

Institutionen för systemteknik

Department of Electrical Engineering

Examensarbete

Search path generation with UAV applications using approximate convex decomposition

Examensarbete utfört i Reglerteknik
vid Tekniska högskolan vid Linköpings universitet
av

Gustav Öst

LiTH-ISY-EX--12/4553--SE

Linköping 2012



Linköpings universitet
TEKNISKA HÖGSKOLAN

Search path generation with UAV applications using approximate convex decomposition

Examensarbete utfört i Reglerteknik
vid Tekniska högskolan i Linköping
av


Gustav Öst

LiTH-ISY-EX--12/4553--SE

Handledare: **Clas Veibäck**
Instrument Control Sweden AB
Per Skoglar
isy, Linköpings universitet
Sina Khoshfetrat Pakazad
isy, Linköpings universitet

Examinator: **Daniel Axehill**
isy, Linköpings universitet

Linköping, 3 Unknown, 2012

	Avdelning, Institution Division, Department Division of Automatic Control Department of Electrical Engineering Linköpings universitet SE-581 83 Linköping, Sweden	Datum Date 2012-30-03			
Språk Language <input type="checkbox"/> Svenska/Swedish <input checked="" type="checkbox"/> Engelska/English <input type="checkbox"/> _____	Rapporttyp Report category <input type="checkbox"/> Licentiatavhandling <input checked="" type="checkbox"/> Examensarbete <input type="checkbox"/> C-uppsats <input type="checkbox"/> D-uppsats <input type="checkbox"/> Övrig rapport <input type="checkbox"/> _____	ISBN _____ ISRN LiTH-isy-ex--12/4553--SE Serietitel och serienummer ISSN Title of series, numbering _____			
URL för elektronisk version http://www.control.isy.liu.se http://www.ep.liu.se					
<table border="0"> <tr> <td data-bbox="167 638 287 684">Titel Title</td> <td data-bbox="293 638 1117 711"> Avsökningsskriptor för UAVer med hjälp av approximativ konvexuppdelning Search path generation with UAV applications using approximate convex decomposition </td> </tr> <tr> <td data-bbox="167 797 287 842">Författare Author</td> <td data-bbox="293 797 1117 842">Gustav Öst</td> </tr> </table>			Titel Title	Avsökningsskriptor för UAVer med hjälp av approximativ konvexuppdelning Search path generation with UAV applications using approximate convex decomposition	Författare Author
Titel Title	Avsökningsskriptor för UAVer med hjälp av approximativ konvexuppdelning Search path generation with UAV applications using approximate convex decomposition				
Författare Author	Gustav Öst				
Sammanfattning Abstract <p> This work focuses on the problem that pertains to area searching with UAVs. Specifically developing algorithms that generate flight paths that are short without sacrificing flyability. For instance, very sharp turns will compromise flyability since fixed wing aircraft cannot make very sharp turns. This thesis provides an analysis of different types of search methods, area decompositions, and combinations thereof. The search methods used are side to side searching and spiral searching. In side to side searching the aircraft goes back and forth only making 90-degree turns. Spiral search searches the shape in a spiral pattern starting on the outer perimeter working its way in. The idea being that it should generate flight paths that are easy to fly since all turns should be with a large turn radii. Area decomposition is done to divide complex shapes into smaller more manageable shapes. The report concludes that with the implemented methods the side to side scanning method without area decomposition yields good and above all very reliable results. The reliability stems from the fact that all turns are 90 degrees and that algorithm never get stuck or makes bad mistakes. Only having 90 degree turns results in only four different types of turns. This allows the airplanes behavior along the route to be predictable after flying the first four turns. Although this assumes that the strength of the wind is a greater influence than the turbulences effect on the aircraft's flight characteristics. This is a very valuable feature for an operator in charge of a flight. The other tested methods and area decompositions often yield a shorter flight path, however, despite extensive adjustments to the algorithms they never came to handle all cases in a satisfactory manner. These methods may also generate any kind of turn at any time, including turns of nearly 180 degrees. These turns can lead to an airplane missing the intended flight path and thus missing to scan the intended area properly. Area decomposition proves to be really effective only when the area has many protrusions that stick out in different directions, think of a starfish shape. In these cases the side to side algorithm generate a path that has long legs over parts that are not in the search area. When the area is decomposed the algorithm starts with, for example, one arm of the starfish at a time and then search the rest of the arms and body in turn. </p>					
Nyckelord Keywords UAV, Search path, Approximate convex decomposition					

Abstract

This work focuses on the problem that pertains to area searching with UAVs. Specifically developing algorithms that generate flight paths that are short without sacrificing flyability. For instance, very sharp turns will compromise flyability since fixed wing aircraft cannot make very sharp turns. This thesis provides an analysis of different types of search methods, area decompositions, and combinations thereof. The search methods used are side to side searching and spiral searching. In side to side searching the aircraft goes back and forth only making 90-degree turns. Spiral search searches the shape in a spiral pattern starting on the outer perimeter working its way in. The idea being that it should generate flight paths that are easy to fly since all turns should be with a large turn radii. Area decomposition is done to divide complex shapes into smaller more manageable shapes. The report concludes that with the implemented methods the side to side scanning method without area decomposition yields good and above all very reliable results. The reliability stems from the fact that all turns are 90 degrees and that algorithm never get stuck or makes bad mistakes. Only having 90 degree turns results in only four different types of turns. This allows the airplanes behavior along the route to be predictable after flying the first four turns. Although this assumes that the strength of the wind is a greater influence than the turbulences effect on the aircraft's flight characteristics. This is a very valuable feature for an operator in charge of a flight. The other tested methods and area decompositions often yield a shorter flight path, however, despite extensive adjustments to the algorithms they never came to handle all cases in a satisfactory manner. These methods may also generate any kind of turn at any time, including turns of nearly 180 degrees. These turns can lead to an airplane missing the intended flight path and thus missing to scan the intended area properly. Area decomposition proves to be really effective only when the area has many protrusions that stick out in different directions, think of a starfish shape. In these cases the side to side algorithm generate a path that has long legs over parts that are not in the search area. When the area is decomposed the algorithm starts with, for example, one arm of the starfish at a time and then search the rest of the arms and body in turn.

Sammanfattning

Detta arbete fokuserar på problemet area-avsökning med UAVer. Specifikt på att utveckla algoritmer som genererar flygbanor som är korta utan att offra flygbarhet. Väldigt skarpa svängar kan till exempel förstöra flygbarheten eftersom flygplan inte kan svänga särdeles tvärt.

Det här arbetet ger en analys av olika sökmetoder, områdesdekompositioner och kombinationer av dessa. Sökmetoderna som används är att flyga sida till sida och i spiral-mönster. I sida till sida-avsökning så flyger flygplanet fram och tillbaka med 90-graders svängar. Vid spiralavsökning så avsöks området i ett spiralmönster utifrån och in. Idén bakom denna metod är att den kommer generera flygbanor som är enkla att följa eftersom alla svängar kommer att vara små. Områdesdekomposition görs för att dela upp komplexa områden i mindre, mer hanterbara områden. I rapporten så dras slutsatsen att med de implementerade metoderna så är sida till sida metoden utan områdesdekomposition en som ger bra och framförallt väldigt tillförlitliga resultat.

Tillförlitligheten kommer som en konsekvens av att alla svängar är 90 grader och att algoritmen aldrig hakar upp sig eller gör stora misstag ifall avsökningsområdet är komplext. Att endast ha 90-gradiga svängar gör att det bara förekommer 4 olika typer av svängar. Detta gör att flygplanets beteende längs med rutten blir förutsägbar efter de fyra första svängarna. Dock så förutsätter detta att vindens inflytande är starkare än turbulensens vad det gäller flygplanets flygprestanda. Detta är en värdefull egenskap för operatören som har hand om flygningen.

Övriga testade metoder och områdesdekompositioner gav ofta kortare flygbanor men trots omfattande justeringar på algoritmerna så kom de aldrig att hantera alla olika fall tillfredsställande. Dessa metoder kan också generera vilken typ av sväng som helst när som helst, inklusive svängar som är nära 180 grader. Sådana svängar kan leda till att flygplanet hamnar långt ifrån den avsedda flygbanan och därmed missa att söka av områden som hör till den planerade flygbanan.

Områdesdekomposition visar sig vara en god idé endast om området som skall sökas av har många utstickande delar åt olika håll, tänk på en sjöstjärna inte en fabrik i siluett. I dessa fall så generar sida till sida algoritmen en flygbana som till stora delar går över områden som inte skall sökas av. När man delar upp området så börjar algoritmen istället med, till exempel, sjöstjärnans armar en i taget och sedan resten av kroppen.

Acknowledgments

Thanks goes out to Paul and Per at Instrument Control Sweden AB for taking me in and for helping and supporting me in my thesis work.

Thanks also goes out to my dear mother for getting me back into school by means of gently pushing me towards school and saying I got nothing better to do. She was right.

Finally I would like to thank evolution for placing me here on the cooler, less dangerous part of this wonderful blue-green oblate spheroid.

Contents

1	Introduction	1
1.1	Background	1
1.2	Goal	2
1.3	Similar work	2
1.4	Delimitation	3
2	Theory	5
2.1	Search area	5
2.1.1	Area definition	5
2.1.2	Polygons and shapes	6
2.2	Shape alteration and decomposition	7
2.2.1	Orthonormal basis	7
2.2.2	Convex hull	8
2.2.3	Convex decomposition	8
2.2.4	Approximate convex decomposition	9
2.3	Search algorithms	13
2.3.1	Spiral	14
2.3.2	Back and forth	16
2.3.3	Shortening path by trying different rotations	20
3	Implementation	23
3.1	Matlab	23
3.2	Commercial Software	23
4	Results and discussion	25
4.1	Performance	25
4.1.1	Methods	25
4.1.2	Simulation results	26
4.1.3	Simulation time consumption	27
4.1.4	Result discussion	27
5	Conclusion	31
5.1	Approximate convex decomposition	31
5.1.1	Pros	31
5.1.2	Cons	31

5.2	Search algorithms	32
5.2.1	Spiral method	32
5.2.2	Side to side method	33
5.3	All in all	33
6	Future work	35
6.1	Decomposition	35
6.2	Search algorithms	35
6.2.1	Spiral	35
6.2.2	Side to side	37
6.3	Implementation	37
	Bibliography	39

Chapter 1

Introduction

This thesis was made at Instrument Control Sweden (ICS) based in Linköping. ICS is a company that builds and develops Unmanned Aircraft Systems (UAS). Particularly, their focus is on the development of an autopilot system called EasyPilot and a Ground Control Station (GCS) called SkyView.

1.1 Background

SkyView is the graphical interface that an operator will use in order to plan, layout, control and execute missions with an unmanned aircraft. In this software the operator can plan routes in a few different ways. It is possible to plan a route by clicking on a map and adding waypoints and/or loitering points.

When working with law enforcement, emergency services or military customers UAVs are often needed for searching an area. This stems from various task assignments such as finding a hidden perpetrator, searching for a missing person and detection of enemy activity, etc.

A waypoint grid to search an area could be constructed by an operator manually but would involve a lot of guessing. For instance, how wide is the ground coverage for this sensor at this altitude and therefore how far apart should the waypoints be. Errors in doing this will result in, at best, time loss due to excessive overlap in coverage and worst case, loss of coverage. Apart from this, route planning is time consuming. Time which they might not have.

Because of difficulty for civilians in getting licenses to fly unmanned aircrafts, the three mentioned groups represent the majority of potential customers in most of the world. Therefore the addition of a simple, quick and reliable way to search an area will add value to SkyView.

1.2 Goal

In this thesis, I develop and implement algorithms to satisfy the need for simplicity and quick generation of a good flight path when searching an area with a UAV. These algorithms are to generate a short flight path that yields full coverage of the desired area.

The algorithms will need to be flexible to consider limitations of UAVs such as elevation of the aircraft, sensor constraints and flying characteristics (speed, turn rate, turn radius, etc). Also it is important that an aircrafts behavior along the path be as predictable as possible as long as flight characteristics are known.

As for the implementation into the product (SkyView), the interface should be easy enough for an operator, who has limited prior knowledge of the product, to start using it directly. Also the only thing the operator will have to do on site is to define the area to be searched by clicking on the map and uploading the mission to the UAV.

1.3 Similar work

This work is related to work done by others in regards to area coverage and optimization of path length. The vast majority of this work is earth bound and therefore does not need to handle a lot of the problems occurring when flying.

For instance if an algorithm generates a path with a lot of sharp turns an aircraft might not be able to follow this path due to wind, too large turn radius or side slip of the aircraft. In this case, a theoretically shorter path would become worthless since coverage might be incomplete or time would be lost in recovering the aircrafts position. However the biggest loss would be in the loss of faith of the operator, if he/she does not trust the algorithm and is reluctant to use the algorithm, since the primary goal in using this when finding a missing person is **knowing** you have searched an area thoroughly.

A lot of similar work has to do with area coverage for automatic lawn mowers and vacuum cleaners, [8]. Aside from not considering aircraft specific problems they have problems that do not necessarily happen in the air, for example obstacle avoidance. In the air, obstacles are rare and therefore are not addressed in this work.

The majority of current work in the field of area coverage has to do with multiple UAVs continuously searching an area in real time, updating search strategies online and how to divide the workload between the UAVs for best efficiency, [14], [16], [10].

There is also work that focuses on search operations using roads and paths as restrictions and then searching for a specific target somewhere along these, [9].

In contrast to this my work will focus on offline route planning for a single UAV, user simplicity and predictability when it comes to aircraft behavior and flight time and distance.

When it comes to this specific approach academic papers seems to be in short

supply.

1.4 Delimitation

This work will be limited to offline route planning for a single UAV. It will also only focus on single non-complex areas, see Figure 2.2 for definition of non-complex area.

Chapter 2

Theory

This chapter offers descriptions and explanations of equations, algorithms and definitions used.

2.1 Search area

2.1.1 Area definition

In this work different kinds of areas and compositions of areas are used for testing search strategies. The basic search area is made up by an area either entered by a user or generated by software. The area is defined using its vertices and always forms a closed shape, [13], [6].

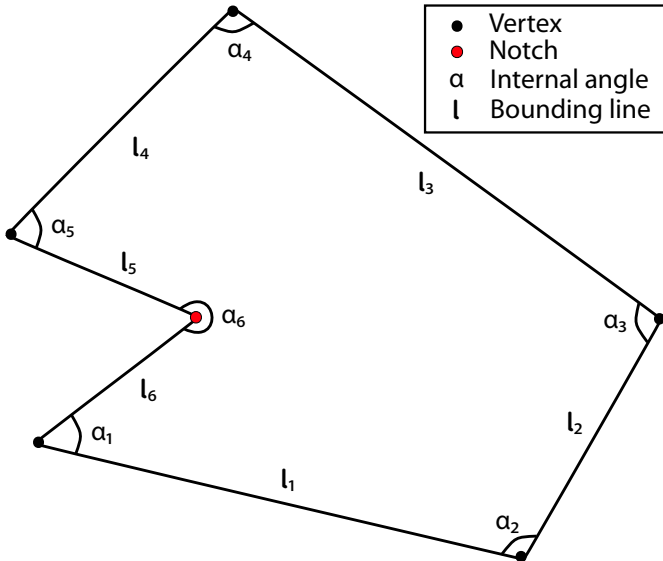


Figure 2.1. Nomenclature for polygon

2.1.2 Polygons and shapes

The shape of polygons used in this thesis can be either convex or concave but not complex, as defined in Figure 2.2. Complex polygons would not make up a well defined search area. It is not well defined since starting on a bounding line and deciding what side is inside the shape will yield different results depending on which bounding line one starts with. This in turn can lead to the whole world minus a small part being selected as a search area. To search the three apparent areas in Figure 2.2, three separate search areas would have to be used. The nomenclature used for different parts of polygons is explained in Figure 2.1

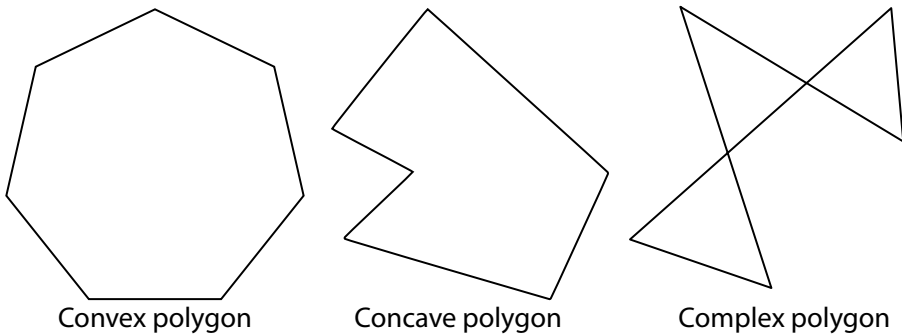


Figure 2.2. Illustration of the different types of polygons, [2], [4], [3].

Convex shape

A convex shape is the simplest of all shape types. It does not intersect with itself and all inner angles are smaller than 180 degrees.

Concave shape

A concave shape in this thesis refers to a shape with at least one inner angle larger than 180 degrees. This shape is common for user entered shapes and is also the most common shape generated by the software produced for this thesis. A big challenge in using this type of shape is that the area searching algorithms being developed in this thesis may not do a good job for this type of area shape. These algorithms might search parts of the same area several times, not search the entire area or just fail to finish properly. For these reasons a few different methods for processing these shapes are used and explained in the upcoming sections.

Complex shape

A complex shape is any shape that contains self intersection. Self intersection is when the shapes borders intersects with itself, see complex polygon in Figure 2.2, [6].

2.2 Shape alteration and decomposition

This section describes different methods of altering the search area in order to make handling of the area more manageable.

2.2.1 Orthonormal basis

The methods that are described and used in the upcoming sections all require that the shapes are described in an orthonormal base. Since the area that is treated is entered by a user by clicking on a map and that map uses latitude-longitude representation, the coordinates of the polygon will be in latitude-longitude format. This is not an orthonormal basis since each degree of change in longitude (λ) shrinks as the latitude (φ) increases, as can be seen in Figure 2.3. Therefore these coordinates need to be converted into an orthonormal base.

This can be solved as follows. The earth's shape is very nearly an oblate spheroid, a sphere flattened at the top and the bottom. However for small areas, when converting to an orthonormal basis, approximating the shape with a sphere and least square adjusting the error gives more than adequate accuracy for this thesis. Conversions will be to meters.

$$\begin{aligned}\Delta y &= 110574 + |\varphi| \cdot 12.4444 \text{ [m]} \\ \Delta x &= 111304 \cdot \cos(\varphi) + 81.2583 \text{ [m]}\end{aligned}\tag{2.1}$$

When changing basis of a shape into orthonormal basis the average latitude value is calculated and used. This is done by finding the middle latitude in the area to be searched and using this latitude as φ in Equation 2.1. An easy way of doing this is

$$\varphi = \frac{\varphi_{max} + \varphi_{min}}{2},$$

where φ_{max} is φ in the vertex in the polygon furthest to the north and φ_{min} is φ in the vertex furthest to the south. Then simply apply (2.1) to all vertices in the polygon to get a good approximation of the shape in an orthonormal base as follows.

$$\begin{bmatrix} \varphi_1 & \lambda_1 \\ \varphi_2 & \lambda_2 \\ \vdots & \vdots \\ \varphi_n & \lambda_n \end{bmatrix} \cdot \begin{bmatrix} \Delta y & 0 \\ 0 & \Delta x \end{bmatrix} = \begin{bmatrix} y_1 & x_1 \\ y_2 & x_2 \\ \vdots & \vdots \\ y_n & x_n \end{bmatrix}\tag{2.2}$$

Orthonormal base is assumed in all sections, examples and equations below.

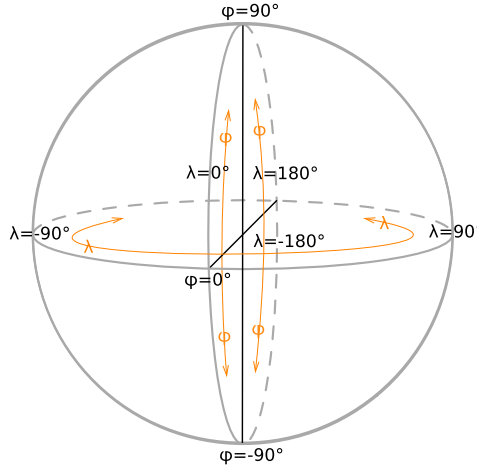


Figure 2.3. Spherical representation of latitude-longitude, [5].

2.2.2 Convex hull

A simple way of generating a convex shape out of a concave one is by using the notion of convex hulls. This is done by removing vertices that are notches. This would add the dotted line and remove the vertex inside it as in Figure 2.4.

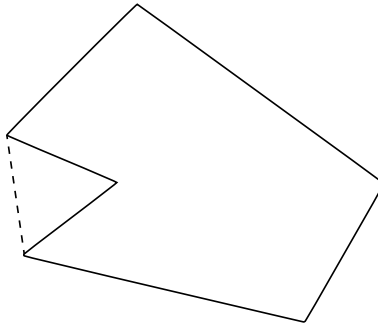


Figure 2.4. Convex hull of a concave shape

2.2.3 Convex decomposition

For algorithms that can only operate properly over convex shapes, non-convex ones are handled by convex decomposition. When decomposing a shape into convex shapes a lot of vertices with very sharp internal angles are often created. Searching areas that have vertices with small internal angles is harder and more wasteful than areas with only large inner angles. This is due to the fact that fixed-wing aircrafts cannot turn on the spot.

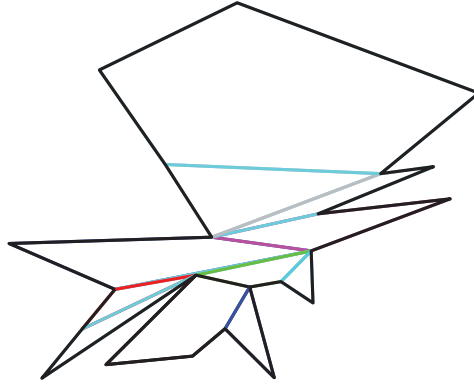


Figure 2.5. Convex decomposition with pointy shapes

2.2.4 Approximate convex decomposition

In some applications complete convex decomposition is not necessary, [11]. When searching an area using a fixed-wing UAV, absolute convexity is not as important as not having to make sharp turns.

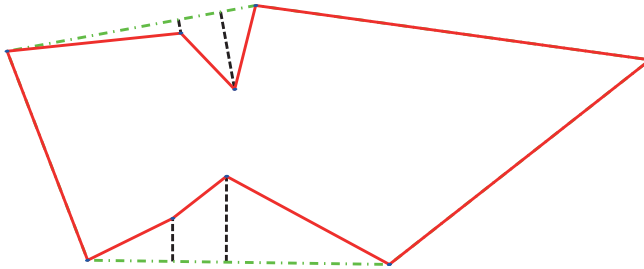


Figure 2.6. A shape with concavities

There are a few different ways of measuring what notch is the most critical to remove, [11]. One way is to see which notch has the greatest inner angle. Another is to see which notch is farthest away from the convex hull, dash-dotted lines in Figure 2.6. In this figure the distances are the dashed lines.

A third way is to calculate which notch removal makes the difference between the new shapes and their convex hulls as small as possible, [12]. Best case is zero, then the two new shapes are convex. There are different measures for quantifying this difference, namely

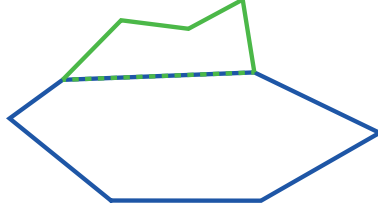


Figure 2.7. Shape decomposed using approximate convex decomposition leaving a concavity undecomposed

I) Distance to convex hull

Distance, d , from notch P0 to line segment between P1 and P2 can be calculated using the formula in (2.3), [15].

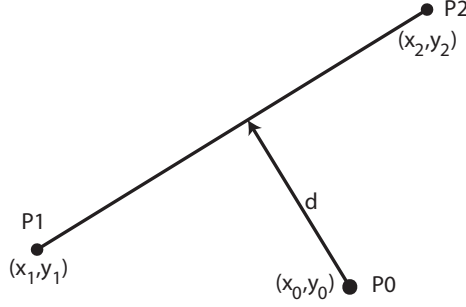


Figure 2.8. Distance from P0 to the line segment between P1 & P2

$$\begin{array}{lll} x_0 & = & P0_x \\ y_0 & = & P0_y \end{array} \quad \begin{array}{lll} x_1 & = & P1_x \\ y_1 & = & P1_y \end{array} \quad \begin{array}{lll} x_2 & = & P2_x \\ y_2 & = & P2_y \end{array}$$

$$d = \frac{\det \left(\begin{bmatrix} x_2 - x_1 & y_2 - y_1 \\ x_0 - x_1 & y_0 - y_1 \end{bmatrix} \right)}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \quad (2.3)$$

However if the normal to the line segment hits the point from a place outside the line segment the distance is instead calculated as the distance to the closest point that makes up the line segment (P1 or P2). Then d is calculated as in equation (2.4).

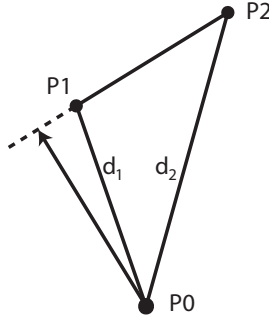


Figure 2.9. Normal beside the line segment

$$\begin{aligned}
 d_1 &= \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2} \\
 d_2 &= \sqrt{(x_2 - x_0)^2 + (y_2 - y_0)^2} \\
 d &= \min \{d_1, d_2\}
 \end{aligned} \tag{2.4}$$

II) Determining largest internal angle

This can be done by finding the notch with the internal angle $\alpha_{max} = \max\{\alpha_1, \alpha_2, \dots, \alpha_k\}$.

III) Area between notch and convex hull

Another way of measuring is to see which notch produces the largest area between the shape and its convex hull. For example, by looking at Figure 2.10 let us calculate the area generated by the lower concavity. Include adjacent vertices until a non-notch is included, P1 to the left and P4 to the right. This yields the four bottommost vertices P1-P4 to disappear. Now by applying the polygon area formula in (2.5), we can calculate the area, A , generated by this concavity, [1].

$$\begin{array}{lcl}
 n & = & 4 \\
 x_0 & = & x_4 \\
 y_0 & = & y_4 \\
 x_5 & = & x_1 \\
 y_5 & = & y_1
 \end{array}
 \quad
 \begin{array}{c}
 \left| \begin{array}{ccc}
 P & x & y \\
 1 & x_1 & y_1 \\
 2 & x_2 & y_2 \\
 3 & x_3 & y_3 \\
 4 & x_4 & y_4
 \end{array} \right|
 \end{array}
 \quad
 A = \frac{\sum_{i=1}^n x_i (y_{i+1} - y_{i-1})}{2} \tag{2.5}$$

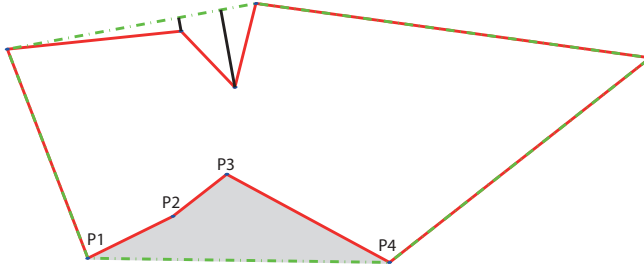


Figure 2.10. Area between notch and hull.

By using any of these measurements we can now decide if a notch is to be removed and in that case, which notch.

Determine if a notch is to be removed

After using the chosen method (distance, angle or area) on all the notches compare the values with a user supplied threshold value. For example, in simulations I used the sensors width on the ground as a value for the distance method as my threshold. In that case all notches that have a distance to the convex hull of less than the sensors width on the ground will remain undecomposed since they would not violate the threshold. Chosen threshold values and method depend on aircraft characteristics and type of mission. Generally, area is a good measurement for this purpose because it measures how much more area needs to be covered if searching the convex hull compared to the concave shape.

Choose the notch that violates the threshold the most. In the area case if several notches yield the same area one can look at either distance or angle to decide among them. In Figure 2.10 notches P2 and P3 yields the same area. Looking at either distance or angle would make P3 the removal candidate since the internal angle is greater and it is further from the convex hull.

Splitting the shape to remove notch

If a notch has been chosen for removal, split the shape by making a cut from the notch to another vertex. There are two ways of deciding which vertex to make the cut to. This is done by either maximizing the "roundness" in the new shapes or minimizing the concavity of the two new shapes. To maximize the roundness of the shapes one can minimize $\sum (\alpha_i - 180)^2$ in the new shapes, where α_i is the internal angle of vertex i in the shapes in degrees. This function punishes small internal angles severely.

To minimize the concavity of the new shapes ($S \rightarrow S1, S2$) minimize:

$$\frac{\text{Area}(\text{ConvexHull}(S1)) + \text{Area}(\text{ConvexHull}(S2))}{\text{Area}(S)}$$

When this is equal to one both shapes are convex. If several cuts are equivalent test them for roundness. If the resulting shapes S1 or S2 has remaining notches larger than the threshold, continue dividing this shape in the same way until no threshold violations are detected.

2.3 Search algorithms

To approach the scanning area one has to consider a number of factors. For instance, area scanning algorithms developed for use with ground moving units will not need to consider that the vehicle will slip off course when turning. However an aircraft will slip and depending on the type of aircraft and how steep the turn is it will slip a varying amount that needs to be considered. Of course, there are aircrafts that can turn on the spot without slipping, namely rotating wing aircrafts, but they are special cases. Next, we describe different search algorithms while bearing issues that might occur for air units in mind.

Here are definitions of a few variables that will be used in algorithms below.

$$\begin{aligned}
 \text{Camera field of view} &= \alpha_{\text{fov}} \\
 \text{Shape } S &= \{x_1, y_1; x_2, y_2; \dots; x_i, y_i\} \\
 \text{Shape width } W_{\text{shape}} &= \max(x \in S) - \min(x \in S) \\
 \text{Shape height } H_{\text{shape}} &= \max(y \in S) - \min(y \in S) \\
 \text{Camera coverage width } W_{\text{camera}} &= \text{altitude} \cdot \frac{\sin(\alpha_{\text{fov}})}{2} \\
 \text{Camera coverage height } H_{\text{camera}} &= \frac{W_{\text{camera}}}{\text{Camera ratio}}
 \end{aligned}$$

2.3.1 Spiral

The idea of this search method is to start at the outer perimeter of the area and work inwards in a spiral pattern, for example see Figure 2.11

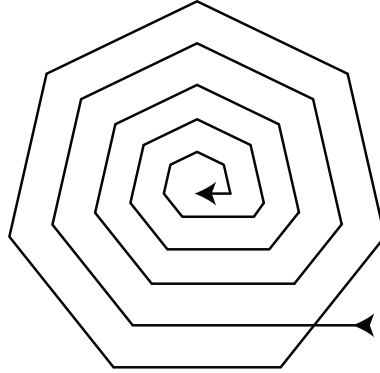


Figure 2.11. Basic spiral search pattern

Such algorithms can be divided into sub-algorithms that are illustrated in Algorithms 1 and 2.

Algorithm 1 Spiral search layers

1. Place waypoints inside the search area, one for each vertex, at a distance of camera coverage width (W_{camera}).
 2. Check if any of the waypoints needs to be removed or merged because they are too close to each other.
 3. Place a new hull inside the waypoints, where it is known that outside this hull, everything has been searched already.
 4. Test if the area inside the new hull is small enough to consider the search complete.
-

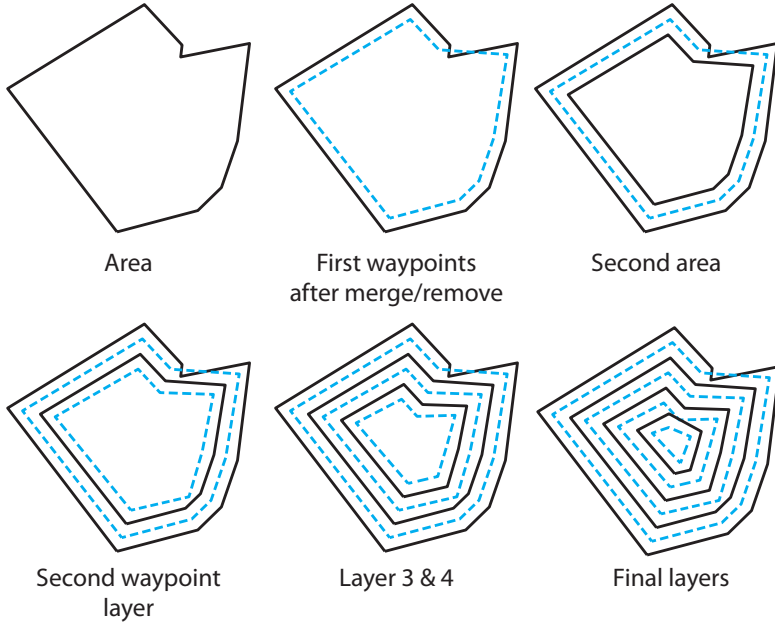


Figure 2.12. Steps used when planning a spiral search

When this is done, in order to finish the spiral search, connect the generated waypoints as in Algorithm 2.

Algorithm 2 Connect waypoint route

1. Take all the waypoint layers (solid lines) and shorten the last waypoint line by the camera coverage height (H_{camera}).
 2. Connect the end of a waypoint layer with the beginning of the next waypoint layer.
-

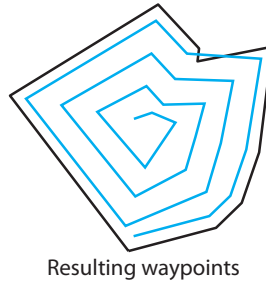


Figure 2.13. The resulting route

2.3.2 Back and forth

In this method the search is done by going back and forth, turning and going back in parallel, see Figure 2.14

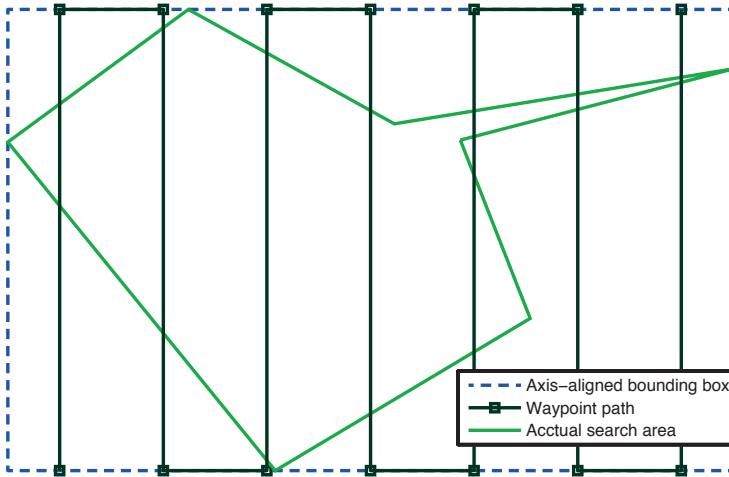


Figure 2.14. The most basic back and forth search

In order to place waypoints in this way the following steps are to be taken.

Algorithm 3 Basic layout

1. Get the axis-aligned bounding box for the shape by getting the minimum and maximum height and width. The dashed line in Figure 2.14.
2. See which corner the UAVs starting point is closest to. The waypoints will start at this corner.
3. Calculate how many rounds (n_r) are needed, if searching up and down. For side to side search the height would be used instead.

$$n_r = \left\lceil \frac{W_{shape}}{W_{camera} \cdot 2} \right\rceil$$

4. Calculate how far apart each run will be, i.e., calculate $W_{round} = \frac{W_{shape}}{n_r}$. This will yield a bit of overlap when the shape is not an exact multiple of W_{camera} .
 5. Place the first two waypoints on a line that is placed W_{round} from the edge of the bounding box.
 6. Place the rest of the waypoints evenly spaced $2 \cdot W_{round}$ from each other.
 7. Connect the waypoints.
-

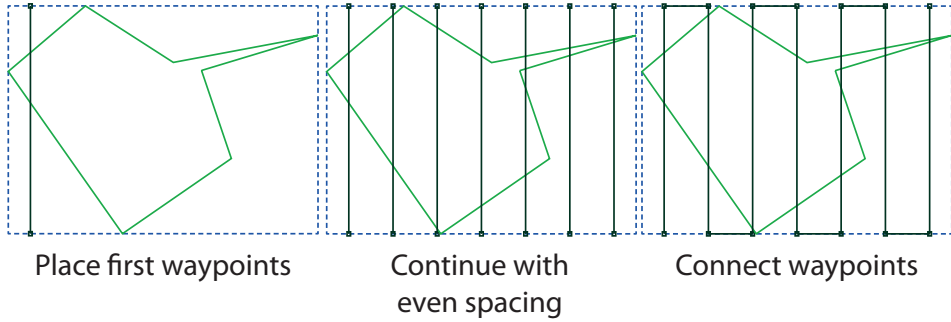


Figure 2.15. Illustration of the main steps of Algorithm 3

The basic layout is now complete. The search path has a lot of overhead so far. Here some post processing will be needed to shorten the flight path without losing coverage. The idea is to cut the waypoints as close to the shape as possible without losing coverage. If waypoints were cut precisely at the shapes borders coverage could be lost in some situations. This is illustrated as the dashed area in Figure 2.16 and explained below.

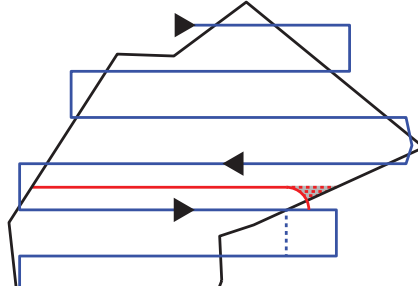


Figure 2.16. Illustration of area that would not be searched if turn is placed on area border. Turns have 90 degree angle restriction imposed on them, explained below.

As can be seen in Figure 2.16 everything above the red line is being searched on the aircraft's way to the left. The dashed grey area is coverage that would have been lost assuming the UAV had made a right turn standing still and going along the vertical dashed line. For simplicity in this illustration it is assumed that $H_{camera} = 0$.

To avoid losing the coverage in Figure 2.16 the following steps are to be taken and are illustrated in Figure 2.17.

Algorithm 4 Increasing coverage

1. Cut the horizontal waypoint lines at the shapes edges.
 2. Insert two virtual lines at distance W_{round} above and below each waypoint line and cut these at the shape edges.
 3. Compare the left side of a waypoint line with the left side of the virtual lines above and below to see which is furthest to the left.
 4. If any of the virtual lines left parts are further to the left than the waypoints. Move the waypoint to the leftmost point.
 5. Compare the right side of a waypoint line with the right side of the virtual lines above and below to see which is furthest to the right.
 6. If any of the virtual lines right parts are further to the right than the waypoints. Move the waypoint to the rightmost point.
-

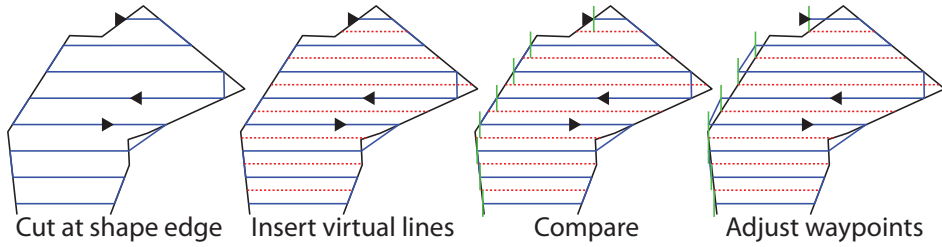


Figure 2.17. Illustration of the steps in Algorithm 4 applied only to the left side to avoid clutter.

This is done in the same way for all waypoint lines. So far we have only considered the width of the cameras field of view, keep in mind that the height also affects the field of view. This can be used to reduce the length of the route without losing coverage.

If we shorten the horizontal waypoint lines a bit the forward part of the sensors field of view will still cover that part. To shorten the route as much as possible simply shorten all waypoint lines by H_{camera} . This is used in future examples. The most visible example is in Figure 2.18 in the "Without turn penalty" subfigure. In this figure many of the waypoints are inside the search area.

At this point, in Figure 2.16, it is also desirable to impose a 90 degree turn restriction. This is imposed in order to create a predictable turn characteristic for a specified aircraft. In other words, if all turns are near 90 degrees, then the behavior of the aircraft will be more predictable.

Predictability is highly desirable and therefore we will make sure that all turns are 90-degrees. This is done by examining all waypoints that have corners, i.e. all but the first and last one. Algorithm 5 describes this procedure.

Algorithm 5 90-degree angle restriction

Place all waypoints in the matrix waypoints with their orthonormal values for position in the x- and y-direction. Let i be the index of the current waypoint being examined, α_i the inner angle for waypoint i and k the number of waypoints.

```

for ( $i = 2$ ;  $i < k - 1$ ;  $i = i + 2$ )
  if  $\alpha_i \leq 90$  then
    waypoints( $i+1, x$ ) = waypoints( $i, x$ )
  else
    waypoints( $i, x$ ) = waypoints( $i+1, x$ )
  end if

```

end for

This gives the waypoints in a 180 degree turn the same position along the x-axis and always moving the waypoints out from and never into the middle of the shape to not lose coverage.

When this is done, the shape is probably fully covered. However, one potential risk remains. If a vertex with a small inner angle wind up in the middle of two

waypoints there is a risk of loosing coverage near that vertex. This can be seen to the far right in Figure 2.16. The following algorithm fixes this glitch.

Algorithm 6 Secure coverage

Let d_i denote the distance to waypoint line i from vertex v and let k be the number of waypoint lines.

\forall vertices v :

if $\min \{d_1, d_2, \dots, d_k\} > W_{camera}$ **then**

$j = i : d_i = \min \{d_1, d_2, \dots, d_k\}$

Insert new waypoint between waypoints that make up waypoint line j .

Place at distance W_{camera} from v in direction $\frac{\alpha_v}{2}$.

end if

2.3.3 Shortening path by trying different rotations

In order to find the shortest possible path or the path that has a good combination of short path and few turns, we propose the following algorithm that will be applied systematically at different rotations between 0 and 180°.

Algorithm 7 Side to side algorithm at rotation θ

1. Rotate shape by $-\theta$
 2. Apply the side to side algorithms, i.e., Algorithms 3 through 6.
 3. Rotate shape and waypoints back again by rotating by θ
-

The length of the route will change depending on the rotation. This changes both in a continuous and a discrete manner; the discrete manner occurs when the rotation either result in a change in the number of turns or if a waypoint moves from one side of a notch to the other.

In some UAV applications turning is highly undesirable due to, for example, very large turn radii. For these cases a turning penalty is also introduced.

$$\text{Route cost} = \text{route length} + \text{turn penalty} \cdot \text{number of turns} \quad (2.6)$$

In this case an operator can set the turn penalty to zero if operating a very agile vehicle or to something very high so that the most important thing is turns being as few as possible.

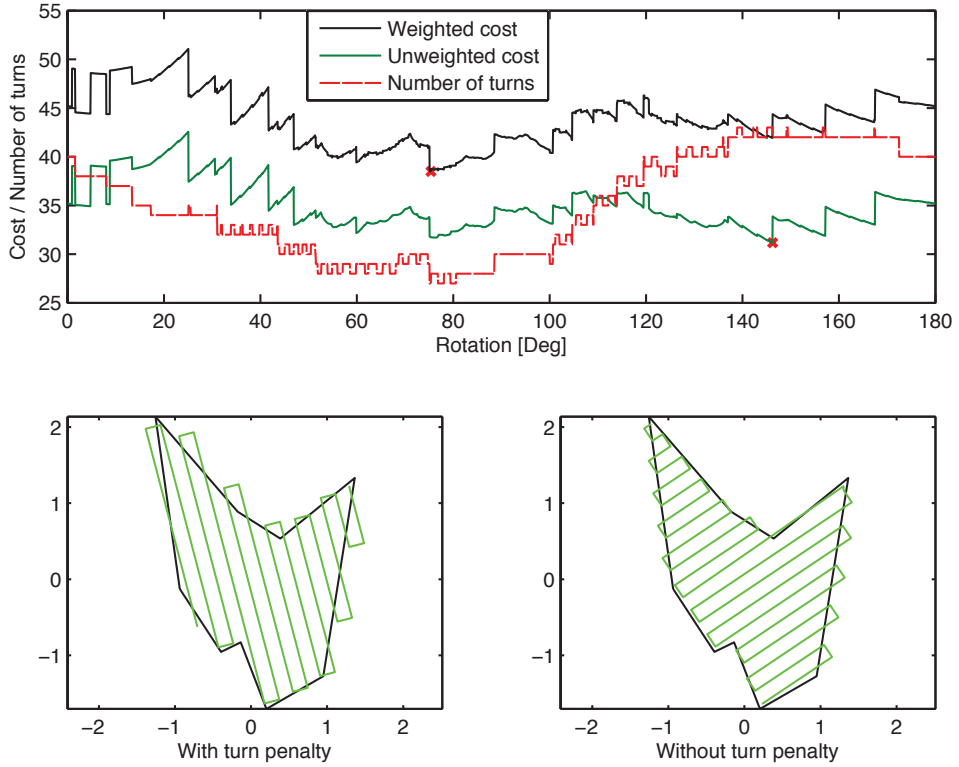


Figure 2.18. Comparison of route cost with and without turn penalty at different rotations. Turn penalty set to W_{camera} . Cost minima marked with X

The objective here is to minimize the cost of searching the area. Let $C_{route}(\theta)$ be the cost of the route at rotation θ according to (2.6).

$$\min_{\theta \in [0, 180]} C_{route}(\theta) \quad (2.7)$$

Example of minimum $C_{route}(\theta)$ with and without turn penalty is illustrated in Figure 2.18.

Chapter 3

Implementation

3.1 Matlab

All development, tests and implementation of the algorithms took place in Matlab. An objective-oriented environment was implemented to handle the search areas. These objects contained the shape and also a left and a right child node to be used when decomposing shapes. The algorithms used for decomposing shapes were implemented recursively.

3.2 Commercial Software

An implementation of the back and forth algorithm was made in ICSs GCS software SkyView. The proposed algorithms were implemented in *C#* and they also relied on built in functionality of the ICS framework used for all their products. The built in functionality included coordinate conversions, clickable maps to enter search areas, arrays for waypoint lists and such, that made the development quicker. The ICS framework was used for interaction with and drawing on, the map. The goal of this implementation was for it to be simple to use and still configurable.

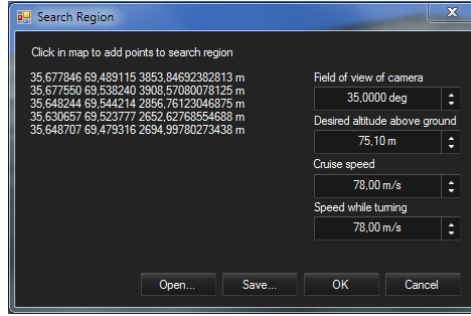


Figure 3.1. Dialog box of search area in SkyView



Figure 3.2. Aircraft and clicked area in SkyView

Configurable options are field of view of the camera fitted in the UAV and at what height the UAV is supposed to fly above the ground level. From this input W_{camera} is calculated. The user is responsible for making sure that the ground resolution of the scan is satisfactory. In this dialog box an overlap variable could also be added to enable coverage of the same area from two or more angles. This implementation is being used in the current version of SkyView. Since this implementation was done in mid development it lacks a few of the features developed in the matlab version e.g. 90 degree angle restriction.

Chapter 4

Results and discussion

4.1 Performance

To test how well the different methods are working. 3506 simulations were made in Matlab with different combinations of search algorithms and area decomposition. The areas had on average nine vertices which I found was about average when choosing a reasonable area in a map. From now on we will abbreviate Approximate Convex Decomposition with ACD.

4.1.1 Methods

The methods that were tested in simulations are listed below.

1. Spiral search on the shape without decomposition
2. Spiral search on the convex hull of the shape without decomposition
3. Spiral search on the shape using ACD and prioritizing roundness
4. Side to side search on the shape using ACD and prioritizing roundness
5. Using the shortest of spiral and side to side on each area in the decomposed area using ACD and prioritizing roundness
6. Spiral search on the shape using ACD and prioritizing distance
7. Side to side search on the shape using ACD and prioritizing distance
8. Using the shortest of spiral and side to side on each area in the decomposed area using ACD and prioritizing distance
9. Side to side search on the shape without decomposition

4.1.2 Simulation results

The success rates of different methods are presented in Table 4.1 and are illustrated in Figure 4.1. Sometimes different methods yield the same result. These cases are considered sharing their solutions with other methods.

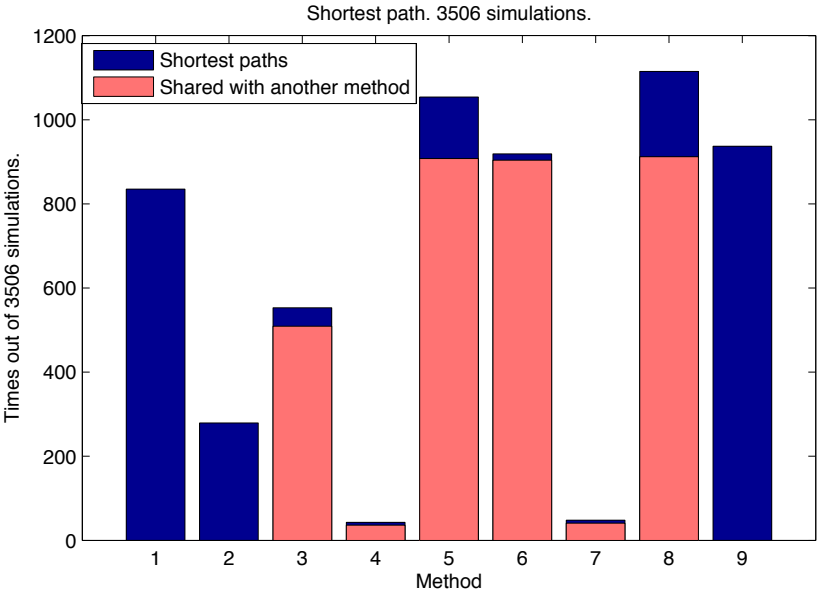


Figure 4.1. Results of 3506 simulations

Method	1	2	3	4	5	6	7	8	9
Shortest paths	835	279	553	43	1054	919	48	1115	937
Shared	0	0	509	36	908	904	41	912	0

Table 4.1. Figure 4.1 in numbers

Method	3	4	5	6	7	8
3	0	0	813	610	0	610
4	0	0	88	77	128	77
5	813	88	0	747	0	749
6	610	77	747	0	0	186
7	0	128	0	0	0	29
8	610	77	749	186	29	0

Table 4.2. Methods sharing the same solution.

Table 4.2 gives a more detailed look at which methods share solutions with each other. As an example, method 3 shared solutions with method 5 a total of 813 times out of the 3506 simulations.

4.1.3 Simulation time consumption

Since the side to side method is one that brute force calculates all different rotations with at selected resolution it takes a bit of time. On my mid 2009 MacBook Pro it took about eight seconds to calculate with an initial resolution of 1 degree. With 0.1 degree initial resolution it took about 47 seconds. The non linearity stems from the search refinement in which the best rotation is searched in greater detail. This would yield, with 1 degree initial resolution, 180 routes to be calculated + 200 for refinement and with 0.1 degrees initial resolution 1800 routes + 200 for refinement. All calculations were done on a single core. Using additional cores, perhaps even GPU processing power, could yield significant speed advantages. Spiral search took about 1/100th of a second.

4.1.4 Result discussion

Because of the nature of the methods, the hybrid methods, five and eight, share a lot of solutions with methods three, four and six, seven respectively. This is due to the fact that if pure spiral produces the shortest path, the hybrid methods can always also use pure spiral.

The few times when methods four and seven are uniquely shortest it is a result of how the different decomposed areas were stitched together. In these cases the hybrid methods can be less effective.

Sometimes prioritizing distance in ACD yields the same decomposition as in the case of prioritizing roundness. In these cases multiple methods will yield the same result.

It is interesting how many unique results are produce by methods 5 and 8. Unique results in methods 5 and 8 means that mixing spiral for some of the decomposed areas with side to side for others yielded the shortest path.

To be able to identify potential problems in the different methods a figure with the result from each simulation was saved. One example of a figure of this kind is Figure 4.2 which at the top left identifies a potential problem with spiral search.

In these figures method one is presented on top left and then from left to right down to method nine at bottom right.

According to the results method one is often a good choice. However as can be seen in Figure 4.2 the coverage achieved with method one is quite bad in this case. If the area is not fully searched the path should be disqualified since the number one priority is coverage.

This is a consequence of the termination condition in the spiral search method. It terminates when the area inside the waypoints is small enough and because of concavity the layers start to self-intersect. The self intersecting area adds a negative number to the total area, making the remaining areas seem smaller to the algorithm.

Method one also often runs in to trouble with getting stuck in the same track as can be seen in Figure 4.3. This problem only occurs when spiral search is forced to handle concave shapes. It occurred so often that an additional termination condition was added to stop when the waypoint path was unreasonably long.

Tweaking the area termination condition proved to be quite difficult. If it is set too small then unnecessary self intersection of the waypoint path will occur in the final layers. If it is set to too large values coverage will be lost because the final layer will not cover the middle of the shape. The final setting was tuned to succeed with convex shapes.

Method nine is not always the shortest but when looking through the images it never fails in a really bad way. It is also the one that, on average, generates the shortest paths as can be seen in Table 4.3. This table was created by adding all lengths from all methods in the 3506 simulations together and making the shortest one the norm.

Method	1	2	3	4	5	6	7	8	9
% longer	166.52	16.09	10.03	7.74	0.76	5.36	6.95	0.24	0

Table 4.3. Average length difference. Method nine is base line.

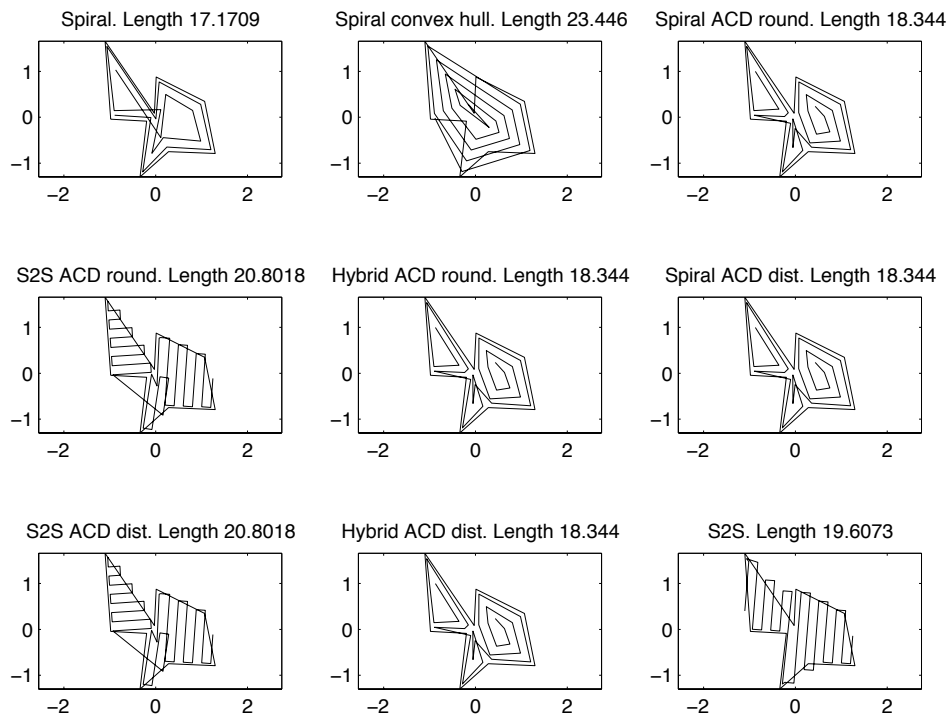


Figure 4.2. Method one generating the shortest path. Not covering the shape properly.

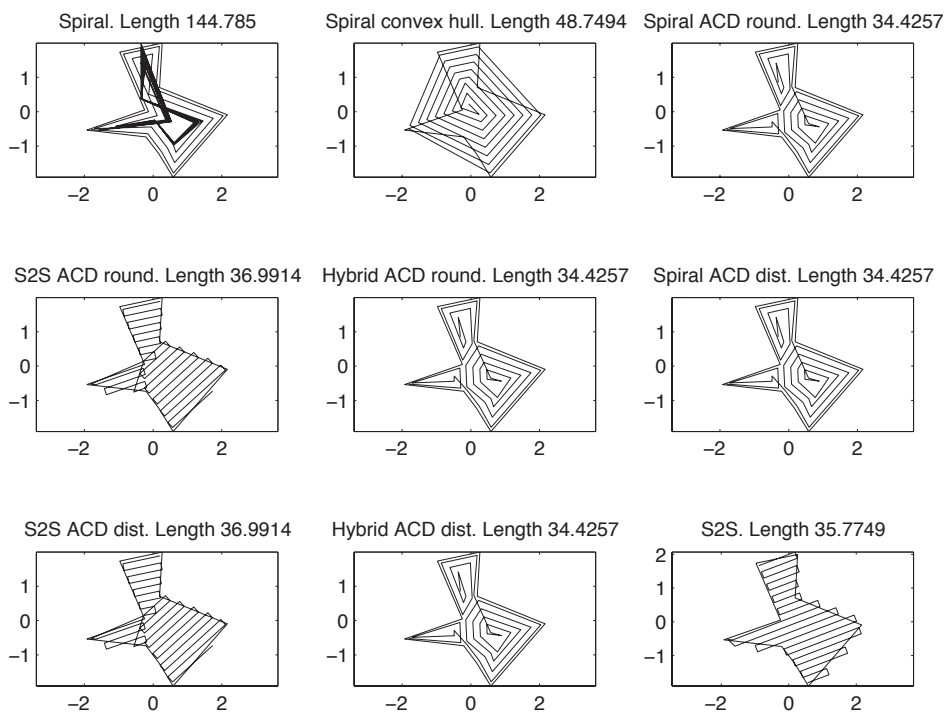


Figure 4.3. Method one running into a self oscillation problem.

Chapter 5

Conclusion

There are quite a few things one can learn from these simulations.

5.1 Approximate convex decomposition

5.1.1 Pros

- The main concept of the algorithms is quite easy to understand. Decompose shape into more convex shapes that are therefore easier to search.
- The main advantage for this type of decomposition is when dealing with areas that have a lot of big concavities like star shaped or hour glass shaped areas, see Figure 5.1.
- The provided decompositions give more control to the user since he/she can choose how much concavity is acceptable depending on what type of aircraft is used.
- They only generate a small amount flight over areas that are not supposed to be searched.

5.1.2 Cons

- ACD can easily cause self intersection when stitching the areas together, in fact it nearly always does.
- It is hard to set a good threshold for concavity, a lot of trial and error involved.
- Decompositions can contain more vertices with quite small inner angles.
- It seldom has natural points to connect different decompositions, i.e., where the path in one shape ends it is not natural to begin in the next one.

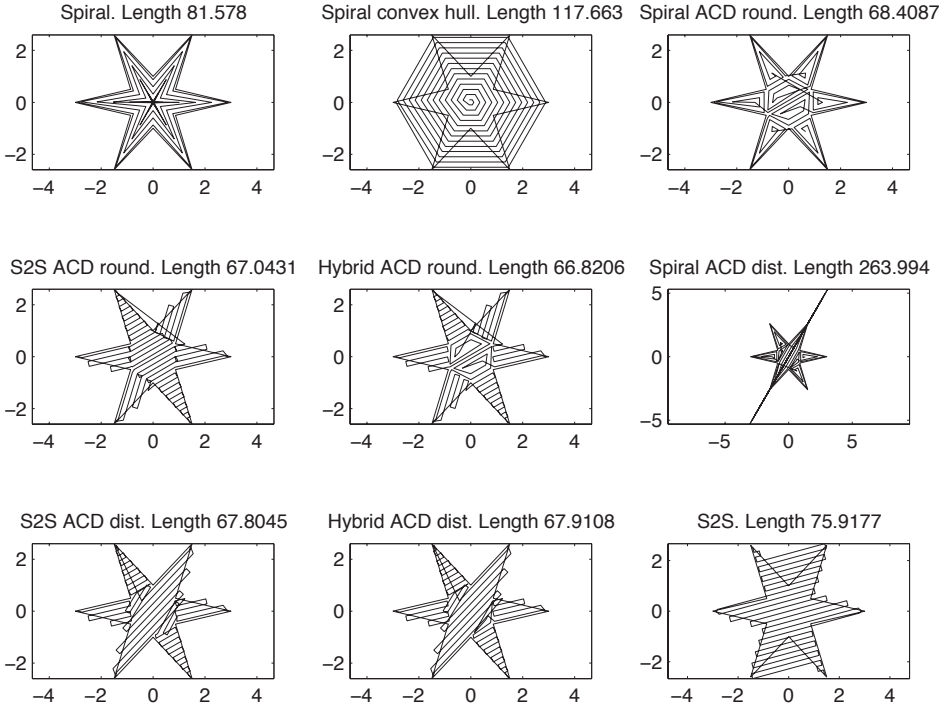


Figure 5.1. Case where ACD yields shorter solutions, although not prettier.

5.2 Search algorithms

5.2.1 Spiral method

Pros

- This method can produce shorter flightpaths that have vertices with large inner angles if the area to be searched is of round character. This can be seen in Figure 5.1 where method two searches the convex hull and generates a very good flight path.

Cons

- Produces very steep turns and difficulty in determining when to remove excess waypoints.
- The algorithm can get stuck and not finish properly. See Figures 4.3 for method one and Figure 5.1 for method six.
- Sometimes finishes before the whole area is properly scanned, see Figure 4.2.

- It is hard to calibrate the termination conditions such that it would suit different types of areas.

5.2.2 Side to side method

Pros

- It is reliable, does not miss parts of areas and does not get stuck.
- When 90-degree angle restriction is applied the behavior of the UAV will be predictable.
- Usually generate short paths and when not shortest usually quite close to the shortest one.
- Can handle complex shapes without missing coverage even though this was not a demand.

Cons

- This method generates longer paths than the spiral method when search areas are of round character.
- Often yields a lot of self intersecting paths when used in conjunction with ACD. This is because of the 90-degree angle restriction.

5.3 All in all

When looking at the results all in all and after seeing simulated flights of these different ways of searching it must be said that the path generated by the side to side algorithm without using ACD is a clear overall winner. Sure other combos have strengths but side to side is never really bad.

Also other than for paths being short, they look good to an operator and the operator will quickly get a feel for the way the aircraft will handle flying along the path. That includes a natural feel for how long the flight will take, this can also be quite accurately calculated of course. The feeling for the aircrafts behavior is easily acquired because the turns are always made with the wind blowing from the same sides. When the first four turns are made the rest should look quite similar to those, the exception being strong turbulence, presuming the direction and speed of the wind is constant during flight.

While performing the flight simulations with the different methods, the uncertainty of the other methods give a feeling of uncertainty about how the aircraft will perform in different parts of the path.

Spiral search will often generate turns of different steepnesses and these will be to and from a lot of different headings. This unfortunately makes it harder for an operator to predict how the aircraft will perform in different parts of the search path. The unreliability, stemming from the termination criterias in the spiral search algorithm, are also a bit off-putting.

When using ACD the path can sometimes be shortened but the flight path will often, however not always, look less natural. The meaning of natural here is "how a human would do it".

Chapter 6

Future work

There are a lot of different parts in this thesis that could be expanded and studied further. Here are a few of the more time consuming ones that I have thought of during the process.

6.1 Decomposition

Regarding the decomposition methods, one must figure out ways to optimize the waypoint layout in the decomposed areas as to avoid self intersection and double scanning. This will probably include looking at which parts are adjacent to other areas and which are facing outwards.

Better patching of areas waypoints. By looking at all start and end points of all areas an optimization can be devised to shorten the path via Dijkstras algorithm, for example see [7]. Currently algorithm looks at pairs of adjacent areas.

6.2 Search algorithms

Development of algorithms that can handle restricted areas inside the search area could be very useful if looking for a person in a forrest but not wanting to look for him/her in the lake in the forrest. This lake could then be added as a restricted area inside the search area.

6.2.1 Spiral

Finding dynamic criteria that can determine when to use the convex hull and when not to is also of interest. These criteria should focus on flyability for a specified aircraft.

In this case one should look for a better way to determine what parts of a shape have already been searched and which parts have not. For instance spiral paths in Figure 5.1 could have been much shorter and could have been achieved by making

a grid and implementing a shape covering algorithm.

Also it is important to determine when to abandon spiral search for instance if a small "fish hook" is sticking out from main shape. This would have to be dynamic to consider type of aircraft and how to handle the anomaly.

It would also be desirable to eliminate self intersection. Self intersection occurs, sometimes as a result of pointy parts of a shape, see Figure 5.1, and sometimes because one part of the shape is narrower than the rest, as in shapes like an hour glass. In the case of pointy shapes, a handler needs to be developed so that different layers are not allowed to intersect. Intersection in this case is always avoidable. In the hour glass shape case there is no good way of doing a straight up spiral. Here ACD is a good idea, see method six in Figure 6.1.

To ensure complete coverage better termination conditions could also be developed. This would also be solved by implementing the grid mentioned above.

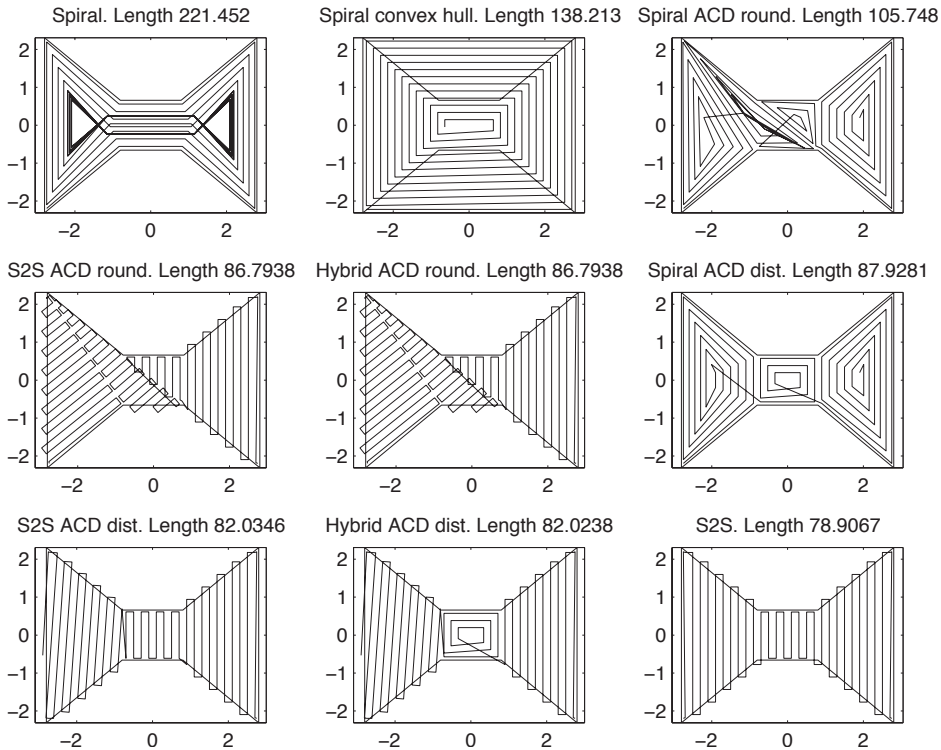


Figure 6.1. Search of an hour glass shape. Method one shows bad handling.

6.2.2 Side to side

The biggest flaw with this search algorithm is that it is a brute force search-all-possible-rotations algorithm. This is very time consuming and is dependent on the resolution. For example if using one tenth degree resolution, that means, first 1800 waypoint routes has to be generated. Then lets say a rotation of 137 gave the shortest route. The span of 136 to 138 is then searched with a resolution of 1/100th of a degree to find the best one and this adds 200 more waypoint routes to be generated. This means that to calculate the shortest side to side route 2000 waypoint routes are actually calculated.

Here it would be quite desirable to be able to analyze the shape and at least avoid searching the rotations that will not yield good results. Most desirable would be to analytically find the optimum rotation. Ceiling and floor functions in the algorithm makes this a piece wise continuous function of rotation. When adding or removing turns the length will change discretely as can be seen in Figure 2.18.

6.3 Implementation

Since leaving ICS in the beginning of summer 2011 the algorithms of the side to side algorithm has been refined and now perform excellent in all conditions. The algorithms in SkyView could therefore be updated to include the new refinements.

Bibliography

- [1] P. Bourke. Calculating the area and centroid of a polygon.
<http://paulbourke.net/geometry/polyarea/>, 1988.
- [2] Wikimedia Commons. Complex polygon graphic.
http://commons.wikimedia.org/wiki/File:Complex_polygon.svg.
- [3] Wikimedia Commons. Concave polygon graphic.
http://commons.wikimedia.org/wiki/File:Simple_polygon.svg.
- [4] Wikimedia Commons. Convex polygon graphic.
http://commons.wikimedia.org/wiki/File:Regular_heptagon.svg.
- [5] Wikimedia Commons. Earth coordinates graphic.
http://en.wikipedia.org/wiki/File:Geographic_coordinates_sphere.svg.
- [6] H. S. M. Coxeter. *Regular Complex Polytopes*.
Cambridge University Press, 1 edition, 1974. ISBN 97-80-52139490-1.
- [7] E. W. Dijkstra. A note on two problems in connexion with graphs.
Numerische Mathematik, 1:269–271, 1959.
- [8] E. M. Arkin, M. A. Bender, E. D. Demaine. et al.
Optimal covering tours with turn costs. In *13th ACM-SIAM sympos. discrete algorithms*, pages 138–147, 2005.
- [9] R. Ayani F. Kamrani. UAV Path Planning in Search Operations. Technical report, Royal Institute of Technology (KTH), Sweden, 2009.
http://www.intechopen.com/source/pdfs/5978/InTech-Uav_path_planning_in_search_operations.pdf
ISBN 978-953-7619-41-1.
- [10] H. Peng, F. Su, Y. Bu et al. Cooperative area search for multiple UAVs based on RRT and decentralized receding horizon optimization. In *Asian Control Conference, 2009. ASCC 2009. 7th*, aug. 2009.
- [11] N. M. Amato J. Lien. Approximate Convex Decomposition of Polygons. Technical report, Texas A & M University, 2004.
<http://cs.gmu.edu/~jmlien/research/app-cd/p218-lien.pdf>.

- [12] M. Mitchell R. Juengling. Combinatorial Shape Decomposition. Technical report, Portland State University, 2007.
<http://web.cecs.pdx.edu/~mm/CombinatorialShapeDecomposition.pdf>.
- [13] I. Ragnemalm. *Polygons feel no pain*. Studentlitteratur, 2 edition, 2009.
<http://computer-graphics.se/TSBK07-files/literature.html>
ISBN is missing in book.
- [14] S. J. Julier S. Waharte and N. Trigoni. Coordinated Search with a Swarm of UAVs. Technical report, University of Oxford and University College London, 2009. <http://www.cs.ox.ac.uk/files/2307/report.pdf>.
- [15] G. Sparr. *Linjär algebra*. Studentlitteratur AB, 2 edition, 1997.
ISBN 9789144197524.
- [16] A.M. Bruckstein Y. Altshuler, V. Yanovski and I.A. Wagner. Efficient Cooperative Search of Smart Targets Using UAV Swarms.
Robotica, 26:551–557, 2008.