One-Pass Error Bounded Trajectory Simplification

Basic Douglas-Peucker algorithm:

Algorithm DP($\ddot{\mathcal{T}}[P_0, \ldots, P_n], \zeta$)

1. **for** each point P_i ($i \in [0, n]$) in $\ddot{\mathcal{T}}[P_0, \ldots, P_n]$ **do**
2. compute $d(P_i, \mathcal{L})$ between P_i and $\mathcal{L}(P_0, P_n)$;
3. **let** $d(P_k, \mathcal{L}) := \max\{d(P_0, \mathcal{L}), \ldots, d(P_n, \mathcal{L})\}$;
4. **if** $d(P_k, \mathcal{L}) \leq \zeta$ **then**
5. **return** $\{\mathcal{L}(P_0, P_n)\}$.
6. **else**
7. **return** DP($\ddot{\mathcal{T}}[P_0, \ldots, P_k], \zeta$) $\cup$ DP($\ddot{\mathcal{T}}[P_k, \ldots, P_n], \zeta$).

BQS:

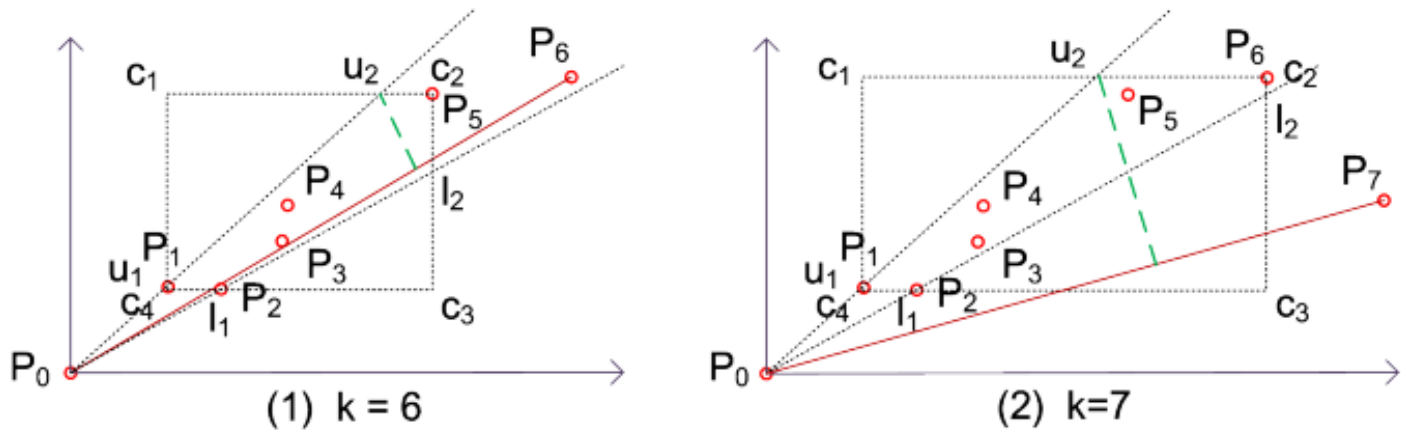


Figure 4: Examples for algorithm BQS.

Fitting function F:

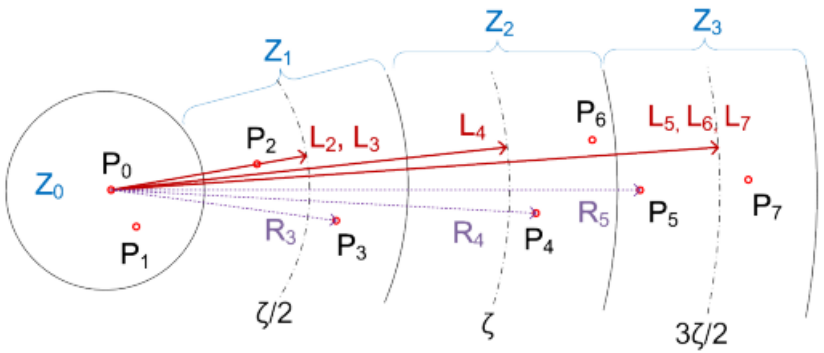


Figure 5: An example of the fitting function

$$\begin{cases} [\mathcal{L}_i = \mathcal{L}_{i-1}] & when\ (|\mathcal{R}_i| - |\mathcal{L}_{i-1}|) \le \frac{\zeta}{4} \quad (1) \\ \begin{bmatrix} |\mathcal{L}_i| = j * \zeta/2 \\ \mathcal{L}_i.\theta = \mathcal{R}_i.\theta \end{bmatrix} & when\ |\mathcal{R}_i| > \frac{\zeta}{4}\ \textbf{and}\ |\mathcal{L}_{i-1}| = 0 \quad (2) \\ \begin{bmatrix} |\mathcal{L}_i| = j * \zeta/2 \\ \mathcal{L}_i.\theta = \mathcal{L}_{i-1}.\theta + f(\mathcal{R}_i, \mathcal{L}_{i-1}) * \arcsin(\frac{d(P_{s+i}, \mathcal{L}_{i-1})}{j * \zeta/2})/j \end{bmatrix} & else\ (3) \end{cases}$$

where (a) $1 \le i \le k+1$; (b) $\mathcal{R}_{i-1} = \overrightarrow{P_s P_{s+i-1}}$, is the directed line segment whose end point P_{s+i-1} is in $\dddot{\mathcal{T}}_s[P_s, \ldots, P_{s+k}]$;

OPERB:

Algorithm OPERB($\dddot{\mathcal{T}}[P_0, \ldots, P_n], \zeta$)

1. $\overline{\mathcal{T}} := \emptyset$; $P_e := P_0$; $(P_a, flag) :=$ getActivePoint($\dddot{\mathcal{T}}, P_0, P_0, \mathcal{L}_0, \zeta$);
2. **while** $P_a \neq$ nil **do** {
3. $\quad P_s := P_e$; $\quad \mathcal{L}_a = \mathbb{F}(P_a, \overrightarrow{P_s P_s})$;
4. $\quad (P_a, flag) :=$ getActivePoint($\ddot{\mathcal{T}}, P_s, P_a, \mathcal{L}_a, \zeta$);
5. $\quad$ **while** $P_a \neq$ nil & $flag = true$ **do** {
6. $\quad\quad \mathcal{L}_a := \mathbb{F}(P_a, \mathcal{L}_a)$; $P_e := P_a$;
7. $\quad\quad (P_a, flag) :=$ getActivePoint($\dddot{\mathcal{T}}, P_s, P_a, \mathcal{L}_a, \zeta$); }
8. $\quad \overline{\mathcal{T}} := \overline{\mathcal{T}} \cup \{\overrightarrow{P_s P_e}\}$; }
9. **return** $\overline{\mathcal{T}}$.

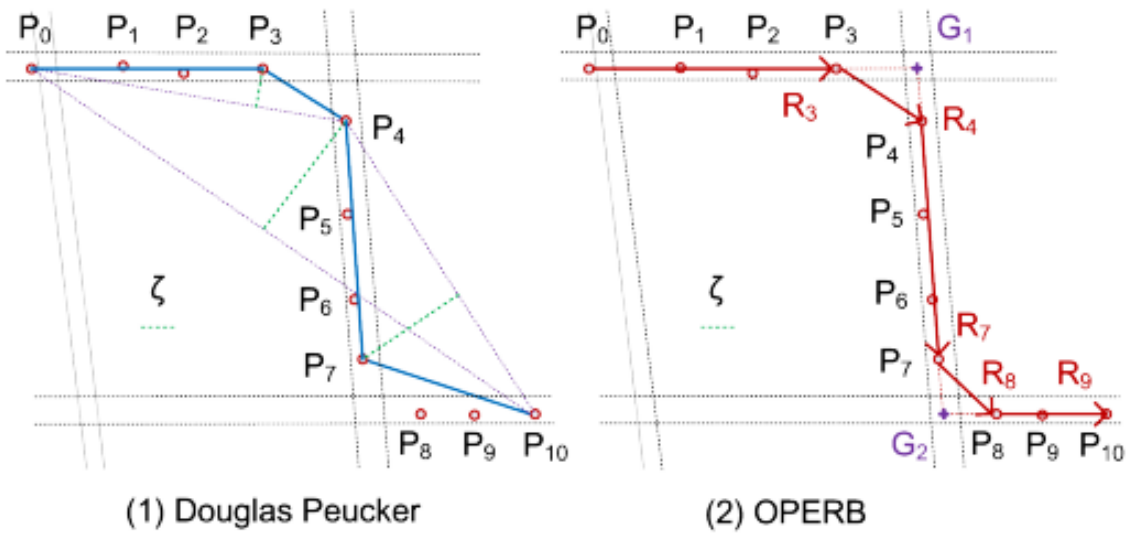
Procedure getActivePoint($\dddot{\mathcal{T}}, P_s, P_a, \mathcal{L}_a, \zeta$)

1. $i := a + 1$; $\quad flag := true$;
2. **while** $((|\mathcal{R}_i| - |\mathcal{L}_a|) \leq \zeta/4$ & $i \leq n$ & $(i - s) \leq 4 \times 10^5$ **do** {
3. $\quad$ **if** $d(P_i, \mathcal{L}_a) > \zeta/2$ **or** $d(P_i, \mathcal{R}_a) > \zeta$ **then**
4. $\quad\quad flag := false$; $\quad$ **break** ;
5. $\quad i := i + 1$; }
6. **if** $d(P_i, \mathcal{L}_a) > \zeta/2$ & $|\mathcal{L}_a| > 0$ **then** $flag := false$;
7. **if** $i = n + 1$ **then** $P_i :=$nil;
8. **return** $(P_i, flag)$.

Optimization techniques:

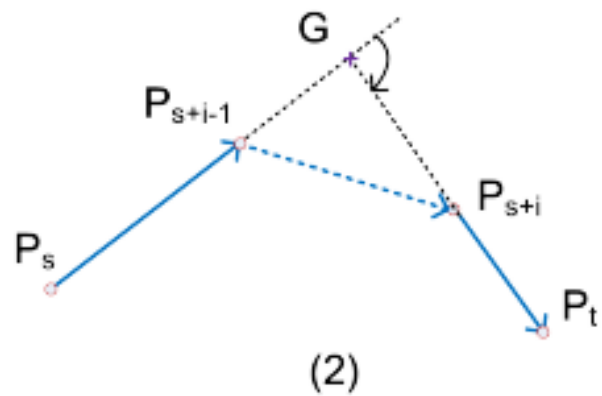
1.Choosing the first active point after P_s.
2.Making L more close to the active points.
3.Absorbing data points after P_{s+k}.

OPERB-A:



(1) Douglas Peucker

(2) OPERB



(2)