Joined Fermat's Spiral Search for Obstacle Environment and Efficient Target Collection

XianmeiLei(CalPolyPomona),AlexCardaras(PasadenaCityCollege),
WangdaLei(PasadenaCityCollege),DavidWu(PasadenaCityCollege),
KevinMacias(CalPolyPomona)

March 25, 2018

Advisor: Jamal Ashraf
Jamal Ashraf certifies that he has read this report prior to submission.
PASADENA CITY COLLEGE, CAL POLY POMONA, CA

Joined Fermat's Spiral Search for Obstacle Environment and Efficient Target Collection

Xianmei Lei, Alex Cardaras, Wangda Lei, Kevin Macias, David Wu

***Abstract*—In 2018 NASA Swarmathon, a mobile robot swarm searching and collecting targets in an environment with static obstacles in a time limit frame demands efficient searching strategies and high accuracy of target finding and retrieving. We propose a joined Fermat's spiral search (JFSS) and solutions to rapid and accurate pick-up and drop-off. The Fermat's spiral search is range expansive and resilient regardless of obstacles due to the centralizing capability. For the fetch process, the robot can rely on the first blur sight of the target to conduct the motion.**

I. Introduction

Many path planning and mapping methods are keypoint-based and highly depend on low-drift localization. However, the current sensing abilities and the performing environment with the lack of valid landmarks and lack of ground features limit the application of some common localization and mapping techniques, such as SLAM(Simultaneous localization and mapping) and VO(visual odometry). The extra motions for obstacle avoidance worsen the mapping quality. The uncertain mapping imposes a possibility of losing targets because the drift could mislead the robot away from potential target locations labeled as a hazardous area on the map. We aimed to develop a search algorithm which needs no mapping. In order to maximize the collective efforts of the swarm searching, each robot performs in a partition of the global space which results in the reduction of collision among robots both in the searching and retrieving path. The final searching algorithm is an joined Fermat's spiral modified to be able to resume and aid to avoid obstacles, combined with auxiliaries which complements the limitation of the pattern search and the handling of the obstacle disruption, for instance, travelling back to the latest locations stored when targets are seen on the way to the collection zone followed by the previous goal location.

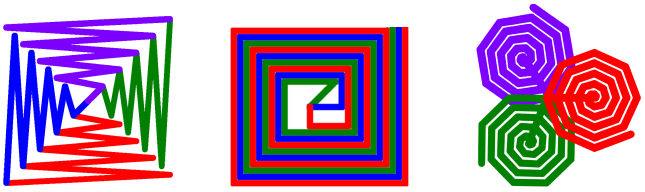
Taking the advantage of the given code skeleton which has implemented a reliable base of control and motion planning system, the project is broken down into low-level motions such as drive, gripper control and high-level motion such as searching and pick-up. The robot chooses and focuses on one of the high-level decisions by prioritizing them differently based on the present process and the sensory feedback. Searching, pick-up, drop-off, obstacle avoidance and localization and navigation were the main courses we addressed. We leveraged frequent design-build-test iterations to tune the parameters in the solutions and finalized a robust motion control. We adopted computer vision for target recognition which allows the robot to identify the target or collection zone prior to detecting the April tags attaching the target cube or collection zone.

II. Related work

Some types of the archimedean spirals have been studied in the swarm searching. For instance, Fricke [1]'s DDFS(distributed deterministic spiral algorithm), using a square spiral as a central search ensuring collection of the nearest targets first, complete coverage, and minimal repeated sampling. This pattern was studied in an error-free and obstacle-free scenario. The resuming after the interruption from the obstacle and other robots and navigation drift is challenging. A continuous curvature in Candeloro [2] as path planning for vehicles to operate in the cluster environment with safety constraints shows the simulating completion of full coverage with reduction of overturning and unnecessary heading changes. The practical modification of the spiral hints a path planning for the Swarmathon scenario also based on mobile vehicles. Differed from JDFS, it has a single center start and an endpoint different from the starting point. The research for the full traversal from a non-central start in Zhao [4] conducts connected Fermat spirals to the application of layered fabrication, demonstrating a perfect space-filling and slight under- and over-fill but high efficiency and flexibility to complex geometry. The algorithm seeks the full space filling regardless of the start and end poses. JDFS is required for 1) the treatment of returning to the initial center(collection zone/home base), 2) target collection(a form cube with April tag on 6 sides) and 3) obstacle handling.

III. Methods

A. Patterns Evaluation



(a) joined zigzag (b) square spiral (c) joined Octal Fermat's spiral

Fig. 1: Searching patterns

Three patterns are developed for this project: zigzag, square spiral and the modified Fermat's spiral. All are uniform and adaptable for any size of the swarm but with different strength. Based on the comparison(Table I) and the demand of the intensive obstacle avoidance, Fermat's spiral shows some extra abilities to be compatible with our obstacle avoidance method which is to give a temporary location to walk past the obstacle

with a new goal location to prevent that the goal location is at the obstacle position. We modify the Fermat's spiral to be a polygon open loop: a counterclockwise octal spiral. Each loop naturally has eight directions which will be an efficient heading as our obstacle avoiding method prefers. Even for the wall avoidance, the goal location can rebound to the navigation space (arena) after a few failures to reach the out-of-bound locations. This nature is resilient to searching and reduces the work to recognize the wall and to manage differently from other avoidance. The expansibility of this pattern guarantees the full traversal.

Pattern	(a)	(b)	(c)
Full coverage	X	√	√
Interruption free	√	X	X
Adaptability	√	√	√
Scalability	√	√	√
Overlapping	X	X	√
Other abilities	–	–	compatible to obstacle avoidance

TABLE I: Three patterns capability comparison

The generated waypoint noted as *(nextX, nextY, nextTheta)* of the spiral is calculated using the equations below. The initial pose*startX, startY)* is used thought the whole pattern as well as two constants - initial spiral size (*initalSprialSize*) and *sizeIncrement*. The pattern also is correlated to the current location in order to resume. More variables the current shape of the pattern: *increaseLen*: the length for the shape growth, *angle*: the heading to control the shape.

$$nextX = startX + (initalSprialSize + increase) * cos(angle)$$

$$nextY = startY + (initalSprialSize + increase) * sin(angle)$$

$$nextTheta = arctan(nextY - currentY, nextX - currentX)$$

The variables to control the shape growth are updated after each waypoint is set.

$$angle = angle + 45(deg)$$

$$increase = increase + increaseLen/8$$

The Fermat's spiral starts from the center of sub-regions for a single robot. This design leads the further search first which is seemingly less efficient. Complementarily, waypoints of the seen targets are stored on the returning path to the collection zone and will be first traveled before resuming the pattern after target drop-off is finished.

One more practical modification is added to the pattern in order to reduce the chance of accidentally breaking into the collection zone caused by the attempt to grab the target cubes near the collection zone and the backup motion after the pick-up. The spiral waypoints are discarded within two meters from the center of the collection zone. The shape of the pattern can be referred in the result section as simulation implement below.

B. Target pickup: Locating Lost Cubes

The major causes of error at the start of the pick-up process involve locking on to a cube. If the camera is unable to keep the April tag in sight, it cannot complete the pick-up process. We propose a solution to fix two of the most common cases that cause a robot to lose track of a cube during pickup.

Case 1: A common issue occurs while the robot is traveling in a straight line and catches a glimpse of the cube at the far left or right of the camera image. The robot may be unable to slow down fast enough, resulting in the robot driving past the cube and losing sight of it. The solution is as simple as rotating towards the last known direction of the cube.

Case 2: Another common issue that causes the robot to lose track of the cube occurs while the robot is rotating. While rotating, the robot may momentarily see the April tag of the cube in one of its frames, but is unable to slow the rotation down quick enough to keep the cube in sight. Unlike the prior case, the solution is not as simple as rotating towards the direction of the last known location of the cube. This is due to the low frame rate of the camera not recording all the positions of the cube. This may result in the robot Turing the wrong direction while attempting to relocate the cube. The solution is to keep track of the rotational velocity of the robot the moment it first sees the April tag. With this rotation, it is easy to locate the cube by slowly rotating the opposite direction until it sees the cube again. Through physical testing, we determined that 0.1 radian per second is a sufficient angular velocity for relocating the cube. Values higher than 0.1 may cause the image to blur, resulting in the April tag being unable to be read.

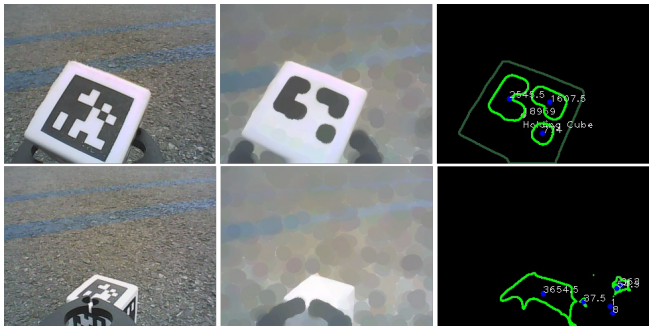


Fig. 2: target pick-up confirmation

Aside from keeping track of the cube, the overall pick up process has also been improved upon. Through physical testing, it became apparent that the robot was over correcting its angle the closer and closer it got to the cube. To fix this issue, the rotation of the robot was set to the following equation:

$$robotrotation = -blockYawError/(0.6/dist)$$

This equation states that the robot should rotate less the closer it gets to the cube. This forces the robot to do all of its major rotational adjustments before it gets too close to the cube.

By performing OpenCV operations on the image we can approximate the distance of the cube from the camera. This is done by filtering the image and observing the area of the largest white space(the cube) in the frame. If the area of the cube is greater than 8000 px, then it can be assumed that the cube is being held up by the rover. To ensure that the rover has a strong grip on the cube, the rover must be able to sustain a white area greater than 8000 px while backing up for at least 8 frames. The first row of images in Fig 2 shows a successful pick-up. The area of the cube is 18959 px in one frame and continues to show similar results as the rover backs up. The second row of images in Fig 2 shows an unsuccessful pick-up. With an area of only 3654 px, it is easy to see that the cube is not in the grasp of the rovers claw and the rover has failed pick-up.

C. Target returning: Predefined stages of motion based on collection zone tag counting

Immediately upon returning the target to the collection zone, the decision of motion control based on the first sight of the edge of the collection zone with at least one collection zone tag(or home tag) being detected usually leads to the best estimate of the relative location of the robot to the collection zone. Therefore, the predefined stages of motion based on the first home tag counting(PSM) can cater to the need of making a one-time appropriate decision in this process. Three general cases are handled by this method(see Algorithm I). Note that 45 degree is designed to compensate the common offset angles incurred by the nonvertical heading of the robot to the collection zone. Although this turning cannot satisfy all cases, it generally wont make the robot drop the target out of the collection zone since the distance traveled in the last stage is not too far.

D. Collection zone detection

We applied the same method as the pick-up uses to process the image using OpenCV when dealing with collection zone detection. We obtained all the contours, including their areas and centroids in the image. The detection only focuses on the contour whose area is just large enough(3000 px in our case). However, meeting this criterion is not enough since the cube being held by the robot or a cluster of cubes can appear to be a collection zone. To distinguish the collection zone, we restricted the collection zone candidates to be the one whose centroids y coordinate in the image is less than 43px. This y value ensures that the robot would not run into a cluster of cubes because, by the time the robot approaches the cluster, the robot shouldve been able to determine if the collection zone is just a cluster. Once the potential collection zone is located, a set of motions will be prioritized based on the x coordinate of the potential collection zones centroid until the robot finds the first home tag. For example, if the x is less than 45% of the image width, the robot needs to turn left. If the x is greater than 55% of the image width, the robot needs to turn right. The middle 10% of the image width is reserved for the case when the robot needs to move forward. An example of how the centroid of collection zone is located in the image is shown in Fig 3. The coordinate of the centroid is shown below the words “Home Base”.

Algorithm 1 predefined stages of motion (PSM)

procedure PSM
  $hTagC \leftarrow$ the count of home tag
  $l \leftarrow$ the count of home tag in the left-side vision
  $r \leftarrow$ the count of home tag in the right-side vision
  $actualDistance \leftarrow$ home tag visual distance
  $collectionZoneLen \leftarrow$ collection zones length
  $COUNT = 3$
  if $hTagC > 0$ **then**
    Stage 1: $Stopmoving$
    if $l - COUNT \geq r$ OR $r - COUNT \geq l$ **then**
      Stage 2: Move forward: actualDistance + 0.25
      * collectionZoneLen
    else if $l - COUNT > r$ **then**
      Stage 2a: Turn 45 degrees to the left
      Stage 2b: Move forward 0.25
      * collectionZoneLen
    else
      Stage 2a: Turn 45 degrees to the right
      Stage 2b: Move forward: actualDistance + 0.25
      * collectionZoneLen

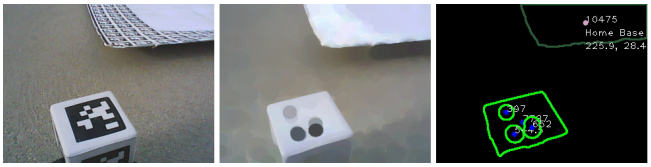


Fig. 3: home detection

E. Obstacle avoidance: bubble rebound algorithm

Swarmathon robot uses three ultrasound sonars to scan obstacles. We defined obstacles as one of the sonar range readings is equal or smaller than 0.9 meter. We applied bubble rebound algorithm(Susnea[3]) to perform avoidance with simplification: when the robot moves toward the goal location which is inside or on the other side of the obstacle, the robot should head to a clear area based on the sonar range reading (see Fig. 4). If robot could not reach the goal location within certain attempts(three attempts in our implement), as Fig. 5 shows, it will be assigned to a new goal location with a new heading. This mechanism intends to eliminate the decaying of the pattern. Finally, robot moves away from the obstacle.

IV. Experiment

A. JFSS in target-free simulation

B. Localization using EKF: a fusion of sensor data

Using the base code location arrangement, two sources of location information are available: Map which is an EKF fusing GPS, IMU and encoder data and Odom which is EKF

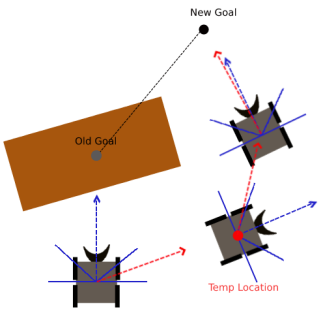


Fig. 4: Robot avoidance illustration

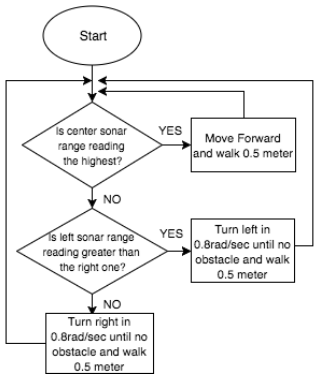
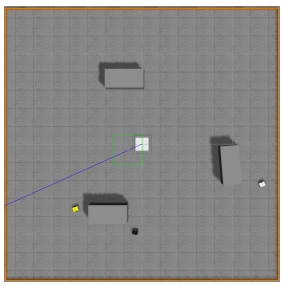
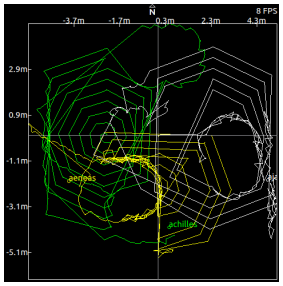


Fig. 5: Obstacle handling process



(a) three robots in the simulation world



(b) simulation map

Fig. 7: Simulation testing with obstacles and target free

fusing IMU and encoder. Map location drifts slowly than Odom's after distancing travels. Fig 6 is the average result of three runs that robot ran back and forth for 2 meters and tried to return to the start position. For a more accurate collection zone location, when the game starts, the robot would stop for fifteen seconds to average all the Map location inputs. Robot updates the collection zone location once robot sees the specific April tag and time elapsed from the last update is more than 150 seconds.

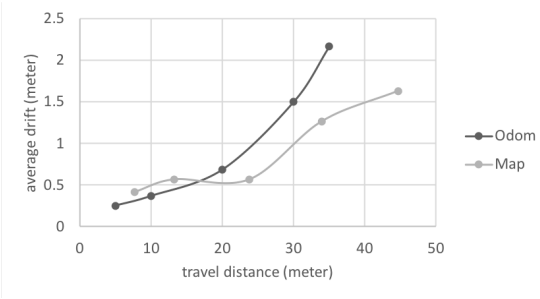


Fig. 6: Drift distance tested for Map and Odom

C. Sonar noise ignorance

Ultrasound sonar is very sensitive to the environmental factors. As we experienced, sonar had a frequent reflection in a windy weather as well as a tough asphalt ground in the parking lot. This noise triggered unnecessary reactions which interfered all the behaviors since obstacle avoidance has the highest priority in most of the process states. We conducted a very simple solution instead of using any filters. The avoiding behavior is only activated if the sonar has been called for four times within one second. The threshold cannot be higher or robot delays the reaction and hit the actual obstacles and cannot be lower for active ignorance of fake sonar calls.

D. Target pick-up and drop-off

Pick-up and drop-off were developed based on the physical testing but not simulation. Here is the physical performance: video: Pasadena City College Pickup and Dropoff 2018 (https://www.youtube.com/watch?v=2wy240HFOMs)

V. Result

We ran JFSS with three robots in the simulation with a clean world but three obstacles. The result can be seen in Fig 7. The holes of each spiral in simulation map (b) involve the obstacles. The green path has the wavy portion on the left side which indicated wall avoidance. This shows that the spiral can be recovered after cutoff under our design. The white path in the map (b) has another blank in the top left other than the one occupied by the obstacle on the right. This corresponds to the skip of goal locations near collection zone.

We ran JFSS in the simulation with uniform and cluster distributions in four trials with three simple obstacles respectively. The result in Table II, however, didn't show the advantage of the pattern for some reasons. For example, the pick-up and drop-off parameters are not fully tuned for the simulation. Some critical issues were not solved such as the robot would run over the collection zone and push the cubes out. Further discussion is made in the CONCLUSION section.

Distribution	1	2	3	4	average
uniform	10	10	13	31	16
cluster	10	13	14	15	13

TABLE II: Simulation Testing Result

VI. Conclusion

JFSS possesses many potentials for swarm search work: adaptability and compatibility to the obstacle world. However, in our testing scenario, we need more perfection on the basic tasks, such as localization, pick-up and drop-off and more efforts to detect and prevent the misbehaviors. This year, we followed the base code motion control and planning design. We spent a lot of time in understanding and learning to work with it correctly but still, we did see the malfunctioning moments due to lack of the time devoted to debugging them. All of these factors hinder the play of JFSS.

Since its time-consuming to set up a full-scale physical testing, the simulation testing is still necessary to make sure all controllers work seamlessly. We need to spend more time in testing the whole logic flow after introducing some functionalities into some controllers.

REFERENCES

[1] A distributed deterministic spiral search algorithm for swarms, author=Fricke, G Matthew and Hecker, Joshua P and Griego, Antonio D and Tran, Linh T and Moses, Melanie E, booktitle=Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on, pages=4430–4436, year=2016, organization=IEEE.

[2] Mauro Candeloro, Anastasios M Lekkas, Asgeir J Sørensen, and Thor I Fossen. Continuous curvature path planning using voronoi diagrams and fermat's spirals. *IFAC Proceedings Volumes*, 46(33):132–137, 2013.

[3] Ioan Susnea, Viorel Minzu, and Grigore Vasiliu. Simple, real-time obstacle avoidance algorithm for mobile robots. In *8th WSEAS International Conference on Computational Intelligence, Man-Machine Systems and Cybernetics (CIMMACS09)*, 2009.

[4] Haisen Zhao, Fanglin Gu, Qi-Xing Huang, Jorge Garcia, Yong Chen, Changhe Tu, Bedrich Benes, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. Connected fermat spirals for layered fabrication. *ACM Transactions on Graphics (TOG)*, 35(4):100, 2016.