e-Yantra Robotics Competition Plus (eYRC+)

Implementation Analysis – Courier Service

The project focuses on developing an automated courier service using a programmed robot to efficiently pick up and deliver virtual packages. The robot operates on a pre-loaded map represented as a black grid with blue nodes, and it can only travel between nodes connected on this map. Various shapes and colors indicate different packages, which the robot must collect from specific nodes and deliver to designated destination nodes. When encountering a red node (traffic light), the robot can either wait for 30 seconds to cross or reroute to an alternative path.

Equipped with a Raspberry Pi B+ controller, motors, a USB camera, and various sensors, the robot navigates and performs tasks autonomously. It determines the shortest paths using algorithms and adjusts its routes in real-time based on traffic light detection. The robot identifies packages using image processing techniques, recognizing shapes and colors to ensure accurate delivery. The system's navigation and package handling capabilities are driven by a combination of pre-coded paths and dynamic adjustments in response to environmental inputs.

The project demonstrates how such automated systems can enhance efficiency in courier services and warehouse automation.

CONTROLLER

1. **Rasberry Pi B+**

   It is the brain of the robot. Rasberry Pi can be considered as a mini motherboard. It has its own operating system (we have used Rasbian_e-Yantra).

   Rasberry Pi will be used for making connections with the computer via Wi-Fi to show the packages picked up and delivered. It will also be used to control the motors to control the motion of robot. The main program for path traversal will be uploaded to the Rpi which will be used for following the shortest path. Rpi will also communicate with the USB camera to find out the status of traffic light and will plan the path accordingly.

2. **Micro SD Card(8 GB)**

   Micro SD Card is used to hold the Raspbian image. We have written Rasbian_e-Yantra on the SD Card using Win32DiskImager.

3. **L293D Motor Driver IC**

   L293D Motor Driver will be used to control the motors which will in turn control the motion of robot. Since directly connecting the motors to the Raspberry Pi will damage it, we are using the motor driver and a separate power connection to operate the motors. The direction of rotation and speed of motors will be controlled via the L239D Motor Driver IC.

COMMUNICATING Components

4. **Edimax Wifi Dongle**

   The Wifi dongle will be used to establish connection with the computer via Wifi signal. All the transfer of data like Python programs, test images, etc will take place through the Wifi Dongle to the Rpi.

SENSORS

5. **Intex USB camera**

   USB camera will be used to gather information about the environment and will be used as a sensor. The Python program will detect the status of traffic light using the frames provided by the USB Camera.

ACTUATORS

6. **BO MOTORS**

   BO motors are DC Battery operated motors which will be used as the main driver mechanism of the robot. It will rotate the wheels of the robot which will move the robot accordingly.

POWER SUPPLY

7. **Zook power bank 5000 mAH**

   Zook power bank will be used to supply power to the Raspberry Pi B+. It will act like a engine of the robot and thus no power cable will be needed for socket connection.

OTHER HARDWARE

8. **Wheels**

   Two wheels are attached to the two BO motors for the motion of the robot.

9. **Chassis**
   It is the body of the robot and will be used to hold the camera, power bank, Raspberry pi, BO motors, Wheels together. This is the skeleton of the robot.

10. **Castor wheels , nuts and bolts**
    Castor wheels are for Omni-directional movement. Nuts and bolts will be used to hold the components on the chassis.

11. **Connecting wires**
    Connecting wires are used to connect the L239D motor driver with the Raspberry Pi's GPIO pins.

12. **Red color paper sheet**
    It will be used for marking the active status of the traffic light on the arena according to the instructions during the course of competition.

How Camera is used

Purpose:

The camera will be used to detect only the status of traffic light. Since the path is already calculated and the links are of the same length, so the motion of the robot can be pre-coded.

How:

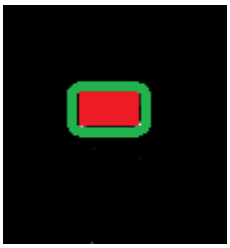
When the next junction is a Traffic signal, then the camera input will be taken to detect its status as shown:

1. Take an image from the camera of the environment from the adjacent junction of the Traffic signal.

2. Convert its format from RGB to HSV.

3. Mask the image to capture only the red color. The HSV range for detecting red color will be used.

4. Find contours on the resultant image.



Green contour on masked image

5. If the number of contours is one, it means that the status of traffic signal is active. Otherwise, it is inactive.

ALTERNATIVE-1

Limitation of above method:

1. The robot might get slightly out of the way when making a turn. Due to pre-coding no correction will be made.

2. If at any point speed of the motor changes, the whole path traversed will become incorrect.

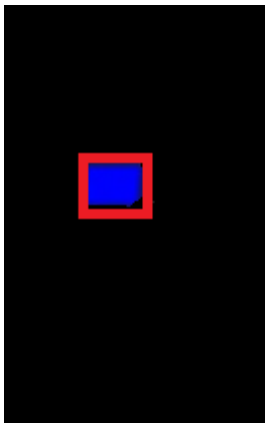
Alternate method:

(The camera setup will have to be changed to take picture of the currently traversing link)

1. From the starting junction, the robot will start capturing images using camera.

Image captured

2. The image captured will be checked for blue link by masking blue color and finding out contours on the masked image as discussed above.



Blue node detected

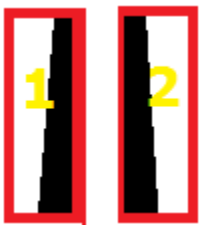
3. If the blue junction is detected, that means we have traversed a link and from that point, we can decide the direction of the next junction to be traversed.
4. If the next junction to be traversed is a traffic signal, then we will mask the image for both red and blue colors to detect its status. Based on the color detected an alternate shortest path may be decided.

ALTERNATIVE-2

For making the line following more reliable, we can use the camera in the following ways:

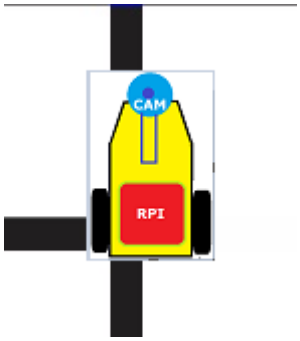
1. Each image captured will be cropped in such a way so as to capture only the path (the black link).

2. The cropped image will be divided into two equal regions.



3. Contours will be found on the white part of both the regions.
4. If the areas of contour found in both the parts are approximately equal, then no corrections need to be made on the motion.
5. If there is large difference between them, then it means we maybe near the edge of the black link. So, the robot will try to be in the middle by slightly turning and aligning towards the part of lesser area.

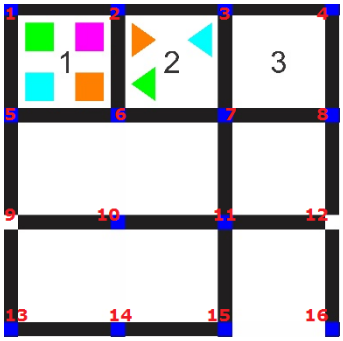
(Robot as shown is near the right edge as the white portion in larger in the right part)



Navigation Scheme

Steps used for line following:

1. Store the grid in the test image as Adjacency list (grid is also a graph)



Example Adjacency list = {1:[2,5], 2:[3,6], 3:[4,7], 4:[3,8], 5:[9,6], 6:[2], 7:[3,11], 8:[4,12], 9:[5,13], 10:[9,11], 11:[10,12], 12: [8,16], 13:[9, 14], 14:[13, 15], 15:[11, 14], 16:[12]}

2. Find the shortest path between starting and end nodes using shortest path algorithms.
3. Converts the series of path into left, straight and right motions.
4. Run the motor for specific interval of time to traverse each link in the path.

Package Identification

Steps to identify packages:

1. The packages are near the junctions. So we will crop the area near junction.

Cropped image of the package

2. Find contours around the package and find out its area.

Red contours. Area in range[600,700]

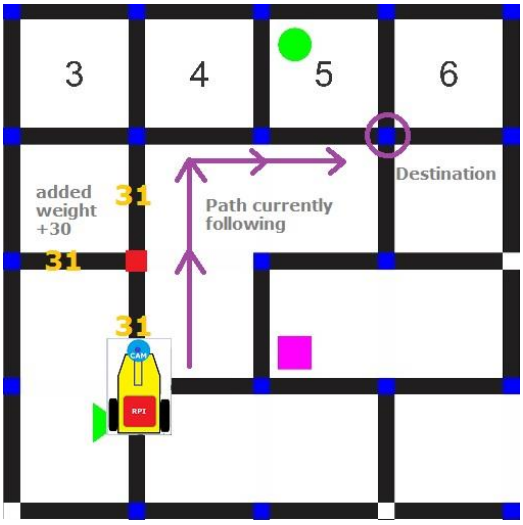
3. From the area we can find out the shape of the package.
4. To detect the color of the package, apply mask the clip for each color and find contours on the masked image.
5. If contours are formed, that means that the color of the package is the color used for masking the image.

green masked and countered image

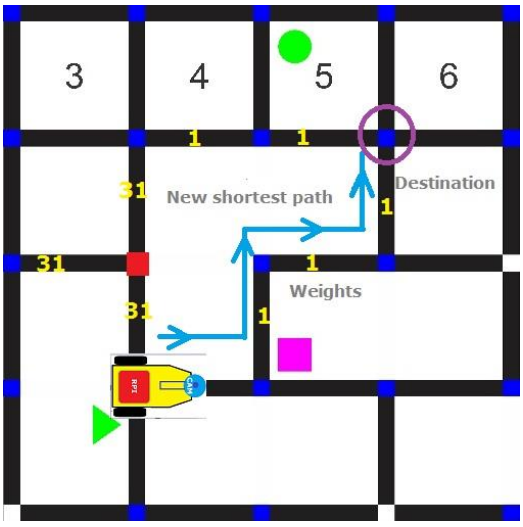
Action on detecting traffic

The camera is set in such a way that the robot can find out the status of the Traffic Signal from one of its adjacent nodes. If the signal is inactive then it will simply store the status of the signal and continue the path, but if it is active, then:

1. It will store the status of the signal and add a weight 30 to all those links directly connected to it.



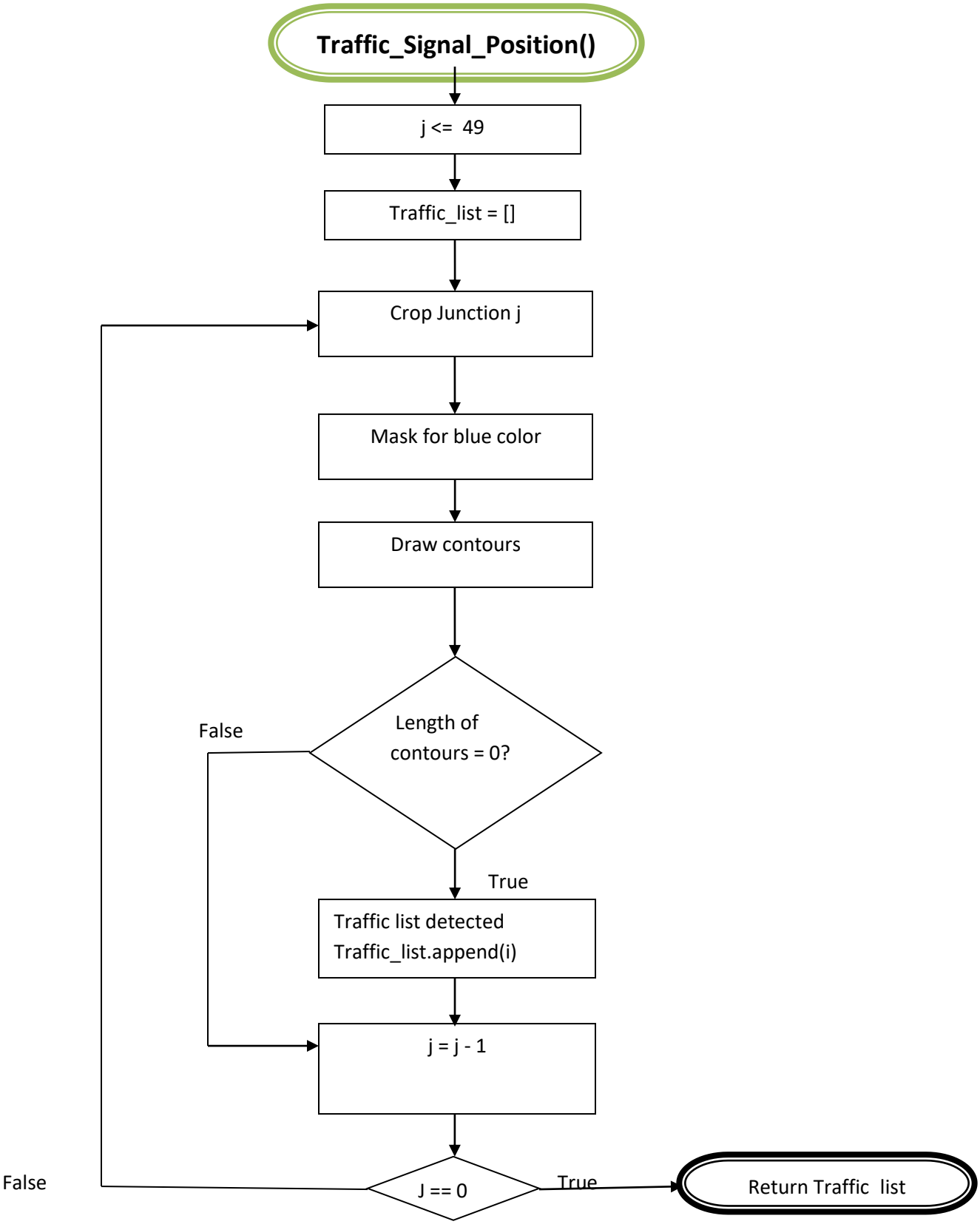
2. From the list of links and their respective weights, it will calculate an alternate path using **Dijkstra's shortest path algorithm**.



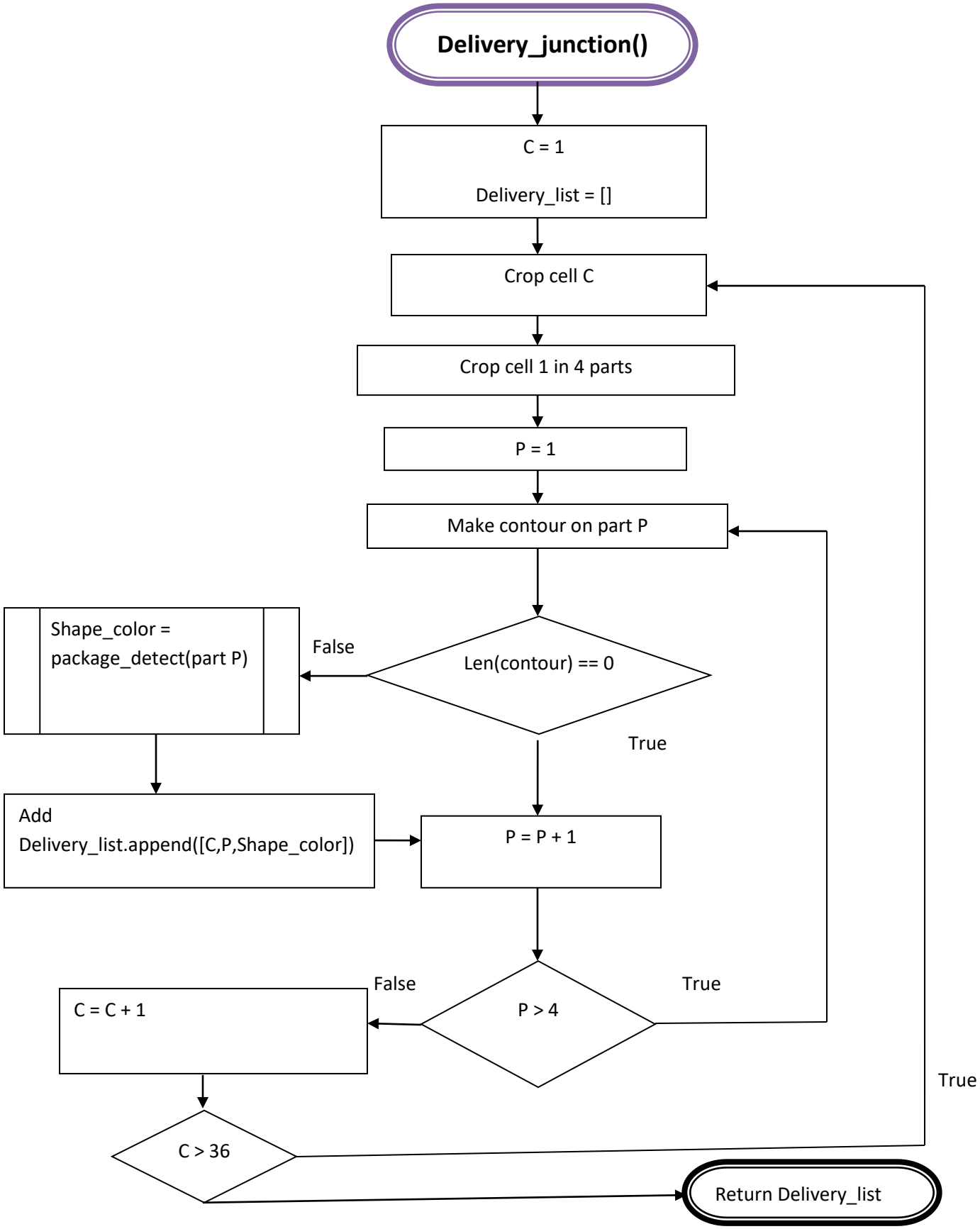
3. The newly generated path will be traversed, and if again an active traffic light is encountered, then these steps will be repeated.

Algorithm Analysis

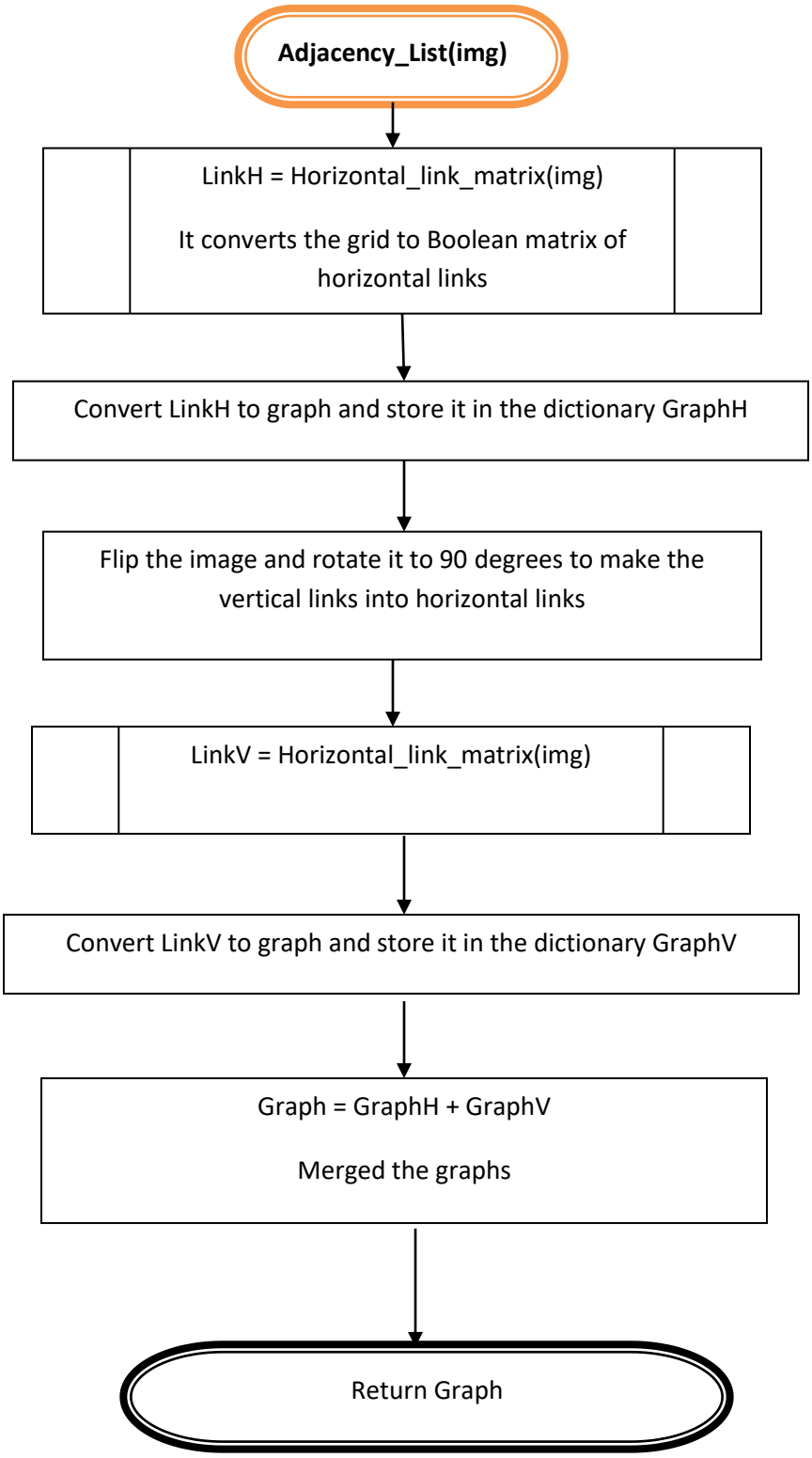
Function 1: Returns a list of all the positions of traffic light



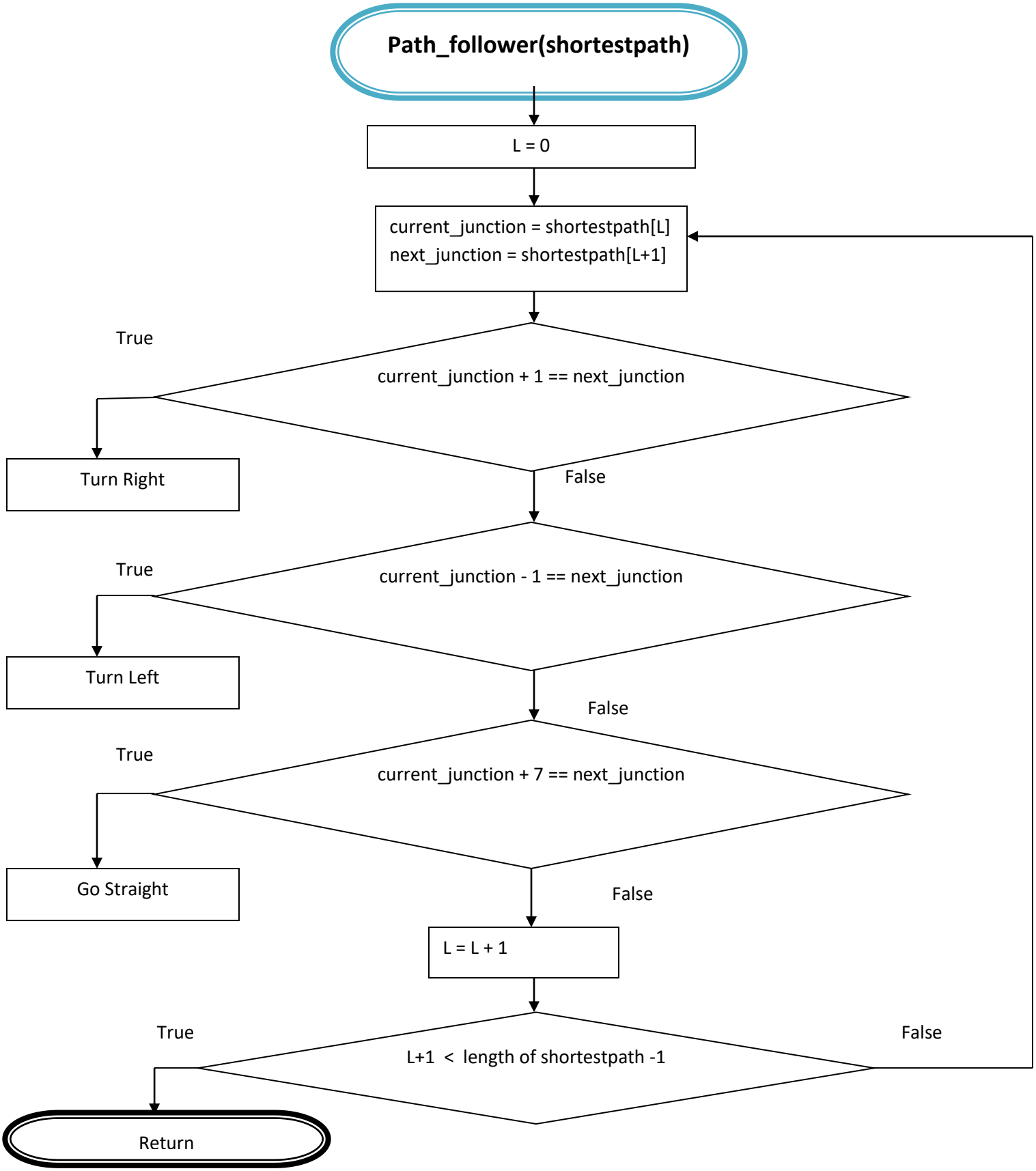
Function 2: Returns a list of all the delivery junctions along with the shape and color of the package.



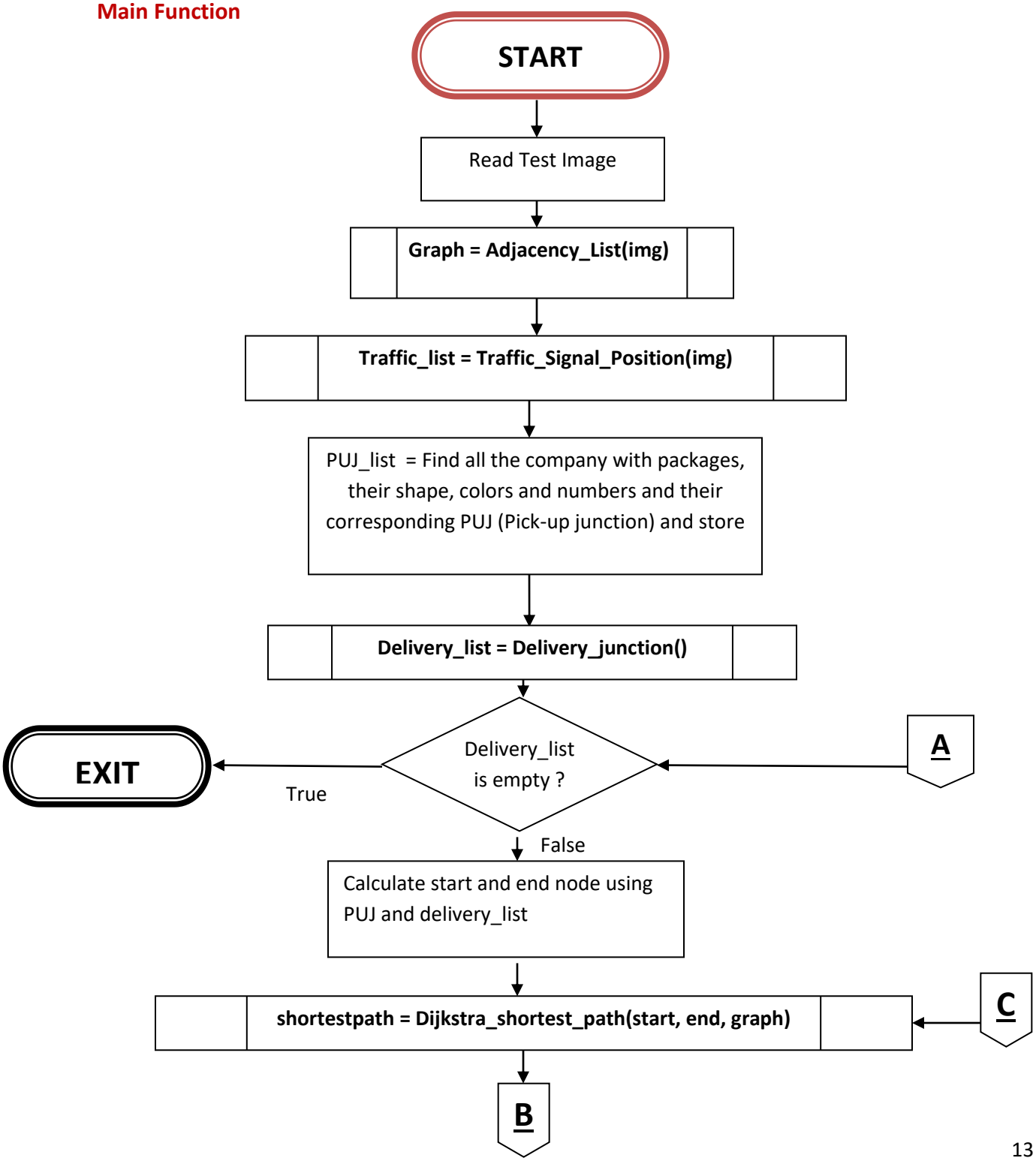
Function 3: Returns adjacency list of the graph

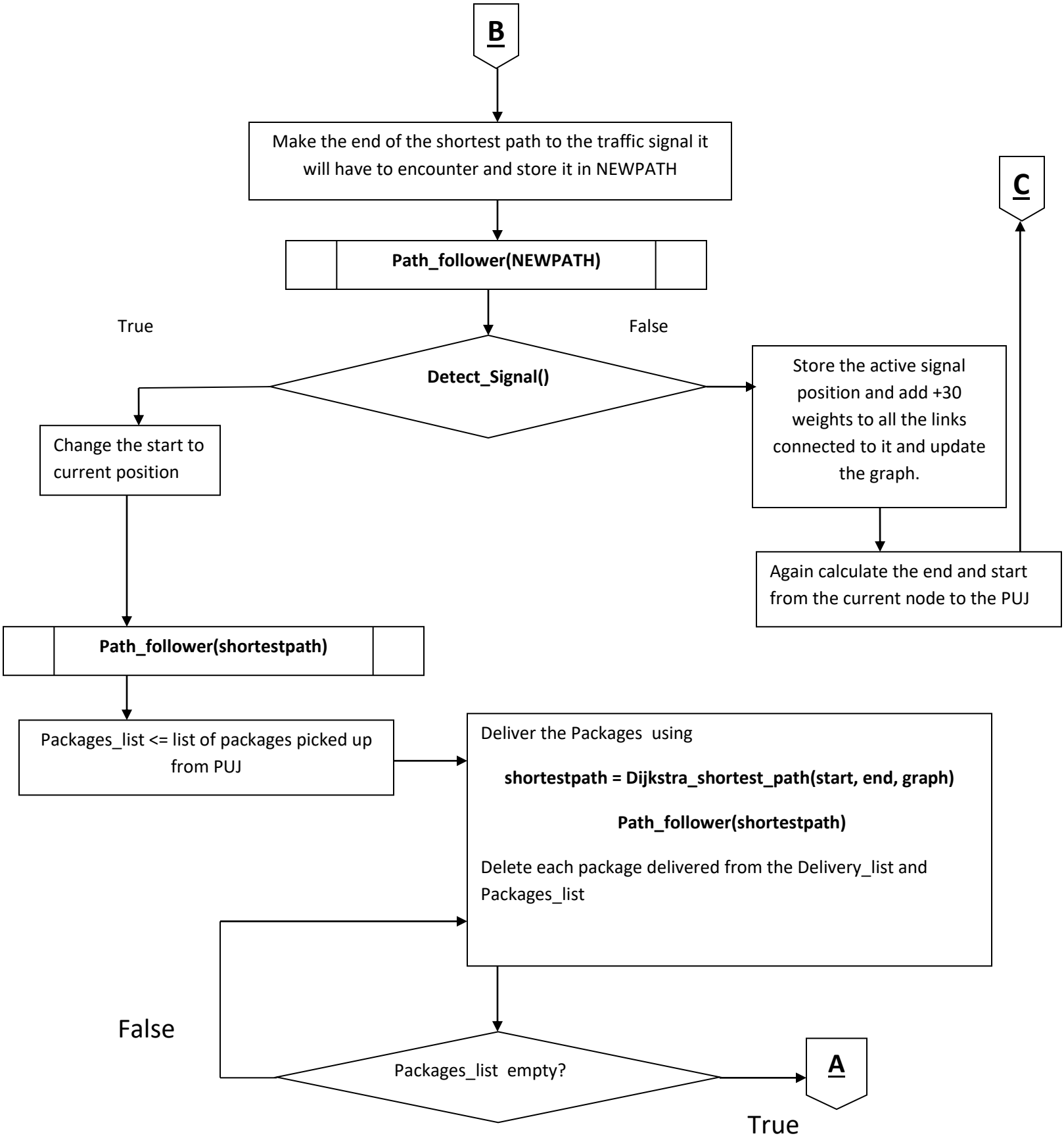


Function 4: Converts path to robot motion



Main Function




B
Make the end of the shortest path to the traffic signal it will have to encounter and store it in NEWPATH
Path_follower(NEWPATH)
True
False
Detect_Signal()
Change the start to current position
Store the active signal position and add +30 weights to all the links connected to it and update the graph.
C
Again calculate the end and start from the current node to the PUJ
Path_follower(shortestpath)
Packages_list <= list of packages picked up from PUJ
Deliver the Packages using
shortestpath = Dijkstra_shortest_path(start, end, graph)
Path_follower(shortestpath)
Delete each package delivered from the Delivery_list and Packages_list
False
Packages_list empty?
A
True

Challenges

1. **Challenge 1**
   To find the shortest path between two junctions that does not passes through any traffic signal.

   Between two junctions there can be multiple shortest paths. It takes a lot of time to find out all the multiple paths, so we had to check time spent in finding out the paths so that not so much time is wasted.

   The path with least number of traffic light was chosen and followed.

2. **Challenge 2**
   To identify the shape and color of the packages along with the position of its delivery junctions.

   The packages color and shape took a lot of time to detect correctly. It took calculations to crop out specific areas to focus on the packages.

3. **Challenge 3**
   To translate the path into motion.

   Since the motion is pre-coded, so correct timing of running motors and turning matters greatly. Even the small mistake could make the whole path incorrect.

4. **Challenge 4**
   Detecting traffic signal and rerouting.

   The traffic signal was hard to detect due to problems like shadow, flash, etc. This had to be taken care by appropriate HSV range for masking.