Outline

- ✓ A. Intro
- ✓ B. Basic proximity problems
- ✓ C. Basic decision and summation problems
- ⇨ D. Basic 'all'-type problems
- E. Basic counting problems
- F. Further topics

'All'-type problems

- **Nearest-neighbor search** $NN(x_q) = \arg\min_r \|x_q - x_r\|$

  All-nearest neighbor (bichromatic): $\forall x_q : NN(x_q) = \arg\min_r \|x_q - x_r\|$

- **Kernel density estimation** $\hat{f}(x_q) = \frac{1}{N}\sum_{r \neq q}^{N} K_h(\|x_q - x_r\|)$

  'All' version (bichromatic): $\forall x_q : \hat{f}(x_q) = \frac{1}{N}\sum_{r \neq q}^{N} K_h(\|x_q - x_r\|)$

Almost always 'all'-type problems

- Kernel density estimation
- Nadaraya-Watson & locally-wgtd regression
- Gaussian process prediction
- Radial basis function networks

Always 'all'-type problems

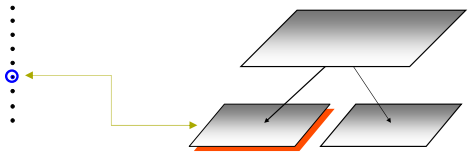
- Bichromatic all-nearest neighbor (e.g. LLE)
- Correlation dimension (pairs)
- *n*-point correlation (*n*-tuples)

Dual-tree idea

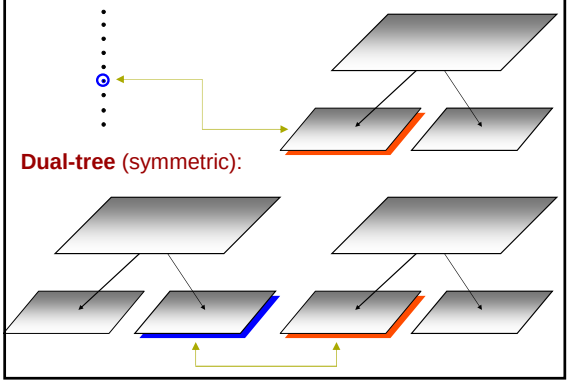
If all the queries are available simultaneously, then it is faster to:

1. Build a tree on the queries as well
2. Effectively process the queries in chunks rather than individually → *work is shared between similar query points*

Single-tree:

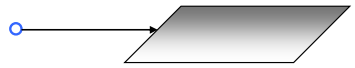


Single-tree:



Exclusion and inclusion,
using point-node *kd*-tree bounds.

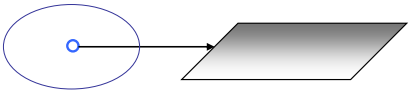
O(D) bounds on distance minima/maxima:



$$\min_i \|x - x_i\| \geq \sum_d^D \left[\max\left\{(l_d - x_d)^2, 0\right\} + \max\left\{(x_d - u_d)^2, 0\right\}\right]$$
$$\max_i \|x - x_i\| \leq \sum_d^D \max\left\{(u_d - x_d)^2, (x_d - l_d)^2\right\}$$

Exclusion and inclusion,
using point-node *kd*-tree bounds.

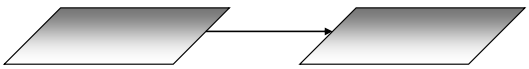
O(D) bounds on distance minima/maxima:



$$\min_i \|x - x_i\| \geq \sum_d^D \left[\max\left\{(l_d - x_d)^2, 0\right\} + \max\left\{(x_d - u_d)^2, 0\right\}\right]$$
$$\max_i \|x - x_i\| \leq \sum_d^D \max\left\{(u_d - x_d)^2, (x_d - l_d)^2\right\}$$

Exclusion and inclusion,
using *kd*-tree node-node bounds.

O(D) bounds on distance minima/maxima:

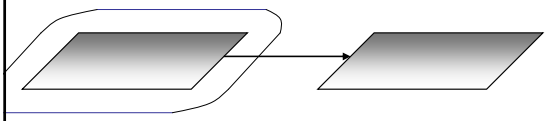


(Analogous to point-node bounds.)

Also needed:
Nodewise bounds.

Exclusion and inclusion,
using *kd*-tree node-node bounds.

O(D) bounds on distance minima/maxima:



(Analogous to point-node bounds.)

Also needed:
Nodewise bounds.

Single-tree: simple recursive algorithm (k=1 case)

```
NN(x_q,R,d_lo,x_sofar,d_sofar)
{
  if d_lo > d_sofar, return.

  if leaf(R), [x_sofar,d_sofar]=NNBase(x_q,R,d_sofar).
  else,
    [R1,d1,R2,d2]=orderByDist(x_q,R.l,R.r).
    NN(x_q,R1,d1,x_sofar,d_sofar).
    NN(x_q,R2,d2,x_sofar,d_sofar).
}
```

Single-tree → Dual-tree

- x_q → Q
- $d_{lo}(x_q,R)$ → $d_{lo}(Q,R)$
- x_{sofar} → $\underline{x}_{sofar}$, d_{sofar} → $\underline{d}_{sofar}$
- store $Q.d_{sofar}=\max_Q \underline{d}_{sofar}$
- 2-way recursion → 4-way recursion

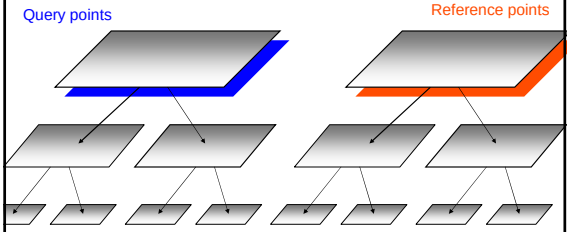
- N x O(logN) → O(N)

Dual-tree: simple recursive algorithm (k=1)

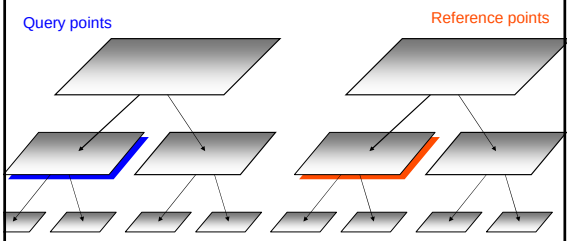
AllNN($Q,R,d_{lo},\underline{x}_{sofar},\underline{d}_{sofar}$)
{
  if $d_{lo} > Q.d_{sofar}$, return.
  if leaf(Q) & leaf(R),
    $[\underline{x}_{sofar},\underline{d}_{sofar}]$=**AllNNBase**($Q,R,\underline{d}_{sofar}$). $Q.d_{sofar}=\max_Q \underline{d}_{sofar}$.
  else if !leaf(Q) & leaf(R), …
  else if leaf(Q) & !leaf(R), …
  else if !leaf(Q) & !leaf(R),
    [R1,d1,R2,d2]=orderByDist(Q.l,R.l,R.r).
    AllNN($Q.l,R1,d1,\underline{x}_{sofar},\underline{d}_{sofar}$).
    AllNN($Q.l,R2,d2,\underline{x}_{sofar},\underline{d}_{sofar}$).
    [R1,d1,R2,d2]=orderByDist(Q.r,R.l,R.r).
    AllNN($Q.r,R1,d1,\underline{x}_{sofar},\underline{d}_{sofar}$).
    AllNN($Q.r,R2,d2,\underline{x}_{sofar},\underline{d}_{sofar}$).
    $Q.d_{sofar} = \max(Q.l.d_{sofar},Q.r.d_{sofar})$.
}

Dual-tree traversal

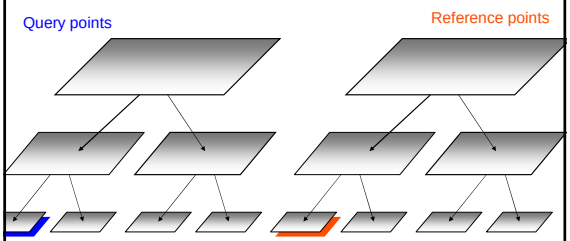
(depth-first)



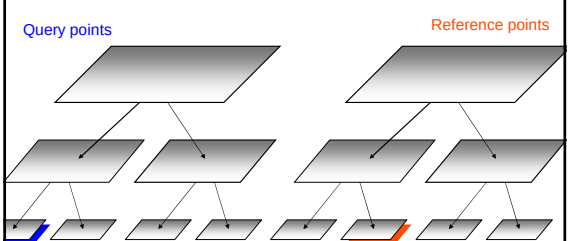
Dual-tree traversal



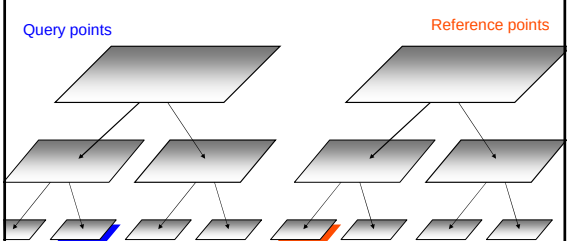
Dual-tree traversal

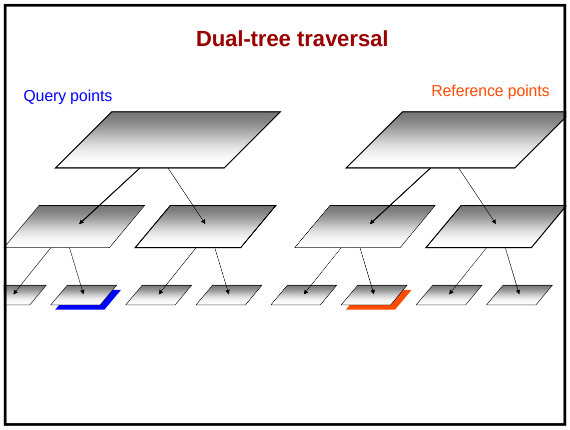


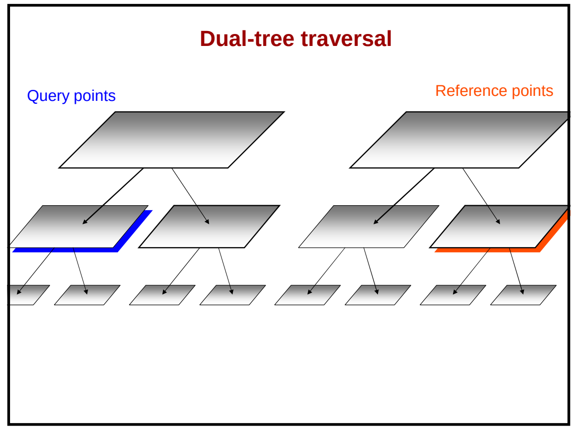
Dual-tree traversal

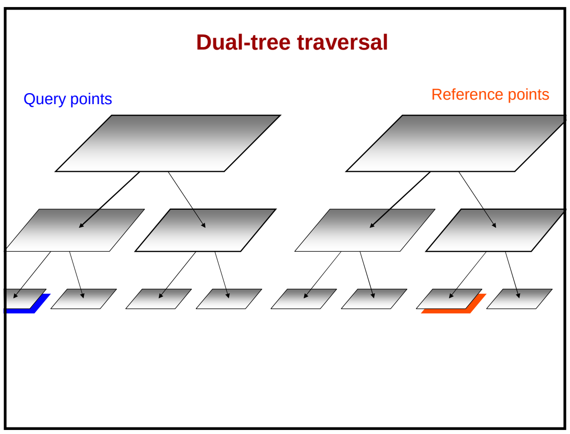


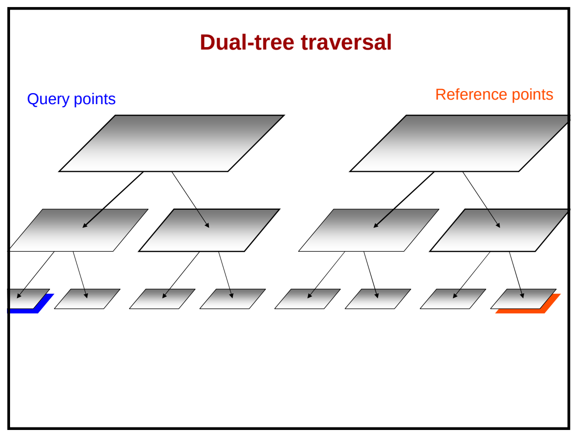
Dual-tree traversal

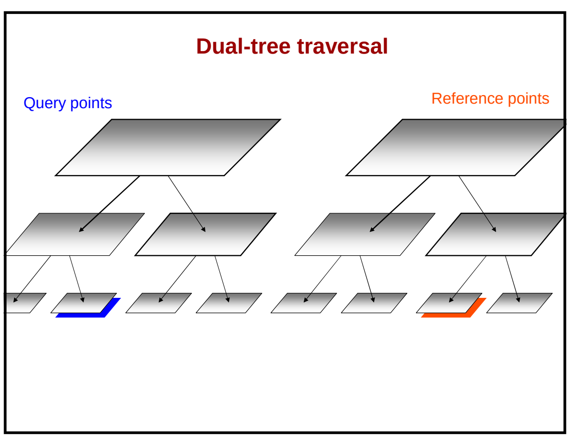


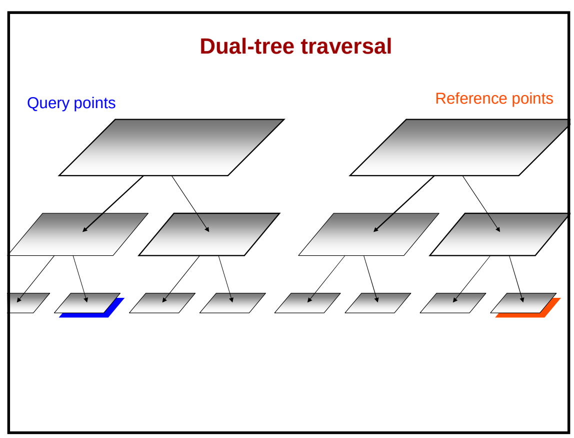

Dual-tree traversal
Query points
Reference points

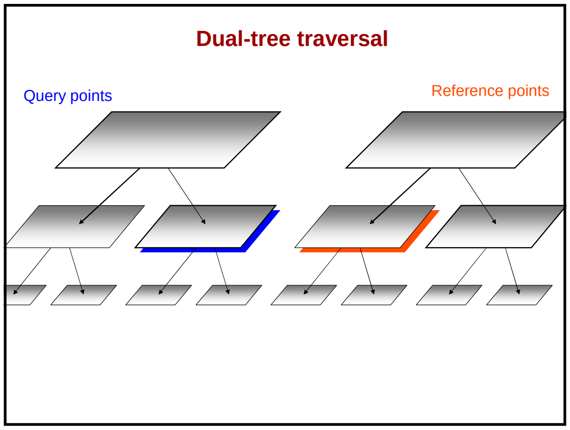

Dual-tree traversal
Query points
Reference points

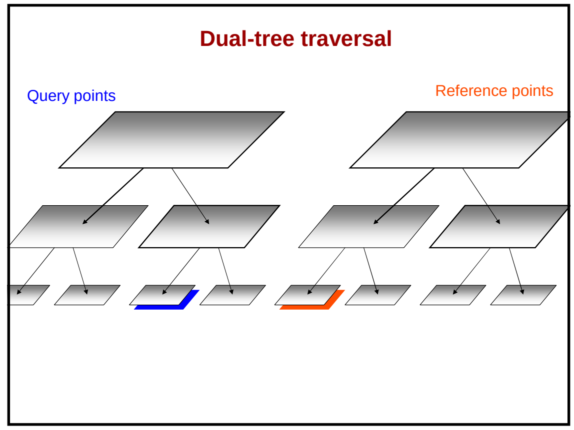

Dual-tree traversal
Query points
Reference points

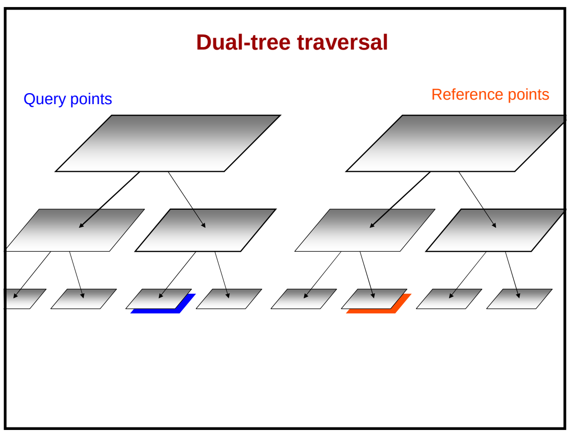

Dual-tree traversal
Query points
Reference points

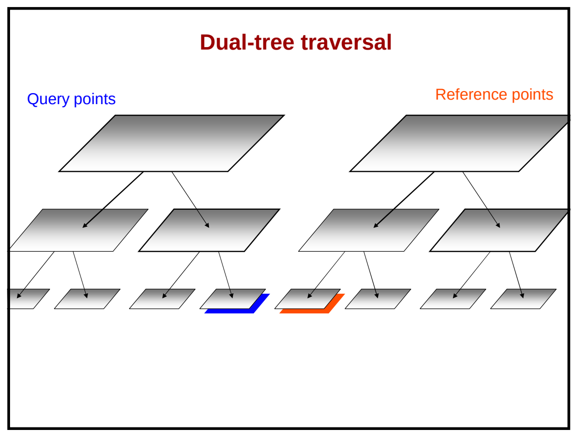

Dual-tree traversal
Query points
Reference points

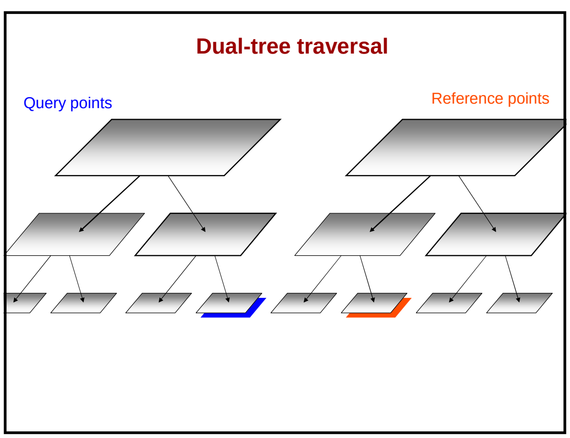

Dual-tree traversal
Query points
Reference points

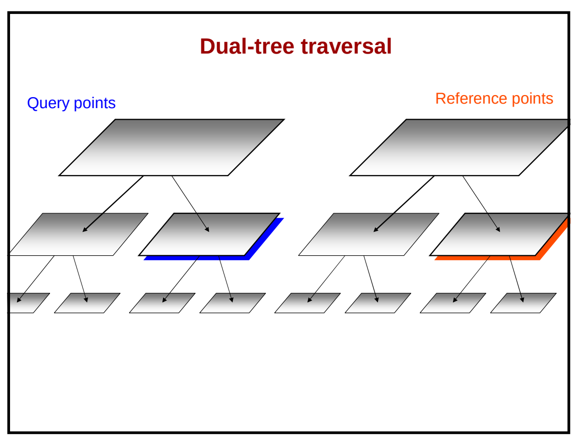

Dual-tree traversal
Query points
Reference points


Dual-tree traversal
Query points
Reference points

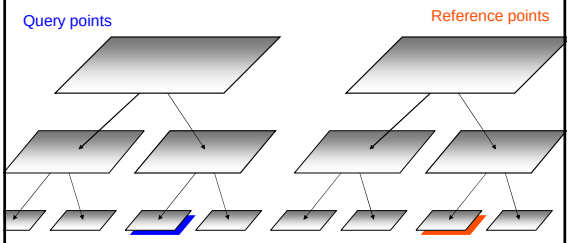

Dual-tree traversal
Query points
Reference points


Dual-tree traversal
Query points
Reference points

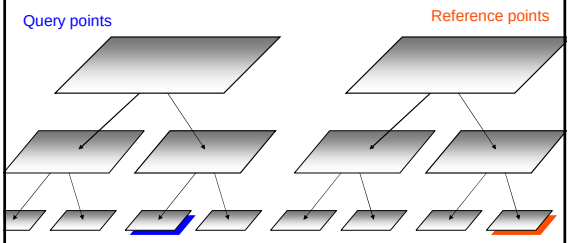

Dual-tree traversal
Query points
Reference points


Dual-tree traversal
Query points
Reference points

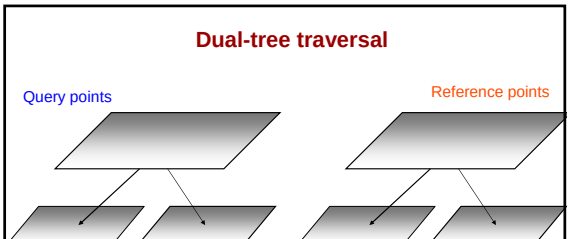
Dual-tree traversal



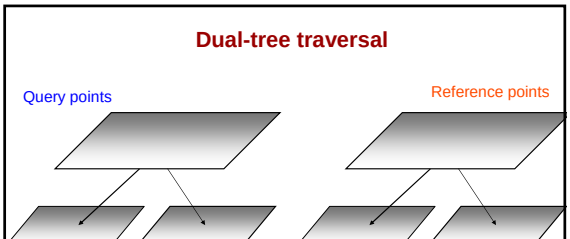
Dual-tree traversal



Dual-tree traversal



Dual-tree traversal



Meta-idea:

Higher-order Divide-and-conquer

Generalizes divide-and-conquer of a single set to divide-and-conquer of multiple sets.

Break each set into pieces.

Solving the sub-parts of the problem and combining these sub-solutions appropriately might be easier than doing this over only one set.

[Gray thesis, 2003]

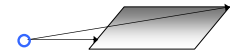
Kernel density estimation (All-queries version)

$$\forall x_q : \hat{f}(x_q) = \frac{1}{N}\sum_{r \neq q}^{N} K_h(\|x_q - x_r\|)$$

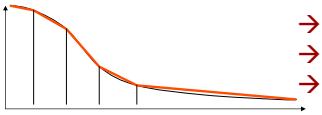
→ Now we need to do **function approximation**

Finite-difference function approximation.

$$f(x) = f(x_i) + \frac{1}{2}\left(\frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}\right)(x - x_i)$$



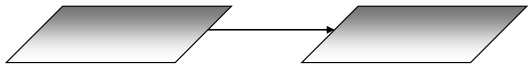
$$K_h(\delta) = K_h(\delta_{lo}) + \frac{1}{2}\left(\frac{K_h(\delta_{hi}) - K_h(\delta_{lo})}{\delta_{hi} - \delta_{lo}}\right)(\delta - \delta_{lo})$$



→stopping rule
→no tweak parameters
→hard error bounds

Approximation within dual-tree.

O(D) bounds on kernel sum contribution of R to Q:



Use distance bounds to obtain

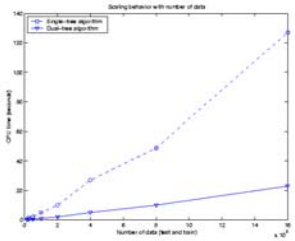
$$K_h(\delta_{hi}), K_h(\delta_{lo})$$

Also needed:
Nodewise bounds.

Speedup Results (KDE)

N	naïve	dual-tree
12.5K	7	.12
25K	31	.31
50K	123	.46
100K	494	1.0
200K	1976*	2
400K	7904*	5
800K	31616*	10
1.6M	35 hrs	23

5500x



One order-of-magnitude speedup over single-tree at ~2M points

Speedup Results

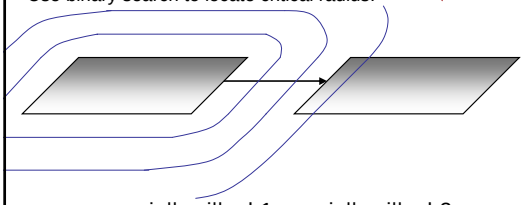
N	Epan.	Gauss.
12.5K	.12	.32
25K	.31	.70
50K	.46	1.1
100K	1.0	2
200K	2	5
400K	5	11
800K	10	22
1.6M	23	51

Only about
2 - 2.5 times less
efficient

Exclusion and inclusion,
on multiple radii simultaneously.

Application of HODC principle

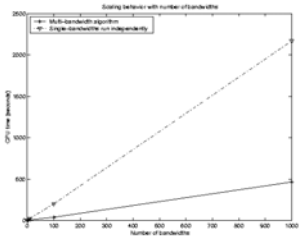
Use binary search to locate critical radius:



min||x-xi|| < h1 => min||x-xi|| < h2

Also needed:
b_lo,b_hi are arguments; store bounds for each b

Speedup Results



One order-of-magnitude speedup
over single-radius at ~10,000 radii

These are all examples of…

Generalized N-body problems

All-NN: $\{\forall, \arg\min, \delta, \cdot\}$

2-point: $\{\Sigma, \Sigma, I_h(\delta), w\}$

3-point: $\{\Sigma, \Sigma, \Sigma, I_H(\delta), w\}$

KDE: $\{\forall, \Sigma, K_h(\delta), \cdot; \{h\}\}$

SPH: $\{\forall, \Sigma, K_h(\delta), w; t\}$

General theory and toolkit for designing algorithms for such problems