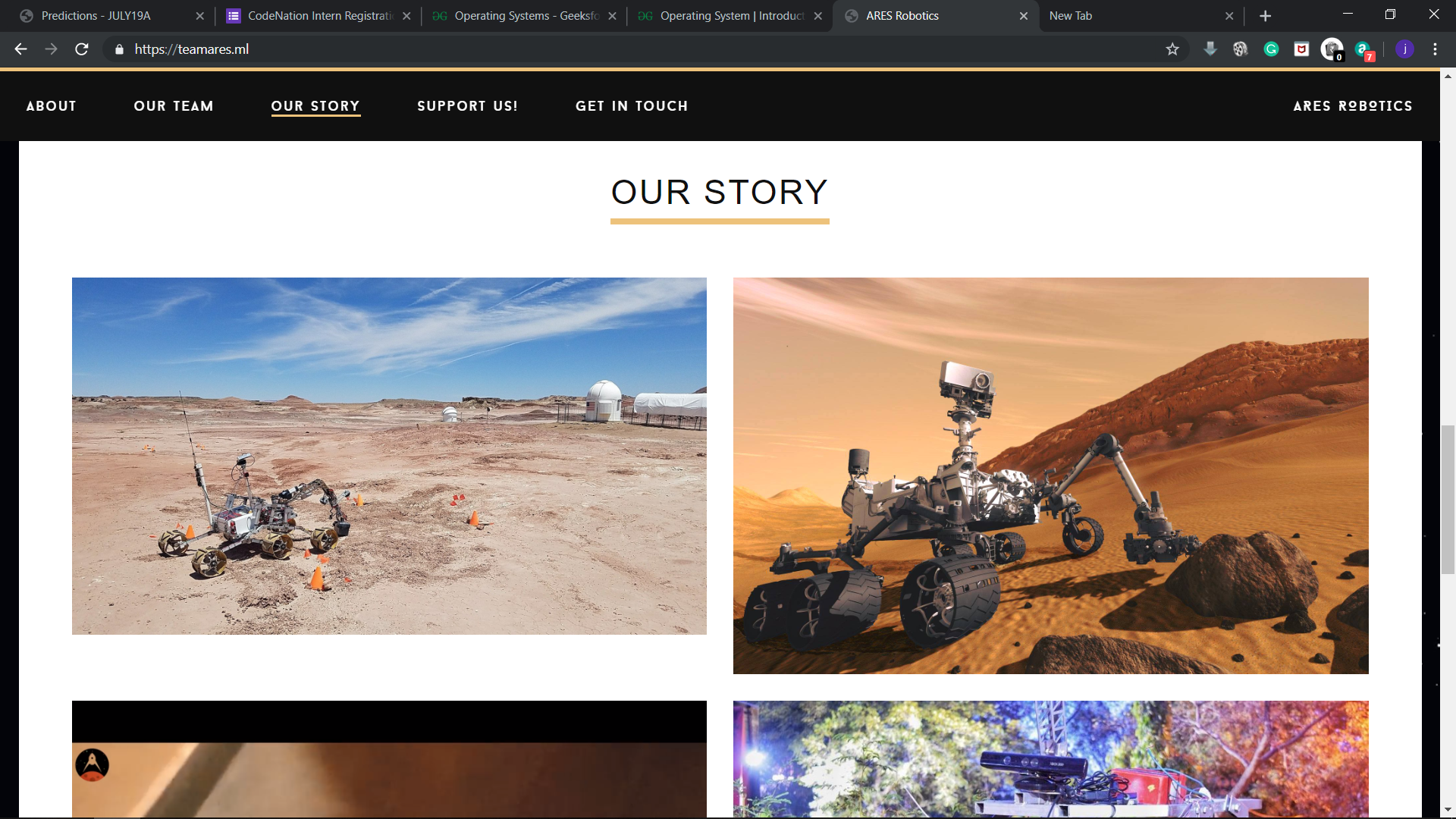
**Team ARES**

**Software Design Analysis**



**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)

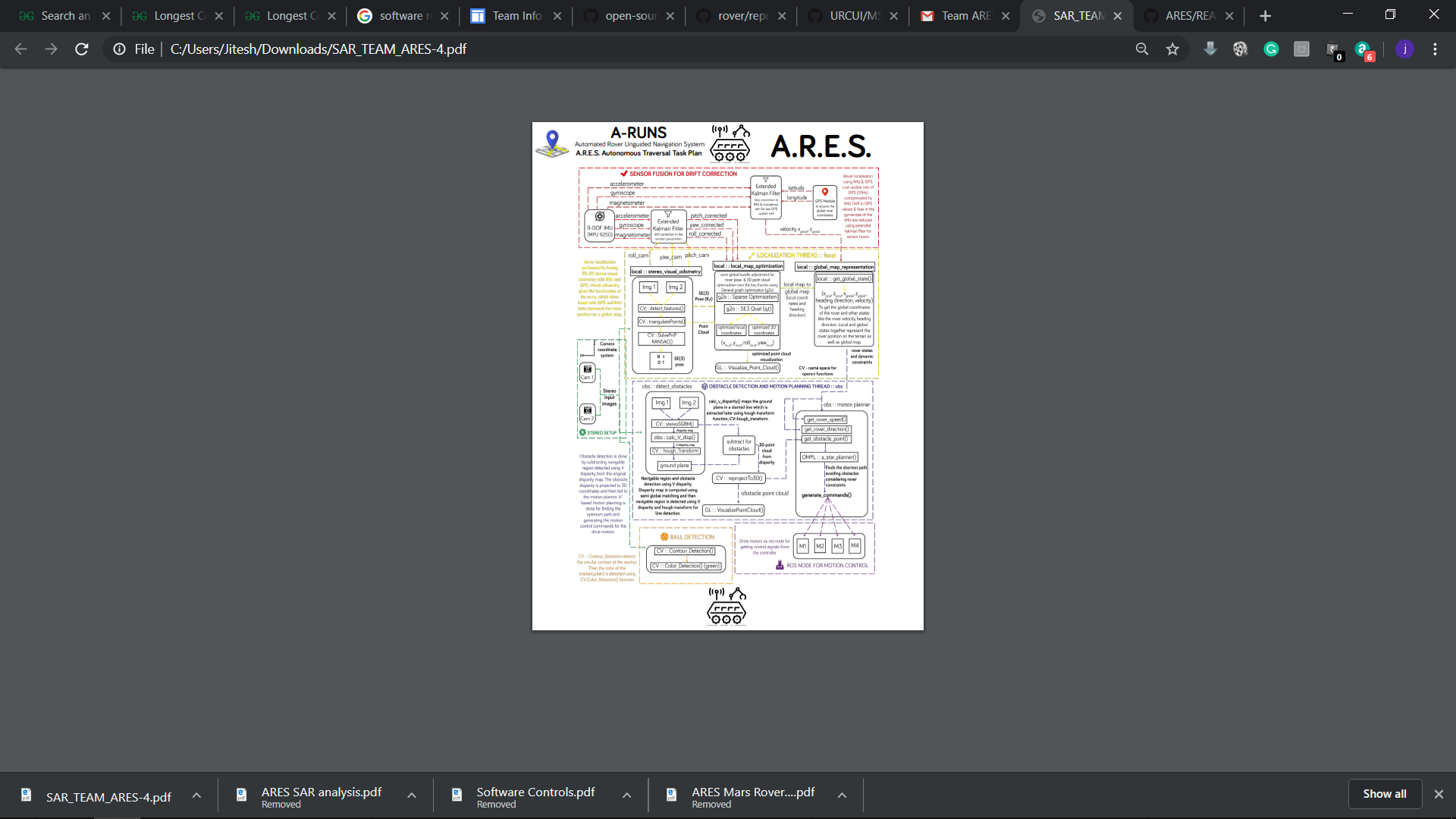
**Level 4**

**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



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.

The three main components of the pipeline are:

* Path Planning
* Localization
* Obstacle Detection

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

**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.

**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

We are using 9-DOF IMU sensor for gathering data while rover is in motion. Since the there is a high chance of error in the data, it isn’t possible to use it as such for localization. Hence Extended Kalman Filter is used to reduce the error in such multi-dimensional data.

In our process, two EKF’s are used, one for extracting the motion parameters i.e. pitch, yaw and roll and the other for extracting location parameters.

Also, visual odometry is employed to create a 2D map of the path traversed by the rover. This is done via using inbuilt functions in OpenCV by detecting common features across stereo images and triangulating the points.

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.

Future Prospect

NA

**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

OpenCV provides inbuilt functions to calculate the V-disparity map and Hough Transform through which we can localize the obstacle relative to the camera coordinates. Using the information from localization thread, these can be easily converted to World coordinates and be updated in the virtual map.

Later using the transformation matrix, we update the 3D point cloud of the environment.

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.