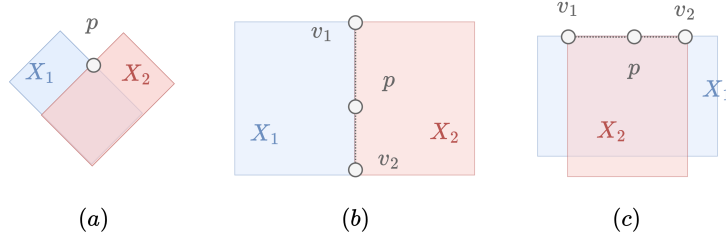


# WAP: Wavefront Arithmetic Propagation Algorithm and Implementation for Motion Planning Among Dynamic Obstacles (Supplementary Material)

Shijie Bao<sup>1</sup>[0009–0008–3258–1117], Shizhe Zhao<sup>1</sup>, and Zhongqiang Ren<sup>1</sup>

Shanghai Jiao Tong University  
thetoror@sjtu.edu.cn

## A Revisit the Repairing Function



**Fig. 1:** Scenarios where a point is on the boundary of two regular sets  $X_1, X_2$  when union them. The blue and red regions represent two regular sets,  $X_1$  and  $X_2$ , respectively, with point  $p$  located on the boundary of both sets. In scenario (a), point  $p$  is at the intersection of the two boundaries. In scenarios (b) and (c), the boundaries  $\partial X_1$  and  $\partial X_2$  overlap on the segment between points  $v_1$  and  $v_2$ , with point  $p$  situated on this segment.

Recall that when defining union operation between two regular sets  $X = X_1 \cup X_2$  (EQ. (5)), a repairing function is introduced to handle the edge case for point  $p$  on the boundary of both sets ( $p \in \partial X_1 \cap \partial X_2$ ). We now show that our algorithm never requires the use of the repairing function. Fig. 1 illustrates three scenarios. In Fig. 1(a), point  $p$  is at the intersection of the two boundaries, which will be a vertex in WG. Since we only evaluate the ISF value for each edge in WG in Alg. 2 and WG guarantees vertices never in the point set of edges, the repairing function is never be called. In Fig. 1(b) and (c), the pair  $v_1, v_2$  forms a duplicated edge in WG, and will be handled in Appendix C. For the same reason, the repairing function is also never be called for the intersection operation (Eq. (9)).

## B Proof of Lemma 2 in Sec. 4

Recall that  $\mathbb{P}_n = \{R_{n-1}\} \cup \{S_w(\tilde{l}) | \tilde{l} \in \partial R_{n-1}\}$  denote all regular sets involved in propagation, and let  $G^w : (E, V)$  denote the WG constructed from  $\mathbb{P}_n$  and  $\mathbb{O}_n$  (obstacle set at  $t_n$ ). For a point  $\vec{p}$  on an edge of  $G^w$ , let

$$\begin{cases} \lambda_1(\vec{p}) &= \sum_{P \in \mathbb{P}_n} \mathcal{F}(\partial P)(\vec{p}), \\ \lambda_2(\vec{p}) &= \sum_{O \in \mathbb{O}_n} \mathcal{F}(\partial O)(\vec{p}), \\ \gamma(a, b) &= 2 - f(2 - f(a) + f(b)), \end{cases} \quad (1)$$

$$\quad (2)$$

$$\quad (3)$$

**Lemma 2.**  $\mathcal{F}(\partial R_n)(\vec{p}) = \gamma(\lambda_1(\vec{p}), \lambda_2(\vec{p}))$  holds when  $\vec{p}$  is not on a duplicated edge of  $G^w$ .

*Proof.* Since  $p$  is not on a duplicated edge, the repairing function (Eq. (6)) is never be used for the union operator, this means that we can reformulate the saturation function  $f(\cdot)$  (Eq. (5)) to union multiple sets:

$$f(a + f(b + c)) = f(a + b + c), \text{ when } a, b, c > 0.$$

Then, applying the union operator (Eq. (5)) on  $\mathbb{P}_n$  and  $\mathbb{O}_n$ , we have:

$$\begin{cases} \mathcal{F}(\partial U_n)(\vec{p}) &= f(\sum_{P \in \mathbb{P}_n} \mathcal{F}(\partial P)(\vec{p})) = f(\lambda_1(\vec{p})), \\ \mathcal{F}(\partial O_n)(\vec{p}) &= f(\sum_{O \in \mathbb{O}_n} \mathcal{F}(\partial O)(\vec{p})) = f(\lambda_2(\vec{p})). \end{cases} \quad (4)$$

With complement operator (Eq. (8)) on  $\mathcal{F}(\partial O_n)$  and intersection operator (Eq. (9)) on  $\mathcal{F}(\partial(U_n \cap \neg O_n))(\vec{p})$  we have

$$\mathcal{F}(\partial(U_n \cap \neg O_n))(\vec{p}) = \gamma(\lambda_1(\vec{p}), \lambda_2(\vec{p})). \quad \square$$

## C Consider Duplicated Edges on Wavelet Graph (WG)

Recall that the WG,  $G^w(\mathbb{P}, \mathbb{O})$ , is constructed from regular sets  $\mathbb{P}$  and  $\mathbb{O}$ , where  $\mathbb{P}$  denote set of reachable regions and  $\mathbb{O}$  denote set of obstacles. For an edge  $e \in G^w$ ,  $A(e)$  is at set of wavelets that contain  $e$ , and  $e$  is a duplicated edge if  $|A(e)| > 1$ . Our problem is to compute  $\mathcal{F}(\partial X)(p)$  for point  $p \in e$  when  $e$  is a duplicated edge, where:

$$\begin{cases} X_P &= \bigcup_{P \in \mathbb{P}} P \\ X_O &= \bigcup_{O \in \mathbb{O}} O \\ X &= X_P \cap \neg X_O; \end{cases} \quad (5)$$

We start with a basic scenario that  $|\mathbb{P}| + |\mathbb{O}| = 2$ . Then, we demonstrate that when  $|\mathbb{P}| + |\mathbb{O}| > 2$ , we can treat  $e$  as a non-duplicated edge in certain cases and compute  $\mathcal{F}(\partial X)(e)$  according to Eq. (3). In the remaining cases, we can reduce the problem to the basic scenario.

**Basic Scenario** Assume  $|\mathbb{P}| + |\mathbb{O}| = 2$  and  $A(e) = \{\tilde{l}_1, \tilde{l}_2\}$ , where  $\tilde{l}_1$  and  $\tilde{l}_2$  are wavelets from regular sets  $X_1$  and  $X_2$  respectively. Fig. 2 illustrates all possible cases. In Fig. 2(a), all points  $p \in e$  are interior points ( $\mathcal{F}(\partial X)(p) = 2$ ), we referred to this case as *merge case*; in other cases, all points  $p \in e$  are boundary points ( $\mathcal{F}(\partial X)(p) = 1$ ).

**General Scenario** Let  $e$  denote a duplicated edge in  $G_n = G^w(\mathbb{P}_n, \mathbb{O}_n)$ , which constructed from  $\mathbb{P}_n$  and  $\mathbb{O}_n$ . Here,  $\mathbb{P}_n$  denote all regular sets involved in propagation at  $t_n$ , and  $\mathbb{O}_n$  denote all obstacles at  $t_n$ . We can start with  $\mathbb{P} = \mathbb{O} = A(e) = \emptyset$ , and iteratively add new regular set  $X^i$  from  $\mathbb{P}_n$  (or  $\mathbb{O}_n$ ) to  $\mathbb{P}$  (or  $\mathbb{O}$ ). The fact that  $e \in G_n$  ensures that adding a new regular set  $X^i$  will never introduce new vertices  $v \in e$ . Let  $A(e) = A_P(e) \cup A_O(e)$ , where  $A_P(e)$  and  $A_O(e)$  denote wavelets from  $\mathbb{P}$  and  $\mathbb{O}$  respectively. In each iteration, we update  $A(e)$  when  $e$  is on the boundary of  $X^i$  ( $e \in \partial X^i$ ). Additionally, if we can confirm that  $e \in \text{Int}(X_P)$  (or  $e \in \text{Int}(X_O)$ ), then  $e$  can be regarded as an interior curve of  $X_P$  (or  $X_O$ ), so that we can let  $A_P(e) = \emptyset$  (or  $A_O(e) = \emptyset$ ) and never update it in the remaining iterations. Determining  $e \in \text{Int}(X_P)$  (or  $e \in \text{Int}(X_O)$ ) can be done by checking (i)  $e \in \text{Int}(X^i)$ ; or (ii)  $X^i \in \mathbb{P}$  and falls into *merge case* (Fig 2(a)). After iterating all regular sets, there are following cases that we can treat  $e$  as a non-duplicated edge, so that we can compute  $\mathcal{F}(\partial X)(e)$  according to Eq. (3):

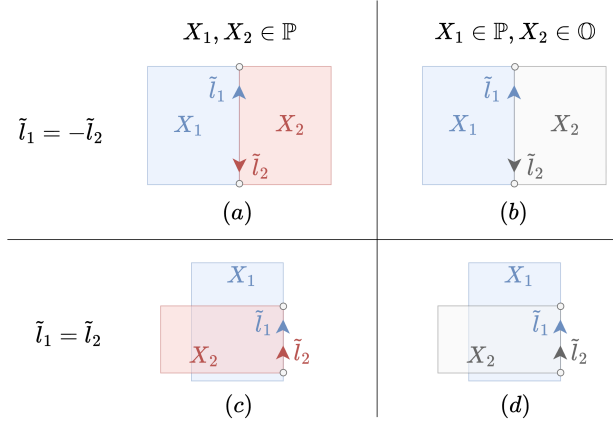
- $A(e) = \emptyset$ , meaning that  $e$  can be represented by interior curve of both  $X_P$  and  $X_O$ ;
- $A_P(e) = \emptyset \wedge A_O(e) \neq \emptyset$ , meaning that  $e$  is on the boundary of  $X_O$ , and since overlapping edges between obstacles are not allowed (Sec. 3), we have  $|A_O(e)| \leq 1$ ;
- $A_P(e) \neq \emptyset \wedge A_O(e) = \emptyset$ , meaning that  $e$  is only on the boundary of  $X_P$ , then we have  $\mathcal{F}(\partial X)(e) = \gamma(1, \lambda_2(e))$ ;

If  $A_P(e) \neq \emptyset \wedge A_O(e) \neq \emptyset$ , then, by letting  $X_1 = X_P, X_2 = X_O$ , we can confirm that  $\mathcal{F}(\partial X)(e) = 1$  according to the basic scenario in Fig. 2(b) or (d),

## D Numerical Robustness

Rounding errors in floating-point arithmetic can impact the result of our algorithm:

- (i) when building the WG (Alg. 1 line 6). For example, in Fig. 3(1), intersection points on  $\tilde{l}_b$  in CCW is  $a - b - c_1$ , while in Fig. 3(2), it is  $a - c_2 - b$ . Different orders lead to different wavelet segments in WG, representing different reachable regions.
- (ii) when computing ISF value of a point, which involves intersection tests and dot products between rays and wavelet segments (Alg. 2 line 12 to 20). For example, in Fig. 3(3),  $p$  is supposed to be on the boundary of  $\tilde{l}_1$ , but when computing with  $\tilde{l}_1$ ,  $p$  might be treated as located at  $p_1$  ( $p_1 \in R(\tilde{l}_1)$ )



**Fig. 2:** All cases of a duplicated edge between two regular sets  $X_1$  and  $X_2$  in the basic scenario ( $|\mathbb{P}| + |\mathbb{O}| = 2$ ). Blue and red regions represent regular sets from  $\mathbb{P}$ , grey region represent regular set from  $\mathbb{O}$ . Arrows indicate the direction of the associated wavelets to the edge.

or  $p'_1$  ( $p'_1 \notin R(\tilde{l}_1)$ ); while  $p$  might also be treated as located at  $p_2$  when computing with  $\tilde{l}_2$ .

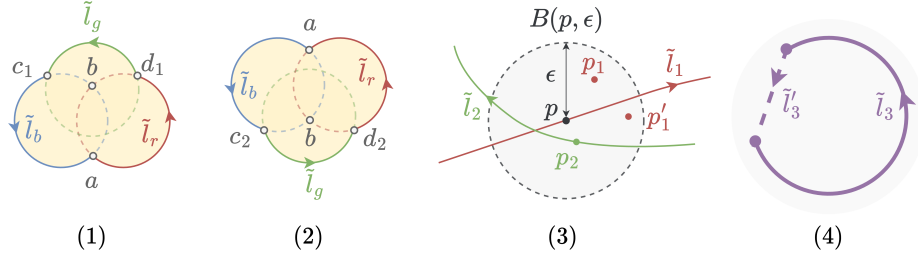
The consequence of rounding errors might be accumulated along propagation and may significantly alter the topology of the reachable region. For example Fig. 3(4) demonstrates that when rounding errors cause missing of wavelet segment  $\tilde{l}_3$ , the boundary of reachable region will become a non-closed curve. This causes all interior points of  $R(\tilde{l}_3)$  to be incorrectly treated as exterior points. Mitigating such numerical issues is non-trivial. Our implementation simply uses an  $\epsilon$ -distance test to determine point coincidence, and restart an iteration whenever detect unclosed boundaries that might due to rounding error. In each restarted iteration, we change the starting edge of *spread*. Experiment results in Appendix E show that this simple strategy is effective.

## E Experiments on Synthetic Datasets

**Dataset Description** We generate the synthetic datasets in following ways:

1. Polygonal obstacles are randomly generated independently at each timestep.
2. Static obstacles combined with moving obstacles, specifically:
  - (a) Obstacle speed is similar to the agent's.
  - (b) Obstacle speed is significantly lower than the agent's.
  - (c) Obstacle speed is significantly higher than the agent's.

The workspace is  $10 \times 10$ . The start and goal positions are randomly sampled from the workspace, and the average euclidean distance between them is 12. There are 1024 instances for each scenario. The size of an obstacle is defined as



**Fig. 3:** (1) and (2): Solid curves represent wavefronts and yellow shaded areas represent reachable regions. In both scenarios,  $\tilde{l}_b$  and  $\tilde{l}_r$  intersect at points  $b, a$ , and  $\tilde{l}_g$  intersect with  $\tilde{l}_b, \tilde{l}_r$  at  $c, d$  respectively; (3) A point  $p$  might be treated as any point in the circle  $B(p, \epsilon)$  due to numerical errors; (4) The dashed segment  $\tilde{l}'_3$  is a missing wavelet segment due to numerical errors.

Scenario	Time (ms)	SR(%)	#W
1	$78 \pm 14$	100%	97.5
2.a	$69 \pm 13$	99.9%	77.3
2.b	$72 \pm 14$	100%	85.7
2.c	$83 \pm 22$	100%	88.5

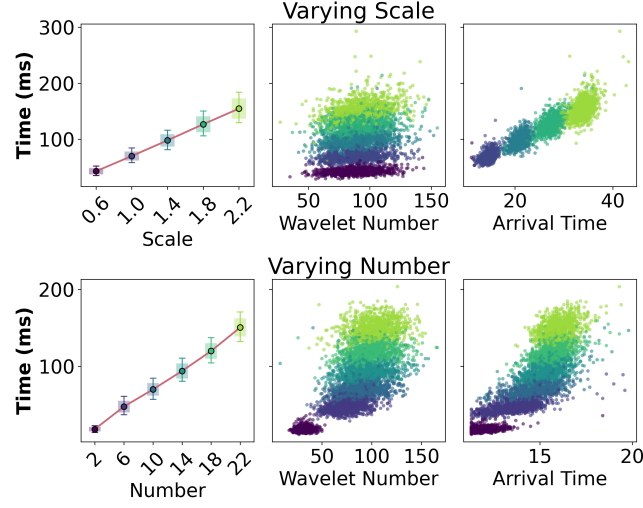
**Table 1:** Performance summary in different scenarios. The column Time shows the median value with standard deviation; SR shows the success rate; and #W shows the average number of wavelets per iteration.

the bounding box of the polygon. At each timestep, there are 10 obstacles of average size 2. Due to the numerical issues discussed in Appendix D, we inspect the result of all instances and ensures the path is valid. Invalid paths are treated as failures.

Table 1 summarizes the performance of WAP (with *spread*) in different scenarios. We can see that WAP can solve all instances in millisecond-level time, and rarely fails due to numerical errors.

Next, we take all 1,024 instances from the scenario 2.b, and vary the workspace size and obstacle density to analyze their impact on WAP. To inspect the impact of workspace size, we set a scale factor  $f \in \{0.6, 1.0, \dots, 2.2\}$ . For each  $f$ , we map a point  $(x, y)$  in the original workspace to  $(f \cdot x, f \cdot y)$  in the new workspace ( $f = 1.0$  means no change). To inspect the impact of obstacle density, we set the average number of obstacles per timestep  $n_o \in \{2, 6, \dots, 22\}$  ( $n_o = 2$  is the default setting). For both experiments, we record the runtime, maximum number of wavelets per iteration (referred to as ‘wavelet number’), and arrival time.

**Impact of Workspace Size.** Fig. 4(a) shows that both the runtime and arrival time increases linearly with the workspace size, while the wavelet number remains stable across different scales. This is because scaling the workspace size has minor



**Fig. 4:** Impact of workspace size (referred to as ‘Scale’) and number of wavelets (referred to as ‘Number’): The color gradient from dark to light indicates an increase in scale or the number.

impact on the topology of the reachable region, resulting in the wavelet number remains stable, while the number of iterations increases linearly with the scale.

**Impact of Obstacle Density.** Fig. 4(b) shows that the runtime increases linearly with the obstacle number. However, the wavelet number and arrival time initially increases with the obstacle number but then saturates when the obstacle number becomes sufficiently large. This is because runtime is impacted by two factors: the number of iterations (i.e., arrival time) and the computational cost per iteration, with only the latter increasing linearly with the obstacle number. Specifically, consider when obstacle number increases from 2 to 6. The wavelet number significantly increases as more obstacles appear in the reachable region, and more wavelets be generated per iteration. However, these obstacles which do not block the optimal path, can not impact the minimum arrival time. When the obstacle number increases further, the reachable region per iteration becomes smaller due to more obstacles block the way, and wavelets won’t be generated for obstacles that appear in non-reachable regions. Therefore, the wavelet number trends to saturate. This observation suggests that we can further improve the runtime by reducing the number obstacles to process in each iteration, for example, pruning these obstacle which cannot generate new wavelets.