Assignment 3 - Collision Avoidance

Problem Statement:

To plan and execute a trajectory to goal location in a dynamic environment using velocity obstacles method.

Given: Environment and Robot Definition

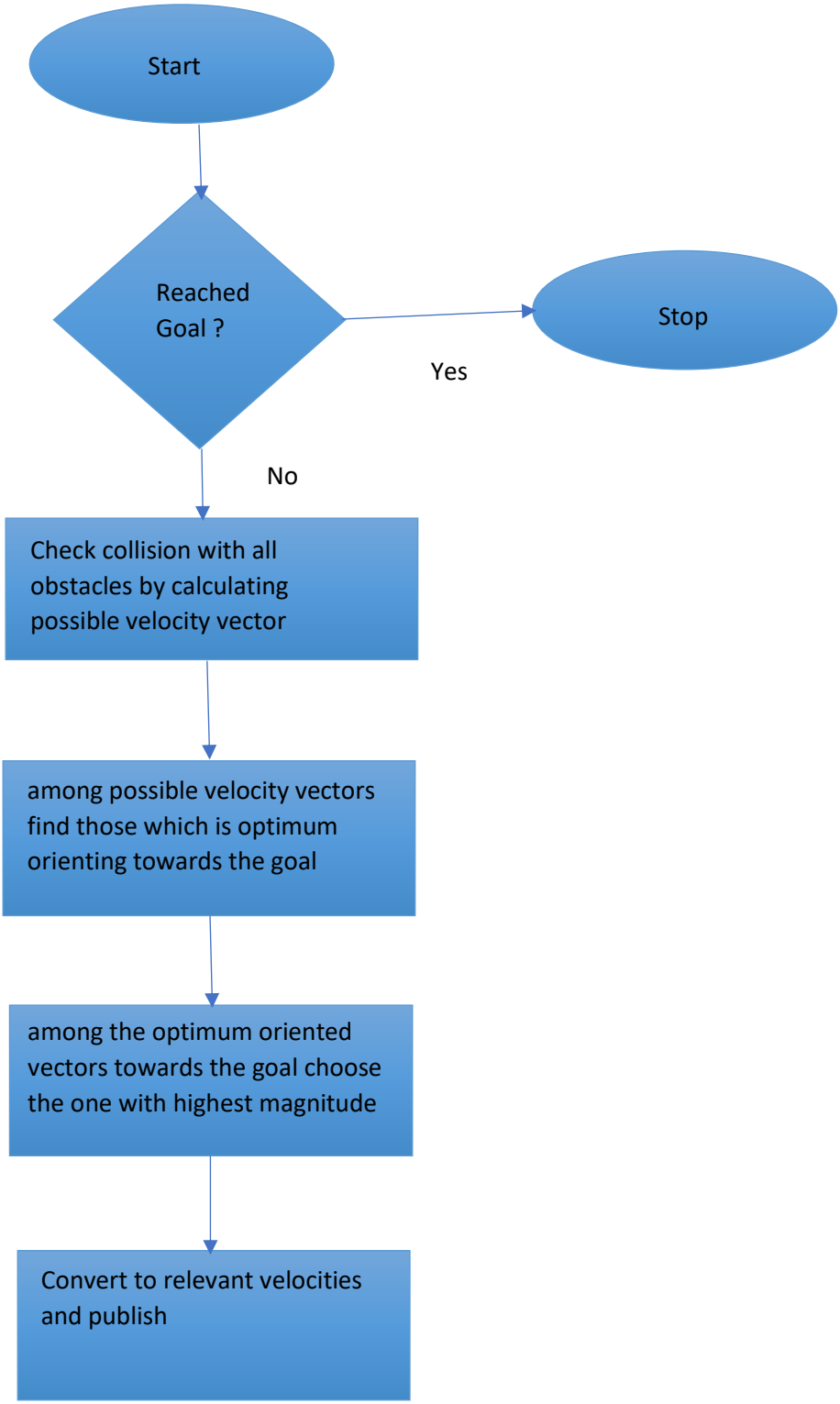
– Start = (0, 0), Goal = (5, 0)
– Robot and Obstacle Diameters, D = 0.15
– Magnitude of Maximum Robot Velocity Vmax = 0.15m/s
– Maximum Deviation in Orientation θmax = 10◦

Theory:

The bot starts from (0,0) position and is required to travel to x=5, y=0 coordinate with allowed max velocity of 0.15m/s and max deviation in orientation is 10° in any step. The robot and obstacles dia is 0.15m.
There are 2 dynamic obstacles and one static obstacle in the environment. The bot is required to reach (5,0) without collision with obstacles.

Bot diameter= 0.15m, but to avoid rubbing against each other we taken 0.2 to keep a safe margin while other obstacles.
Velocity vectors of the robot are discretized in steps of 0.01m/s and 1° in angle.
At any given position, the set of obtained vectors is referred to confirm vectors do not have collision with other. Consider bot as point robot and enlarge obstacle by adding bot dia to obstacle dia.
From the bot point, the orientation of the obstacle is calculated and the tangent lines orientation is also calculated. The tangent lines are oriented to the line joining the centre of obstacle and
robot by an angle. If the velocity is oriented between the tangent lines it will lead to collision.

Optimum velocity is taken amongst the non-colliding velocities. The selection of the best velocity is based on the TG strategies. In essence the best velocity is the one most closely oriented towards the goal with the maximum velocity. This velocity is selected and commanded to the robot. Code stops when the robot reaches to the goal.



Algorithm Collision_avoidance.py

Result :

The bot successfully moves near to the goal avoiding the collision with obstacles. The bot could not exactly reach to the goal as its velocity is steering it a bit away .

The collision_avoidance.py program is created and the plots for X-coordinate Vs time & bot path was plotted. The output.txt file is generated and saved in folder specifying the X & Y coordinate of bot at various instances. Time taken by bots to reach balanced position is 32.1 sec.