An analysis of the System acceptance review(SAR) of Team ARES

Akhilesh Singh
IT, 2nd year

The given activity diagram of the rover has been analysed for all the entities independently, and their interaction has been mentioned wherever necessary.

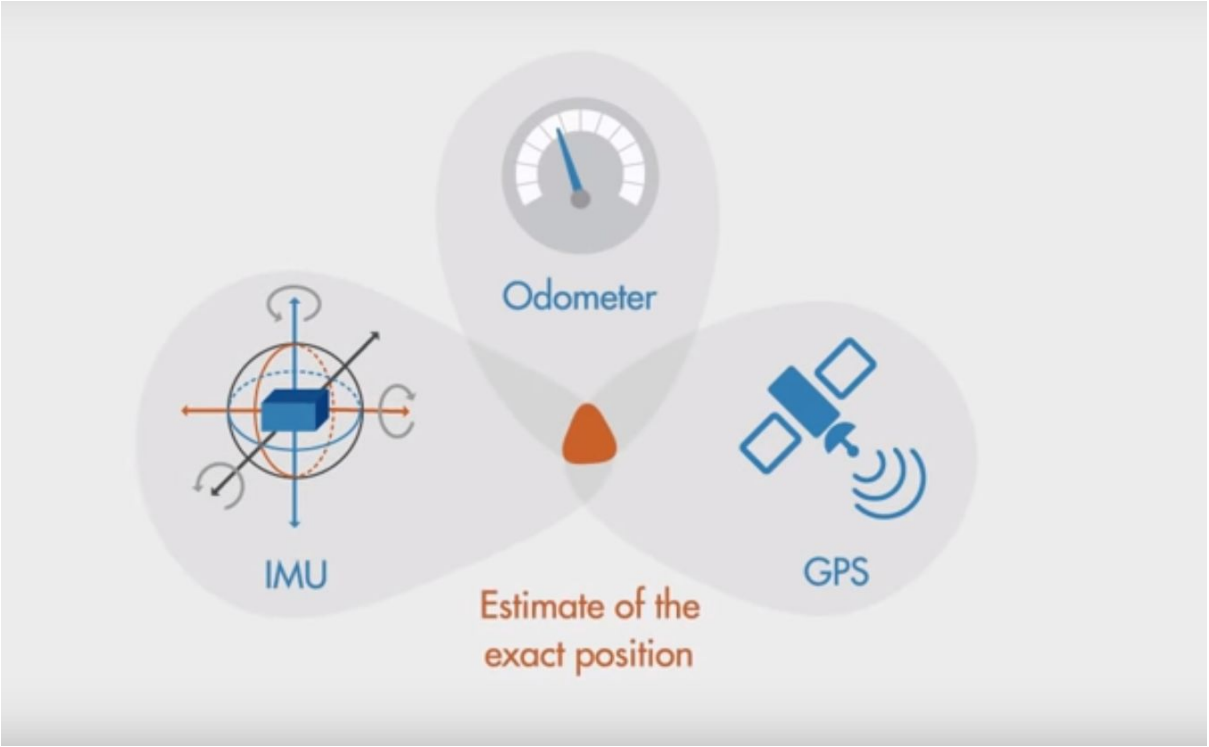
Sensor fusion for Drift Correction

(Red entity : Used for fusion of the stereo visual odometry, IMU and GPS for drift correction)

Odometry is the use of data from motion sensors to estimate change in position over time.

This unit contains a 9 Degrees of freedom Inertial measurement unit, which comprises of 3 sensors, an accelerometer, a gyroscope, and a magnetometer. The unit also has two extended Kalman filters.

Kalman Filter: Estimation from an indirect measurement. Predicts the parameter of interest in the presence of noisy environments. Combining measurements from various sensors in the presence of noise.



The Extended Kalman Filter is an extension of Kalman Filter which can be applied to non-linear systems.

The first filter performs estimation of the rover's roll, pitch and yaw values by combining the feed from the sensor and the roll, pitch and yaw values from the stereo setup. It then outputs the drift corrected values and sends them back to the localization thread.

The second filter performs estimation of the rover's velocity and heading direction by combining the feed from the GPS module(latitude and longitude) and the IMU sensors. This estimation is important because the update rate of packets from the GPS is comparatively lower. This estimation is then sent to the localization thread for global map representation.

Localization Thread

(Yellow Entity: Set of instructions for rover localization.)

Function 1: stereo_visual_odometry

Inputs the two image frames from the stereo camera, and extracts features from these images as flattened 2D matrices (2xN matrices) using the OpenCV's detect_features() function. This output is sent as an input to triangulatePoints() function.

Triangulation refers to the process of determining a point in 3D space given its projections onto two, or more, images, i.e., the triangulatepoints() function rebuilds the 3d image from the two image frames(along with the projection matrices) and outputs a 3xN matrix. This is what stereo vision is, combining two or more 3d images to construct an image.

The **solvePnpRANSAC()** function is further used for pose estimation of the desired object in the 3d space. The object coordinates and the corresponding image objects are given as input parameters. **Random Sample Consensus(RANSAC)** is used for pose estimation of noisy data or data which might have outliers.The parameters are estimated by randomly selecting the minimum number of points required. It outputs the corresponding rotation and translation vectors (R, t) for the pose of the object. These rotation and translation matrices belong to the lie group(SE3).

Function 2: local_map_optimization

This function receives the pose and the drift corrected roll, pitch, and yaw values of the rover. An odometry measurement between two consecutive poses depends only on the connected poses.This is represented as a graph.
Bundle adjustment for rover pose requires the minimization of a non-linear error function, which in our case, is being represented as a graph.
General graph optimization(g2o) framework of the OpenSLAM library is being used.

Sparse optimization and SE3 Quat give us the optimized local and 3D coordinates after minimization of the non-linear error function.

OpenGL library is then used to visualize the resulting coordinates by the Visual_point_cloud() function. These coordinates are also sent to the global_map_representation() function.

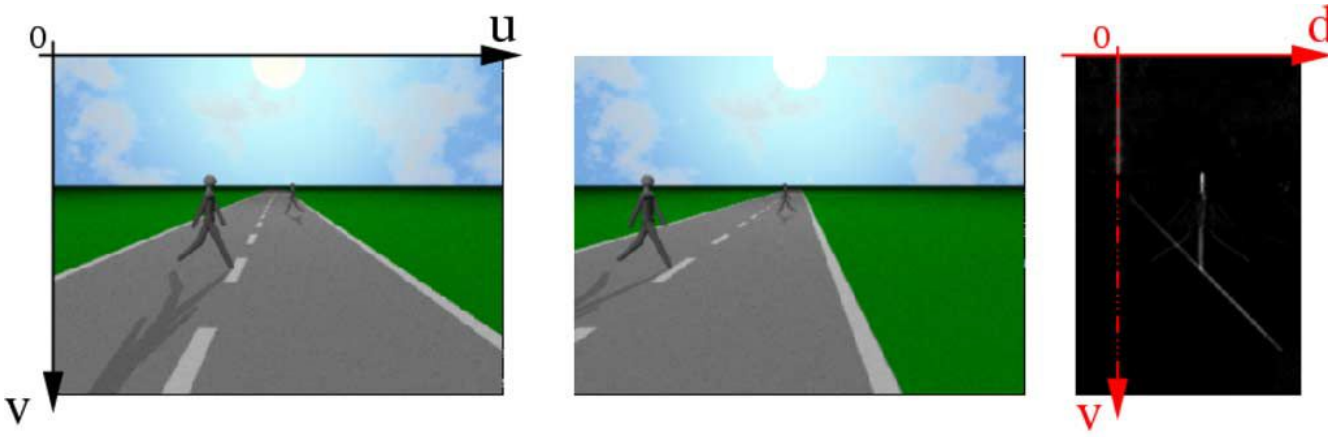
Function 3: global_map_representation()
This function uses the local coordinates and states(heading direction, velocity) to determine the global states. The drift corrected states are received from the sensor fusion unit. These states together represent the current position and state of the rover on the global map. These states and constraints will now be used by the motion planning thread to carry out motion planning.

Obstacle detection and motion planning thread

(Blue entity: Set of instructions for object for the rover to detect obstacles and perform autonomous traversal)

Function 1: Obstacle detection

Accounts for the disparity between the two image frames from the two stereo cameras. **Disparity** refers to the difference in image location of an object seen by the left and right eyes, resulting from the eyes' horizontal separation (parallax). **Disparity map** refers to the apparent pixel difference or motion between a pair of stereo images. SGBM (Semi block matching algorithm) is an algorithm to get disparity map from the multiple stereo images. OpenCV’s stereoSBGM() function is used for this task. The disparities of the points on the ground plane will appear as a strong line in the **V-disparity map**.



Left image Right image V-disparity

calc_v_disparity() maps the ground plane in a slanted line and then hough_transform is used for line detection. When this navigable region of V disparity is subtracted from the original disparity map, we get the obstacle in the image. This object disparity is projected to 3D coordinates and sent to the motion planner.

Function 2 : Motion Planner

The motion planner gets the states (coordinates, velocity and heading direction) of the rover from the localization thread and the obstacle position from the object detector of this thread. **Open Motion Planning Library(OMPL)** uses this information for motion planning, finding the shortest path to the destination, avoiding the obstacles, taking into account the rover

constraints. The **A* search algorithm** is being used to find the shortest path. Commands are generated accordingly for movement and are sent the ROS node.

ROS node for motion control

(Purple entity: Robot operating system)

It receives the computed commands from the Motion planning thread and drives the rover motors by sending the control signals accordingly.

Ball Detection

This is a task of object detection(a green ball specifically) which the rover is performing.

A **contour** is the boundary of an object in an image. OpenCV's Contour_Detection() is used to get the contours from the stereo images. It detects the circular contour of the marker. Then the color of the marker(green) is detected using Color_detection() function.

Proposed changes in the system:

1. The rover draws a disparity map every time for all the image frames to detect the obstacles. I believe that a Mars Rover will encounter some very specific and limited obstacles (rocks, boulders, workstation etc.). We can pre-train a Convolutional Neural network(CNN) with many images of these standard obstacles. The rover can then instantly detect these obstacles and this would save us a lot of computation.

2. I read somewhere that a single Extended Kalman Filter can fuse all the three inputs (Stereo visual odometry, IMU and GPS) and then perform the required drift corrections and estimations. If that is the case, a single extended Kalman Filter can be used in the sensor fusion and drift correction unit, instead of two.