Computing Positively Weighted Straight Skeletons of Simple Polygons Using an Induced Line Arrangement

Günther Eder and Martin Held

University of Salzburg
FB Computerwissenschaften
5020 Salzburg, Austria

Alicante, June 2017

Related Work & Overview

Related Work

- The straight skeleton was introduced by Aichholzer et al. 1995 [1].
- Eppstein and Erickson [3] introduced in 1999 an algorithm with the current best worst-case complexity: For a simple polygon (with holes) it requires $\mathcal{O}(n^{17/11+\varepsilon})$ time and space to compute the (weighted) straight skeleton.
- More resent results with lower time/space-complexity are known [2, 6][1] but only for (unweighted) straight skeletons.

[1] Cheng, Mencel, and Vigneron as well as Vigneron and Yan.

Related Work & Overview

Related Work

- The straight skeleton was introduced by Aichholzer et al. 1995 [1].
- Eppstein and Erickson [3] introduced in 1999 an algorithm with the current best worst-case complexity: For a simple polygon (with holes) it requires $\mathcal{O}(n^{17/11+\varepsilon})$ time and space to compute the (weighted) straight skeleton.
- More resent results with lower time/space-complexity are known [2, 6][1] but only for (unweighted) straight skeletons.

[1] Cheng, Mencel, and Vigneron as well as Vigneron and Yan.

Related Work & Overview

Related Work

- The straight skeleton was introduced by Aichholzer et al. 1995 [1].
- Eppstein and Erickson [3] introduced in 1999 an algorithm with the current best worst-case complexity: For a simple polygon (with holes) it requires $\mathcal{O}(n^{17/11+\varepsilon})$ time and space to compute the (weighted) straight skeleton.
- More resent results with lower time/space-complexity are known [2, 6][1] but only for (unweighted) straight skeletons.

[1] Cheng, Mencel, and Vigneron as well as Vigneron and Yan.

Related Work & Overview

Related Work

- The straight skeleton was introduced by Aichholzer et al. 1995 [1].
- Eppstein and Erickson [3] introduced in 1999 an algorithm with the current best worst-case complexity: For a simple polygon (with holes) it requires $\mathcal{O}(n^{17/11+\varepsilon})$ time and space to compute the (weighted) straight skeleton.
- More resent results with lower time/space-complexity are known [2, 6][1] but only for (unweighted) straight skeletons.

Overview

- Our work is based on the work of Huber and Held (IJCGA 2012) on straight-skeleton computation based on motorcycle graphs [4].
- Using an *extended wavefront* they transform split events into edge events, shifting the complexity to another event.
- We revisit their work and show required changes to apply their approach to the weighted scenario.

[1] Cheng, Mencel, and Vigneron as well as Vigneron and Yan.

Related Work & Overview

Related Work

- The straight skeleton was introduced by Aichholzer et al. 1995 [1].
- Eppstein and Erickson [3] introduced in 1999 an algorithm with the current best worst-case complexity: For a simple polygon (with holes) it requires $\mathcal{O}(n^{17/11+\varepsilon})$ time and space to compute the (weighted) straight skeleton.
- More resent results with lower time/space-complexity are known [2, 6][1] but only for (unweighted) straight skeletons.

Overview

- Our work is based on the work of Huber and Held (IJCGA 2012) on straight-skeleton computation based on motorcycle graphs [4].
- Using an *extended wavefront* they transform split events into edge events, shifting the complexity to another event.
- We revisit their work and show required changes to apply their approach to the weighted scenario.

[1] Cheng, Mencel, and Vigneron as well as Vigneron and Yan.

Related Work & Overview

Related Work

- The straight skeleton was introduced by Aichholzer et al. 1995 [1].
- Eppstein and Erickson [3] introduced in 1999 an algorithm with the current best worst-case complexity: For a simple polygon (with holes) it requires $\mathcal{O}(n^{17/11+\varepsilon})$ time and space to compute the (weighted) straight skeleton.
- More resent results with lower time/space-complexity are known [2, 6][1] but only for (unweighted) straight skeletons.

Overview

- Our work is based on the work of Huber and Held (IJCGA 2012) on straight-skeleton computation based on motorcycle graphs [4].
- Using an *extended wavefront* they transform split events into edge events, shifting the complexity to another event.
- We revisit their work and show required changes to apply their approach to the weighted scenario.

[1] Cheng, Mencel, and Vigneron as well as Vigneron and Yan.

Introduction

- Introduced by Aichholzer et al. 1995 [1].
- Consists only of straight line segments.
- Defined by a propagation process:
  - Edges move inwards in a parallel manner at unit speed.
  - The vertices of the *wavefront polygons* trace out *arcs*.
  - Two events: edge event and split event

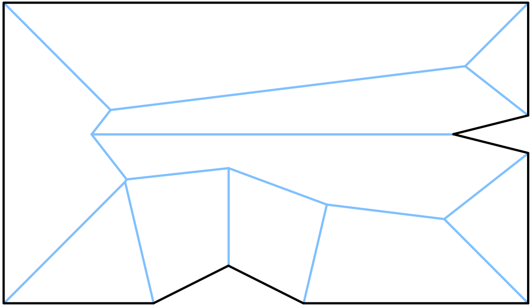


Image credit [5].

Introduction

- Introduced by Aichholzer et al. 1995 [1].
- Consists only of straight line segments.
- Defined by a propagation process:
  - Edges move inwards in a parallel manner at unit speed.
  - The vertices of the *wavefront polygons* trace out *arcs*.
  - Two events: edge event and split event

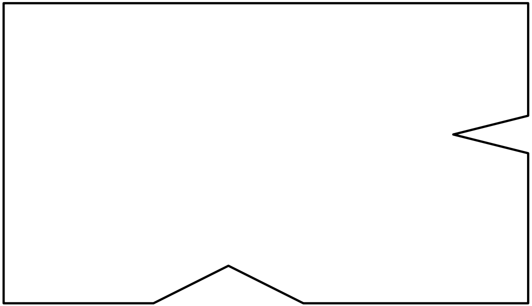


Image credit [5].

Introduction

- Introduced by Aichholzer et al. 1995 [1].
- Consists only of straight line segments.
- Defined by a propagation process:
  - Edges move inwards in a parallel manner at unit speed.
  - The vertices of the *wavefront polygons* trace out *arcs*.
  - Two events: edge event and split event

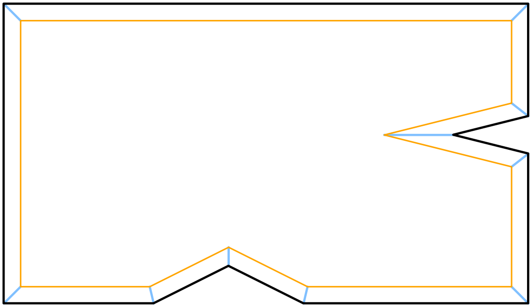


Image credit [5].

Introduction

- Introduced by Aichholzer et al. 1995 [1].
- Consists only of straight line segments.
- Defined by a propagation process:
  - Edges move inwards in a parallel manner at unit speed.
  - The vertices of the *wavefront polygons* trace out *arcs*.
  - Two events: edge event and split event

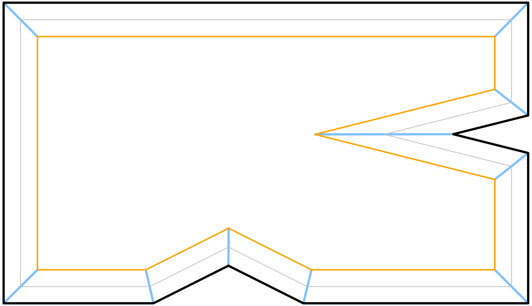


Image credit [5].

Introduction

- Introduced by Aichholzer et al. 1995 [1].
- Consists only of straight line segments.
- Defined by a propagation process:
  - Edges move inwards in a parallel manner at unit speed.
  - The vertices of the *wavefront polygons* trace out *arcs*.
  - Two events: edge event and split event

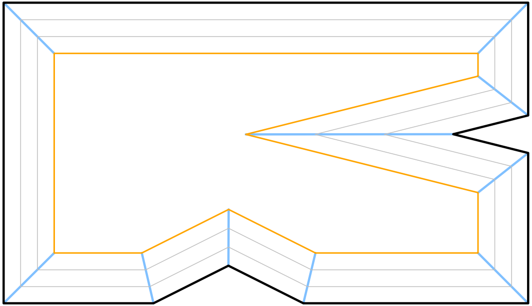


Image credit [5].

Introduction

- Introduced by Aichholzer et al. 1995 [1].
- Consists only of straight line segments.
- Defined by a propagation process:
  - Edges move inwards in a parallel manner at unit speed.
  - The vertices of the *wavefront polygons* trace out *arcs*.
  - Two events: edge event and split event

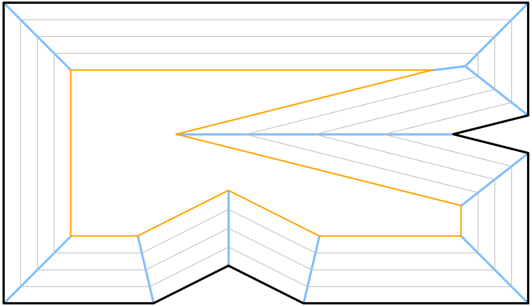


Image credit [5].

Introduction

- Introduced by Aichholzer et al. 1995 [1].
- Consists only of straight line segments.
- Defined by a propagation process:
  - Edges move inwards in a parallel manner at unit speed.
  - The vertices of the *wavefront polygons* trace out *arcs*.
  - Two events: edge event
  - Two events: edge event and split event

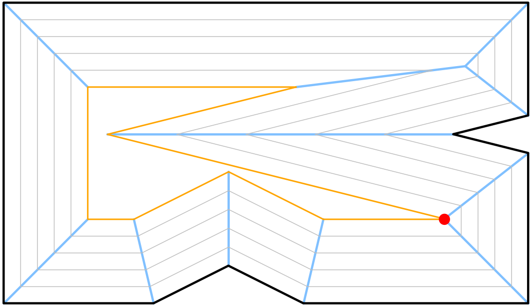


Image credit [5].

Introduction

- Introduced by Aichholzer et al. 1995 [1].
- Consists only of straight line segments.
- Defined by a propagation process:
  - Edges move inwards in a parallel manner at unit speed.
  - The vertices of the *wavefront polygons* trace out *arcs*.
  - Two events: edge event and split event
  - Two events: edge event and split event

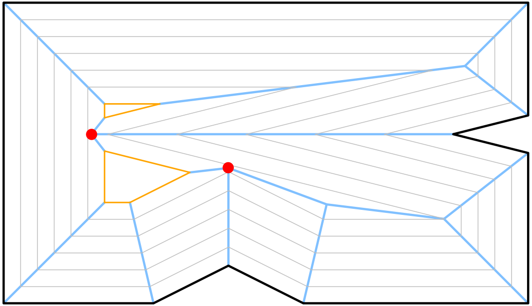


Image credit [5].

Introduction

- Introduced by Aichholzer et al. 1995 [1].
- Consists only of straight line segments.
- Defined by a propagation process:
  - Edges move inwards in a parallel manner at unit speed.
  - The vertices of the *wavefront polygons* trace out *arcs*.
  - Two events: edge event and split event

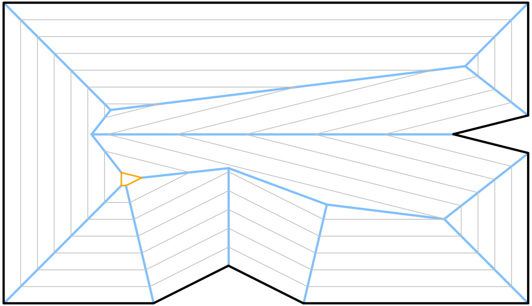


Image credit [5].

Introduction

- Introduced by Aichholzer et al. 1995 [1].
- Consists only of straight line segments.
- Defined by a propagation process:
  - Edges move inwards in a parallel manner at unit speed.
  - The vertices of the *wavefront polygons* trace out *arcs*.
  - Two events: edge event and split event

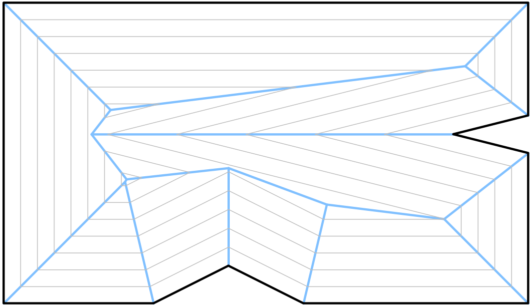


Image credit [5].

Introduction

- Introduced by Aichholzer et al. 1995 [1].
- Consists only of straight line segments.
- Defined by a propagation process:
  - Edges move inwards in a parallel manner at unit speed.
  - The vertices of the *wavefront polygons* trace out *arcs*.
  - Two events: edge event and split event
  - The *straight skeleton* consists of the blue arcs.

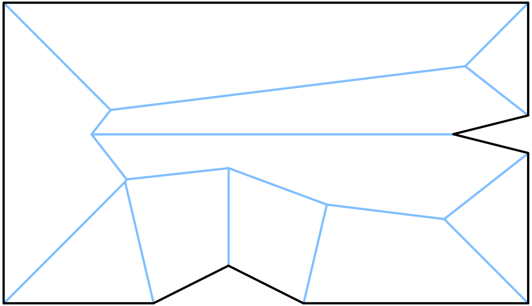
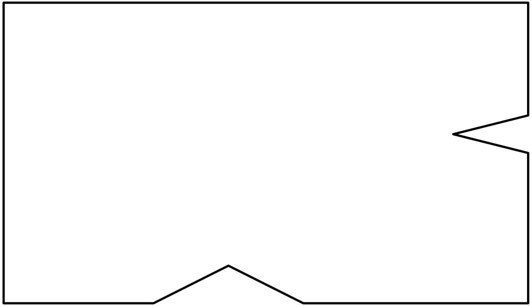


Image credit [5].

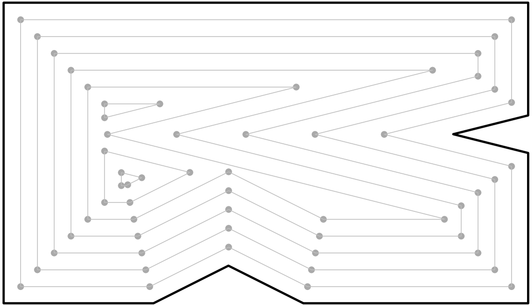
Preliminaries

- Let P a simple polygon with n vertices, r of which are reflex.
- Then $\mathcal{W}_P(t)$ is the wavefront of P at time t.
- After all components of $\mathcal{W}_P(t)$ have vanished we call the traces of the vertices of $\mathcal{W}_P(t)$ over the time-span of the propagation straight skeleton $\mathcal{S}(P)$.



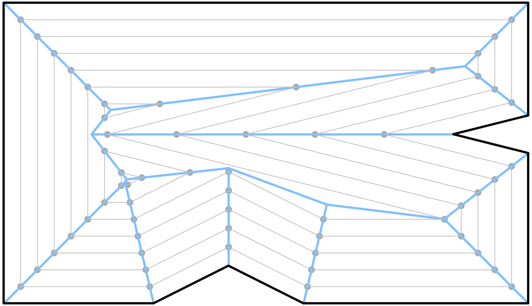
Preliminaries

- Let P a simple polygon with n vertices, r of which are reflex.
- Then $\mathcal{W}_P(t)$ is the wavefront of P at time t.
- After all components of $\mathcal{W}_P(t)$ have vanished we call the traces of the vertices of $\mathcal{W}_P(t)$ over the time-span of the propagation straight skeleton $\mathcal{S}(P)$.



Preliminaries

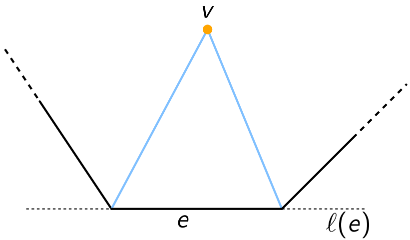
- Let P a simple polygon with n vertices, r of which are reflex.
- Then $\mathcal{W}_P(t)$ is the wavefront of P at time t.
- After all components of $\mathcal{W}_P(t)$ have vanished we call the traces of the vertices of $\mathcal{W}_P(t)$ over the time-span of the propagation straight skeleton $\mathcal{S}(P)$.



Straight Skeletons – Finding Events

Edge Events

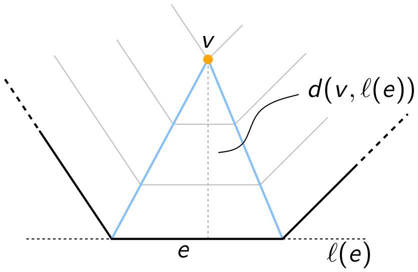
- Compute vanishing time for every edge if finite.
- Enqueue all events in priority queue.
- $\mathcal{O}(n \log n)$ time and $\mathcal{O}(n)$ space.



Straight Skeletons – Finding Events

Edge Events

- Compute vanishing time for every edge if finite.
- Enqueue all events in priority queue.
- $\mathcal{O}(n \log n)$ time and $\mathcal{O}(n)$ space.



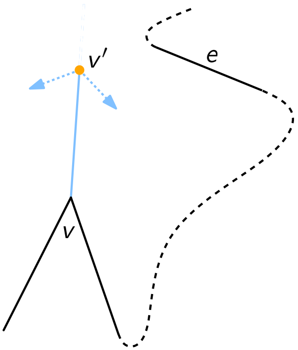
Straight Skeletons – Finding Events

Edge Events

- Compute vanishing time for every edge if finite.
- Enqueue all events in priority queue.
- $\mathcal{O}(n \log n)$ time and $\mathcal{O}(n)$ space.

Split Events

- Let v be a reflex vertex of P. Then there are $\mathcal{O}(n)$ possible split events for $v(t)$ in $\mathcal{W}_P(t)$.
- Enqueue event v' closest to v in $\mathcal{O}(n \log n)$ time.
- If v' is not reached again $\mathcal{O}(n \log n)$ for v''.
- We may miss $\mathcal{O}(n)$ times.
- $\mathcal{O}(n^2 r \log n)$ time and $\mathcal{O}(n)$ space.



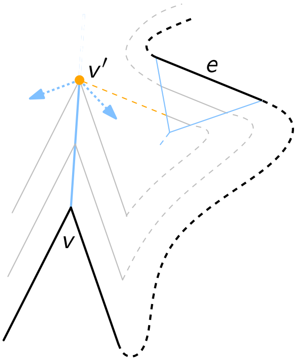
Straight Skeletons – Finding Events

Edge Events

- Compute vanishing time for every edge if finite.
- Enqueue all events in priority queue.
- $\mathcal{O}(n \log n)$ time and $\mathcal{O}(n)$ space.

Split Events

- Let v be a reflex vertex of P. Then there are $\mathcal{O}(n)$ possible split events for $v(t)$ in $\mathcal{W}_P(t)$.
- Enqueue event v' closest to v in $\mathcal{O}(n \log n)$ time.
- If v' is not reached again $\mathcal{O}(n \log n)$ for v''.
- We may miss $\mathcal{O}(n)$ times.
- $\mathcal{O}(n^2 r \log n)$ time and $\mathcal{O}(n)$ space.



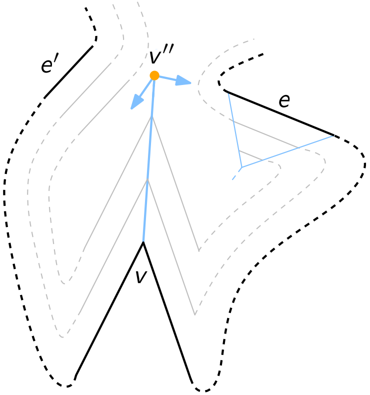
Straight Skeletons – Finding Events

Edge Events

- Compute vanishing time for every edge if finite.
- Enqueue all events in priority queue.
- $\mathcal{O}(n \log n)$ time and $\mathcal{O}(n)$ space.

Split Events

- Let v be a reflex vertex of P. Then there are $\mathcal{O}(n)$ possible split events for $v(t)$ in $\mathcal{W}_P(t)$.
- Enqueue event v' closest to v in $\mathcal{O}(n \log n)$ time.
- If v' is not reached again $\mathcal{O}(n \log n)$ for v''.
- We may miss $\mathcal{O}(n)$ times.
- $\mathcal{O}(n^2 r \log n)$ time and $\mathcal{O}(n)$ space.



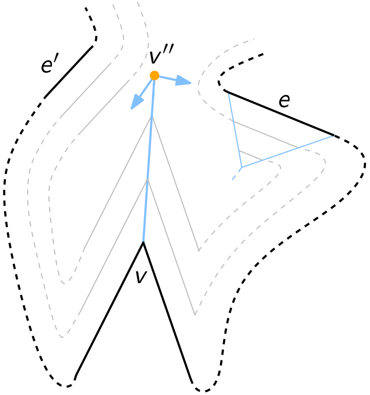
Straight Skeletons – Finding Events

Edge Events

- Compute vanishing time for every edge if finite.
- Enqueue all events in priority queue.
- $\mathcal{O}(n \log n)$ time and $\mathcal{O}(n)$ space.

Split Events

- Let v be a reflex vertex of P. Then there are $\mathcal{O}(n)$ possible split events for $v(t)$ in $\mathcal{W}_P(t)$.
- Enqueue event v' closest to v in $\mathcal{O}(n \log n)$ time.
- If v' is not reached again $\mathcal{O}(n \log n)$ for v''.
- We may miss $\mathcal{O}(n)$ times.
- $\mathcal{O}(n^2 r \log n)$ time and $\mathcal{O}(n)$ space.



Straight Skeletons – Finding Events

Edge Events

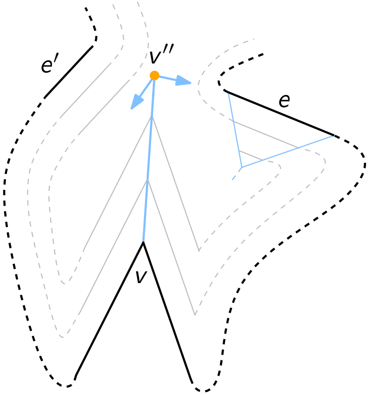
- Compute vanishing time for every edge if finite.
- Enqueue all events in priority queue.
- $\mathcal{O}(n \log n)$ time and $\mathcal{O}(n)$ space.

Split Events

- Let v be a reflex vertex of P. Then there are $\mathcal{O}(n)$ possible split events for $v(t)$ in $\mathcal{W}_P(t)$.
- Enqueue event v' closest to v in $\mathcal{O}(n \log n)$ time.
- If v' is not reached again $\mathcal{O}(n \log n)$ for v''.
- We may miss $\mathcal{O}(n)$ times.
- $\mathcal{O}(n^2 r \log n)$ time and $\mathcal{O}(n)$ space.

Multi-Split Event

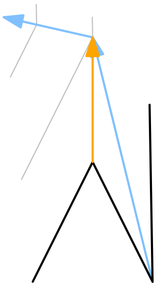
- When two reflex wavefront vertices meet at a common point[2].



[2] Related to vertex event [3].

Tracing Reflex Arcs

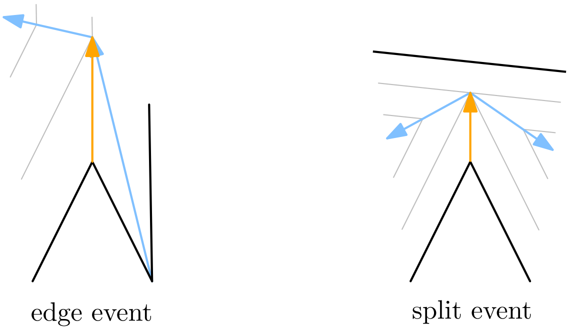
- Every event, where a reflex wavefront vertex is involved, reduces the number of reflex vertices in $\mathcal{W}_P(t)$.
- *Can we know these reflex arcs in advance?*



edge event

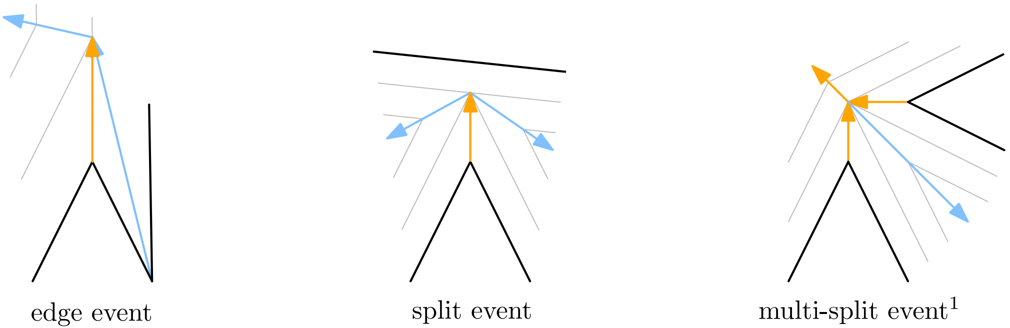
Tracing Reflex Arcs

- Every event, where a reflex wavefront vertex is involved, reduces the number of reflex vertices in $\mathcal{W}_P(t)$.
- *Can we know these reflex arcs in advance?*



Tracing Reflex Arcs

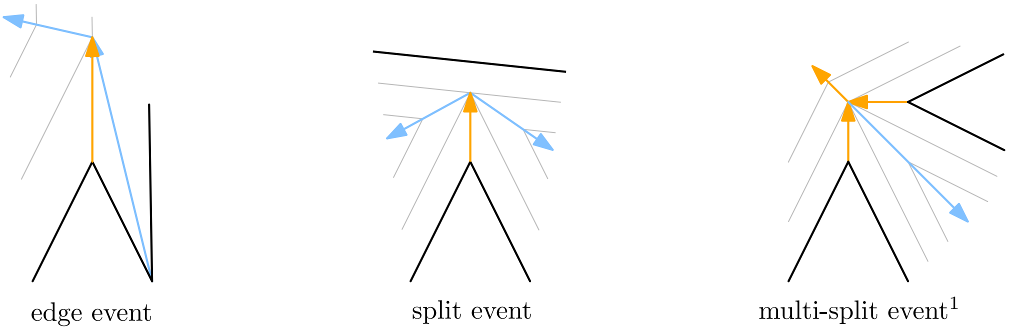
- Every event, where a reflex wavefront vertex is involved, reduces the number of reflex vertices in $\mathcal{W}_P(t)$.
- *Can we know these reflex arcs in advance?*



[1] Also called vertex event [3].

Tracing Reflex Arcs

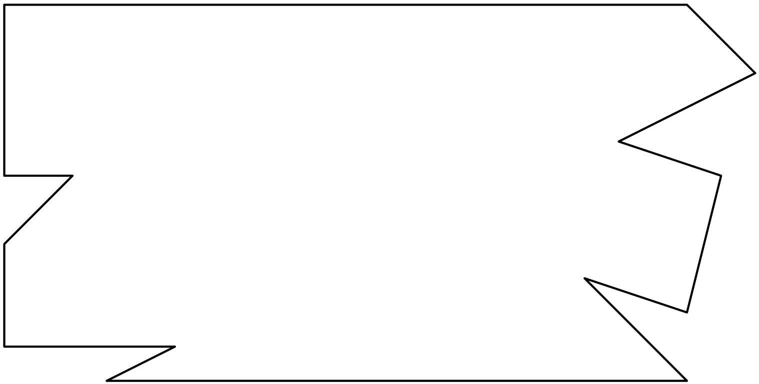
- Every event, where a reflex wavefront vertex is involved, reduces the number of reflex vertices in $\mathcal{W}_P(t)$.
- *Can we know these reflex arcs in advance?*



[1] Also called vertex event [3].

Motorcycle Graph and $\mathcal{S}(P)$

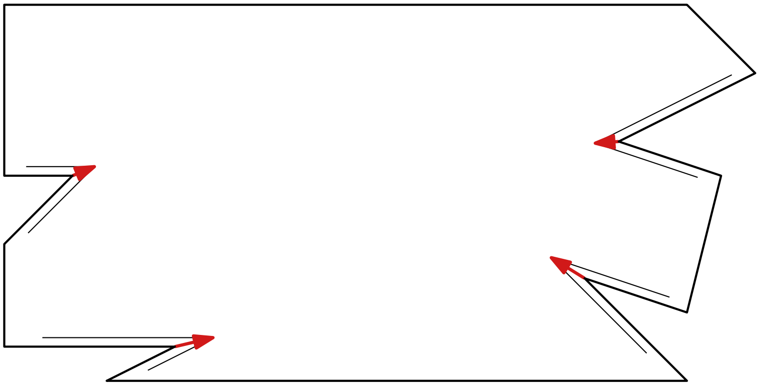
- Start a motorcycle m for every reflex vertex v of $\mathcal{W}_P(t)$ such that m inherits the velocity vector of v.
- Every motorcycle leaves a trace behind and stops if it crashes into another trace or the polygon boundary.
- The *motorcycle graph* $\mathcal{M}(P)$[3] is formed when all motorcycles have crashed.



[3] Introduced by Eppstein and Erickson [3]

Motorcycle Graph and $\mathcal{S}(P)$

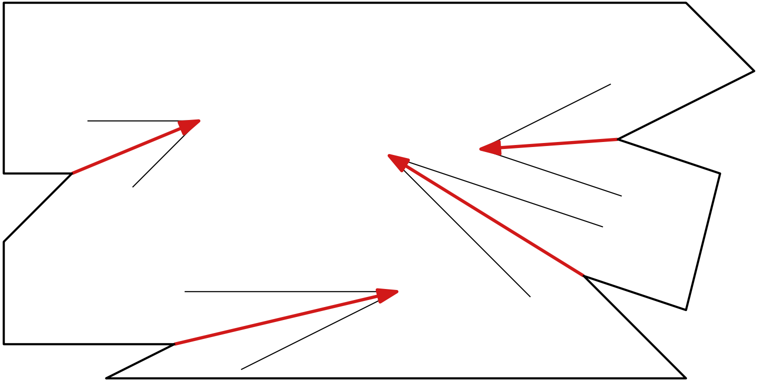
- Start a motorcycle m for every reflex vertex v of $\mathcal{W}_P(t)$ such that m inherits the velocity vector of v.
- Every motorcycle leaves a trace behind and stops if it crashes into another trace or the polygon boundary.
- The *motorcycle graph* $\mathcal{M}(P)$[3] is formed when all motorcycles have crashed.



[3]Introduced by Eppstein and Erickson [3]

Motorcycle Graph and $\mathcal{S}(P)$

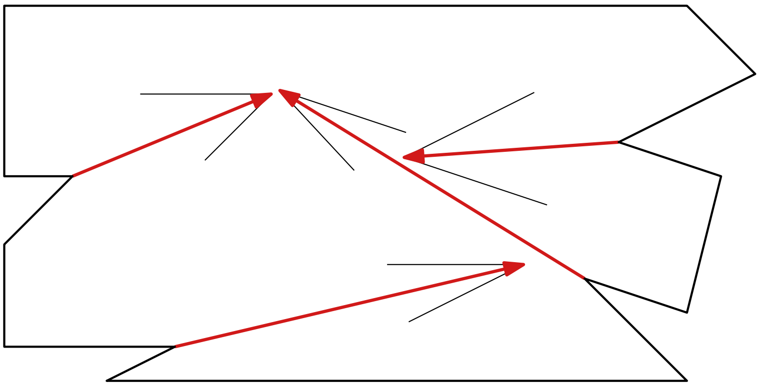
- Start a motorcycle m for every reflex vertex v of $\mathcal{W}_P(t)$ such that m inherits the velocity vector of v.
- Every motorcycle leaves a trace behind and stops if it crashes into another trace or the polygon boundary.
- The *motorcycle graph* $\mathcal{M}(P)$[3] is formed when all motorcycles have crashed.



[3] Introduced by Eppstein and Erickson [3]

Motorcycle Graph and $\mathcal{S}(P)$

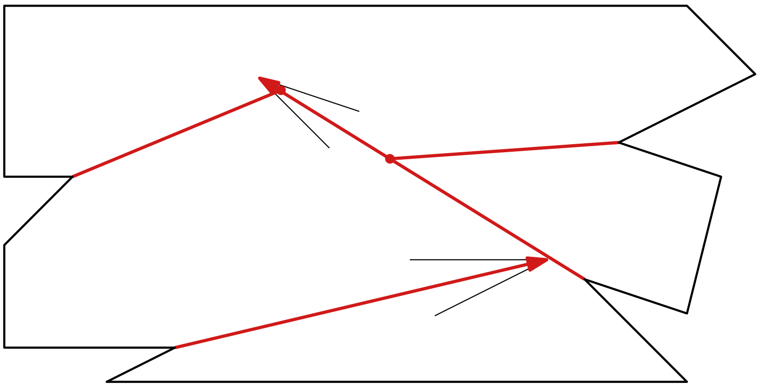
- Start a motorcycle m for every reflex vertex v of $\mathcal{W}_P(t)$ such that m inherits the velocity vector of v.
- Every motorcycle leaves a trace behind and stops if it crashes into another trace or the polygon boundary.
- The *motorcycle graph* $\mathcal{M}(P)$[3] is formed when all motorcycles have crashed.



[3] Introduced by Eppstein and Erickson [3]

Motorcycle Graph and $\mathcal{S}(P)$

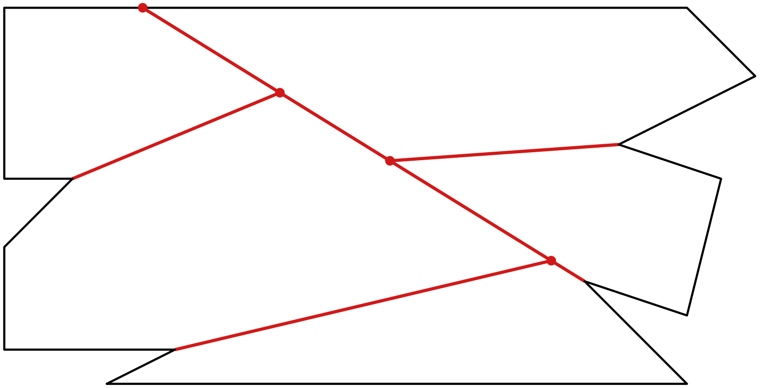
- Start a motorcycle m for every reflex vertex v of $\mathcal{W}_P(t)$ such that m inherits the velocity vector of v.
- Every motorcycle leaves a trace behind and stops if it crashes into another trace or the polygon boundary.
- The *motorcycle graph* $\mathcal{M}(P)$[3] is formed when all motorcycles have crashed.



[3] Introduced by Eppstein and Erickson [3]

Motorcycle Graph and $\mathcal{S}(P)$

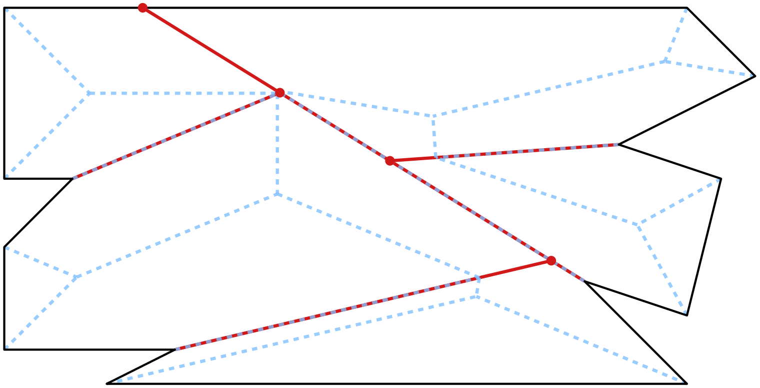
- Start a motorcycle m for every reflex vertex v of $\mathcal{W}_P(t)$ such that m inherits the velocity vector of v.
- Every motorcycle leaves a trace behind and stops if it crashes into another trace or the polygon boundary.
- The *motorcycle graph* $\mathcal{M}(P)$[3] is formed when all motorcycles have crashed.



[3] Introduced by Eppstein and Erickson [3]

Motorcycle Graph and $\mathcal{S}(P)$ (cont'd)

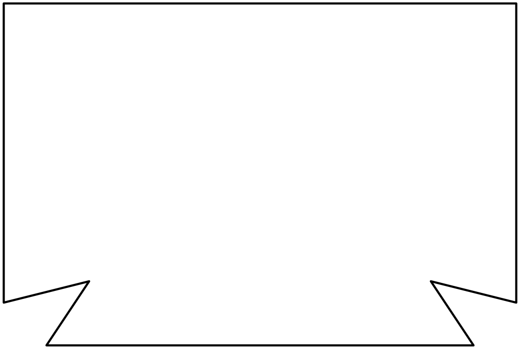
- $\mathcal{M}'(P)$ covers all *reflex arcs* of $\mathcal{S}(P)$[4].



[4] Various publications [2, 3]

Motorcycle Graph and $\mathcal{S}(P)$ (cont'd)

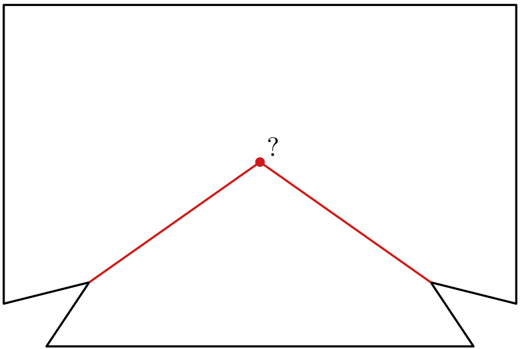
- $\mathcal{M}'(P)$ covers all *reflex arcs* of $\mathcal{S}(P)$[4].



[4] Various publications [2, 3]

Motorcycle Graph and $\mathcal{S}(P)$ (cont'd)

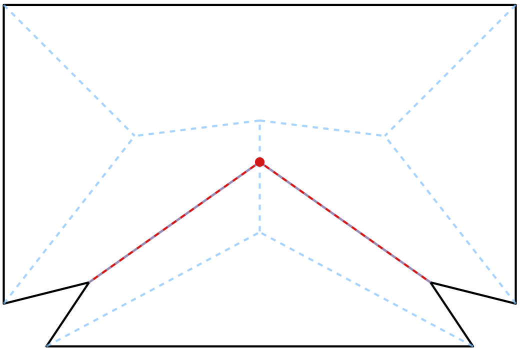
- $\mathcal{M}'(P)$ covers "all" *reflex arcs* of $\mathcal{S}(P)$[4], such that no two motorcycles reach the same point simultaneously.
- Huber and Held [4] allow motorcycles to have different starting times and call it *generalized motorcycle graph* $\mathcal{M}(P)$.



[4] Various publications [2, 3]

Motorcycle Graph and $\mathcal{S}(P)$ (cont'd)

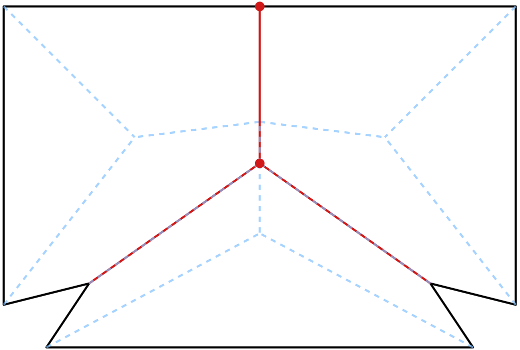
- $\mathcal{M}'(P)$ covers "all" *reflex arcs* of $\mathcal{S}(P)$[4], such that no two motorcycles reach the same point simultaneously.
- Huber and Held [4] allow motorcycles to have different starting times and call it *generalized motorcycle graph* $\mathcal{M}(P)$.



[4] Various publications [2, 3]

Motorcycle Graph and $\mathcal{S}(P)$ (cont'd)

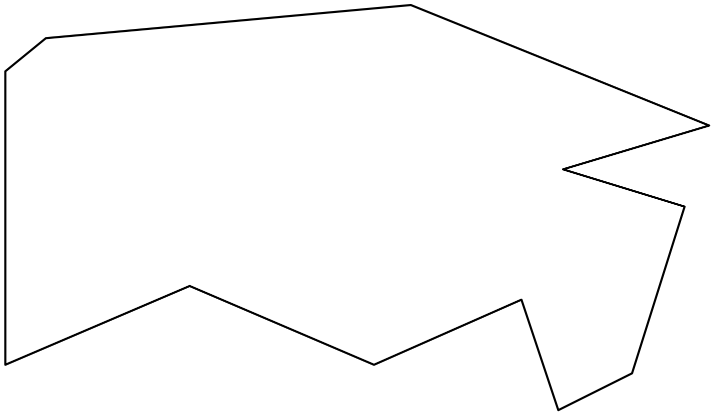
- $\mathcal{M}'(P)$ covers "all" *reflex arcs* of $\mathcal{S}(P)$[4], such that no two motorcycles reach the same point simultaneously.
- Huber and Held [4] allow motorcycles to have different starting times and call it *generalized motorcycle graph* $\mathcal{M}(P)$.



[4] Various publications [2, 3]

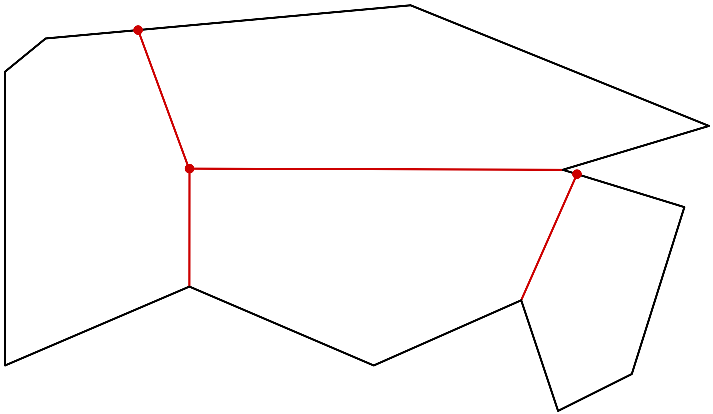
Extended Wavefront $\mathcal{W}_P^*$

- Huber and Held introduce the *extended* wavefront $\mathcal{W}_P^*(t)$.
- $\mathcal{W}_P^*(t)$ consists of the wavefront at time t and the portion of $\mathcal{M}(P)$ that lies inside that wavefront.
- All regions in $\mathcal{W}_P^*(t)$ are convex and all events are edge events.



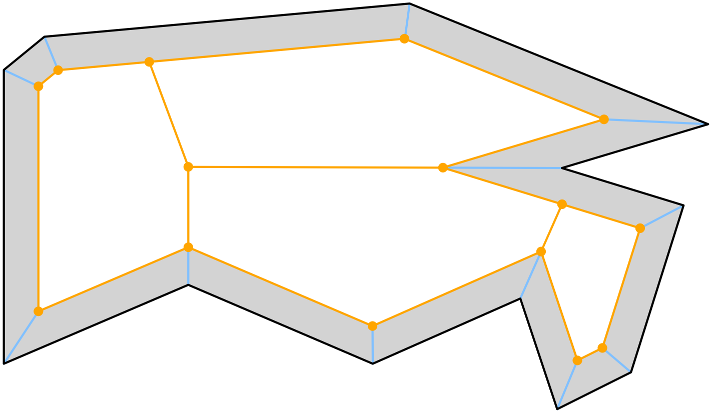
Extended Wavefront $\mathcal{W}_P^*$

- Huber and Held introduce the *extended* wavefront $\mathcal{W}_P^*(t)$.
- $\mathcal{W}_P^*(t)$ consists of the wavefront at time t and the portion of $\mathcal{M}(P)$ that lies inside that wavefront.
- All regions in $\mathcal{W}_P^*(t)$ are convex and all events are edge events.



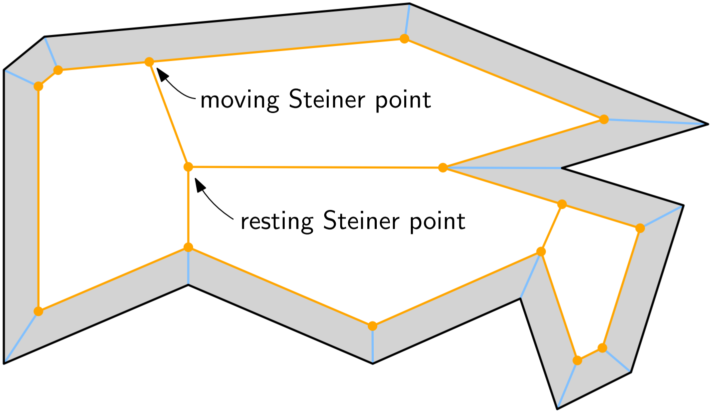
Extended Wavefront $\mathcal{W}_P^*$

- Huber and Held introduce the *extended* wavefront $\mathcal{W}_P^*(t)$.
- $\mathcal{W}_P^*(t)$ consists of the wavefront at time t and the portion of $\mathcal{M}(P)$ that lies inside that wavefront.
- All regions in $\mathcal{W}_P^*(t)$ are convex and all events are edge events.



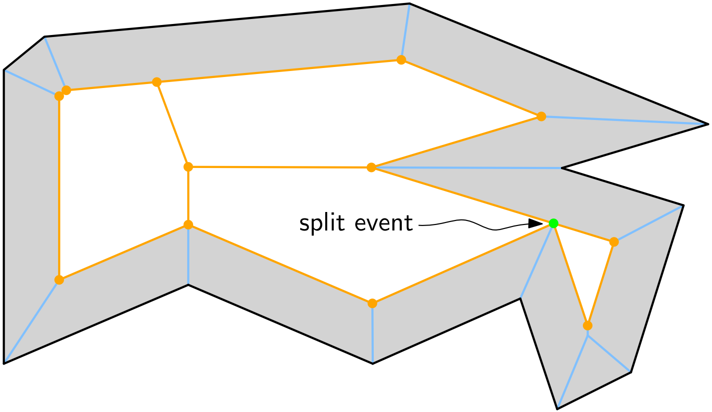
Extended Wavefront $\mathcal{W}_P^*$

- Huber and Held introduce the *extended* wavefront $\mathcal{W}_P^*(t)$.
- $\mathcal{W}_P^*(t)$ consists of the wavefront at time t and the portion of $\mathcal{M}(P)$ that lies inside that wavefront.
- All regions in $\mathcal{W}_P^*(t)$ are convex and all events are edge events.



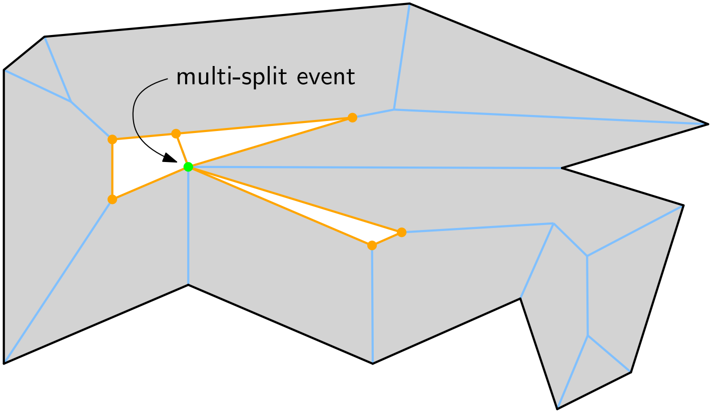
Extended Wavefront $\mathcal{W}_P^*$

- Huber and Held introduce the *extended* wavefront $\mathcal{W}_P^*(t)$.
- $\mathcal{W}_P^*(t)$ consists of the wavefront at time t and the portion of $\mathcal{M}(P)$ that lies inside that wavefront.
- All regions in $\mathcal{W}_P^*(t)$ are convex and all events are edge events.



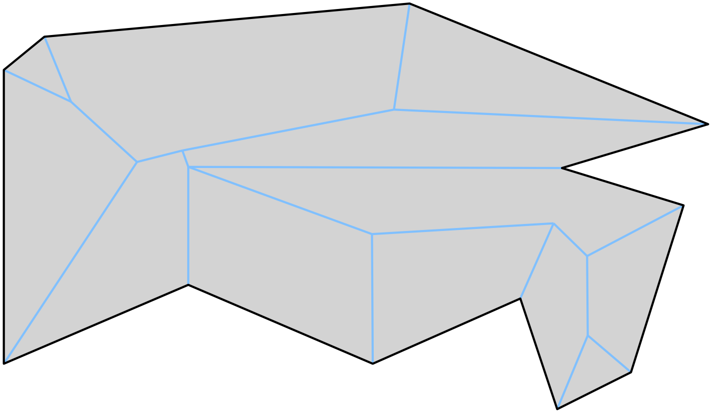
Extended Wavefront $\mathcal{W}_P^*$

- Huber and Held introduce the *extended* wavefront $\mathcal{W}_P^*(t)$.
- $\mathcal{W}_P^*(t)$ consists of the wavefront at time t and the portion of $\mathcal{M}(P)$ that lies inside that wavefront.
- All regions in $\mathcal{W}_P^*(t)$ are convex and all events are edge events.



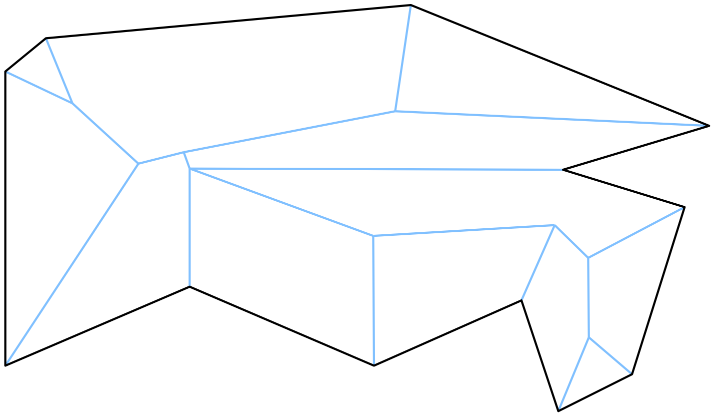
Extended Wavefront $\mathcal{W}_P^*$

- Huber and Held introduce the *extended* wavefront $\mathcal{W}_P^*(t)$.
- $\mathcal{W}_P^*(t)$ consists of the wavefront at time t and the portion of $\mathcal{M}(P)$ that lies inside that wavefront.
- All regions in $\mathcal{W}_P^*(t)$ are convex and all events are edge events.



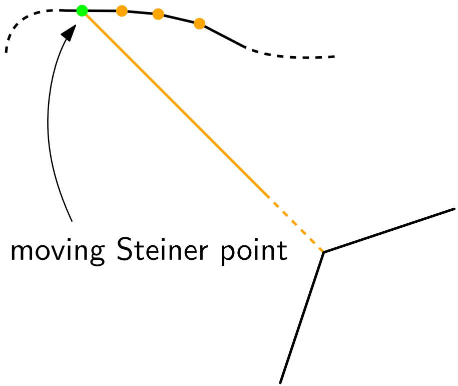
Extended Wavefront $\mathcal{W}_P^*$

- Huber and Held introduce the *extended* wavefront $\mathcal{W}_P^*(t)$.
- $\mathcal{W}_P^*(t)$ consists of the wavefront at time t and the portion of $\mathcal{M}(P)$ that lies inside that wavefront.
- All regions in $\mathcal{W}_P^*(t)$ are convex and all events are edge events.



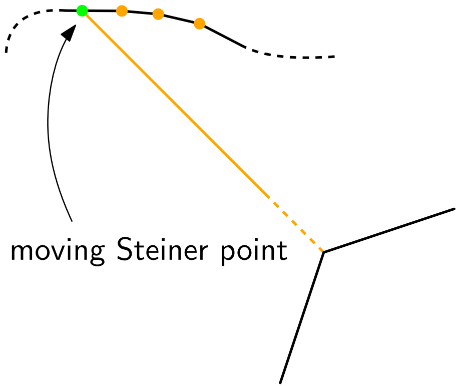
Extended Wavefront $\mathcal{W}_P^*$ (cont'd)

- Using $\mathcal{W}_P^*(t)$ one can compute $\mathcal{S}(P)$ in $\mathcal{O}((n + nr) \log n)$ time and linear space.
- The number of *switch events*, i.e., when a wavefront vertex meets a moving Steiner point (where the arc of a motorcycle edge ends) is in $\mathcal{O}(nr)$.



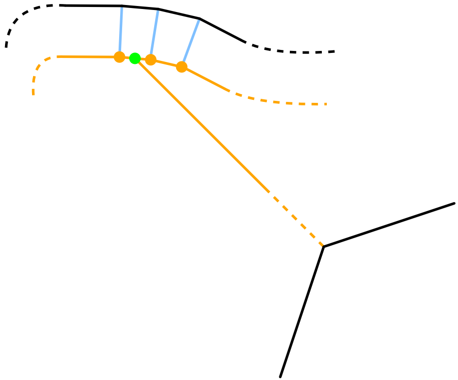
Extended Wavefront $\mathcal{W}_P^*$ (cont'd)

- Using $\mathcal{W}_P^*(t)$ one can compute $\mathcal{S}(P)$ in $\mathcal{O}((n + nr)\log n)$ time and linear space.
- The number of *switch events*, i.e., when a wavefront vertex meets a moving Steiner point (where the arc of a motorcycle edge ends) is in $\mathcal{O}(nr)$.



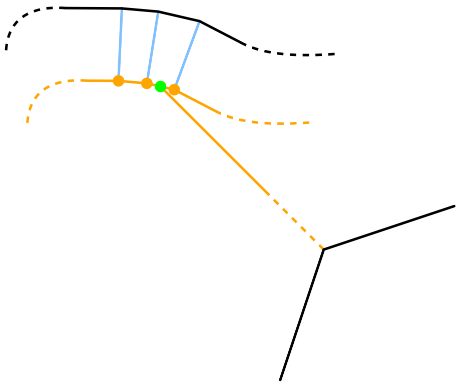
Extended Wavefront $\mathcal{W}_P^*$ (cont'd)

- Using $\mathcal{W}_P^*(t)$ one can compute $\mathcal{S}(P)$ in $\mathcal{O}((n + nr) \log n)$ time and linear space.
- The number of *switch events*, i.e., when a wavefront vertex meets a moving Steiner point (where the arc of a motorcycle edge ends) is in $\mathcal{O}(nr)$.



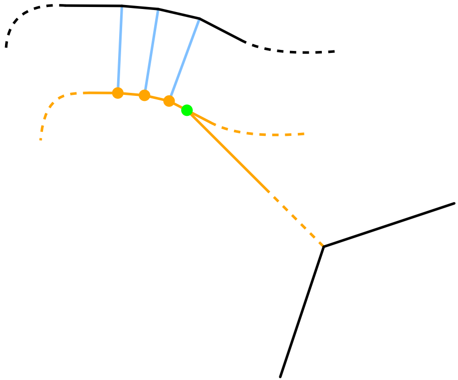
Extended Wavefront $\mathcal{W}_P^*$ (cont'd)

- Using $\mathcal{W}_P^*(t)$ one can compute $\mathcal{S}(P)$ in $\mathcal{O}((n + nr) \log n)$ time and linear space.
- The number of *switch events*, i.e., when a wavefront vertex meets a moving Steiner point (where the arc of a motorcycle edge ends) is in $\mathcal{O}(nr)$.



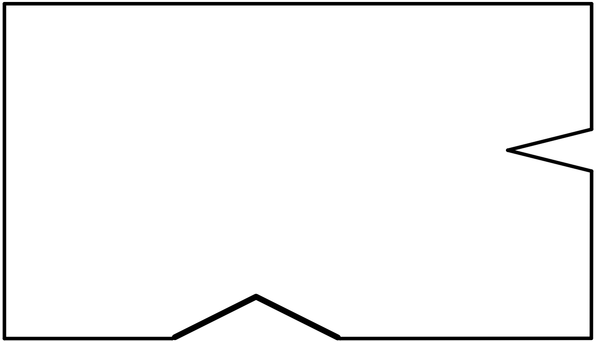
Extended Wavefront $\mathcal{W}_P^*$ (cont'd)

- Using $\mathcal{W}_P^*(t)$ one can compute $\mathcal{S}(P)$ in $\mathcal{O}((n + nr)\log n)$ time and linear space.
- The number of *switch events*, i.e., when a wavefront vertex meets a moving Steiner point (where the arc of a motorcycle edge ends) is in $\mathcal{O}(nr)$.



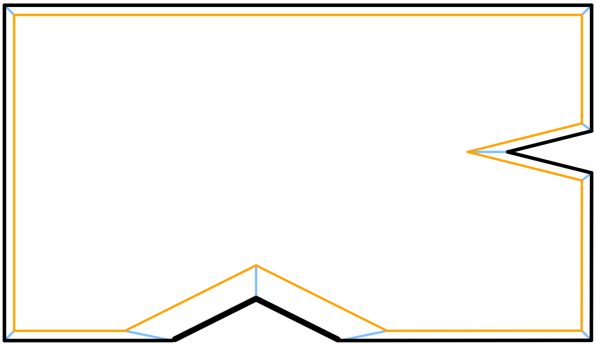
Weighted Straight Skeleton $\mathcal{S}(P, \sigma)$

- Wavefront $\mathcal{W}_P(t, \sigma)$ traces out the weighted straight skeleton $\mathcal{S}(P, \sigma)$, s.t. the weight function $\sigma \in \mathbb{R}^+$ provides the strictly positive edge weights.
- Thick edges have σ of about 3, unit weight otherwise.
- $\mathcal{S}(P)$ (left) and $\mathcal{S}(P, \sigma)$ (right).



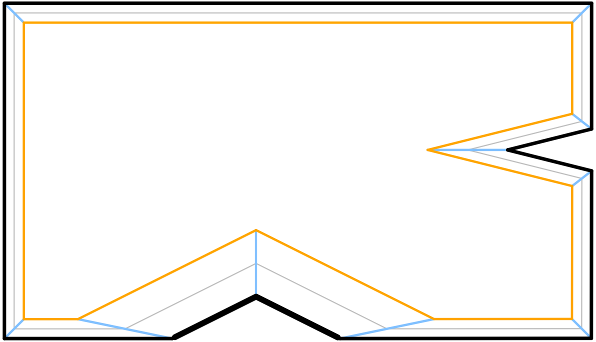
Weighted Straight Skeleton $\mathcal{S}(P,\sigma)$

- Wavefront $\mathcal{W}_P(t,\sigma)$ traces out the weighted straight skeleton $\mathcal{S}(P,\sigma)$, s.t. the weight function $\sigma \in \mathbb{R}^+$ provides the strictly positive edge weights.
- Thick edges have σ of about 3, unit weight otherwise.
- $\mathcal{S}(P)$ (left) and $\mathcal{S}(P,\sigma)$ (right).



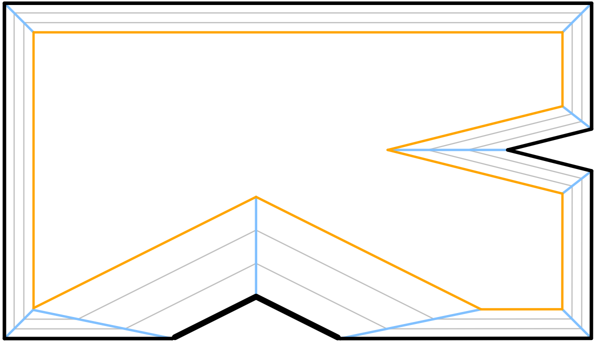
Weighted Straight Skeleton $\mathcal{S}(P, \sigma)$

- Wavefront $\mathcal{W}_P(t, \sigma)$ traces out the weighted straight skeleton $\mathcal{S}(P, \sigma)$, s.t. the weight function $\sigma \in \mathbb{R}^+$ provides the strictly positive edge weights.
- Thick edges have σ of about 3, unit weight otherwise.
- $\mathcal{S}(P)$ (left) and $\mathcal{S}(P, \sigma)$ (right).



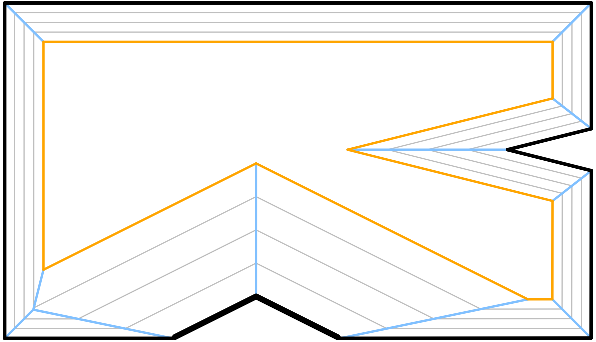
Weighted Straight Skeleton $\mathcal{S}(P, \sigma)$

- Wavefront $\mathcal{W}_P(t, \sigma)$ traces out the weighted straight skeleton $\mathcal{S}(P, \sigma)$, s.t. the weight function $\sigma \in \mathbb{R}^+$ provides the strictly positive edge weights.
- Thick edges have σ of about 3, unit weight otherwise.
- $\mathcal{S}(P)$ (left) and $\mathcal{S}(P, \sigma)$ (right).



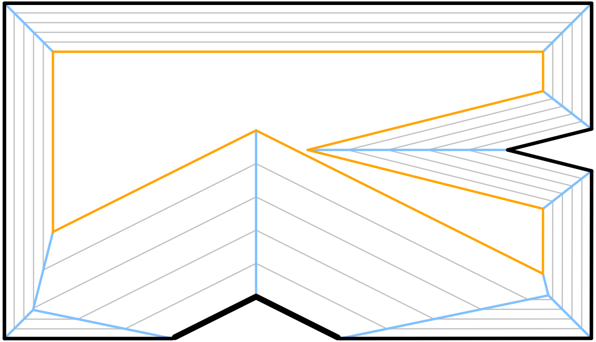
Weighted Straight Skeleton $\mathcal{S}(P, \sigma)$

- Wavefront $\mathcal{W}_P(t, \sigma)$ traces out the weighted straight skeleton $\mathcal{S}(P, \sigma)$, s.t. the weight function $\sigma \in \mathbb{R}^+$ provides the strictly positive edge weights.
- Thick edges have σ of about 3, unit weight otherwise.
- $\mathcal{S}(P)$ (left) and $\mathcal{S}(P, \sigma)$ (right).



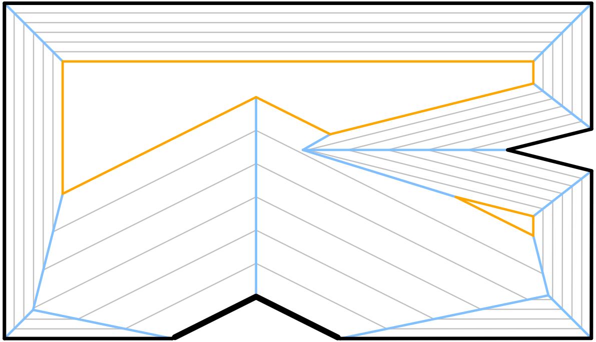
Weighted Straight Skeleton $\mathcal{S}(P, \sigma)$

- Wavefront $\mathcal{W}_P(t, \sigma)$ traces out the weighted straight skeleton $\mathcal{S}(P, \sigma)$, s.t. the weight function $\sigma \in \mathbb{R}^+$ provides the strictly positive edge weights.
- Thick edges have σ of about 3, unit weight otherwise.
- $\mathcal{S}(P)$ (left) and $\mathcal{S}(P, \sigma)$ (right).



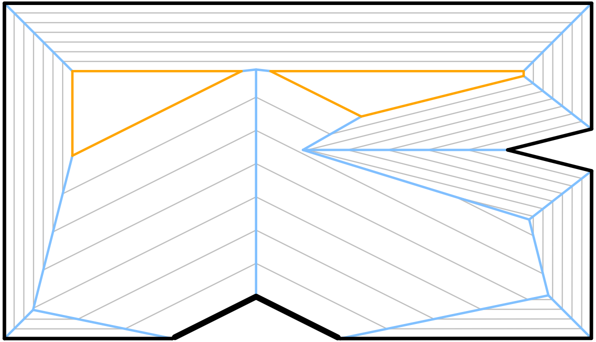
Weighted Straight Skeleton $\mathcal{S}(P, \sigma)$

- Wavefront $\mathcal{W}_P(t, \sigma)$ traces out the weighted straight skeleton $\mathcal{S}(P, \sigma)$, s.t. the weight function $\sigma \in \mathbb{R}^+$ provides the strictly positive edge weights.
- Thick edges have σ of about 3, unit weight otherwise.
- $\mathcal{S}(P)$ (left) and $\mathcal{S}(P, \sigma)$ (right).



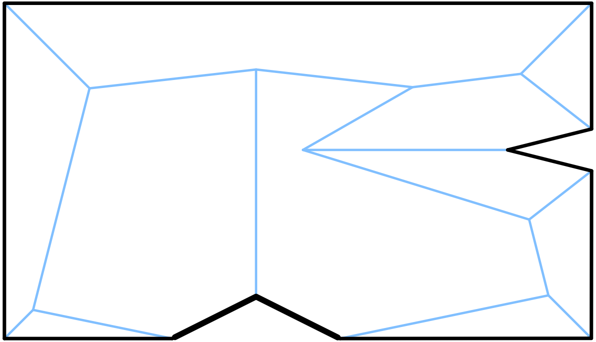
Weighted Straight Skeleton $\mathcal{S}(P, \sigma)$

- Wavefront $\mathcal{W}_P(t, \sigma)$ traces out the weighted straight skeleton $\mathcal{S}(P, \sigma)$, s.t. the weight function $\sigma \in \mathbb{R}^+$ provides the strictly positive edge weights.
- Thick edges have σ of about 3, unit weight otherwise.
- $\mathcal{S}(P)$ (left) and $\mathcal{S}(P, \sigma)$ (right).



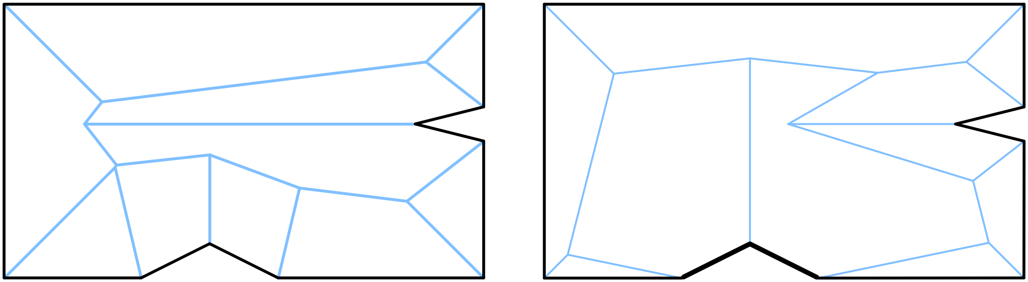
Weighted Straight Skeleton $\mathcal{S}(P, \sigma)$

- Wavefront $\mathcal{W}_P(t, \sigma)$ traces out the weighted straight skeleton $\mathcal{S}(P, \sigma)$, s.t. the weight function $\sigma \in \mathbb{R}^+$ provides the strictly positive edge weights.
- Thick edges have σ of about 3, unit weight otherwise.
- $\mathcal{S}(P)$ (left) and $\mathcal{S}(P, \sigma)$ (right).



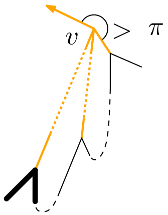
Weighted Straight Skeleton $\mathcal{S}(P, \sigma)$

- Wavefront $\mathcal{W}_P(t, \sigma)$ traces out the weighted straight skeleton $\mathcal{S}(P, \sigma)$, s.t. the weight function $\sigma \in \mathbb{R}^+$ provides the strictly positive edge weights.
- Thick edges have σ of about 3, unit weight otherwise.
- $\mathcal{S}(P)$ (left) and $\mathcal{S}(P, \sigma)$ (right).



Weighted Straight Skeleton (cont'd)

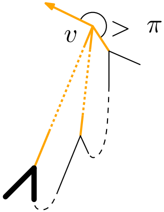
- Reflex vertices of $\mathcal{W}(P, \sigma)$ are not limited by first event.
- $\mathcal{M}(P)$ covers only initial *reflex* arcs of $\mathcal{S}(P, \sigma)$.
- $\mathcal{M}(P)$ may have to be updated n times.
- Updating $\mathcal{M}(P)$ results in re-computation.



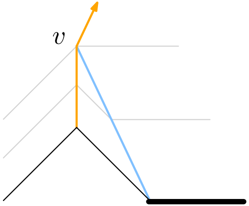
mutli-split event

Weighted Straight Skeleton (cont'd)

- Reflex vertices of $\mathcal{W}(P, \sigma)$ are not limited by first event.
- $\mathcal{M}(P)$ covers only initial *reflex* arcs of $\mathcal{S}(P, \sigma)$.
- $\mathcal{M}(P)$ may have to be updated n times.
- Updating $\mathcal{M}(P)$ results in re-computation.



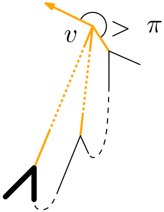
mutli-split event



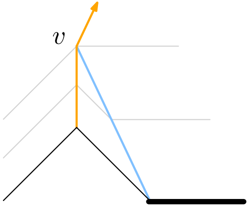
edge event

Weighted Straight Skeleton (cont'd)

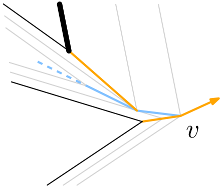
- Reflex vertices of $\mathcal{W}(P, \sigma)$ are not limited by first event.
- $\mathcal{M}(P)$ covers only initial *reflex* arcs of $\mathcal{S}(P, \sigma)$.
- $\mathcal{M}(P)$ may have to be updated n times.
- Updating $\mathcal{M}(P)$ results in re-computation.



mutli-split event



edge event

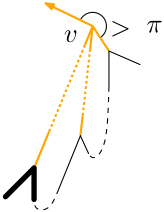


edge event[1]

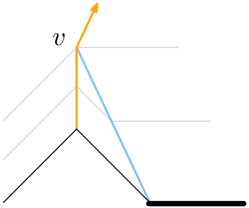
[1] after a split event.

Weighted Straight Skeleton (cont'd)

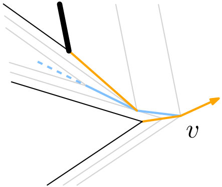
- Reflex vertices of $\mathcal{W}(P, \sigma)$ are not limited by first event.
- $\mathcal{M}(P)$ covers only initial *reflex* arcs of $\mathcal{S}(P, \sigma)$.
- $\mathcal{M}(P)$ may have to be updated n times.
- Updating $\mathcal{M}(P)$ results in re-computation.



mutli-split event



edge event

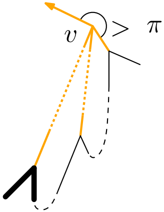


edge event[1]

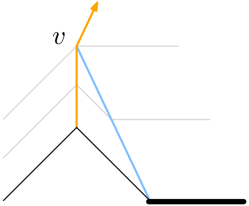
[1] after a split event.

Weighted Straight Skeleton (cont'd)

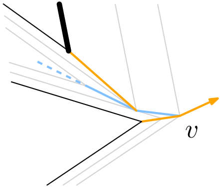
- Reflex vertices of $\mathcal{W}(P, \sigma)$ are not limited by first event.
- $\mathcal{M}(P)$ covers only initial *reflex* arcs of $\mathcal{S}(P, \sigma)$.
- $\mathcal{M}(P)$ may have to be updated n times.
- Updating $\mathcal{M}(P)$ results in re-computation.



mutli-split event



edge event



edge event[1]

[1] after a split event.

Induced Line Arrangement $\mathcal{A}(P)$

- For every reflex vertex v of P we construct a line segment s.
- Let s start at v, lie on $v(t)$ of $\mathcal{W}_P(t, \sigma)$, and end at the first intersection with the boundary of P.
- The set of all such line segments forms $\mathcal{A}(P)$.

Induced Line Arrangement $\mathcal{A}(P)$

- For every reflex vertex v of P we construct a line segment s.
- Let s start at v, lie on $v(t)$ of $\mathcal{W}_P(t, \sigma)$, and end at the first intersection with the boundary of P.
- The set of all such line segments forms $\mathcal{A}(P)$.

Induced Line Arrangement $\mathcal{A}(P)$

- For every reflex vertex v of P we construct a line segment s.
- Let s start at v, lie on $v(t)$ of $\mathcal{W}_P(t, \sigma)$, and end at the first intersection with the boundary of P.
- The set of all such line segments forms $\mathcal{A}(P)$.

Induced Line Arrangement $\mathcal{A}(P)$

- For every reflex vertex v of P we construct a line segment s.
- Let s start at v, lie on $v(t)$ of $\mathcal{W}_P(t,\sigma)$, and end at the first intersection with the boundary of P.
- The set of all such line segments forms $\mathcal{A}(P)$.

Linear or Quadratic Space

- Store all $\mathcal{O}(r^2)$ intersections in a sorted manner: $\mathcal{O}(n^2 \log n)$ time and $\mathcal{O}(r^2)$ space. Obtain the next intersection in $\mathcal{O}(1)$ time.
- Store only the closest intersection to a point for all segments in $\mathcal{A}(P)$, i.e., overall $\mathcal{O}(r)$ intersections. Obtain the next intersection in $\mathcal{O}(r)$ time.

Induced Line Arrangement $\mathcal{A}(P)$

- For every reflex vertex v of P we construct a line segment s.
- Let s start at v, lie on $v(t)$ of $\mathcal{W}_P(t,\sigma)$, and end at the first intersection with the boundary of P.
- The set of all such line segments forms $\mathcal{A}(P)$.

Linear or Quadratic Space

- Store all $\mathcal{O}(r^2)$ intersections in a sorted manner: $\mathcal{O}(n^2 \log n)$ time and $\mathcal{O}(r^2)$ space. Obtain the next intersection in $\mathcal{O}(1)$ time.
- Store only the closest intersection to a point for all segments in $\mathcal{A}(P)$, i.e., overall $\mathcal{O}(r)$ intersections. Obtain the next intersection in $\mathcal{O}(r)$ time.

Induced Line Arrangement $\mathcal{A}(P)$

- For every reflex vertex v of P we construct a line segment s.
- Let s start at v, lie on $v(t)$ of $\mathcal{W}_P(t, \sigma)$, and end at the first intersection with the boundary of P.
- The set of all such line segments forms $\mathcal{A}(P)$.

Linear or Quadratic Space

- Store all $\mathcal{O}(r^2)$ intersections in a sorted manner: $\mathcal{O}(n^2 \log n)$ time and $\mathcal{O}(r^2)$ space. Obtain the next intersection in $\mathcal{O}(1)$ time.
- Store only the closest intersection to a point for all segments in $\mathcal{A}(P)$, i.e., overall $\mathcal{O}(r)$ intersections. Obtain the next intersection in $\mathcal{O}(r)$ time.

Space Time Trade-off

- For a fixed k in $1 \leq k \leq r$. Let s in $\mathcal{A}(P)$ and let p the next intersection point on s.
- Instead of p we compute and store the next k intersections in $\mathcal{O}(r \log r)$ time.
- On s are at most r intersections points, thus we have to compute the next k intersections at most r/k times.
- We require $\mathcal{O}(kr)$ space for $\mathcal{A}(P)$.

Induced Line Arrangement $\mathcal{A}(P)$

- For every reflex vertex v of P we construct a line segment s.
- Let s start at v, lie on $v(t)$ of $\mathcal{W}_P(t, \sigma)$, and end at the first intersection with the boundary of P.
- The set of all such line segments forms $\mathcal{A}(P)$.

Linear or Quadratic Space

- Store all $\mathcal{O}(r^2)$ intersections in a sorted manner: $\mathcal{O}(n^2 \log n)$ time and $\mathcal{O}(r^2)$ space. Obtain the next intersection in $\mathcal{O}(1)$ time.
- Store only the closest intersection to a point for all segments in $\mathcal{A}(P)$, i.e., overall $\mathcal{O}(r)$ intersections. Obtain the next intersection in $\mathcal{O}(r)$ time.

Space Time Trade-off

- For a fixed k in $1 \leq k \leq r$. Let s in $\mathcal{A}(P)$ and let p the next intersection point on s.
- Instead of p we compute and store the next k intersections in $\mathcal{O}(r \log r)$ time.
- On s are at most r intersections points, thus we have to compute the next k intersections at most r/k times.
- We require $\mathcal{O}(kr)$ space for $\mathcal{A}(P)$.

Induced Line Arrangement $\mathcal{A}(P)$

- For every reflex vertex v of P we construct a line segment s.
- Let s start at v, lie on $v(t)$ of $\mathcal{W}_P(t, \sigma)$, and end at the first intersection with the boundary of P.
- The set of all such line segments forms $\mathcal{A}(P)$.

Linear or Quadratic Space

- Store all $\mathcal{O}(r^2)$ intersections in a sorted manner: $\mathcal{O}(n^2 \log n)$ time and $\mathcal{O}(r^2)$ space. Obtain the next intersection in $\mathcal{O}(1)$ time.
- Store only the closest intersection to a point for all segments in $\mathcal{A}(P)$, i.e., overall $\mathcal{O}(r)$ intersections. Obtain the next intersection in $\mathcal{O}(r)$ time.

Space Time Trade-off

- For a fixed k in $1 \leq k \leq r$. Let s in $\mathcal{A}(P)$ and let p the next intersection point on s.
- Instead of p we compute and store the next k intersections in $\mathcal{O}(r \log r)$ time.
- On s are at most r intersections points, thus we have to compute the next k intersections at most r/k times.
- We require $\mathcal{O}(kr)$ space for $\mathcal{A}(P)$.

Induced Line Arrangement $\mathcal{A}(P)$

- For every reflex vertex v of P we construct a line segment s.
- Let s start at v, lie on $v(t)$ of $\mathcal{W}_P(t, \sigma)$, and end at the first intersection with the boundary of P.
- The set of all such line segments forms $\mathcal{A}(P)$.

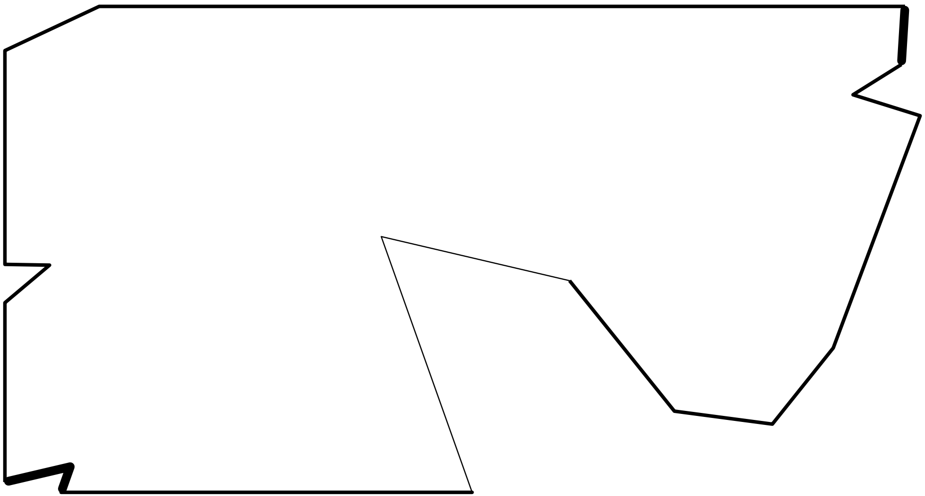
Linear or Quadratic Space

- Store all $\mathcal{O}(r^2)$ intersections in a sorted manner: $\mathcal{O}(n^2 \log n)$ time and $\mathcal{O}(r^2)$ space. Obtain the next intersection in $\mathcal{O}(1)$ time.
- Store only the closest intersection to a point for all segments in $\mathcal{A}(P)$, i.e., overall $\mathcal{O}(r)$ intersections. Obtain the next intersection in $\mathcal{O}(r)$ time.

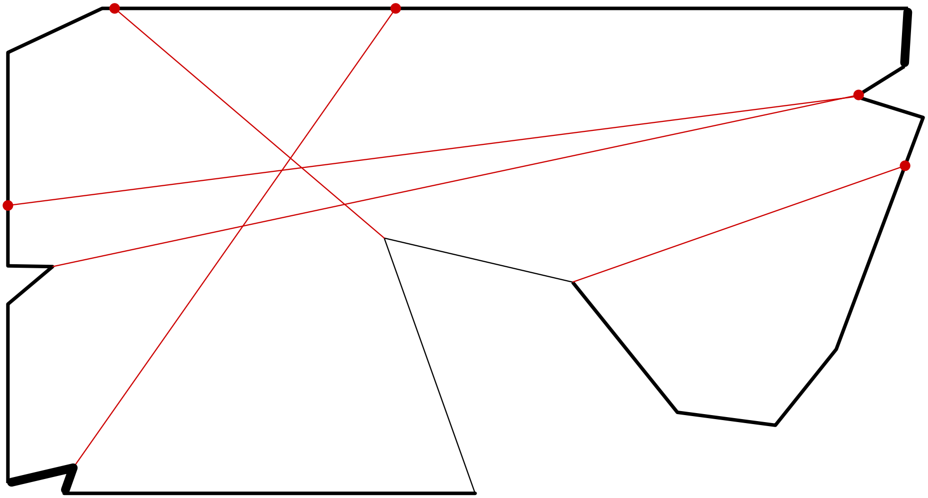
Space Time Trade-off

- For a fixed k in $1 \leq k \leq r$. Let s in $\mathcal{A}(P)$ and let p the next intersection point on s.
- Instead of p we compute and store the next k intersections in $\mathcal{O}(r \log r)$ time.
- On s are at most r intersections points, thus we have to compute the next k intersections at most r/k times.
- We require $\mathcal{O}(kr)$ space for $\mathcal{A}(P)$.

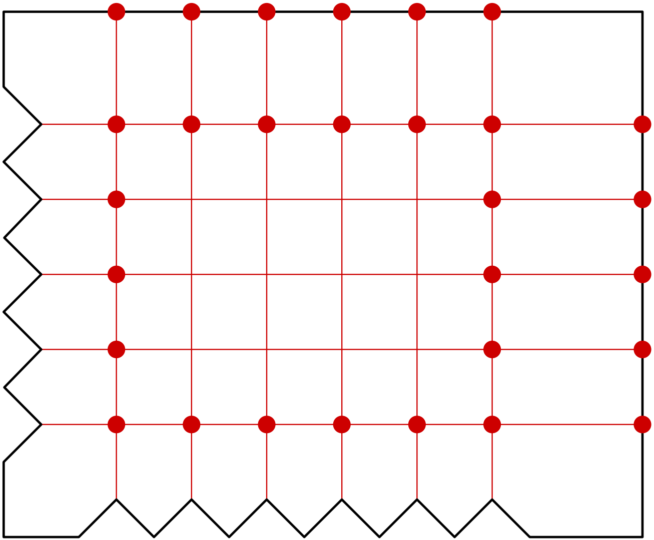
Extended Wavefront $\mathcal{W}_P^*(t, \sigma)$



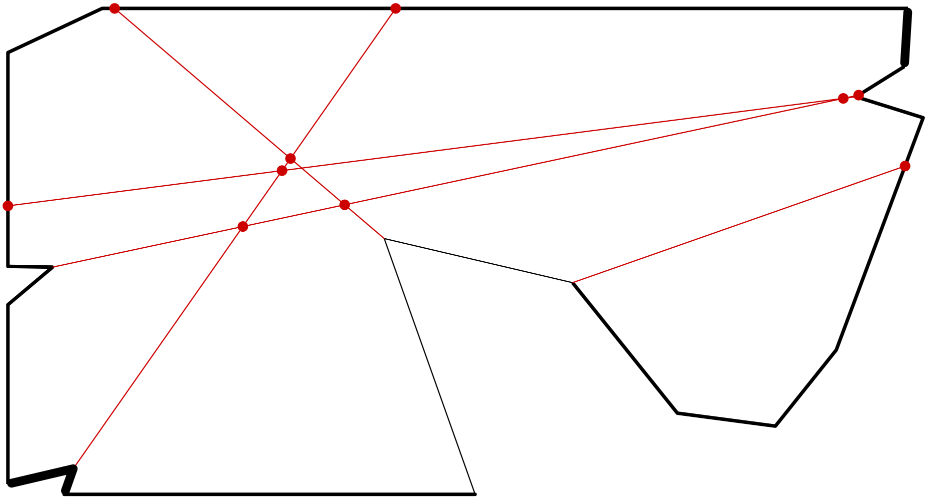
Extended Wavefront $\mathcal{W}_P^*(t, \sigma)$



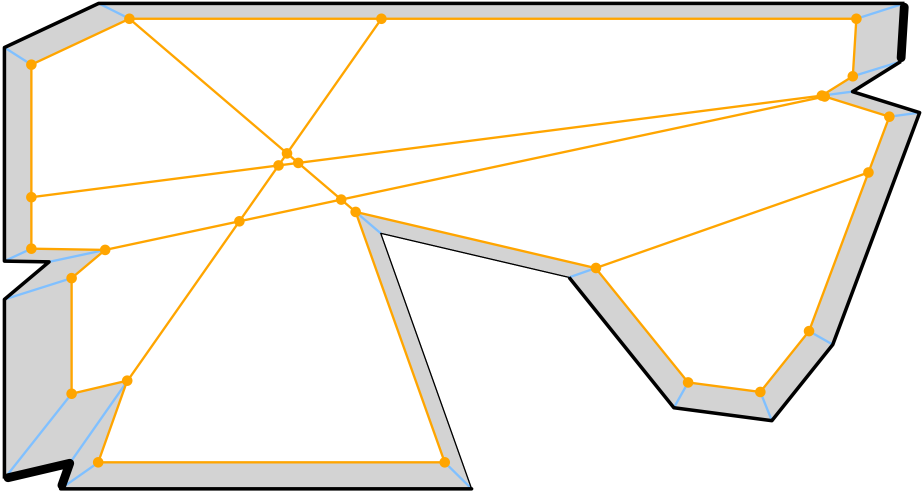
Extended Wavefront $\mathcal{W}_P^*(t,\sigma)$



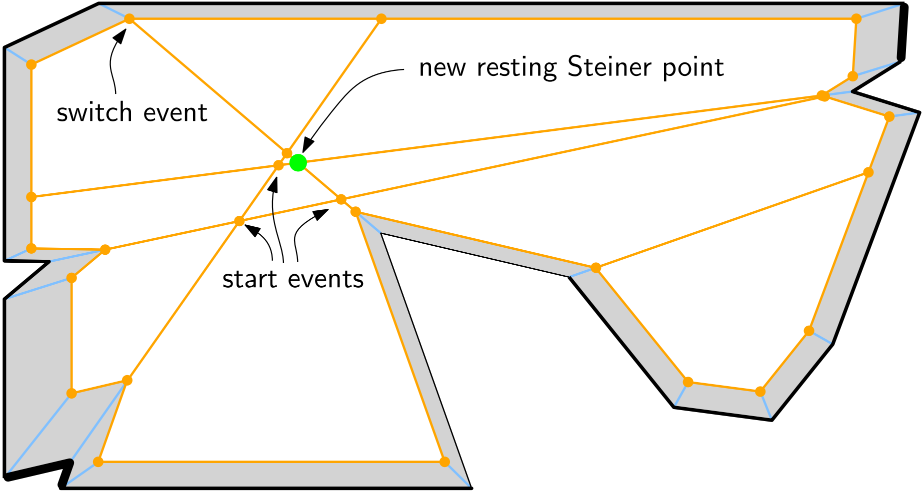
Extended Wavefront $\mathcal{W}_P^*(t, \sigma)$



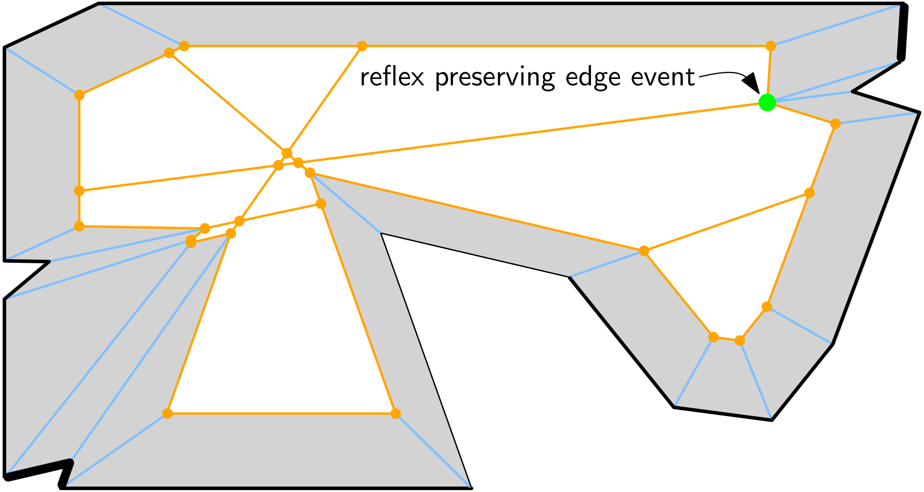
Extended Wavefront $\mathcal{W}_P^*(t, \sigma)$



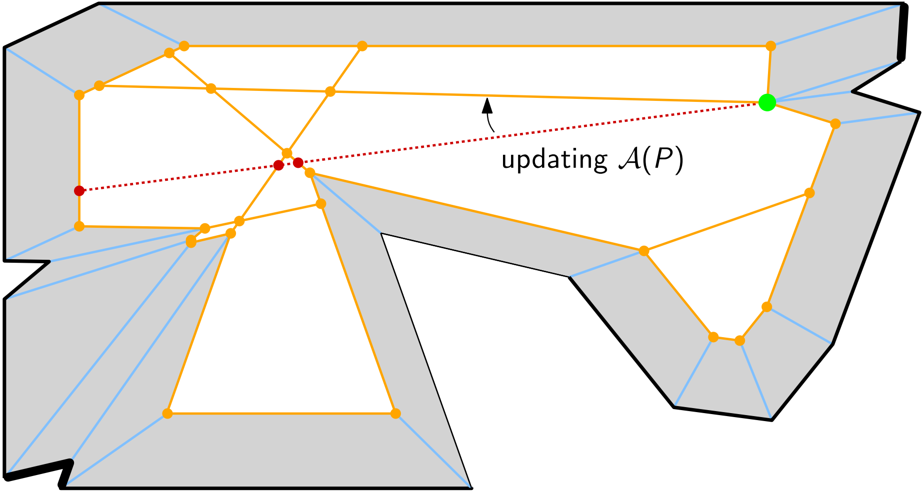
Extended Wavefront $\mathcal{W}^*_P(t, \sigma)$



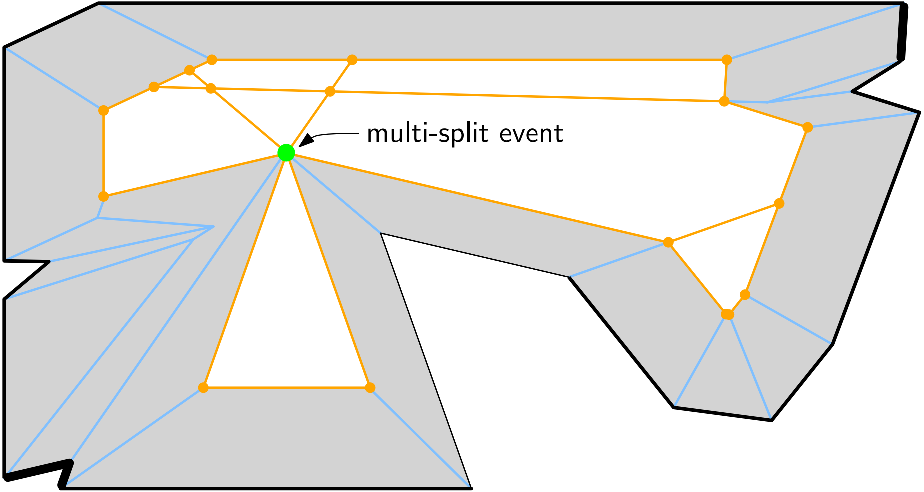
Extended Wavefront $\mathcal{W}^*_P(t,\sigma)$



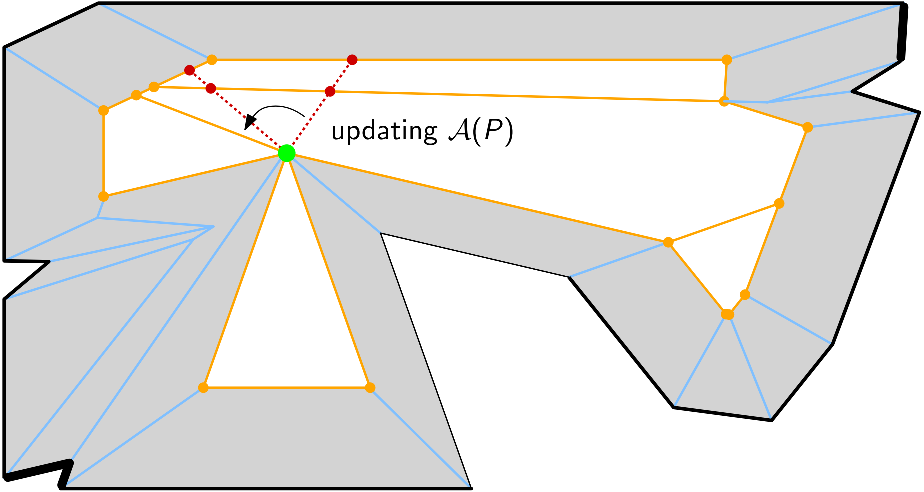
Extended Wavefront $\mathcal{W}^*_P(t, \sigma)$



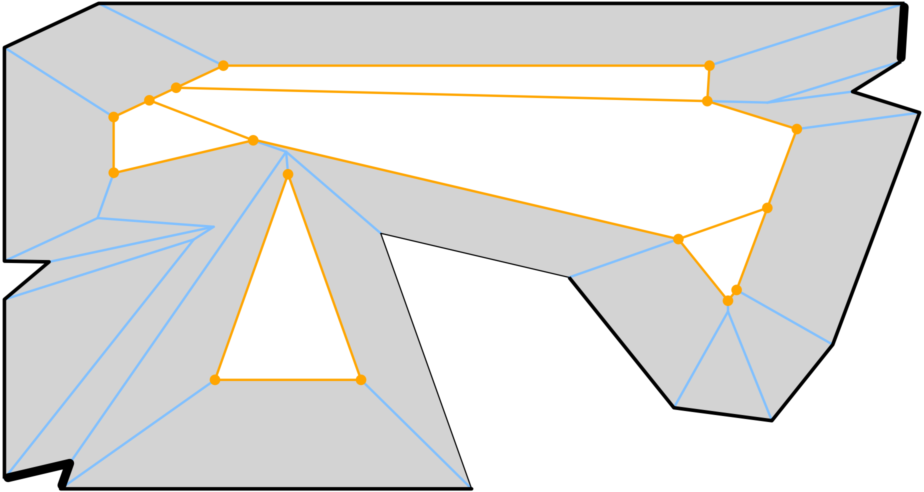
Extended Wavefront $\mathcal{W}_P^*(t, \sigma)$



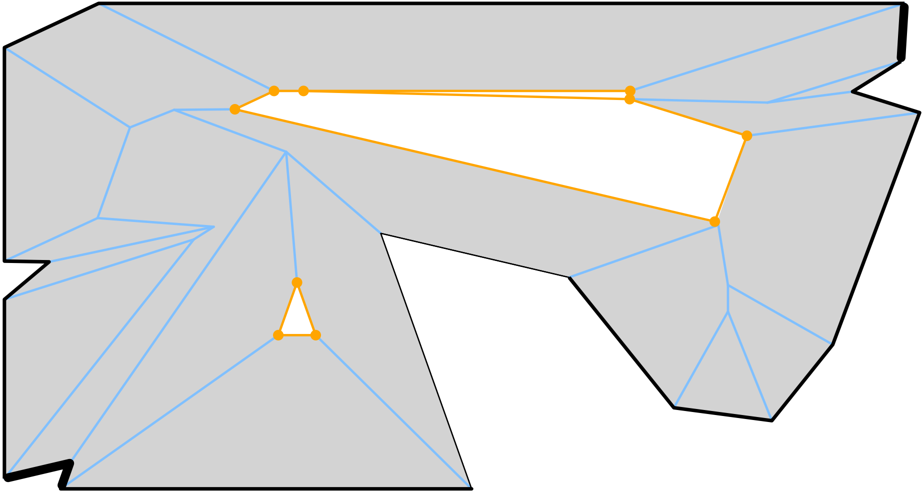
Extended Wavefront $\mathcal{W}_P^*(t, \sigma)$



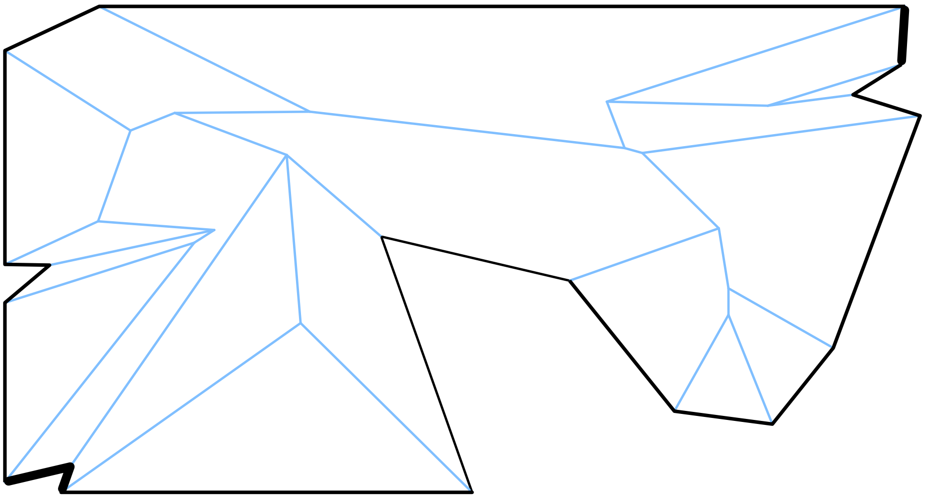
Extended Wavefront $\mathcal{W}_P^*(t, \sigma)$



Extended Wavefront $\mathcal{W}^*_P(t, \sigma)$



Extended Wavefront $\mathcal{W}_P^*(t, \sigma)$



Complexity Analysis

P consists of n vertices, r of which are reflex. The propagation of $\mathcal{W}_P^*(t, \sigma)$ results in:

- $\mathcal{O}(n)$ split/edge events,
- $\mathcal{O}(nr)$ switch events, and
- $\mathcal{O}(r^2)$ start events.

Complexity Analysis

P consists of n vertices, r of which are reflex. The propagation of $\mathcal{W}_P^*(t, \sigma)$ results in:

- $\mathcal{O}(n)$ split/edge events,
- $\mathcal{O}(nr)$ switch events, and
- $\mathcal{O}(r^2)$ start events.

Complexity Analysis

P consists of n vertices, r of which are reflex. The propagation of $\mathcal{W}^*_P(t, \sigma)$ results in:

- $\mathcal{O}(n)$ split/edge events,
- $\mathcal{O}(nr)$ switch events, and
- $\mathcal{O}(r^2)$ start events.

Complexity Analysis

P consists of n vertices, r of which are reflex. The propagation of $\mathcal{W}_P^*(t, \sigma)$ results in:

- $\mathcal{O}(n)$ split/edge events,
- $\mathcal{O}(nr)$ switch events, and
- $\mathcal{O}(r^2)$ start events.

Complexity Analysis

P consists of n vertices, r of which are reflex. The propagation of $\mathcal{W}_P^*(t, \sigma)$ results in:

- $\mathcal{O}(n)$ split/edge events,
- $\mathcal{O}(nr)$ switch events, and
- $\mathcal{O}(r^2)$ start events.

- Computing $\mathcal{W}_P^*(t, \sigma)$ at $t = 0$ takes $\mathcal{O}(n \log n + nr)$ time.
- Handling one (reflex preserving) edge event takes $\mathcal{O}(n + r + r \log n)$ time:
  - Updating $\mathcal{A}(P)$: $\mathcal{O}(r)$ time.
  - Adding/removing a segment of the wavefront: $\mathcal{O}(n)$ time.
  - The new segment in $\mathcal{A}(P)$ may invalidate $\mathcal{O}(r)$ events in $\mathcal{Q}$: $\mathcal{O}(r \log n)$.
- Handling one start event takes $\mathcal{O}(r + \log n)$ time:
  - $\mathcal{O}(r)$ time to find the next intersection in $\mathcal{A}(P)$.
  - $\mathcal{O}(\log n)$ time to add the event to $\mathcal{Q}$.

Complexity Analysis

P consists of n vertices, r of which are reflex. The propagation of $\mathcal{W}_P^*(t, \sigma)$ results in:

- $\mathcal{O}(n)$ split/edge events,
- $\mathcal{O}(nr)$ switch events, and
- $\mathcal{O}(r^2)$ start events.

- Computing $\mathcal{W}_P^*(t, \sigma)$ at $t = 0$ takes $\mathcal{O}(n \log n + nr)$ time.
- Handling one (reflex preserving) edge event takes $\mathcal{O}(n + r + r \log n)$ time:
  - Updating $\mathcal{A}(P)$: $\mathcal{O}(r)$ time.
  - Adding/removing a segment of the wavefront: $\mathcal{O}(n)$ time.
  - The new segment in $\mathcal{A}(P)$ may invalidate $\mathcal{O}(r)$ events in $\mathcal{Q}$: $\mathcal{O}(r \log n)$.
- Handling one start event takes $\mathcal{O}(r + \log n)$ time:
  - $\mathcal{O}(r)$ time to find the next intersection in $\mathcal{A}(P)$.
  - $\mathcal{O}(\log n)$ time to add the event to $\mathcal{Q}$.

Complexity Analysis

P consists of n vertices, r of which are reflex. The propagation of $\mathcal{W}^*_P(t,\sigma)$ results in:

- $\mathcal{O}(n)$ split/edge events,
- $\mathcal{O}(nr)$ switch events, and
- $\mathcal{O}(r^2)$ start events.

- Computing $\mathcal{W}^*_P(t,\sigma)$ at $t = 0$ takes $\mathcal{O}(n \log n + nr)$ time.
- Handling one (reflex preserving) edge event takes $\mathcal{O}(n + r + r \log n)$ time:
  - Updating $\mathcal{A}(P)$: $\mathcal{O}(r)$ time.
  - Adding/removing a segment of the wavefront: $\mathcal{O}(n)$ time.
  - The new segment in $\mathcal{A}(P)$ may invalidate $\mathcal{O}(r)$ events in $\mathcal{Q}$: $\mathcal{O}(r \log n)$.
- Handling one start event takes $\mathcal{O}(r + \log n)$ time:
  - $\mathcal{O}(r)$ time to find the next intersection in $\mathcal{A}(P)$.
  - $\mathcal{O}(\log n)$ time to add the event to $\mathcal{Q}$.

Complexity Analysis

P consists of n vertices, r of which are reflex. The propagation of $\mathcal{W}_P^*(t,\sigma)$ results in:

- $\mathcal{O}(n)$ split/edge events,
- $\mathcal{O}(nr)$ switch events, and
- $\mathcal{O}(r^2)$ start events.

- Computing $\mathcal{W}_P^*(t,\sigma)$ at $t=0$ takes $\mathcal{O}(n\log n + nr)$ time.
- Handling one (reflex preserving) edge event takes $\mathcal{O}(n + r + r\log n)$ time:
  - Updating $\mathcal{A}(P)$: $\mathcal{O}(r)$ time.
  - Adding/removing a segment of the wavefront: $\mathcal{O}(n)$ time.
  - The new segment in $\mathcal{A}(P)$ may invalidate $\mathcal{O}(r)$ events in $\mathcal{Q}$: $\mathcal{O}(r\log n)$.
- Handling one start event takes $\mathcal{O}(r + \log n)$ time:
  - $\mathcal{O}(r)$ time to find the next intersection in $\mathcal{A}(P)$.
  - $\mathcal{O}(\log n)$ time to add the event to $\mathcal{Q}$.

Complexity Analysis

P consists of n vertices, r of which are reflex. The propagation of $\mathcal{W}_P^*(t,\sigma)$ results in:

- $\mathcal{O}(n)$ split/edge events,
- $\mathcal{O}(nr)$ switch events, and
- $\mathcal{O}(r^2)$ start events.

- Computing $\mathcal{W}_P^*(t,\sigma)$ at $t = 0$ takes $\mathcal{O}(n \log n + nr)$ time.
- Handling one (reflex preserving) edge event takes $\mathcal{O}(n + r + r \log n)$ time:
  - Updating $\mathcal{A}(P)$: $\mathcal{O}(r)$ time.
  - Adding/removing a segment of the wavefront: $\mathcal{O}(n)$ time.
  - The new segment in $\mathcal{A}(P)$ may invalidate $\mathcal{O}(r)$ events in $\mathcal{Q}$: $\mathcal{O}(r \log n)$.
- Handling one start event takes $\mathcal{O}(r + \log n)$ time:
  - $\mathcal{O}(r)$ time to find the next intersection in $\mathcal{A}(P)$.
  - $\mathcal{O}(\log n)$ time to add the event to $\mathcal{Q}$.

Complexity Analysis

P consists of n vertices, r of which are reflex. The propagation of $\mathcal{W}_P^*(t, \sigma)$ results in:

- $\mathcal{O}(n)$ split/edge events,
- $\mathcal{O}(nr)$ switch events, and
- $\mathcal{O}(r^2)$ start events.

- Computing $\mathcal{W}_P^*(t, \sigma)$ at $t = 0$ takes $\mathcal{O}(n \log n + nr)$ time.
- Handling one (reflex preserving) edge event takes $\mathcal{O}(n + r + r \log n)$ time:
  - Updating $\mathcal{A}(P)$: $\mathcal{O}(r)$ time.
  - Adding/removing a segment of the wavefront: $\mathcal{O}(n)$ time.
  - The new segment in $\mathcal{A}(P)$ may invalidate $\mathcal{O}(r)$ events in $\mathcal{Q}$: $\mathcal{O}(r \log n)$.
- Handling one start event takes $\mathcal{O}(r + \log n)$ time:
  - $\mathcal{O}(r)$ time to find the next intersection in $\mathcal{A}(P)$.
  - $\mathcal{O}(\log n)$ time to add the event to $\mathcal{Q}$.

Complexity Analysis

P consists of n vertices, r of which are reflex. The propagation of $\mathcal{W}_P^*(t, \sigma)$ results in:

- $\mathcal{O}(n)$ split/edge events,
- $\mathcal{O}(nr)$ switch events, and
- $\mathcal{O}(r^2)$ start events.

- Computing $\mathcal{W}_P^*(t, \sigma)$ at $t = 0$ takes $\mathcal{O}(n \log n + nr)$ time.
- Handling one (reflex preserving) edge event takes $\mathcal{O}(n + r + r \log n)$ time:
  - Updating $\mathcal{A}(P)$: $\mathcal{O}(r)$ time.
  - Adding/removing a segment of the wavefront: $\mathcal{O}(n)$ time.
  - The new segment in $\mathcal{A}(P)$ may invalidate $\mathcal{O}(r)$ events in $\mathcal{Q}$: $\mathcal{O}(r \log n)$.
- Handling one start event takes $\mathcal{O}(r + \log n)$ time:
  - $\mathcal{O}(r)$ time to find the next intersection in $\mathcal{A}(P)$.
  - $\mathcal{O}(\log n)$ time to add the event to $\mathcal{Q}$.

Complexity Analysis

P consists of n vertices, r of which are reflex. The propagation of $\mathcal{W}^*_P(t, \sigma)$ results in:

- $\mathcal{O}(n)$ split/edge events,
- $\mathcal{O}(nr)$ switch events, and
- $\mathcal{O}(r^2)$ start events.

- Computing $\mathcal{W}^*_P(t, \sigma)$ at $t = 0$ takes $\mathcal{O}(n \log n + nr)$ time.
- Handling one (reflex preserving) edge event takes $\mathcal{O}(n + r + r \log n)$ time:
  - Updating $\mathcal{A}(P)$: $\mathcal{O}(r)$ time.
  - Adding/removing a segment of the wavefront: $\mathcal{O}(n)$ time.
  - The new segment in $\mathcal{A}(P)$ may invalidate $\mathcal{O}(r)$ events in $\mathcal{Q}$: $\mathcal{O}(r \log n)$.
- Handling one start event takes $\mathcal{O}(r + \log n)$ time:
  - $\mathcal{O}(r)$ time to find the next intersection in $\mathcal{A}(P)$.
  - $\mathcal{O}(\log n)$ time to add the event to $\mathcal{Q}$.

Complexity Analysis

P consists of n vertices, r of which are reflex. The propagation of $\mathcal{W}_P^*(t,\sigma)$ results in:

- $\mathcal{O}(n)$ split/edge events,
- $\mathcal{O}(nr)$ switch events, and
- $\mathcal{O}(r^2)$ start events.

- Computing $\mathcal{W}_P^*(t,\sigma)$ at $t=0$ takes $\mathcal{O}(n\log n + nr)$ time.
- Handling one (reflex preserving) edge event takes $\mathcal{O}(n + r + r\log n)$ time:
  - Updating $\mathcal{A}(P)$: $\mathcal{O}(r)$ time.
  - Adding/removing a segment of the wavefront: $\mathcal{O}(n)$ time.
  - The new segment in $\mathcal{A}(P)$ may invalidate $\mathcal{O}(r)$ events in $\mathcal{Q}$: $\mathcal{O}(r\log n)$.
- Handling one start event takes $\mathcal{O}(r + \log n)$ time:
  - $\mathcal{O}(r)$ time to find the next intersection in $\mathcal{A}(P)$.
  - $\mathcal{O}(\log n)$ time to add the event to $\mathcal{Q}$.

Overall Complexity

	time	*space*
classical	$\mathcal{O}(n^2 + r^3 + nr\log n)$	$\mathcal{O}(n)$

Complexity Analysis

P consists of n vertices, r of which are reflex. The propagation of $\mathcal{W}_P^*(t, \sigma)$ results in:

- $\mathcal{O}(n)$ split/edge events,
- $\mathcal{O}(nr)$ switch events, and
- $\mathcal{O}(r^2)$ start events.

- Computing $\mathcal{W}_P^*(t, \sigma)$ at $t = 0$ takes $\mathcal{O}(n \log n + nr)$ time.
- Handling one (reflex preserving) edge event takes $\mathcal{O}(n + r + r \log n)$ time:
  - Updating $\mathcal{A}(P)$: $\mathcal{O}(r)$ time.
  - Adding/removing a segment of the wavefront: $\mathcal{O}(n)$ time.
  - The new segment in $\mathcal{A}(P)$ may invalidate $\mathcal{O}(r)$ events in $\mathcal{Q}$: $\mathcal{O}(r \log n)$.
- Handling one start event takes $\mathcal{O}(r + \log n)$ time:
  - $\mathcal{O}(r)$ time to find the next intersection in $\mathcal{A}(P)$.
  - $\mathcal{O}(\log n)$ time to add the event to $\mathcal{Q}$.

Overall Complexity

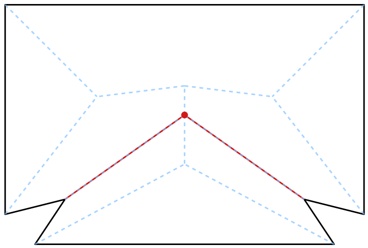
	time	*space*
classical	$\mathcal{O}(n^2 + r^3 + nr \log n)$	$\mathcal{O}(n)$
trade-off [5]	$\mathcal{O}(n^2 + r^3/k + nr \log n)$	$\mathcal{O}(n + kr)$

[5] with a fixed k s.t. $1 \leq k \leq r$.

Summary and Q & A

Summary

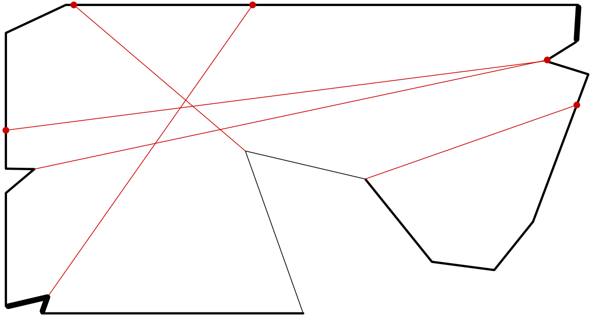
- Motorcycle Graph vs. $\mathcal{S}(P, \sigma)$.
- Induced Line Arrangement $\mathcal{A}(P)$.
- $\mathcal{W}_P^*(t, \sigma)$ as kinetic PSLG.
- Practical Candidate for an Implementation.
- Simple Space Time Trade-Off.
- $\mathcal{O}(n^2 + r^3/k + nr \log n)$ time and $\mathcal{O}(n + kr)$ space.



Summary and Q & A

Summary

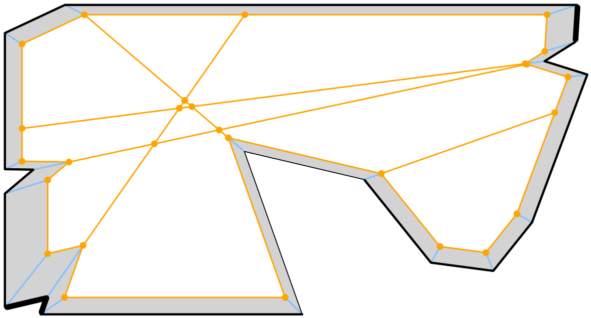
- Motorcycle Graph vs. $\mathcal{S}(P, \sigma)$.
- Induced Line Arrangement $\mathcal{A}(P)$.
- $\mathcal{W}_P^*(t, \sigma)$ as kinetic PSLG.
- Practical Candidate for an Implementation.
- Simple Space Time Trade-Off.
- $\mathcal{O}(n^2 + r^3/k + nr \log n)$ time and $\mathcal{O}(n + kr)$ space.



Summary and Q & A

Summary

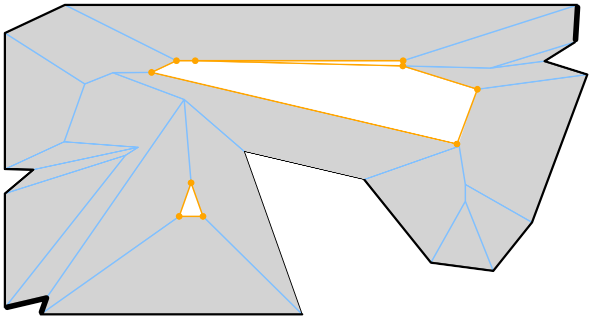
- Motorcycle Graph vs. $\mathcal{S}(P, \sigma)$.
- Induced Line Arrangement $\mathcal{A}(P)$.
- $\mathcal{W}_P^*(t, \sigma)$ as kinetic PSLG.
- Practical Candidate for an Implementation.
- Simple Space Time Trade-Off.
- $\mathcal{O}(n^2 + r^3/k + nr \log n)$ time and $\mathcal{O}(n + kr)$ space.



Summary and Q & A

Summary

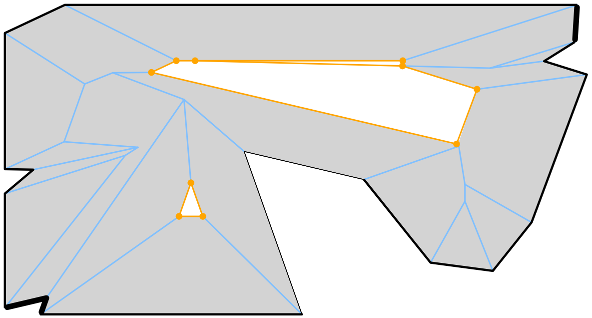
- Motorcycle Graph vs. $\mathcal{S}(P, \sigma)$.
- Induced Line Arrangement $\mathcal{A}(P)$.
- $\mathcal{W}_P^*(t, \sigma)$ as kinetic PSLG.
- Practical Candidate for an Implementation.
- Simple Space Time Trade-Off.
- $\mathcal{O}(n^2 + r^3/k + nr \log n)$ time and $\mathcal{O}(n + kr)$ space.



Summary and Q & A

Summary

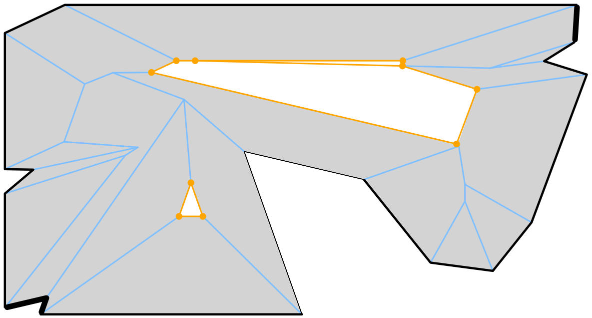
- Motorcycle Graph vs. $\mathcal{S}(P, \sigma)$.
- Induced Line Arrangement $\mathcal{A}(P)$.
- $\mathcal{W}_P^*(t, \sigma)$ as kinetic PSLG.
- Practical Candidate for an Implementation.
- Simple Space Time Trade-Off.
- $\mathcal{O}(n^2 + r^3/k + nr \log n)$ time and $\mathcal{O}(n + kr)$ space.



Summary and Q & A

Summary

- Motorcycle Graph vs. $\mathcal{S}(P, \sigma)$.
- Induced Line Arrangement $\mathcal{A}(P)$.
- $\mathcal{W}_P^*(t, \sigma)$ as kinetic PSLG.
- Practical Candidate for an Implementation.
- Simple Space Time Trade-Off.
- $\mathcal{O}(n^2 + r^3/k + nr \log n)$ time and $\mathcal{O}(n + kr)$ space.



Questions?

References I

[1] O. Aichholzer, F. Aurenhammer, D. Alberts, and B. Gärtner. A Novel Type of Skeleton for Polygons. *Journal of Universal Computer Science*, 1(12):752–761, 1995.

[2] S.-W. Cheng, L. Mencel, and A. Vigneron. A Faster Algorithm for Computing Straight Skeletons. 12(3):44:1–44:21, Apr. 2016.

[3] D. Eppstein and J. Erickson. Raising Roofs, Crashing Cycles, and Playing Pool: Applications of a Data Structure for Finding Pairwise Interactions. *Discrete & Computational Geometry*, 22(4):569–592, 1999.

[4] S. Huber and M. Held. A Fast Straight-Skeleton Algorithm Based on Generalized Motorcycle Graphs. *International Journal of Computational Geometry*, 22(5):471–498, 2012.

[5] P. Palfrader. Phd Defense.

[6] A. Vigneron and L. Yan. A Faster Algorithm for Computing Motorcycle Graphs. *Discrete & Computational Geometry*, 52(3):492–514, Oct. 2014.