

The Navigation Functions Approach for the Label Anti-Overlapping Problem

Stavros Kakos and Kostas J. Kyriakopoulos
Control Systems Laboratory
National Technical University of Athens
Greece

1. INTRODUCTION

We aim at developing a System enabling the *collision-free* motion of aircraft labels, treated as autonomous “robots” in this work, using an extension of our Decentralized Navigation Functions (NF) methodology [6].

NFs guarantee collision avoidance in perfectly known and stationary environments where point-mass robots move in a sphere world. To achieve that, suitable coordinate transformations are used to map the original workspace to a sphere world - a disc encapsulating an arbitrary number of smaller disjoint discs, representing obstacles. Furthermore, point-mass robots are created by simply by deducting the volume from the label and adding it to the obstacles.

Here, we consider a “decentralized” approach where each controlled label (robot) treats as the labels, the position symbols/tracks history and the leader lines of all other aircrafts obstacles. Those are considered to be quasi-static due to the fact that the update rate of the position symbol of the aircraft is slow when compared with the anti-overlapping motion of the labels. Early MATLAB simulations depict the principles of our methodology. The full and customized design along with the OO code will be available at the final report.

2. PROBLEM REQUIREMENTS

The general requirements are:

- i. The relationship between the aircraft position symbol and its label is established by the means of a leader line connecting them.
- ii. The ‘selected’ aircraft label being displayed as in-filled
- iii. The connecting point between the leader line and the label could be any point located anywhere along an edge of the label, for example the midpoint and the corners of each edge.

The anti-overlap requirements of the system analyzed are (in descending priority):

1. **Callsign** should never overlap.
2. The **label** of an aircraft should not overlap with the **position symbol/track history** of another aircraft.
3. **Labels** should not overlap.
4. The **leader line** of a label should not cross the **leader line** of another aircraft’s label.
5. The **leader line** of a label should not cross the **label** of another aircraft.

6. The **label** of an aircraft should not overlap with the **speed vector** of another aircraft.
7. The **leader line** of a label should not interfere with the **position symbol/track history** of that aircraft.
8. The **leader line** of a label should not interfere with the **position symbol/track history** of another aircraft.
9. The **leader line** of a label should not interfere with the **speed vector** of that aircraft.
10. The **leader line** of a label should not interfere with the **speed vector** of another aircraft.

If inclusion of other features is needed:

11. The **label** of an aircraft should not overlap with **beacon** symbols, **airway** symbols, **DFL**, **crossing path labels**...

Oscillation amplitude of the speed vector relative to the route of the aircraft, which does not trigger a modification of the label position: (ex: +/- 5°).

The leader line length must have:

- Minimum length (ex.: 5 mm).
- Maximum length (ex.:50 mm).
- Preferential length (*default*) (ex.:20 mm).

Regarding the label motion:

- The Optimum label movement distance in X, Y axes: (ex.: 10 pixels, or 2.5 mm).
- The Maximum label movement distance in X, Y axes: (ex.: 40 pixels).

The label rest position is:

- Identical for all aircraft (*default option*): 90°, 135° (*default*), 225°, 270°... from the speed vector.
- Depending upon route orientation: label position at 90°, 135°, 225°, 270°... from the speed vector for each of the following orientation ‘quadrants’: 0°-89° (*default* 90°), 90°-179° (*default* 135°), 180°-269° (*default* 90°), 270°-359° (*default* 135°). There is also a constraint which implies that, in some of the one-way route cases, the selected label rest position would be displayed symmetrically on the opposite side (i.e. 270° instead of 90°, 225° instead of 135°, etc).

Example: With the label rest position located on the right hand side of the aircraft route (for example at 135°), if the opposite direction route is also located on the right hand side, the situation illustrated in Figure 1 would occur:

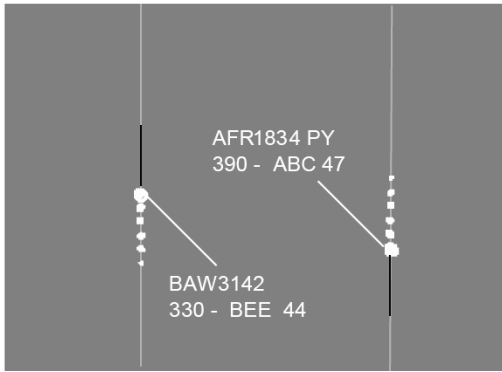


Figure 1: Example of one-way routes placed on the left of each other, with the label rest position at 135°.

- For arriving aircraft: 90°, 135° (*default*), 225°, 270°... from the speed vector, for each of the possible aircraft destination (Fig. 2).

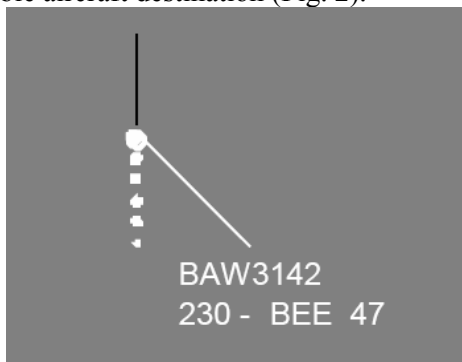


Figure 2: Illustration of the default values for the label rest position (135° from the speed vector) and for the leader line length (20 mm).

3. PATH-PLANNING

This chapter introduces the methodology of robotic path-planning using Navigation Functions. First, the potential fields approach is presented to demonstrate the need for improvement to introduce next, the navigation functions methodology.

Potential Fields

Potential fields make a robot moving in a gradient vector field. This can be visualized as a ball rolling down a hill. The basic features of this approach are:

- An attractive potential $\Phi_{attractive}(q)$ is assigned to the target.
- A repulsive potentials $\Phi_{repulsive}(q)$ are assigned to all obstacles.
- The sum of those potential fields: $\Phi(q) = \Phi_{attractive}(q) + \Phi_{repulsive}(q)$ is used to move the robot through the gradient descent: $\dot{q} = -\alpha \cdot \nabla \Phi(q)$ $\alpha > 0$ demonstrated in

Potential field techniques have been successful for real-time obstacle avoidance in changing environments, but for motion planning, there are several limitations. One major problem is the spurious local minima (**figure**). To escape these, one must

resort to either randomizing techniques or navigation functions.

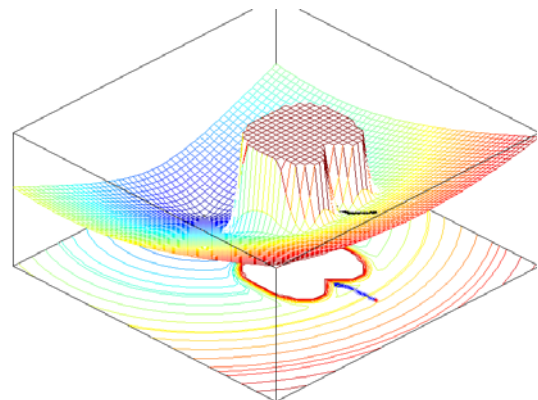
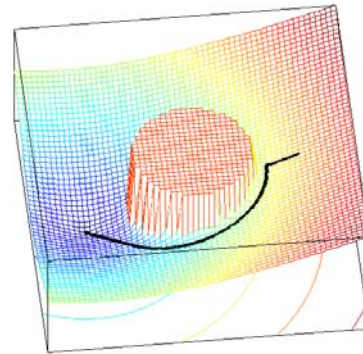


Figure 3: Potential Function surface and robot trajectory: (a) converging to destination (b) with local minimum

Navigation Functions

A Navigation Function (NF) ϕ is a potential field appropriately designed so that it has a unique minimum within F , the robot free configuration space, and this is at the goal configuration. NFs:

- require information about the complete workspace, and
- consider the case of a point mass robot moving in a generalized sphere world.

The main attributes of NF are:

- Almost global convergence: from almost every initial condition is achievable, due to the unavoidable appearance of interior saddle points. Every field has at least as many saddle points as there are internal obstacles, causing no practical difficulties as it can be guaranteed that “few” initial conditions will “get stuck” on them.
- Sphere Worlds: The navigation problem is solved:
 - in a compact, connected subset of E^n .
 - with a boundary formed by disjoint union of finite number of spheres.
 - the valid sphere world provided obstacle closures are contained within the workspace.
- Properties of Navigation Functions remain invariant under diffeomorphisms (smooth, one-to-one and with no smooth inverse maps between two spaces).

These preserve the properties of navigation functions. This implies that two navigation problems no matter how they differ in geometric detail are actually the same if there exists a coordinate transformation mapping one to the other. This suits our problem because obstacles are not sphere like and need to be transformed (Fig. 4).

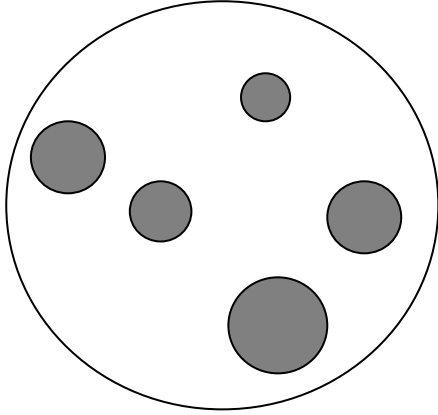


Figure 4: A typical example of a Sphere World.

Consider the real-valued map constructed on the robot's free configuration space, $V : F \rightarrow R$, which has a unique minimum at q_d , the goal configuration, and is uniformly maximal over the boundary of F (q_d must be specified in the interior of F). For example, V on a planar configuration space can be visualized as a "sculptured" two-dimensional surface constructed over F (Figure 5).

Path planning takes:

- Exponential time in the number of degrees of freedom;

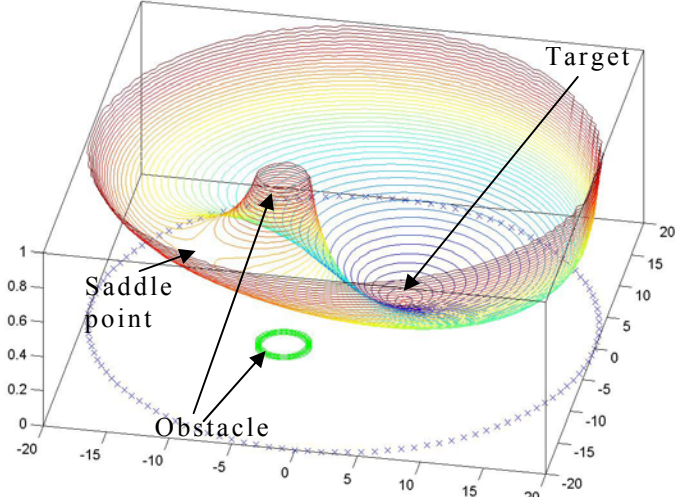


Figure 5: A suitable navigation function.

- Polynomial time in the complexity of the C-obstacle region (number of surface patches and degree of their algebraic equations).

4. ENVIRONMENTAL MODELLING

This chapter describes the methodology of modelling the environment in which every label (robot) moves.

In particular, the transition from Work Space to Configuration Space is presented. The robot becomes a mass-moving point altering the volume of the obstacles. The obstacles are approximated with properly chosen functions.

4.1 Preliminary Notions

The aircraft labels must avoid overlap with the obstacles in their environment navigated using the navigation functions methodology.

- Each label considers a different environment consisting of the obstacles that are included in a circle of radius of maximum length of leader line, and having as a centre the position symbol of the aircraft.
 - Labels are chosen to move one at a time by an infinitesimal move. Thus during every label movement - taking advantage of the fact that update rate of position symbol of the aircrafts is very slow - the environment is considered as static and known.
- The first step is to decompose the obstacles in convex polygons: the label, the leader line and the position / track history rectangles, depicted clearly in Figure 6

The leader line of every moving label is not considered as an obstacle because it is created wrt the position symbol and the label position.

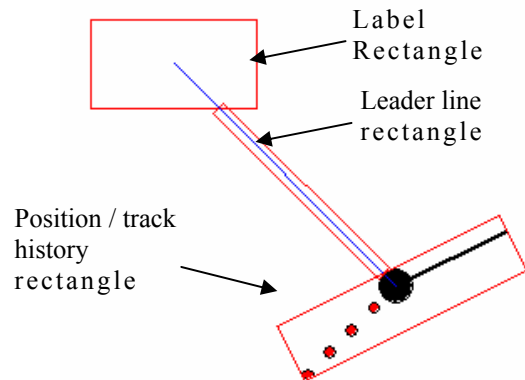


Figure 6: Obstacle decomposition in three rectangles.

4.2 Definitions

This section contains some basic definitions of robotic motion planning. First, a **robot configuration** is a specification of the positions of all robot points relative to a fixed coordinate system. Usually a configuration is expressed as a "vector" $q = (x, y, \theta)$ of position & orientation parameters. The label is a robot which cannot rotate, therefore the angle θ is always zero and the reference direction is constant. The reference point is the centre of every label. The environment in which the robot operates is called its **Work Space (WS)**, consisting of a set of obstacles with which the robot should not intersect. This space is static. A complete geometric description of the work space is also available. Representing the robot as

a point, called a configuration, results in a space with all possible robot configurations. This is called **Configuration Space (CS)**. The *free configuration space* F is a subset of CS ($F \subset CS$) obtained by removing all configurations involving intersection of the robot with physical obstacles or intersection between the robot's links. The regions removed from CS are referred to as configuration-space obstacles.

4.3 Representation of Robot (Label) as a Point

In order to represent the robot as a point, one must alter the work space to configuration space simply by sweeping away the volume of the robot, adding it to the perimeter of the obstacles (known as *Minkowski sum*) and subtracting it from the perimeter of the work space (known as *Minkowski subtraction*) always wrt the chosen reference point (Fig. 7).

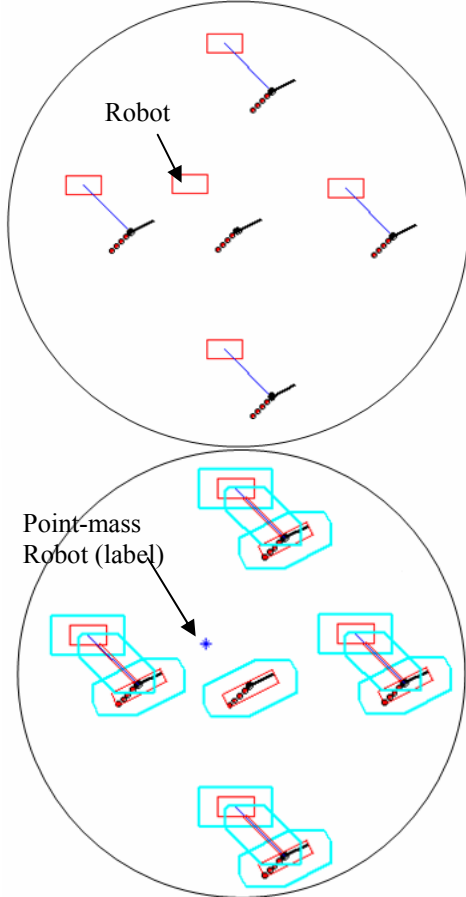


Figure 7: Workspace with 4 obstacles, a moving label and its configuration space.

We devised an algorithm achieving the representation of the robot as a point. The algorithm can be applied for convex obstacles and a convex robot. First, the algorithm sorts out the 3 different parts of every obstacle. These are the 3 rectangles:

- one containing the speed vector and the position/track history dots,
- the label, and

- one with the leader line.

For each of these 3 rectangles the algorithm places the robot in all vertices and keeps the positions of the robot that are outside the volume of the rectangle. Then the algorithm adds the outer lines of the robot and calculates the Minkowski sum with no reference point, or adds them to the reference point of the label (which is set to the centre), respectively.

Figure 8 illustrates the Minkowski sum separately to a typical obstacle that consists by 3 rectangles. The red colour represents the physical dimension of the obstacle, the green is the robot that is added in all vertices of the obstacles, the blue colour is the complete Minkowski's sum without reference point and the light blue is the Minkowski sum with reference point in the middle of the robot.

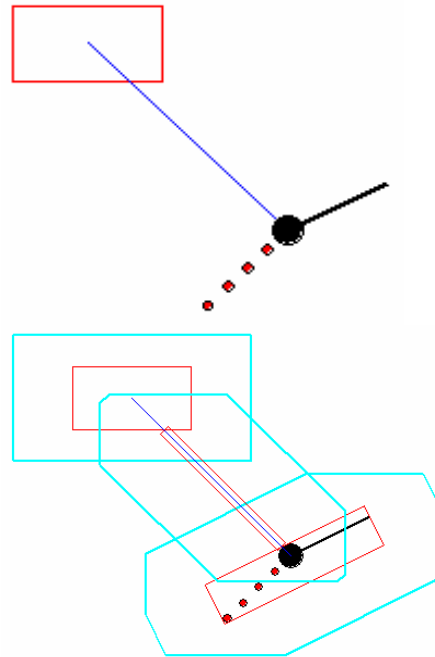


Figure 8: Minkowski sum of an obstacle.

5. OBSTACLE APPROXIMATION WITH N-ELLIPSOIDS

After obtaining the configuration space every polygonal obstacle must be approximated by a smooth curve (at least a $C^{(2)}$ function) to apply navigation functions. An ellipsoid is a higher dimensional analogue of an ellipse. The equation of a standard ellipsoid in an x - y Cartesian coordinate system is:

$$\left(\frac{x-x_c}{a}\right)^{2 \cdot n} + \left(\frac{y-y_c}{b}\right)^{2 \cdot n} = 1 \quad (2.8)$$

where a and b are fixed positive real numbers determining the shape of the ellipsoid and (x_c, y_c) is the centre of the ellipsoid. The use of ellipsoids is to satisfy the $C^{(2)}$ function requirement needed to use navigation functions.

If an n -*ellipsoid* is used to approximate a rectangle, as $n \rightarrow \infty$ the n -*ellipsoid* tends to have exactly the shape of the rectangle. The volume of the n -ellipsoid is less than the actual volume of the rectangle (Fig 9). If the shape for approximation is not a rectangle but a polygon, the volume of the n -ellipsoid is greater than the actual volume of the rectangle. In the figures following this is clear to the rotated leader-line polygon approximation as well as in the position/tracks history dots polygon.

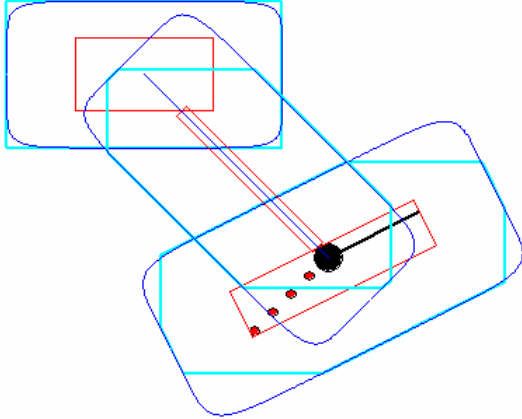


Figure 9: Obstacle (light blue) representation with 4-ellipsoid (blue).

Due to the use of n -ellipsoids, the volume of the obstacle can vary from slightly smaller to slightly greater than the actual size with satisfactory approximation. The advantage of using n -ellipsoids is that they are analytic and every polygon can be described by a closed form function. Thus they can be easily used in the navigation functions methodology.

So far the robot's obstacles have been represented into the configuration space. Briefly, each obstacle was decomposed into three parts. These parts are the label, leader line and position/tracks history rectangles. For every rectangle the Minkowski sum was applied respectively to the robot's centre, in order to create a point-moving robot. The resulting shapes were approximated by n -ellipsoids in order to have a function describing them (necessary for navigation functions construction).

6. IMPLEMENTATION OF NAVIGATION FUNCTIONS

A few notions related to NF are presented:

- **Star-shaped sets** include every convex set (although many non-convex sets are also star shaped) and are characterized by their "centre point", from which all the rays cross their boundary once and only once.
- A **tree-of-stars** is a finite union of overlapping stars whose adjacency graph is a tree.
- A **forest-of-stars** is an n -dimensional star-shaped set, T_0 , punctured by an arbitrary number of smaller

disjoint tree-of-stars obstacles, $T_i, i=1, \dots, M$. In the special case where each tree of stars consists of one star, the resulting space is a star world.

- **Distance-to-the-goal function**

$$\gamma_\kappa(q) = \|q - q_d\|^\kappa$$

where $\|\cdot\|$ is the Euclidean norm, q is a robot configuration, q_d is robot's goal (destination) configuration and $\kappa > 0$ is a parameter.

- **Obstacle implicit representation:** The various configuration-space obstacles will appear in the constructions to follow via their implicit representation i.e. the obstacle function:

$$\beta_i(x) = \left(\frac{x-x_c}{a}\right)^{2n} + \left(\frac{y-y_c}{b}\right)^{2n} - 1$$

that takes negative values inside the obstacle boundary, exactly zero on the boundary of the obstacle and positive outside the obstacle boundary.

- **1st Step: find M (model sphere world)**

For the considered problem the configuration space is contained into a sphere, so the sphere world boundary is chosen exactly the same as the configuration space boundary. If it was not, one would have to transform it. So what is left is to determine the position of the spheres wrt to the following specifications.

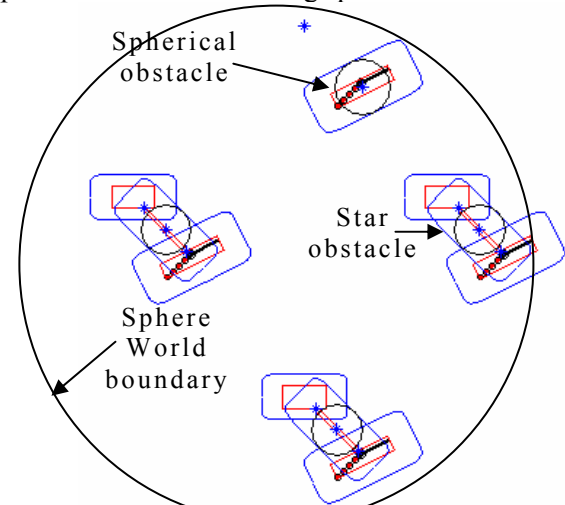


Figure 10: A model sphere world M .

Specifications

1. The transformation becomes especially simple if the spheres are placed in such a way that their centres coincide with the stars' centre (Figure 10).
2. The formula for choosing the parameter of the transformation becomes simpler if the radii of the internal spheres are chosen sufficiently small so that the Euclidean spheres are completely contained in the respective stars.
3. Similarly, the outer sphere (bound of the configuration space) is chosen sufficiently large so that it contains the outer boundary of the star world.

4.If two or more stars overlap, then purging transformation is applied and a sphere is placed in the remaining of purging star centre.

5.Finally, it is convenient to set the goal point in M , p_d , to be identical to the desired goal in F , q_d .

2nd Step: Purging transformation. Transition

to model star world \hat{F}

Each obstacle is considered as a *tree-of-stars*, consisting of stars. The root of each tree-of-stars is the leader line n-ellipsoid for symmetry sake. The maximal tree depth is 2, since the root is considered with depth one. Taking into account all the tree-of-stars (obstacles) in the configuration space, a *forest-of-stars* F is create. Purging transformation will be applied and convert the given *forest-of-stars* F to a model star world \hat{F} . The following specifications stand.

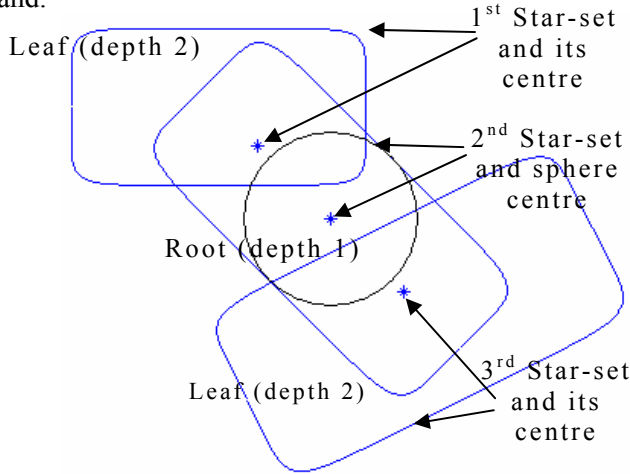


Figure 11: A typical tree-of-stars (blue).

Specifications

- 1.The centre point of each star is contained in its parent in the tree.
- 2.Each of the stars is connected to its parent via a unique patch.
- 3.This patch is star shaped wrt the centre point of the star with which it is connected (i.e. the rays from this centre intersect with the patch at most once). Applying purging transformation to a tree-of-stars with maximal tree depth d results to a tree-of-stars with maximal tree depth $d-1$. One must apply purging transformation as many times as needed till the maximal tree depth d becomes one (tree-of-stars become single stars). The given forest-of-stars F (all the 3 part-obstacles of configuration space), after applying purging transformation, a model star world \hat{F} is found. In our problem, the maximal tree depth is 2 and purging will be applied once.

3rd Step: Transformation from model Star

World \hat{F} to model Spheres World M

The n-ellipsoid that remains after the purging transformation is considered a star set and under this transformation becomes a disk. Figure 12 depicts two sets that are both topological discs. D is a standard-Euclidean disk with centre p_i and radius ρ . S is a rather general shape called a "star shape" with "centre point" $q_i \in S$.

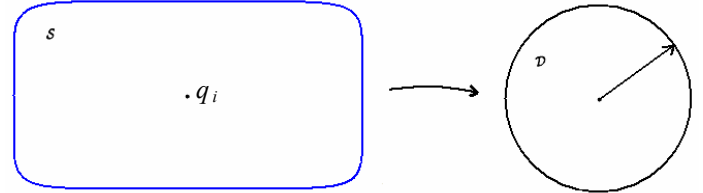


Figure 12: A change of coordinates transforms the star-shaped set into a sphere.

4th Step: Generate navigation function ϕ

5th Step: Robot Navigation

Robot moves materializing gradient descent:

$$q((n+1) \cdot \Delta T) = q(n \cdot \Delta T) - \alpha \cdot \nabla \phi(q(n \cdot \Delta T)) \cdot \Delta T$$

Tuning of Parameters

Implementation of navigation functions requires parameter tuning. Certain parameters must be chosen larger than a lower bound in order to eliminate local minimums (Figure 13). Above that lower acceptable value the robot follows different trajectories (Fig 14).

7. SUMMARY

In this paper, an implementation of navigation functions is introduced as a possible solution to the label anti-overlap problem. The labels approach at goal destinations following a "smooth" trajectory without hitting obstacles, as the plane move. Almost all negative-gradient trajectories converge to the desired destination q_d , and none can leave the free space F . The shape of the trajectories varies in respect to parameters' values; having in mind that not proper tuning may result to no-convergence.

BIBLIOGRAPHY

- [1] Latombe, J.C. (1991). «Robot Motion Planning». Kluger Academic Pub.
- [2] Rimon, E. and D. Koditschek (1992). «Exact robot navigation using artificial potential functions». IEEE Transactions on Robotics and Automation 8(5), 501-5
- [3] Rimon, E. and D. Koditschek (1990). «Robot navigation functions on manifolds with boundary». Advances in Applied Mathematics, vol. 11: 412-442.
- [4] D. Theodorakatos (2004), «Contribution in the development of algorithms for transforming geometrical shapes in sphere equivalent», Diploma Thesis, Control Systems Lab, National Technical University of Athens, Greece

[5] D. Xatzistefanou (2004), «Modelling of polygonal space for the implementation of Navigation Functions», Diploma Thesis, Control Systems Lab, National Technical University of Athens, Greece

[6] D.V. Dimarogonas, S.G. Loizou, K.J. Kyriakopoulos and M.M. Zavlanos “A Feedback Stabilization and Collision Avoidance Scheme for Multiple Independent Non-point Agents”, to appear in AUTOMATICA

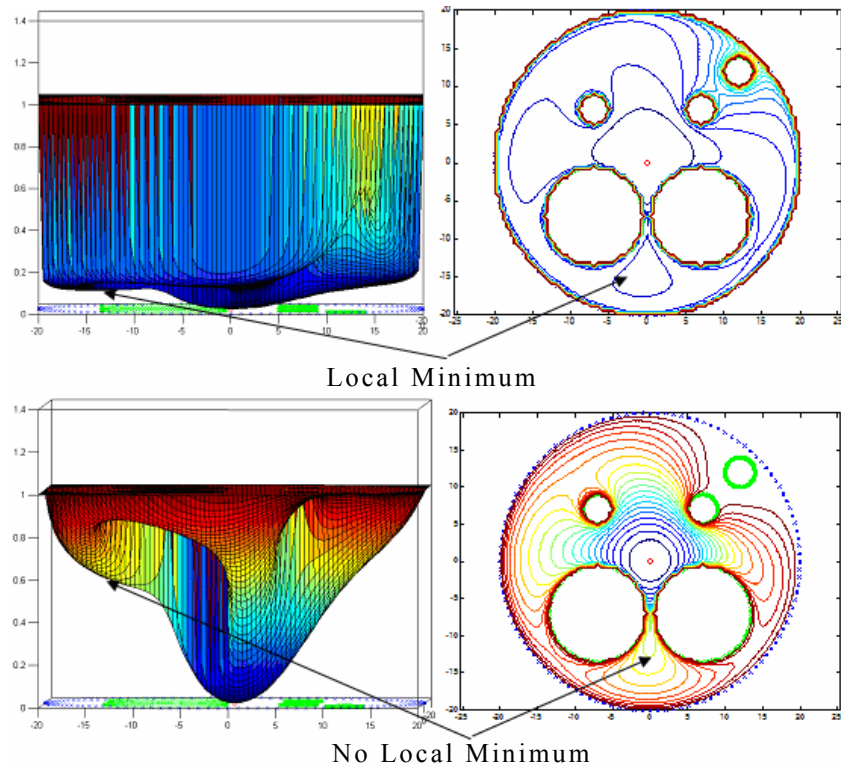


Figure 13: Increasing certain parameters, local minima vanish.

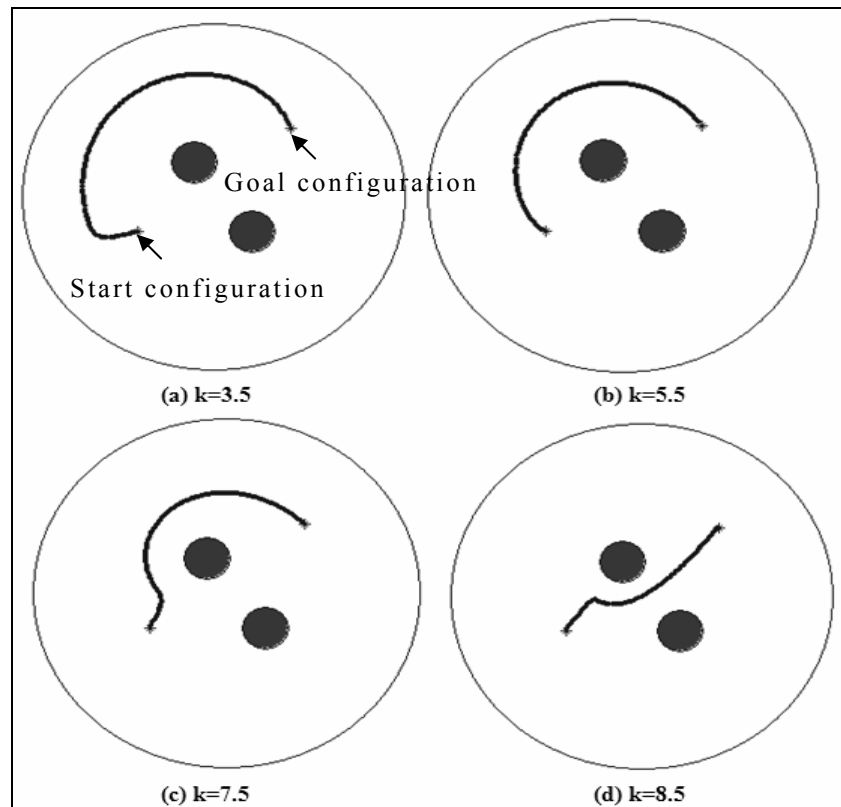


Figure 14: How parameter k affects robot motion.