Robot Online Algorithms in Computational Geometry: Searching and Exploration in the Plane

Subir Kumar Ghosh\textsuperscript{1}

Department of Computer Science
School of Mathematical Sciences
Ramakrishna Mission Vivekananda University
Belur Math, Howrah 711202, West Bengal, India
ghosh@tifr.res.in

\textsuperscript{1}Ex-Professor, School of Technology & Computer Science, Tata Institute of Fundamental Research, Mumbai 400005, India
Overview of the Lecture

1. Off-line and online algorithms.
Overview of the Lecture

1. Off-line and online algorithms.
2. Efficiency of online algorithms.
Overview of the Lecture

1. Off-line and online algorithms.
2. Efficiency of online algorithms.
Overview of the Lecture

1. Off-line and online algorithms.
2. Efficiency of online algorithms.
Overview of the Lecture

1. Off-line and online algorithms.
2. Efficiency of online algorithms.
5. Continuous and discrete visibility.
7. Searching for a target in an unknown star-shaped polygon.
1. Off-line and online algorithms.
2. Efficiency of online algorithms.
5. Continuous and discrete visibility.
Overview of the Lecture

1. Off-line and online algorithms.
2. Efficiency of online algorithms.
5. Continuous and discrete visibility.
7. Searching for a target in an unknown star-shaped polygon.
Overview of the Lecture

1. Off-line and online algorithms.
2. Efficiency of online algorithms.
5. Continuous and discrete visibility.
7. Searching for a target in an unknown star-shaped polygon.
Overview of the Lecture

1. Off-line and online algorithms.
2. Efficiency of online algorithms.
5. Continuous and discrete visibility.
7. Searching for a target in an unknown star-shaped polygon.
Overview of the Lecture

1. Off-line and online algorithms.
2. Efficiency of online algorithms.
5. Continuous and discrete visibility.
7. Searching for a target in an unknown star-shaped polygon.
Overview of the Lecture

1. Off-line and online algorithms.
2. Efficiency of online algorithms.
5. Continuous and discrete visibility.
7. Searching for a target in an unknown star-shaped polygon.
Survey

Online algorithms for searching and exploration in the plane

Subir Kumar Ghosh\textsuperscript{a,*}, Rolf Klein\textsuperscript{b}

\textsuperscript{a} School of Computer Science, Tata Institute of Fundamental Research, Mumbai 400005, India
\textsuperscript{b} Institute of Computer Science I, University of Bonn, Bonn 53117, Germany

\textbf{Abstract}

In this paper, we survey online algorithms in computational geometry that have been designed for mobile robots for searching a target and for exploring a region in the plane.

© 2010 Elsevier Inc. All rights reserved.
Starting from $s$, a point robot is searching for the point $t$ in $R$. 
Starting from $s$, a point robot is searching for the point $t$ in $R$. If the robot has the complete geometric information (or map) of $R$ and also knows the exact location of $t$, then the robot can choose a path inside $R$ to move from $s$ to $t$. 
Starting from $s$, a point robot is searching for the point $t$ in $R$.

If the robot has the complete geometric information (or map) of $R$ and also knows the exact location of $t$, then the robot can choose a path inside $R$ to move from $s$ to $t$.

In many situations, it is expected that the robot follows the Euclidean shortest path from $s$ to $t$ inside $R$. 
Offline Algorithms

Starting from $s$, a point robot is searching for the point $t$ in $R$.

If the robot has the complete geometric information (or map) of $R$ and also knows the exact location of $t$, then the robot can choose a path inside $R$ to move from $s$ to $t$.

In many situations, it is expected that the robot follows the Euclidean shortest path from $s$ to $t$ inside $R$.

In some situation, the robot may be asked to follow a minimum link (or, turn) path from $s$ to $t$ inside $R$. 
There are known efficient sequential algorithms for computing such paths.
There are known efficient sequential algorithms for computing such paths.

Thus, the robot can compute an optimal path, depending upon the optimization criteria, using its on-board computer system and then follows the path from $s$ to $t$. 


There are known efficient sequential algorithms for computing such paths.
Thus, the robot can compute an optimal path, depending upon the optimization criteria, using its on-board computer system and then follows the path from $s$ to $t$.
Such algorithms are called *offline algorithms* of a robot path planning for a target searching problem in a known environment.
There are known efficient sequential algorithms for computing such paths. Thus, the robot can compute an optimal path, depending upon the optimization criteria, using its on-board computer system and then follows the path from $s$ to $t$.

Such algorithms are called offline algorithms of a robot path planning for a target searching problem in a known environment.

Suppose, a robot does not have the complete knowledge of the geometry of \( R \) apriori.
Suppose, a robot does not have the complete knowledge of the geometry of $R$ apriori.

The robot also does not know the location of the target $t$, but the target can be recognized by the robot.
Robot Searching Problem in an Unknown Environment

- Suppose, a robot does not have the complete knowledge of the geometry of $R$ apriori.
- The robot also does not know the location of the target $t$, but the target can be recognized by the robot.
- In such a situation, the robot is asked to reach $t$ from its starting position $s$ using its sensory input provided by acoustic, visual, or tactile sensors of its on-board sensor system.
Robot Searching Problem in an Unknown Environment

- Suppose, a robot does not have the complete knowledge of the geometry of $R$ apriori.
- The robot also does not know the location of the target $t$, but the target can be recognized by the robot.
- In such a situation, the robot is asked to reach $t$ from its starting position $s$ using its sensory input provided by acoustic, visual, or tactile sensors of its on-board sensor system.
- The problem here is to design an efficient *online algorithm* which a robot can use to search for the target $t$. 
Suppose, a robot does not have the complete knowledge of the geometry of $R$ apriori.

The robot also does not know the location of the target $t$, but the target can be recognized by the robot.

In such a situation, the robot is asked to reach $t$ from its starting position $s$ using its sensory input provided by acoustic, visual, or tactile sensors of its on-board sensor system.

The problem here is to design an efficient online algorithm which a robot can use to search for the target $t$.

Observe that any such algorithm is ‘online’ in the sense that decisions must be made based only on what the robot has received input so far from its sensor system.
Robot Searching Problem in an Unknown Environment

- Suppose, a robot does not have the complete knowledge of the geometry of $R$ apriori.
- The robot also does not know the location of the target $t$, but the target can be recognized by the robot.
- In such a situation, the robot is asked to reach $t$ from its starting position $s$ using its sensory input provided by acoustic, visual, or tactile sensors of its on-board sensor system.
- The problem here is to design an efficient *online algorithm* which a robot can use to search for the target $t$.
- Observe that any such algorithm is ‘online’ in the sense that decisions must be made based only on what the robot has received input so far from its sensor system.
- The algorithms for these types of online searching problems in an unknown environment are known as *robot online algorithms*. 
Imagine that a robot is to explore the interior of a collapsed building, which has crumbled due to an earthquake, in order to search for human survivors.
Imagine that a robot is to explore the interior of a collapsed building, which has crumbled due to an earthquake, in order to search for human survivors.

It is clearly impossible to have knowledge of the building's interior geometry prior to the exploration.
Imagine that a robot is to explore the interior of a collapsed building, which has crumbled due to an earthquake, in order to search for human survivors.

It is clearly impossible to have knowledge of the buildings interior geometry prior to the exploration.

Thus, the robot must be able to see, with its on-board vision sensors, all points in the buildings interior while following its exploration path.
Imagine that a robot is to explore the interior of a collapsed building, which has crumbled due to an earthquake, in order to search for human survivors.

It is clearly impossible to have knowledge of the buildings interior geometry prior to the exploration.

Thus, the robot must be able to see, with its on-board vision sensors, all points in the buildings interior while following its exploration path.

In this way, no potential survivors will be missed by the exploring robot.
Imagine that a robot is to explore the interior of a collapsed building, which has crumbled due to an earthquake, in order to search for human survivors.

It is clearly impossible to have knowledge of the building's interior geometry prior to the exploration.

Thus, the robot must be able to see, with its on-board vision sensors, all points in the building's interior while following its exploration path.

In this way, no potential survivors will be missed by the exploring robot.

The algorithms for these types of online exploration problems in an unknown environment are also known as *robot online algorithms*. 
In our model for robotic exploration, we consider an unknown polygonal environment $R$ (with or without holes) as the search space, and the robot as a moving point in $R$. 
In our model for robotic exploration, we consider an unknown polygonal environment $R$ (with or without holes) as the search space, and the robot as a moving point in $R$.

One of the difficulties in working with incomplete information is that the path cannot be pre-planned and therefore, its global optimality can hardly be achieved.
In our model for robotic exploration, we consider an unknown polygonal environment $R$ (with or without holes) as the search space, and the robot as a moving point in $R$.

One of the difficulties in working with incomplete information is that the path cannot be pre-planned and therefore, its global optimality can hardly be achieved.

Instead, one can judge the online algorithm performance based on how it stands with respect to other existing or theoretically feasible algorithms.
The efficiency of online algorithms for searching and exploration algorithms is generally measured using their competitive ratios.
The efficiency of online algorithms for searching and exploration algorithms is generally measured using their competitive ratios.

\[
\text{Competitive ratio} = \frac{\text{Cost of the online algorithm}}{\text{Cost of an optimal offline algorithm}}
\]
The efficiency of online algorithms for searching and exploration algorithms is generally measured using their competitive ratios.

\[
\text{Competitive ratio} = \frac{\text{Cost of the online algorithm}}{\text{Cost of an optimal offline algorithm}}
\]


Suppose, the target point \( t \) is placed on a line \( L \) in an unknown location.
Suppose, the target point $t$ is placed on a line $L$ in an unknown location.

Starting from a given position $O$ on $L$, the problem is to design an online algorithm for a point robot for locating $t$. 
Searching for a Target on a Line

- Suppose, the target point $t$ is placed on a line $L$ in an unknown location.
- Starting from a given position $O$ on $L$, the problem is to design an online algorithm for a point robot for locating $t$.
- It is assumed that the robot can detect $t$ if it stands on top of $t$ or reaches $t$. 
Suppose, the target point $t$ is placed on a line $L$ in an unknown location.

Starting from a given position $O$ on $L$, the problem is to design an online algorithm for a point robot for locating $t$.

It is assumed that the robot can detect $t$ if it stands on top of $t$ or reaches $t$.

The problem may be viewed as an autonomous robot is facing a very long wall and it wants go to the other side of the wall through a door on the wall but it does not known whether the door is located to the left or right of its current position.
Suppose the robot knows that $t$ is located exactly $d$ distance away from $O$. Then the robot first walks $d$ distance to the left. If $t$ is not found, then the robot returns to $O$ and then walks $d$ distance to the right. So, the competitive ratio of this straightforward on-line algorithm is 3.

What is the competitive ratio of the search if $d$ is not known apriori?
- Suppose the robot knows that \( t \) is located exactly \( d \) distance away from \( O \).
- Then the robot first walks \( d \) distance to the left.
Suppose the robot knows that $t$ is located exactly $d$ distance away from $O$. Then the robot first walks $d$ distance to the left. If $t$ is not found, then the robot returns to $O$ and then walks $d$ distance to the right.
Suppose the robot knows that $t$ is located exactly $d$ distance away from $O$. Then the robot first walks $d$ distance to the left. If $t$ is not found, then the robot returns to $O$ and then walks $d$ distance to the right. So, the competitive ratio of this straightforward on-line algorithm is 3.
Suppose the robot knows that $t$ is located exactly $d$ distance away from $O$.
Then the robot first walks $d$ distance to the left.
If $t$ is not found, then the robot returns to $O$ and then walks $d$ distance to the right.
So, the competitive ratio of this straightforward on-line algorithm is 3.
What is the competitive ratio of the search if $d$ is not known apriori?
The robot walks one unit to the right along $L$. If $t$ is not found, then it returns to its starting point $O$. The process of doubling the length is known as the doubling strategy.
The robot walks one unit to the right along $L$. If $t$ is not found, then it returns to its starting point $O$.

In the next step, the robot walks two units to the left of $O$ along $L$. If $t$ is not found again, the robot returns to $O$.
The robot walks one unit to the right along $L$. If $t$ is not found, then it returns to its starting point $O$.

In the next step, the robot walks two units to the left of $O$ along $L$. If $t$ is not found again, the robot returns to $O$.

In the next step, the robot walks four units to the right along $L$ and if it is again unsuccessful to locate $t$, it returns to $O$.
The robot walks one unit to the right along $L$. If $t$ is not found, then it returns to its starting point $O$.

In the next step, the robot walks two units to the left of $O$ along $L$. If $t$ is not found again, the robot returns to $O$.

In the next step, the robot walks four units to the right along $L$ and if it is again unsuccessful to locate $t$, it returns to $O$.

After some steps, the robot locates $t$. 

The process of doubling the length is known as the **doubling strategy**.
The robot walks one unit to the right along $L$. If $t$ is not found, then it returns to its starting point $O$.

In the next step, the robot walks two units to the left of $O$ along $L$. If $t$ is not found again, the robot returns to $O$.

In the next step, the robot walks four units to the right along $L$ and if it is again unsuccessful to locate $t$, it returns to $O$.

After some steps, the robot locates $t$.

The process of doubling the length is known as *doubling strategy*. 
Assume that $t$ is located at a distance $d$ from the origin on the positive axis.
- Assume that \( t \) is located at a distance \( d \) from the origin on the positive axis.
- Assume that \( 2^{k-1} < d \leq 2^{k+1} \) for some \( k \).
Assume that $t$ is located at a distance $d$ from the origin on the positive axis.

Assume that $2^{k-1} < d \leq 2^{k+1}$ for some $k$.

The total distance traveled during the alternative walk is
\[
(2.1 + 2.1 - 2| + 2.4 + 2.1 - 8| + \ldots + 2.2^{k-1} + 2.1 - 2^k| + d
= 2.2^{k+1} + d).
\]
Assume that $t$ is located at a distance $d$ from the origin on the positive axis.

Assume that $2^{k-1} < d \leq 2^{k+1}$ for some $k$.

The total distance traveled during the alternative walk is

$$(2.1 + 2.1 - 2 + 2.4 + 2.4 - 8 + \ldots + 2.2^{k-1} + 2.4 - 2^k + d = 2.2^{k+1} + d).$$

If the location of $t$ is known apriori, then it is a straight walk of length $d$ from the origin to $t$. 
Assume that \( t \) is located at a distance \( d \) from the origin on the positive axis.

Assume that \( 2^{k-1} < d \leq 2^{k+1} \) for some \( k \).

The total distance traveled during the alternative walk is
\[
(2.1 + 2.1 - 2 + 2.4 + 2.4 - 8 + \ldots + 2.2^{k-1} + 2.4 - 2^k) = 2.2^{k+1} + d.
\]

If the location of \( t \) is known apriori, then it is a straight walk of length \( d \) from the origin to \( t \).

So, the competitive ratio of the alternate walk is
\[
\frac{2.2^{k+1} + d}{d} = 1 + \frac{2.2^{k+1}}{d} \text{ which is at most } 1 + \left( \frac{2.2^{k+1}}{2^{k-1}} \right) = 9.
\]
Searching for a Target on $m$ Rays

A beautiful young cow Ariadne is at the entrance of a simple labyrinth which branches in $m \geq 2$ corridors. She knows that the handsome Minotaur is waiting somewhere in the labyrinth. What is the best searching strategy for Ariadne to locate Minotaur?
Searching for a Target on \( m \) Rays

A beautiful young cow Ariadne is at the entrance of a simple labyrinth which branches in \( m \geq 2 \) corridors. She knows that the handsome Minotaur is waiting somewhere in the labyrinth. What is the best searching strategy for Ariadne to locate Minotaur?

Visit $m \geq 2$ rays in a cyclic order starting with an initial walk of length one.
- Visit $m \geq 2$ rays in a cyclic order starting with an initial walk of length one.
- Increase the length of the walk each time by a factor of $m/(m-1)$ till $t$ is located.
Visit \( m \geq 2 \) rays in a cyclic order starting with an initial walk of length one.
Increase the length of the walk each time by a factor of \( m/(m-1) \) till \( t \) is located.
This strategy gives the competitive ratio of \( 1 + 2m^m/(m-1)^{m-1} \), which is optimal.

- Visit $m \geq 2$ rays in a cyclic order starting with an initial walk of length one.
- Increase the length of the walk each time by a factor of $m/(m - 1)$ till $t$ is located.
- This strategy gives the competitive ratio of $1 + 2m^m/(m - 1)^{m-1}$, which is optimal.


Assume that the point robot knows the exact location of $t$ but does not know the positions of unknown polygonal obstacles $h_1, h_2, \ldots, h_k$. 
Assume that the point robot knows the exact location of $t$ but does not know the positions of unknown polygonal obstacles $h_1, h_2, \ldots, h_k$.

The robot starts from $s$, and moves towards $t$ following the segment $st$ till the robot detects by its tactile sensor that it has hit a polygonal obstacle (say, $h_i$) at some point $u_i$. 
Assume that the point robot knows the exact location of \( t \) but does not know the positions of unknown polygonal obstacles \( h_1, h_2, \ldots, h_k \).

The robot starts from \( s \), and moves towards \( t \) following the segment \( st \) till the robot detects by its tactile sensor that it has hit a polygonal obstacle (say, \( h_i \)) at some point \( u_i \).

Then the robot goes around the boundary of \( h_i \) to locate the boundary point of \( h_i \) (say, \( v_i \)) which is closest to \( t \).
Assume that the point robot knows the exact location of \( t \) but does not know the positions of unknown polygonal obstacles \( h_1, h_2, \ldots, h_k \).

The robot starts from \( s \), and moves towards \( t \) following the segment \( st \) till the robot detects by its tactile sensor that it has hit a polygonal obstacle (say, \( h_i \)) at a some point \( u_i \).

Then the robot goes around the boundary of \( h_i \) to locate the boundary point of \( h_i \) (say, \( v_i \)) which is closest to \( t \).

Then the robots moves from \( u_i \) to \( v_i \) following the shorter of the two paths from \( u_i \) to \( v_i \) along the boundary of \( h_i \).
Assume that the point robot knows the exact location of \( t \) but does not know the positions of unknown polygonal obstacles \( h_1, h_2, \ldots, h_k \).

The robot starts from \( s \), and moves towards \( t \) following the segment \( st \) till the robot detects by its tactile sensor that it has hit a polygonal obstacle (say, \( h_i \)) at a some point \( u_i \).

Then the robot goes around the boundary of \( h_i \) to locate the boundary point of \( h_i \) (say, \( v_i \)) which is closest to \( t \).

Then the robots moves from \( u_i \) to \( v_i \) following the shorter of the two paths from \( u_i \) to \( v_i \) along the boundary of \( h_i \).

Then the robots moves from \( u_i \) to \( v_i \) following the shorter of the two paths from \( u_i \) to \( v_i \) along the boundary of \( h_i \).
Treating $v_i$ as $s$, the robot repeats the same process of moving towards $t$ following the segment $v_i t$ till $t$ is reached.
Treating \( v_i \) as \( s \), the robot repeats the same process of moving towards \( t \) following the segment \( v_i t \) till \( t \) is reached.

The length of the path traversed by the robot is bounded by the length of \( st \) and 1.5 times the perimeters of those polygonal obstacles that are hit by the robot.
Treating $v_i$ as $s$, the robot repeats the same process of moving towards $t$ following the segment $v_i t$ till $t$ is reached.

The length of the path traversed by the robot is bounded by the length of $st$ and 1.5 times the perimeters of those polygonal obstacles that are hit by the robot.

Though examples can be constructed which make the competitive ratio of this online algorithm unbounded, the algorithm is still one of the best algorithms for the general polygonal obstacles in the plane.
Treating $v_i$ as $s$, the robot repeats the same process of moving towards $t$ following the segment $v_i t$ till $t$ is reached.

The length of the path traversed by the robot is bounded by the length of $st$ and 1.5 times the perimeters of those polygonal obstacles that are hit by the robot.

Though examples can be constructed which make the competitive ratio of this online algorithm unbounded, the algorithm is still one of the best algorithms for the general polygonal obstacles in the plane.


Two points $u$ and $v$ of $P$ are said to be visible if the line segment $uv$ lies totally inside $P$. 

---

Two points $u$ and $v$ of $P$ are said to be visible if the line segment $uv$ lies totally inside $P$.

The visibility polygon of $P$ from a point $p$ (denoted as $VP(P, p)$) is the set of all points of $P$ that are visible from $p$. 

References:

Two points $u$ and $v$ of $P$ are said to be \emph{visible} if the line segment $uv$ lies totally inside $P$.

The visibility polygon of $P$ from a point $p$ (denoted as $VP(P, p)$) is the set of all points of $P$ that are visible from $p$.

In other words, for every point $z \in P$, if the line segment joining $z$ and $p$ lies inside $P$, then $z$ belongs to $VP(P, p)$. 

---

Visibility Polygon

Two points $u$ and $v$ of $P$ are said to be *visible* if the line segment $uv$ lies totally inside $P$.

The visibility polygon of $P$ from a point $p$ (denoted as $VP(P, p)$) is the set of all points of $P$ that are visible from $p$.

In other words, for every point $z \in P$, if the line segment joining $z$ and $p$ lies inside $P$, then $z$ belongs to $VP(P, p)$.

If the robot computes visibility polygons from each point on its path, we say that \( P \) is explored under continuous visibility.

If the robot computes visibility polygons from a selected set of points on its path, we say that \( P \) is explored under discrete visibility.
Continuous and Discrete Visibility

- If the robot computes visibility polygons from each point on its path, we say that $P$ is explored under continuous visibility.

- If the robot computes visibility polygons from a selected set of points on its path, we say that $P$ is explored under discrete visibility.
Let $u_1, u_2, \ldots u_{n/4}$ be the nearest points of $s$ in the alleys of a simple polygon $P$ of distance $d$ such that if the robot moves from $s$ to $u_i$ for each $i$, the robot can see the alley completely.
Let $u_1, u_2, \ldots, u_{n/4}$ be the nearest points of $s$ in the alleys of a simple polygon $P$ of distance $d$ such that if the robot moves from $s$ to $u_i$ for each $i$, the robot can see the alley completely.

In order to search $t$, the robot moves from $s$ to $u_i$ in each alley and then returns to $s$ if it does not locate $t$. 
Let $u_1, u_2, \ldots, u_{n/4}$ be the nearest points of $s$ in the alleys of a simple polygon $P$ of distance $d$ such that if the robot moves from $s$ to $u_i$ for each $i$, the robot can see the alley completely.

In order to search $t$, the robot moves from $s$ to $u_i$ in each alley and then returns to $s$ if it does not locate $t$.

For every unsuccessful search, the robot travels $2d$ distance.
In the worst case, the robot locates $t$ in the last alley.
In the worst case, the robot locates \( t \) in the last alley.

So, the total distance travelled by the robot is at least \( 2d(n/4 - 1) + d \).
- In the worst case, the robot locates $t$ in the last alley.
- So, the total distance travelled by the robot is at least $2d(n/4 - 1) + d$.
- Hence, the lower bound of the competitive ratio for this problem is $n/2 - 1$. 

---


Competitive ratio: $2^{n/7}$. 
In the worst case, the robot locates \( t \) in the last alley.

So, the total distance travelled by the robot is at least \( 2d(n/4 - 1) + d \).

Hence, the lower bound of the competitive ratio for this problem is \( n/2 - 1 \).


Searching for a Target in an Unknown Street

A simple polygon $P$ is said to be a street (also called LR-visibility polygon) if there exists two points $s$ and $t$ on the boundary of $P$ such that every point of the clockwise boundary from $s$ to $t$ of $P$ (denoted as $L$) is visible from some point of the counterclockwise boundary of $P$ from $s$ to $t$ (denoted as $R$) and vice versa.
A simple polygon \( P \) is said to be a \textit{street} (also called \textit{LR-visibility polygon}) if there exists two points \( s \) and \( t \) on the boundary of \( P \) such that every point of the clockwise boundary from \( s \) to \( t \) of \( P \) (denoted as \( L \)) is visible from some point of the counterclockwise boundary of \( P \) from \( s \) to \( t \) (denoted as \( R \)) and vice versa.

Observe that if a point robot moves along any path between \( s \) and \( t \) inside the street \( P \), it can see all points of \( P \).
Algorithms for Target Searching in an Unknown Street


The left and right constructed edges of $VP(P, s)$ decide the movement of the robot initially. If $\theta < \pi/2$, then the robot follows the bisector of $\theta$ till it reaches a point where $\theta$ becomes $\pi/2$. 
The left and right constructed edges of \( VP(P, s) \) decide the movement of the robot initially. If \( \theta < \pi/2 \), then the robot follows the bisector of \( \theta \) till it reaches a point where \( \theta \) becomes \( \pi/2 \).

Then the robot follows a curve path toward \( v_l, v_r \) which is defined by an algebraic expression based on positions of current \( p, v_l \) and \( v_r \).
Another problem for searching $t$ in an unknown street $P$ is find a path such that the number of links (or, turns) in the path is as small as possible.

All right pockets occur before all left pockets while traversing the boundary of $P$ in counterclockwise order from $s$. 
- All right pockets occur before all left pockets while traversing the boundary of $P$ in counterclockwise order from $s$.

- Observe that $t$ belongs to either the leftmost top pocket or the rightmost top pocket.
All right pockets occur before all left pockets while traversing the boundary of $P$ in counterclockwise order from $s$.

Observe that $t$ belongs to either the leftmost top pocket or the rightmost top pocket.

If the robot takes any path within $VP(P, s)$ from $s$ to a boundary point between the leftmost and rightmost pockets, it can see all points in every pocket except possibly one.
If the shortest path from $s$ to $t$ makes only right turns or only left turns, then $m + 1$ links are sufficient for the robot to reach from $s$ to $t$, where $m$ is the link distance between $s$ and $t$.

The robot has decided not to turn at $z$ which turns out to be a correct decision as the shortest path from $s$ to $t$ makes only right turn.
The robot has decided not to turn at $z$ as before but it is a wrong decision as the shortest path from $s$ to $t$ makes both types of turns. So, the robot backtracks to $z$ and follows the correct path.
The robot has decided not to turn at $z$ as before but it is a wrong decision as the shortest path from $s$ to $t$ makes both types of turns. So, the robot backtracks to $z$ and follows the correct path.

Since the robot takes one extra link for every such change in turn in the shortest path the robot takes at most $2m - 1$ links to reach from $s$ to $t$. So, the competitive ratio of the online algorithm is $2 - 1/m$ which is shown to be optimal.
Starting from the initial position $s$, the problem is to design a competitive strategy to walk into the kernel of $P$.


Starting from a point $s$ inside $P$, the exploration problem is to design an online algorithm which a point robot can use for moving inside $P$ such that every point of $P$ becomes visible from some point on the exploration path of the robot.
Starting from a point $s$ inside $P$, the exploration problem is to design an online algorithm which a point robot can use for moving inside $P$ such that every point of $P$ becomes visible from some point on the exploration path of the robot.

However, if $P$ contains holes, the exploration problem does not admit competitive strategy, except for very special cases.
Exploring Unknown Polygons: Continuous Visibility

- Starting from a point $s$ inside $P$, the exploration problem is to design an online algorithm which a point robot can use for moving inside $P$ such that every point of $P$ becomes visible from some point on the exploration path of the robot.

- However, if $P$ contains holes, the exploration problem does not admit competitive strategy, except for very special cases.

Observe that if both edges of every reflex vertex $u_i$ of $P$ are seen by the robot, then the entire $P$ has been explored by the robot.

In the next part of the lecture, exploration algorithms and their competitive ratios are presented from the following papers on discrete visibility.


Exploring Unknown Polygons: Discrete Visibility

- In the next part of the lecture, exploration algorithms and their competitive ratios are presented from the following papers on discrete visibility.


Motivation for Discrete Visibility

- Many on-line computational geometry algorithms for exploring unknown polygons assume that the visibility region can be determined in a continuous fashion from each point on a path of a robot. Is this assumption reasonable?
Motivation for Discrete Visibility

- Many on-line computational geometry algorithms for exploring unknown polygons assume that the visibility region can be determined in a continuous fashion from each point on a path of a robot. Is this assumption reasonable?
- Autonomous robots can only carry a limited amount of on-board computing capability.
Many on-line computational geometry algorithms for exploring unknown polygons assume that the visibility region can be determined in a continuous fashion from each point on a path of a robot. Is this assumption reasonable?

Autonomous robots can only carry a limited amount of on-board computing capability.

At the current state of the art, computer vision algorithms that could compute visibility polygons are time consuming.
Motivation for Discrete Visibility

- Many on-line computational geometry algorithms for exploring unknown polygons assume that the visibility region can be determined in a continuous fashion from each point on a path of a robot. Is this assumption reasonable?
- Autonomous robots can only carry a limited amount of on-board computing capability.
- At the current state of the art, computer vision algorithms that could compute visibility polygons are time consuming.
- The computing limitations suggest that it may not be practically feasible to continuously compute the visibility polygon along the robot’s trajectory.
Motivation for Discrete Visibility

- Many on-line computational geometry algorithms for exploring unknown polygons assume that the visibility region can be determined in a continuous fashion from each point on a path of a robot. Is this assumption reasonable?
- Autonomous robots can only carry a limited amount of on-board computing capability.
- At the current state of the art, computer vision algorithms that could compute visibility polygons are time consuming.
- The computing limitations suggest that it may not be practically feasible to continuously compute the visibility polygon along the robot’s trajectory.
- For good visibility, the robot’s camera will typically be mounted on a mast and such devices vibrate during the robot’s movement.
Many on-line computational geometry algorithms for exploring unknown polygons assume that the visibility region can be determined in a continuous fashion from each point on a path of a robot. Is this assumption reasonable?

Autonomous robots can only carry a limited amount of on-board computing capability.

At the current state of the art, computer vision algorithms that could compute visibility polygons are time consuming.

The computing limitations suggest that it may not be practically feasible to continuously compute the visibility polygon along the robot’s trajectory.

For good visibility, the robot’s camera will typically be mounted on a mast and such devices vibrate during the robot’s movement.

Hence for good precision the camera must be stationary while computing visibility polygons.
Motivation for Discrete Visibility

- Many on-line computational geometry algorithms for exploring unknown polygons assume that the visibility region can be determined in a continuous fashion from each point on a path of a robot. Is this assumption reasonable?
- Autonomous robots can only carry a limited amount of on-board computing capability.
- At the current state of the art, computer vision algorithms that could compute visibility polygons are time consuming.
- The computing limitations suggest that it may not be practically feasible to continuously compute the visibility polygon along the robot’s trajectory.
- For good visibility, the robot’s camera will typically be mounted on a mast and such devices vibrate during the robot’s movement.
- Hence for good precision the camera must be stationary while computing visibility polygons.
- It seems feasible to compute visibility polygons only at a discrete number of points.
Exploration Cost

- Is the cost associated with a robot’s physical movement dominate all other associated costs?
Exploration Cost

- Is the cost associated with a robot’s physical movement dominate all other associated costs?

- The essential components that contribute to the total cost required for a robotic exploration can be analyzed as follows. Each move will have two associated costs as follows.

  1. There is the time required to physically execute the move. If we crudely assume that the robot moves at a constant rate, \( r \), during a move, the total time required for motion will be \( rD \), where \( D \) is the total path length.

  2. In an exploratory process where the robot has no apriori knowledge of the environment’s geometry, each move must be planned immediately prior to the move so as to account for the most recently acquired geometric information. The robot will be stationary during this process, which we assume to take time \( t_M \).

  3. Since the robot is stationary during each sensing operation, we assume that it takes time \( t_S \).
Is the cost associated with a robot’s physical movement dominate all other associated costs?

The essential components that contribute to the total cost required for a robotic exploration can be analyzed as follows. Each move will have two associated costs as follows.

1. There is the time required to physically execute the move. If we crudely assume that the robot moves at a constant rate, \( r \), during a move, the total time required for motion will be \( rD \), where \( D \) is the total path length.
Is the cost associated with a robot’s physical movement dominate all other associated costs?

The essential components that contribute to the total cost required for a robotic exploration can be analyzed as follows. Each move will have two associated costs as follows.

1. There is the time required to physically execute the move. If we crudely assume that the robot moves at a constant rate, $r$, during a move, the total time required for motion will be $rD$, where $D$ is the total path length.

2. In an exploratory process where the robot has no apriori knowledge of the environment’s geometry, each move must be planned immediately prior to the move so as to account for the most recently acquired geometric information. The robot will be stationary during this process, which we assume to take time $t_M$. 
Exploration Cost

- Is the cost associated with a robot’s physical movement dominate all other associated costs?

- The essential components that contribute to the total cost required for a robotic exploration can be analyzed as follows. Each move will have two associated costs as follows.

1. There is the time required to physically execute the move. If we crudely assume that the robot moves at a constant rate, $r$, during a move, the total time required for motion will be $rD$, where $D$ is the total path length.

2. In an exploratory process where the robot has no apriori knowledge of the environment’s geometry, each move must be planned immediately prior to the move so as to account for the most recently acquired geometric information. The robot will be stationary during this process, which we assume to take time $t_M$.

3. Since the robot is stationary during each sensing operation, we assume that it takes time $t_S$. 
Let $N_M$ and $N_S$ be respectively the number of moves and the number of sensor operations required to complete the exploration of $P$. Hence, the total cost of an exploration is equated to the total time $T$ required to explore $P$: $T(P) = t_M N_M + t_S N_S + r D$. 

Let $N_M$ and $N_S$ be respectively the number of moves and the number of sensor operations required to complete the exploration of $P$. Hence, the total cost of an exploration is equated to the total time $T$ required to explore $P$: $T(P) = t_M N_M + t_S N_S + r D$.

Now, $(t_M N_M + t_S N_S)$ can be viewed as the time required for computing and maintaining visibility polygons by computer vision algorithms, which is indeed a significant fraction of $T(P)$ because computer vision algorithms consume significant time on modest computers in a relatively cluttered environment.
Let $N_M$ and $N_S$ be respectively the number of moves and the number of sensor operations required to complete the exploration of $P$. Hence, the total cost of an exploration is equated to the total time $T$ required to explore $P$: $T(P) = t_M N_M + t_S N_S + r D$.

Now, $(t_M N_M + t_S N_S)$ can be viewed as the time required for computing and maintaining visibility polygons by computer vision algorithms, which is indeed a significant fraction of $T(P)$ because computer vision algorithms consume significant time on modest computers in a relatively cluttered environment.

Therefore, we assume that the overall cost of exploration is proportional to the cost for computing visibility polygons.

Let $N_M$ and $N_S$ be respectively the number of moves and the number of sensor operations required to complete the exploration of $P$. Hence, the total cost of an exploration is equated to the total time $T$ required to explore $P$: $T(P) = t_M N_M + t_S N_S + r D$.

Now, $(t_M N_M + t_S N_S)$ can be viewed as the time required for computing and maintaining visibility polygons by computer vision algorithms, which is indeed a significant fraction of $T(P)$ because computer vision algorithms consume significant time on modest computers in a relatively cluttered environment.

Therefore, we assume that the overall cost of exploration is proportional to the cost for computing visibility polygons.

The criteria for minimizing the cost for robotic exploration is to reduce the number of visibility polygons that the on-line algorithms compute.

We present an exploration algorithm that a point robot can use to explore an unknown polygonal environment $P$ under discrete visibility.
We present an exploration algorithm that a point robot can use to explore an unknown polygonal environment $P$ under discrete visibility.

In order to explore $P$, the robot starts from a given position, and sees all points of the free space incrementally.
We present an exploration algorithm that a point robot can use to explore an unknown polygonal environment $P$ under discrete visibility.

In order to explore $P$, the robot starts from a given position, and sees all points of the free space incrementally.

It may appear that it is enough to see all vertices and edges of $P$ in order to see the entire free-space. However, this is not the case.
An Exploration Algorithm

We present an exploration algorithm that a point robot can use to explore an unknown polygonal environment \( P \) under discrete visibility.

In order to explore \( P \), the robot starts from a given position, and sees all points of the free space incrementally.

It may appear that it is enough to see all vertices and edges of \( P \) in order to see the entire free-space. However, this is not the case.

Three views from \( p_1 \), \( p_2 \) and \( p_3 \) are enough to see all vertices and edges of \( P \) but not the entire free-space of \( P \).
(i) Let $S$ denote the set of viewing points that the algorithm has computed so far. (ii) The triangulation of $P$ is denoted as $T(P)$. (iii) The visibility polygon of $P$ from a point $p_i$ is denoted as $VP(P, p_i)$. 
(i) Let $S$ denote the set of viewing points that the algorithm has computed so far. (ii) The triangulation of $P$ is denoted as $T(P)$. (iii) The visibility polygon of $P$ from a point $p_i$ is denoted as $VP(P, p_i)$.

**Step 1:** $i := 1; \ T(P) := \emptyset; \ S := \emptyset$; Let $p_1$ denote the starting position of the robot.
(i) Let $S$ denote the set of viewing points that the algorithm has computed so far. (ii) The triangulation of $P$ is denoted as $T(P)$. (iii) The visibility polygon of $P$ from a point $p_i$ is denoted as $VP(P, p_i)$.

- **Step 1:** $i := 1; \ T(P) := \emptyset; \ S := \emptyset$; Let $p_1$ denote the starting position of the robot.

- **Step 2:** Compute $VP(P, p_i)$; Construct the triangulation $T'(P)$ of $VP(P, p_i)$; $T(P) := T(P) \cup T'(P); \ S = S \cup p_i; \ T(P)$
(i) Let $S$ denote the set of viewing points that the algorithm has computed so far. (ii) The triangulation of $P$ is denoted as $T(P)$. (iii) The visibility polygon of $P$ from a point $p_i$ is denoted as $VP(P, p_i)$.

**Step 1:** $i := 1; \ T(P) := \emptyset; \ S := \emptyset; \ $ Let $p_1$ denote the starting position of the robot.

**Step 2:** Compute $VP(P, p_i)$; Construct the triangulation $T'(P)$ of $VP(P, p_i); \ T(P) := T(P) \cup T'(P); \ S = S \cup p_i$;

**Step 3:** While $VP(P, p_i) - T(P) = \emptyset$ and $i \neq 0$ then $i := i - 1$;
Step 4: If $i = 0$ then goto Step 7;
- **Step 4:** If $i = 0$ then goto Step 7;
- **Step 5:** If $VP(P, p_i) - T(P) \neq \emptyset$ then choose a point $z$ on any constructed of $VP(P, p_i)$ lying outside $T(P)$;
- **Step 4:** If $i = 0$ then goto Step 7;

- **Step 5:** If $VP(P, p_i) - T(P) \neq \emptyset$ then choose a point $z$ on any constructed of $VP(P, p_i)$ lying outside $T(P)$;

- **Step 6:** $i := i + 1; \quad p_i := z; \quad$ goto Step 2;
Step 4: If $i = 0$ then goto Step 7;

Step 5: If $VP(P, p_i) - T(P) \neq \emptyset$ then choose a point $z$ on any constructed of $VP(P, p_i)$ lying outside $T(P)$;

Step 6: $i := i + 1; \quad p_i := z; \quad$ goto Step 2;

Step 7: Output $S$ and $T(P)$;
- **Step 4:** If $i = 0$ then goto Step 7;
- **Step 5:** If $VP(P, p_i) - T(P) \neq \emptyset$ then choose a point $z$ on any constructed of $VP(P, p_i)$ lying outside $T(P)$;
- **Step 6:** $i := i + 1; \ p_i := z; \$ goto Step 2;
- **Step 7:** Output $S$ and $T(P)$;
- **Step 8:** Stop.
The algorithm needs $r + 1$ views. Competitive ratio is $(r + 1)/2$, where $r$ denotes the number of reflex vertices of the polygon.
The algorithm needs $r + 1$ views. Competitive ratio is $(r + 1)/2$, where $r$ denotes the number of reflex vertices of the polygon.

Open Problem: Can the bound be improved?
We wish to design an algorithm that a convex robot $C$ can use to explore an unknown polygonal environment $P$ (under translation) following the similar strategy of a point robot.
We wish to design an algorithm that a convex robot $C$ can use to explore an unknown polygonal environment $P$ (under translation) following the similar strategy of a point robot.

- $C$ needs more than $r + 1$ views for exploration.
We wish to design an algorithm that a convex robot $C$ can use to explore an unknown polygonal environment $P$ (under translation) following the similar strategy of a point robot.

$C$ needs more than $r + 1$ views for exploration.

**Open problem:** Can one derive an upper bound on the number of views for a convex robot exploration?
Computer vision range sensors or algorithms, such as stereo or structured light range finder, can reliably compute the 3D scene locations only up to a depth $R$. The reliability of depth estimates is inversely related to the distance from the camera. Thus, the range measurements from a vision sensor for objects that are far away are not at all reliable.
Computer vision range sensors or algorithms, such as stereo or structured light range finder, can reliably compute the 3D scene locations only up to a depth $R$. The reliability of depth estimates is inversely related to the distance from the camera. Thus, the range measurements from a vision sensor for objects that are far away are not at all reliable.

Therefore, the portion of the boundary of a polygonal environment within the range distance $R$ is only considered to be visible from the camera of the robot.
Exploring an Unknown Polygon: Bounded Visibility

- Computer vision range sensors or algorithms, such as stereo or structured light range finder, can reliably compute the 3D scene locations only up to a depth $R$. The reliability of depth estimates is inversely related to the distance from the camera. Thus, the range measurements from a vision sensor for objects that are far away are not at all reliable.

- Therefore, the portion of the boundary of a polygonal environment within the range distance $R$ is only considered to be visible from the camera of the robot.

- Vertices of restricted visibility polygon from $p_i$ with range $R$ are $u_1, u_2, \ldots, u_{12}$. 
The algorithm starts by computing the restricted visibility polygon \( RVP(P, p_1) \) from the starting position \( p_1 \).
An Exploration Algorithm using Restricted Visibility

- The algorithm starts by computing the restricted visibility polygon $RVP(P, p_1)$ from the starting position $p_1$.

- It chooses the next viewing point $p_i$ on a constructed edge or a circular edge of $RVP(P, p_{i-1})$ for $i \geq 1$ till a boundary point $z$ of $P$ becomes visible.
Taking $z$ as the next viewing point $p_i$, $RVP(P, p_i)$ is computed. Taking viewing points along the boundary of $P$ in this fashion, restricted visibility polygons are computed till all points of this boundary of $P$ become visible.
Taking $z$ as the next viewing point $p_i$, $RVP(P, p_i)$ is computed. Taking viewing points along the boundary of $P$ in this fashion, restricted visibility polygons are computed till all points of this boundary of $P$ become visible.

The process of computing restricted visibility polygons ends once the entire $P$ is explored.
The maximum number of views needed to explore the unknown polygon $P$ with $h$ obstacles of size $n$ is bounded by

$$\left\lfloor \frac{8 \times \text{Area}(P)}{3 \times R^2} \right\rfloor + \left\lfloor \frac{\text{Perimeter}(P)}{R} \right\rfloor + r + h + 1.$$
The maximum number of views needed to explore the unknown polygon $P$ with $h$ obstacles of size $n$ is bounded by

$$\left\lceil \frac{8 \times \text{Area}(P)}{3 \times R^2} \right\rceil + \left\lceil \frac{\text{Perimeter}(P)}{R} \right\rceil + r + h + 1.$$ 

The competitive ratio of the algorithm is

$$\left\lceil \frac{8\pi}{3} + \frac{\pi R \times \text{Perimeter}(P)}{\text{Area}(P)} + \frac{(r+h+1) \times \pi R^2}{\text{Area}(P)} \right\rceil.$$ 

Open problem: Can one improve the competitive ratio of the algorithm?
The maximum number of views needed to explore the unknown polygon $P$ with $h$ obstacles of size $n$ is bounded by

$$\left\lceil \frac{8 \times \text{Area}(P)}{3 \times R^2} \right\rceil + \left\lceil \frac{\text{Perimeter}(P)}{R} \right\rceil + r + h + 1.$$ 

The competitive ratio of the algorithm is

$$\left\lceil \frac{8\pi}{3} + \frac{\pi R \times \text{Perimeter}(P)}{\text{Area}(P)} + \frac{(r+h+1) \times \pi R^2}{\text{Area}(P)} \right\rceil.$$ 

Open problem: Can one improve the competitive ratio of the algorithm?
Exploration and Coverage Algorithms


Simple Robots

- So far, we have considered autonomous mobile robots that have best capabilities for movement, sensing, computation and communications.
Simple Robots

- So far, we have considered autonomous mobile robots that have best capabilities for movement, sensing, computation and communications.
- Naturally, such sophisticated robots are costly and deploying a large number of them (e.g. Swarm robots) for a given task may be impractical.

Such simple design robots in terms of hardware are cheap and are, therefore, suitable for mass market.

The question is: What capabilities does a robot need at the very least for a given task?
Simple Robots

- So far, we have considered autonomous mobile robots that have best capabilities for movement, sensing, computation and communications.

- Naturally, such sophisticated robots are costly and deploying a large number of them (*e.g.* Swarm robots) for a given task may be impractical.

- On the other hand, there are tasks that do not require sophisticated robots. For example, many cleaning robots depend on contact sensors only, and make somewhat random movements rather than using sophisticated hardware for optimizing their trajectories.
So far, we have considered autonomous mobile robots that have best capabilities for movement, sensing, computation and communications.

Naturally, such sophisticated robots are costly and deploying a large number of them (e.g. *Swarm robots*) for a given task may be impractical.

On the other hand, there are tasks that do not require sophisticated robots. For example, many cleaning robots depend on contact sensors only, and make somewhat random movements rather than using sophisticated hardware for optimizing their trajectories.

Such simple design robots in terms of hardware are cheap and are, therefore, suitable for mass market.
Simple Robots

- So far, we have considered autonomous mobile robots that have best capabilities for movement, sensing, computation and communications.
- Naturally, such sophisticated robots are costly and deploying a large number of them (e.g. *Swarm robots*) for a given task may be impractical.
- On the other hand, there are tasks that do not require sophisticated robots. For example, many cleaning robots depend on contact sensors only, and make somewhat random movements rather than using sophisticated hardware for optimizing their trajectories.
- Such simple design robots in terms of hardware are cheap and are, therefore, suitable for mass market.
- The question is: What capabilities does a robot need at the very least for a given task?
What capabilities does a robot need at the very least for exploring an unknown polygon?
Mobile Agents

- What capabilities does a robot need at the very least for exploring an unknown polygon?
- Is it enough for a robot to produce a rough map of the polygon?
What capabilities does a robot need at the very least for exploring an unknown polygon?

Is it enough for a robot to produce a rough map of the polygon?

To address these questions, we use a very basic theoretical robot model with additional atomic capabilities.

---


Mobile Agents

- What capabilities does a robot need at the very least for exploring an unknown polygon?
- Is it enough for a robot to produce a rough map of the polygon?
- To address these questions, we use a very basic theoretical robot model with additional atomic capabilities.
- Results from such theoretical models can provide a reference for a realistic design.
Mobile Agents

- What capabilities does a robot need at the very least for exploring an unknown polygon?
- Is it enough for a robot to produce a rough map of the polygon?
- To address these questions, we use a very basic theoretical robot model with additional atomic capabilities.
- Results from such theoretical models can provide a reference for a realistic design.
- These theoretical robots are referred as Agents rather than (realistic) robots.

Mobile Agents

- What capabilities does a robot need at the very least for exploring an unknown polygon?
- Is it enough for a robot to produce a rough map of the polygon?
- To address these questions, we use a very basic theoretical robot model with additional atomic capabilities.
- Results from such theoretical models can provide a reference for a realistic design.
- These theoretical robots are referred to as Agents rather than (realistic) robots.

Unlike normal movements of a robot, agents are restricted to move only along the edges of the visibility graph of the polygon $P$. 

$P$
Unlike normal movements of a robot, agents are restricted to move only along the edges of the visibility graph of the polygon $P$.

The visibility graph of $P$ is a graph whose vertex set consists of the vertices of $P$ and whose edges are visible pairs of vertices of $P$. 
Unlike normal movements of a robot, agents are restricted to move only along the edges of the visibility graph of the polygon $P$.

The visibility graph of $P$ is a graph whose vertex set consists of the vertices of $P$ and whose edges are visible pairs of vertices of $P$.

This means that an agent moves from a vertex to another vertex inside $P$ along the lines of sights.
While located at a vertex, an agent can use its sensor to locate the vertices of $P$ visible from the current position in the counter-clockwise order along the boundary of $P$. However, the agent neither can provide coordinates of these visible vertices nor knows the polygonal numbering of these visible vertices. Moreover, the agent cannot recognize vertices that are seen earlier from other vertices. After exploration, the agent outputs the visibility graph of $P$ as a rough map of $P$. 
While located at a vertex, an agent can use its sensor to locate the vertices of $P$ visible from the current position in the counter-clockwise order along the boundary of $P$.

However, the agent neither can provide co-ordinates of these visible vertices nor knows the polygonal numbering of these visible vertices.
While located at a vertex, an agent can use its sensor to locate the vertices of $P$ visible from the current position in the counter-clockwise order along the boundary of $P$.

However, the agent neither can provide co-ordinates of these visible vertices nor knows the polygonal numbering of these visible vertices.

Moreover, the agent cannot recognize vertices that are seen earlier from other vertices.
While located at a vertex, an agent can use its sensor to locate the vertices of $P$ visible from the current position in the counter-clockwise order along the boundary of $P$.

However, the agent neither can provide co-ordinates of these visible vertices nor knows the polygonal numbering of these visible vertices.

Moreover, the agent cannot recognize vertices that are seen earlier from other vertices.

After exploration, the agent outputs the visibility graph of $P$ as a rough map of $P$. 

Exploration Strategy

- Exploration strategy consists of data collection phase and computation phase.
Exploration Strategy

- Exploration strategy consists of data collection phase and computation phase.
- Data collection phase is always the same: the agent traverses the boundary of $P$ and stops at each vertex for collecting available data including the list of visible vertices.
Exploration strategy consists of data collection phase and computation phase.

Data collection phase is always the same: the agent traverses the boundary of $P$ and stops at each vertex for collecting available data including the list of visible vertices.

The computation phase does not involve any further exploration of $P$, and in this phase, computation is carried out for constructing the visibility graph of $P$. 

Can visibility graph of $P$ be constructed always from available data?
Exploration strategy consists of data collection phase and computation phase.

Data collection phase is always the same: the agent traverses the boundary of $P$ and stops at each vertex for collecting available data including the list of visible vertices.

The computation phase does not involve any further exploration of $P$, and in this phase, computation is carried out for constructing the visibility graph of $P$.

Can visibility graph of $P$ be constructed always from available data?
Starting from a vertex, an agent can traverse the boundary of $P$ in counter-clockwise order by following the first counter-clockwise visible edge from the current position.
Starting from a vertex, an agent can traverse the boundary of $P$ in counter-clockwise order by following the first counter-clockwise visible edge from the current position.

If the agent can distinguish a vertex from all other vertices of $P$, then the visibility graph of $P$ can be constructed easily.
Starting from a vertex, an agent can traverse the boundary of $P$ in counter-clockwise order by following the first counter-clockwise visible edge from the current position.

If the agent can distinguish a vertex from all other vertices of $P$, then the visibility graph of $P$ can be constructed easily.

If the visibility graph is not symmetric, then there is a good chance to locate a vertex that can be distinguished for all other vertices.
Suppose an agent knows the total number of vertices $n$ of $P$ before the boundary traversal. It also has an additional capability to measure the angle at each vertex of $P$. 

Suppose an agent knows the total number of vertices $n$ of $P$ before the boundary traversal. It also has an additional capability to measure the angle at each vertex of $P$.

Even with these enhanced capabilities, the agent cannot always reconstruct the visibility graph of $P$. 

---

Suppose an agent knows the total number of vertices $n$ of $P$ before the boundary traversal. It also has an additional capability to measure the angle at each vertex of $P$.

Even with these enhanced capabilities, the agent cannot always reconstruct the visibility graph of $P$.


Let $u$ and $w$ be two consecutive visible vertices in the angular order of any vertex $v$. 

- Suppose the capability of a basic agent is enhanced with an angle sensor such that the agent can measure the exact angle between $(v, u)$ and $(v, w)$ at $v$ for all $v$, $u$, and $w$ in $P$. 
- Such agents can always construct the visibility graph of a simple polygon $P$ with or without the prior knowledge of $n$. 

---

Let $u$ and $w$ be two consecutive visible vertices in the angular order of any vertex $v$.

Suppose, the capability of an basic agent is enhanced with an angle sensor such that the agent can measure the exact angle between $(v, u)$ and $(v, w)$ at $v$ for all $v$, $u$ and $w$ in $P$. 

---

Let $u$ and $w$ be two consecutive visible vertices in the angular order of any vertex $v$.

Suppose, the capability of an basic agent is enhanced with an angle sensor such that the agent can measure the exact angle between $(v, u)$ and $(v, w)$ at $v$ for all $v$, $u$ and $w$ in $P$.

Such agents can always construct the visibility graph of a simple polygon $P$ with or without the prior knowledge of $n$. 

Let $u$ and $w$ be two consecutive visible vertices in the angular order of any vertex $v$.

Suppose, the capability of an basic agent is enhanced with an angle sensor such that the agent can measure the exact angle between $(v, u)$ and $(v, w)$ at $v$ for all $v$, $u$ and $w$ in $P$.

Such agents can always construct the visibility graph of a simple polygon $P$ with or without the prior knowledge of $n$.

Assume that all visible edges connecting vertices of $\text{chain}(v_i,v_j)$ are identified except the edge $(v_i, v_j)$, and the algorithm wants to determine whether $(v_i, v_j)$ is a visible edge.
Assume that all visible edges connecting vertices of \( \text{chain}(v_i, v_j) \) are identified except the edge \((v_i, v_j)\), and the algorithm wants to determine whether \((v_i, v_j)\) is a visible edge.

Consider a vertex \( v_l \in \text{chain}(v_i, v_j) \) such that \((v_i, v_l)\) and \((v_l, v_j)\) are visible edges.
Assume that all visible edges connecting vertices of $\text{chain}(v_i, v_j)$ are identified except the edge $(v_i, v_j)$, and the algorithm wants to determine whether $(v_i, v_j)$ is a visible edge.

Consider a vertex $v_l \in \text{chain}(v_i, v_j)$ such that $(v_i, v_l)$ and $(v_l, v_j)$ are visible edges.

If no such vertex $v_l$ exists, then obviously $(v_i, v_j)$ is not a visible edge.
Assume that all visible edges connecting vertices of $\text{chain}(v_i, v_j)$ are identified except the edge $(v_i, v_j)$, and the algorithm wants to determine whether $(v_i, v_j)$ is a visible edge.

Consider a vertex $v_l \in \text{chain}(v_i, v_j)$ such that $(v_i, v_l)$ and $(v_l, v_j)$ are visible edges.

If no such vertex $v_l$ exists, then obviously $(v_i, v_j)$ is not a visible edge.

If the internal angles at $v_i$ and $v_j$ of the triangle $(v_i, v_l, v_j)$ match with the corresponding measured angles at $v_i$ and $v_j$ by the agent, then $(v_i, v_j)$ is a visible edge.
By testing every pair of vertices \((v_i, v_j)\) of \(P\) with distances 2, 3, 4, \ldots, all pairs of visible vertices of \(P\) can be identified.
By testing every pair of vertices \((v_i, v_j)\) of \(P\) with distances \(2, 3, 4, \ldots\), all pairs of visible vertices of \(P\) can be identified.

In addition, a simple polygon \(P'\) can also be reconstructed from the visibility graph of \(P\) using the measured angles.
By testing every pair of vertices \((v_i, v_j)\) of \(P\) with distances 2, 3, 4, \ldots, all pairs of visible vertices of \(P\) can be identified.

In addition, a simple polygon \(P'\) can also be reconstructed from the visibility graph of \(P\) using the measured angles.

The overall running time of the algorithm can be improved from \(O(n^3 \log n)\) to \(O(n^2)\).
By testing every pair of vertices \((v_i, v_j)\) of \(P\) with distances 2, 3, 4, \ldots, all pairs of visible vertices of \(P\) can be identified.

In addition, a simple polygon \(P'\) can also be reconstructed from the visibility graph of \(P\) using the measured angles.

The overall running time of the algorithm can be improved from \(O(n^3 \log n)\) to \(O(n^2)\).

Agents with Other Type Sensors

- The angle-type sensor allows to determine whether an angle is convex or reflex.
Agents with Other Type Sensors

- The angle-type sensor allows to determine whether an angle is convex or reflex.
- A look-back sensor allows an agent to return to its previous position, i.e., if an agent moves from a vertex $u$ to another vertex $v$, the agent can return to $u$ using its look-back sensor.
Agents with Other Type Sensors

- The angle-type sensor allows to determine whether an angle is convex or reflex.
- A look-back sensor allows an agent to return to its previous position, i.e., if an agent moves from a vertex \( u \) to another vertex \( v \), the agent can return to \( u \) using its look-back sensor.
- A compass allows an agent to measure the angle at each edge with respect to the global directions.
Agents with Other Type Sensors

- The angle-type sensor allows to determine whether an angle is convex or reflex.
- A look-back sensor allows an agent to return to its previous position, i.e., if an agent moves from a vertex $u$ to another vertex $v$, the agent can return to $u$ using its look-back sensor.
- A compass allows an agent to measure the angle at each edge with respect to the global directions.
- Several algorithms have been proposed for constructing visibility graphs of unknown simple polygons using different combination of sensors.
Agents with Other Type Sensors

- The angle-type sensor allows to determine whether an angle is convex or reflex.
- A look-back sensor allows an agent to return to its previous position, i.e., if an agent moves from a vertex $u$ to another vertex $v$, the agent can return to $u$ using its look-back sensor.
- A compass allows an agent to measure the angle at each edge with respect to the global directions.
- Several algorithms have been proposed for constructing visibility graphs of unknown simple polygons using different combination of sensors.


If $P$ is a polygon with holes, the problem of constructing visibility graphs of $P$ becomes a much a harder problem.
If $P$ is a polygon with holes, the problem of constructing visibility graphs of $P$ becomes a much harder problem.

Recently, an exploration algorithm has been designed for exploring such polygons using basic agents having compass with a prior knowledge of an upper bound on the number of vertices of $P$. 
If $P$ is a polygon with holes, the problem of constructing visibility graphs of $P$ becomes a much a harder problem.

Recently, an exploration algorithm has been design for exploring such polygons using basic agents having compass with a prior knowledge of an upper bound on the number of vertices of $P$.


Open problems: There are several open problems for constructing visibility graphs of unknown polygons $P$ with or without holes for boundary traversal as well for unrestricted traversal of mobile agents with or without additional capabilities.

Thank You.