Visibility and Transformation of Optimal Paths in Simple Polygons

Subir Kumar Ghosh

School of Technology and Computer Science
Tata Institute of Fundamental Research
Mumbai 400005, India
ghosh@tifr.res.in
Organization

1. Visibility in Polygons
2. Euclidean Shortest Paths
3. Minimum Link Paths
4. Optimal Diffused Reflection Paths
Outline

1. Visibility in Polygons
2. Euclidean Shortest Paths
3. Minimum Link Paths
4. Optimal Diffused Reflection Paths
A polygon $P$ is defined as a closed region in the plane bounded by a finite set of line segments (called edges of $P$) such that there exists a path between any two points of $P$ which does not intersect any edge of $P$. 
A polygon $P$ is defined as a closed region in the plane bounded by a finite set of line segments (called edges of $P$) such that there exists a path between any two points of $P$ which does not intersect any edge of $P$.

If the boundary of $P$ consists of two or more cycles, then $P$ is called a polygon with holes. Otherwise, $P$ is called a simple polygon or a polygon without holes.
Two points $u$ and $v$ in a polygon $P$ are said to be *visible* if the line segment joining $u$ and $v$ lies entirely inside $P$. 

---

*Visibility in Polygons* | *Euclidean Shortest Paths* | *Minimum Link Paths* | *Optimal Diffused Reflection Paths*
Two points \( u \) and \( v \) in a polygon \( P \) are said to be \textit{visible} if the line segment joining \( u \) and \( v \) lies entirely inside \( P \).

Using this definition of visibility, a \textit{path} in \( P \) can be defined as a sequence of line segments such that the two endpoints of every line segment are mutually visible, i.e., every such line segment lies totally inside \( P \).
Visibility in Polygons

Euclidean Shortest Paths

Minimum Link Paths

Optimal Diffused Reflection Paths

Two points $u$ and $v$ in a polygon $P$ are said to be *visible* if the line segment joining $u$ and $v$ lies entirely inside $P$.

Using this definition of visibility, a *path* in $P$ can be defined as a sequence of line segments such that the two endpoints of every line segment are mutually visible, i.e., every such line segment lies totally inside $P$.

Outline

1. Visibility in Polygons
2. Euclidean Shortest Paths
3. Minimum Link Paths
4. Optimal Diffused Reflection Paths
Definitions and properties

- **The Euclidean shortest path** (denoted as $SP(s, t)$) between two points $s$ and $t$ in a polygon $P$ is the path of smallest length between $s$ and $t$ lying totally inside $P$. 
The *Euclidean shortest path* (denoted as $SP(s, t)$) between two points $s$ and $t$ in a polygon $P$ is the path of smallest length between $s$ and $t$ lying totally inside $P$.

Let $SP(s, t) = (s, u_1, u_2, ..., u_k, t)$. Then, (i) $SP(s, t)$ is a simple path, (ii) $u_1, u_2, ..., u_k$ are vertices of $P$ and (iii) for all $i$, $u_i$ and $u_{i+1}$ are mutually visible in $P$. $SP(s, t)$ is *outward convex* at every vertex on the path.
Computing $SP(s, t)$

- The dual graph of a triangulation of a simple polygon is a tree.
Computing $SP(s, t)$

- The dual graph of a triangulation of a simple polygon is a tree.
- $SP(s, t)$ passes only through the triangles in the path from $T_s$ and $T_t$ in the dual tree.
Computing $SP(s, t)$

- The dual graph of a triangulation of a simple polygon is a tree.
- $SP(s, t)$ passes only through the triangles in the path from $T_s$ and $T_t$ in the dual tree.


Running time: $O(n)$. 
Let \((u, v, z)\) be a triangle such that \(SP(s, u)\) and \(SP(s, v)\) have already been computed. Then \(SP(s, z)\) can be computed by drawing tangent from \(z\) to \(SP(s, u)\) or \(SP(s, v)\).
Let \((u, v, z)\) be a triangle such that \(SP(s, u)\) and \(SP(s, v)\) have already been computed. Then \(SP(s, z)\) can be computed by drawing tangent from \(z\) to \(SP(s, u)\) or \(SP(s, v)\).

**Open Problem:** Can \(SP(s, t)\) be computed in a simple polygon in \(O(n)\) time without triangulation?
Let \((u, v, z)\) be a triangle such that \(SP(s, u)\) and \(SP(s, v)\) have already been computed. Then \(SP(s, z)\) can be computed by drawing tangent from \(z\) to \(SP(s, u)\) or \(SP(s, v)\).

**Open Problem**: Can \(SP(s, t)\) be computed in a simple polygon in \(O(n)\) time without triangulation?

Outline

1. Visibility in Polygons
2. Euclidean Shortest Paths
3. Minimum Link Paths
4. Optimal Diffused Reflection Paths
A minimum link path connecting two points \( s \) and \( t \) inside a polygon \( P \) with or without holes (denoted by \( MLP(s, t) \)) is a polygonal path with the smallest number of turns or links.
A minimum link path connecting two points $s$ and $t$ inside a polygon $P$ with or without holes (denoted by $MLP(s, t)$) is a polygonal path with the smallest number of turns or links.

A minimum link path connecting two points $s$ and $t$ inside a polygon $P$ with or without holes (denoted by $MLP(s, t)$) is a polygonal path with the smallest number of turns or links.


Computing $MLP(s, t)$: Suri’s algorithm

- $V(1)$ is the visibility polygon of $s$. 

![Diagram showing visibility polygons and windows]
Computing $MLP(s, t)$: Suri’s algorithm

- $V(1)$ is the visibility polygon of $s$.
- For $i > 1$, $V(i)$ is the set of points of $P$ that are visible from some point on a window of $V(i - 1)$. 
Computing $MLP(s, t)$: Suri’s algorithm

- $V(1)$ is the visibility polygon of $s$.
- For $i > 1$, $V(i)$ is the set of points of $P$ that are visible from some point on a window of $V(i - 1)$.
- So, number of links (called link distance) required from $s$ to any point of $V(i)$ is $i$. 
Computing $MLP(s, t)$: Suri’s algorithm

- $V(1)$ is the visibility polygon of $s$.
- For $i > 1$, $V(i)$ is the set of points of $P$ that are visible from some point on a window of $V(i - 1)$.
- So, number of links (called link distance) required from $s$ to any point of $V(i)$ is $i$.
- The turing points of $MLP(s, t)$ are on the windows of $V(i)$ for all $i$. 
Computing $MLP(s, t)$: Ghosh’s algorithm

- Ghosh’s algorithm transforms $SP(s, t)$ into $MLP(s, t)$ in $O(n)$ time.

Let $SP(s, t) = (s, u_1, ..., u_k, t)$. 
Computing $MLP(s, t)$: Ghosh’s algorithm

- Ghosh’s algorithm transforms $SP(s, t)$ into $MLP(s, t)$ in $O(n)$ time.

Let $SP(s, t) = (s, u_1, \ldots, u_k, t)$.

An edge $u_ju_{j-1}$ of $SP(s, t)$ is called eave if $u_{j-2}$ and $u_{j+1}$ lie on the opposite sides of the line passing through $u_j$ and $u_{j-1}$. 
Ghosh’s algorithm

- If an edge $u_k u_{k+1}$ of $SP(s, t)$ is a sub-segment of a link in a link path, we say that the link path contains $u_k u_{k+1}$. 
Ghosh’s algorithm

- If an edge $u_ku_{k+1}$ of $SP(s, t)$ is a sub-segment of a link in a link path, we say that the link path contains $u_ku_{k+1}$.
- There exists a minimum link path between $s$ and $t$ that contains all eaves of $SP(s, t)$. 
Decompose $P$ into sub-polygons by extending each eave from both ends to the boundary of $P$. 
Ghosh’s algorithm

- Decompose $P$ into sub-polygons by extending each eave from both ends to the boundary of $P$.
- If two consecutive extensions intersect at a point $z$, then $z$ is a turning point of $MLP(s, t)$. 
Ghosh’s algorithm

- Decompose $P$ into sub-polygons by extending each eave from both ends to the boundary of $P$.
- If two consecutive extensions intersect at a point $z$, then $z$ is a turning point of $MLP(s, t)$.
- Construct minimum link paths connecting the extensions of every pair of consecutive eaves on $SP(s, t)$ to form $MLP(s, t)$. 
Ghosh’s algorithm

Consider one such sub-polygon (say, \( P_{ij} \)) between the non-intersecting extensions of two consecutive eaves \( u_i u_{i+1} \) and \( u_{j-1} u_j \) of SP(s, t).
Consider one such sub-polygon (say, $P_{ij}$) between the non-intersecting extensions of two consecutive eaves $u_i u_{i+1}$ and $u_{j-1} u_j$ of $SP(s, t)$.

Let $L_{ij}$ denote a minimum link path from a point on $u_{i+1} w_{i+1}$ to some point on $u_{j-1} w_{j-1}$. 
Ghosh’s algorithm

Consider one such sub-polygon (say, $P_{ij}$) between the non-intersecting extensions of two consecutive eaves $u_iu_{i+1}$ and $u_{j-1}u_j$ of $SP(s, t)$.

Let $L_{ij}$ denote a minimum link path from a point on $u_{i+1}w_{i+1}$ to some point on $u_{j-1}w_{j-1}$.

A link path is called convex if it makes only left or only right turns at every turning point in the path.
Ghosh’s algorithm

Consider one such sub-polygon (say, $P_{ij}$) between the non-intersecting extensions of two consecutive eaves $u_i u_{i+1}$ and $u_{j-1} u_j$ of $SP(s, t)$.

Let $L_{ij}$ denote a minimum link path from a point on $u_{i+1} w_{i+1}$ to some point on $u_{j-1} w_{j-1}$.

A link path is called convex if it makes only left or only right turns at every turning point in the path.

A minimum link path $L_{ij}$ is a convex path inside $P_{ij}$. 
The segment $zu_p$ is called the *left tangent* (or *right tangent*) of $z$ at the vertex $u_p \in SP(u_{i+1}, u_{j-1})$ if $zu_p$ lies inside $P_{ij}$ and $z$ lies to the right of $\overrightarrow{u_{p-1}u_p}$ (respectively, $\overrightarrow{u_pu_{p-1}}$) and to the left of $\overrightarrow{u_pu_{p+1}}$ (respectively, $\overrightarrow{u_{p+1}u_p}$).
Ghosh’s algorithm

- The segment $zu_p$ is called the left tangent (or right tangent) of $z$ at the vertex $u_p \in SP(u_{i+1}, u_{j-1})$ if $zu_p$ lies inside $P_{ij}$ and $z$ lies to the right of $u_{p-1}u_p$ (respectively, $u_pu_{p-1}$) and to the left of $u_pu_{p+1}$ (respectively, $u_{p+1}u_p$).

- If a point $z \in P_{ij}$ is on a convex path between $u_{i+1}w_{i+1}$ and $u_{j-1}w_{j-1}$ inside $P_{ij}$, then $z$ has both left and right tangents.
Ghosh’s algorithm

Let $R_{ij}$ denote the set of all points of $P_{ij}$ such that every point of $R_{ij}$ has both left and right tangents to $SP(u_i, u_j)$. 
Let $R_{ij}$ denote the set of all points of $P_{ij}$ such that every point of $R_{ij}$ has both left and right tangents to $SP(u_i, u_j)$.

$R_{ij}$ is called complete visible region of $P_{ij}$ and it can be computed in $O(n)$ time.
Ghosh’s algorithm

- Let $R_{ij}$ denote the set of all points of $P_{ij}$ such that every point of $R_{ij}$ has both left and right tangents to $SP(u_i, u_j)$.
- $R_{ij}$ is called **complete visible region** of $P_{ij}$ and it can be computed in $O(n)$ time.
- Choose an appropriate point $z_1$ on $u_{i+1}w_{i+1}$. 

![Diagram of Ghosh's algorithm](image-url)
Ghosh’s algorithm

Let $R_{ij}$ denote the set of all points of $P_{ij}$ such that every point of $R_{ij}$ has both left and right tangents to $SP(u_i, u_j)$.

$R_{ij}$ is called complete visible region of $P_{ij}$ and it can be computed in $O(n)$ time.

Choose an appropriate point $z_1$ on $u_{i+1}w_{i+1}$.

Draw the right tangent from $z_1$ to $SP(u_{i+1}, u_{j-1})$ and extend the tangent till it meets the boundary of $R_{ij}$ at some point $z_2$. 
Ghosh’s algorithm

- Again, draw the right tangent from $z_2$ to $SP(u_{i+1}, u_{j-1})$ and extend the tangent till it meets the boundary of $R_{ij}$ at some point $z_3$. 
Ghosh’s algorithm

- Again, draw the right tangent from $z_2$ to $SP(u_{i+1}, u_{j-1})$ and extend the tangent till it meets the boundary of $R_{ij}$ at some point $z_3$.
- Repeat this process of construction till a point $z_q$ is found on $u_{j-1}w_{j-1}$.
Ghosh’s algorithm

- Again, draw the right tangent from $z_2$ to $SP(u_{i+1}, u_{j-1})$ and extend the tangent till it meets the boundary of $R_{ij}$ at some point $z_3$.
- Repeat this process of construction till a point $z_q$ is found on $u_{j-1}w_{j-1}$.
- Thus, the greedy path $z_1z_2$, $z_2z_3$, ..., $z_{q-1}z_q$ is constructed between $u_{i+1}w_{i+1}$ and $u_{j-1}w_{j-1}$.
Ghosh’s algorithm

- Again, draw the right tangent from $z_2$ to $SP(u_{i+1}, u_{j-1})$ and extend the tangent till it meets the boundary of $R_{ij}$ at some point $z_3$.
- Repeat this process of construction till a point $z_q$ is found on $u_{j-1}w_{j-1}$.
- Thus, the greedy path $z_1z_2, z_2z_3, ..., z_{q-1}z_q$ is constructed between $u_{i+1}w_{i+1}$ and $u_{j-1}w_{j-1}$.
- The greedy path $z_1z_2, z_2z_3, ..., z_{q-1}z_q$ is a minimum link path inside $P_{ij}$. 
Ghosh’s algorithm

- **Overall Algorithm:**

  Compute $SP(s, t)$ using the algorithm of Lee and Preparata.

  Decompose $P$ into sub-polygons by extending each eave of $SP(s, t)$ from both ends to the boundary of $P$. Also extend the first and the last edges of $SP(s, t)$.

  In each sub-polygon of $P$, construct the greedy path between the extensions of the eaves.

  Connect the greedy paths using the extension of the eaves to form a minimum link path between $s$ and $t$.

  $SP(s, t)$ can be transformed to $MLP(s, t)$ in $O(n)$ time.

  $SP(s, t)$ and $MLP(s, t)$ belong to the same homotopy class.

Ghosh’s algorithm

**Overall Algorithm:**

- Compute $SP(s, t)$ using the algorithm of Lee and Preparata.

Ghosh’s algorithm

**Overall Algorithm:**

- Compute \( SP(s, t) \) using the algorithm of Lee and Preparata.
- Decompose \( P \) into sub-polygons by extending each eave of \( SP(s, t) \) from both ends to the boundary of \( P \). Also extend the first and the last edges of \( SP(s, t) \).
Ghosh’s algorithm

**Overall Algorithm:**

- Compute $SP(s, t)$ using the algorithm of Lee and Preparata.
- Decompose $P$ into sub-polygons by extending each eave of $SP(s, t)$ from both ends to the boundary of $P$. Also extend the first and the last edges of $SP(s, t)$.
- In each sub-polygon of $P$, construct the greedy path between the extensions of the eaves.
Ghosh’s algorithm

**Overall Algorithm:**

- Compute $SP(s, t)$ using the algorithm of Lee and Preparata.
- Decompose $P$ into sub-polygons by extending each eave of $SP(s, t)$ from both ends to the boundary of $P$. Also extend the first and the last edges of $SP(s, t)$.
- In each sub-polygon of $P$, construct the greedy path between the extensions of the eaves.
- Connect the greedy paths using the extension of the eaves to form a minimum link path between $s$ and $t$.

Ghosh’s algorithm

- **Overall Algorithm:**
  - Compute $SP(s, t)$ using the algorithm of Lee and Preparata.
  - Decompose $P$ into sub-polygons by extending each eave of $SP(s, t)$ from both ends to the boundary of $P$. Also extend the first and the last edges of $SP(s, t)$.
  - In each sub-polygon of $P$, construct the greedy path between the extensions of the eaves.
  - Connect the greedy paths using the extension of the eaves to form a minimum link path between $s$ and $t$.
  
- $SP(s, t)$ can be transformed to $MLP(s, t)$ in $O(n)$ time.
Ghosh’s algorithm

- **Overall Algorithm:**
  - Compute $SP(s, t)$ using the algorithm of Lee and Preparata.
  - Decompose $P$ into sub-polygons by extending each eave of $SP(s, t)$ from both ends to the boundary of $P$. Also extend the first and the last edges of $SP(s, t)$.
  - In each sub-polygon of $P$, construct the greedy path between the extensions of the eaves.
  - Connect the greedy paths using the extension of the eaves to form a minimum link path between $s$ and $t$.

  - $SP(s, t)$ can be transformed to $MLP(s, t)$ in $O(n)$ time.
  - $SP(s, t)$ and $MLP(s, t)$ belong to the same homotopy class.
Ghosh’s algorithm

**Overall Algorithm:**

- Compute $SP(s, t)$ using the algorithm of Lee and Preparata.
- Decompose $P$ into sub-polygons by extending each eave of $SP(s, t)$ from both ends to the boundary of $P$. Also extend the first and the last edges of $SP(s, t)$.
- In each sub-polygon of $P$, construct the greedy path between the extensions of the eaves.
- Connect the greedy paths using the extension of the eaves to form a minimum link path between $s$ and $t$.

$SP(s, t)$ can be transformed to $MLP(s, t)$ in $O(n)$ time.

$SP(s, t)$ and $MLP(s, t)$ belong to the same homotopy class.

Outline

1. Visibility in Polygons
2. Euclidean Shortest Paths
3. Minimum Link Paths
4. Optimal Diffused Reflection Paths
Indirect visibility

- Assume that all edges of $P$ reflect light like mirrors.
Indirect visibility

- Assume that all edges of $P$ reflect light like mirrors.
- Some points of $P$, not directly visible or illuminated from $s$, may become visible due to one or more reflections on the edges of $P$. 
Indirect visibility

- Assume that all edges of $P$ reflect light like mirrors.
- Some points of $P$, not directly visible or illuminated from $s$, may become visible due to one or more reflections on the edges of $P$.
- As per the standard law of reflection, reflection of a light ray at a point is called *specular* if the angle of incidence is the same as the angle of reflection.
Indirect visibility

- Assume that all edges of $P$ reflect light like mirrors.
- Some points of $P$, not directly visible or illuminated from $s$, may become visible due to one or more reflections on the edges of $P$.
- As per the standard law of reflection, reflection of a light ray at a point is called \textit{specular} if the angle of incidence is the same as the angle of reflection.
- There is another type of reflection of light called \textit{diffuse} reflection, where a light ray incident at a point is reflected in all possible interior directions.
So, specular reflections can be viewed as a special type of diffuse reflections.
Indirect visibility

- So, specular reflections can be viewed as a special type of diffuse reflections.
- We assume that the light ray incident at a vertex is absorbed and not reflected.
Indirect visibility

- So, specular reflections can be viewed as a special type of diffuse reflections.
- We assume that the light ray incident at a vertex is absorbed and not reflected.

Visibility with multiple reflections

• Visibility with multiple reflections arises in three-dimensional scenarios naturally where pixels of a screen are rendered to generate a realistic image.

Visibility with multiple reflections

- Visibility with multiple reflections arises in three-dimensional scenarios naturally where pixels of a screen are rendered to generate a realistic image.
- The rendering process needs accumulated illumination information from possible incident directions at each reflection point.

Visibility with multiple reflections

- Visibility with multiple reflections arises in three-dimensional scenarios naturally where pixels of a screen are rendered to generate a realistic image.
- The rendering process needs accumulated illumination information from possible incident directions at each reflection point.
- Since a smaller number of reflections would contribute light more intensely, computing paths of light rays reachable by the minimum number of reflections are naturally important in illumination modeling.

Visibility with multiple reflections

Visibility with multiple reflections arises in three-dimensional scenarios naturally where pixels of a screen are rendered to generate a realistic image.

The rendering process needs accumulated illumination information from possible incident directions at each reflection point.

Since a smaller number of reflections would contribute light more intensely, computing paths of light rays reachable by the minimum number of reflections are naturally important in illumination modeling.

Our motivation is the computation of such a path with the minimum number of diffuse reflections in polynomial time from a point light source $s$ to any point $t$ within a polygon $P$. 

Visibility with multiple reflections

- Visibility with multiple reflections arises in three-dimensional scenarios naturally where pixels of a screen are rendered to generate a realistic image.
- The rendering process needs accumulated illumination information from possible incident directions at each reflection point.
- Since a smaller number of reflections would contribute light more intensely, computing paths of light rays reachable by the minimum number of reflections are naturally important in illumination modeling.
- Our motivation is the computation of such a path with the minimum number of diffuse reflections in polynomial time from a point light source $s$ to any point $t$ within a polygon $P$.
Previous results

Previous results


Previous results

Previous results

Previous results

Previous results

Previous results

Previous results

Previous results


Computing diffuse reflection paths

A path between two points inside $P$ is called a *diffuse reflection path* if all turning points of the path lie on edges of $P$. 
Computing diffuse reflection paths

- A path between two points inside $P$ is called a *diffuse reflection path* if all turning points of the path lie on edges of $P$.

- A diffuse reflection path between two points is said to be *optimal* if it has the minimum number of reflections among all diffuse reflection paths between them.
A path between two points inside $P$ is called a *diffuse reflection path* if all turning points of the path lie on edges of $P$.

A diffuse reflection path between two points is said to be *optimal* if it has the minimum number of reflections among all diffuse reflection paths between them.

**Problem:** Given a polygon $P$ and two internal points $s$ and $t$ inside $P$, compute an optimal diffuse reflection path between $s$ to $t$ in polynomial time.
A path between two points inside $P$ is called a *diffuse reflection path* if all turning points of the path lie on edges of $P$.

A diffuse reflection path between two points is said to be *optimal* if it has the minimum number of reflections among all diffuse reflection paths between them.

**Problem:** Given a polygon $P$ and two internal points $s$ and $t$ inside $P$, compute an optimal diffuse reflection path between $s$ to $t$ in polynomial time.

**Status:** There is no polynomial time algorithm known for the above problem. On the other hand, the problem is also not known to be NP-hard.
Computing diffuse reflection paths

- **Results:** For this problem, we present three different algorithms which produce sub-optimal diffused reflection paths in polynomial time:
Computing diffuse reflection paths

Results: For this problem, we present three different algorithms which produce sub-optimal diffused reflection paths in polynomial time:

- The first algorithm uses a greedy method with the help of Euclidean shortest paths.
Results: For this problem, we present three different algorithms which produce sub-optimal diffused reflection paths in polynomial time:

- The first algorithm uses a greedy method with the help of Euclidean shortest paths.
- The second algorithm uses a transformation of a minimum link path.
Computing diffuse reflection paths

Results: For this problem, we present three different algorithms which produce sub-optimal diffused reflection paths in polynomial time:

- The first algorithm uses a greedy method with the help of Euclidean shortest paths.
- The second algorithm uses a transformation of a minimum link path.
- The third algorithm uses the edge-edge visibility graph of $P$. 
Computing diffuse reflection paths

- **Results:** For this problem, we present three different algorithms which produce sub-optimal diffused reflection paths in polynomial time:
  - The first algorithm uses a greedy method with the help of Euclidean shortest paths.
  - The second algorithm uses a transformation of a minimum link path.
  - The third algorithm uses the edge-edge visibility graph of $P$.

Greedy method

- Compute the Euclidean shortest path \((u_0, u_1, \ldots, u_j)\), where \(s = u_0\) and \(t = u_j\).
Greedy method

- Compute the Euclidean shortest path \((u_0, u_1, \ldots, u_j)\), where \(s = u_0\) and \(t = u_j\).
- Extend the first edge \(u_0u_1\) from \(u_1\) meeting the boundary of \(P\) at some point \(w_1\).
Greedy method

- Compute the Euclidean shortest path \((u_0, u_1, \ldots, u_j)\), where \(s = u_0\) and \(t = u_j\).
- Extend the first edge \(u_0 u_1\) from \(u_1\) meeting the boundary of \(P\) at some point \(w_1\).
- Treating \(w_1\) as \(s\), compute the next link \(w_1 w_2\) by extending the first edge of \(SP(w_1, t)\) to the boundary of \(P\).
Greedy method

- Compute the Euclidean shortest path \((u_0, u_1, \ldots, u_j)\), where \(s = u_0\) and \(t = u_j\).
- Extend the first edge \(u_0u_1\) from \(u_1\) meeting the boundary of \(P\) at some point \(w_1\).
- Treating \(w_1\) as \(s\), compute the next link \(w_1w_2\) by extending the first edge of \(SP(w_1, t)\) to the boundary of \(P\).
- Repeat this process until \(w_k\) is computed such that \(w_k\) is directly visible from \(t\).
Greedy method

- Compute the Euclidean shortest path \((u_0, u_1, \ldots, u_j)\), where \(s = u_0\) and \(t = u_j\).
- Extend the first edge \(u_0u_1\) from \(u_1\) meeting the boundary of \(P\) at some point \(w_1\).
- Treating \(w_1\) as \(s\), compute the next link \(w_1w_2\) by extending the first edge of \(SP(w_1, t)\) to the boundary of \(P\).
- Repeat this process until \(w_k\) is computed such that \(w_k\) is directly visible from \(t\).
- The greedy path \((sw_1, w_1w_2, \ldots, w_{k-1}w_k, w_kt)\) is a diffuse reflection path from \(s\) to \(t\). Note that the path is simple.
Greedy method

- Compute the Euclidean shortest path \((u_0, u_1, \ldots, u_j)\), where \(s = u_0\) and \(t = u_j\).
- Extend the first edge \(u_0u_1\) from \(u_1\) meeting the boundary of \(P\) at some point \(w_1\).
- Treating \(w_1\) as \(s\), compute the next link \(w_1w_2\) by extending the first edge of \(SP(w_1, t)\) to the boundary of \(P\).
- Repeat this process until \(w_k\) is computed such that \(w_k\) is directly visible from \(t\).
- The greedy path \((sw_1, w_1w_2, \ldots, w_{k-1}w_k, w_k t)\) is a diffuse reflection path from \(s\) to \(t\). Note that the path is simple.
- The greedy path can be computed in \(O(n^2)\) time.
Greedy method

- Instead of computing shortest paths repeatedly, the algorithm computes the shortest path tree rooted at $t$, and then constructs the shortest path map by extending the edges of the tree.
**Greedy method**

- Instead of computing shortest paths repeatedly, the algorithm computes the shortest path tree rooted at \( t \), and then constructs the shortest path map by extending the edges of the tree.

- Observe that the next vertex (say, \( v_i \)) of \( w_{i-1} \) in the shortest path from \( w_{i-1} \) to \( t \) is the vertex of the triangle in the shortest path map which contains \( w_{i-1} \).
Greedy method

- Instead of computing shortest paths repeatedly, the algorithm computes the shortest path tree rooted at $t$, and then constructs the shortest path map by extending the edges of the tree.
- Observe that the next vertex (say, $v_i$) of $w_{i-1}$ in the shortest path from $w_{i-1}$ to $t$ is the vertex of the triangle in the shortest path map which contains $w_{i-1}$.
- The greedy path can be computed in $O(n + k \log n)$ time.
Greedy method

Instead of computing shortest paths repeatedly, the algorithm computes the shortest path tree rooted at $t$, and then constructs the shortest path map by extending the edges of the tree.

Observe that the next vertex (say, $v_i$) of $w_{i-1}$ in the shortest path from $w_{i-1}$ to $t$ is the vertex of the triangle in the shortest path map which contains $w_{i-1}$.

The greedy path can be computed in $O(n + k \log n)$ time.

The number of links in the greedy path can be at most \((n - 1)/2 \) times that of an optimal diffuse reflection path.
Transforming a $MLP(s, t)$

- If all turning points of a $MLP(s, t)$ lie on edges of $P$, then the path is an optimal diffuse reflection path.
Transforming a $MLP(s, t)$

For every turning point $z_i$ not lying on any edge of $P$, extend $z_i z_{i+1}$ from $z_i$ to the boundary of $P$ meeting it at a point $a_i$. 
For every turning point $z_i$ not lying on any edge of $P$, extend $z_iz_{i+1}$ from $z_i$ to the boundary of $P$ meeting it at a point $a_i$.

Similarly, extend $z_iz_{i-1}$ from $z_i$ to the boundary of $P$ meeting it at a point $c_i$. 
Transforming a $\text{MLP}(s, t)$

- For every turning point $z_i$ not lying on any edge of $P$, extend $z_iz_{i+1}$ from $z_i$ to the boundary of $P$ meeting it at a point $a_i$.
- Similarly, extend $z_iz_{i-1}$ from $z_i$ to the boundary of $P$ meeting it at a point $c_i$.
- If the segment $a_ic_i$ lies inside $P$, then the diffuse reflection path is $(sz_1, z_1z_2, \ldots, z_{i-1}c_i, c_ia_i, a_iz_{i+1}, \ldots, z_{m-1}z_m, z_mt)$. 
Transforming a $MLP(s, t)$

- Otherwise, $a_i$ and $c_i$ are connected by a greedy path to construct a diffuse reflection path.
Transforming a $MLP(s, t)$

Otherwise, $a_i$ and $c_i$ are connected by a greedy path to construct a diffuse reflection path.

The greedy paths connecting every pair of $a_i$ and $c_i$ lie in disjoint regions of $P$. 
Transforming a $MLP(s, t)$

- Otherwise, $a_i$ and $c_i$ are connected by a greedy path to construct a diffuse reflection path.
- The greedy paths connecting every pair of $a_i$ and $c_i$ lie in disjoint regions of $P$.
- Therefore, a $MLP(s, t)$ can be transformed into a diffuse reflection path from $s$ to $t$ in $O(n + k \log n)$ time.
Let $m'$ be the number of turning points of the $MLP(s, t)$ not lying on the boundary of $P$. 
Let $m'$ be the number of turning points of the $MLP(s, t)$ not lying on the boundary of $P$.

So, the number of links in the greedy path can be at most $n - 2m' + 2m' - m - 1$ because
Let $m'$ be the number of turning points of the $MLP(s, t)$ not lying on the boundary of $P$.

So, the number of links in the greedy path can be at most $n - 2m' + 2m' - m - 1$ because

- the greedy link path from $c_i$ to $a_i$ does not pass through one vertex of the edge containing $c_i$ and another vertex of the edge containing $a_i$,
Worst case ratio

- the last two links for each of the $m'$ greedy paths do not pass through vertices of $P$, and
Worst case ratio

- the last two links for each of the $m'$ greedy paths do not pass through vertices of $P$, and
- the number of vertices in the clockwise boundary of $P$ from $v_i$ to $v_l$ (including $v_i$ and $v_l$) must be at least $2 + m - 1$ as $m - 1$ links of the minimum link path pass through distinct vertices the shortest path.
Worst case ratio

- the last two links for each of the $m'$ greedy paths do not pass through vertices of $P$, and
- the number of vertices in the clockwise boundary of $P$ from $v_i$ to $v_l$ (including $v_i$ and $v_l$) must be at least $2 + m - 1$ as $m - 1$ links of the minimum link path pass through distinct vertices the shortest path.

Therefore, the diffuse reflection path has at most $n - m - 1 + m = n - 1$ links.
Worst case ratio

- the last two links for each of the $m'$ greedy paths do not pass through vertices of $P$, and
- the number of vertices in the clockwise boundary of $P$ from $v_i$ to $v_l$ (including $v_i$ and $v_l$) must be at least $2 + m - 1$ as $m - 1$ links of the minimum link path pass through distinct vertices the shortest path.

Therefore, the diffuse reflection path has at most $n - m - 1 + m = n - 1$ links.
- So, the number of links in the transformed path is at most $(n - 1)/m$ times that of an optimal diffuse reflection path.
Combinatorial approach

- Two edges of $P$ are said to be *weakly visible* if some internal point of one edge is visible from an internal point of the other edge.
Combinatorial approach

- Two edges of $P$ are said to be *weakly visible* if some internal point of one edge is visible from an internal point of the other edge.
- The edge-edge visibility graph $G_e$ of $P$ is a graph with nodes $V_e$ representing all edges of $P$, and arcs between nodes that correspond to weakly visible pairs of edges in $P$. 
Combinatorial approach

- Two edges of $P$ are said to be *weakly visible* if some internal point of one edge is visible from an internal point of the other edge.
- The edge-edge visibility graph $G_e$ of $P$ is a graph with nodes $V_e$ representing all edges of $P$, and arcs between nodes that correspond to weakly visible pairs of edges in $P$.
- Construct the edge-edge visibility graph $G_e$ of $P$, and add two nodes representing $s$ and $t$ in $V_e$.
Combinatorial approach

- Two edges of $P$ are said to be \textit{weakly visible} if some internal point of one edge is visible from an internal point of the other edge.
- The edge-edge visibility graph $G_e$ of $P$ is a graph with nodes $V_e$ representing all edges of $P$, and arcs between nodes that correspond to weakly visible pairs of edges in $P$.
- Construct the edge-edge visibility graph $G_e$ of $P$, and add two nodes representing $s$ and $t$ in $V_e$.
- The node $s$ (or, $t$) is connected by arcs in $G_e$ to those nodes in $V_e$ whose corresponding edges in $P$ are partially or totally visible from $s$ (respectively, $t$).
Combinatorial approach

- Two edges of $P$ are said to be *weakly visible* if some internal point of one edge is visible from an internal point of the other edge.
- The edge-edge visibility graph $G_e$ of $P$ is a graph with nodes $V_e$ representing all edges of $P$, and arcs between nodes that correspond to weakly visible pairs of edges in $P$.
- Construct the edge-edge visibility graph $G_e$ of $P$, and add two nodes representing $s$ and $t$ in $V_e$.
- The node $s$ (or, $t$) is connected by arcs in $G_e$ to those nodes in $V_e$ whose corresponding edges in $P$ are partially or totally visible from $s$ (respectively, $t$).
- Between $s$ and $t$, the number of reflections in any diffuse reflection path in $P$ is at least the minimum number of edges of $P$ in a path between $s$ and $t$ in $G_e$. 

Combinatorial approach

- Two edges of \( P \) are said to be \textit{weakly visible} if some internal point of one edge is visible from an internal point of the other edge.
- The edge-edge visibility graph \( G_\text{e} \) of \( P \) is a graph with nodes \( V_\text{e} \) representing all edges of \( P \), and arcs between nodes that correspond to weakly visible pairs of edges in \( P \).
- Construct the edge-edge visibility graph \( G_\text{e} \) of \( P \), and add two nodes representing \( s \) and \( t \) in \( V_\text{e} \).
- The node \( s \) (or, \( t \)) is connected by arcs in \( G_\text{e} \) to those nodes in \( V_\text{e} \) whose corresponding edges in \( P \) are partially or totally visible from \( s \) (respectively, \( t \)).
- Between \( s \) and \( t \), the number of reflections in any diffuse reflection path in \( P \) is at least the minimum number of edges of \( P \) in a path between \( s \) and \( t \) in \( G_\text{e} \).

Combinatorial approach

- Compute the shortest path from $s$ to $t$ in $G_e$ using BFS.
Combinatorial approach

- Compute the shortest path from $s$ to $t$ in $G_e$ using BFS.
- Let $g_1, g_2, \ldots, g_{k-1}$ be the sequence of edges of $P$ corresponding to the nodes of $V_e$ in the path.
Combinatorial approach

- Compute the shortest path from $s$ to $t$ in $G_e$ using BFS.
- Let $g_1, g_2, \ldots, g_{k-1}$ be the sequence of edges of $P$ corresponding to the nodes of $V_e$ in the path.
- Locate a pair of internal points $z_i \in g_i$ and $u_i \in g_{i+1}$, for all $i$, such that the segment $z_i u_i$ lies inside $P$. 
Combinatorial approach

- Compute the shortest path from $s$ to $t$ in $G_e$ using BFS.
- Let $g_1, g_2, \ldots, g_{k-1}$ be the sequence of edges of $P$ corresponding to the nodes of $V_e$ in the path.
- Locate a pair of internal points $z_i \in g_i$ and $u_i \in g_{i+1}$, for all $i$, such that the segment $z_i u_i$ lies inside $P$.
- Let $u_0$ be a point in $g_1$ visible from $s$. Let $z_{k-1}$ be a point in $g_{k-1}$ visible from $t$. 
Combinatorial approach

- Compute the shortest path from $s$ to $t$ in $G_e$ using BFS.
- Let $g_1, g_2, \ldots, g_{k-1}$ be the sequence of edges of $P$ corresponding to the nodes of $V_e$ in the path.
- Locate a pair of internal points $z_i \in g_i$ and $u_i \in g_{i+1}$, for all $i$, such that the segment $z_i u_i$ lies inside $P$.
- Let $u_0$ be a point in $g_1$ visible from $s$. Let $z_{k-1}$ be a point in $g_{k-1}$ visible from $t$.
- So, a sequence of links $su_0, z_1u_1, \ldots, z_{k-2}u_{k-2}, z_{k-1}t$ can be constructed.
If $z_i = u_{i-1}$, for all $i$, then we have a diffuse reflection path $sz_1, z_1z_2, \ldots, z_{k-1}t$ with the minimum number of reflections.
Combinatorial approach

Otherwise, for every \( z_i \neq u_{i-1} \), locate a point \( z_i' \) on an edge \( e_i \) of \( P \) such that all points of \( g_i \) are visible from \( z_i' \), and then add two links \( u_{i-1}z_i' \) and \( z_i'z_i \) to connect \( u_{i-1} \) with \( z_i \).
Combinatorial approach

- Otherwise, for every $z_i \neq u_{i-1}$, locate a point $z_i'$ on an edge $e_i$ of $P$ such that all points of $g_i$ are visible from $z_i'$, and then add two links $u_{i-1}z_i'$ and $z_i'z_i$ to connect $u_{i-1}$ with $z_i$.

- The point $z_i'$ can be located by extending the edge $g_i$ to the nearest polygonal edge $e_i$ and then choosing a point arbitrary close to the intersection point.
Combinatorial approach

- Otherwise, for every \( z_i \neq u_{i-1} \), locate a point \( z'_i \) on an edge \( e_i \) of \( P \) such that all points of \( g_i \) are visible from \( z'_i \), and then add two links \( u_{i-1}z'_i \) and \( z'_iz_i \) to connect \( u_{i-1} \) with \( z_i \).
- The point \( z'_i \) can be located by extending the edge \( g_i \) to the nearest polygonal edge \( e_i \) and then choosing a point arbitrary close to the intersection point.
- Hence, \((su_0, u_0z'_1, z'_1z_1, z_1u_1, \ldots, u_{k-2}z'_{k-1}, z'_{k-1}z_{k-1}, z_{k-1}t)\) becomes a diffuse reflection path between \( s \) and \( t \).
Combinatorial approach

- Otherwise, for every \( z_i \neq u_{i-1} \), locate a point \( z'_i \) on an edge \( e_i \) of \( P \) such that all points of \( g_i \) are visible from \( z'_i \), and then add two links \( u_{i-1}z'_i \) and \( z'_iz_i \) to connect \( u_{i-1} \) with \( z_i \).
- The point \( z'_i \) can be located by extending the edge \( g_i \) to the nearest polygonal edge \( e_i \) and then choosing a point arbitrary close to the intersection point.
- Hence, \( (su_0, u_0z'_1, z'_1z_1, z_1u_1, \ldots, u_{k-2}z'_{k-1}, z'_{k-1}z_{k-1}, z_{k-1}t) \) becomes a diffuse reflection path between \( s \) and \( t \).
- This path can be at most three times that of an optimal diffusion reflection path.
Combinatorial approach

- All pairs of weakly visible edges of $P$ can be located in $O(n \log n + E)$ time (i) by computing the visibility graph of vertices of $P$ and (ii) by traversing the visibility graph using funnel sequences with polygons edges as bases of the funnels.
Combinatorial approach

- All pairs of weakly visible edges of $P$ can be located in $O(n \log n + E)$ time (i) by computing the visibility graph of vertices of $P$ and (ii) by traversing the visibility graph using *funnel sequences* with polygons edges as bases of the funnels.

- In addition, links connecting pairs of weakly visible edges of $P$ can also be constructed using funnel sequences in $O(n^2)$ time.
Combinatorial approach

- All pairs of weakly visible edges of $P$ can be located in $O(n \log n + E)$ time (i) by computing the visibility graph of vertices of $P$ and (ii) by traversing the visibility graph using funnel sequences with polygons edges as bases of the funnels.

- In addition, links connecting pairs of weakly visible edges of $P$ can also be constructed using funnel sequences in $O(n^2)$ time.

- By traversing the funnel sequences again, edges $g_1, g_2, \ldots, g_{k-1}$ can be extended to the respective nearest edges in $P$ to locate points $z'_1, z'_2, \ldots, z'_{k-1}$ respectively in $O(n^2)$ time.
Combinatorial approach

- All pairs of weakly visible edges of $P$ can be located in $O(n \log n + E)$ time (i) by computing the visibility graph of vertices of $P$ and (ii) by traversing the visibility graph using *funnel sequences* with polygons edges as bases of the funnels.

- In addition, links connecting pairs of weakly visible edges of $P$ can also be constructed using funnel sequences in $O(n^2)$ time.

- By traversing the funnel sequences again, edges $g_1, g_2, \ldots, g_{k-1}$ can be extended to the respective nearest edges in $P$ to locate points $z'_1, z'_2, \ldots, z'_{k-1}$ respectively in $O(n^2)$ time.

- Hence, the entire diffusion reflection path can be computed in $O(n^2)$ time.
Combinatorial approach

- All pairs of weakly visible edges of $P$ can be located in $O(n \log n + E)$ time (i) by computing the visibility graph of vertices of $P$ and (ii) by traversing the visibility graph using funnel sequences with polygons edges as bases of the funnels.
- In addition, links connecting pairs of weakly visible edges of $P$ can also be constructed using funnel sequences in $O(n^2)$ time.
- By traversing the funnel sequences again, edges $g_1, g_2, \ldots, g_{k-1}$ can be extended to the respective nearest edges in $P$ to locate points $z'_1, z'_2, \ldots, z'_{k-1}$ respectively in $O(n^2)$ time.
- Hence, the entire diffusion reflection path can be computed in $O(n^2)$ time.

We have presented three polynomial time algorithms for computing diffuse reflection paths from a light source \( s \) to a target point \( t \) inside \( P \) which produce sub-optimal paths.
Remarks on diffused reflection paths

- We have presented three polynomial time algorithms for computing diffuse reflection paths from a light source \( s \) to a target point \( t \) inside \( P \) which produce sub-optimal paths.

- Observe that the combinatorial approach gives a better bound but it does not give a simple or structured path. On the other hand, the greedy approach gives a simple and structured path but it does not give a good bound.
Remarks on diffused reflection paths

- We have presented three polynomial time algorithms for computing diffuse reflection paths from a light source $s$ to a target point $t$ inside $P$ which produce sub-optimal paths.

- Observe that the combinatorial approach gives a better bound but it does not give a simple or structured path. On the other hand, the greedy approach gives a simple and structured path but it does not give a good bound.

- It will be interesting to design an algorithm, possibly by transforming a $MLP(s, t)$, giving a simple and structured path as well as giving a good bound.
Remarks on diffused reflection paths

- We have presented three polynomial time algorithms for computing diffuse reflection paths from a light source $s$ to a target point $t$ inside $P$ which produce sub-optimal paths.

- Observe that the combinatorial approach gives a better bound but it does not give a simple or structured path. On the other hand, the greedy approach gives a simple and structured path but it does not give a good bound.

- It will be interesting to design an algorithm, possibly by transforming a $MLP(s, t)$, giving a simple and structured path as well as giving a good bound.

- Finally, it is open whether an optimal path can be computed for this problem in a low-order polynomial time.
Concluding remarks

- Our algorithms demonstrate how geometric and topological properties like convexity, simplicity, complete visibility, homotopy, etc., are crucial in computing, transforming, and understanding different paths inside simple polygons that are optimal or close to optimal.
Concluding remarks

- Our algorithms demonstrate how geometric and topological properties like convexity, simplicity, complete visibility, homotopy, etc., are crucial in computing, transforming, and understanding different paths inside simple polygons that are optimal or close to optimal.

- THANK YOU