Supplementary: A Time-Dependent Inclusion-Based Method for Continuous Collision Detection between Parametric Surfaces

XUWEN CHEN, School of Intelligence Science and Technology, Peking University, China
CHENG YU, School of Intelligence Science and Technology, Peking University, China
XINGYU NI, School of Computer Science, Peking University, China
MENGYU CHU, State Key Laboratory of General Artificial Intelligence, Peking University, China
BIN WANG*, State Key Laboratory of General Artificial Intelligence, BIGAI, China
BAOQUAN CHEN*, State Key Laboratory of General Artificial Intelligence, Peking University, China

1 INCLUSION INTERSECTION DETECTION

Our time-dependent inclusion-based CCD method is built upon the subdivision framework, where in each iteration we detect potential collisions between two subpatches by conservatively finding the time subinterval when the two time-dependent inclusions of the subpatches intersect during the given time step $[t^{\mathrm{l}}, t^{\mathrm{u}}]$. Specifically, we perform inclusion intersection detection by solving Eq. (18), as described in Alg. 3 in Section 4. It mainly relies on solving an inequality formed as

$$\min_{0\le\alpha<M_1} a_\alpha t + b_\alpha \le \max_{0\le\beta<M_2} c_\beta t + d_\beta, \tag{S1}$$

where on each side the max/min operator is applied on a set of linear functions with respect to t. As illustrated in Fig. 3, we solve this inequality by first compute the convex/concave boundary contour of the max/min function and then find the time subinterval where the min contour exceeds the max contour. We here elaborate our implementation of the algorithm for these two subroutines.

1.1 contour computation

The computation of the max/min contour is indeed to select a subset of the lines that makes up the max/min contour of the original set. We use the index sets $\mathcal{I} \subset \{0, 1, \ldots, M_1 - 1\}$ and $\mathcal{J} \subset \{0, 1, \ldots, M_2 - 1\}$ to indicate the selected lines that forms the contour of the max function and min function, respectively.

Without loss of generality, we focus on the computation of the max contour. As shown in Alg. S1, after sorting and removing the duplicates, we traverse the lines in the original set with the subscript α and store the selected lines in the stack $\mathcal{I}$. When dealing with the α-th line, we (1) first recursively check whether the previously selected line still forms part of the max contour if it were added and (2) then check whether it actually forms part of the max contour and should be added into the stack. Let $\Gamma = \mathrm{SizeOf}(\mathcal{I}) - 1$ so α_Γ denotes the index of the last selected line in $\mathcal{I}$ (the top element in the stack). We formalize the two tests as follows.

ALGORITHM S1: Max contour Computation

Input: A set of lines denoted by slopes and intercepts $\{a_\alpha t + b_\alpha \mid 0 \le \alpha < M_1\}$, the candidate interval $[t^{\mathrm{l}}, t^{\mathrm{u}}]$
Output: An index set $\mathcal{I}$.

```
Sort the lines in ascending order by slope;
Erase lines with same slope but smaller intercept;
Initialize the index set of the selected lines I ← {0};
foreach α do
    do
        if Φ < 0 then
            break
        else
            pop the last index in I
        end
    while I is not empty;
    if I is empty or φ < 0 then
        push α into I
    end
end
return I;
```

Test (1) equals to whether the α-th line intersects with the $\alpha_{\Gamma-1}$-th line after the α_Γ-th line does, as shown in Fig. S1. If so, the α_Γ-th line should be popped out of $\mathcal{I}$. When there is only one line left in the stack, the "$\alpha_{\Gamma-1}$-th line" is chosen as $t - t^{\mathrm{l}} = 0$. So the whole test is interpreted as $\Phi < 0$, where

$$\Phi = \begin{cases} (a_{\alpha_\Gamma} - a_{\alpha_{\Gamma-1}})(b_\alpha - b_{\alpha_{\Gamma-1}}) - (a_\alpha - a_{\alpha_{\Gamma-1}})(b_{\alpha_\Gamma} - b_{\alpha_{\Gamma-1}}), & \Gamma > 0, \quad \text{(S2a)} \\ (a_\alpha - a_{\alpha_\Gamma})t^{\mathrm{l}} + (b_\alpha - b_{\alpha_\Gamma}), & \Gamma = 0. \quad \text{(S2b)} \end{cases}$$

The two cases are depicted in Fig. S1. If the test passes, the Γ-th line is popped and the test continues for the next top line in the stack until the stack is cleared or the test fails.

Test (2) equals to whether the α-th line intersects the α_Γ-th line before t^{u}, as shown in Alg. S2. This test is interpreted as $\phi < 0$, where

$$\phi = (a_{\alpha_\Gamma} - a_\alpha)t^{\mathrm{u}} + (b_{\alpha_\Gamma} - b_\alpha). \tag{S3}$$

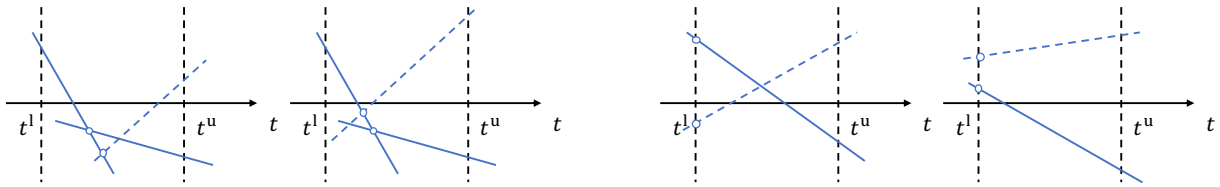
If the test passes, the α-th line is selected as constructing part of the max contour.

We point out that we exclude equal sign in both inequality tests in order to leave out redundant lines.

The computation of the min contour shares a similar process, with a totally reversed sorting and a little revision in computing

*corresponding authors

Authors' addresses: Xuwen Chen, pku_xwchen@163.com, School of Intelligence Science and Technology, Peking University, Beijing, China; Cheng Yu, chengyupku@pku.edu.cn, School of Intelligence Science and Technology, Peking University, Beijing, China; Xingyu Ni, nixy@pku.edu.cn, School of Computer Science, Peking University, Beijing, China; Mengyu Chu, mchu@pku.edu.cn, State Key Laboratory of General Artificial Intelligence, Peking University, China; Bin Wang, binwangbuaa@gmail.com, State Key Laboratory of General Artificial Intelligence, BIGAI, Beijing, China; Baoquan Chen, baoquan@pku.edu.cn, State Key Laboratory of General Artificial Intelligence, Peking University, Beijing, China



case1: $SizeOf(\mathcal{I}) > 1$ case2: $SizeOf(\mathcal{I}) = 1$

Fig. S1. Two different cases of testing whether the top line in the stack should be popped. The solid lines denote the lines in the stack and the dotted line denotes the line being checked. In each case, the test fails in the left figure and passes in the right figure.

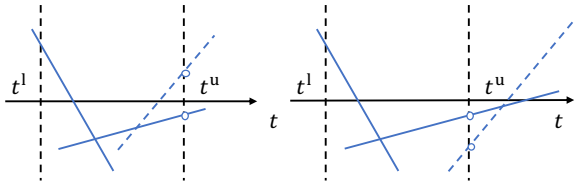


Fig. S2. Testing whether the top line in the stack should be popped. The solid lines denote the lines in the stack and the dotted line denotes the line being checked. The test passes in the left figure and fails in the right figure.

ALGORITHM S2: Left Endpoint of contour Intersection

Input: A max contour denoted by $\{a_{\alpha_\gamma} t + b_{\alpha_\gamma} \mid \alpha_\gamma \in \mathcal{I}\}$, a min contour denoted by $\{c_{\beta_\lambda} t + d_{\beta_\lambda} \mid \beta_\lambda \in \mathcal{J}\}$, the candidate interval $[t^{\mathrm{l}}, t^{\mathrm{u}}]$

Output: The left endpoint of the intersection interval of the two contours

if $\psi < 0$ **then**
  return t^{l};
end
do
  if $a_{\alpha_\gamma} \geq c_{\beta_\lambda}$ **then break**;
  if $\Psi_{\max} < 0$ **then**
    if $\Psi_{\min} < 0$ **then**
      return $-(b_{\alpha_\gamma} - d_{\beta_\lambda})/(a_{\alpha_\gamma} - c_{\beta_\lambda})$;
    else
      $\lambda \leftarrow \lambda + 1$;
    end
  else
    $\gamma \leftarrow \gamma + 1$;
  end
while $\gamma <$`SizeOf` $(\mathcal{I})$ **and** $\lambda <$`SizeOf` $(\mathcal{J})$;
return null;

Φ and ϕ as

$$\Phi = \begin{cases} (a_{\alpha_\Gamma} - a_{\alpha_{\Gamma-1}})(b_\alpha - b_{\alpha_{\Gamma-1}}) - (a_\alpha - a_{\alpha_{\Gamma-1}})(b_{\alpha_\Gamma} - b_{\alpha_{\Gamma-1}}), & \Gamma > 0, \text{(S4a)} \\ -[(a_{\alpha_\Gamma} - a_\alpha)t^{\mathrm{l}} + (b_{\alpha_\Gamma} - b_\alpha)], & \Gamma = 0, \text{(S4b)} \end{cases}$$

$$\phi = -[(a_{\alpha_\Gamma} - a_\alpha)t^{\mathrm{u}} + (b_{\alpha_\Gamma} - b_\alpha)]. \tag{S5}$$

1.2 contour intersection computation

After obtaining the max-contour subset $\{(a_{\alpha_\gamma}, b_{\alpha_\gamma})\}_{\alpha_\gamma \in \mathcal{I}}$ and the min-contour subset $\{(c_{\beta_\lambda}, d_{\beta_\lambda})\}_{\beta_\lambda \in \mathcal{J}}$ sorted by slope in ascending and descending order, respectively, we simultaneously traverse the max-contour and min-contour subsets by traversing the index sets $\mathcal{I}$ and $\mathcal{J}$ using subscripts γ and λ individually. Let $\Gamma = \mathrm{SizeOf}(\mathcal{I}) - 1$ and $\Lambda = \mathrm{SizeOf}(\mathcal{J}) - 1$. Since the max contour is convex and the min contour is concave, the intersection would be null or a single connected interval of t. So next we explain how to find the left end of this interval or achieve the conclusion that intersection does not exist. The process for finding the right end follows the same approach with the traversing order reversed.

As shown in Alg. S2, to find the left end is to find where the two segment contours first intersect. If the max contour starts at a value below the value of the min contour at the start of the candidate time interval t^{l}, satisfying

$$\psi = (a_{\alpha_\gamma} - c_{\beta_\lambda})t^{\mathrm{l}} + (b_{\alpha_\gamma} - d_{\beta_\lambda}) < 0, \tag{S6}$$

we can directly set the left end as t^{l}, otherwise we continue to sweep the contours. If the α_γ-th max-contour line and the β_λ-th min-contour line intersect, the intersection point must (1) lie before the α_γ-th line segment ends and (2) lie before the β_λ-th line segment ends.

Condition (1) is checked by comparing the intersection of the β_λ-th min-contour line with the α_γ-th max-contour line and with the $\alpha_{\gamma+1}$-th max-contour line respectively. "The $\alpha_{\gamma+1}$-th max-contour line" is set as $t - t^{\mathrm{u}} = 0$ when there is no succeeding line after the α_γ-th line. Though there exists three different cases depicted in Fig. S3, condition (1) is satisfied if and only if $\Psi_{\max} < 0$, where

$$\Psi_{\max} = \begin{cases} (a_{\alpha_{\gamma+1}} - c_{\beta_\lambda})(b_{\alpha_\gamma} - d_{\beta_\lambda}) - (a_{\alpha_\gamma} - c_{\beta_\lambda})(b_{\alpha_{\gamma+1}} - d_{\beta_\lambda}), & \gamma < \Gamma, \text{(S7a)} \\ (a_{\alpha_\gamma} - c_{\beta_\lambda})t^{\mathrm{u}} + (b_{\alpha_\gamma} - d_{\beta_\lambda}), & \gamma \geq \Gamma. \text{(S7b)} \end{cases}$$

Condition (2) is satisfied if and only if $\Psi_{\min} < 0$ where

$$\Psi_{\min} = \begin{cases} (a_{\alpha_\gamma} - c_{\beta_{\lambda+1}})(b_{\alpha_\gamma} - d_{\beta_\lambda}) - (a_{\alpha_\gamma} - c_{\beta_\lambda})(b_{\alpha_\gamma} - d_{\beta_{\lambda+1}}), & \lambda < \Lambda, \text{(S8a)} \\ (a_{\alpha_\gamma} - c_{\beta_\lambda})t^{\mathrm{u}} + (b_{\alpha_\gamma} - d_{\beta_\lambda}), & \lambda \geq \Lambda. \text{(S8b)} \end{cases}$$

Once the two conditions are both satisfied, we calculate the left end of the intersection by

$$t = -\frac{b_{\alpha_\gamma} - d_{\beta_\lambda}}{a_{\alpha_\gamma} - c_{\beta_\lambda}}. \tag{S9}$$

If test for condition (1)/(2) fails, i.e., the horizontal coordinate of the intersection point gets beyond the end point of max-contour/min-contour segment, we add γ/λ by one, moving on to the next line

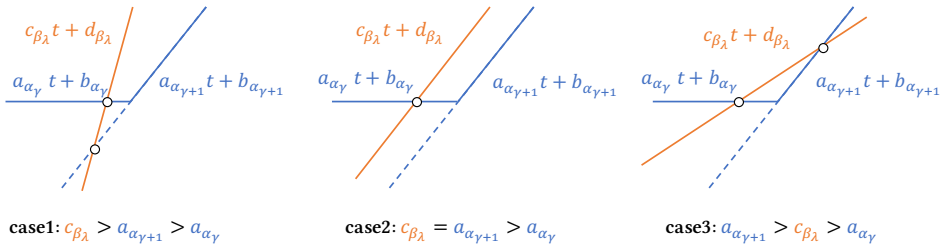


Fig. S3. Three different cases when the γ-th max-contour line and the λ-th min-contour line intersect before the γ-th segment ends.

in the max/min contour. We recursively conduct the tests until c_{β_λ} becomes no larger than a_{α_γ} or we have checked all the lines in either of the two sets before we find an intersection, indicating no intersection happens between the two contours.

If the left end exists, we then compute the right end of the intersection interval using a similar algorithm. The main difference is that the sweep is done from end to start. Each time when condition (1)/(2) fails, we subtract γ/λ by one, moving on to the previous line in the max/min contour. The formalized conditions are written as

$$\psi = (a_{\alpha_\gamma} - c_{\beta_\lambda})t^{\mathrm{u}} + (b_{\alpha_\gamma} - d_{\beta_\lambda}), \tag{S10}$$

$$\Psi_{\max} = \begin{cases} (a_{\alpha_\gamma} - c_{\beta_\lambda})(b_{\alpha_{\gamma-1}} - d_{\beta_\lambda}) - (a_{\alpha_{\gamma-1}} - c_{\beta_\lambda})(b_{\alpha_\gamma} - d_{\beta_\lambda}), & \gamma - 1 \geq 0, \text{(S11a)} \\ (a_{\alpha_\gamma} - c_{\beta_\lambda})t^{\mathrm{u}} + (b_{\alpha_\gamma} - d_{\beta_\lambda}), & \gamma - 1 < 0, \text{(S11b)} \end{cases}$$

$$\Psi_{\min} = \begin{cases} (a_{\alpha_\gamma} - c_{\beta_\lambda})(b_{\alpha_\gamma} - d_{\beta_{\lambda-1}}) - (a_{\alpha_\gamma} - c_{\beta_{\lambda-1}})(b_{\alpha_\gamma} - d_{\beta_\lambda}), & \lambda - 1 \geq 0, \text{(S12a)} \\ (a_{\alpha_\gamma} - c_{\beta_\lambda})t^{\mathrm{u}} + (b_{\alpha_\gamma} - d_{\beta_\lambda}), & \lambda - 1 < 0. \text{(S12b)} \end{cases}$$

1.3 Analysis

The whole algorithm of solving Eq. (S1) for the feasible time interval is at $O(M_1 \log M_1 + M_2 \log M_2)$. Specifically, we compute the max contour by first sorting the line set and then sweeping the set once, which takes $O(M_1 \log M_1)$ and $O(M_1)$ time, respectively. Similarly, the sorting and sweeping during computing the min contour takes $O(M_2 \log M_2)$ and $O(M_2)$ time, respectively. Then we compute the contour intersection by sweeping the two contour subsets simultaneously, which takes $O(M_1 + M_2)$ time. Therefore the bottleneck of time complexity is the sorting process during constructing the max contour and the min contour.

We have tried different ways of constructing the criteria Φs and Ψs, for example, using different intersection points to determine the relationship between the lines, and using the division form of the intersection points without rearranging them into multiplication. All the implementation ways have the same time complexity. We find that such little modifications hardly harm the performance of the overall method. We believe that any reasonable implementation of the sorting-and-sweeping algorithm can work as well as ours.