haya ejecutado correctamente build\_ros.sh

```
cd ORB_SLAM2
chmod +x build_ros.sh
./build_ros.sh
```

### SOLUCIONAR ERRORES (build\_ros.sh)

Si obtiene un error al ejecutar ./build\_ros.sh deberá de hacer lo siguiente. Ingresa a la carpeta con el archivo CMakeLists.txt que se encuentra en la dirección **PATH**/ORB\_SLAM2/Examples/ROS/ORB\_SLAM2/ para editarlo.

Ingresa el comando \${EIGEN3\_INCLUDE\_DIR} y -lboost\_system tal como se muestra a continuación. Esta sección se encuentra antes de llegar a la parte de creación de nodos para la cámara monocular, stereo y RGB-D:

```

include_directories(
${PROJECT_SOURCE_DIR}
${PROJECT_SOURCE_DIR}/../..../
${PROJECT_SOURCE_DIR}/../..../include
${Pangolin_INCLUDE_DIRS}
${EIGEN3_INCLUDE_DIR}
)

set(LIBS
${OpenCV_LIBS}
${EIGEN3_LIBS}
${Pangolin_LIBRARIES}
${PROJECT_SOURCE_DIR}/../..../Thirdparty/DBoW2/lib/libDBoW2.so
${PROJECT_SOURCE_DIR}/../..../Thirdparty/g2o/lib/libg2o.so
${PROJECT_SOURCE_DIR}/../..../lib/libORB_SLAM2_MAP.so
-lboost_system
)

```

Vuelva a ejecutar `./build_ros.sh` para construir correctamente los nodos de ros. Si no aparece ningún error, entonces agregue el comando de source también.

## CONFIGURAR CÁMARA WEB CON ORB\_SLAM 2 (MONO)

Deberá de instalar el `usb_cam`, escriba los siguientes comandos en una terminal:

```

git clone https://github.com/ros-drivers/usb_cam.git
cd usb_cam
mkdir build
cd build
cmake ..
make

```

Dentro de `PATH/ORB_SLAM2/Examples/ROS/ORB_SLAM2/src` modificar el archivo `ros_mono.cc`, si tiene instalado el editor geany modificar las líneas de código 64 para que quede de la siguiente manera:

```
ros::Subscriber sub = nodeHandler.subscribe("/usb_cam/image_raw", 1, &ImageGrabber::GrabImage,&igb);
```

En una terminal nueva, deberá de construir `build_ros.sh` de la siguiente manera:

```

cd ORB_SLAM2
./build.sh

```

Listo, ya tiene configurado ORB\_SLAM2 para ser utilizado con la cámara web de su computadora en MONO.

Para más información, visitar <https://blog.csdn.net/KYJL888/article/details/88427486>

## PUBLICACIÓN DE POSE DE ORB\_SLAM2 PARA CÁMARA MONO

Se utilizó la orientación dada en la página:

[https://blog.csdn.net/learning\\_tortosie/article/details/81384717](https://blog.csdn.net/learning_tortosie/article/details/81384717)

Modificar el código de `ros_mono.cc` que está en `ORB_SLAM2/Examples/ROS/ORB_SLAM2/src` a lo siguiente (para ser utilizado con la cámara web):

```
/**
 * This file is part of ORB-SLAM2.
 *
 * Copyright (C) 2014-2016 Raúl Mur-Artal <raulmur at unizar dot es> (University of Zaragoza)
 * For more information see <https://github.com/raulmur/ORB_SLAM2>
 *
 * ORB-SLAM2 is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * ORB-SLAM2 is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with ORB-SLAM2. If not, see <http://www.gnu.org/licenses/>.
 */

#include<iostream>
#include<algorithm>
#include<fstream>
#include<chrono>

#include<ros/ros.h>
#include <cv_bridge/cv_bridge.h>
#include "geometry_msgs/PoseStamped.h"
#include <tf/transform_broadcaster.h>

#include<opencv2/core/core.hpp>
#include "Converter.h"

#include"../../../../include/System.h"

using namespace std;

class ImageGrabber
{
public:
    ImageGrabber(ORB_SLAM2::System* pSLAM):mpSLAM(pSLAM){}

    void GrabImage(const sensor_msgs::ImageConstPtr& msg);

    void PublishPose(cv::Mat Tcw); //

    ORB_SLAM2::System* mpSLAM;
    ros::Publisher* pPosPub; //
};

//ros::Publisher pPosPub;

void ImageGrabber::PublishPose(cv::Mat Tcw)
{
    geometry_msgs::PoseStamped poseMSG;
    if(!Tcw.empty())
    {
        cv::Mat Rwc = Tcw.rowRange(0,3).colRange(0,3).t();
        cv::Mat twc = -Rwc*Tcw.rowRange(0,3).col(3);
        vector<float> q = ORB_SLAM2::Converter::toQuaternion(Rwc);

        /*
```

```

cv::Mat Rwc = Tcw.rowRange(0,3).colRange(0,3).t();
cv::Mat twc = -Rwc*Tcw.rowRange(0,3).col(3);
tf::Matrix3x3 M(Rwc.at<float>(0,0),Rwc.at<float>(0,1),Rwc.at<float>(0,2),
                Rwc.at<float>(1,0),Rwc.at<float>(1,1),Rwc.at<float>(1,2),
                Rwc.at<float>(2,0),Rwc.at<float>(2,1),Rwc.at<float>(2,2));
tf::Vector3 V(twc.at<float>(0), twc.at<float>(1), twc.at<float>(2));

tf::Transform tfTcw(M,V);

//mTfBr.sendTransform(tf::StampedTransform(tfTcw,ros::Time::now(),
"ORB_SLAM/Camera"));
*/
poseMSG.pose.position.x = twc.at<float>(0);
poseMSG.pose.position.y = twc.at<float>(2);
poseMSG.pose.position.z = twc.at<float>(1);
poseMSG.pose.orientation.x = q[0];
poseMSG.pose.orientation.y = q[1];
poseMSG.pose.orientation.z = q[2];
poseMSG.pose.orientation.w = q[3];
poseMSG.header.frame_id = "VSLAM";
poseMSG.header.stamp = ros::Time::now();
//cout << "PublishPose position.x = " << poseMSG.pose.position.x << endl;

(pPosPub)->publish(poseMSG);

//mlbLost.push_back(mState==LOST);
}
}

int main(int argc, char **argv)
{
    ros::init(argc, argv, "Mono");
    ros::start();
    //bool bReuseMap = false; //
    if(argc != 3)
    {
        cerr << endl << "Usage: rosrn ORB_SLAM2 Mono path_to_vocabulary path_to_settings" << endl;
        ros::shutdown();
        return 1;
    }

    // Create SLAM system. It initializes all system threads and gets ready to process frames.
    //if (!strcmp(argv[3], "true")) //
    //{ //
    //    bReuseMap = true; //
    //}
    ORB_SLAM2::System SLAM(argv[1],argv[2],ORB_SLAM2::System::MONOCULAR,true); //

    ImageGrabber igb(&SLAM);

    ros::NodeHandle nodeHandler;
    ros::Subscriber sub = nodeHandler.subscribe("usb_cam/image_raw", 1,
&ImageGrabber::GrabImage,&igb);
    ros::Publisher PosPub = nodeHandler.advertise<geometry_msgs::PoseStamped>("ORB_SLAM/pose", 5); //
    igb.pPosPub = &(PosPub); //
    ros::spin();

    // Stop all threads
    SLAM.Shutdown();

    // Save map //
    //SLAM.SaveMap("Slam_latest_Map.bin"); //

    // Save camera trajectory
    SLAM.SaveKeyFrameTrajectoryTUM("KeyFrameTrajectory.txt");
    ros::shutdown();

    return 0;
}

void ImageGrabber::GrabImage(const sensor_msgs::ImageConstPtr& msg)

```

```

{
    // Copy the ros image message to cv::Mat.
    cv_bridge::CvImageConstPtr cv_ptr;
    try
    {
        cv_ptr = cv_bridge::toCvShare(msg);
    }
    catch (cv_bridge::Exception& e)
    {
        ROS_ERROR("cv_bridge exception: %s", e.what());
        return;
    }

    cv::Mat Tcw= mpSLAM->TrackMonocular(cv_ptr->image,cv_ptr->header.stamp.toSec()); //
    PublishPose(Tcw); //
}

```

Una vez ajustado correctamente en el archivo `ros_mono.cc` procedemos a escribir dentro de una terminal dirigiéndonos dentro de `ORB_SLAM2/`

```

cd ORB_SLAM2
./build_ros.sh

```

## CALIBRACIÓN CÁMARA ZED

Se debe hacer una calibración con las matrices D, K, R y P, factores de corrección y valores de cx, cy, fx y fy. Estas matrices son necesarias para tener una correcta calibración de la cámara y que las nubes de puntos se den de forma correcta. Esto se podrá notar cuando se creen los mapas, pues los puntos han de describir formas similares a las que se observaron mientras se creó el mapa.

Para generar una correcta calibración, es importante seguir los pasos obtenidos de [http://wiki.ros.org/camera\\_calibration/Tutorials/StereoCalibration](http://wiki.ros.org/camera_calibration/Tutorials/StereoCalibration). Se debe descargar el [archivo](#) e imprimirlo. Es importante que se mida el tamaño de los cuadros ya impresos.

En una terminal, ingrese los siguientes comandos.

```

rosdep install camera_calibration
rosmake camera_calibration

```

Posteriormente, se han de publicar los tópicos que ofrece la ZED con el siguiente comando.

```

rostopic list

```



```
gabriel@gabriel-Lenovo-Y720-15IKB: ~
gabriel@gabriel-Lenovo-Y720-15IKB:~$ rostopic list
/diagnostics
/rosout
/rosout_agg
/tf
/tf_static
/zed/joint_states
/zed/zed_node/confidence/camera_info
/zed/zed_node/confidence/confidence_image
/zed/zed_node/confidence/confidence_image/compressed
/zed/zed_node/confidence/confidence_image/compressed/parameter_descriptions
/zed/zed_node/confidence/confidence_image/compressed/parameter_updates
/zed/zed_node/confidence/confidence_image/compressedDepth
/zed/zed_node/confidence/confidence_image/compressedDepth/parameter_descriptions
/zed/zed_node/confidence/confidence_image/compressedDepth/parameter_updates
/zed/zed_node/confidence/confidence_image/theora
/zed/zed_node/confidence/confidence_image/theora/parameter_descriptions
/zed/zed_node/confidence/confidence_image/theora/parameter_updates
/zed/zed_node/confidence/confidence_map
/zed/zed_node/depth/camera_info
/zed/zed_node/depth/depth_registered
/zed/zed_node/depth/depth_registered/compressed
/zed/zed_node/depth/depth_registered/compressed/parameter_descriptions
/zed/zed_node/depth/depth_registered/compressed/parameter_updates
```

Es necesario escoger los tópicos correctos para introducirlos en el siguiente comando. Estos tópicos son `/zed/zed_node/<camera_side>/image_rect_color` como se muestra en el ejemplo ejemplo del siguiente comando.

```
roslaunch camera_calibration cameracalibrator.py --approximate 0.1 --size 8x6 --square 0.022
--no-service-check right:=/zed/zed_node/right/image_rect_color
left:=/zed/zed_node/left/image_rect_color right_camera:=/zed/zed_node/right
left_camera:=/zed/zed_node/left
```

Del tutorial de la liga anterior, se agregó la línea, `--no-service-check`, ya que sin esta línea no nos permite continuar. Ya que se ejecutó el comando, se sigue el resto del tutorial con normalidad.

Finalmente, dentro de la carpeta `/ORB_SLAM2/Examples/Stereo` modifique el archivo `EuRoC.yaml` y guárdelo como `ZED.yaml`, ingresando todos los valores de las matrices obtenidos con anterioridad en la terminal o en el archivo generado dentro de la carpeta que indica el tutorial. Deberá de quedar de la siguiente forma:

```
%YAML:1.0

#-----
# Camera Parameters. Adjust them!
#-----

# Camera calibration and distortion parameters (OpenCV)
#Camera.fx: 435.2046959714599
#Camera.fy: 435.2046959714599
#Camera.cx: 367.4517211914062
#Camera.cy: 252.2008514404297

#Camera.k1: 0.0
#Camera.k2: 0.0
#Camera.p1: 0.0
#Camera.p2: 0.0

Camera.fx: 695.637636
Camera.fy: 695.417728
```

Camera.cx: 605.594646  
Camera.cy: 375.427060

Camera.k1: 0.008185  
Camera.k2: -0.015304  
Camera.p1: -0.000139  
Camera.p2: -0.002273

Camera.width: 1280  
Camera.height: 720

# Camera frames per second  
Camera.fps: 60.0

# stereo baseline times fx  
#Camera.bf: 47.90639384423901  
Camera.bf: 84

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

# Close/Far threshold. Baseline times.  
ThDepth: 35

#-----  
# Stereo Rectification. Only if you need to pre-rectify the images.  
# Camera.fx, .fy, etc must be the same as in LEFT.P  
#-----

LEFT.height: 720  
LEFT.width: 1280  
LEFT.D: !!opencv-matrix  
  rows: 1  
  cols: 5  
  dt: d  
  data: [0.009746563829491045, -0.012801547992642019, 0.0011337134547784087, -0.002838599465352049, 0.0]  
LEFT.K: !!opencv-matrix  
  rows: 3  
  cols: 3  
  dt: d  
  data: [698.4638019932769, 0.0, 601.9773645876436, 0.0, 695.8599917510946, 379.30240046007236, 0.0, 0.0, 1.0]  
LEFT.R: !!opencv-matrix  
  rows: 3  
  cols: 3  
  dt: d  
  data: [0.9999986303614505, -0.0016456148799013591, -0.00017671131820226399, 0.0016460533417453596, 0.9999954941015149, 0.002510435233817601, 0.00017257931238320303, -0.00251072267168458, 0.9999968332390091]  
LEFT.P: !!opencv-matrix  
  rows: 3  
  cols: 4  
  dt: d  
  data: [700.0926444637008, 0.0, 593.1592178344727, 0.0, 0.0, 700.0926444637008, 378.26103591918945, 0.0, 0.0, 0.0, 1.0, 0.0]

RIGHT.height: 720  
RIGHT.width: 1280  
RIGHT.D: !!opencv-matrix  
  rows: 1  
  cols: 5  
  dt: d  
  data: [0.008185238123726804, -0.01530425089961923, -0.0001392957961312843, -0.0022725715343652774, 0.0]  
RIGHT.K: !!opencv-matrix  
  rows: 3  
  cols: 3  
  dt: d  
  data: [695.637635943541, 0.0, 605.5946457057464, 0.0, 695.4177284755905, 375.42706032463053, 0.0, 0.0, 1.0]  
RIGHT.R: !!opencv-matrix  
  rows: 3

```

cols: 3
dt: d
data: [0.9999834128657124, -0.0015902280746336018, 0.005535807810315697, 0.0016041212079382044,
0.9999955729935311, -0.0025061501330988054, -0.00553179795295781, 0.0025149886699615993,
0.9999815368512548]
RIGHT.P: !!opencv-matrix
rows: 3
cols: 4
dt: d
data: [700.0926444637008, 0.0, 593.1592178344727, -73.70401959384539, 0.0, 700.0926444637008,
378.26103591918945, 0.0, 0.0, 0.0, 1.0, 0.0]

#-----
# ORB Parameters
#-----

# ORB Extractor: Number of features per image
ORBExtractor.nFeatures: 2000

# ORB Extractor: Scale factor between levels in the scale pyramid
ORBExtractor.scaleFactor: 1.2

# ORB Extractor: Number of levels in the scale pyramid
ORBExtractor.nLevels: 8

# ORB Extractor: Fast threshold
# Image is divided in a grid. At each cell FAST are extracted imposing a minimum response.
# Firstly we impose iniThFAST. If no corners are detected we impose a lower value minThFAST
# You can lower these values if your images have low contrast
ORBExtractor.iniThFAST: 20 #20
ORBExtractor.minThFAST: 7 #7

#-----
# Viewer Parameters
#-----
Viewer.KeyFrameSize: 0.6 #0.05
Viewer.KeyFrameLineWidth: 2 #1
Viewer.GraphLineWidth: 1 #0.9
Viewer.PointSize: 2 #2
Viewer.CameraSize: 0.7 #0.08
Viewer.CameraLineWidth: 3 #3
Viewer.ViewpointX: 0 #0
Viewer.ViewpointY: -100 #-0.7
Viewer.ViewpointZ: -0.1 #-1.8
Viewer.ViewpointF: 2000 #500

```

Dentro de estos parámetros, es importante mencionar:

- Se pueden cambiar los fps en “Camera.fps”.
- Se puede cambiar la cantidad nube de puntos leídos en “ORBExtractor.nFeatures”. En varios foros se recomienda que el máximo sea de 3000, pero la cámara se pone lenta. Se ha elegido el valor de 2000, porque se muestra estable el programa.
- También es posible cambiar el Threshold para cuando se están en un ambiente donde se ve poco contraste. Esto se hace en las líneas que dicen “ORBExtractor.iniThFAST” y “ORBExtractor.minThFAST”. Favor de leer el comentario del código para saber cómo modificar los parámetros.

Siempre que realice cambios deberá de correr dentro de ORB\_SLAM el archivo build\_ros.sh de la siguiente manera:

```

cd ORB_SLAM2
./build_ros.sh

```

## CONFIGURAR CÁMARA ZED CON ORB\_SLAM2 (STEREO)

Dentro de **PATH**/ORB\_SLAM2/Examples/ROS/ORB\_SLAM2/src modificar el archivo `ros_stereo.cc`, si tiene instalado el editor geany modificar las líneas de código 118 y 119 para que queden de la siguiente manera:

```
message_filters::Subscriber<sensor_msgs::Image> left_sub(nh, "/zed/zed_node/left/image_rect_color", 1);  
message_filters::Subscriber<sensor_msgs::Image> right_sub(nh, "/zed/zed_node/right/image_rect_color", 1);
```

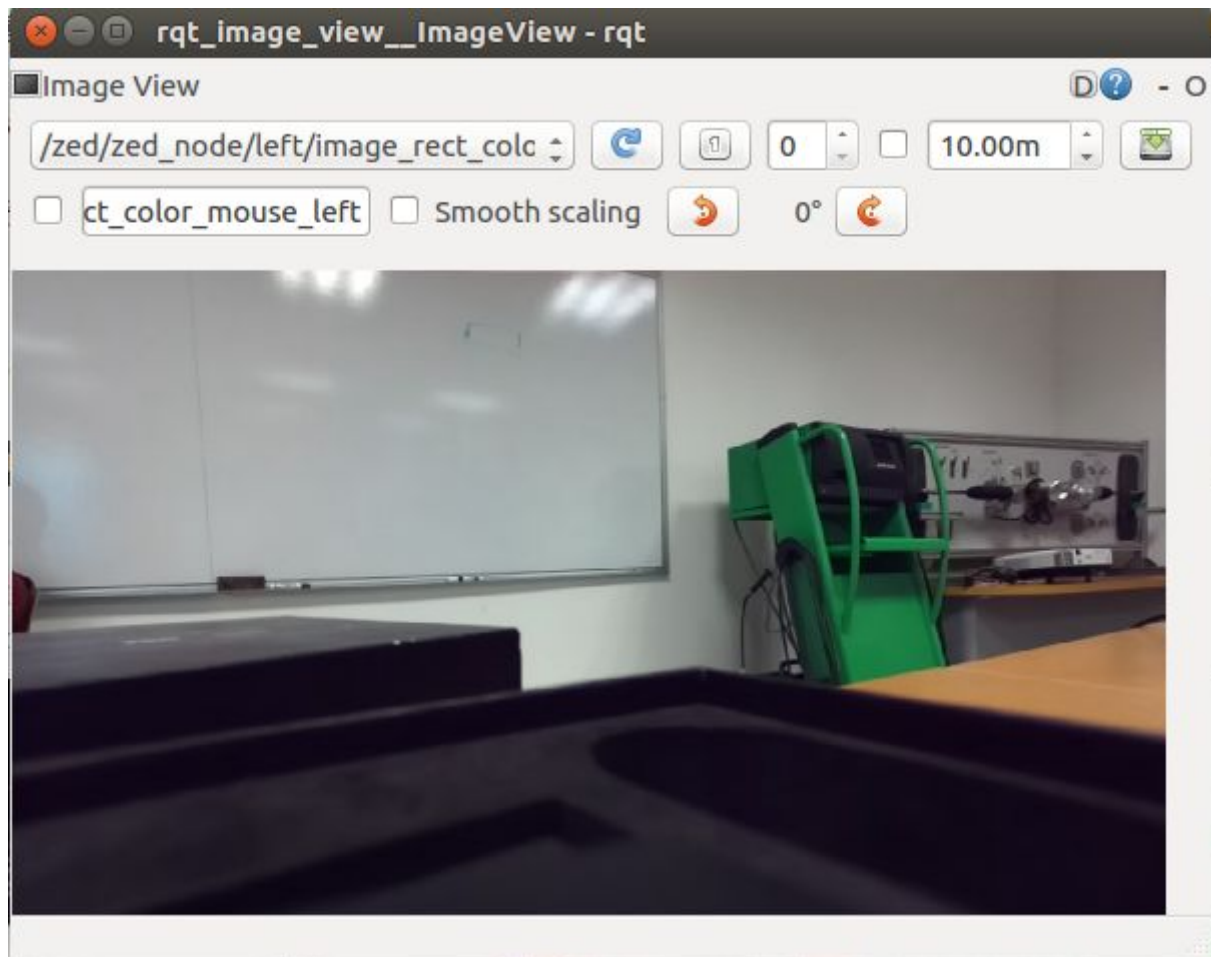
Deberá de verificar si tiene la misma dirección del nodo que corresponde a la cámara izquierda y derecha de la cámara zed, para ello en una nueva terminal debe de correr el código:

```
roslaunch zed_wrapper zed.launch
```

En otra terminal ingresamos el código:

```
rqt_image_view
```

Con esta terminal podremos ver el nodo correspondiente a la cámara izquierda y derecha de la ZED. También podemos utilizar estas direcciones para ser utilizadas en la cámara MONO, donde solamente ocuparemos una de las dos cámaras de la ZED Stereo Camera.



Una vez ajustado correctamente la dirección de la cámara ZED en el archivo `ros_stereo.cc` procedemos a escribir dentro de una terminal dirigiéndonos dentro de ORB\_SLAM2/

```
cd ORB_SLAM2
./build_ros.sh
```

Listo, ya tenemos configurado todo para correr el programa ORB\_SLAM2 junto con la cámara ZED. Esto también se puede verificar al escribir en una terminal: `rostopic list`

## **PUBLICACIÓN DE POSE, GUARDAR Y CARGAR MAPAS EN ORB\_SLAM2 PARA CÁMARA STEREO**

Partiendo del funcionamiento de la sección anterior, se hicieron las siguientes modificaciones al archivo llamado `ros_stereo.cc` que está en `ORB_SLAM2/Examples/ROS/ORB_SLAM2/src`

```
/**
 * This file is part of ORB-SLAM2.
 *
 * Copyright (C) 2014-2016 Raúl Mur-Artal <raulmur at unizar dot es> (University of Zaragoza)
 * For more information see <https://github.com/raulmur/ORB_SLAM2>
 *
 * ORB-SLAM2 is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
```

```

* (at your option) any later version.
*
* ORB-SLAM2 is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with ORB-SLAM2. If not, see <http://www.gnu.org/licenses/>.
*/

#include <iostream>
#include <algorithm>
#include <fstream>
#include <chrono>

#include <ros/ros.h>
#include <cv_bridge/cv_bridge.h>
#include <message_filters/subscriber.h>
#include <message_filters/time_synchronizer.h>
#include <message_filters/sync_policies/approximate_time.h>

#include <cv_bridge/cv_bridge.h>
#include "geometry_msgs/PoseStamped.h"
#include <tf/transform_broadcaster.h>

#include <opencv2/core/core.hpp>
#include "Converter.h"

#include "../include/System.h"

using namespace std;

class ImageGrabber
{
public:
    ImageGrabber(ORB_SLAM2::System* pSLAM):mpSLAM(pSLAM){}

    void GrabStereo(const sensor_msgs::ImageConstPtr& msgLeft,const sensor_msgs::ImageConstPtr& msgRight);

    void PublishPose(cv::Mat Tcw); //
    ORB_SLAM2::System* mpSLAM;
    bool do_rectify;
    cv::Mat M1l,M2l,M1r,M2r;
    ros::Publisher* pPosPub; //
};

void ImageGrabber::PublishPose(cv::Mat Tcw)
{
    geometry_msgs::PoseStamped poseMSG;
    if(!Tcw.empty())
    {
        cv::Mat Rwc = Tcw.rowRange(0,3).colRange(0,3).t();
        cv::Mat twc = -Rwc*Tcw.rowRange(0,3).col(3);

        vector<float> q = ORB_SLAM2::Converter::toQuaternion(Rwc);

        /*
        cv::Mat Rwc = Tcw.rowRange(0,3).colRange(0,3).t();
        cv::Mat twc = -Rwc*Tcw.rowRange(0,3).col(3);
        tf::Matrix3x3 M(Rwc.at<float>(0,0),Rwc.at<float>(0,1),Rwc.at<float>(0,2),
                        Rwc.at<float>(1,0),Rwc.at<float>(1,1),Rwc.at<float>(1,2),
                        Rwc.at<float>(2,0),Rwc.at<float>(2,1),Rwc.at<float>(2,2));
        tf::Vector3 V(twc.at<float>(0), twc.at<float>(1), twc.at<float>(2));

        tf::Transform tfTcw(M,V);

```

```

        //mTfBr.sendTransform(tf::StampedTransform(tfTcw,ros::Time::now(),
"ORB_SLAM/Camera"));
        */
        poseMSG.pose.position.x = twc.at<float>(0);
        poseMSG.pose.position.y = twc.at<float>(2);
        poseMSG.pose.position.z = twc.at<float>(1);
        poseMSG.pose.orientation.x = q[0];
        poseMSG.pose.orientation.y = q[1];
        poseMSG.pose.orientation.z = q[2];
        poseMSG.pose.orientation.w = q[3];
        poseMSG.header.frame_id = "VSLAM_Stereo";
        poseMSG.header.stamp = ros::Time::now();
        //cout << "PublishPose position.x = " << poseMSG.pose.position.x << endl;

        (pPosPub)->publish(poseMSG);

        //mlbLost.push_back(mState==LOST);
    }
}

int main(int argc, char **argv)
{
    ros::init(argc, argv, "RGBD");
    ros::start();

    if(argc != 6)
    {
        cerr << endl << "Usage: rosrund ORB_SLAM2 Stereo path_to_vocabulary path_to_settings do_rectify
path_to_map bOnlyTracking" << endl;
        ros::shutdown();
        return 1;
    }

    // Create SLAM system. It initializes all system threads and gets ready to process frames.
    ORB_SLAM2::System SLAM(argv[1],argv[2],ORB_SLAM2::System::STEREO,true, bool(std::string(argv[5])
== "true"));
    /*****Load Map*****/
    std::string OnlyTracking = std::string(argv[5]);
    if (OnlyTracking == "true")
    {
        SLAM.LoadMap("map.bin");
        std::cout << "---bOnlyTracking: " << OnlyTracking << std::endl;
    }
    ImageGrabber igb(&SLAM);

    stringstream ss(argv[3]);
    ss >> boolalpha >> igb.do_rectify;

    if(igb.do_rectify)
    {
        // Load settings related to stereo calibration
        cv::FileStorage fsSettings(argv[2], cv::FileStorage::READ);
        if(!fsSettings.isOpened())
        {
            cerr << "ERROR: Wrong path to settings" << endl;
            return -1;
        }

        cv::Mat K_l, K_r, P_l, P_r, R_l, R_r, D_l, D_r;
        fsSettings["LEFT.K"] >> K_l;
        fsSettings["RIGHT.K"] >> K_r;

        fsSettings["LEFT.P"] >> P_l;
        fsSettings["RIGHT.P"] >> P_r;

        fsSettings["LEFT.R"] >> R_l;
        fsSettings["RIGHT.R"] >> R_r;

        fsSettings["LEFT.D"] >> D_l;
        fsSettings["RIGHT.D"] >> D_r;
    }
}

```

```

        int rows_l = fsSettings["LEFT.height"];
        int cols_l = fsSettings["LEFT.width"];
        int rows_r = fsSettings["RIGHT.height"];
        int cols_r = fsSettings["RIGHT.width"];

        if(K_l.empty() || K_r.empty() || P_l.empty() || P_r.empty() || R_l.empty() || R_r.empty() ||
        D_l.empty() || D_r.empty() ||
            rows_l==0 || rows_r==0 || cols_l==0 || cols_r==0)
        {
            cerr << "ERROR: Calibration parameters to rectify stereo are missing!" << endl;
            return -1;
        }

        cv::initUndistortRectifyMap(K_l,D_l,R_l,P_l.rowRange(0,3).colRange(0,3),cv::Size(cols_l, rows_l),CV_32F,igb.
        M1l,igb.M2l);

        cv::initUndistortRectifyMap(K_r,D_r,R_r,P_r.rowRange(0,3).colRange(0,3),cv::Size(cols_r, rows_r),CV_32F,igb.
        M1r,igb.M2r);
    }

    ros::NodeHandle nh;

    message_filters::Subscriber<sensor_msgs::Image> left_sub(nh,
"/zed/zed_node/left/image_rect_color", 1);
    message_filters::Subscriber<sensor_msgs::Image> right_sub(nh,
"/zed/zed_node/right/image_rect_color", 1);
    typedef message_filters::sync_policies::ApproximateTime<sensor_msgs::Image, sensor_msgs::Image>
sync_pol;
    message_filters::Synchronizer<sync_pol> sync(sync_pol(10), left_sub,right_sub);
    sync.registerCallback(boost::bind(&ImageGrabber::GrabStereo,&igb,_1,_2));

    ros::Publisher PosPub = nh.advertise<geometry_msgs::PoseStamped>("ORB_SLAM2/pose", 5); //
    igb.pPosPub = &(PosPub); //

    ros::spin();

    // Stop all threads
    SLAM.Shutdown();

    /*****Save Map*****/
    if (OnlyTracking == "false")
        SLAM.SaveMap("./map.bin");

    // Save camera trajectory
    SLAM.SaveKeyFrameTrajectoryTUM("KeyFrameTrajectory_TUM_Format.txt");
    SLAM.SaveTrajectoryTUM("FrameTrajectory_TUM_Format.txt");
    SLAM.SaveTrajectoryKITTI("FrameTrajectory_KITTI_Format.txt");

    ros::shutdown();

    return 0;
}

void ImageGrabber::GrabStereo(const sensor_msgs::ImageConstPtr& msgLeft,const sensor_msgs::ImageConstPtr&
msgRight)
{
    // Copy the ros image message to cv::Mat.
    cv_bridge::CvImageConstPtr cv_ptrLeft;
    try
    {
        cv_ptrLeft = cv_bridge::toCvShare(msgLeft);
    }
    catch (cv_bridge::Exception& e)
    {
        ROS_ERROR("cv_bridge exception: %s", e.what());
        return;
    }

    cv_bridge::CvImageConstPtr cv_ptrRight;

```



```

try
{
cv_ptrRight = cv_bridge::toCvShare(msgRight);
}
catch (cv_bridge::Exception& e)
{
ROS_ERROR("cv_bridge exception: %s", e.what());
return;
}

if(do_rectify)
{
cv::Mat imLeft, imRight;
cv::remap(cv_ptrLeft->image, imLeft, M1l, M2l, cv::INTER_LINEAR);
cv::remap(cv_ptrRight->image, imRight, M1r, M2r, cv::INTER_LINEAR);
cv::Mat Tcw= mpSLAM->TrackStereo(imLeft, imRight, cv_ptrLeft->header.stamp.toSec()); //
PublishPose(Tcw); //
}
else
{
cv::Mat
mpSLAM->TrackStereo(cv_ptrLeft->image, cv_ptrRight->image, cv_ptrLeft->header.stamp.toSec()); // Tcw=
PublishPose(Tcw); //
}
}

```

Una vez ajustado correctamente la dirección de la cámara ZED en el archivo `ros_stereo.cc` procedemos a escribir dentro de una terminal dirigiéndonos dentro de `ORB_SLAM2/`

```
cd ORB_SLAM2
```

```
./build_ros.sh
```

## VERIFICAR FUNCIONAMIENTO

### ORB\_SLAM - ZED CAMERA

Ingresar los siguientes comandos en la terminal correspondiente:

En la terminal 1 ingresar el comando:

roscore

```
roscore http://gabriel-Lenovo-Y720-15IKB:11311/
gabriel@gabriel-Lenovo-Y720-15IKB:~$ roscore
... logging to /home/gabriel/.ros/log/b168eb3c-e475-11e9-b7dd-9822efdd8b65/ros-launch-gabriel-Lenovo-Y720-15IKB-4596.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://gabriel-Lenovo-Y720-15IKB:33259/
ros_comm version 1.12.14

SUMMARY
=====
```

En la terminal 2 ingresar el comando:

roslaunch zed\_wrapper zed.launch

```
/home/gabriel/ZED/src/zed-ros-wrapper/zed_wrapper/launch/zed.launch http://localhost:11311
gabriel@gabriel-Lenovo-Y720-15IKB:~$ roslaunch zed_wrapper zed.launch
... logging to /home/gabriel/.ros/log/b168eb3c-e475-11e9-b7dd-9822efdd8b65/ros-launch-gabriel-Lenovo-Y720-15IKB-4704.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://gabriel-Lenovo-Y720-15IKB:42529/

SUMMARY
=====

PARAMETERS
* /roscpp: kinetic
* /rosversion: 1.12.14
* /zed/zed_description: <?xml version="1....
* /zed/zed_node/auto_exposure: True
* /zed/zed_node/camera_model: zed
* /zed/zed_node/confidence: 100
* /zed/zed_node/depth/confidence_root: confidence
* /zed/zed_node/depth/depth_stabilization: 1
* /zed/zed_node/depth/depth_topic_root: depth
* /zed/zed_node/depth/disparity_topic: disparity/dispari...
* /zed/zed_node/depth/min_depth: 0.3
```

## GUARDAR MAPA

En la terminal 3 ingresar los siguientes comandos dentro de la carpeta cd ORB\_SLAM2:

```
cd ORB_SLAM2
```

```
roslaunch ORB_SLAM2 Stereo Vocabulary/ORBvoc.txt Examples/Stereo/ZED2.1.yaml true false false
```

En la instrucción anterior, el primer true se refiere a hacer una rectificación. El primer false se refiere a si se desea cargar el mapa. El último false se refiere a si se desea activar el modo localización.

```
gabriel@gabriel-Lenovo-Y720-15IKB: ~/ORB_SLAM2
gabriel@gabriel-Lenovo-Y720-15IKB:~$ cd ORB_SLAM2
gabriel@gabriel-Lenovo-Y720-15IKB:~/ORB_SLAM2$ roslaunch ORB_SLAM2 Stereo Vocabulary/ORBvoc.txt Examples/Stereo/ZED2.1.yaml true false false

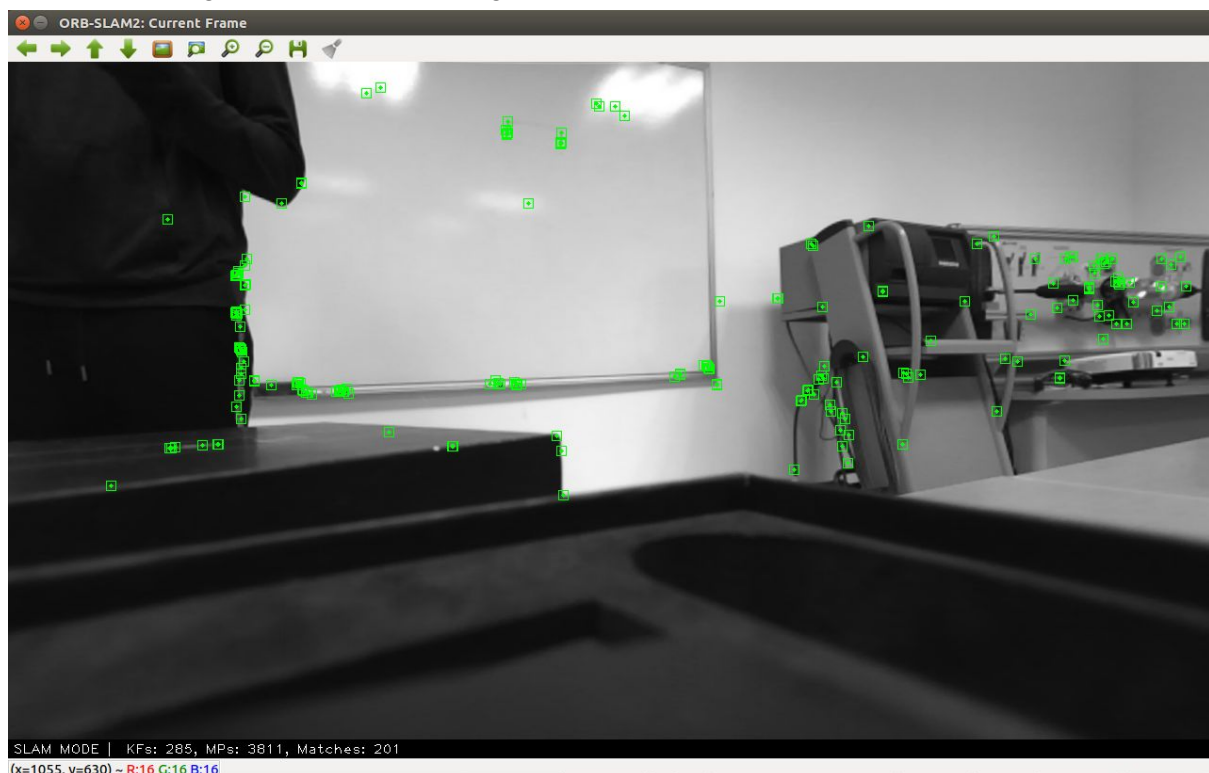
ORB-SLAM2 Copyright (C) 2014-2016 Raul Mur-Artal, University of Zaragoza.
This program comes with ABSOLUTELY NO WARRANTY;
This is free software, and you are welcome to redistribute it
under certain conditions. See LICENSE.txt.

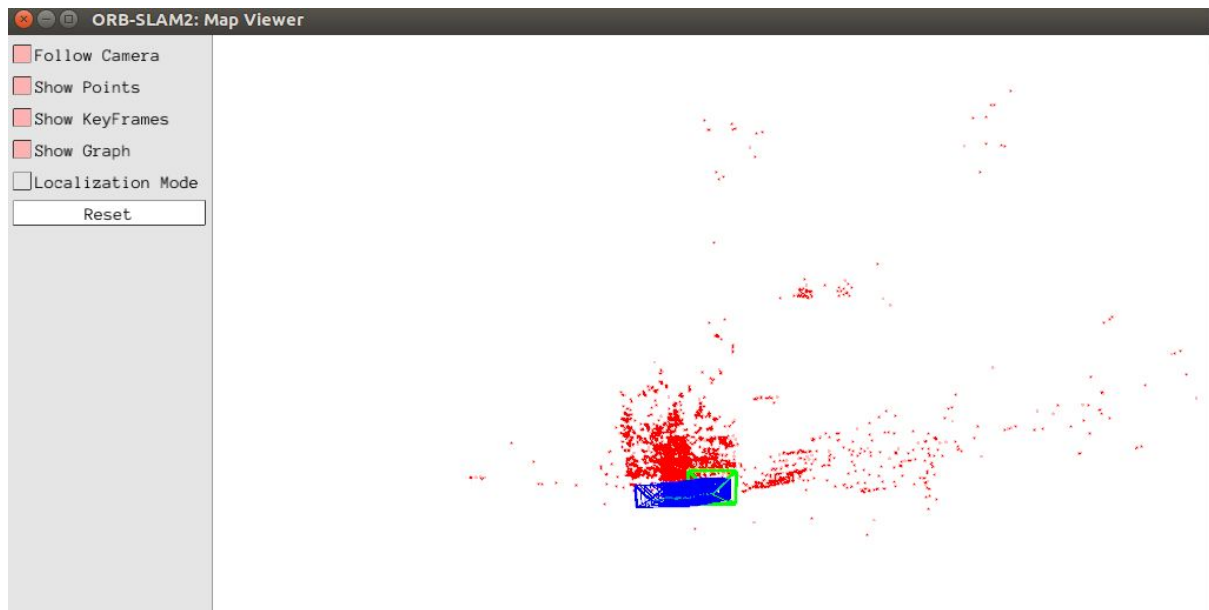
Input sensor was set to: Stereo

Loading ORB Vocabulary. This could take a while...
Vocabulary loaded!

Camera Parameters:
- fx: 699.883
- fy: 699.883
```

Obtendrás el siguiente resultado al ingresar todo lo de la terminal 3:





Una vez haya terminado de mapear, deberá de oprimir el comando `ctrl + c` en la terminal donde corrió ORB\_SLAM2, para que así el mapa pueda guardarse tal como se muestra a continuación (Save Map to ./map.bin):

```

gabriel@gabriel-Lenovo-Y720-15IKB: ~/ORB_SLAM2
Depth Threshold (Close/Far Points): 4.8008
-----KeyFrame nNextId: 0
New map created with 352 points
---Initialize Done!--
^C-----
Save Map to: ./map.bin
The number of MapPoints is: 352
The number of KeyFrame: 1
Map Saving finished!

Saving keyframe trajectory to KeyFrameTrajectory_TUM_Format.txt ...
trajectory saved!

Saving camera trajectory to FrameTrajectory_TUM_Format.txt ...
trajectory saved!

Saving camera trajectory to FrameTrajectory_KITTI_Format.txt ...
trajectory saved!
QObject::~QObject: Timers cannot be stopped from another thread
Violación de segmento ('core' generado)
gabriel@gabriel-Lenovo-Y720-15IKB:~/ORB_SLAM2$

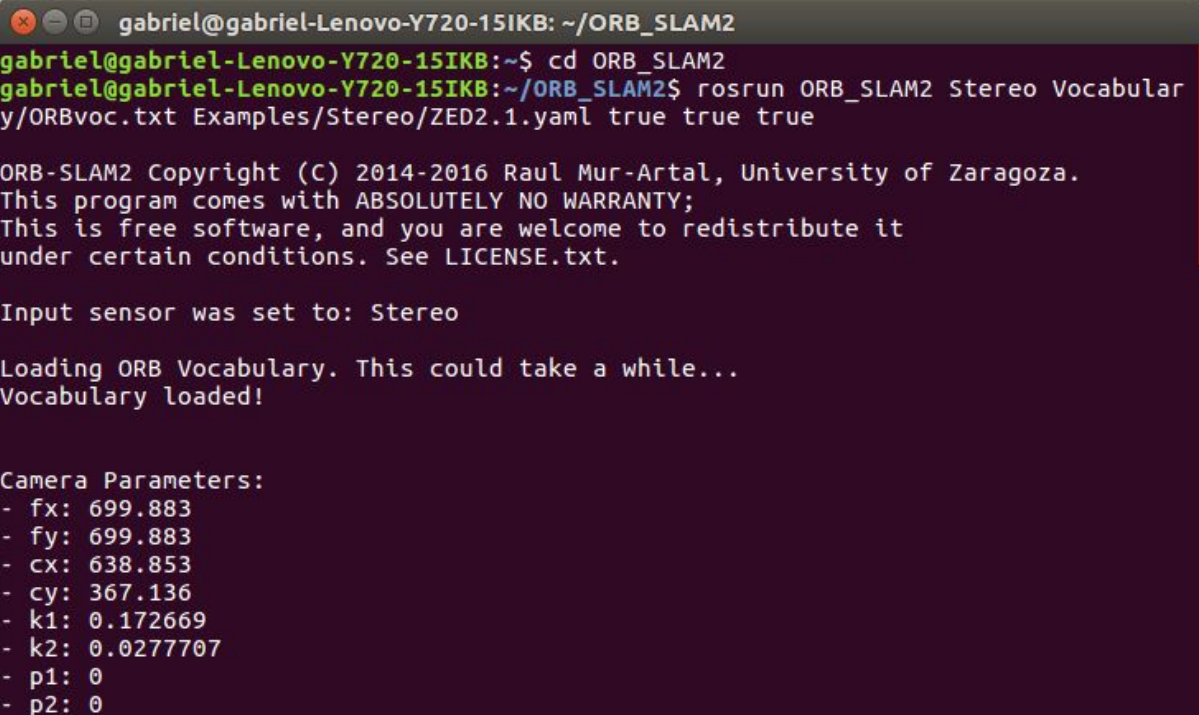
```

## CARGAR MAPA

Con la terminal 1 y 2 inicializadas, deberá de ejecutar en la terminal 3 el siguiente comando:

```
cd ORB_SLAM2
roslaunch ORB_SLAM2 Stereo Vocabulary/ORBvoc.txt Examples/Stereo/ZED.yaml true true true
```

Terminal 3:



```
gabriel@gabriel-Lenovo-Y720-15IKB: ~/ORB_SLAM2
gabriel@gabriel-Lenovo-Y720-15IKB:~$ cd ORB_SLAM2
gabriel@gabriel-Lenovo-Y720-15IKB:~/ORB_SLAM2$ roslaunch ORB_SLAM2 Stereo Vocabulary
y/ORBvoc.txt Examples/Stereo/ZED2.1.yaml true true true

ORB-SLAM2 Copyright (C) 2014-2016 Raul Mur-Artal, University of Zaragoza.
This program comes with ABSOLUTELY NO WARRANTY;
This is free software, and you are welcome to redistribute it
under certain conditions. See LICENSE.txt.

Input sensor was set to: Stereo

Loading ORB Vocabulary. This could take a while...
Vocabulary loaded!

Camera Parameters:
- fx: 699.883
- fy: 699.883
- cx: 638.853
- cy: 367.136
- k1: 0.172669
- k2: 0.0277707
- p1: 0
- p2: 0
```

Esto permitirá cargar el mapa que haya realizado con anterioridad, o cualquier otro mapa guardado con el nombre map.bin dentro de la carpeta de ORB\_SLAM2.

## ORB\_SLAM2 - USB\_CAM

Ingresa los siguientes comandos en cada terminal:

Terminal 1:

```
roscore
```

Terminal 2:

```
roslaunch usb_cam usb_cam-test.launch
```



```
/opt/ros/kinetic/share/usb_cam/launch/usb_cam-test.launch http://localhost:11311
gabriel@gabriel-Lenovo-Y720-15IKB:~$ roslaunch usb_cam usb_cam-test.launch
... logging to /home/gabriel/.ros/log/eb131d02-ec64-11e9-bb4c-9822efdd8b65/rosla
unch-gabriel-Lenovo-Y720-15IKB-25903.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://gabriel-Lenovo-Y720-15IKB:41629/

SUMMARY
=====

PARAMETERS
* /image_view/autosize: True
* /roscdistro: kinetic
* /rosversion: 1.12.14
* /usb_cam/camera_frame_id: usb_cam
* /usb_cam/image_height: 480
* /usb_cam/image_width: 640
* /usb_cam/io_method: mmap
* /usb_cam/pixel_format: yuyv
* /usb_cam/video_device: /dev/video0

NODES
```

Terminal 3:

cd ORB\_SLAM2

roslaunch ORB\_SLAM2 Mono Vocabulary/ORBvoc.txt Examples/Monocular/EuRoC.yaml

```
gabriel@gabriel-Lenovo-Y720-15IKB: ~/ORB_SLAM2
cgabriel@gabriel-Lenovo-Y720-15IKB:~$ cd ORB_SLAM2
gabriel@gabriel-Lenovo-Y720-15IKB:~/ORB_SLAM2$ roslaunch ORB_SLAM2 Mono Vocabulary/
ORBvoc.txt Examples/Monocular/EuRoC.yaml

ORB-SLAM2 Copyright (C) 2014-2016 Raul Mur-Artal, University of Zaragoza.
This program comes with ABSOLUTELY NO WARRANTY;
This is free software, and you are welcome to redistribute it
under certain conditions. See LICENSE.txt.

Input sensor was set to: Monocular

Loading ORB Vocabulary. This could take a while...
Vocabulary loaded!

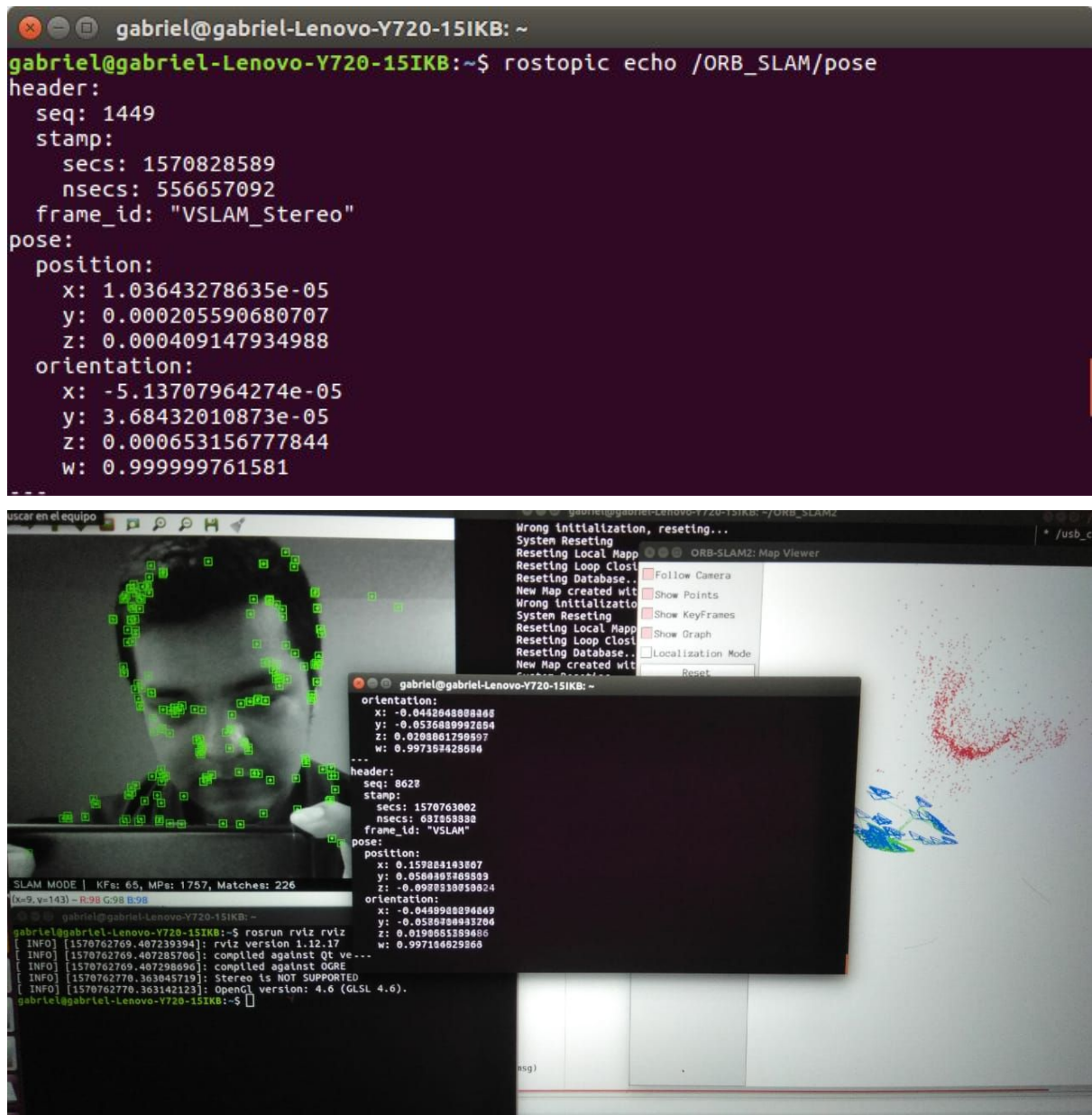
Camera Parameters:
- fx: 458.654
- fy: 457.296
- cx: 367.215
- cy: 248.375
- k1: -0.283408
- k2: 0.0739591
- p1: 0.00019359
- p2: 1.76187e-05
```

## POSE

Tanto en el modo GUARDAR MAPA y CARGAR MAPA, podemos utilizar la localización de la cámara dentro del SLAM. Por lo tanto, es posible ejecutar la POSE para identificar la ubicación de nuestra cámara dentro del mapa. Al correr `rostopic list`, encontraremos un tópico llamado `/ORB_SLAM/pose`. Entonces para obtener el valor de la pose se deberá de escribir en la terminal:

Terminal 4:

```
rostopic echo /ORB_SLAM/pose
```



## KEYFRAMES

## FUENTES BIBLIOGRÁFICAS

Para más información, puede consultar los siguientes fuentes que fueron consultadas para instalar correctamente estos programas.

[https://github.com/raulmur/ORB\\_SLAM2](https://github.com/raulmur/ORB_SLAM2)

[https://blog.csdn.net/sinat\\_38343378/article/details/79153834](https://blog.csdn.net/sinat_38343378/article/details/79153834)

<https://www.youtube.com/watch?v=2Pboq2LFoal&t=188s>

[http://www.daslhub.org/unlv/wiki/doku.php?id=opencv\\_install\\_ubuntu](http://www.daslhub.org/unlv/wiki/doku.php?id=opencv_install_ubuntu)

<https://blog.csdn.net/u010821666/article/details/79817033>

<https://www.cnblogs.com/williamc17/p/10263188.html>

[https://github.com/raulmur/ORB\\_SLAM2/issues/403](https://github.com/raulmur/ORB_SLAM2/issues/403)

<https://blog.csdn.net/betterethan/article/details/81748575>

[https://blog.csdn.net/learning\\_tortosie/article/details/81384717](https://blog.csdn.net/learning_tortosie/article/details/81384717)