# Team ARES

# Software Design Analysis
# ATOM-2018

**Jitesh Singla**
*COE, Batch of 2021*

## Introduction

The aim of this report is to understand the working of the software components designed for ATOM-2018.

The main features of ATOM have been inspired from the tasks that are to be performed during the on-site round of URC (University Rover Challenge).

Following is a brief overview of the Autonomous Traversal Task – the primary target for the software team.

### Autonomous Traversal Task

Rovers shall be required to autonomously traverse between markers in this staged task across moderately difficult terrain.

The task has four stages, each with increasing difficulty. The information at hand which can be leveraged, the problem to be solved and the techniques which can be employed are described below for each of the levels.

### Level 1

**Equipped with** - GPS coordinates, teleoperation allowed using antenna, markers very near to GPS coordinates

**Difficulty** – Tele operated scouting to be achieved

**Possible techniques to leverage**

- Using omni direction antenna to improve the control

### Level 2

**Equipped with** - GPS coordinates, teleoperation allowed using antenna

**Difficulty** - Marker need to be identified as these would not be located at exact GPS location provided

**Possible techniques to leverage**

- Computation algorithms using nearby GPS coordinates
- Using ML to enable accurate identification of tennis ball markers. ***Data augmentation*** can be employed to train the model using limited possible imagery training dataset.

### Level 3

**Equipped with** - GPS coordinates

**Difficulty** – Purely autonomous navigation, autonomous marker

**Possible tools & techniques to leverage**

- Simultaneous localisation and mapping **(SLAM)** technique to develop relatively accurate maps of unknown environment. Extended Kalman filter can be used to achieve the same.
- Continuous updating of SLAM model by feeding two streams of data viz. **terrain mapping data** from LIDAR and **localised position data** by GNSS (Global Navigation Satellite System)

## Level 4

**Equipped with** – Bearing and distance to subsequent markers

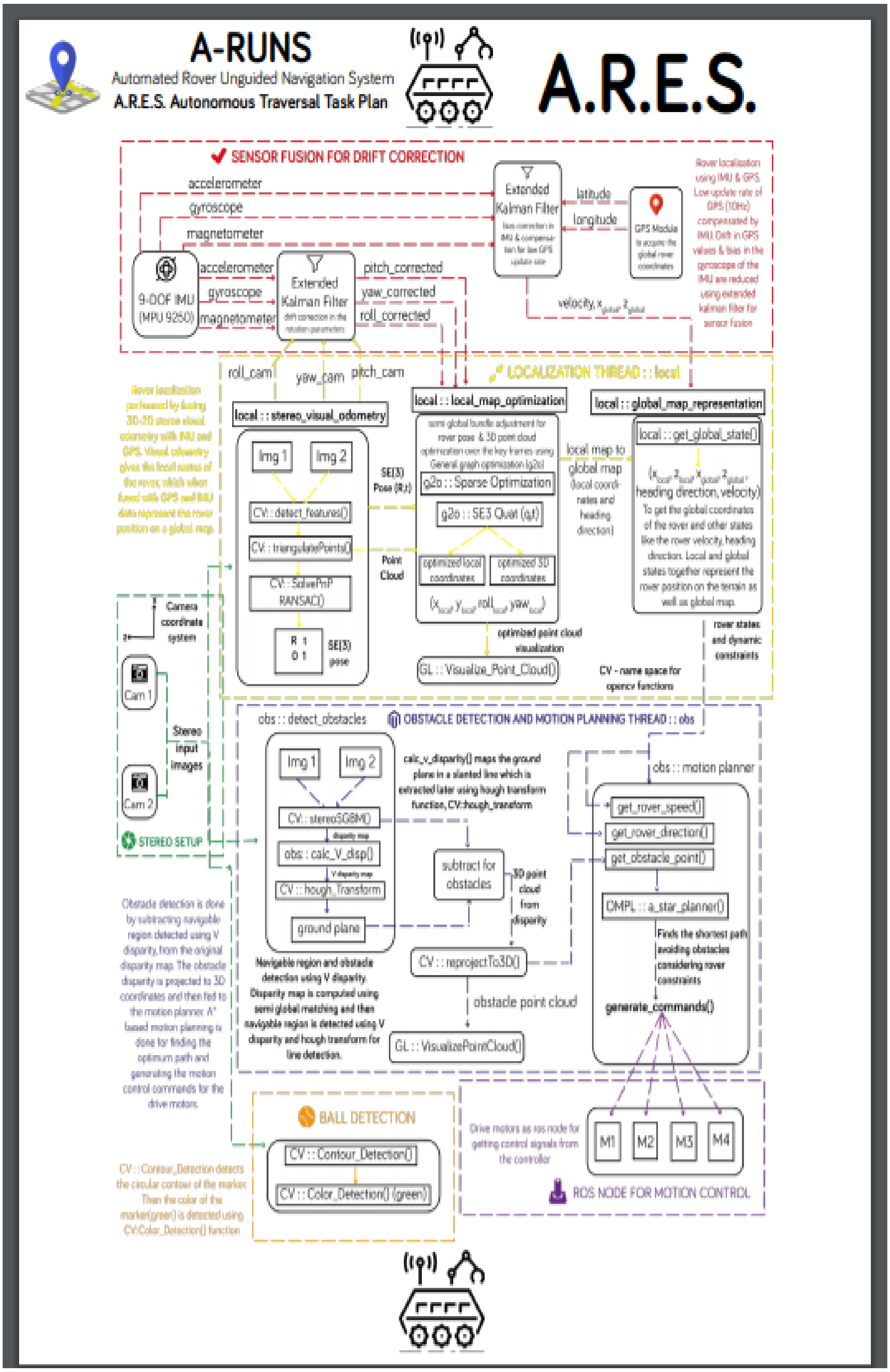
**Difficulty** – Purely autonomous navigation, autonomous marker identification, significant obstacles between the markers

**Possible techniques to leverage**

- *Obstacle detection:* Dynamic correction of rover's path by utilising **motion compensation**. This is achieved by shifting of subsequent frame and analysing the difference between adjacent frames to easily auto detect the obstacles
- *Obstacle detection:* For obstacle avoidance we use LIDAR sensor that is continually sweeping 180 degree cone in front of the rover with which we can detect any insurmountable obstacle
- *Autonomous navigation:* Using together data from GPS radar and multiple **IMUs (inertial measurement units)** for accurate odometry (measuring change in position) and obstacle avoidance

**Jitesh Singla**
*COE, Batch of 2021*

**SAR Design- ATOM 2018**


A-RUNS
Automated Rover Unguided Navigation System
A.R.E.S. Autonomous Traversal Task Plan
A.R.E.S.
SENSOR FUSION FOR DRIFT CORRECTION
accelerometer
gyroscope
magnetometer
Extended Kalman Filter
latitude
longitude
GPS Module
9-DOF IMU (MPU 9250)
pitch_corrected
yaw_corrected
roll_corrected
velocity, x_global, z_global
Rover localization using IMU & GPS. Low update rate of GPS (10Hz) compensated by IMU. Drift in GPS values & bias in the gyroscope of the IMU are reduced using extended kalman filter for sensor fusion
roll_cam
yaw_cam
pitch_cam
LOCALIZATION THREAD :: local
local :: stereo_visual_odometry
local :: local_map_optimization
local :: global_map_representation
Img 1
Img 2
CV::detect_features()
CV::triangulatePoints()
CV::SolvePnP RANSAC()
SE(3) Pose (R,t)
Point Cloud
SE(3) pose
g2o :: Sparse Optimization
g2o :: SE3 Quat (q,t)
optimized local coordinates
optimized 3D coordinates
(x_local, y_local, roll_local, yaw_local)
local map to global map
local :: get_global_state()
heading direction, velocity)
optimized point cloud visualization
GL :: Visualize_Point_Cloud()
rover states and dynamic constraints
CV - name space for opencv functions
Rover localization performed by fusing 3D-2D stereo visual odometry with IMU and GPS. Visual odometry gives the local states of the rover, which when fused with GPS and IMU data represent the rover position on a global map.
Camera coordinate system
Cam 1
Cam 2
Stereo input images
STEREO SETUP
obs :: detect_obstacles
OBSTACLE DETECTION AND MOTION PLANNING THREAD :: obs
CV :: stereoSGBM()
obs :: calc_V_disp()
CV :: hough_Transform
ground plane
subtract for obstacles
3D point cloud from disparity
CV :: reprojectTo3D()
obstacle point cloud
GL :: VisualizePointCloud()
obs :: motion planner
get_rover_speed()
get_rover_direction()
get_obstacle_point()
OMPL :: a_star_planner()
Finds the shortest path avoiding obstacles considering rover constraints
generate_commands()
Navigable region and obstacle detection using V disparity. Disparity map is computed using semi global matching and then navigable region is detected using V disparity and hough transform for line detection.
Obstacle detection is done by subtracting navigable region detected using V disparity, from the original disparity map. The obstacle disparity is projected to 3D coordinates and then fed to the motion planner. A* based motion planning is done for finding the optimum path and generating the motion control commands for the drive motors.
BALL DETECTION
CV :: Contour_Detection()
CV :: Color_Detection() (green)
CV::Contour_Detection detects the circular contour of the marker. Then the color of the marker(green) is detected using CV:Color_Detection() function
Drive motors as ros node for getting control signals from the controller
M1
M2
M3
M4
ROS NODE FOR MOTION CONTROL

**Jitesh Singla**

*COE, Batch of 2021*

To provide the rover with the required capabilities, we have created paralleled threads that will perform specific functions during real time operation.

Requirements

- **GPU**
  The rover needs to traverse autonomously for a major portion of the tasks involved. Hence the autonomous subsystem should not only be accurate but also fast enough to avoid any lag. Our design involves three threads running in parallel – Localization, Mapping and Path Planning. Since the autonomous navigation system is based on dense depth map generated from the stereo setup, it demands good computational power for real time implementation. Also, the other threads need to work accurately and in sync with each other for real time decision making and smooth functioning. Hence, to meet the computational requirements, a graphics processing unit is required for onboard processing. Nvidia Jetson Nano  has been used which not only suffices our computational requirements but also cost efficient.
- **High resolution USB cameras**
  Our approach to provide the autonomous traversal feature employs data feed in the form of stereo images to map the external environment and manoeuvre the rover accordingly. A pipeline has been constructed which takes input these stereo images and provides the desired output in the form of path coordinates leveraging state of the art computer vision techniques at the middle stages. The efficiency of the entire pipeline depends on the quality of data feed given as input. Hence to achieve a high-level accuracy, 2 High resolution USB cameras have been used which provide real time feed of the external environment.
- **IMU Sensors**
  One of the main functions of the autonomous subsystem is the localization of the rover. It assists in creating a virtual map of the environment and providing real time data of the rover's location for path planning. To achieve the required precision and accuracy, IMU sensors have been used which provide a real time stream of data that helps in locating the rover.

**Jitesh Singla**
*COE, Batch of 2021*

The three main components of the pipeline are:

- Localization
- Obstacle Detection
- Path Planning and controls

We will breakdown the working into three components for providing a better understanding.

## Localization

It involves employing the information from the various sensors attached to the rover to calculate the various physical parameters of the rover's motion.

Input to this thread

- Current position
- Coordinates of the obstacle
- Destination coordinates

All the three inputs are dynamic and are subject to changes as the rover traverses.

The Localization thread provides update in the form of current location, speed and acceleration.

The Obstacle Detection thread provides the coordinates of the obstacles as they are detected

The destination coordinates depend on the type of task at hand.

Algorithms Used

Visual odometry is employed to create a 2D map of the path traversed by the rover. It helps in determining the position and orientation of a robot by analysing the associated stereo images captured as the robot traverses. This is done via using inbuilt functions in OpenCV by detecting common features across stereo images and triangulating the points.

An outline of the visual odometry thread:

1. Capture images: Itl, Itr, It+1l, It+1r

2. Undistort, Rectify the above images.

3. Compute the disparity map Dt from Itl, Itr and the map Dt+1 from It+1l, It+1r.

4. Use most suitable feature detection algorithm to detect features in Itl, It+1l and match them.

5. Use the disparity maps Dt, Dt+1 to calculate the 3D positions of the features detected in the previous steps. Two point-clouds Wt, Wt+1 will be obtained

6. Select a subset of points from the above point cloud such that all the matches are mutually compatible.

7. Estimate R,t from the inliers that were detected in the previous step.

Visual Odometry universally suffers from precision problems. The error is compounded when the vehicle operates on non-smooth surfaces. Odometry readings become increasingly unreliable as these errors accumulate and compound over time.

Hence to reduce the error, we are using the pose data obtained earlier with the data from 9-DOF IMU sensor using Extended Kalman Filter the error in our multi-dimensional data.

We have used two EKF, one for extracting the motion parameters i.e. pitch, yaw and roll and the other for extracting location parameters.

**Kalman Filter 1:**

This calculates the drift correction in the rotation parameters for our rover. It's inputs are:

1. Accelerometer 2. Gyroscope 3. Magnetometer

Its outputs are:

1. Pitch correction 2. Yaw correction 3. Roll correction

These outputs are fed into the local_map_optimization module of the Localisation Thread.

**Kalman Filter 2:**

This filter calculates the bias correction in the IMU & compensation for a Low GPS update rate. Its inputs are:

1. Accelerometer 2. Gyroscope 3. Magnetometer 4. Latitude 5. Longitude

Latitude and Longitude are obtained from the GPS Module (acquires Global Rover coordinates)

Its outputs are:

1. Velocity 2. Xglobal, Zglobal

Which are fed into the global_map_representation module of Localization Thread.

OpenGL is used to create a 3-D point cloud map of the environment to improve the accuracy of global data using loop closure.

In this way, localization thread aims at providing reliable data to the system. The accuracy achieved in this thread directly impacts the efficiency of the path planning thread.

**Jitesh Singla**
*COE, Batch of 2021*

## Obstacle Detection

The process for obstacle detection aims at calculating the depth of the obstacle from the stereo images It involves creating a V-disparity map of the scene currently captured by the camera and then subtracting the ground plane via mapping with Hough Transform.

Input to this thread

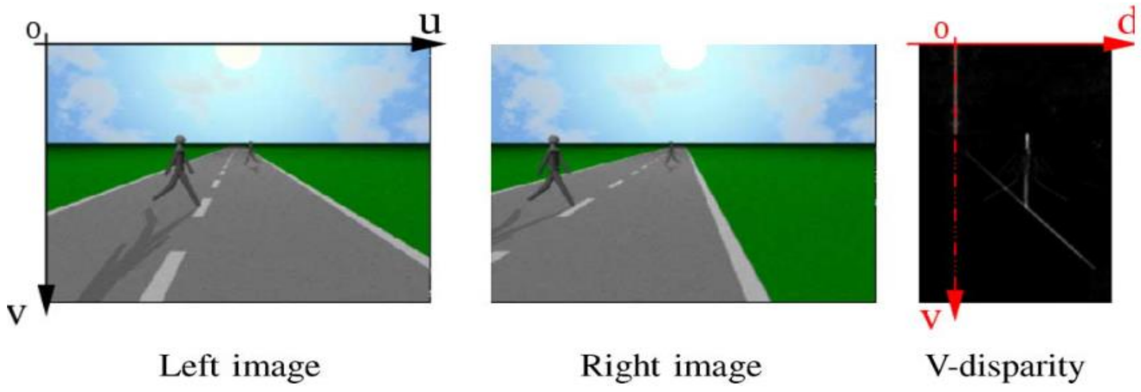
- Stereo images

Algorithms Used

The outline for obstacle detection is:

1. A disparity map is created from the stereo images obtained from the camera feed. For this, cv::stereoSGBM is used.

2. Based on the disparity map, a V disparity map is created. A V-disparity map is basically a histogram of the various values of disparity that appeared on that row in the disparity map. It is used to detect the ground plane as the disparities of the points on the ground plane appear as strong line in it.



3. Hough Transform is used to detect the ground plane from the V-disparity map.

4. The ground plane is subtracted from the disparity map computed earlier to localize the newly detected obstacles in the 3-D point cloud.

In this way, we are able to detect both the navigable region as well the obstacles which is then used by the path planning thread to compute the motion of the rover.

Future Prospect

V-disparity map only provides us with the depth of the obstacle i.e. the relative distance b/w the rover and the obstacle. By using U-disparity map to create a fake disparity map, we can have an estimate of the exact orientation of the obstacle.

**Jitesh Singla**
*COE, Batch of 2021*

## Path Planning

It involves employing the information about the environment to build up a least cost traversable path.

Input to this thread

- Current position
- Coordinates of the obstacle
- Destination coordinates

All the three inputs are dynamic and are subject to changes as the rover traverses.

The Localization thread provides update in the form of current location, speed and acceleration.

The Obstacle Detection thread provides the coordinates of the obstacles as they are detected

The destination coordinates depend on the type of task at hand.

Algorithms Used

The process for path planning started with using a vector-based path planning algorithm. However due to its computational inefficiency, it wasn't possible to implement it for a real time system. In order to deal with this, we modified the A-star algorithm which is the current state of the art algorithm for path planning. Also, we worked on to include a feedback system with autocorrect feature to account for sudden changes in the rover's location owing to external factors.

Future Prospect

One of the main reasons behind going for A-star algorithm is its computational efficiency. To build up on that we have encountered two more algorithms that might provide better results:

1. HCTnav

2. All direction A-star

However, these haven't been tested yet. Hence it is yet to be verified if they actually surpass the performance achieved via A-star algorithm.

**Jitesh Singla**
*COE, Batch of 2021*