A- RUNS

Automated Rover Unguided Navigation System

The A-RUNS or the ARES Autonomous traversal task plan is the system for an unguided self driving vehicle. The whole system is divided into 6 parts -

1. Drift Correction System
2. Rover Localisation System
3. Obstacle Detection and Motion Planning System
4. Stereo set up
5. Ball Detection System
6. ROS Node for Motion Control

Each system has its on tasks and combined they work in making a completely self driving rover. Let us start discussing the different subsystems -

1. Drift Correction System

   It is the system which used to get the correct position of the Rover.
   When the rover is steady it can get GPS signals from multiple directions. These signals are reflections of the signals from different objects. The Signals are from main satellite only
   It uses the GPS and IMU technologies to find the position and an add on Kalman Filter to improve the estimation

   1. IMU - stands for Inertial Measurement Unit, it is used to measure acceleration, and angular velocity. Sometimes magnetic field as well
   2. Gyroscope - it is used to measure angular velocity
   3. Accelerometer- electromechanical device used to measure accleration
   4. Magnetometer - Device used to measure magnetism, direction & strength of it
   5. GPS - it stands for Global Positioning System, as the name goes it is used to find the position of the position of the system using satellite positioning

      These sensors are used to find the position of the system but we know the problem of drift, to we use the Kalman Filter which is an Optimal estimation Algorithm to find the position of the system to the maximum accuracy possible.

Working of Kalman filter

Kalman filter is an algorithm which allows us to make a guess about a quantity if we know certain things about it.
In our case as the rover is on mars and there the rover can't completely rely upon the GPS signals for positioning and for certain terrains more accuracy is required than the number achieved by GPS so the inputs from the IMU are also used, to track the position better.
Kalman filter assumes that all variables are random and gaussian distributed. Each variable has mean and variance.
Then the algorithm tries to find the correlation between the variables
In the way that if it finds the correlation between position and velocity then we can predict the new position on the basis of the current velocity obtained from the IMU and the position obtained from the GPS.

There are two Kalman filters so one gives the pitch, yaw and roll to be the best estimation
And other gives the best velocity estimation.
These outputs are fed into the Rover Localisation system

2. Stereo set up

It consists of two cameras which takes images which are used by obstacle detection system, ball detection system and the Rover Localisation system .

3. Ball Detection

The Ball detection system has to parts, the first in the contour detection whose output is fed into the color detection system.
Contour Detection- Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition.
it is traced out using the openCV library and to the best of my knowledge it is done using the findContours() function. The function returns a python list of all the contours in the image.
Colour detection system- The color detection system over here is used detect green color. The openCV library is used to find the color, the following are the summarized steps on how to do it. Take each frame, convert it from BGR to HSV color space, threshold the HSV color space for a range of green.

4. Rover Localisation System

The Rover Localisation System uses the stereo visual odometry with IMU and GPS to together to represent the position of the rover on the Global Map.
The system is divided in 3 parts -

1. Stereo Visual Odometry
2. Local Map optimisation
3. Global Map Representation

Stereo Visual Odometry - the images are taken from the stereo set up, here the openCV library is used. The first step is feature detection , features are the patterns found in an image which repeat themselves. Then the extracted features are used in the triangulation. Triangulation is the process of finding the position of a point in 3D space. So the we used OpenCV to find the 3D points in an array. Then the 3D reconstruction is done using the solvePnp
The output of the triangulation and the feature detection are passed to the local map optimization process

Local Map Optimisation - semi global bundle adjustments for rover pose and 3D point cloud optimisation using the General graph optimisation. Bundle Adjustments can be phrased as least square optimization of an error function that can be represented as a graph.
G2O is the optimisation algo used here, it is an open source framework for optimising non linear error functions.
The reason its usage according to me is that it is easy to use by extended a few lines of code and cane be used to solve even new cases of the problem.
From this we get optimised local coordinates and optimised 3D coordinates. These coordinates are then fed into the global map representation

Global Map Representation - The system calibrates all the state variables of the system and the absolute global position of the system . The local and global states represent the rover position on the terrain as well as global map.
These states can be accessed by other systems of the rover when they need it

5. Obstacle Detection and Motion Planning Thread

This is the most important system of the Rover it decides the path and commands the movement of the rover on the terrain, various steps are taken in this block to create the shortest obstacle free path.

First a disparity map is created, a disparity map is a measure of how different two images are, how far away similar are the egdes/corners/feature points are from one image to the other.
These are used in stereoscopic vision algorithms in calculation of depth.
In this the algorithm being used is the stereoSGBM, which is the OpenCV function for the semi global block matching algorithm . Semi Global Matching performs a pixel wise matching allowing you to shape object boundaries and fine details unlike single global matching algorithms it is not prone to streaking effect because of symmetrically compute the pixel matching cost through several paths in the image . FInally the algorithm chooses the pixel matching solution with the minimum cost using dynamic programming approach.
The disparity map generated is fed into the subtract for obstacles block and into the calc_v_display() function which as stated maps the grounded plane in the slant line which is extracted later using the hough_transform()
The Hough Transform is a feature extraction technique, The purpose of the technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure. This voting procedure is carried out in a parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform.
The Hough Transform does the line detection
Then the subtract for obstacles block detects the object by subtracting the disparity map and the output of hough transform giving the 3D point cloud from disparity

The reprojectTo3D() function creates a 3D obstacle point cloud which is diplayed using the VisualizePointCloud() function of the OpenGL library

The 3D obstacle point cloud is also fed to the motion planner system which also gets the velocity and rover locomotion direction from the rover localisation system, i.e. from the Global Map Representation, the a_star_planner() finds the shortest path avoiding obstacles and considering all constraints and commands are generated for ROS node for motion Control

6. ROS Node for Motion Control

ROS stands for Robot Operating System, it is the node of the operating system used for the controlling the movement of the robot after getting the motion plan from the obstacle detection and motion planning unit.

Suggestions for improvement

1. The First Change that I would like to suggest would be that we can use a better algorithm for creation of the disparity map, an improved version of Semi Global Mapping has been developed which is known as MGM (More Global Matching)
A change has been made in the recursive update formula of SGM due to which the influence of more nodes traversed earlier can be be incorporated in the currently processing node thus reducing the number of streaks in the disparity map even further. Link to the research paper and the github repository is quoted in the bibliography

2. Deep learning based contour detection - in the ball detection system we one of our systems does contour detection using the default function available in OpenCV, Libraries like OpenCV use very old functions like the Canny edge detection algorithm for contour detection, which fails to indentify contours in certain scenarios, instead of using it we can use a deep learning based approach to solve our problem. This provides more accurate results. The article for execution of this is linked in the bibliography

3. Alternate to A* planner, as A* uses good amount of compute power and consumes high energy and alternative algorithm for path planning in autonomous robots is HCTnav, it is designed for robots with low computation resources and limited energy supply, therefore bringing down the cost of the rover.

Bibliography

1. g2o - https://www.cct.lsu.edu/~kzhang/papers/g2o.pdf
2. Kalman filer - https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/
3. Comparision between global and semi global matching algorithm - https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XL-5/187/2014/isprsarchives-XL-5-187-2014.pdf
4. MGM research paper -http://dev.ipol.im/~facciolo/mgm/mgm.pdf
5. Github Repository for MGM - https://github.com/gfacciol/mgm

6. Deep Learning based contour detection - https://cv-tricks.com/opencv-dnn/edge-detection-hed/
7. HCTNav research paper - https://www.mdpi.com/2220-9964/2/3/729/pdf
8.