29.05.19

Ann-Algo

Q = prequne of pointers to nodes in kd-tree sorted by distance to q (query pt)

p° = candidate of NN,
init v = root
$p^\circ = p^t$ at ∞
Q = empty

while $d(v,q) < \frac{1}{1+\varepsilon} d(p^\circ, q)$

    while v is inner node
        let v_1, v_2 be children of v_1
            with v_1 closer to q
        insert v_2 into Q
        $v := v_1$

    if $d(p_v, q) < d(p^\circ, q)$.
        $p^\circ = p_v$
        v = extractMin(Q)

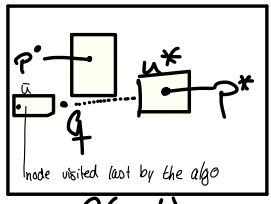
... return p°

Correctness:

Let u^* be leaf containing $p^* = u$

a) algo visits $u^* \rightarrow p^\circ = p^*$

b) u^* is not visited $\rightarrow p^\circ \neq p^*$



$$d(p^*, q) \geq d(f, u^*) \geq d(q, \bar{u})$$

(d is metric; closer node are visited first)

$d(q, \bar{u}) \geq \frac{1}{1+\varepsilon} d(q, p^\circ)$ // by condition in the outer while

$\Rightarrow d(q, p^*) \geq \frac{1}{1+\varepsilon} \cdot d(q, p^\circ) \rightarrow q^\circ$ is $(1+\varepsilon)$-ANN

Time

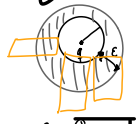
One outer while iteration: 1x extract Min

inner loop: $O(\log n)$ times insert op. into $\mathcal{L}$

normal heaps $O(\log n)$ for extract & insert $\Rightarrow$ ANN $\in O(\mathcal{L} \cdot \log^2 n)$

Fibonacci heaps: $O(\log n)$ extract, insert oper. has $O(1)$ amortized time. $\Rightarrow$ ANN $\in O(\mathcal{L} \cdot \log n)$

$\mathcal{L} = \#\text{leaves} = O(\log^{d-1} n)$



Argument considers # boxes that can penetrate anulus of thickness ε, where boxes emerge from kd-tree with longest-side splitting!

"Best" ANN algorith

Randomized kd-tree (RKD):

For a split: - determine D many axes (dim's) with the longest variance

- choose one of those randomly
- split across median

D ≈ 5 is fine in most cases

RKD forest.

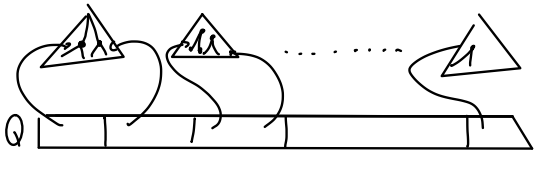
build several RKD's over pt set P

ANN search:

maintain one p-queue (for all RKD's forest)

beginning: descent into each RKD tree down to the leaf containing q (= first iteration of outer while in orig)

- put the "other" nodes (v2's) along the way into Q



proceed as before with orig ANN algo

Application: texture synthesis

Input: Image I, texture, i.e. color of pixel depends on neighbor-hood only

Output: Image T, bigger, looking "similar" to I

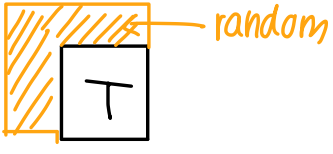
Notation: p_i := pixel $\in I$

p_o := output pixel $\in T$

$N(p)$ = neighborhood around p (shape depends on specifics of algo)

Algorithm:

Init T := empty, plus random border around (randomly colors)



for all p_o's $\in T$ in scan line order:

NN search with d-dim pts, where d = # pixels in $N(p)$

find $p_i \in I$ with

$d(N(p_o), N(p_i)) = \min$

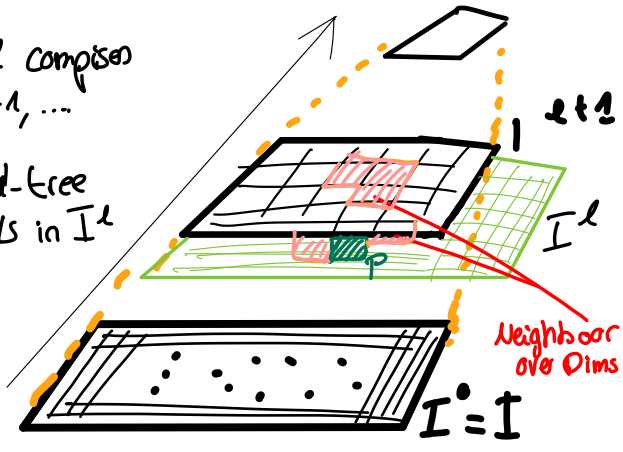
$p_o = p_i$

Build image pyramid over I

$I_0 = I$, I^{l+1} created from I^l
by averaging 2×2 pixels (or other smoothing op)

Neighborhood $N(p)$ on level l composed pixels on levels l, $l+1$, ...

For each level l: build kd-tree over all $N(p)$'s for all pixels in I^l



Build image pyramid for T, but top-down!

On every level l: images T^{l+1}, ...
are already created,
create random border around T^l,
proceed as before.