Integrating Clearance in the Weighted Region Problem

Ioannis Koutsoumpas

stud no: 3914518

Path Planning
Game and Media Technology

-Capita Selecta-

Contents

Integrating Clearance in the Weighted Region Problem

IOANNIS KOUTSOUMPAS, University of Utrecht

1. INTRODUCTION

Various problems of scientific and engineering importance can be related to the general problem of finding the optimal path from a given start position to a goal position. These problems include robot navigation, routing telephone traffic, designing the layout of printed circuit boards and mechanical theorem proving. In this paper, we propose two methods for efficient motion planning in weighted regions considering moving entities represented by disks of arbitrary radius.

Consider a plane subdivided into polygonal regions each of which refers to a weight specifying the cost of traversing this region. The problem of finding a path that minimizes the travel cost according to a weighted Euclidean metric is know as the Weighted Region Problem (WRP) [Mitchell and Papadimitriou 1991]. We describe various WRP path schemes in more detail in Section 2.1.

In order for the paths to be valid and plausible, characters have to maintain a specific distance from the obstacles. Clearance can be interpreted in two ways, as *absolute* or *relative*. Regrading the first category, the character never comes closer than a certain distance to an obstacle. Concerning the latter category, the moving entity stays as far away as possible from the obstacles, while preventing long detours. We adopt absolute distance because we intend to maintain a fixed distance in order to simulate the disk entity motion. In Section 2.2 we present various clearance-based path planning methods.

Until now, the weighted region problem has not been discussed for disk-shaped moving entities. An additional challenge is that these entities need to preserve clearance from obstacles or high-cost regions. On the one hand, solutions for clearance-based path planning consider the optimal path as the route with shortest Euclidean distance. On the other hand, weighted region approaches do not consider clearance information by referring only to point entities. Every region has an associated weight that corresponds to the cost of traversing one unit in that region. Driving through sand is more expensive than driving on a road. Nevertheless, this does not mean that this path is valid since there is the possibility of traversing a passage narrower than the size of the entity. Our methods combine path planning in weighted regions and clearance information for disk-shaped moving entities. A central problem we tackle is how to compute the costs of a disk-traversal between two points in weighted regions. Both approaches avoid grid-based methods to tessellate the environment. Grids do not capture the exact geometry of the terrain and lack accuracy regrading the computed path (see Section 3.1). Our new methods compute paths that can be used as indicative routes (e.g. [Sharir and Schorr 1984], [Kavraki et al. 1996], [Geraerts 2010]) and can be combined with local collision avoidance techniques (e.g. [van den Berg et al. 2008], [Karamouzas et al. 2009], [Jaklin et al. 2013]).

This essay presents two global path planners dealing with clearance information and the weights of the central path ("Indicative route techniques" in Section 4). Next, two additional algorithms are proposed exclusively focusing on the cost estimations of the swiping disk over weighed regions ("Disk cost estimation approaches" in Section 5). The clearance and WRP information from the first and second phase are finally combined in order to generate the final path. The schemes from Section 5 can be independently combined with both global planners as their final step (see Figure 1).

Given a terrain with obstacles (no weighted regions defined yet) and a disk diameter, the first of the indicative route techniques initially transforms narrow paths to obstacles. Then, considering the terrain's weighted triangulation, it searches for safe channels and computes the paths that the indicative route will step on. Regarding

the second indicative route scheme, *Steiner points* [Aleksandrov et al. 1998] are employed and go through various processes. Specifically, they are filtered in order to preserve a specific distance from obstacles while a the path is computed to step on them according to the region's weights.

Disk cost estimation approaches approximate the intersection area of the overlapping swiping disk (closed curve disk traversal) in different ways. The first generates arbitrary distributes points inside the swiping disk and counts how many of them fall in each weighted region. The second method, creates multiple parallel lines inside the swiping disk and measures the lengths of the intersections between these lines and the weighted regions.

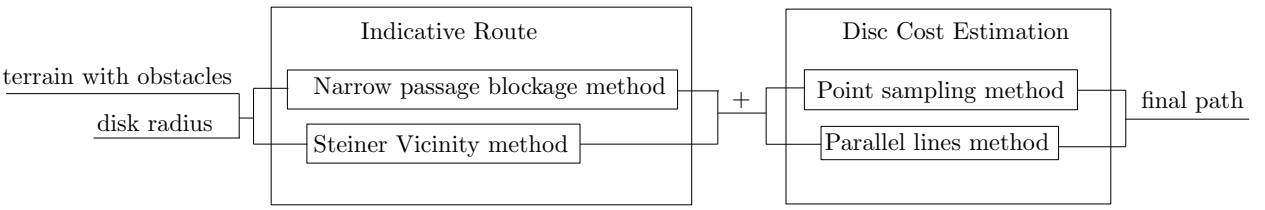

Fig. 1. Diagram representing the steps of producing the final path. Indicative Route schemes can be independently combined with Disc Cost Estimation approaches.

2. RELATED WORK

2.1 Weighted Region problem

Consider a terrain subdivided into non-overlapping polygons. Each polygon is associated with a nonnegative weight (with a possibility of assigning different weights to the boundaries of the polygon). The objective is to find an optimal path (least cost path) from a given start and goal position according to the *weighted Euclidean metric*. The "cost per unit distance" traveled by the agent is homogeneous and isotropic within each polygon. This means that the weight is uniformly distributed in each polygon and it is not affected by the position and velocity vector of the character.

Methods using continuous Dijkstra method [Mitchell and Papadimitriou 1991] have very high complexity and are difficult to implement. Grid-based approaches [Dijkstra 1959] [LaValle 2006] restrict their accuracy to the connectivity of the grid, thus yielding jagged and non-optimal routes. Furthermore, in region graph approaches [Rodriguez et al. 2008] the underlying graph is based on the region adjacency. Since this does not always relate to path optimality, the result can be far from optimal. Other methods [Mata and Mitchell 1997] adopt a graph called *Pathnet*. That is, the system sets angular limits using cones indicating the orientation of the path that is going to be extended from a vertex extension. Then, Dijkstra's algorithm is used to find an optimal path. Unfortunately, this method is inappropriate for spatial datasets because of its high complexity is $O(n^3)$ (where n is the number of vertices) and it is prone to numerical errors.

In our approach we apply edge subdivision method using Steiner points [Aleksandrov et al. 1998] which expose significant balance between performance and plausibility of the resulting path.

2.2 Clearance-based path planning

Relating to clearance-based path planning, existing approaches belong to three major categories: roadmap-based methods, potential fields and cell decomposition techniques. Concerning road-maps, the probabilistic roadmap planner [Amato and Wu 1996] [Barraquand et al. 1996] is created by generating random points in the plane and connecting these points to the k-nearest neighbors. If the connecting edges do not cross any obstacle, the placements are added as nodes to a roadmap graph. Although the method's performance advantages, poor quality paths are generated due to the inherent randomness in the graph representing the free space connectivity. Furthermore, even if one exists, there are cases that a path may never be detected. Moreover, visibility graph [Lozano-Pérez and

Wesley 1979] [de Berg et al. 2008] is a roadmap whose vertices are the vertices of the obstacles themselves and there is an edge between every pair of vertices visible to each other. Integrating arbitrary clearance that reflects practical implementations is very difficult and the performance deteriorates significantly. Finally, Voronoi diagram approach [Hoff et al. 1999] [Geraerts 2010] constructs paths suitable for specific problems and is incapable of dealing with arbitrary clearance. Potential field [Khatib 1985] approach generates a potential field in the free configuration space in which the character is attracted towards the goal position while repulsing from obstacles. This method is prone in local minima (e.g. concave obstacles), deadlocks and some paths can be arbitrary long. In addition, cell decomposition methods [Hart et al. 1968] do not demand complex implementations but due to connectivity limitations in the grid, the path is not always optimal. Moreover, the performance of the method is heavily depended on the complexity of the scene (i.e. grid resolution), especially in large terrains.

During the description of our methods we exclusively rely on triangulated navigational meshes and integrate specific techniques from [Kallmann 2010].

3. PRELIMINARIES

3.1 Triangulations and grid-based methods

In contrast to grid approaches, triangulations are adaptable depending on the input vertex set and more versatile on representing shapes. Apart from weighted region discretization, triangulated meshes also prove very useful regarding navigation strategies for autonomous agents. Triangulations can approximate surfaces through interpolation from many different data sources such as point data, lines, and polygons. Although, there are several triangulation methods available [Watson 1992], we choose Delaunay triangulation (see Section 3.2). In Delaunay triangulations the triangles are as equi-angular as possible and the triangulation is not affected by the order of the points to be considered.

As indicated in the introduction, the proposed approaches are totally free of grid segmentations. Comparing to grid-based interpolation methods, triangulation offers several advantages: First, when the surfaces have significant relief features, triangulation generates more accurate surfaces. Secondly, triangulations are more accurate than grid-based methods because original data points are located exactly on the surface. On the other hand, due to aliasing, grids occasionally respect the original data (points,lines etc). The quality of the path is heavily-dependent on the resolution of the grid. In case the resolution is too low, the quality of the path will not be improved, even after the insertion of additional terrain points. Moreover, vector-based methods which are based on triangulations do not sample. That is, they are more suitable for narrow passage detection since grid-based methods tend to close them during sampling. Furthermore, the grid cells take the weight of the highest pixel they contain. Thus, obstacles in grid-based methods tend to be bigger decreasing the traversable space.

3.2 Constrained Delaunay Triangulation

Let P a set of points in the plane. The Voronoi diagram of P is defined as the subdivision of the plane into n regions, one for each point in P, such that the region of a point $p \in P$ contains all points in the plane for which p is the closest point. The Delaunay triangulation is the dual structure of the Voronoi diagram in $\mathbb{R}^2$. This means that there is a line segment connecting two Voronoi vertices if their Voronoi polygons share an edge. That is, the Delaunay triangulation of P is a collection of edges that satisfy the criterion defined as: for every edge there is a circle containing the edge's endpoints without containing any other points. The Constrained Delaunay triangulation of P and a segment set S is the triangulation that is as close to the Delaunay triangulation of P under the constraint that all segments of S appear as edges of the triangulation. Worth noting that, Delaunay triangulations avoid narrow triangles and are widely used due to their theoretical guarantees as well as their practical performance.Delaunay Triangulations and Constrained Delaunay Triangulations are popular in applications involving data visualization [Treinish 1995], reconstruction [Boissonnat 1988] and mesh generation [Legrand et al. 2000].

3.3 Steiner points

Steiner points [Aleksandrov et al. 1998] introduced an extension on existing triangulation methods. Specifically, in this discretization scheme new points are placed on the edges of the triangulation following a logarithmic distribution. Because this distribution starts from each vertex and expands through the incident edges, it yields an infinite high concentration of Steiner points close to the vertices. For this reason, a circle around each vertex is defined called vertex vicinity which prohibits any Steiner point placement. Considering an edge, there are two expanding distribution of Steiner points starting from both endpoint vertices. In order for these point sets to get merged, authors define the interval as the distance between two consecutive Steiner points. Thus, the sets are merged by discarding the Steiner points with the largest interval for every set of overlapping intervals.

3.4 Uniform point Sampling

Generally, sampling in computers is used to approximate datasets and gather information with adjustable accuracy. Thus, sampling information can comprise a representative and statistically valid sample of the whole. The two major categories of sampling distributions are: sampling from a uniform distribution and sampling from an arbitrary distribution. For the first, if the value v is sampled from a uniform distribution and $[a,b]$ is the distribution set, then the value $y = a + (b-a)v$ follows the uniform distribution parametrized by a and b. Concerning the latter category, arbitrary distributions are also based on uniform distribution. Frequently used arbitrary sampling methods are the *inverse transform sampling* which uses the cumulative distribution function (CDF) and the *rejection sampling*

In this essay we use point sampling in two ways. The first use is the area approximation (see Section 5.1). That is, instead of measuring an area (this also extends to volumes), we sample regions of interest and then count the points that fall in each polygon. In the other technique (see Section 5.2), sampling is used as a tool for parametric curve approximation. In the current problem uniform sampling is employed (since circular arcs are considered) but in curves with varying gradients, adaptive sampling is preferred. Without sapling methods, computers have to store every point p_n on the curve since the Euclidean distance between the stored points $d \to 0 \Rightarrow n \to \infty$ which will require huge amounts of data and plenty of off-line computations.

3.5 Oriented walk

Oriented Walk [Lawson 1977] is a randomized point location algorithm needless of preprocessing. Given a Delaunay Triangulation D over n predefined points and a query point q, the algorithm specifies the simplex D containing q, if it exists. The steps briefly are:

—Randomly select an edge ϵ from D.

—Indicate the triangle d adjacent to ϵ such that d and q are on the same side. Consider the remaining edges of d as ϵ_1 ϵ_2.

—Determine ϵ_i such that the half-plane h_i passing through ϵ_i and doesn't contain d, contains q. If both ϵ_i fulfill the aforementioned condition, choose one randomly. Otherwise, if neither of two ϵ_i have this property, return d as the triangle that contains q.

—Update $\epsilon \leftarrow \epsilon_i$ and repeat 2^{nd} and 4^{th} step.

The advantage of Lawson's Oriented Walk is that it handles geometric degeneracy better in practice compared with the Walkthrough method (in which some edges of D might be collinear with the walking segment). Since it works only for Delaunay Triangulations [Weller 1998] [Devillers et al. 2001] an additional test was included in order to guarantee Constrained Delaunay Triangulation's search convergence. Initially, each visited triangle is marked. Then, if there are two edges that separate the centroid of the current triangle and q in distinct semi planes, the technique choses one leading to a non-marked triangle.

3.6 Local Clearance Triangulation

The use of Constrained Delaunay Triangulation is widely spread in motion planning problems. However, when paths of arbitrary clearance have to be computed, there are additional properties we have to consider. For that reason, Kallmann introduced *Local Clearance Triangulation* [Kallmann 2010].

Before proceeding to the specific algorithm, we provide some necessary definitions. When an agent steps into a triangle, its movement is characterized by the edge that was traversed when entering the shape, the exiting edge and the vertex(corner) connecting these two edges. Let the traversal τ_{abc} of a triangle τ with corners a, b, c, be the movement that relates to the traversal corner b, entrance edge ba and exit edge bc (see Figure 5). Let $\overline{ba}$ indicate the distance between vertices b and a and $dist(b,s)$ the distance between vertex b and line s. Considering a traversal τ_{abc}, clearance $cl(a,b,c)$ represents the circular sector among the entrance and exit edges with radius $r = min\{\overline{ba}, \overline{bc}\}$. There are cases that a constrained edge s exists in the opposite side of ac with respect to b such that $\overline{bs} < min\{\overline{ba}, \overline{bc}\}$. This means that, there is a point b' on s that along with b, define the clearance as $cl(a,b,c) = dist(b,s) = \overline{bb'}$ (see Figure 5).

LCT is characterized by the *local clearance property* which guarantees that only local clearance tests are required during the search for a path based on a given clearance value . In case of local property failure in CDT, iterative refinement operations are performed on the triangulation based on *disturbances criteria*. These operations will convert *CDT* to *LCT* (see Figure 2). The transformation finishes after the execution of all necessary refinements. This happens when local clearance property is preserved for every traversal in the triangulation.

Let v be a vertex connected to c and b a vertex connected to a and c on the same side of ac. Let s be a constrained edge such that s is either ac (if ac is constrained) or s and v are on opposite sides in respect to ac (see Figure 4). Vertex v is a disturbance to traversal τ_{abc} if:

—v can be orthogonally projected on ac.

—v is not shared by two collinear constraints.

—$dist(v,s) = cl(a,b,c)$

—$dist(v,s) = dist(v,c)$

Refinement process includes the subdivision of a constrained edge according to the refinement point p_{ref}. Point p_{ref} is located in a way that guarantees that both the corner vertex b of the traversal and the disturbance vertex v will become connected to p_{ref} by edges after the refinement. Specifically, x_1 and x_2 are the points of intersection among constrained edge s and the circle passing by u, v and c. Vertex u is the next vertex around c, when rotating from v to b. Thus, p_{ref} is considered as the midpoint between x_1 and x_2 (see Figure 4).

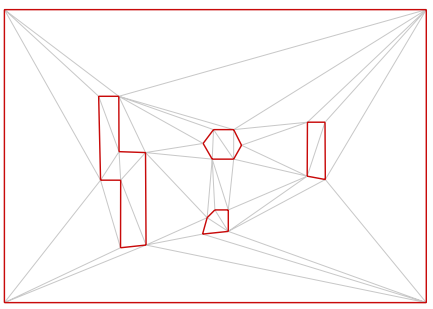

Fig. 2. Constrained Delaunay Triangulation.

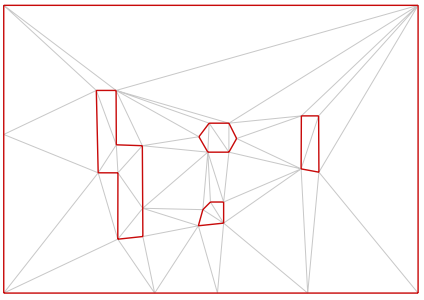

Fig. 3. Local Clearance Triangulation after refinements.

Having defined disturbances, we can describe LCT based on these two properties: a) A traversal τ_{abc} in a CDT has local clearance if it does not have disturbances. b) A LCT is a CDT triangulation with all traversals having local clearance.

A free path between a start and end point will cross several triangles sharing unconstrained edges. The union of all traversed triangles is called a *channel*. Given distance r, LCT comprises the base for computing the optimal path across consecutive triangles. A path is called *locally optimal* when firstly, distance r is maintained from constrained edges across its length and secondly, has the smallest length considering the specific channel. If a shorter path with clearance r across the remaining channels of the terrain doesn't exist, the current path is called *globally optimal*.

In LCT, two clearance values are precomputed and stored for each edge, depending on current entry and exit edge. Specifically, in Figure 5 the two clearance values stored in edge cb are $cl(a,b,c)$ and $cl(g,c,b)$. The two values stored at the edge bc, are the clearances of the traversals (circular arrows in Figure 5) having a mutual edge as exit edge. In our case, τ_{abc} and τ_{gcb} are the traversals and bc is the mutual exit edge that the two clearances are going to be stored. According to *local clearance property*, if two clearance values for each edge are known, one can determine if a disk character fits to pass through a passage.

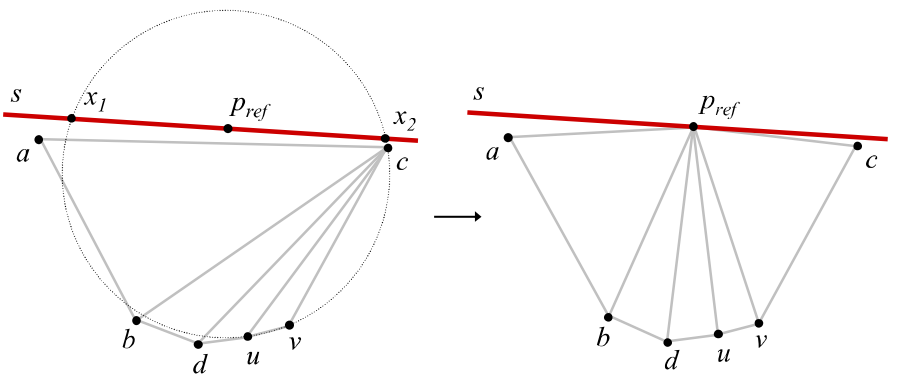

Fig. 4. Point p_{ref} is located in a way that guarantees that both the corner of the traversal and the disturbance vertex will become connected to p_{ref} by edges after the refinement. Points x_1 and x_2 define the intersection among constrained edge s and the circle passing by u, v and c.

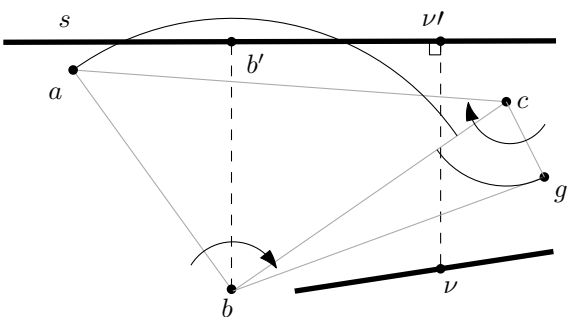

Fig. 5. In this example $cl(a,b,c) = dist(b,s) = \overline{bb'}$. Since $\overline{bb'} < 2r_d$, according to the proposed method (see Section 4.1.1) bb' will be considered as a constrained edge. Circular arrows indicate the two traversal τ_{abc} and τ_{gcb}. Both traversals have bc as an exit edge.

4. INDICATIVE ROUTE TECHNIQUES

The following two algorithms comprise the initial parts of the global planner since they are dealing with clearance information. They define the space according to the disk radius and secure the channels in order to keep the appropriate distance from obstacles and avoid narrow passages. Thus, the character avoids deadlocks while the whole process becomes less complex through the pruning of the search space. This section does not define a complete global planner since the presented algorithms have to be combined with one of the two swiping-cost estimation approaches (see Section 5) in order to calculate the indicative route.

4.1 Narrow Passage Blockage Method

This technique starts by detecting narrow passages on a terrain with obstacles (no weighted regions defined yet). Given a disk diameter, our algorithm uses *Local Clearance Triangulation* technique (see Section 3.6) in order to transform narrow paths to obstacles. The next step receives as an input only the terrain with the updated non-traversable regions (no LCT triangulation exists anymore) and triangulates it according to the weighted regions' vertices (CDT). Finally, the channel search follows, marking the triangles (and specifically the traversable part of every unconstrained edge) that the indicative route will step on. Figure 6 presents the different stages of the current method.

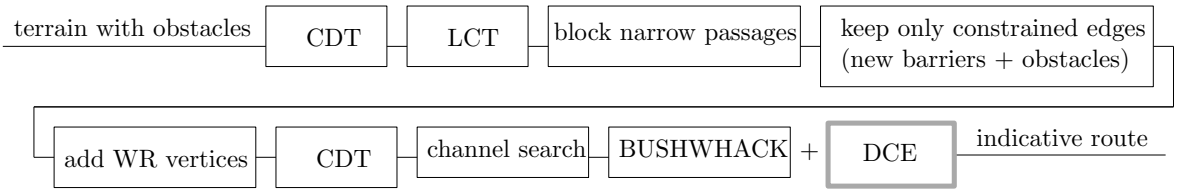

Fig. 6. Depiction of the stages regarding the first indicative route technique (see Section 4.1). *WR* vertices stands for *Weighted Regions'* vertices. Gray orthogonal represents one of the two disk cost estimation schemes (see Section 5) that contribute in the final path estimation.

4.1.1 *Block narrow passages.* The current section mainly contributes to the improvement of the performance of the subsequent channel search algorithm (see Section 4.1.2). Initially, the input is comprised only by the obstacles (no vertices of weighted regions). Having already computed the constrained Delaunay triangulation according to the constrained edges' vertices, we transform it to *Local Clearance Triangulation* [Kallmann 2010] and propose several extensions.

The modification we propose takes place after the clearance calculation step. Kallmann's technique precomputes clearance values for every edge of the LCT in order to prepare the terrain for arbitrary clearance tests (see Section 3.6). This is because *LCT* presents *local clearance property* and can guarantee that only local clearance tests are required during the narrow passage detection algorithms. We take advantage of this step and use it to our needs. After all LCT refinements finish, we result to a triangulation with stored clearance values on every edge. Worth noting that, it is not possible to measure clearances on-the-fly (i.e. between refinements) because edges might be added or erased in the next refinement step. That is, it is not possible to store clearances in every edge without first entirely converting *CDT* to *LCT*.

Technique: Given the disk radius, we insert a constrained edge in the position of bb' wherever $cl_e < 2r_d$ or $cl'_e < 2r_d$ (see Figure 5), where e is the currently checked edge and cl_e, cl'_e are the two stored clearance values (see Section 3.6). Taking in consideration that $bs < 2r_d$, if the distance from the traversal corner b to the constrained edge s is smaller than both incident to b edges ba and bc, a vertical to s barrier is created starting from b. The barrier is established exactly where clearance is measured regardless where is stored. A naive solution would be to transom the edge from which we read the clearance values into a constrained edge. However, this is a false approach since clearance is not exactly measured where the edge lies (e.g. in Figure 5 clearance is measured on bb'

but is stored in *bc*). That is, *bc* is just a medium where information is stored. In other words, clearance stored on *bc* does not indicate that the disk cannot step on *bc*. It informs the planner that the disk doesn't fit to cross the triangle, thus there is no need to step on *bc*. Hence, if we transformed a traversable edge of LCT into a constrained edge, we would deprive traversable space from the next phase. This is because the next phase will involve only constrained edges and weighted region triangulation (CDT on weighted region's vertices) and no LCT will be present anymore.

Remarks on technique: Finally, every narrow passage in the scene is blocked, thus it is no longer necessary to consider disk-fitting problems during the following steps of the path planning solution. Since *bs* is the minimum distance between two constrains, the method informs the graph search that there is no reason to search beyond this passage because is guaranteed that the character doesn't fit. In certain occasions, this step can save many unnecessary calculations or even prevent the path planner form starting (e.g. when having an enormous disk comparing to the passages). Having blocked unsuitable paths, the current concern is to keep clearance distance $d \geq r_d$ from the obstacles (see Section 4.1.2). The next step (see Section 4.1.2), receives only the constrained edges of the scene (newly inserted and preexisting obstacles) and discard any other edges remained from LCT.

General remarks: Worth noting that, on his method, Kallmann used the stored clearances in every edge along with a graph search algorithm in order to find suitable channels for a given radius. Our case is much different since we expect to use weighted triangulation accompanied by the reformed constrained edges of the terrain in the next step. Common pitfalls could be the blockage of larger passages or ignorance of narrow paths. Since Kallmann's technique is robust until this point, our extension guarantees its results. The barrier is created exactly where the clearance is measured and thus there are no logical gaps in the transformation of this methodology. Furthermore, worth noting that in LCT, all edges have their extremes on constrained vertices (see Figure 5). This means that every time we insert a new barrier, a path blockage will be established since both extrema of the barrier will lie on constrained edges.

4.1.2 *Channel search.* This phase receives as input the terrain including the initial obstacles as well as the new barriers. Then, it constructs the CDT of the terrain considering the vertices that define the weighted regions. Next, we use Hierarchical A* [Holte et al. 1996] for weighted graphs to search the graph for appropriate channels according to a cost metric. During channel search we initially ensure that the disk will be placed on valid reference points on the traversable edges in terms of clearance. Then, an additional visibility test is employed ensuring that the disk traversal from one edge to another (regarding the two side segments of the swiping-disk curve) will not have any disturbances from obstacles.

Cost metric: We now have to transform the triangulation into a graph. This means that we have different choices for picking a point in a triangle that will represent a vertex in our graph. Hence, we define the cost metric that is used during the graph search expansion. During channel search, most techniques consider the centroid of each triangle as the reference point. Using the centroid of the triangle, the traversal cost accounts for the length of the line segment lying in two different triangles. This may result to unnecessary zigzags concluding to a false path. For that reason, we use the midpoint of each traversed edge as our graph's vertex. This reduces the disparity thus increasing the possibilities to select the optimal channel.

Maintaining distance: In Section 4.1.1 we used the diameter of the disk in order to block narrow passages and save computational time for the current step. Nevertheless, this does not guarantee that a disk traversing an unconstrained edge will not intersect with a neighboring obstacle (see Figure 8). Here, we describe how we preserve distance equal or greater to disk radius between reference points and constrained edges. For every edge incident to a constrained edge, we ensure that each taken point (disk center) on the traversed edge has distance $d \geq r_d$ from any neighboring constrained edge (obstacle or terrain contour). This means that we have to translate the point on the edge accordingly if needed. That is, we employ the technique in Section 4.2.3 depicted in Figure 11. The scheme discards candidate Steiner points according to the distance from the adjacent constrained edge. In our case, we use this in order to indicate which part of the traversable edge is valid for the disk to step on.

Avoid disturbances: Here we describe the way we guarantee collision-free disk traversals. The method in the previous paragraph only ensures that the disc can be placed on unconstrained reference points in the graph. However, there are traversals from one vertex to another within a channel that might not be collision-free (see Figure 7). Hence, a visibility test is introduced in order to keep the traversal (i.e. side line segments of the swiping disk curve) out of disturbances. Let ε be the outer tangent connecting the start and final disk with centers k and c respectively. The visibility test checks if ε is disturbed by a constrained edge. That is, in case ε intersects with a constrained edge, the traversal is not valid. One can argue that the visibility test could be enough to ensure clearance without the initial distance tests (see previous paragraph). This is not true since this visibility scheme checks for disturbances only for the side edges of the swiping-disk curve. This does not guarantee that the front semicircular part of the curve will not intersect with an obstacle on its final destination. Next paragraph describes the visibility test in more detail.

Outer tangent: A more detailed application of the visibility test is depicted in Figure 8. Considering k as a fixed point (starting point of the traversal), we show how c is translated across the traversable edge ab according to the outer tangent line segment pe. Having changed the position and slope of pe, $p'a$ shows its new location. Given a line of the form $Ax + By + C = 0$ and a point (h,k), the perpendicular distance from the line to the point is represented by $|Ah + Bk + C|/\sqrt{(A^2 + B^2)}$. Next, consider two circles centered on (k_x, k_y) and (c_x, c_y) with equal radii r. The two outer tangents of the form $Ax + By + C = 0$ and $A'x + B'y + C' = 0$ are represented as [Casey 1882]:

$$A = A' = ((k_y - c_y)\sqrt{(k_x - c_x)^2 + (k_y - c_y)^2})/((k_x - c_x)^2 + (k_y - c_y)^2) \tag{1}$$

$$B = B' = ((k_x - c_x)\sqrt{(k_x - c_x)^2 + (k_y - c_y)^2})/((k_x - c_x)^2 + (k_y - c_y)^2) \tag{2}$$

Since the distance from the outer tangent to the circle center (k_x, k_y) is equal to radius r, we have:

$$C = -r\sqrt{A^2 + B^2} - Ak_x - Bk_y \tag{3}$$

$$C' = r\sqrt{A^2 + B^2} - Ak_x - Bk_y \tag{4}$$

In other words, we search for all the values of (c_x, c_y) across ab that do not give a solution to the equations system of line segments between the outer tangent and the constrained edge. These values represent all the valid points that the center c of the destination disk can be placed.

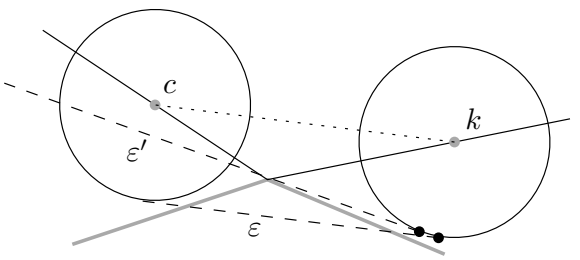

Fig. 7. In this example, we depict a disk traversal from k to c. Point c preserves the appropriate clearance from the obstacle but the traversal is false. Hence, outer tangent visibility test is employed to ensure that the tangent ε doe not intersect with the obstacle.

Edge rejection: We completely ignore an edge as a traversable edge in cases that the translation of the destination disk center outreaches the length of the current edge. By respecting this condition, we ensure that the computed indicative route retains the appropriate clearance. Since it is proven that the midpoint as reference point is an efficient approach during channel search [Kallmann 2010], we propose the translation technique as an extension

in order to optimize this phase. That is, in case a midpoint didn't fit, it would be very greedy to discard the entire edge. Thus, we prefer to investigate for the nearest best solution while searching in the unobstructed direction of the edge.

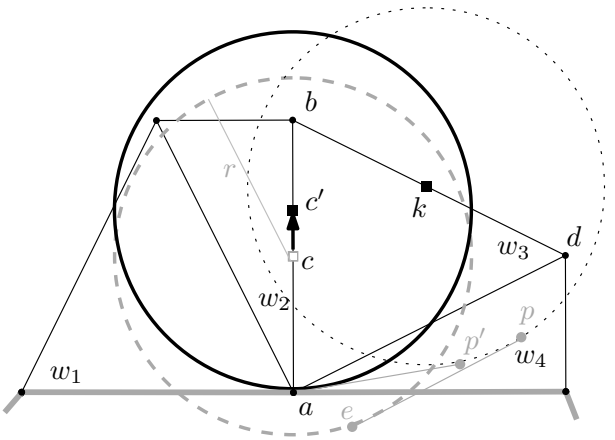

Fig. 8. Application of midpoint translation technique. This guarantees collision-free reference points and traversal in the graph. Gray edges are constrained(preexisting obstacle or added barrier) whereas black are traversable (weighted region). Point k is the center of the initial and c the final candidate disk position. Regarding *maintaining distance* rule, the midpoint(and circle center) c of edge ab is selected during channel search. Since $ac \leq r$, c gets translated to c'. Concerning *avoid disturbance* rule, outer tangent pe intersects with an obstacle, thus the disk center needs to be translated. If c' outreaches ab, the edge is denoted as non-traversable.

Graph search: Now, we have defined the points on which our graph will be constructed, in order to start the search for the channels. We apply Hierarchical A* because it exposes increased speed and lower memory usage comparing to similar method, for instance, A*. This occurs mainly because it requires a smaller search space due to the use of abstraction layers approach. After Hierarchical A* finishes, we mark the triangles that the resulting path traverses as the channel C_r. Worth noting that during the execution of the channel search, if one edge is shared between two faces with different weights, we consider the smaller weight as the weight of the edge. Moreover, oriented walk [Lawson 1977] is employed in order to locate the triangle that contains our starting point.

BUSHWHACK preparation: BUSHWHACK [Sun and Reif 2001] algorithm plays a major role in the following step which involves indicative route estimation. Regardless edge midpoint translation, as we explain in the following paragraph, BUSHWHACK will be applied in the entire marked triangle. This could result in false decisions because it may consider points on edges that should have been rejected. In other words, BUSHWHACK's interval will extend beyond the allowed point on an edge. Hence, a way has to be proposed in order to preserve the clearance information we already acquired. Thus, every time a reference midpoint gets translated, the edge is marked as "translated" along with one of the two directions. For instance, in Figure 8, edge ab is marked as "translated" with direction towards b. In case BUSHWHACK detects an unmarked edge (e.g. bd), it will function according to its default manner. On the other hand, if the edge is marked, BUSHWHACK's interval would start from the translated reference point to the unconstrained direction.

Remarks: The way we choose to eliminate inappropriate paths is to create barriers. Furthermore, triangulation construction during current phase is avoided inside closed eliminated regions thus decreasing computation time. In addition, another improvement is that, in several occasions these barriers will exclude multiple regions that otherwise would be considered by the graph search (explained below).

4.1.3 *Indicative route from channels.* Until now, we have ensured that the disk traversals on inside the channels retain the appropriate distance from obstacles during channel selection (see Section 4.1.2). In this final phase, we use BUSHWHACK [Sun and Reif 2001] (which creates the Steiner points during its execution) along with the methods in Section 5 to construct our indicative route through the marked triangles. BUSHWHACK contributes to the final cost estimation combined by the cost of the moving center of the disk and the cost of the swiping-disk area. It is a fact that the monotonic property of the *intervals* appearing in BUSHWHACK (BUSHWHACK was proposed based on point entities) may conflict with the disk entity since the character steps on multiple triangles at each step. Monotonic property could affect negatively the result in case we initially constructed the path on BUSHWHACK's output (disk center movement) and then execute disk area cost schemes. However, since the final path cost is computed considering the center and disk area cost at the same time, BUSHWHACK's monotonic property contributes and doesn't hinders the method.

4.2 Steiner Vicinity Method

This solution is comprised by two main phases. Initially, the terrain including the obstacles and the predefined weighted region gets triangulated. In contrast with the previous approach (see Section 4.1), graph pruning starts based on Steiner points disapproval. Point rejection phase is comprised by the *Steiner vicinity rejection* and *Steiner points filtering* schemes. As a result, the search space will shrink and be prepared for the indicative route computation. The channel search takes place in a more detailed level, following certain rules we are going to describe below. Figure 9 presents the different stages of the current method.

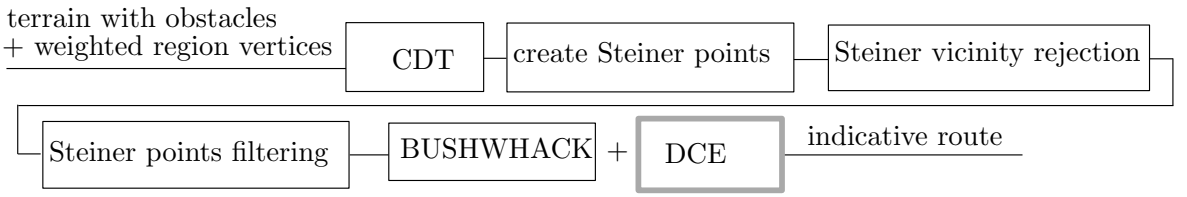

Fig. 9. Depiction of the stages regarding the second indicative route technique (see section 4.2). Gray orthogonal represents one of the two disk cost estimation schemes (see Section 5) that contribute in the final path estimation.

4.2.1 *Create triangulation.* Constrained Delaunay triangulation is used considering the vertices that define the weighted regions and the constrained edges of the terrain.

4.2.2 *Steiner vicinity rejection.* Initially, oriented walk point location algorithm [Lawson 1977] is employed in order to locate the triangle that contains the starting point. Next, we tweak the Steiner vicinity around every vertex located in the endpoints of every constrained edge. That is, now the vicinity is equal to the disk radius r_d in order to prevent other Steiner points to get created closer. Thus, if a disk nearly fits to pass, there will be only one Steiner point on the middle of the edge. On the other hand, if the two radii overlap, this means that the disk doesn't fit, therefore no Steiner points will exist there.

4.2.3 *Steiner points filtering.* Since the aforementioned rule doesn't cover all cases (some triangles may be very oblong) we define a second additional criterion (see Figure 11). The first rule possibly leaves some undesirable traversable (Steiner) points that case the disk to intersect with a neighboring constrained edge. Referring to Figure 11, let θ be the angle between a constrained and an unconstrained edge ($\theta < {}^{\pi}/_{2}$ due to Delaunay triangulation properties). Next, let l be the length of the segment defined as the distance from the constrained vertex to the candidate point (situated on an unconstrained edge). Considering d as the distance of the point to the opposite

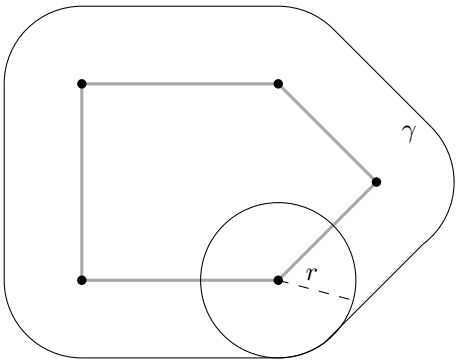

Fig. 10. Offset polygon between a polygonal obstacle and a disk with radius r.

constrained edge, it holds that $d = r_d = lsin(\theta)$. Thus, we have:

$$0 < \theta < {}^{\pi}/_{2} \Rightarrow sin(\theta) > 0 \quad thus \quad l \geq \frac{r_d}{sin(\theta)} \tag{5}$$

Hence, for edges incident to constrained vertices we have to build Steiner points from the length l and above.

Generally, we could ignore the vicinity technique and use only the distance one. However, it is preferred to conduct an initial massive filtering to candidate Steiner points. This is because it allows us to decide about multiple Steiner points in one step instead of checking if $d \geq r_d$ for each one.

4.2.4 *Indicative route.* Then, BUSHWHACK along with the two disk cost approaches (see Section 5) completes the indicative route generation process. Worth to note that in this solution It is preferable to define and filter the Steiner points prior to BUSHWHACK search, if possible. We mention this because BUSHWHACK is designed to create Steiner points during its execution.

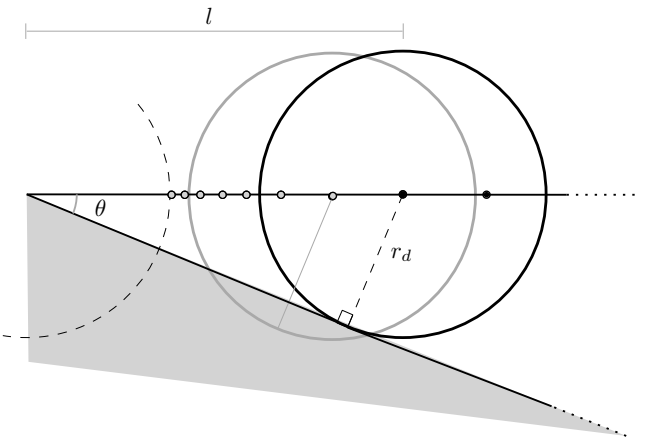

Fig. 11. Steiner points filtering. Filled dots are the valid Steiner points.

4.2.5 *Remarks.* In this section we prove our method's correctness using the definition of *offset polygon* (see Figure 10). Offset polygon is defined as:

Given a set $A \subseteq \mathbb{R}$ the r-offset is a super-set A : offset$(A,r) = \{p \in \mathbb{R}^2 | d(p,A) \leq r\} = A \oplus D_r$ with Minkowski sum $A \oplus D_r = \{a+d | a \in A, d \in D_r\}$ and disk $D_r = \{p \in \mathbb{R}^2 | d(\vec{O},p) \leq r\}$.

Offset polygon's curve maintains distance equal to r in each of its points in a continuous way. That is, it defines a closed curve around an obstacle. Concerning the obstacle, it is not necessary to be a closed shape; it can be, for instance, a constrained edge. The method we propose eliminates Steiner points for distance smaller than r similarly to what the use of the offset polygon would eliminate. Our method is more efficient because it functions in a discrete way. Instead of checking if a candidate Steiner point lies inside the offset polygon, it measures the distance from the closest constrained edge. At this point, worth noting that this applies after constraining the regions inside the circles with radius r centered on every constrained edges' intersection.

In the presence of two intersecting offset polygons, no Steiner point will be created inside the polygons' union. Furthermore, if two offset polygons abut (for example, on a single point) and there is a traversable edge passing through this point, a Steiner point will be constructed. In both these special occasions, the proposed scheme behaves in the exact same way.

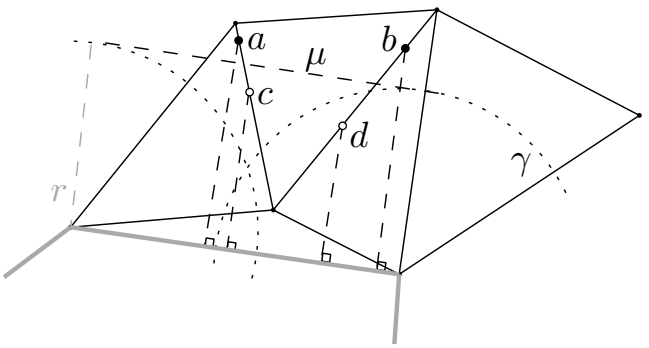

Fig. 12. Example of constrained Delaunauy triangulation representing the resemblance between offset polygon (see Figure 10) and the proposed method (see Section 4.2). Example Steiner points a and b are not discarded. Point c is discarded because is located below the minimum distance border μ. Point d lies inside the constrained vertex radius and thus is not valid.

Using an alternative mathematical description of the offset polygon, here we provide an intuitive description of our method. A polygonal obstacle is comprised by multiple consecutive line segments. Let a line segment of the obstacle described by the line equation $ax+by+c=0$. The locus of all points (x,y) lying at distance greater or equal to r from the line $ax+by+c=0$ can be described by the *conic curve* equation as:

$$\frac{(ax+by+c)^2}{a^2+b^2} \geq r^2 \tag{6}$$

Since this equation represents the offset polygon restriction concerning candidate Steiner points, it can also precisely describe the combination of restrictions of the two basic elimination rules (constrained vertex radius and distance from constrained edge) of our scheme.

In Figure 12 we demonstrate the resemblance between offset polygon method and the proposed technique. Specifically, we depict the constrained Delaunauy triangulation involving part of an obstacle along with vertices that define the weighted regions. The line segment(dotted) connecting the two circles defines the border between Steiner point validity. It symbolizes the maximum distance of the candidate Steiner point to the closest constrained edge that we measure in our technique. That is, the points a and b are valid because the distance from closest obstacle is larger than r. On the other hand, points c and d are not valid because the former is located below the aforementioned border and the latter lies inside the constrained vertex radius and will be eliminated in the first pass. Finally, we observe how the combination of the elimination circles along with the rule referring to the

minimum distance from the closest obstacle, define the offset polygon curve γ (see Figure 10, Figure 12). This means that the proposed technique guarantees collision-free movement of the disc through the entire scene.

5. DISK COST ESTIMATION APPROACHES

During cost estimation, previous weighted region problem approaches consider non-dimensional moving entities. In our case, besides maintaining a minimum clearance from obstacles, we have to modify the input cost of the planner according to disk properties. This happens because disk translation is characterized by traversing weighted areas and not only weighted line segments (see Figure 13). Thus, the general idea is to integrate the weighted area produced by the intersection of the swiping disk and the weighted terrain into the existing cost function (see Figure 14). Every time BUSHWHACK discovers a new node on the graph, the adjacent disk cost translation will be added (see Section 4.1.3 and 4.2.4). Calculating the intersecting area between triangles and shapes including arcs is a complex procedure, thus we propose two simplified and reliable methods. The first approach is based on point sampling on the included weighted regions. The second incorporates weighted line segments resulting from multiple line-terrain intersections.

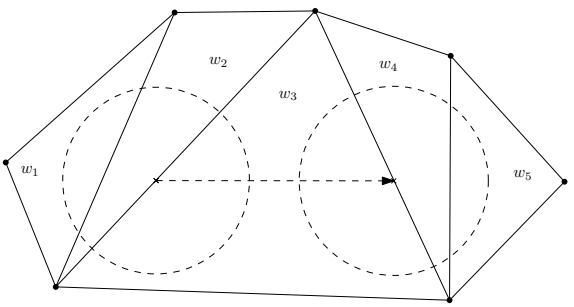

Fig. 13. Disk traversing weighted regions.

5.1 Point sampling method

This method is comprised by seven steps. It starts with the curve (capsule cross-cut) approximation to a polygonal shape. After the triangulation of the polygon, we apply uniform point sampling. Firstly, we randomly pick a triangle and then generate the vertices inside it. Until now, there was no involvement of the weighed terrain tessellation. Having stored the coordinates of the randomly generated vertices, we then count how many fall in each triangle of the weighted region triangulation. Given the number k of the points to generate, the idea is to iterate k times the step: choose a triangle randomly in which a random point will be generated.

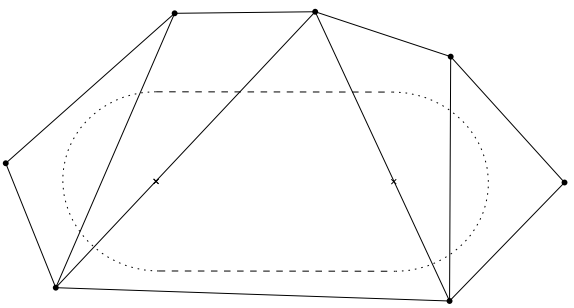

Fig. 14. Area produced by swiping disk.

We decided to approximate the input shape to a polygon for several reasons. Triangles -and generally shapes defined by line segments- can be processed more easily by computers. Moreover, triangle combinations can produce any (triangulated) shape (e.g. polygon mesh). Concerning random point sampling inside closed curvatures, polygonal approximation is also preferred. Presenting an approach based on mathematical theorems is one part of reaching your goal to the final problem solution. Computers cannot represent irrational numbers. This means that, computations involving "pure" curves or circles are very computationally demanding. Since shapes including precise arcs cannot exist in digital world, we have to approximate the shape to a polygon. Generally speaking, an ultra-high resolution polygon is (technically) a curve. In conclusion, whenever hardware is involved to an experiment, controlling the ratio between performance and calculations' precision is of great importance.

5.1.1 *Curve approximation.* The curve formed by the swiping disk involves two semicircles connected by two line segments. To wit, the parts needing approximation are the two semicircles. Because semicircles have constant gradient at each point, we use uniform point sampling. That is, after connecting the points generated on the curves, we have a polygonal approximation of the shape (see Figure 15). Let $\gamma : [0,1] \rightarrow \mathbb{R}^2$ describe a curve in 2-dimensional space. In order to approximate γ, choose n points with equal distance described as $0 = t_1 < t_2 < ... < t_n$, which define the vertices $u_0, u_1, ..., u_n$ where $u_i = \gamma(t_i)$. The goal is to choose sample points $t_1 ... t_n$ sufficiently describing the curve while keeping n small. Depending on the preferred balance between accuracy and efficiency, different number of sampling points can be generated. The most common technique is to use trial and error through automated heuristic tests [Lindgren et al. 1992].

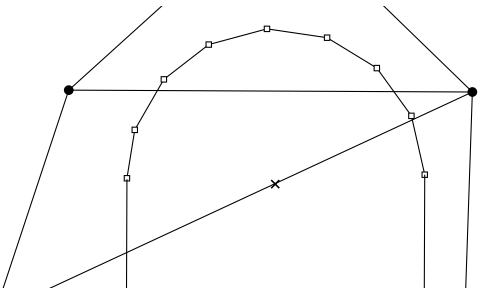

Fig. 15. Curve approximation with $n = 9$ for each semicircle.

5.1.2 *Polygon triangulation.* Having the polygonal estimation of the curve, the next step aims to a fast convex (and consequently, simple) polygon triangulation. Hence, a suitable algorithm of $O(n)$ complexity is the one proposed in [Chazelle 1990]. Although, due to its complex implementation, a simpler randomized algorithm can be used which also guarantees linear complexity described in [Amato et al. 2000]. Thus, given the polygonal approximated curve γ_{approx} this step produces the triangulation γ_{tr} (see Figure 16).

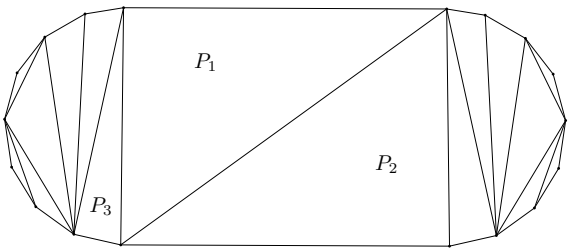

Fig. 16. Triangulation of approximated swiping area.

5.1.3 *Triangle picking.* At this phase, we will use finite distributions in order to pick a triangle from γ_{tr} randomly according to their area ratio. Discrete or finite distribution, is the distribution whose variables can take on only discrete values. A discrete distribution with probability function $P(x_k)$ defined over k has distribution function:

$$D(x_n) = \sum_{k=1}^{n} P(x_k) \tag{7}$$

We assign an index $k \in \mathbb{N}^0$ to each triangle (see Figure 16) in γ_{tr} and assign a weight as follows:

$$P(x_k) = \frac{area_k}{\sum_j^n area_j} \tag{8}$$

where n is the total number of triangles in γ_{tr}. We then generate a random index k from the discrete distribution over indexes given their weights by:

$$P(x_k) = \begin{cases} \frac{area_0}{\sum_j^n area_j} & \text{when } k = 0 \\ \frac{area_1}{\sum_j^n area_j} & \text{when } k = 1 \\ \dots & \\ \frac{area_n}{\sum_j^n area_j} & \text{when } k = n \end{cases}$$

The selected index indicates the chosen triangle.

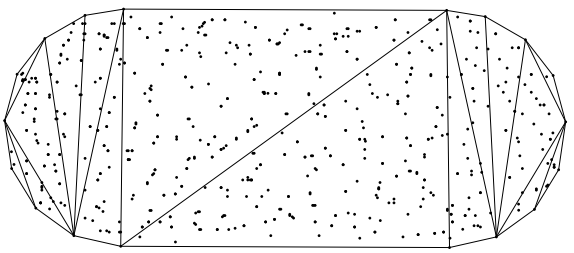

Fig. 17. Sampled points inside the approximated curve.

5.1.4 *Random points generation.* During this step we will treat each triangle independently. This means that the method is able to uniformly generate random vertices at any kind of triangulated shape such that $\gamma_{tr} \to \mathbb{R}^2$. Initially, we uniformly generate two random numbers a_0, a_1 from the interval $[0,1]$. The uniform density function on the interval $[a,b]$ is the constant function $f(n) = \frac{1}{b-a} \Rightarrow f(n) = 1$. As digital computers make calculations deterministically, a common technique for generating pseudo-random numbers is to employ linear congruential generators or quasi-random numbers [Gentle 2003] [Seydel 2012]. Consider the vertices v_0, v_1, v_2 as the three vertices of a triangle of the polygon triangulation so that $v_i = (x_i, y_i)$. Then, we calculate the random point p_{rnd} by:

$$p_{rnd} = a_0(v_1 - v_0) + a_1(v_2 - v_0) \tag{9}$$

The computation of the random vertex in (9) implies that there is a probability equal to 0.5 for p_{rnd} to lie outside the triangle. Nevertheless, this area is not unknown and is defined by another known triangle (see Figure 18). To wit, the union of the two triangles form a parallelogram $\rho_{v_0 v_1 v_2 v_3}$ which can be described as the merging of the

triangle $\tau_{v_0v_1v_2}$ with its reflection $\tau'_{v_1v_2v_3}$ with respect to the axis δ defined by v_1, v_2 . Thus, if the point lies outside $\tau_{v_0v_1v_2}$ we transform it using:

$$p'_{rnd} = u_0 + R_{\delta,\pi}(p_{rnd} - u_3) \tag{10}$$

in order to lie inside $\tau_{v_0v_1v_2}$ (where $R_{\alpha,\theta} \in \mathbb{R}^2$ is the rotation matrix, where α is the rotation axis and θ the rotation angle). image

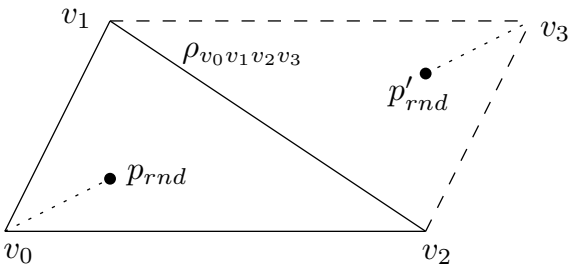

Fig. 18. Picked triangle $\tau_{v_0v_1v_2}$ with its reflection $\tau'_{v_1v_2v_3}$.

5.1.5 *Transition from polygon to weighted regions.* In this step, we transfer the problem from the polygon region (composed by merged triangles) to the weighted terrain triangulation. To wit, after the random vertex generation the only data useful for the rest of this method is the coordinates of each point. Then, in order to prepare the data for the next step, we transform these coordinates from the polygon coordinate system to the weighted region coordinate system.

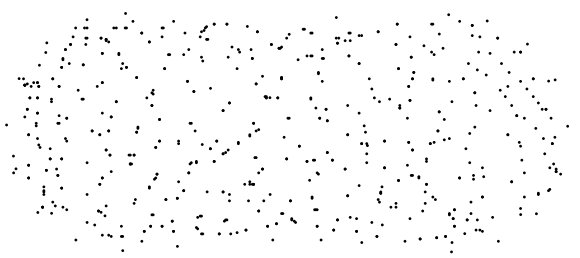

Fig. 19. Sampling points without the approximated curve.

5.1.6 *Points counting.* During this phase, the system counts the number of points that fall in each triangle of the weighted terrain triangulation. For the counting, we use *barycentric coordinates* method which is very fast especially for a fixed triangle with many test points (see Figure 20). This is because we only have to iterate part of the algorithm computations every time we want to check another point. *Barycentric coordinates* are widely used in computer graphics (ray-plane intersection [Shirley and Marschner 2009]). For the current problem, consider triangle $\tau_{v_0v_1v_2}$ and a point $p \in \mathbb{R}^2$. We first compute the vectors:

$$n_0 = v_2 - v_0 \tag{11}$$

$$n_1 = v_1 - v_0 \tag{12}$$

$$n_2 = p - v_0 \tag{13}$$

We then compute the dot products:

$$d_{00} = n_0 \cdot n_0 \tag{14}$$

$$d_{01} = n_0 \cdot n_1 \tag{15}$$

$$d_{02} = n_0 \cdot n_2 \tag{16}$$

$$d_{11} = n_1 \cdot n_1 \tag{17}$$

$$d_{12} = n_1 \cdot n_2 \tag{18}$$

Calculate the inverse denominator

$$D_{inv} = \frac{1}{d_{00}d_{11} - d_{01}d_{01}} \tag{19}$$

and the barycentric coordinates:

$$b = (d_{11}d_{02} - d_{01}d_{12})D_{inv} \tag{20}$$

$$b' = (d_{00}d_{12} - d_{01}d_{02})D_{inv} \tag{21}$$

Finally, the point lies inside the triangle $\tau_{v_0v_1v_2}$ if:

$$b \geq 0, \quad b' \geq 0, \quad b + b' < 1$$

In the current example, barycentric coordinates are computed with respect to v_0 but v_1 or v_2 would work as well. To test additional points you the algorithm only needs to recompute n_2, d_{02}, d_{12}, b(20) and b'(21). Quantities such as D_{inv} remain the same for each triangle.

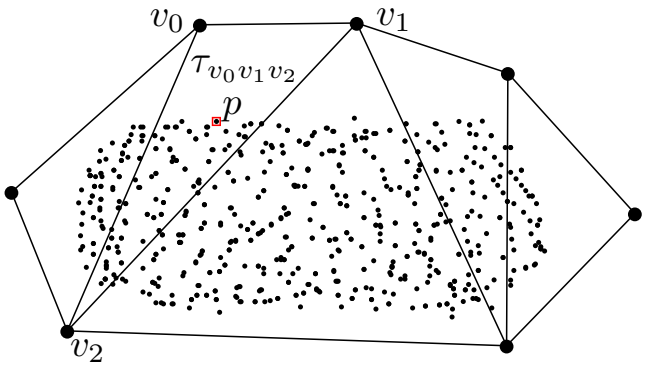

Fig. 20. Checking if point p lies inside triangle $\tau_{v_0v_1v_2}$.

5.1.7 *Swipe cost.* In this final phase the algorithm multiplies the number of points computed in the previous step with the adjacent region weights in order to compute the total cost of the point sampling method. Hence, the total cost of every swipe C_{swipe} can be written as:

$$C_{swipe} = \sum_{i=1}^{K} \sum_{j=1}^{N_i} w_i \tag{22}$$

where N is number of points in each weighted region triangle and K is the total number of weighted region triangles that include at least one point.

5.1.8 *Algorithm overview.* Here is the algorithmic representation of the point sampling method:

ALGORITHM 1: Point sampling

$\gamma \leftarrow$ input curve
$WRT \leftarrow$ weighted region triangulation
$K \leftarrow$ number of points to generate

triangulate γ to γ_{tr}
for *each* $triangle(\tau)$ *in* γ_{tr} **do**
    assign index i and weight w_τ
end
repeat
    randomly choose τ_i given a distribution $D(x_n)$
    generate point p_{rnd} inside τ_i
    store p_{rnd} in $array_{p_{rnd}}$
    k++
until k =K;
transform $array_{p_{rnd}}$ from $\gamma_{tr} system$ to $WRTsystem$
for *each* WRT_{tr} *in* WRT **do**
    $counter \leftarrow$ count p_{rnd} that lie in each triangle
    $C_{swipe} = C_{swipe} + counter \cdot w_{WRT_{tr}}$
end
return C_{swipe}

5.2 Parallel lines method

This technique is based on the intersection of multiple parallel lines (generated inside the capsule cross-cut contour) with the edges that define the weighted regions.That is, we took elements from the definition of the integral and modified them to a more finite approach. In other words, we approach the area of the capsule cross-cut using a specific number of lines, thus making the problem less complex. Consider the integral definition:

$$\int_P f(x)\,dx = \lim_{n\to\infty} \sum_{i=1}^{n} (f(x_i)\delta(x_i)) \tag{23}$$

where P is the partition of a set D with thinness $||P|| \to 0$ and $\delta(x_i)$ is an element of partition P positioned on x_i. For the current problem, we keep $||P||$ as it is (since we are referring to lines) and change n to point to a finite number $n \to N$ as well as modify $f(x_i)$ to denote the length of the line segment with respect to each intersecting weighted region, starting from sample point x_i (and consequently ending to the adjacent x_i').

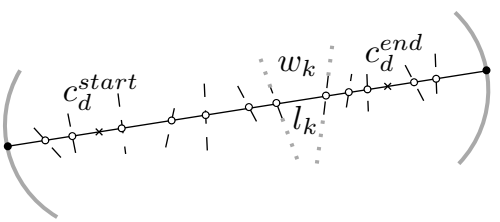

Fig. 21. Every segment defined among two intersections is characterized by the weight w_k of the triangle k including the intersecting (dashed) edges and the length l_k of the segment.

Another concrete depiction of our idea derives from the *Darboux Integral* [Weisstein 1999]. Darboux approximated the area covered by a function using the *upper* and *lower integral*. Consider, for instance, the upper integral

as the infimum of all upper sums of the form:

$$U(P) = \sum_{r=1}^{n} M(f, \delta_r)(\alpha(x_r) - \alpha(x_{r-1})) \tag{24}$$

where f and α are real functions bounded on interval $[a, b]$ and P is a partition given by $a = x_0 < x_1 < ...x_n = b$, let $\delta_r = [x_{r-1}, x_r]$. For $\delta_r \to 0$ Darboux precisely describes the aforementioned area as the sum of all the areas of infinitely thin orthogonal parallelograms (see Figure 22).

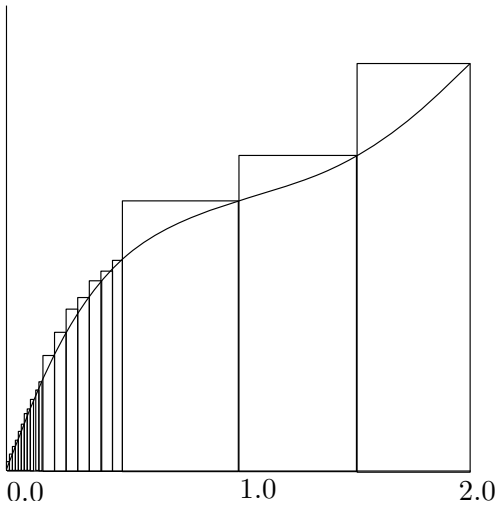

Fig. 22. Darboux upper sums of a function f for $step = \frac{1}{16}, \frac{1}{4} and \frac{1}{2}$.

We consider these infinitely thin parallelograms as lines placed based on density selected by the user depending on performance and precision preferences. Specifically, the method considers two semicircles which actually define the starting and ending arcs of the swipe shape. Then, we generate multiple points using uniform distribution in each semicircle and use them as the endpoints of each line segment. Since we are not dealing with functions, we use parametric equations to describe the curves. Finally, by combining the partial lengths of each line segment, we conclude to the final swipe cost equation (27) in paragraph 5.2.5.

5.2.1 *Define the semicircles.* In contrast with the method in 5.1, we do not have to represent the entire curve. The parts that this algorithm considers are the two external semicircles with radius $\rho = \rho_d$ centered and the start and end points defined by the disk center $c_d \in \mathbb{R}^2$. Firstly, we align the axis of disk translation with the Cartesian system the arcs are going to be generated. This is because the angular interval has to be clearly defined. Thus we use two different sets of parametric equations:

$$x = c_d^{start} + sin(t), \quad y = c_d^{start} + cos(t) \tag{25}$$

$$x' = c_d^{end} + sin(t), \quad y' = c_d^{end} + cos(t) \tag{26}$$

where t is the angle such as $0 < t < \pi$. Each of (25) and (26) refer to the initial and final disk location (see Figure 23).

5.2.2 *Curve point sampling.* We apply uniform point sampling similar to Section 5.1.4 on both semicircle curves γ and γ' created in the previous paragraph. The difference is that we use the generated n points not to transform the curve to a polygon but to define the start and ending vertices of each of the line segments. Since the lines are parallel, we could alternatively generate points only on one semicircle and define the adjacent points

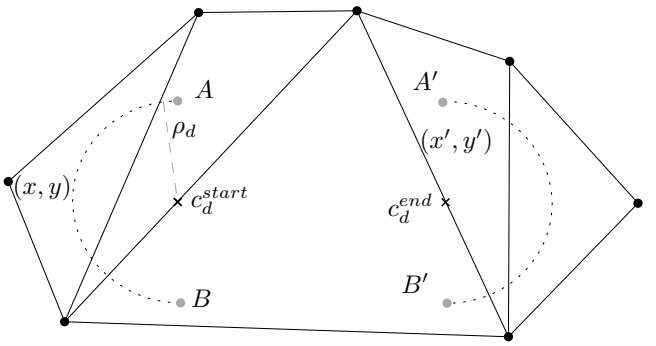

Fig. 23. Parametric curves define the two semicircles.

defined by the intersection of the line with the other curve. The number n of starting points is specified depending on the ratio between performance and accuracy the user intends to achieve. In Figure 24 we depict point picking on a semicircle with $step$ equal to 7.5° for $\widehat{AC}$, 11.25° for $\widehat{CD}$ and 30° for $\widehat{DB}$.

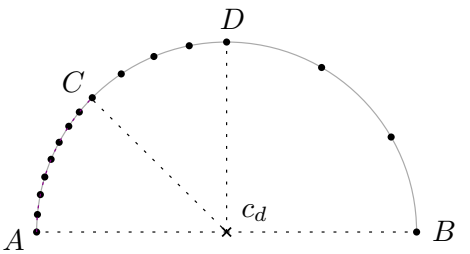

Fig. 24. Circle point picking for $step = 7.5°, 11.25°$ and 30°.

5.2.3 *Line segment generation.* Let $\widehat{AB}$ and $\widehat{A'B'}$ (see Figure 23) be the two arcs with $\theta = \pi$ and $r = \rho_d$. Moreover, consider n parallel line segments $\overline{uu'}$ from $\widehat{AB}$ to $\widehat{A'B'}$. Besides the density -denoted by n- of lines, the shape will always have to include the two bounding segments AB and $A'B'$ as well as the middle segment defined by c_d translation (see Figure 25). Thus, the inferior limit of n, in order for the problem to refer to disk and not point movement, is $n \geq 3$.

5.2.4 *Intersections of Weighted segments.* Having computed the line segments in the previous step, we continue by indicating the line-triangle intersections and then computing the total weights for each line segment (see Figure 25). In order to save computations, we first apply line-triangle collision and if necessary calculate the intersection points.

First, since both the triangle and the line are convex objects, we introduce an intersection calculation method based on *Separating Hyperplane test* (and not theorem as most commonly referred). It is commonly used in computer science for fast collision detection between polygon meshes. The Separating Hyperplane test is merely a version of the *Hahn-Banach theorem* [Pales 1989] for functional analysis. Briefly, it states that two convex sets can be separated by a hyperplane(line in our case) with one of the convex sets lying on one side of it and the other set sitting on the other. That is,if we can find an axis along which the projection of the two shapes does not overlap, then the shapes don't overlap [Houle 1991] [Hladík 2004] . Then, if an intersection exists we solve the equation system between the line segment and the triangle's segments.

In the second phase, every line segment gets separated in smaller subsegments depending on the intersecting weighted regions. Thus, for a segment $\overline{uu'}$ with k intersections (excluding starting and final vertices) there are $k+1$ segments with different weights in the form of $\overline{uu_1}, \overline{u_1u_2}, ...\overline{u_{k-1}u_k}, \overline{u_ku'}$.

5.2.5 *Swipe cost.* Consider the index j over the intersection vertices (including u and u') and function $F(u_j)$ such as:

$$F(u_j) = \begin{cases} u & \text{when } j = 0 \\ u_1 & \text{when } j = 1 \\ ... & \\ u_k & \text{when } j = k \\ u' & \text{when } j = k+1 \end{cases}$$

The function $F(u_j)$ returns the intersection vertex u_j starting from the point u to u'. Thus, the final cost equation for all the lines in the swiping disk shape is:

$$C_{swipe} = \sum_{i=1}^{n} \sum_{j=0}^{k} d(F(u_j^i), F(u_{j+1}^i)) w(j \rightarrow j+1) \tag{27}$$

where i is the index of the uniformly generated points on $\widehat{AB}$ or $\widehat{A'B'}$, $d(n,n')$ is a distance function and $w(j \rightarrow j+1)$ is the weight that corresponds to the subsegment $\overline{u_ju_{j+1}}$.

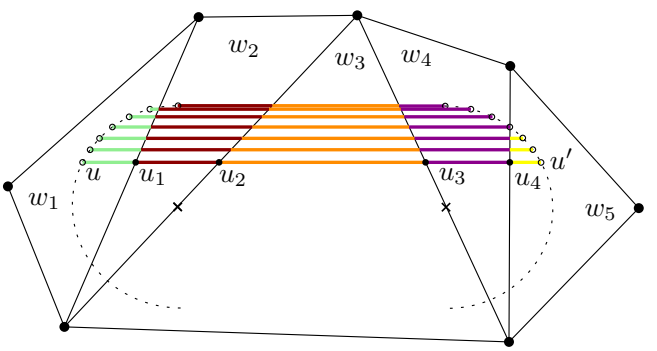

Fig. 25. Intersection of parallel line segments defined among the two parametric curves.

5.2.6 *Algorithm overview.* This is the algorithmic representation of parallel lines method

ALGORITHM 2: Point sampling

$c_d^{start} \leftarrow$ disk start vertex
$c_d^{end} \leftarrow$ disk end vertex

generate semicircles $\widehat{AB}$ and $\widehat{A'B'}$
repeat
    generate point u^n on $\widehat{AB}$
    n++
until $n = N$;
for *each* $\overline{uu'}$ *from* $\widehat{AB}$ *to* $\widehat{A'B'}$ **do**
    find $k+2$ intersections on WRT
    for *each* $k+2$ *on* $\overline{uu'}$ **do**
        $C_{line} = C_{line} + \overline{u_k u_{k+1}} w_{k \to k+1}$
    end
    $C_{swipe} = C_{swipe} + C_{line}$
end
return C_{swipe}

6. CONCLUSION

In this report, we proposed a method for creating indicative routes in weighted regions while preserving clearance. The first part of the essay (see Section 4) presents two methods responsible for maintaining clearance and computing the cost of the moving point. The second part (see Section 5) includes two methods that approximate the cost of the moving disk. Methods from the first and second part can be independently combined to produce the final indicative routes.

The first method of the indicative routes techniques (see Section 4.1) works better in case of many narrow passages considering a given disk radius. That is, using barrier construction it eliminates large parts of the scene thus excluding them from further computations. On the other hand, the second scheme (see Section 4.2) works better in presence of less narrow passages and produce more accurate indicative routes. Improved accuracy is due to the use of Steiner points as candidates comparing to the midpoint of every traversable edge in the former scheme. However, it is slower during narrow passage detection. Disk radius around constrained edges' intersections may discard massive number of candidate points but then, checking for valid points is done one by one.

Concerning disk cost estimation approaches, computation time for both mainly depends on user preferences (see Sections 5.1 and 5.2). Regarding *Point sampling method* process time increases when too many random points are generated because then, they have to be identified in terms of the triangles that lie in. The same stands for the *Parallel lines method* since the generation of two many line segments interprets to linearly increased number of weighted segment length calculations. It is possible for the second technique to present increased performance for large-area weighted regions because of the decreased intersections between the weighted line segments and weighted terrain. Large weighted regions also improve the first method but slower because all random points have to be counted and categorized again. Here, we have to mention that the magnitude of the area of every weighted region is characterized according to the size of the disk.

Different combinations of the first and second phase techniques can be tested regarding various variables. The main factors are the number of narrow passages, the size of the tessellation of the triangulated regions and given disk radius. Different numbers of narrow passages will reveal the validity of deadlock detection and various weighted region triangulations will test the plausibility and validity of the resulted paths. The former two experiments can produce interesting additional results considering various disk sizes.

In conclusion, this essay described ways to tackle the weighted region problem including clearance information. A global path planner, entirely based on graph-search methods comprised by two main parts. The first, presents techniques that narrate the subsequent planner to more specific calculations. This is conducted either by channel search or massive point filtering. At the second stage, two cost estimation approaches described which can independently combined with the algorithms from the previous stage. Moreover, these cost functions indicate the end of the procedure by generating the final indicative route. Then, the route can be used as input for local motion planners [van den Berg et al. 2008][Jaklin et al. 2013] in order to construct a complete path planner.

7. FUTURE WORK

Several ideas can be proposed as future work referring to multiple parts of the essay. Initially, the proposed methods in Section 4 and 5 can be extended to 3-dimensional path planning. To wit, the character is depicted by a sphere and the weighted regions are represented by weighted volumes. Another enhancement would be to include non-convex and dynamic obstacles as well as dynamic disc size. This requires a fast indicative route method able to be executed in real time. Moreover, dynamic properties in the scene will allow the inclusion of moving obstacles and more plausible character behaviors (e.g. extending character's hands).

In Section 5 BUSHWHACK can be replaced by BUSHWHACK+ [Sun and Reif 2003], a variation of the former. that uses several cost-saving heuristics. In addition, regarding Section 5.1.4, random points can be counted by incorporating the faster Sweep line algorithm [Shamos and Hoey 1976]. In case of variant gradient curvatures (i.e. non-circular character), parametric curve approximation (see Section 5.2) could be implemented using Polygonal Approximation of Parametric Surfaces [Henrique and Figueiredo 1996].

Furthermore the tessellation of the area can be conducted by constrained Delaunay triangulation based on Off-centers [Üngör 2009]. Off-center method needs fewer triangles to explore thus less process complexity (while the triangulation optimality is maintained) comparing to the usual CDT based Steiner points segmentation.

REFERENCES

Lyudmil Aleksandrov, Mark Lanthier, Anil Maheshwari, and Jörg-R. Sack. 1998. An ϵ-Approximation algorithm for weighted shortest paths on polyhedral surfaces. In *Algorithm Theory - SWAT'98*, Stefan Arnborg and Lars Ivansson (Eds.). Lecture Notes in Computer Science, Vol. 1432. Springer Berlin Heidelberg, 11–22. DOI:http://dx.doi.org/10.1007/BFb0054351

N.M. Amato and Y. Wu. 1996. A randomized roadmap method for path and manipulation planning. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, Vol. 1. 113–120 vol.1. DOI:http://dx.doi.org/10.1109/ROBOT.1996.503582

Nancy M. Amato, Michael T. Goodrich, and Edgar A. Ramos. 2000. Linear-time Triangulation of a Simple Polygon Made Easier via Randomization. In *Proceedings of the Sixteenth Annual Symposium on Computational Geometry (SCG '00)*. ACM, New York, NY, USA, 201–212. DOI:http://dx.doi.org/10.1145/336154.336206

Jerome Barraquand, Lydia Kavraki, Jean-Claude Latombe, Tsai-Yen Li, Rajeev Motwani, and Prabhakar Raghavan. 1996. A Random Sampling Scheme for Path Planning. *INTERNATIONAL JOURNAL OF ROBOTICS RESEARCH* 16 (1996), 759–774.

Jean-Daniel Boissonnat. 1988. Shape reconstruction from planar cross sections. *Computer Vision, Graphics, and Image Processing* 44, 1 (1988), 1 – 29. DOI:http://dx.doi.org/10.1016/S0734-189X(88)80028-8

J. Casey. 1882. *A Sequel to the First Six Books of the Elements of Euclid, Containing an Easy Introduction to Modern Geometry: With Numerous Examples*. Hodges, Figgis & Company.

B. Chazelle. 1990. Triangulating a simple polygon in linear time. In *Foundations of Computer Science, 1990. Proceedings., 31st Annual Symposium on*. 220–230 vol.1. DOI:http://dx.doi.org/10.1109/FSCS.1990.89541

Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. 2008. Visibility Graphs. In *Computational Geometry*. Springer Berlin Heidelberg, 323–333. DOI:http://dx.doi.org/10.1007/978-3-540-77974-2_15

Olivier Devillers, Sylvain Pion, and Monique Teillaud. 2001. Walking in a triangulation. *Internat. J. Found. Comput. Sci* 13 (2001), 106–114.

E. W. Dijkstra. 1959. A Note on Two Problems in Connexion with Graphs. *NUMERISCHE MATHEMATIK* 1, 1 (1959), 269–271.

J.E. Gentle. 2003. *Random Number Generation and Monte Carlo Methods*. Springer.

Roland Geraerts. 2010. Planning short paths with clearance using explicit corridors.. In *ICRA*. IEEE, 1997–2004. http://dblp.uni-trier.de/db/conf/icra/icra2010.html#Geraerts10

P.E. Hart, N.J. Nilsson, and B. Raphael. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *Systems Science and Cybernetics, IEEE Transactions on* 4, 2 (1968), 100–107. DOI:http://dx.doi.org/10.1109/TSSC.1968.300136

Luiz Henrique and De Figueiredo. 1996. Optimal Adaptive Polygonal Approximation of Parametric Surfaces. (1996). http://citeseerx.ist.psu.edu/viewdoc/download?rep=rep1&type=pdf&doi=10.1.1.3.8331

Milan Hladík. 2004. Explicit description of all separating hyperplanes of two convex polyhedral sets with RHS-parameters. In *WDS 2004 - Proceedings of Contributed Papers, Part I, June 15-18, Matfyzpress, Prague*, Jana Šafránková (Ed.). 63–70.

Kenneth E. Hoff, III, John Keyser, Ming Lin, Dinesh Manocha, and Tim Culver. 1999. Fast Computation of Generalized Voronoi Diagrams Using Graphics Hardware. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 277–286. DOI:http://dx.doi.org/10.1145/311535.311567

R.C. Holte, R. C. Holte, M.B. Perez, M. B. Perez, R. M. Zimmer, R. M. Zimmer, A.J. MacDonald, and A. J. Macdonald. 1996. Hierarchical A*: Searching Abstraction Hierarchies Efficiently. In *In Proceedings of the National Conference on Artificial Intelligence*. 530–535.

MichaelE. Houle. 1991. Theorems on the existence of separating surfaces. *Discrete & Computational Geometry* 6, 1 (1991), 49–56. DOI:http://dx.doi.org/10.1007/BF02574673

Norman Jaklin, Atlas Cook, and Roland Geraerts. 2013. Real-time path planning in heterogeneous environments. *Computer Animation and Virtual Worlds* 24, 3-4 (2013), 285–295. DOI:http://dx.doi.org/10.1002/cav.1511

Marcelo Kallmann. 2010. Shortest Paths with Arbitrary Clearance from Navigation Meshes. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '10)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 159–168. http://dl.acm.org/citation.cfm?id=1921427.1921451

Ioannis Karamouzas, Roland Geraerts, and Mark Overmars. 2009. Indicative Routes for Path Planning and Crowd Simulation. (2009).

L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on* 12, 4 (Aug 1996), 566–580. DOI:http://dx.doi.org/10.1109/70.508439

O. Khatib. 1985. Real-time obstacle avoidance for manipulators and mobile robots. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, Vol. 2. 500–505. DOI:http://dx.doi.org/10.1109/ROBOT.1985.1087247

S.M. LaValle. 2006. *Planning Algorithms*. Cambridge University Press.

C. L. Lawson. 1977. Software for C^1 surface interpolation. In *Mathematical Software III*, J. Rice (Ed.). Academic Press, 161–194.

S. Legrand, V. Legat, and E. Deleersnijder. 2000. Delaunay mesh generation for an unstructured-grid ocean general circulation model. *Ocean Modelling* 2, 1âĂŞ2 (2000), 17 – 28. DOI:http://dx.doi.org/10.1016/S1463-5003(00)00005-6

Terence Lindgren, Juan Sanchez, and Jim Hall. 1992. Graphics Gems III. Academic Press Professional, Inc., San Diego, CA, USA, Chapter Curve Tessellation Criteria Through Sampling, 262–265. http://dl.acm.org/citation.cfm?id=130745.130790

Tomás Lozano-Pérez and Michael A. Wesley. 1979. An Algorithm for Planning Collision-free Paths Among Polyhedral Obstacles. *Commun. ACM* 22, 10 (Oct. 1979), 560–570. DOI:http://dx.doi.org/10.1145/359156.359164

Cristian S. Mata and J.S.B. Mitchell. 1997. A New Algorithm for Computing Shortest Paths in Weighted Planar Subdivisions (Extended Abstract). In *In Proc. 13th Annu. ACM Sympos. Comput. Geom.* ACM Press, 264–273.

Joseph S. B. Mitchell and Christos H. Papadimitriou. 1991. The Weighted Region Problem: Finding Shortest Paths Through a Weighted Planar Subdivision. *J. ACM* 38, 1 (Jan. 1991), 18–73. DOI:http://dx.doi.org/10.1145/102782.102784

Zsolt Pales. 1989. A Hahn-Banach theorem for separation of semigroups and its applications. *aequationes mathematicae* 37, 2-3 (1989), 141–161. DOI:http://dx.doi.org/10.1007/BF01836441

Samuel Rodriguez, Shawna Thomas, Roger Pearce, and NancyM. Amato. 2008. RESAMPL: A Region-Sensitive Adaptive Motion Planner. In *Algorithmic Foundation of Robotics VII*, Srinivas Akella, NancyM. Amato, WesleyH. Huang, and Bud Mishra (Eds.). Springer Tracts in Advanced Robotics, Vol. 47. Springer Berlin Heidelberg, 285–300. DOI:http://dx.doi.org/10.1007/978-3-540-68405-3_18

Rudiger Seydel. 2012. Generating Random Numbers with Specified Distributions. In *Tools for Computational Finance*. Springer London, 75–108. DOI:http://dx.doi.org/10.1007/978-1-4471-2993-6_2

Michael Ian Shamos and Dan Hoey. 1976. Geometric intersection problems. In *Foundations of Computer Science, 1976., 17th Annual Symposium on*. 208–215. DOI:http://dx.doi.org/10.1109/SFCS.1976.16

Micha Sharir and Amir Schorr. 1984. On Shortest Paths in Polyhedral Spaces. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing (STOC '84)*. ACM, New York, NY, USA, 144–153. DOI:http://dx.doi.org/10.1145/800057.808676

Peter Shirley and Steve Marschner. 2009. *Fundamentals of Computer Graphics* (3rd ed.). A. K. Peters, Ltd., Natick, MA, USA.

Zheng Sun and John Reif. 2001. BUSHWHACK: An Approximation Algorithm for Minimal Paths Through Pseudo-Euclidean Spaces. In *In Proceedings of the 12th Annual International Symposium on Algorithms and Computation*. Springer, 160–171.

Zheng Sun and John Reif. 2003. Adaptive and Compact Discretization for Weighted Regions Optimal Path Finding. In *In Proc. 14th Sympos. on Fundamentals of Computation Theory, LNCS*. Springer-Verlag, 258–270.

L.A. Treinish. 1995. Visualization of scattered meteorological data. *Computer Graphics and Applications, IEEE* 15, 4 (1995), 20–26. DOI:http://dx.doi.org/10.1109/38.391486

Alper Üngör. 2009. Off-centers: A new type of Steiner points for computing size-optimal quality-guaranteed Delaunay triangulations. *Computational Geometry* 42, 2 (2009), 109 – 118. DOI:http://dx.doi.org/10.1016/j.comgeo.2008.06.002

J. van den Berg, Ming Lin, and D. Manocha. 2008. Reciprocal Velocity Obstacles for real-time multi-agent navigation. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. 1928–1935. DOI:http://dx.doi.org/10.1109/ROBOT.2008.4543489

D.F. Watson. 1992. *Contouring: a guide to the analysis and display of spatial data*. Number v. 1 in Computer methods in the geosciences. Pergamon Press.

Eric W Weisstein. 1999. *CRC concise encyclopedia of mathematics*. CRC Press, Boca Raton, FL.

Frank Weller. 1998. On the Total Correctness of Lawson's Oriented Walk Algorithm (Extended Abstract). In *The 10th International Canadian Conference on Computational Geometry, Montral, Qubec*. 10–12.