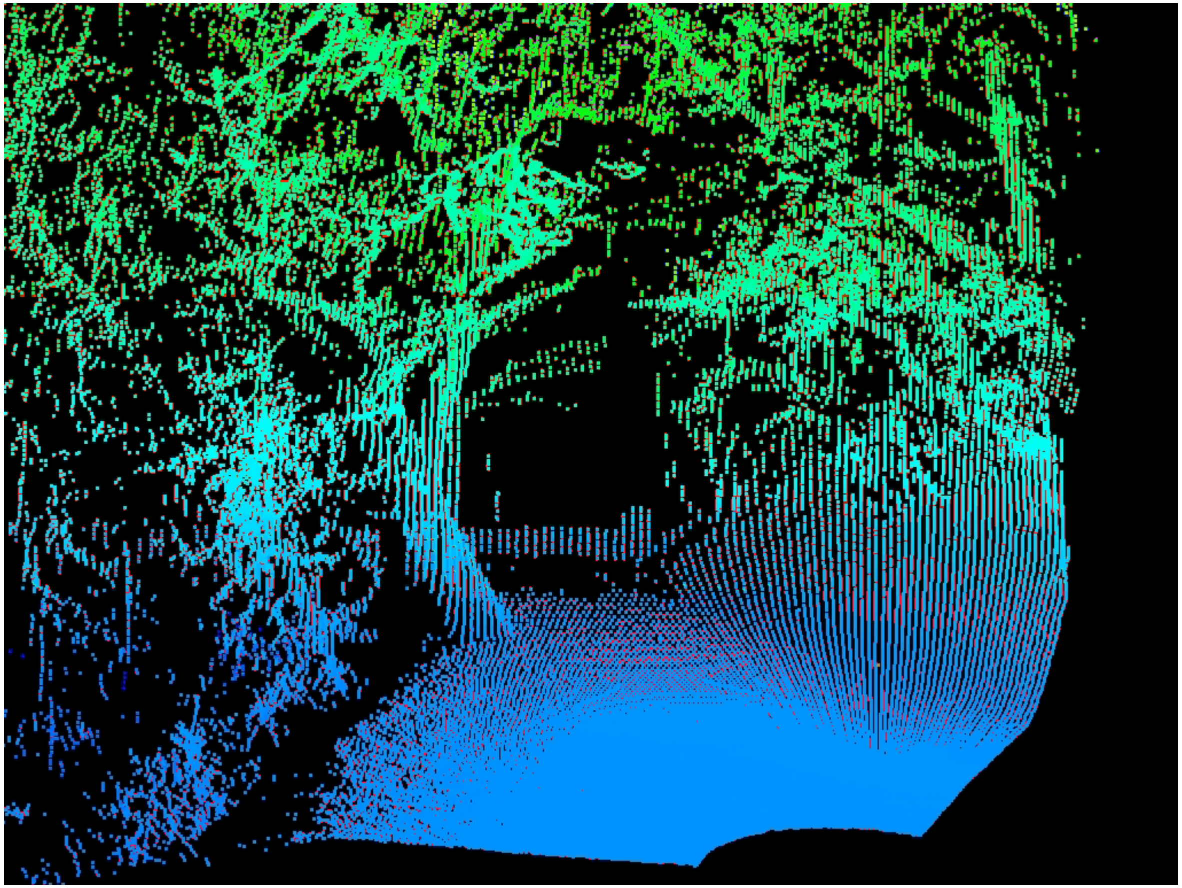
Data Structure for Real-Time Processing in 3-D

Jean-François Lalonde, Nicolas Vandapel
and Martial Hebert

Carnegie Mellon University

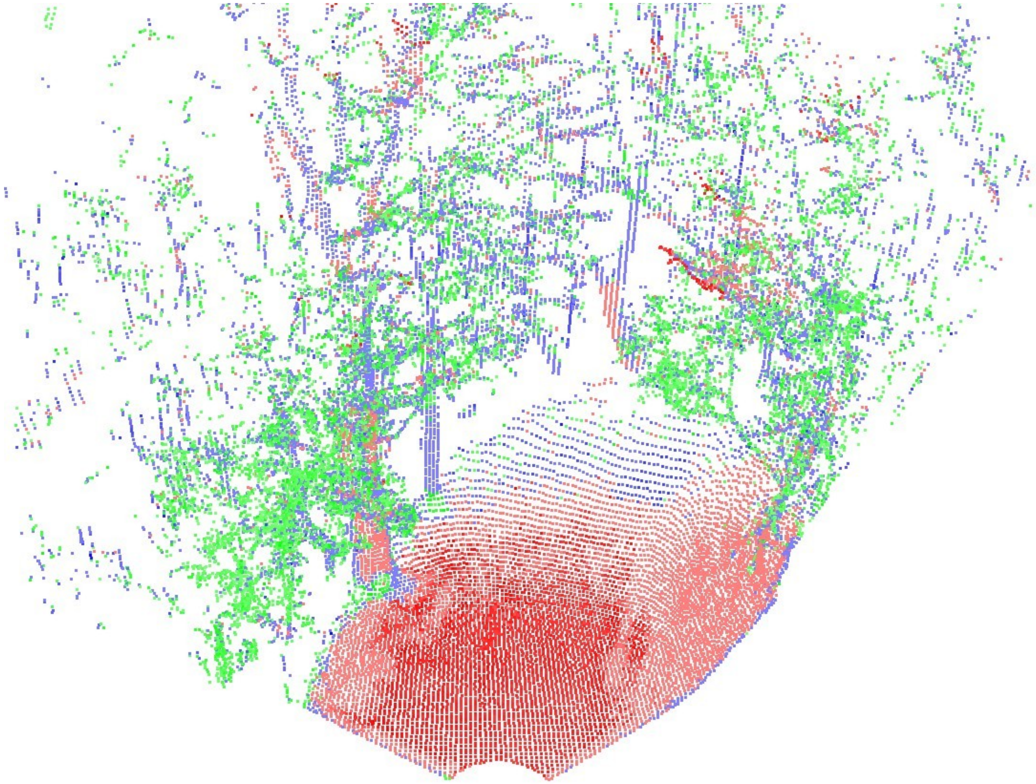
Problem

Dynamic processing of large 3-D point cloud data from ladar



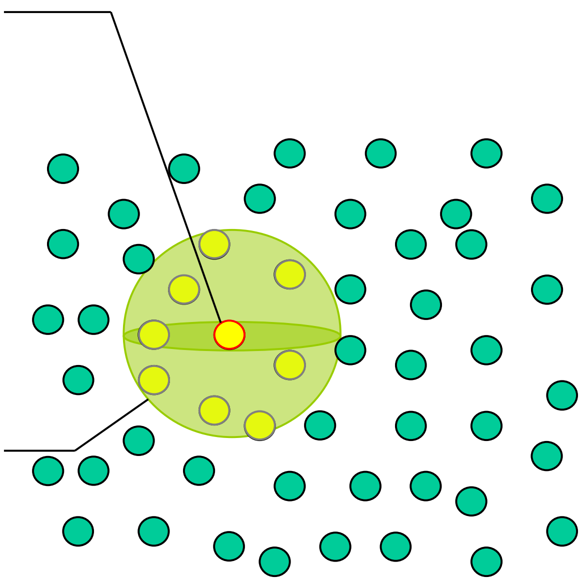
Example

- Terrain classification
  - Through local processing [Vandapel-ICRA04]



Local computation on 3-D point sets

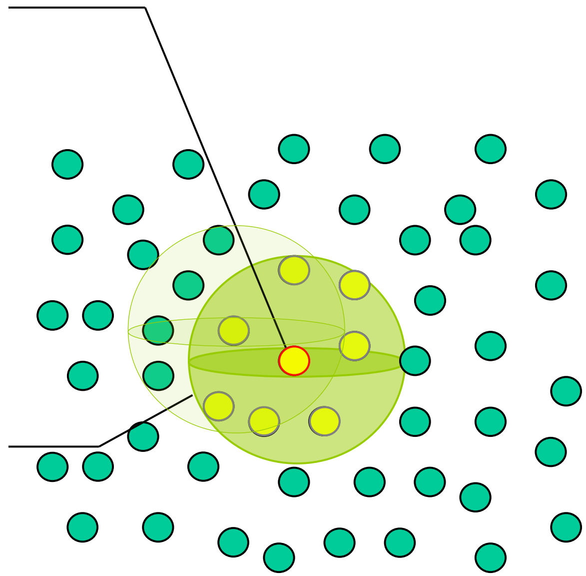
Point of interest
Scan through all points in the dataset



Support region
Local computation on highlighted points

Local computation on 3-D point sets

Point of interest
Scan through all points in the dataset



Support region
Local computation on highlighted points

Very expensive, but can reuse data from overlap regions

Local computation on 3-D point sets: example

- Compute scatter matrix within support volume
- Extract principal components
- Features are linear combination of eigenvalues [Tang-PAMI04]

Scatter	Planar	Linear
$\lambda_0 \approx \lambda_1 \approx \lambda_2$	$\lambda_0 \approx \lambda_1 >> \lambda_2$	$\lambda_0 >> \lambda_1 \approx \lambda_2$

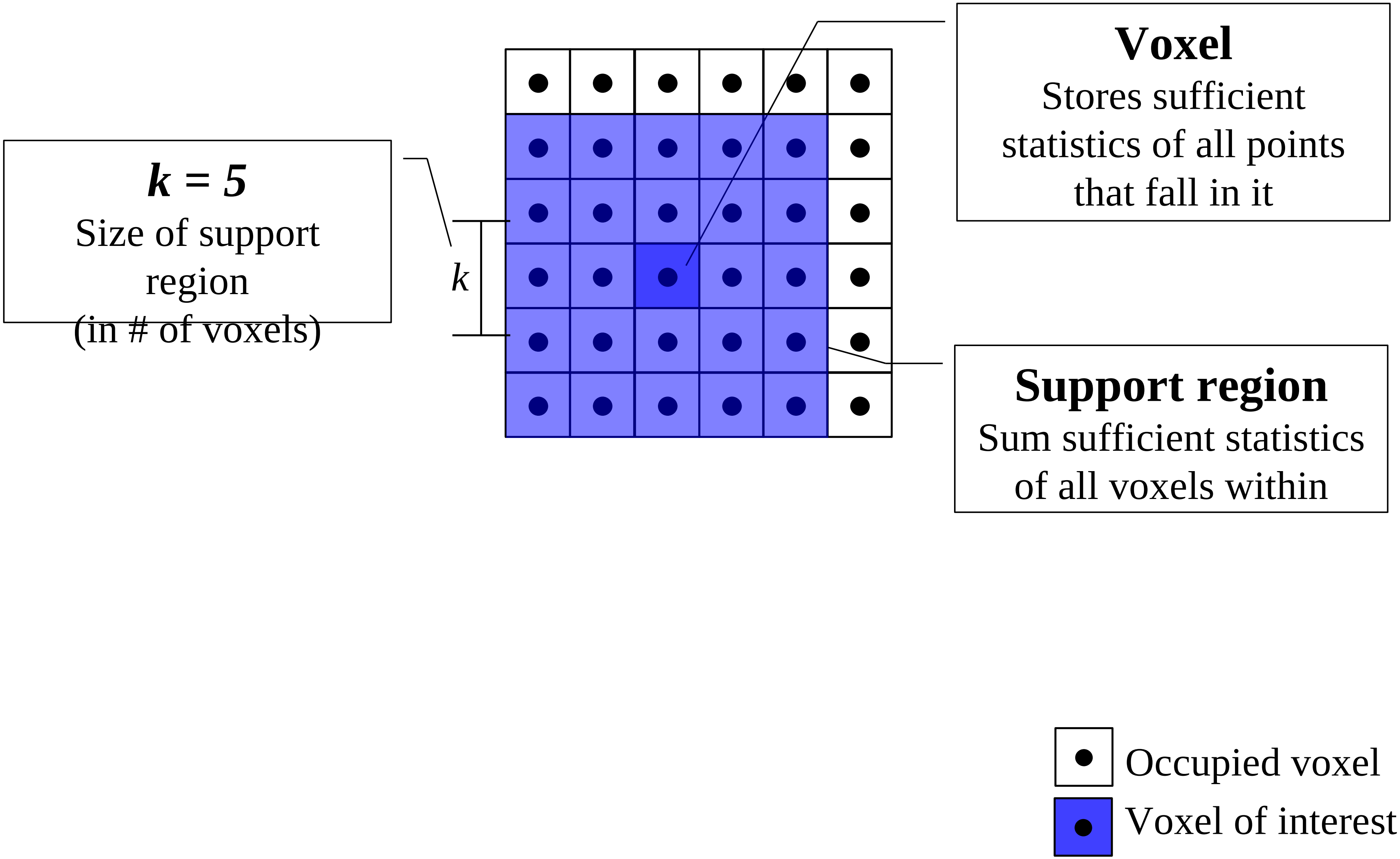
$F_{scatter} = \lambda_0$	$F_{planar} = (\lambda_1 - \lambda_2) \cdot e_2$	$F_{linear} = (\lambda_0 - \lambda_1) \cdot e_0$
Scatter	Planar	Linear

- Voxelize data
- Store sufficient statistics for scatter matrix in voxels
  - Sums, sums of squared and sums of cross-products of 3-D points coordinates
  - Minimize storage, reduce amount of data without losing information for later processing
- Partial sums: suitable for data reuse

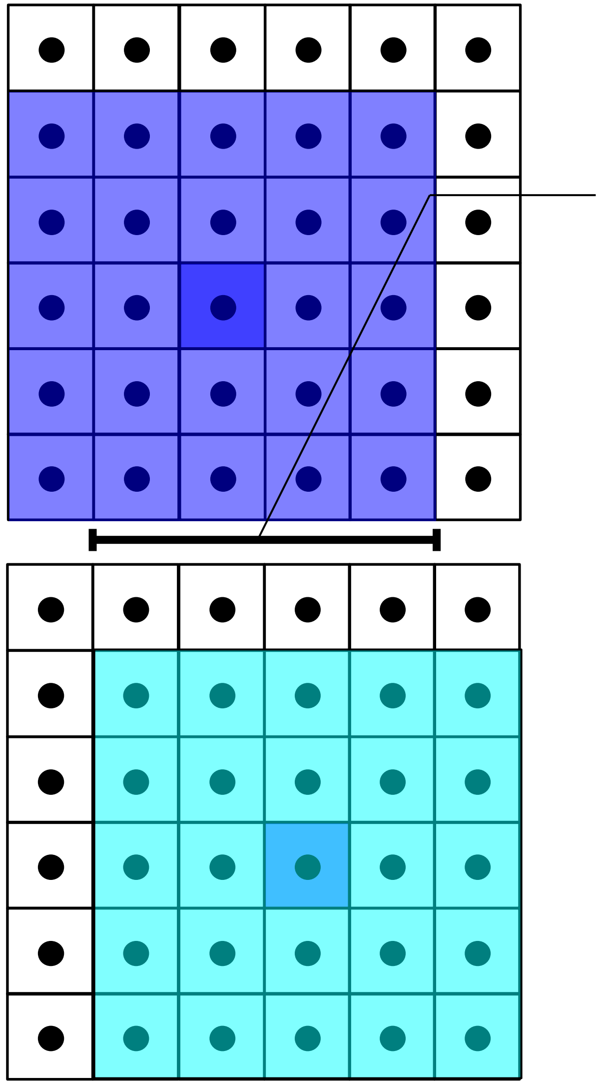
Challenges

- Nature of data
  - Ladar on a moving platform [Lacaze-AUVSI02]
    - Dynamic (accumulation)
  - Need to process data continuously
- Efficient operations
  - Insertion and access
  - Range search
    - Local computations
- Traditional techniques do not apply
  - Tree-based data structures [Samet81, Liu-NIPS04, Gray-ICML04]
    - Suitable for static and high-dimensional data

Concept – 2-D example

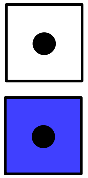


Concept – 2-D example

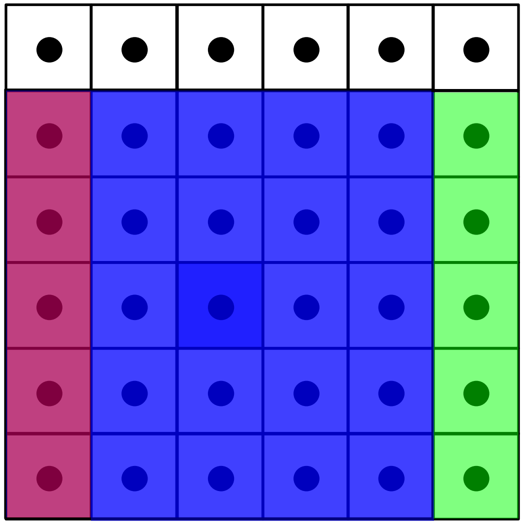


Overlap

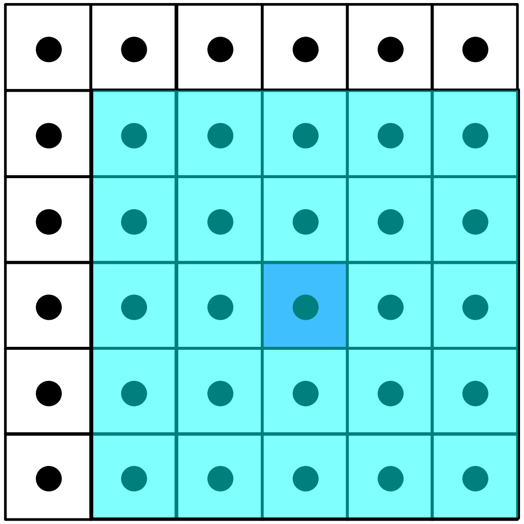
How can we reuse pre-computed data?

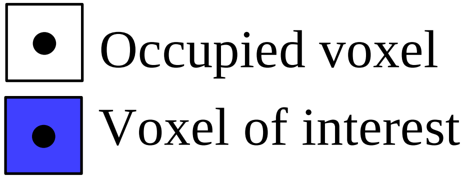


Concept – 2-D example

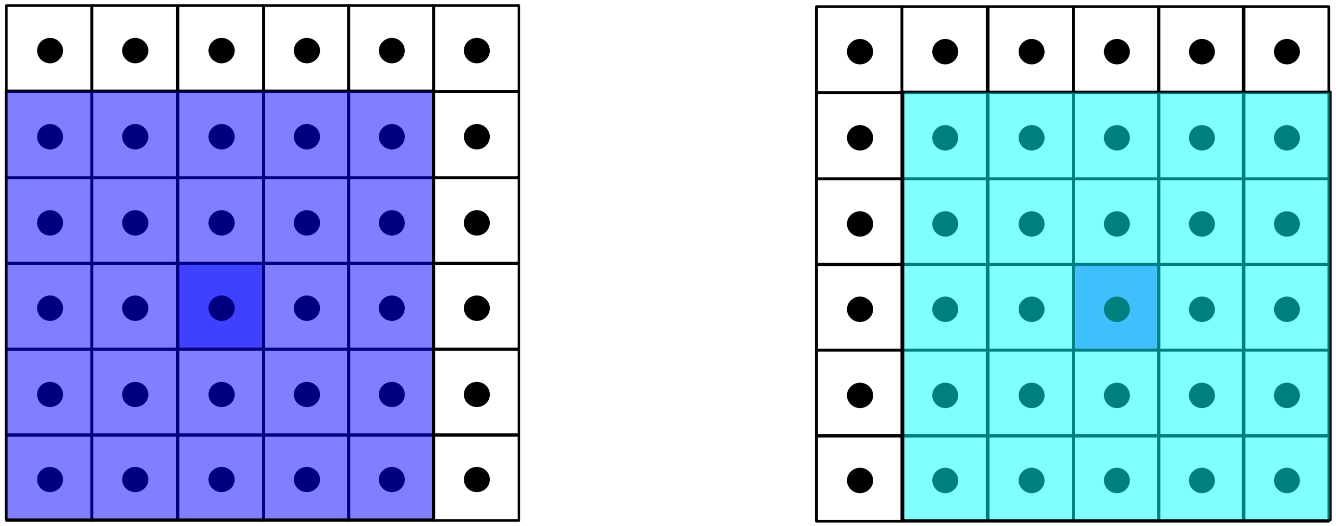


1. Start with the blue region
2. Add the green column
3. Subtract the red column

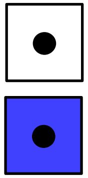




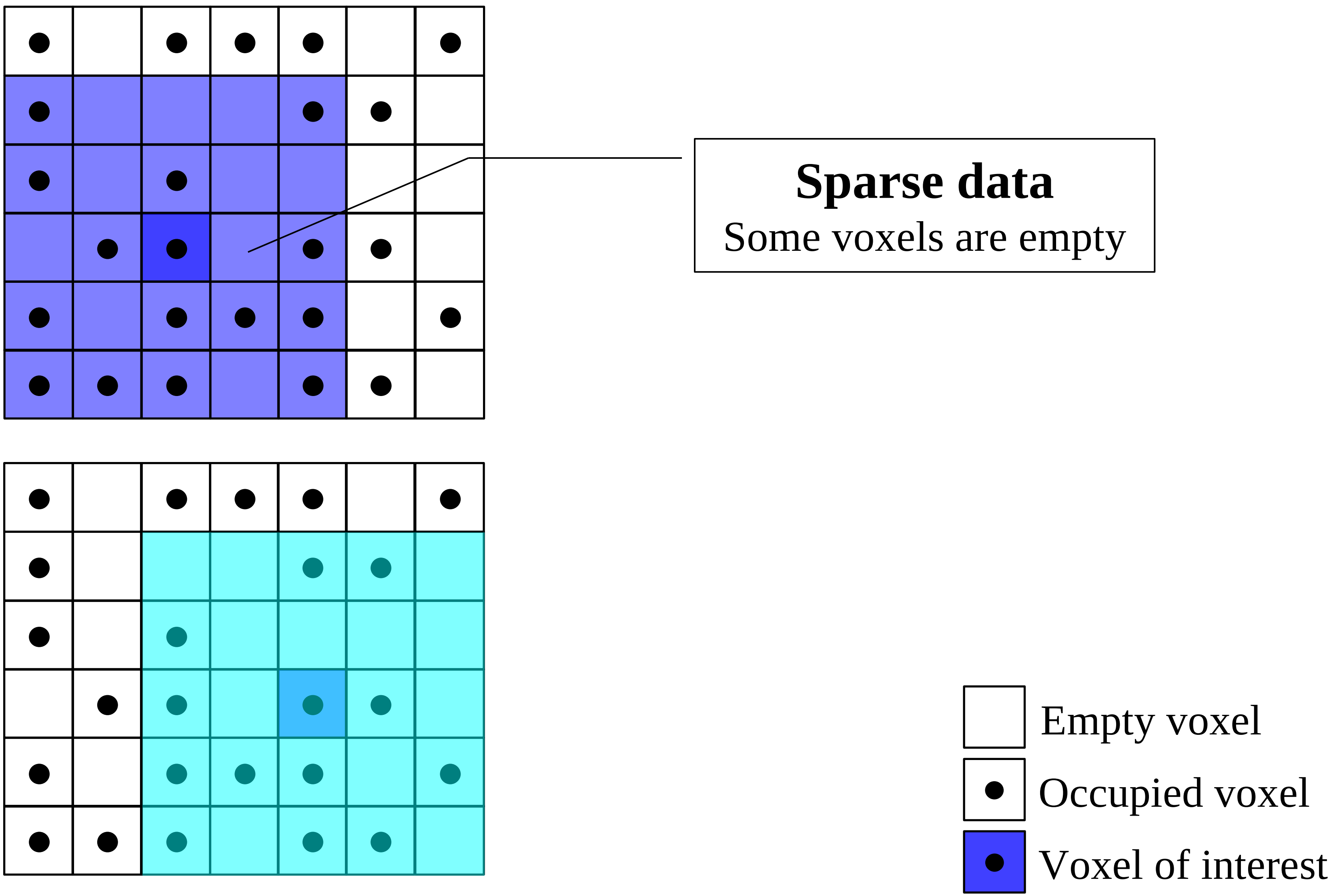
Concept – 2-D example



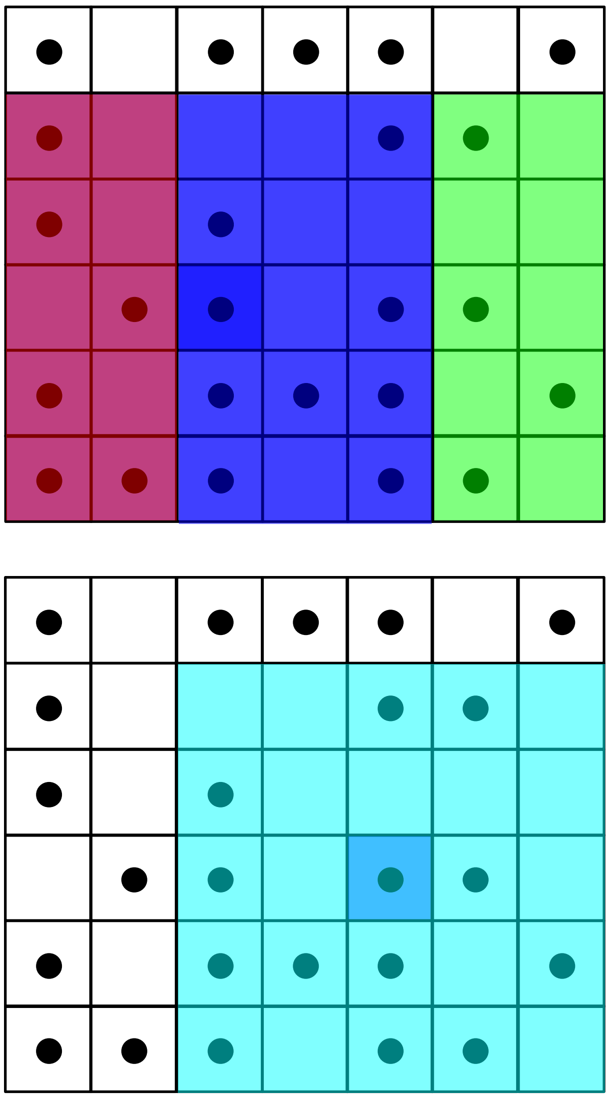
- Proven to be efficient in image processing [Faugeras93]
- Challenge in 3-D: data is sparse



2-D example, sparse data

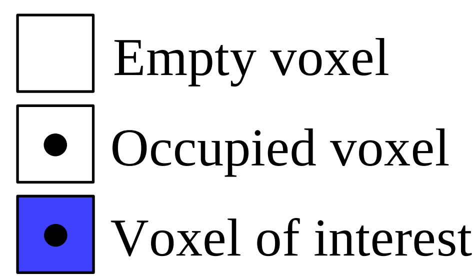


2-D example, sparse data

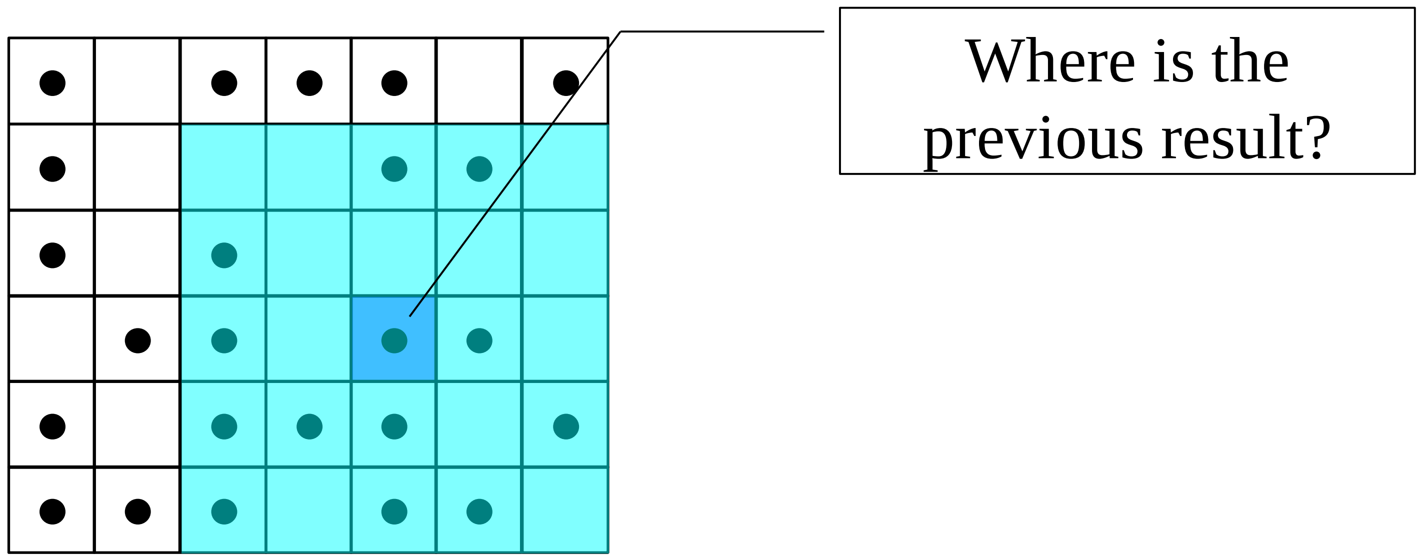


1. Start with the blue region
2. Add the green columns
3. Subtract the red columns

May not always be useful to reuse data



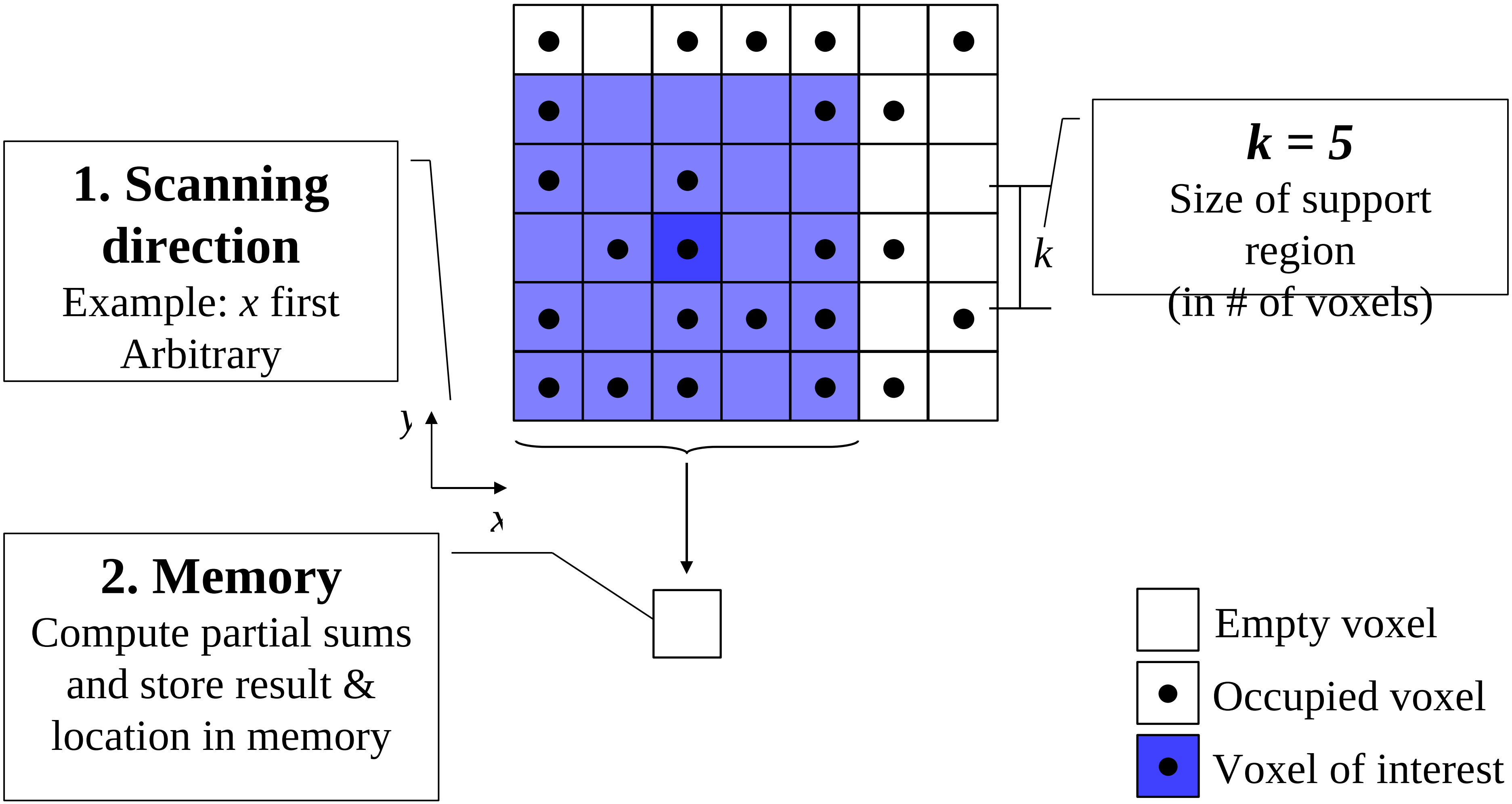
2-D example, sparse data



2 approaches:

2. Default scan
3. Optimized scan

Approach 1: default scan

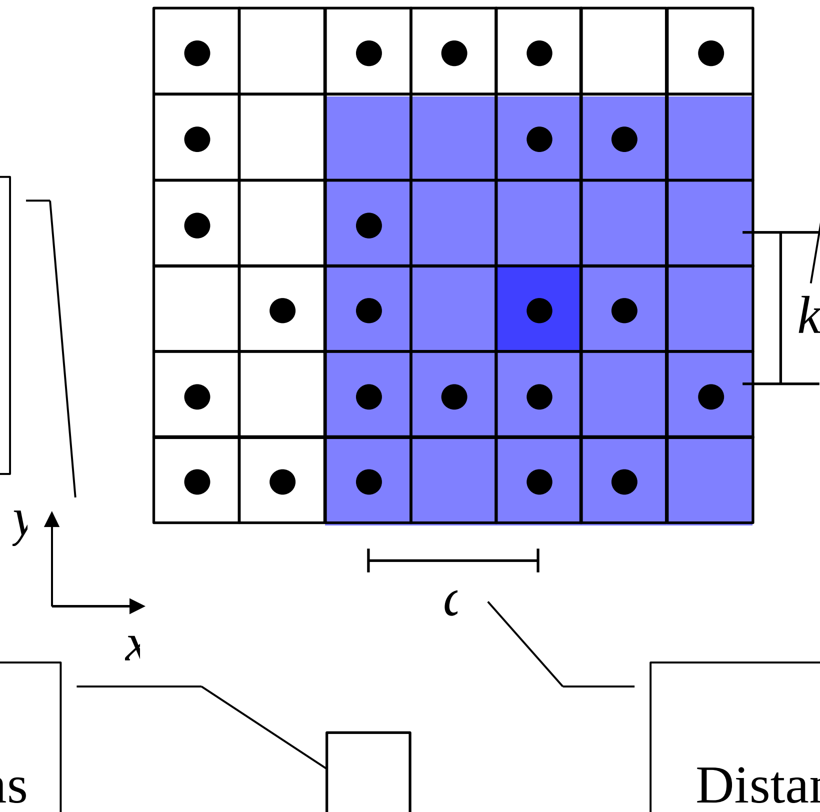


Approach 1: default scan

1. Scanning direction

Example: *x* first

Arbitrary



***k* = 5**

Size of support region

(in # of voxels)

2. Memory

Compute partial sums and store result & location in memory

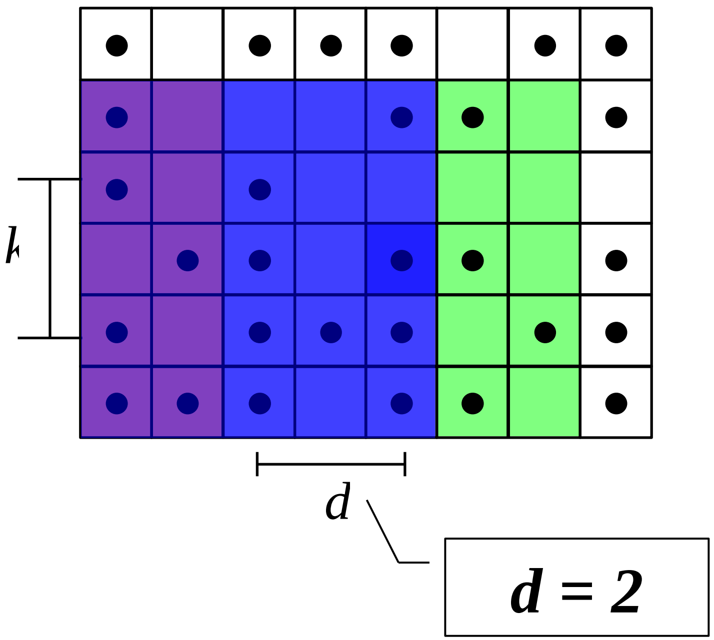
***d* = 2**

Distance between interest voxel and previous result

(in # of voxels)

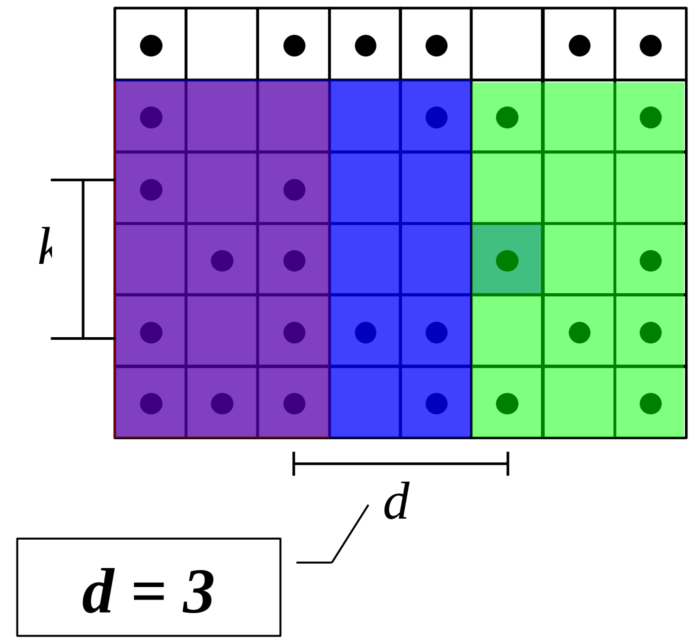
2 cases

$$d < \frac{k}{2}$$



Reuse previous results

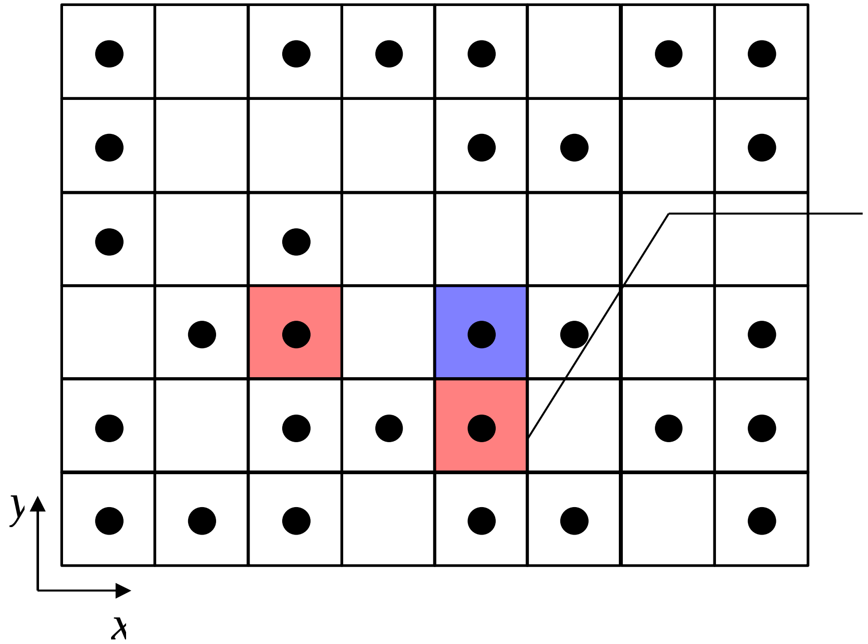
$$d > \frac{k}{2}$$



Do not reuse, recompute

Approach 2: optimized scan

- Can we do better?



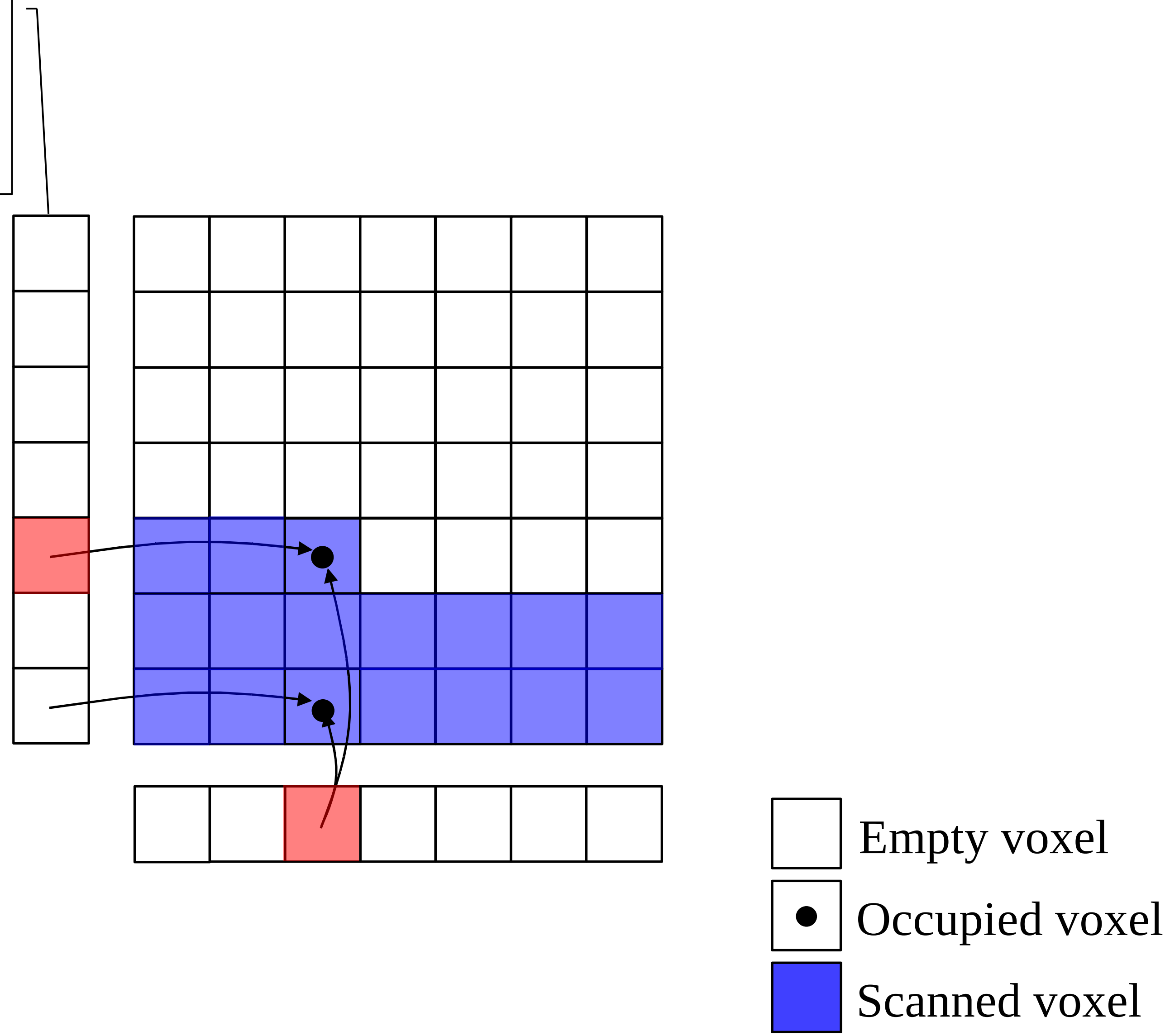
Would be better to choose the result from this voxel

- Choose closest (along *x*, *y* or *z*)

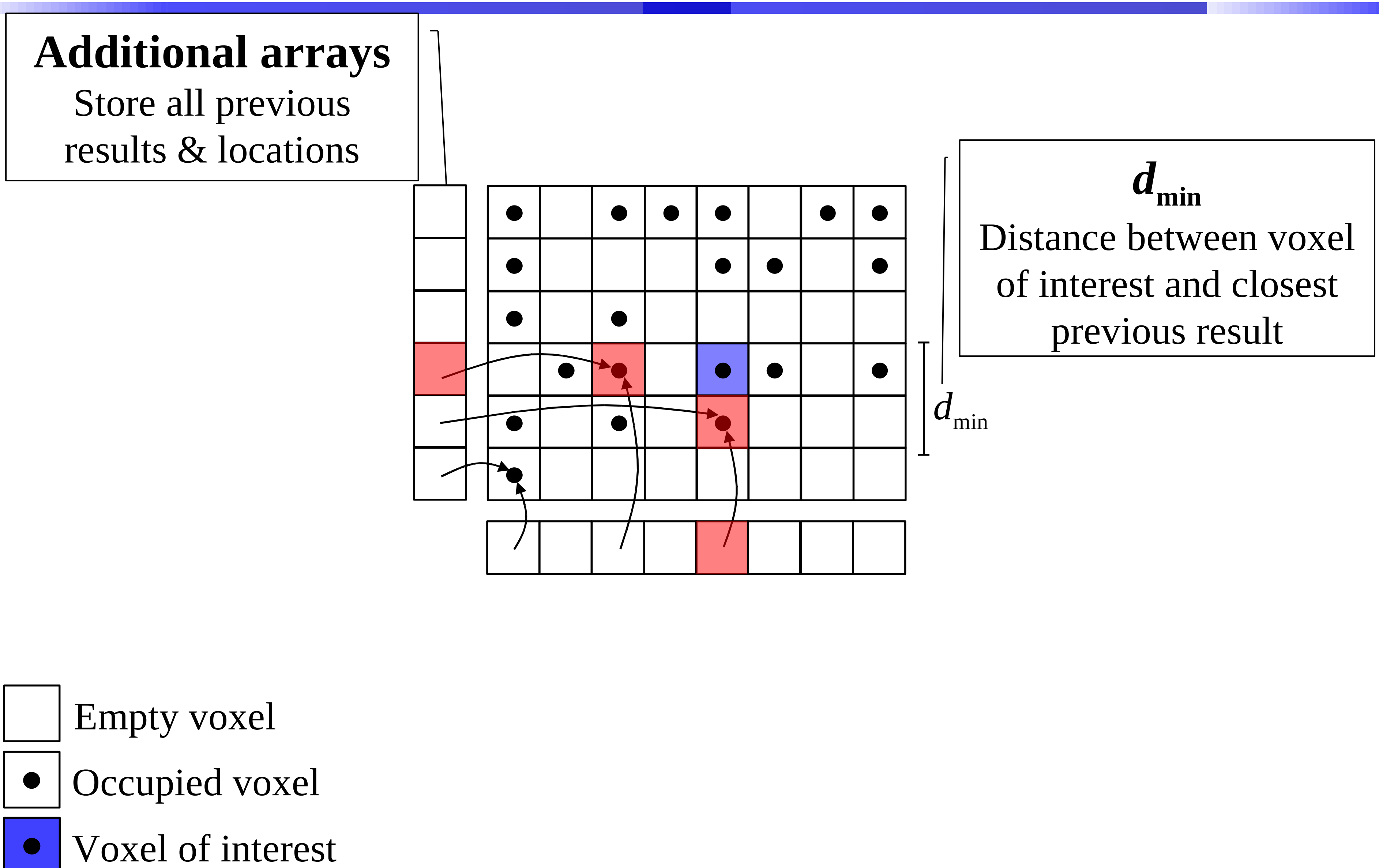
Approach 2: optimized scan

Additional arrays

Store all previous results & locations



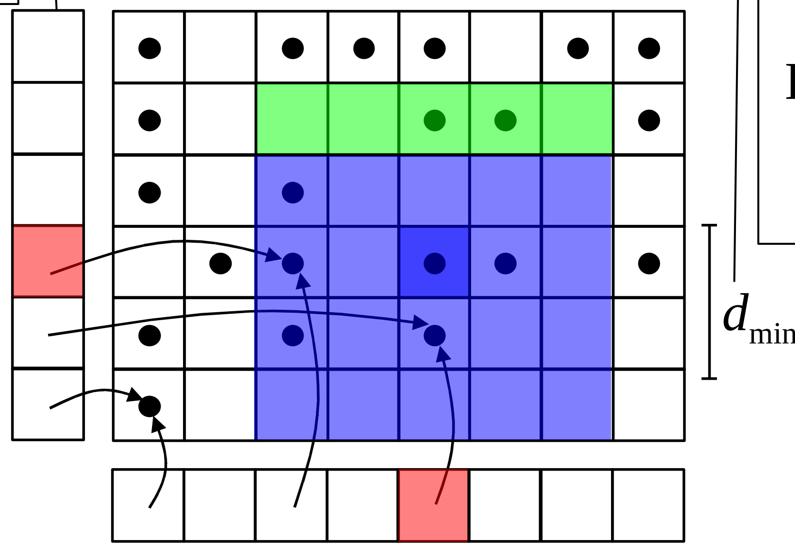
Approach 2: optimized scan



Approach 2: optimized scan

Additional arrays
Store all previous results & locations

$\boldsymbol{d_{\min}}$
Distance between voxel of interest and closest previous result



Reuse data if condition is met

$$d_{\min} < \frac{k}{2}$$

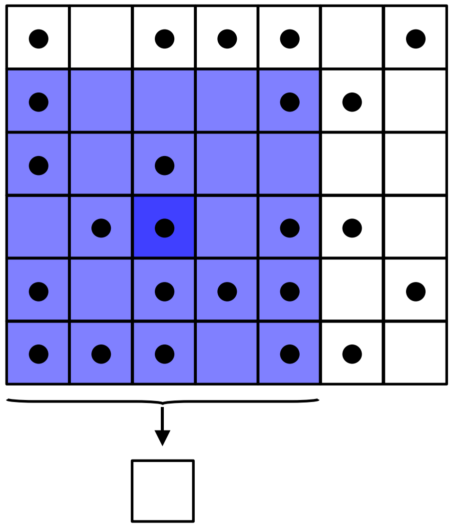
Empty voxel

Occupied voxel

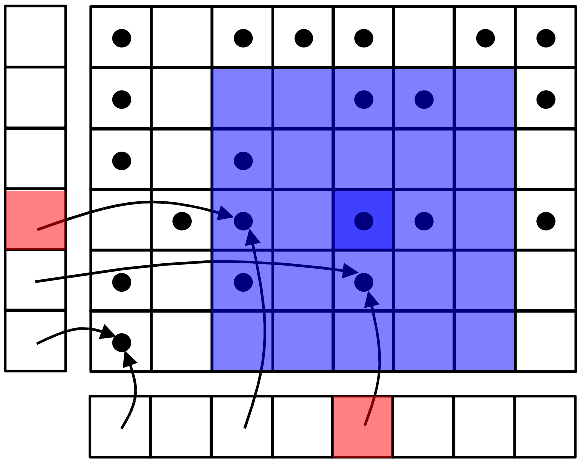
Voxel of interest

Comparison

Default scan



Optimized scan



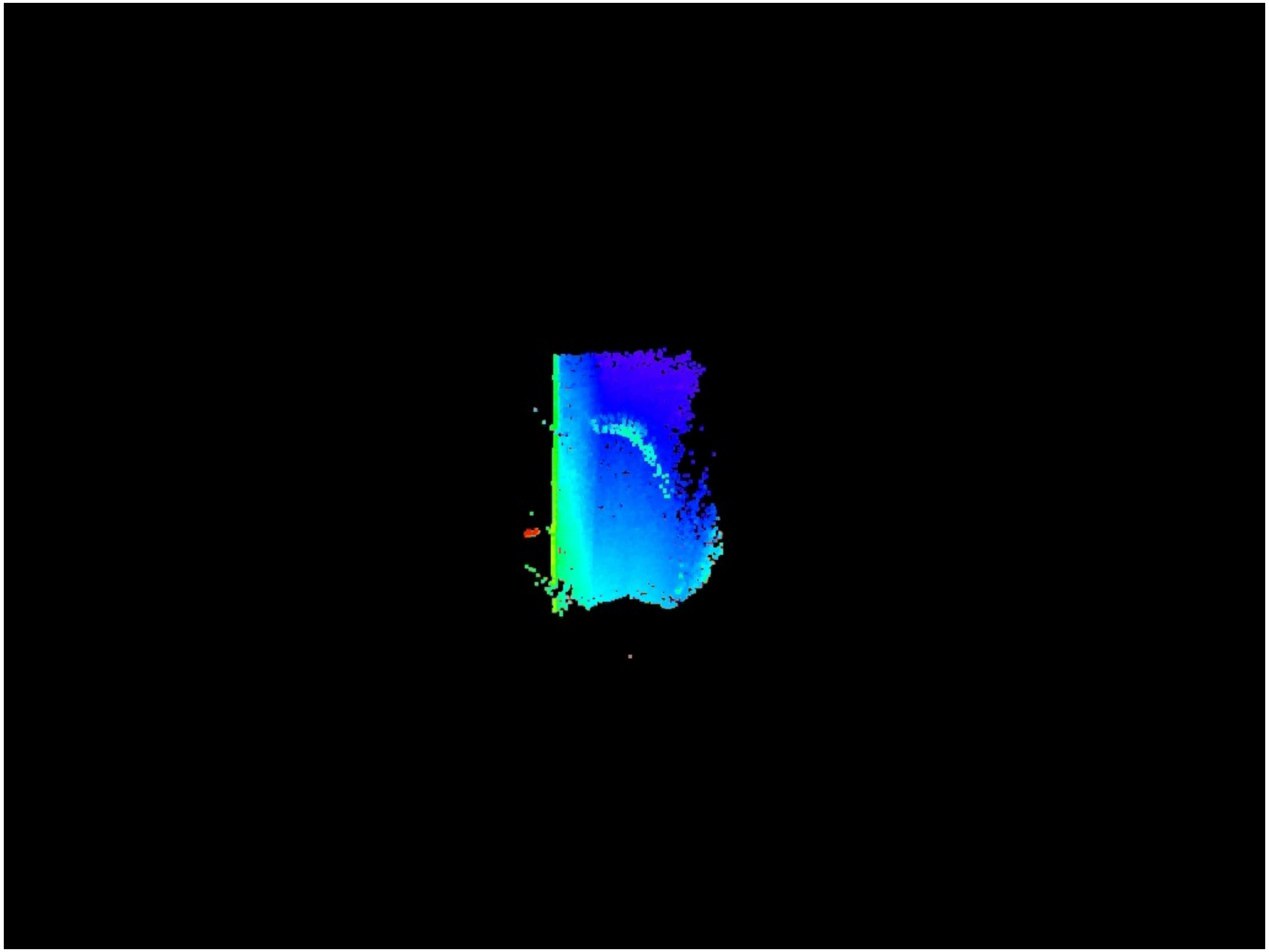
+ Very easy to implement
+ Minimal overhead
    one memory location
    one distance computation
- Dependent on scanning direction

(user input)

+ Independent on scanning direction
+ Provide highest speedup
- Harder to implement
    direction determined dynamically
- Additional overhead
    memory usage
    3 distance computations

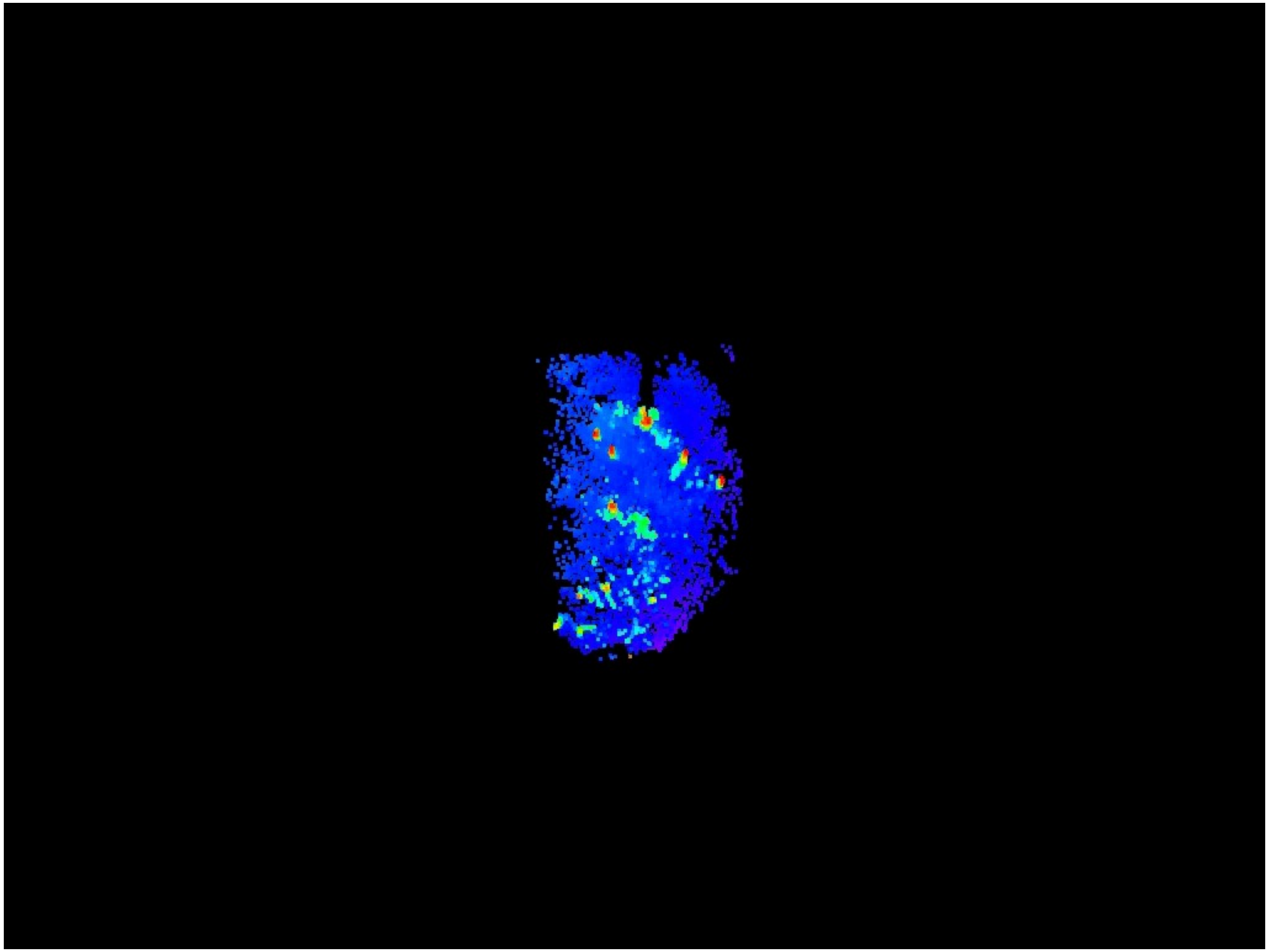
Experiments - overview

Flat ground dataset 59,000 voxels



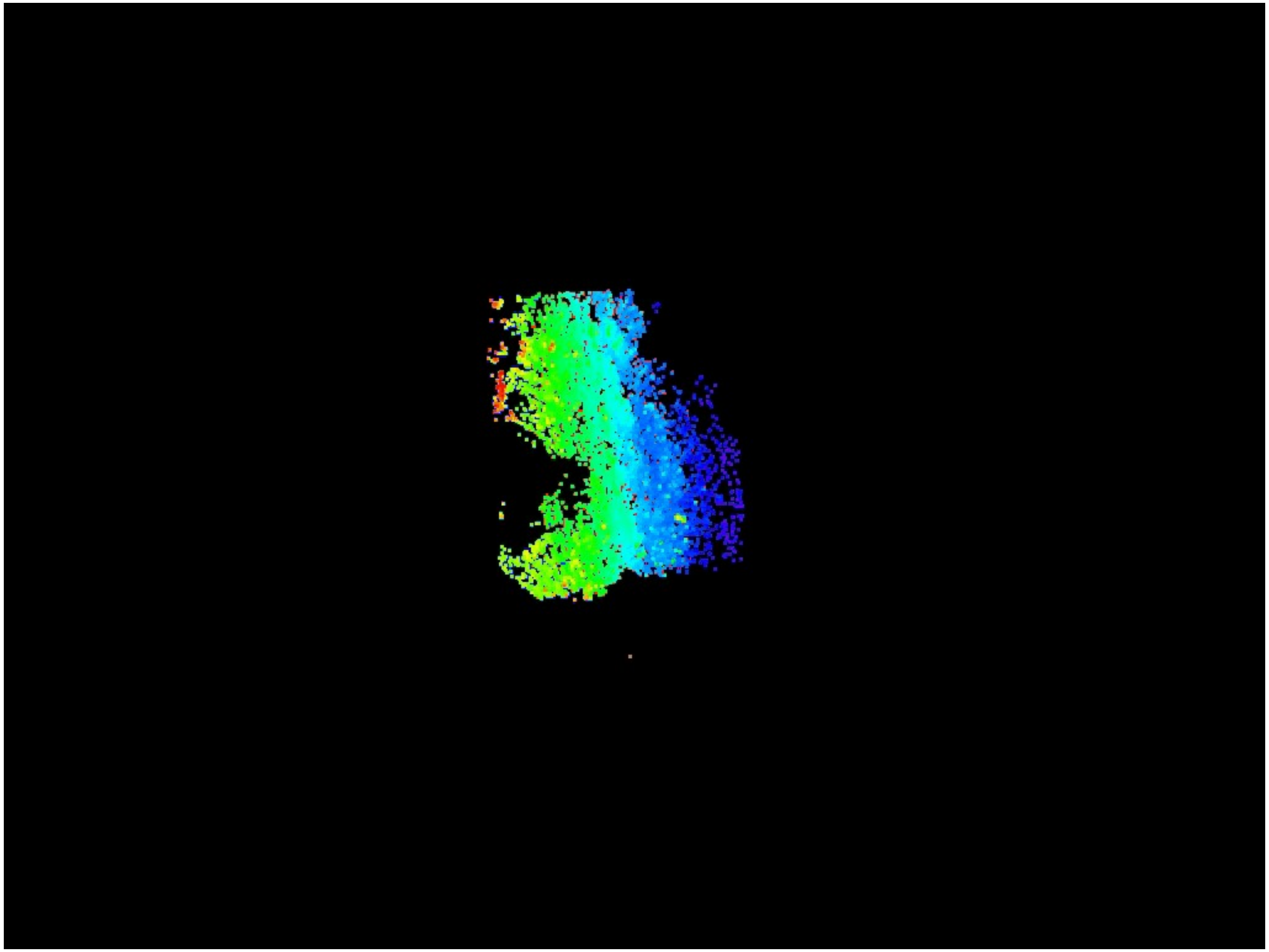
Experiments - overview

Forest dataset 112,000 voxels



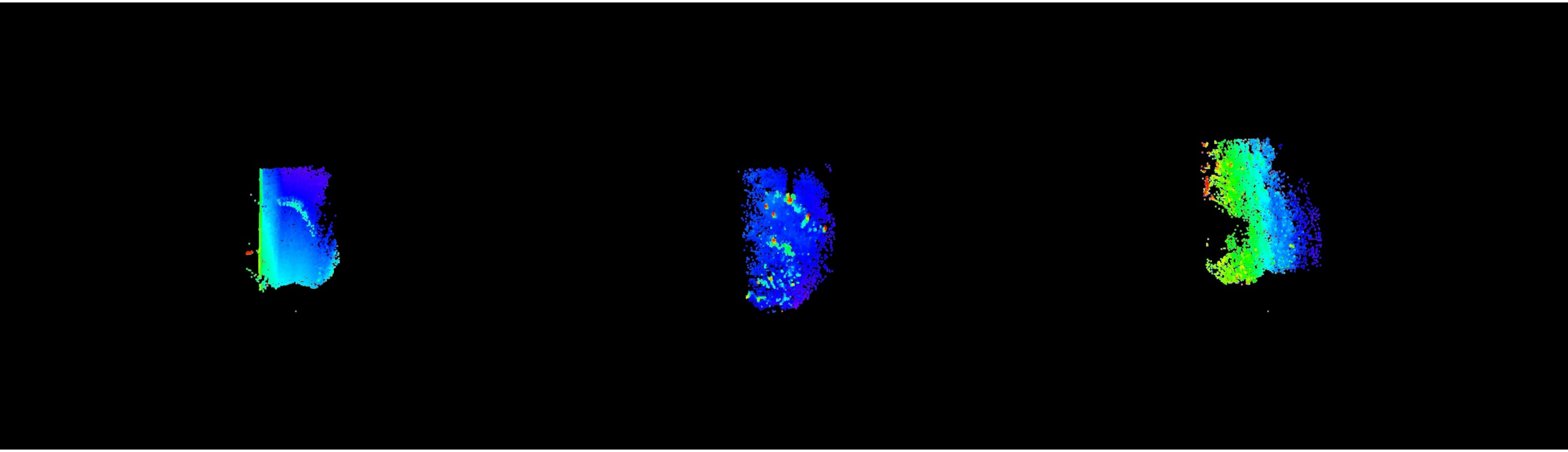
Experiments - overview

Tall grass dataset 117,000 voxels



Experiments - overview

Flat ground dataset | Forest dataset | Tall grass dataset

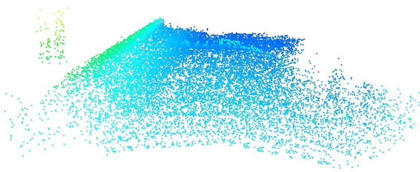


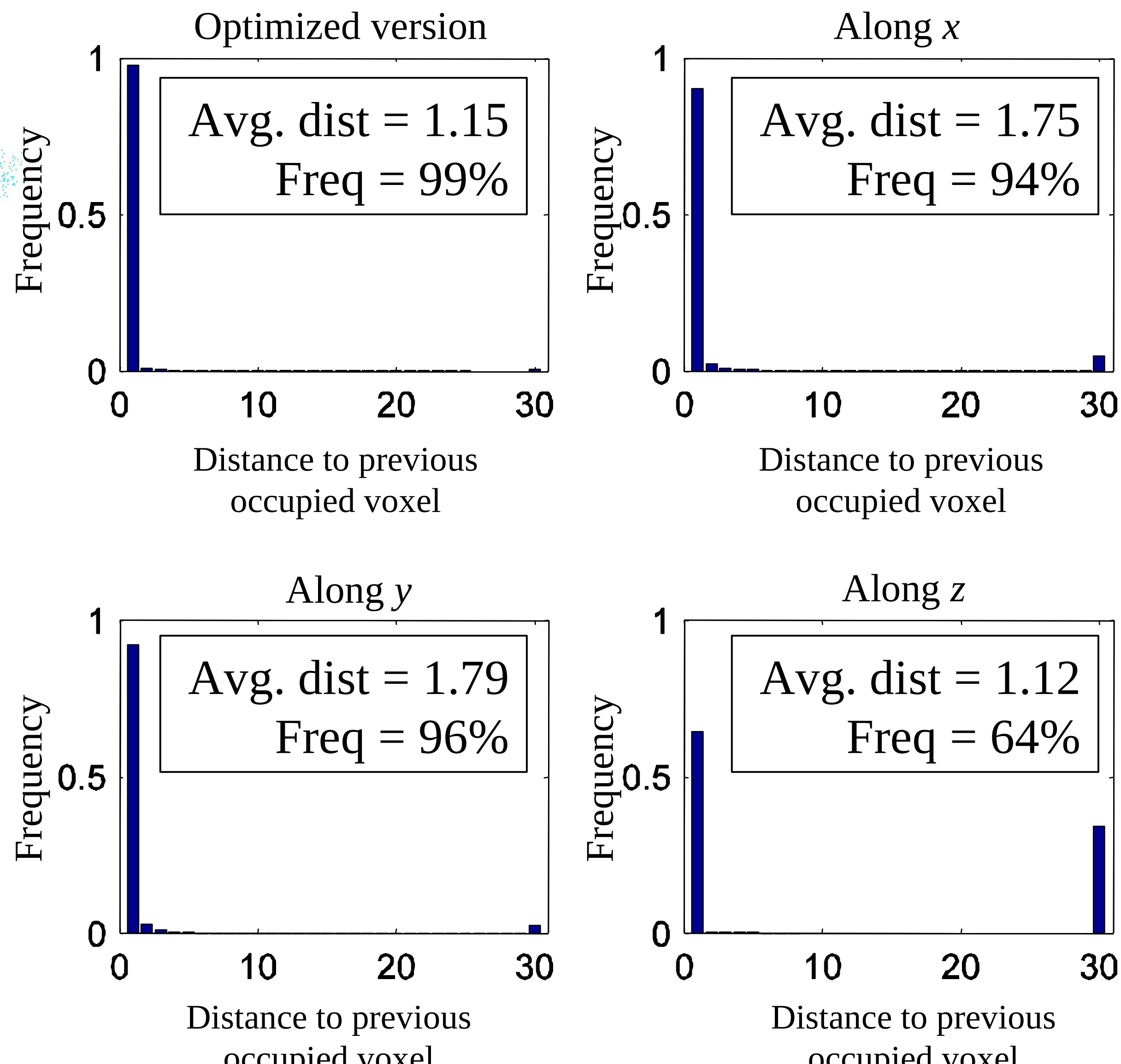
59,000 occupied voxels | 112,000 occupied voxels | 117,000 occupied voxels

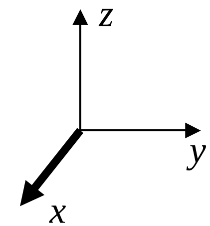
- Voxel size of 0.1m
- Experiments:
  - Influence of scanning direction
  - Speedup on different scenes
  - Influence of data density
- Data collected by the robot
- Offline data processing
- All tests performed on the same computer (valid comparison)

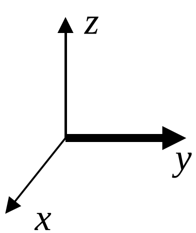
Experiments – scanning direction

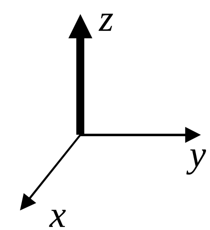
Flat ground dataset





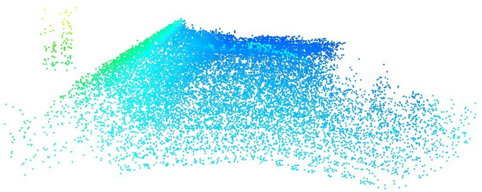




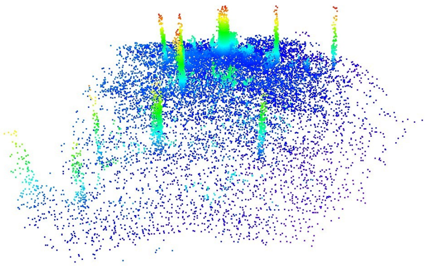


Experiments – scanning direction

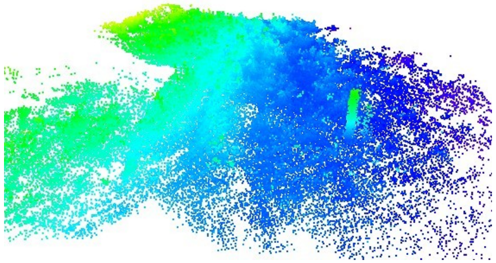
Flat ground dataset



Forest dataset

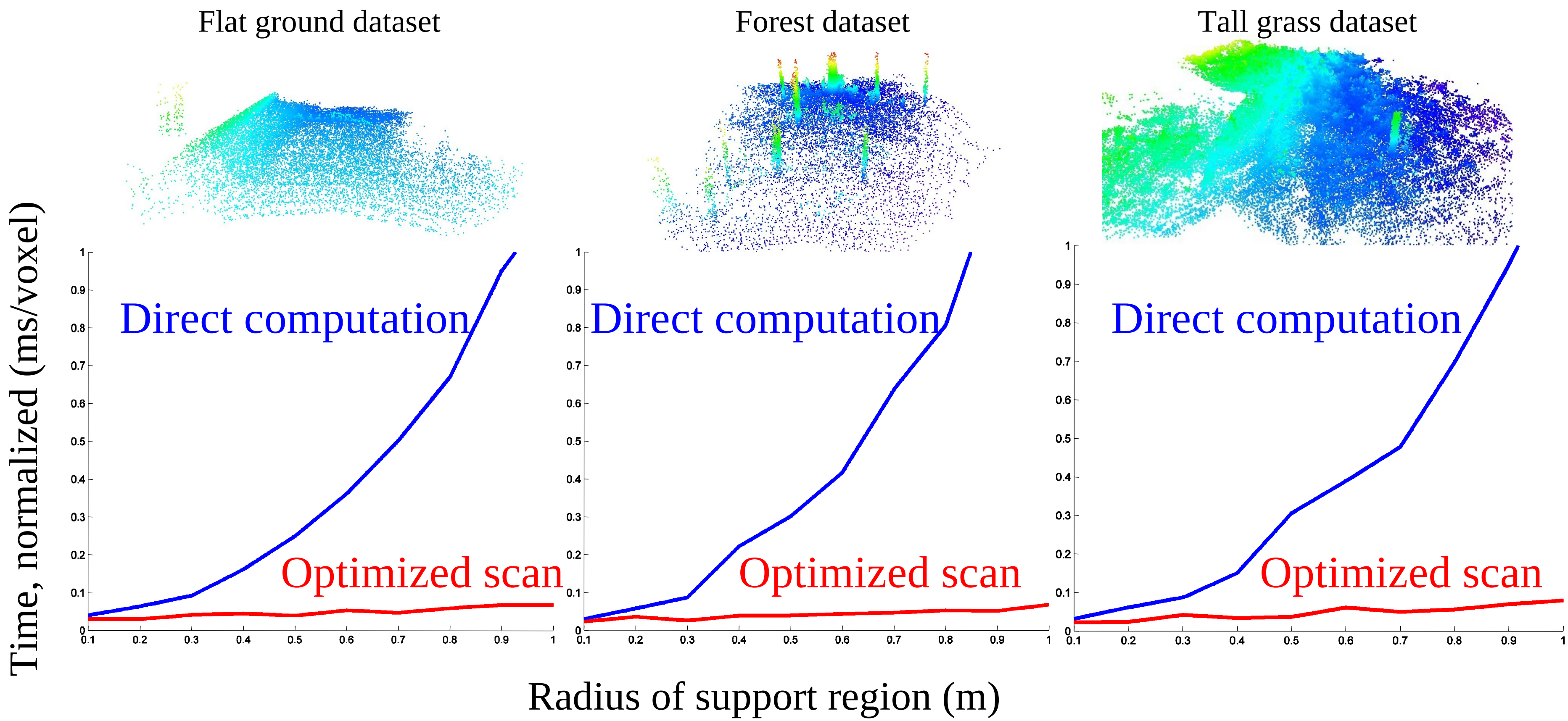


Tall grass dataset



No significant difference

Experiments - speedup



- Speedup of 4.5x at radius of 0.4m (k = 9)

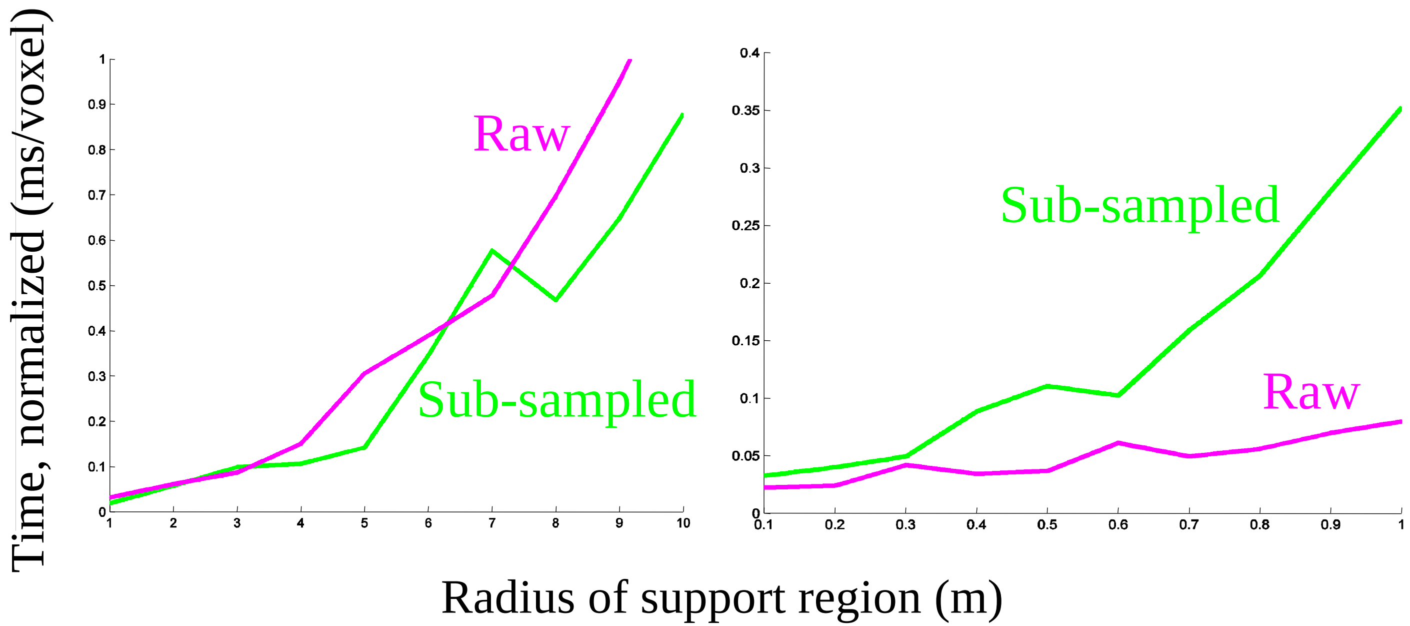
Experiments - density

Tall grass dataset

Nb of voxels
Raw: 117,000
Sub-sampled: 9,000

Old method,
Direct computation

New method,
Optimized scan



- Lower density results in lower gain

What can we predict?

v
Number of occupied voxels

k
Size of support region

$$\frac{v}{n} > \frac{1}{\left(k^3 - 2\bar{d}k^2\right)\underbrace{P[X < \tfrac{k}{2}]}}$$

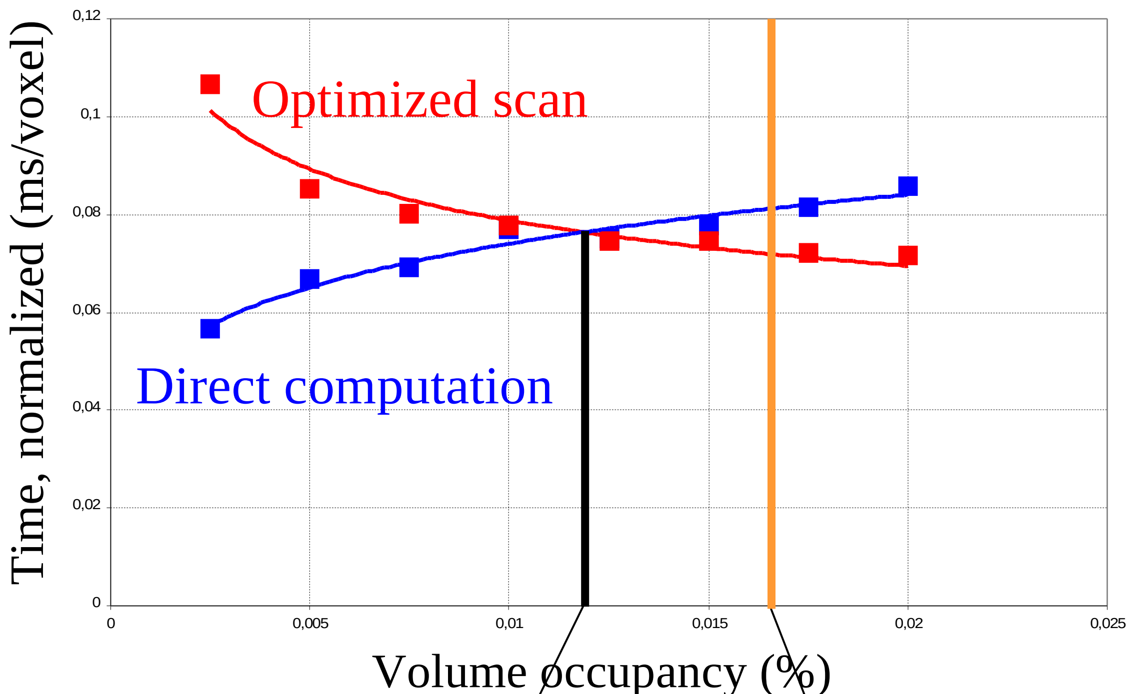
n
Total number of voxels in the volume

$\bar{d}$
Average distance between interest voxel and previous result

***P*[*X* < *k*/2]**
Probability that condition for reuse is satisfied

- Lower bound that guarantees gain over direct computation method

Experimental validation



Point where both approaches are equivalent experimentally

Lower bound provided by previous equation

Conclusion

- Summary
  - Data structure with corresponding approach to speedup full 3-D data processing
  - Analyze influence of various parameters
  - Significant speedup on different scenes
- Limitations
  - Depend on scene density
  - Trade-off: hard to evaluate a priori
    - Gain of reusing data
    - Memory and processing overhead of more complex methods

Future work

- Extension to live processing
  - Implementation under way
- Acknowledgements
  - General Dynamics Robotics Systems
  - U.S. Army Research Laboratory