Parallel Computing In Mobile Robotics For RISE

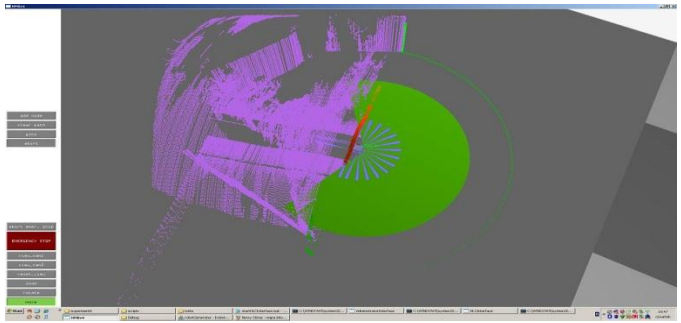
Janusz Będkowski, Ph.D.

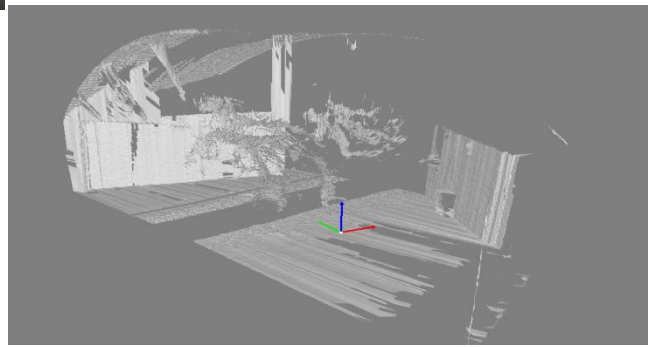
Institute of Mathematical Machines, Warsaw, Poland

GPU TECHNOLOGY CONFERENCE

GPU TECHNOLOGY CONFERENCE 2012
MAY 14-17, 2012 | SAN JOSE, CALIFORNIA

RISE (Risky Intervention and Surveillance Environment)



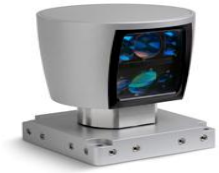


An accurate 3D model acquisition and reconstruction that can help Fire Fighter services during risky intervention.

Sensors for RISE

3DLSN unit 361.221 data points

3DLSK-2 unit 1.036.800 data points



VELODYNE

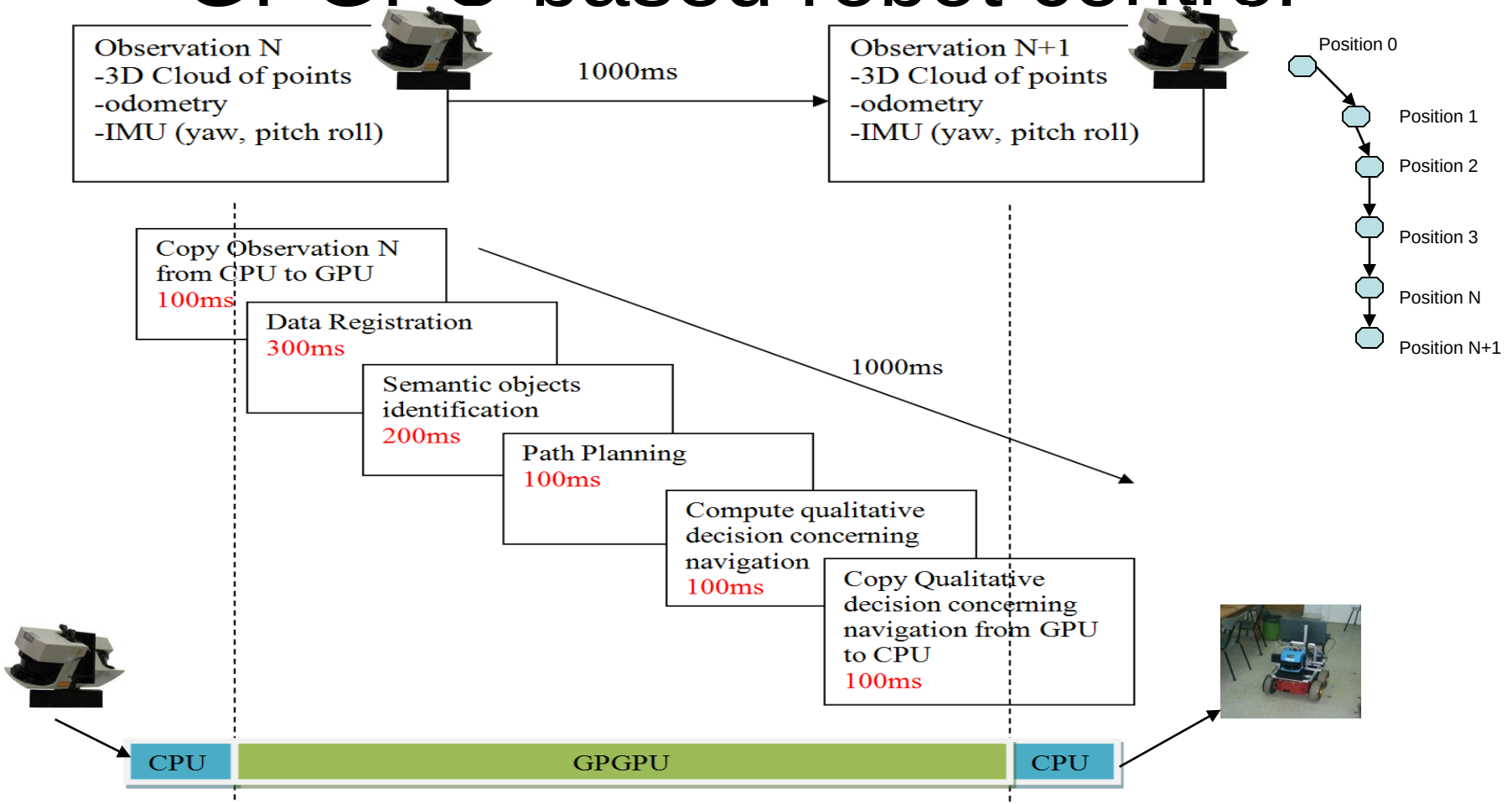
HDL-32E 700.000 data points

HDL-64E 1.300.000 data points

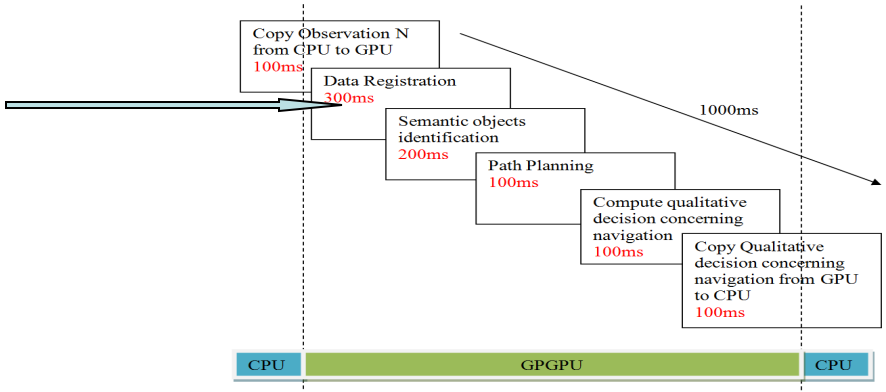
KINECT 307.200 data points

Mobile robots during CELROB2011

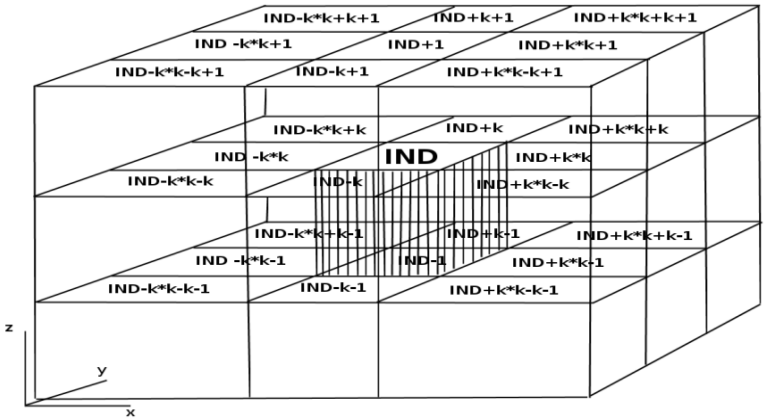
GPGPU based robot control

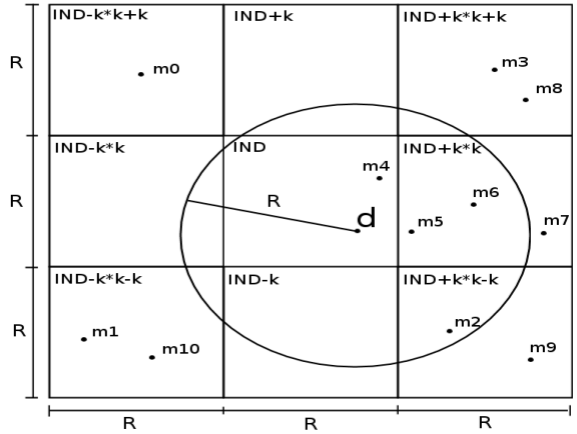


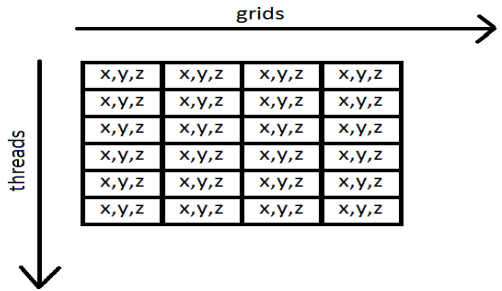
Data registration



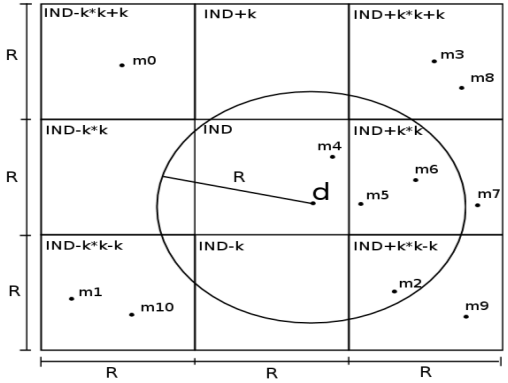
The concept of regular grid decomposition for NNS(1)

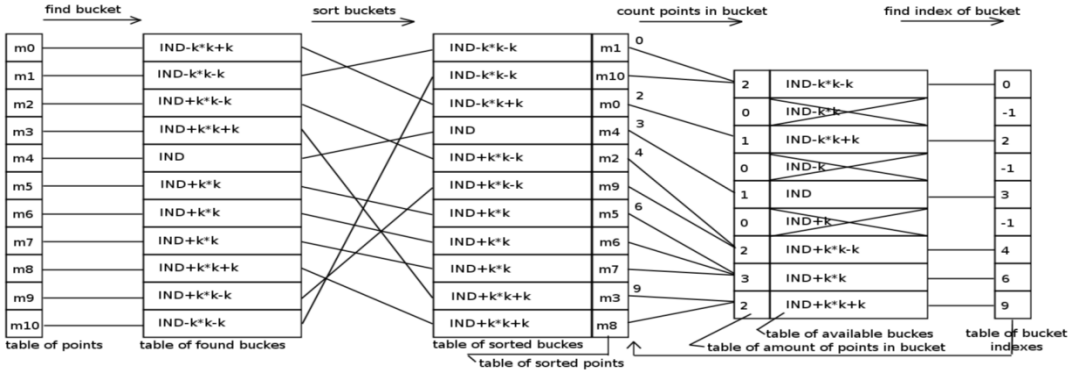


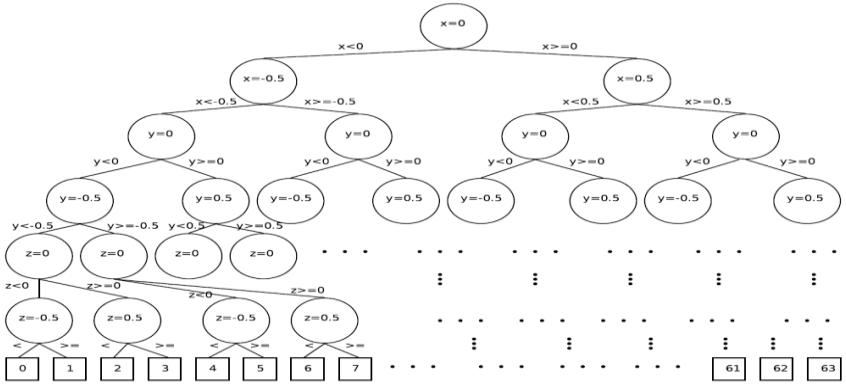




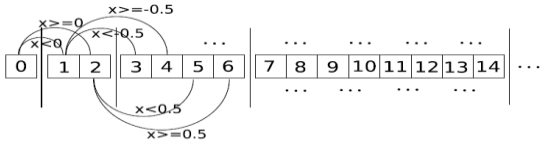
The core concept
regular grid decomposition for NNS(2)







Tree structure used for XYZ space decomposition into 64 buckets. From root to the leaf/bucket chosen left or right brunch depends on the current separation line.

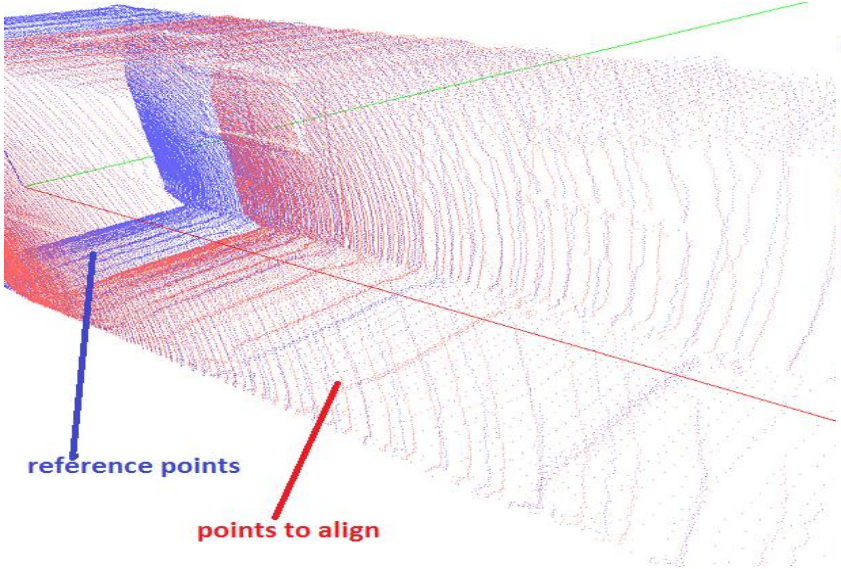


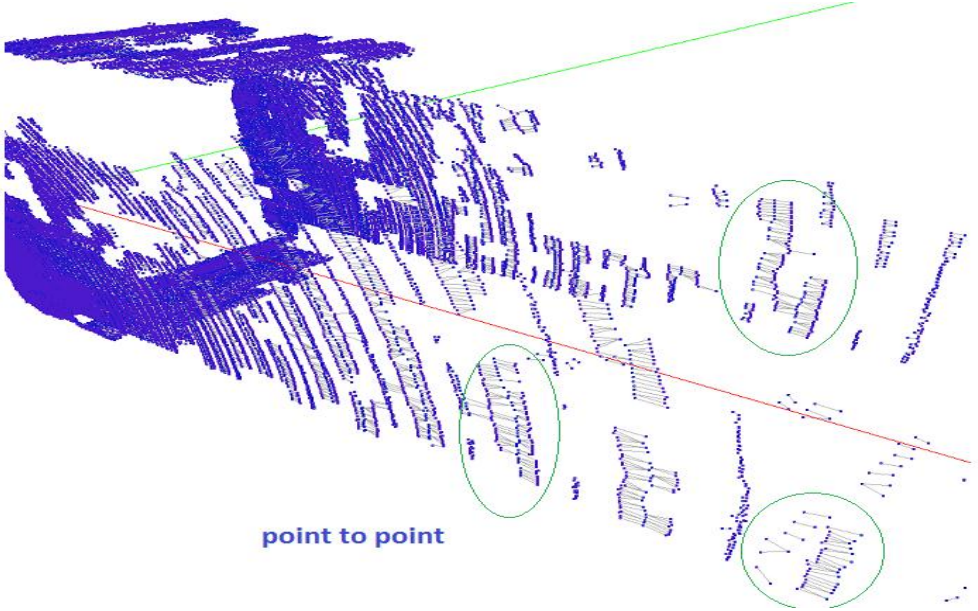
The scheme of bucket indexing procedure.

Iterative Closest Point

The alignment of two data sets for point to point ICP is solved by minimization of the following cost function:

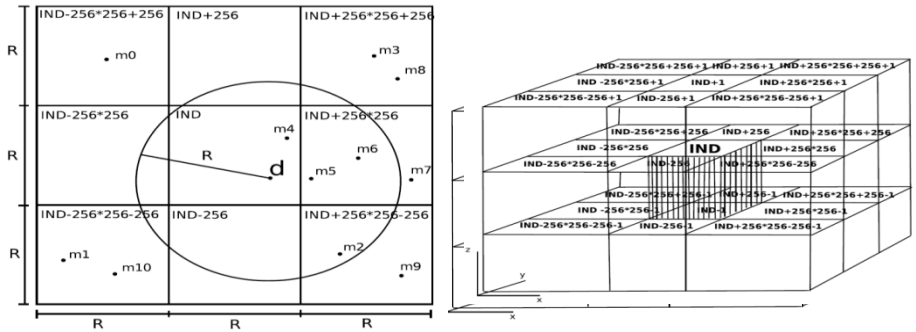
$$E(\mathbf{R},\mathbf{t}) = \sum_{i=1}^{N_m}\sum_{j=1}^{N_d} w_{ij}\left\|\mathbf{m}_i - \left(\mathbf{R}\mathbf{d}_j + \mathbf{t}\right)\right\|^2$$





Algorithm 1 Classic ICP

INPUT: Two point clouds A = $\{a_i\}$, B= $\{b_i\}$, an initial transformation T_0
OUTPUT: The correct transformation T, which aligns A and B
$T \leftarrow T_0$
for $iter \leftarrow 0$ to $maxIterations$ **do**
  for $i \leftarrow 0$ to N **do**
    $m_i \leftarrow \text{FindClosestPointInA}(T \cdot b_i)$
    if $\|m_i - T \cdot b_i\| \leq d_{max}$ **then**
      $w_i \leftarrow 1$
    else
      $w_i \leftarrow 0$
    end if
  end for
  $T \leftarrow \underset{T}{argmin} \{\sum_i w_i \|T \cdot b_i - m_i\|^2\}$
end for



Algorithm 2 ICP - parallel computing approach

INPUT: Two point clouds M = $\{m_i\}$, D= $\{d_i\}$, an initial transformation T_0
OUTPUT: The correct transformation T, which aligns M and D
$M_{device} \leftarrow M$
$D_{device} \leftarrow D$
$T_{device} \leftarrow T_0$
for $iter \leftarrow 0$ to $maxIterations$ **do**
  for $i \leftarrow 0$ to N {in parallel} **do**
    $m_i \leftarrow \text{FindClosestPointInM}\ (T_{device} \cdot d_i)$ {using regular grid decomposition}
    if $foundClosestPointInNeighboringBuckets$ **then**
      $w_i \leftarrow 1$
    else
      $w_i \leftarrow 0$
    end if
  end for
  $T_{device} \leftarrow \underset{T_{device}}{argmin} \{\sum_i w_i \|T \cdot d_i - m_i\|^2\}$ {calculation T←R,t with SVD}
end for
$M \leftarrow M_{device}$
$D \leftarrow D_{device}$
$T \leftarrow T_{device}$

1

2

1. „Smart indexing” using regular grid decomposition
2. Calculation of correlation matrix using all-prefix-sum reduction
3. Minimization of T using SVD performed on GPGPU

To register two cloud of point we have to minimize following cost function: (Nüchter A. 2007)(Arun K. S., Huang T. S. et al. 1987)):

$$E(\boldsymbol{R},\boldsymbol{t}) = \sum_{i=1}^{N_m}\sum_{j=1}^{N_d} w_{ij}\left\|\boldsymbol{m}_i - (\boldsymbol{R}\boldsymbol{d}_j + \boldsymbol{t})\right\|^2 \quad (1)$$

We are rewriting (1):

$$E(\mathbf{R},\mathbf{t}) \propto \frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{m}_i - (\mathbf{R}\mathbf{d}_\mathbf{i} + \mathbf{t})\right\|^2, \quad (2) \qquad N = \sum_{i=1}^{N_m}\sum_{j=1}^{N_d} w_{ij} \quad (3)$$

After computing centroids:

$$\mathbf{c_m} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{m_i} \qquad \mathbf{c_d} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{d_i} \quad (4)$$

We obtain modified data sets: $\mathbf{M'} = \{\mathbf{m_i'} = \mathbf{m_i} - \mathbf{c_m}\}_{1,\ldots,N}$ $\mathbf{D'} = \{\mathbf{d_i'} = \mathbf{d_i} - \mathbf{c_d}\}_{1,\ldots,N}$ (5)

After applying equations 4, 5 to the mean square error function E(**R**, **t**), equation 2 takes the following form:

$$E(\mathbf{R},\mathbf{t}) \propto \frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{m}_i - \mathbf{R}\mathbf{d}_\mathbf{i} - (\mathbf{t} - \mathbf{c_m} + \mathbf{R}\mathbf{c_d})\right\|^2 \quad (6)$$

Assuming that: $\mathbf{t} - \mathbf{c_m} + \mathbf{R}\mathbf{c_d} = \tilde{t}$ equation 6 takes following form:

$$E(\mathbf{R},\mathbf{t}) \propto \frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{m}_i - \mathbf{R}\mathbf{d}_\mathbf{i}'\right\|^2 - \frac{2}{N}\tilde{t}\sum_{i=1}^{N}(\mathbf{m_i'} - \mathbf{R}\mathbf{d_i'}) + \frac{1}{N}\sum_{i=1}^{N}\|\tilde{t}\|^2 \quad (7)$$

Arun K. S., Huang T. S., et al. (1987). "Least-Squares Fitting of Two 3-D Point Sets." IEEE Transactions on Pattern Analysis and Machine Intelligence **9**(5): 698-700

Nüchter A., Lingemann K., et al. (2007). Cached k-d tree search for ICP algorithms. Sixth International Conference on 3-D Digital Imaging and Modeling, Washington, DC, USA

To minimize $$E(\mathbf{R},\mathbf{t}) \propto \frac{1}{N}\sum_{i=1}^{N}\left\|\mathbf{m}_i - \mathbf{R}\mathbf{d}_i'\right\|^2 - \frac{2}{N}\tilde{t}\sum_{i=1}^{N}(\mathbf{m}_i' - \mathbf{R}\mathbf{d}_i') + \frac{1}{N}\sum_{i=1}^{N}\|\tilde{t}\|^2$$

the algorithm has to minimize the term: $\sum_{i=1}^{N}\left\|\mathbf{m}_i' - \mathbf{R}\mathbf{d}_i'\right\|^2$ with the assumption: $\tilde{t} = 0$

The optimal rotation is calculated by **R=VU^T**, where matrices **V** and **U** are derived from the singular value decomposition of a correlation matrix **C** = **USV^T** given by:

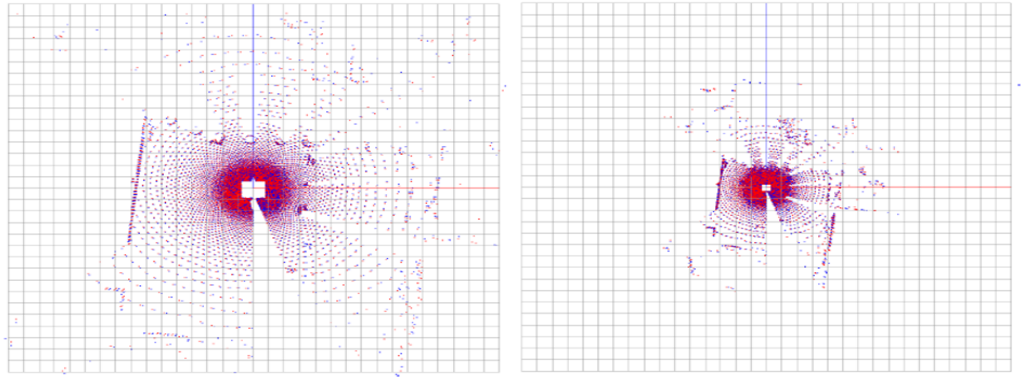
$$\mathbf{C} = \sum_{i=1}^{N}\mathbf{m}_i'^{T}\,\mathbf{d}_i' = \begin{bmatrix} c_{xx} & c_{xy} & c_{xz} \\ c_{yx} & c_{yy} & c_{yz} \\ c_{zx} & c_{zy} & c_{zz} \end{bmatrix}, \qquad c_{xx} = \sum_{i=1}^{N} m_{ix}' d_{ix}', c_{xy} = \sum_{i=1}^{N} m_{ix}' d_{ix}', \ldots, c_{zz} = \sum_{i=1}^{N} m_{iz}' d_{iz}'$$

Calcultion of R using SVD is based on all-prefix-sum reduction.

The optimal translation **t** is derived from equation $\mathbf{t} = \mathbf{c}_\mathbf{m} - \mathbf{R}\mathbf{c}_\mathbf{d}$

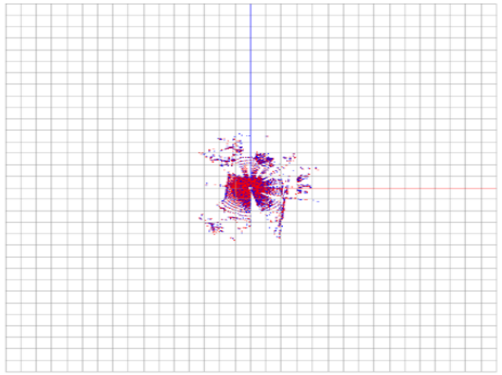
GPGPU ICP parameters' tuning

- div (data set normalization)
- amount of buckets for each dimension
- d_{max}
- amount of ICP iterations
- thresholds for amount of processed points in buckets

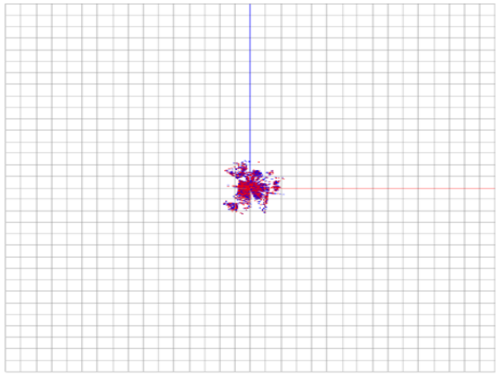


(a) nr of buckets 32, div = 2

(b) nr of buckets 32, div = 5

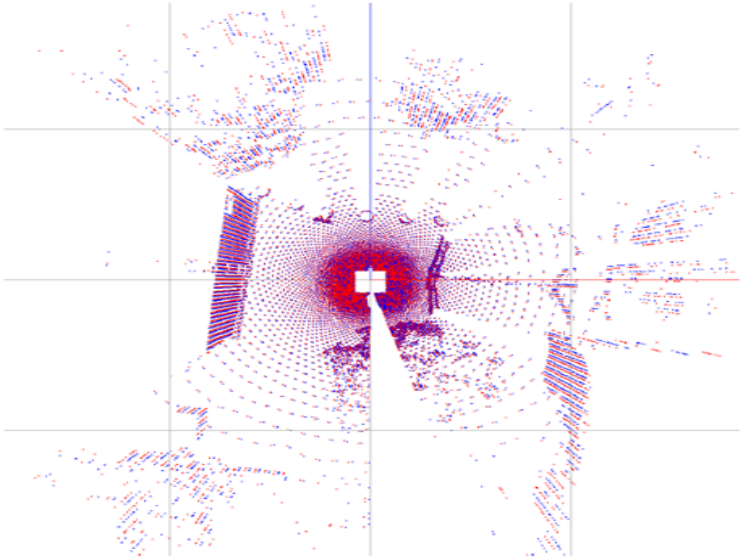


(c) nr of buckets 32, div = 10

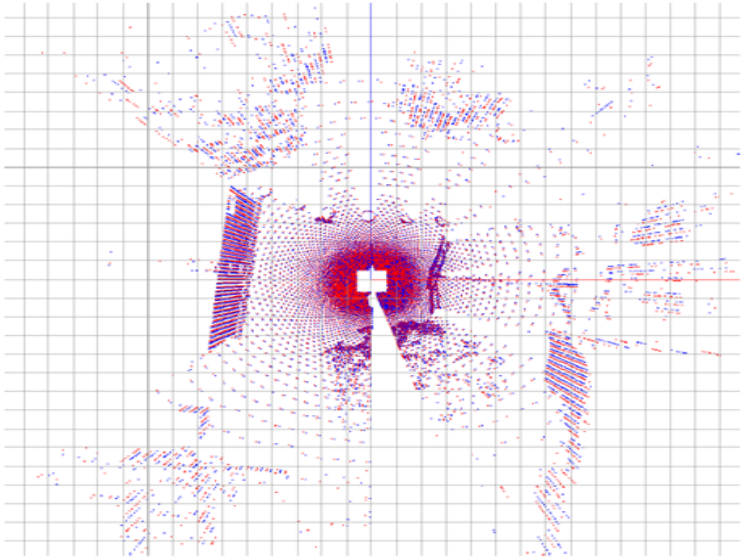


(d) nr of buckets 32, div = 20

An example of parameter div tuning - data set normalization.
Red points - reference points, Blue points - points to be aligned



(a) 32 buckets, div = 20.



(b) 256 buckets, div = 20.

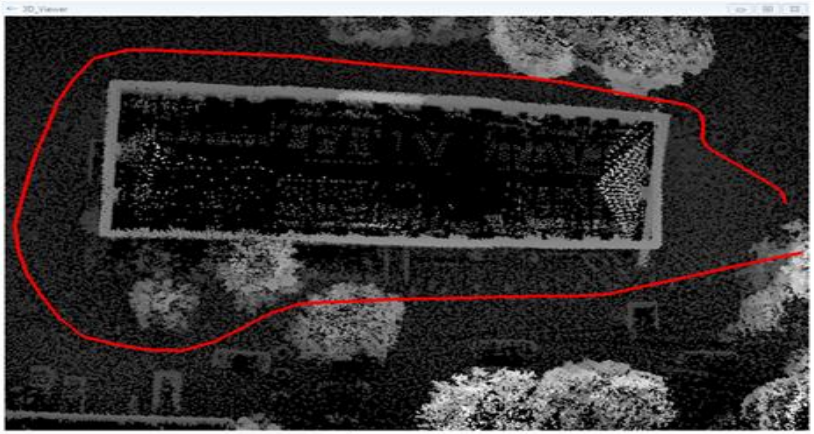
An example of different amount of buckets.

a) 32 buckets for each dimension,

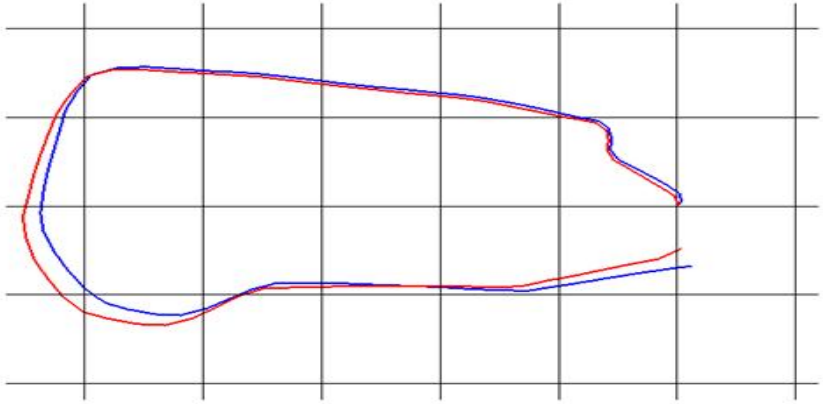
b) 256 buckets for each dimension.

Red points - reference points, Blue points – points to be aligned

An experiment based on SoA data set



a)



b)

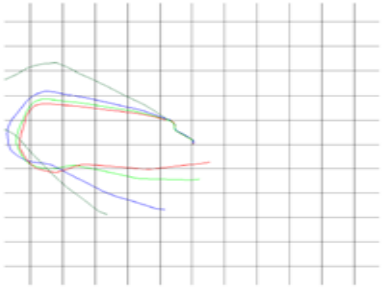
a) Visualization of the 3D Viewer application with data set Hannover1.
Red line - robot trajectory obtained by ICP available in 3DTK package (Nüchter A. 2011).

b) Red color – trajectory obtained using 3DTK,
blue color – trajectory obtained using our “naive” implementation of ICP (Besl and McKay 1992)
with SVD for least square fitting (Arun K. S., Huang T. S. et al. 1987).
We used default parameters: dmax = 7.5cm, number of ICP iterations = 30, “metascan” for matching
the current scan against a meta scan of all previous scans. Our “naive” implementation of ICP is used for the generation of reference trajectories for parameters’ tuning analyses.

Nüchter A. (2011). "3DTK." from http://slam6d.sourceforge.net.

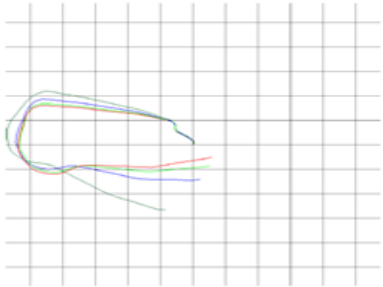
Besl, P. J. and H. D. McKay (1992). "A method for registration of 3-D shapes”
IEEE Transactions on Pattern Analysis and Machine Intelligence(2): 239 - 256

Arun K. S., Huang T. S., et al. (1987). "Least-Squares Fitting of Two 3-D Point Sets."
IEEE Transactions on Pattern Analysis and Machine Intelligence **9**(5): 698-700.



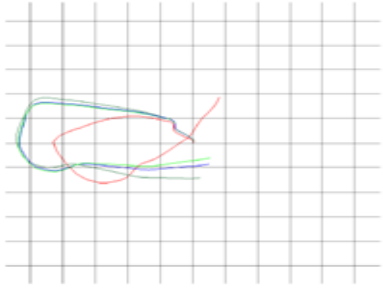
(a) robot trajectories

(b) amount of buckets - 32 (d_{max} = 2/32), div=2,5,10,20

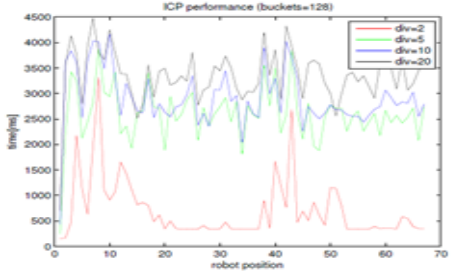


(c) robot trajectories

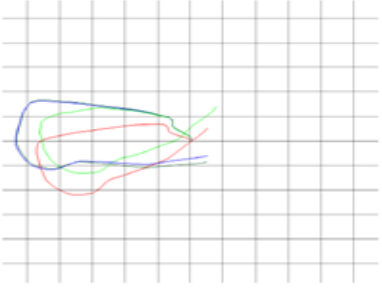
(d) amount of buckets - 64 (d_{max} = 2/64), div=2,5,10,20



(e) robot trajectories



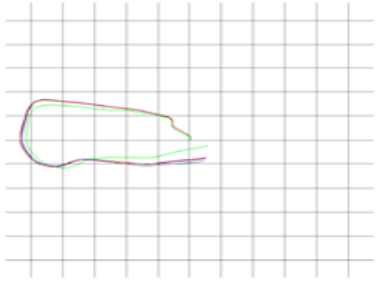
(f) amount of buckets - 128 (d_{max} = 2/128), div=2,5,10,20



(g) robot trajectories

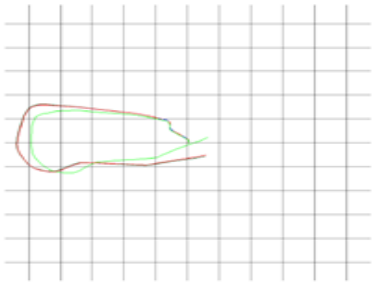
(h) amount of buckets - 256 (d_{max} = 2/256), div=2,5,10,20

Reference data obtained using the State of the Art ICP method.
16 different robot trajectories (4 values of div parameter: 2,5,10,20 for 4 different amounts of buckets: 32, 64, 128, 256) obtained by the GPUICP method without additional parameter tuning.

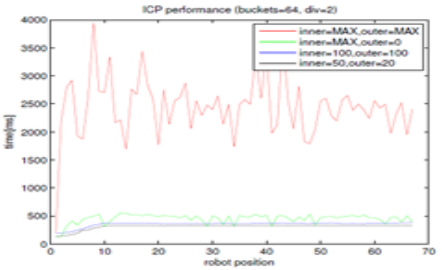


(a) robot trajectories

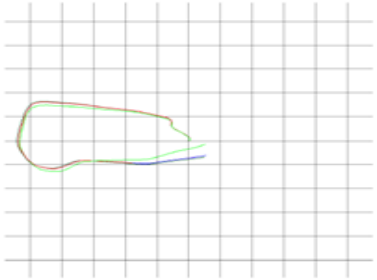
(b) amount of buckets - 32 ($d_{max} = 2/32$), div $= 2$



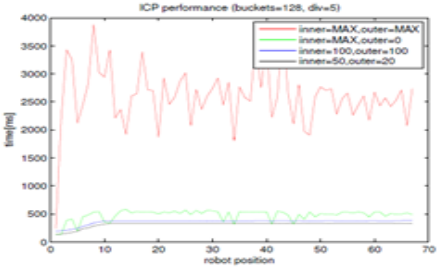
(c) robot trajectories



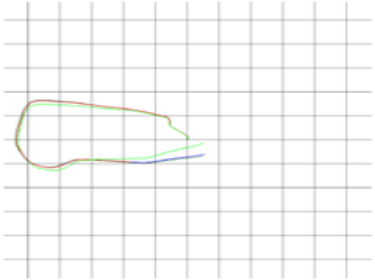
(d) amount of buckets - 64 ($d_{max} = 2/64$), div $= 2$



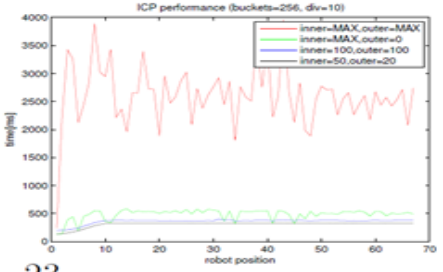
(e) robot trajectories



(f) amount of buckets - 128 ($d_{max} = 2/128$), div $= 5$

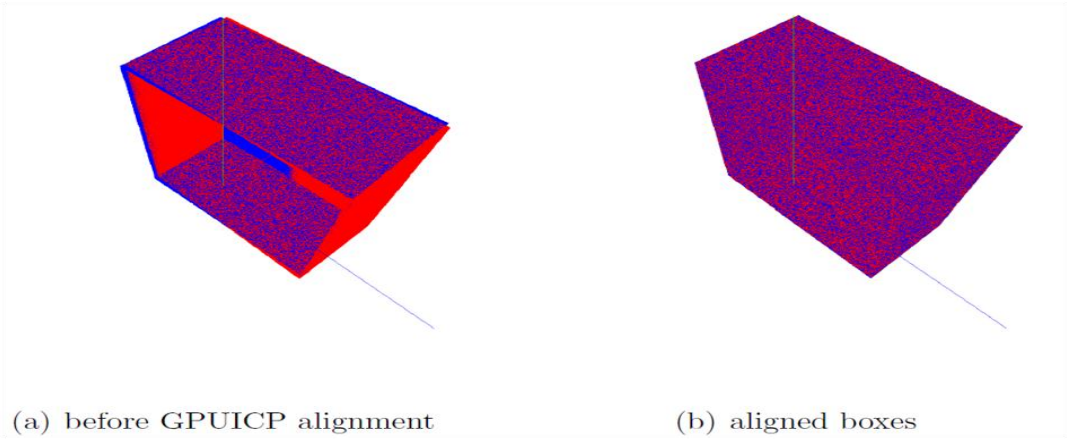


(g) robot trajectories



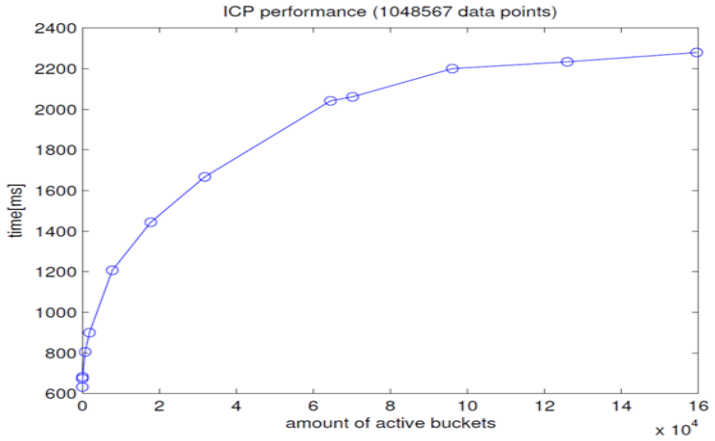
(h) amount of buckets - 256 ($d_{max} = 2/256$), div$=$ 10

Four different sets of trajectories. Each set is related with one reference trajectory chosen from previous figure (red color).
The goal of this experiment is to demonstrate that our GPUICP is keeping the accuracy compared to SoA ICP implementation, even after substantial parameter tuning. The worst case is demonstrated as a green trajectory
(law accuracy, high variance of computation time), in this case the NN procedure is limited to finding corresponding points only inside the central bucket.

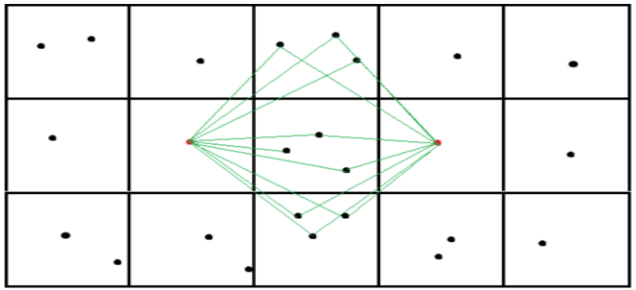


(a) before GPUICP alignment (b) aligned boxes

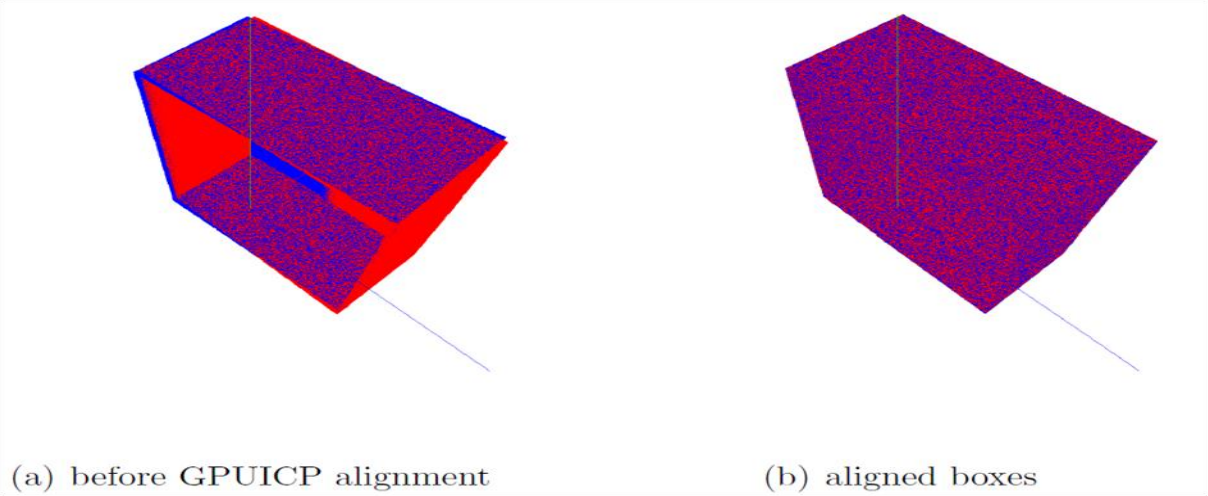
Artificial data set. a) boxes before GPUICP alignment, b) aligned boxes.



The performance in function of the amount of active buckets (alignment of 2 data sets composed of 1.048.576 points, the parameter div was changed to obtain different amounts of active buckets, inner=5, outer=1, amount of buckets for each dimension 256).

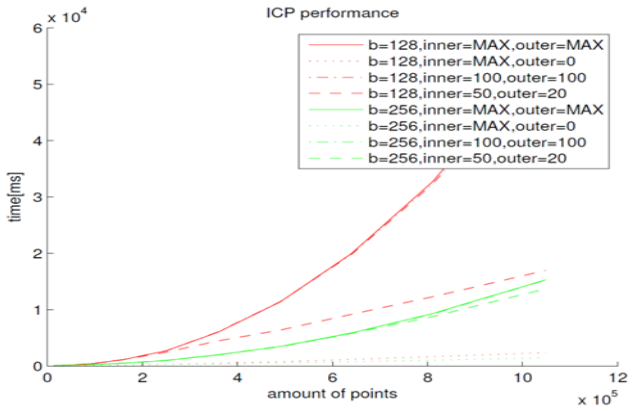


The problem of reading the same memory address in GPU
by many threads at the same time during the NNS procedure.



(a) before GPUICP alignment

(b) aligned boxes

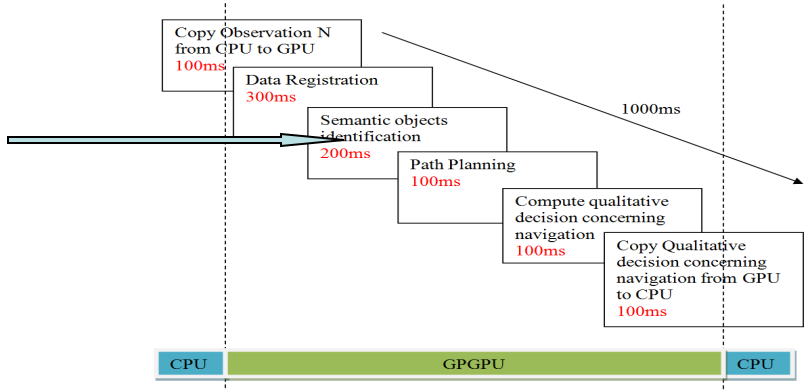


The performance measure for the datasets with different amounts of data. GPUICP performance with different improvements is shown (b-amount of buckets for each dimension).

Conclusions concerning GPGPU point to point ICP

- The GPUICP time computation is strongly dependent on the density of points in the buckets, (more points in the buckets lead to a longer computation time). We can solve this problem by setting thresholds.
- The variance of the computing time is larger in the case of higher diversity of density of data points in the buckets.
- The corrected trajectory is not accurate for large/small values of d_{max}.
- A very short execution time can lead to low accuracy. This is caused by the decreased amount of the data points per bucket.

Semantic objects identification



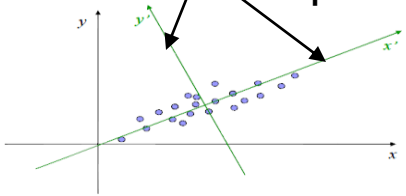
Surface normal estimation

The normal vector is computed using Principal Component Analysis (PCA) of following covariance matrix **C** = **USV**^{T} using SVD – Singular Value Decomposition

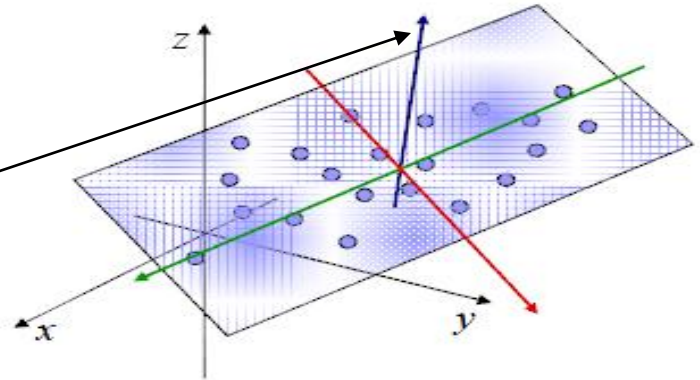
$$\mathcal{C} = \frac{1}{k}\sum_{i=1}^{k} \cdot (\boldsymbol{p}_i - \overline{\boldsymbol{p}}) \cdot (\boldsymbol{p}_i - \overline{\boldsymbol{p}})^T$$

k-number of neighbors
$\overline{\boldsymbol{p}}$ 3D centroind
$\boldsymbol{p}_i$ data points

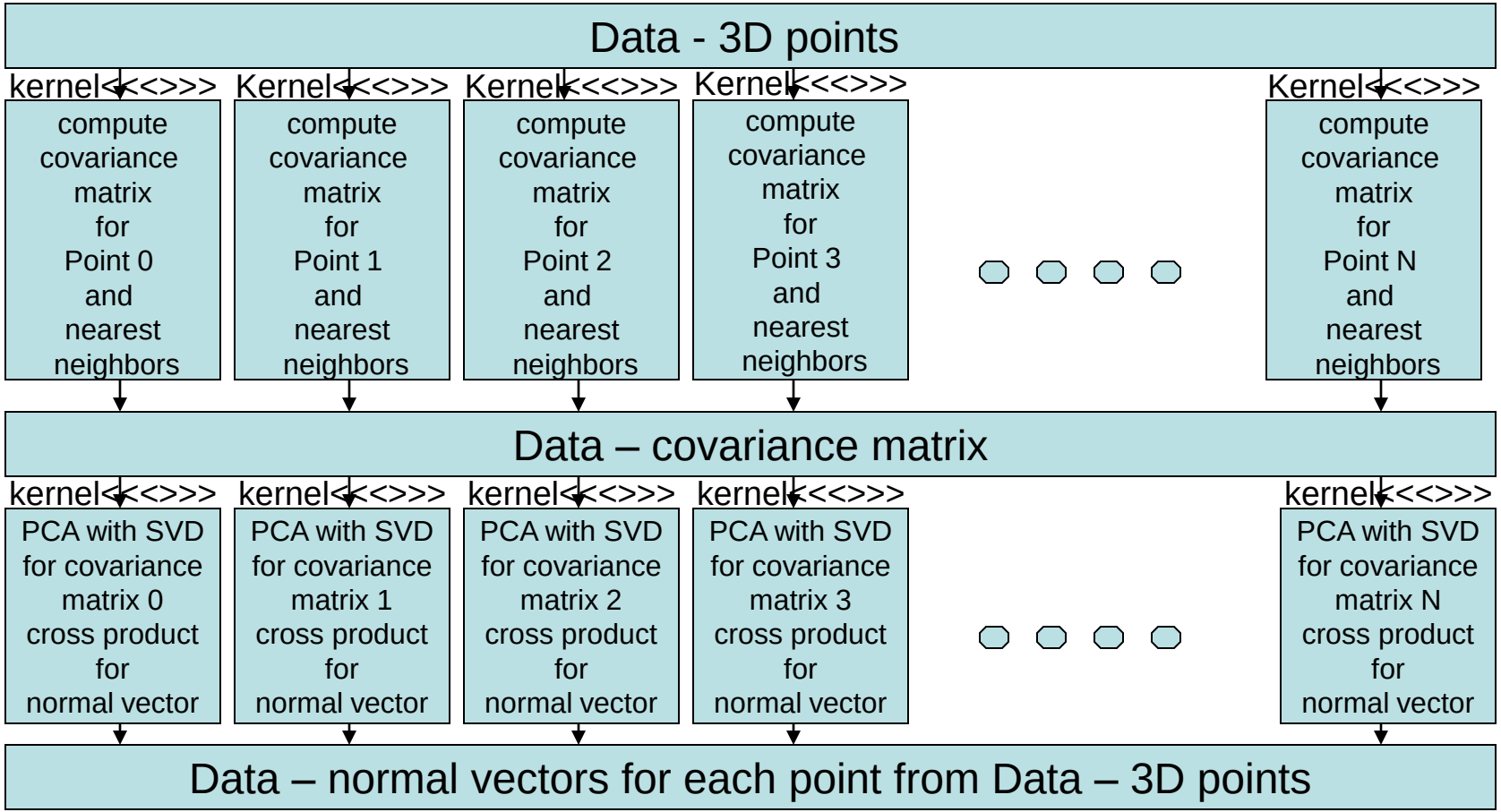
The columns of *U* (**u**$_1$, **u**$_2$, **u**$_3$) are called the left singular vectors.
They are the axes of the ellipsoid.



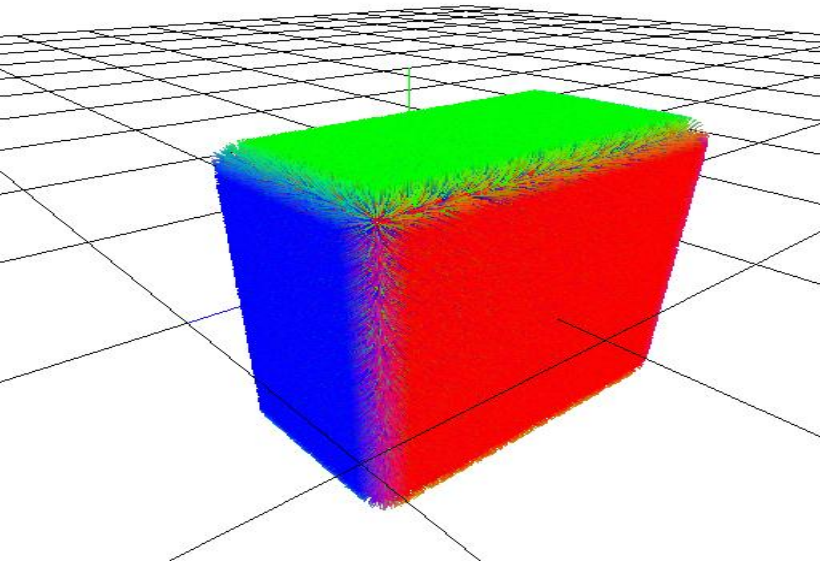
Cross product(**U**$_1$,**U**$_2$)



Algorithm of parallel surface normals' estimation

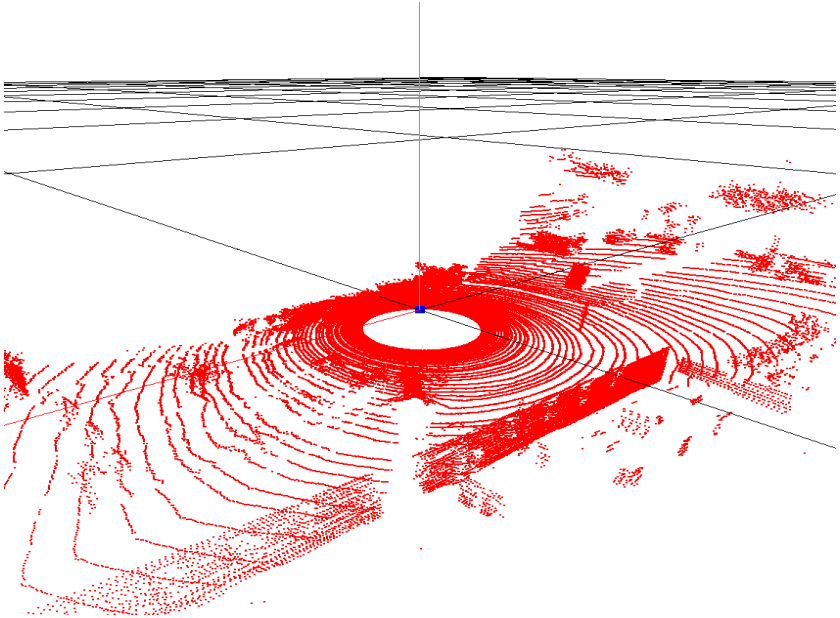


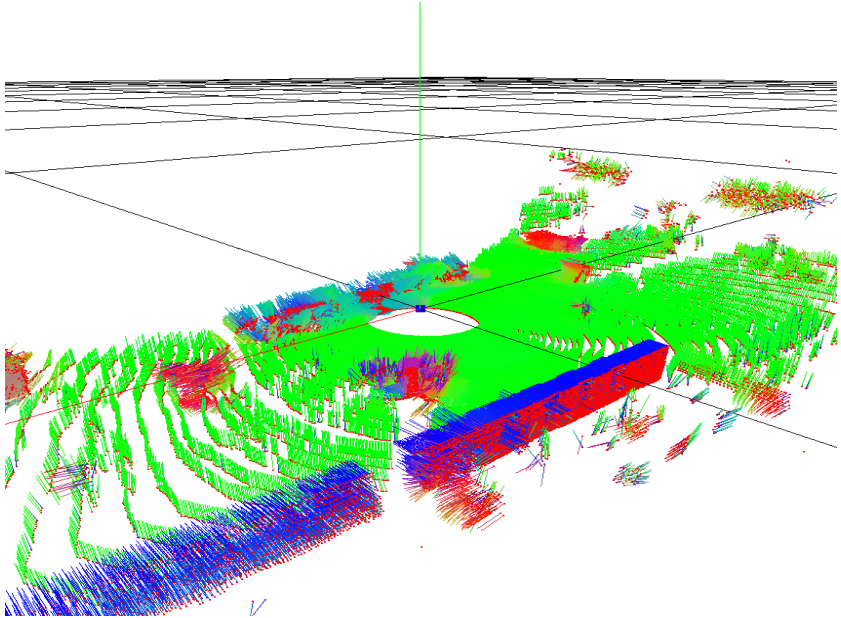
Performance of estimating surface normal



(inner100 outer 100)	32	64	128
200x200	1,2/19,24	1,5/9,5	1,5/7,0
300x300	1,4/121,0	1,5/40,3	1,5/19,3
400x400	1,95/367,6	2,1/126,6	2,1/47,6
500x500	2,3/677,6	2,5/337,5	2,4/109,8
640x480	3,0/858,7	3,1/521,5	3,9/161,6

Normal vectors for VELODYNE data (102912 3D data points)

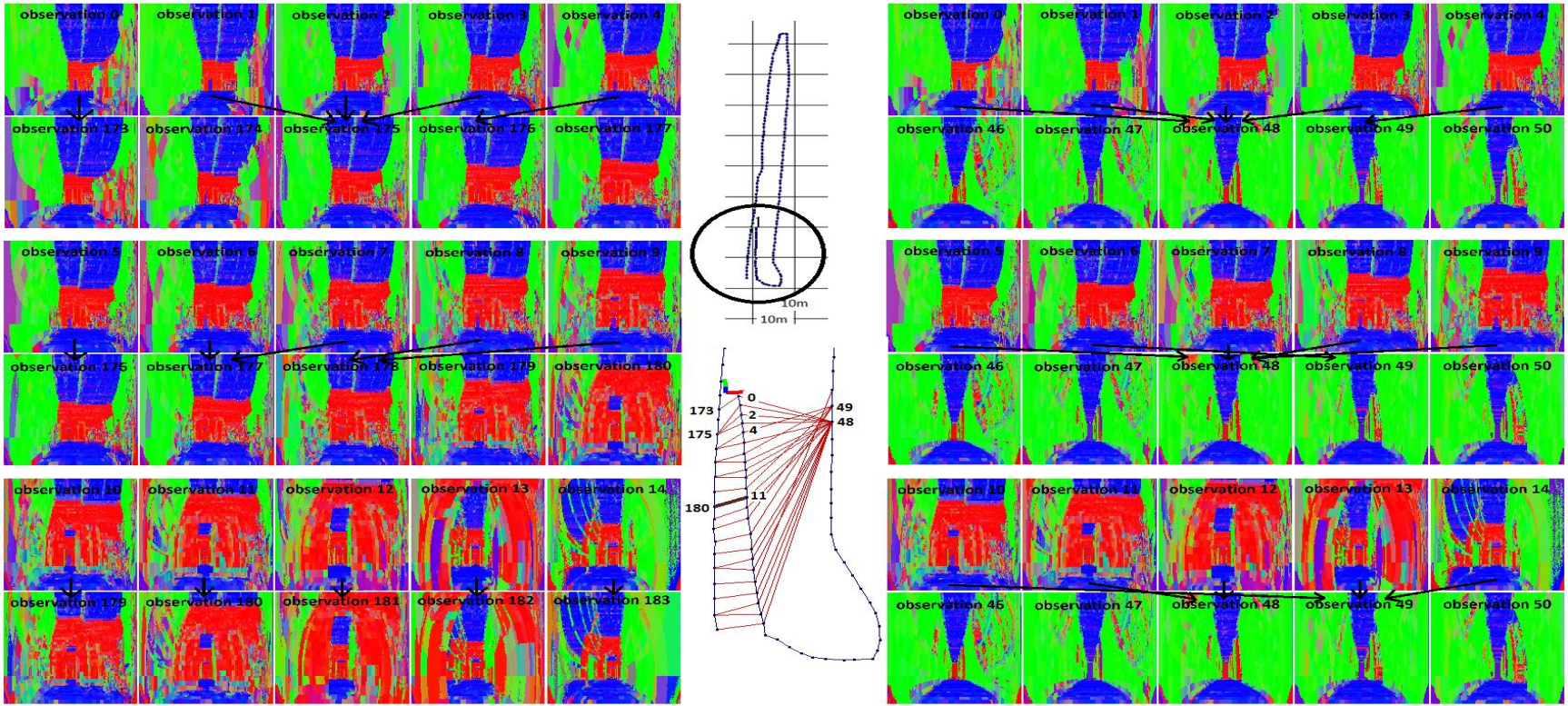




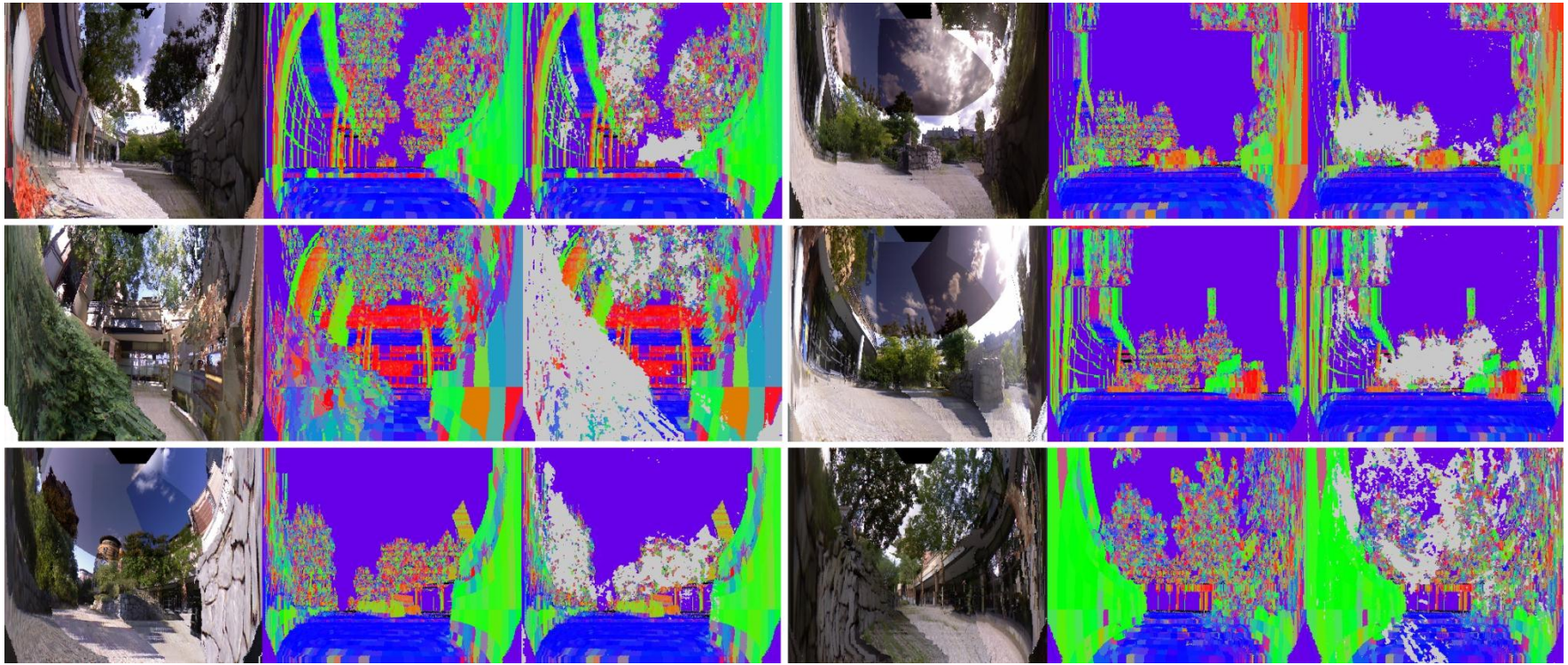
1,5ms – normal vectors computation

40,5ms – total computation time including copy from/to host/device

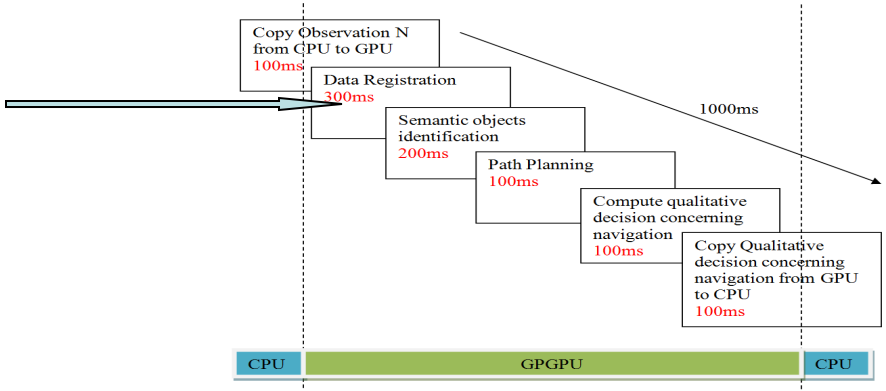
Loop closing by comparison of semantic images



Vegetation detection using images and normal vectors

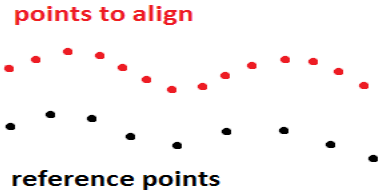


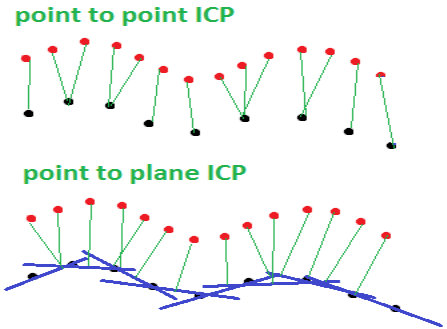
Back to data registration- point to plane ICP

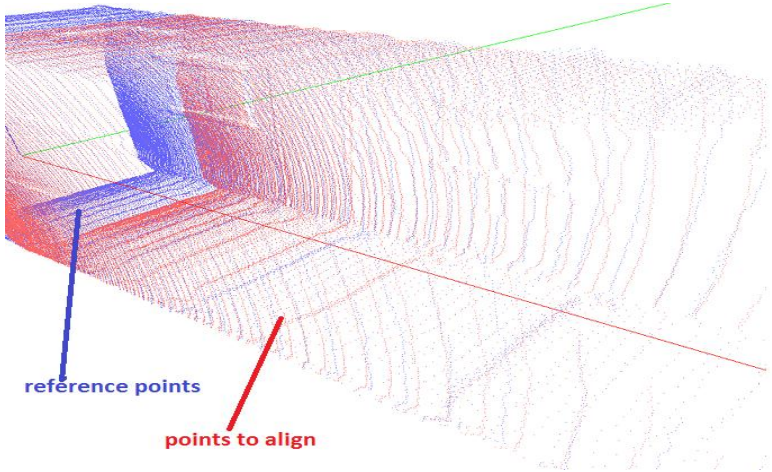


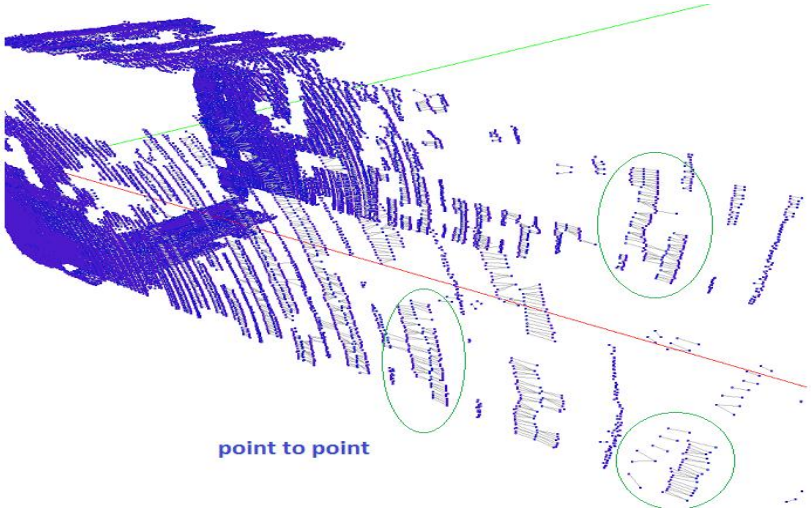
Algorithm ICP - point to plane parallel computing approach

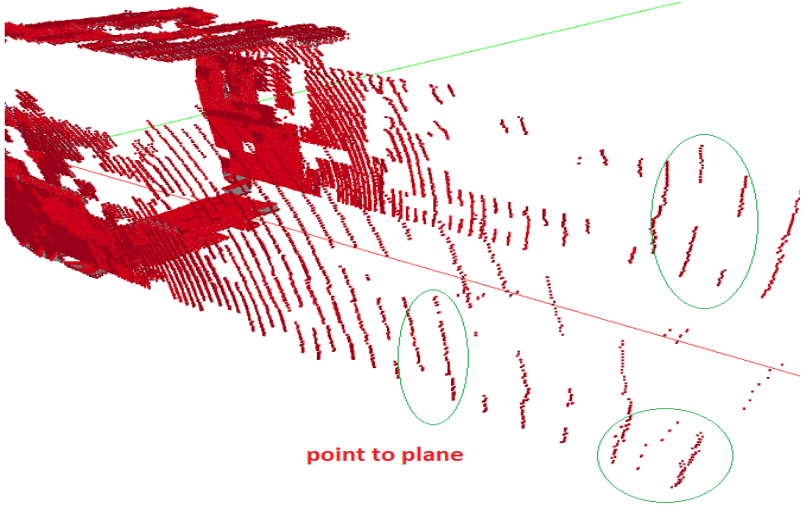
allocate the memory
copy data from the host(M_{host}, D_{host}) to the device(M_{device}, D_{device})
in parallel compute M_{planes} for M_{device} points
choose best $M_{planeofbucket}$ for each bucket
for $iter = 0$ to $max_iterations$ **do**
  in parallel find $M_{planeofbucket}$ for each data D_{device}
  in parallel compute distances between data D_{device} and its projection onto $M_{planeofbucket}$
  calculate (R, t) that minimizes equation 1
  transform D_{device} by (R, t) and put the results into $D_{deviceRt}$
  copy $D_{deviceRt}$ to D_{device}
  if D_{device} is aligned to M_{device} **then**
    break
  end if
end for
copy D_{device} to D_{host}
free memory

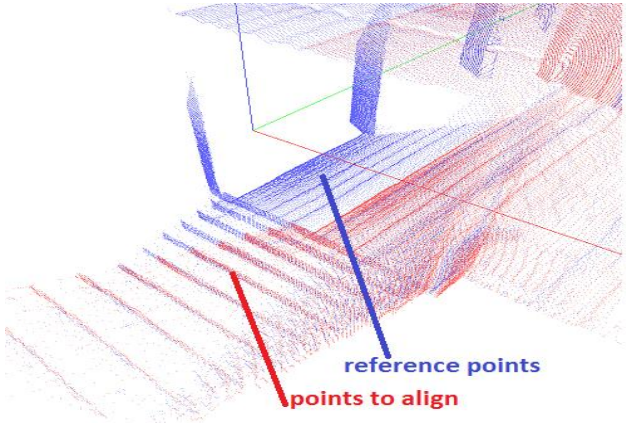


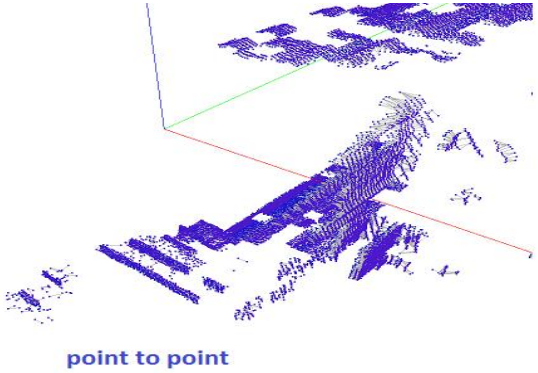


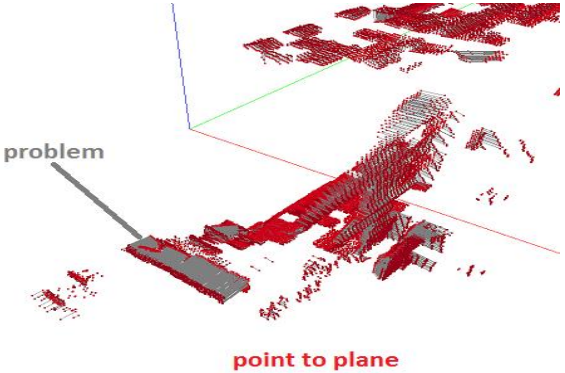

reference points
points to align

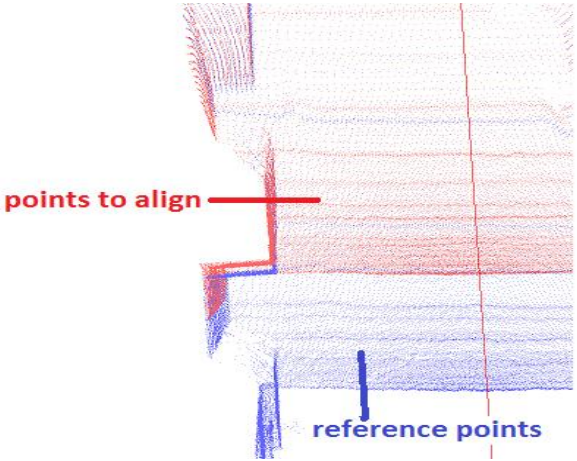

point to point

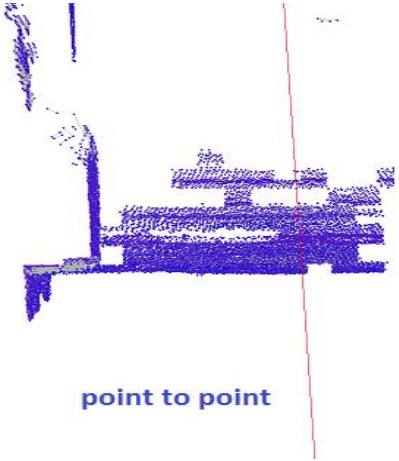

point to plane

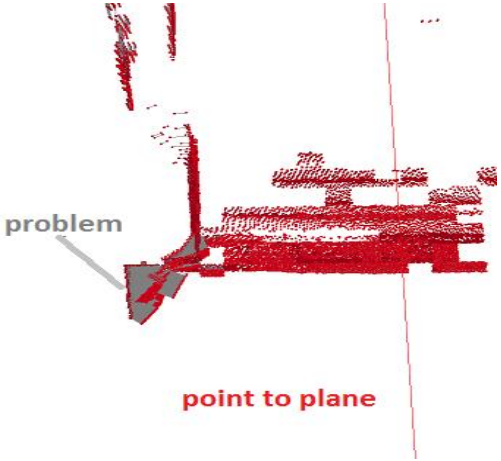

reference points
points to align


point to point

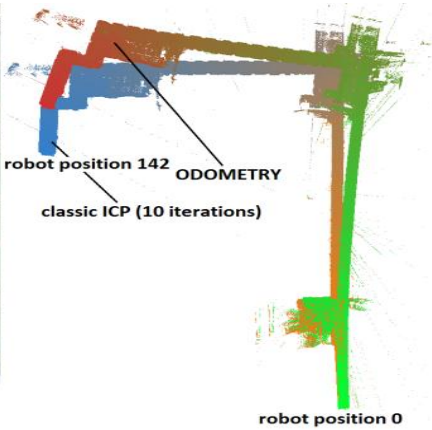

problem
point to plane

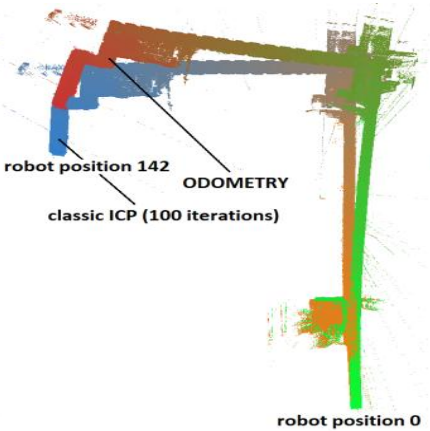

points to align
reference points

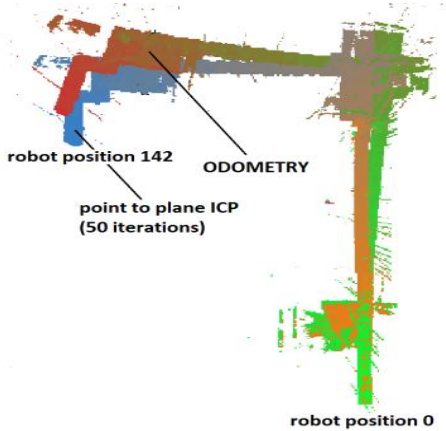

point to point


problem
point to plane

Comparison point to point ICP with point to plane ICP

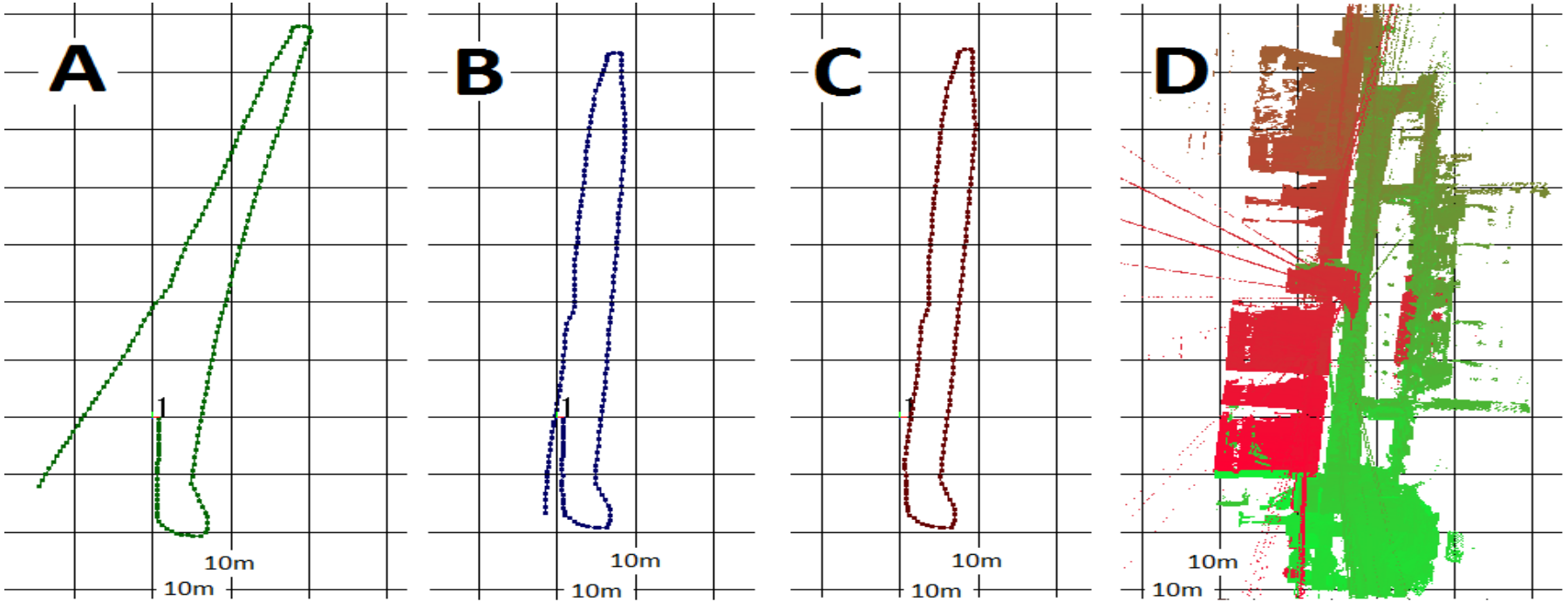




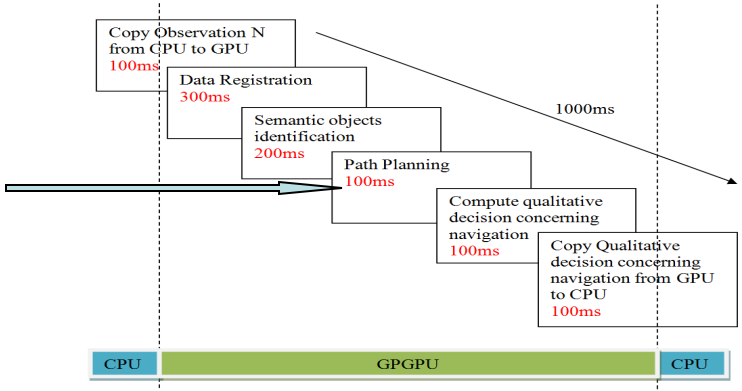


1. Point to point GPGPU ICP takes in average 300ms
2. Point to plane GPGPU ICP is four times faster
3. At this stage we can observe that point to plane method is not always more accurate!

6D SLAM
(GPGPU ICP + loop closing based on semantic images comparison)



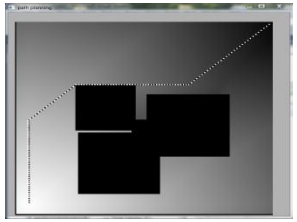
Path planning



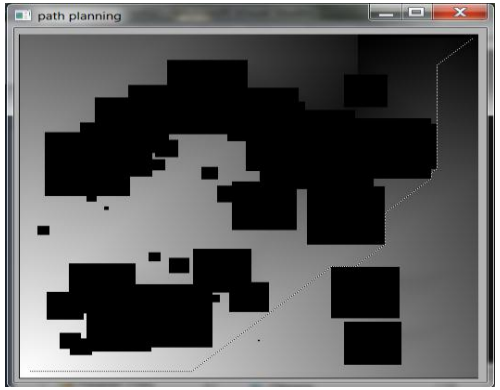
Algorithm

- Propagate in parallel the distance information as a wave that starts from the goal
- Find path using gradient method

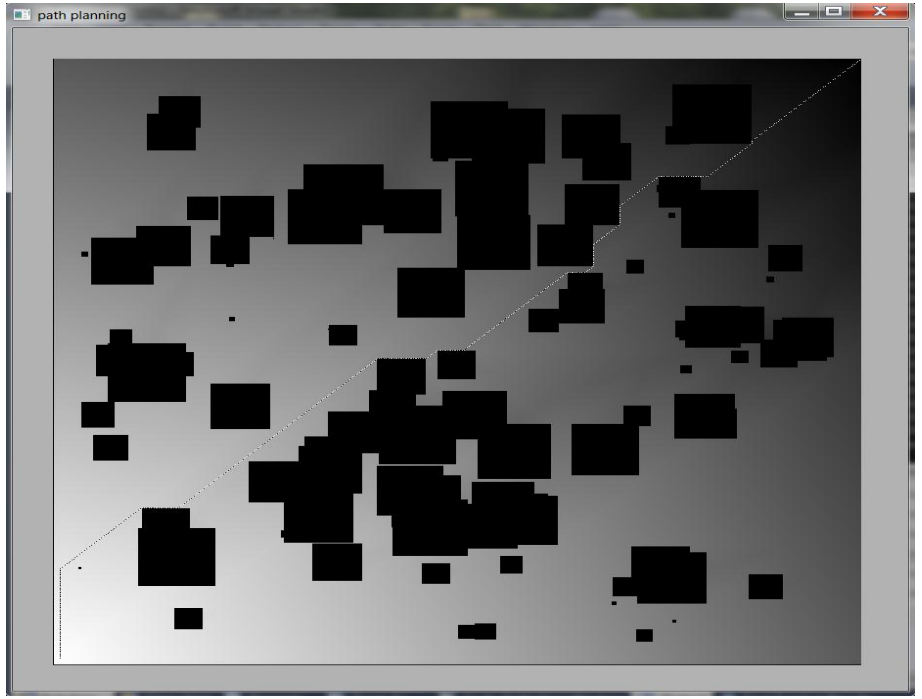
Path planning - performance



Grid 200x200
path in 82 ms



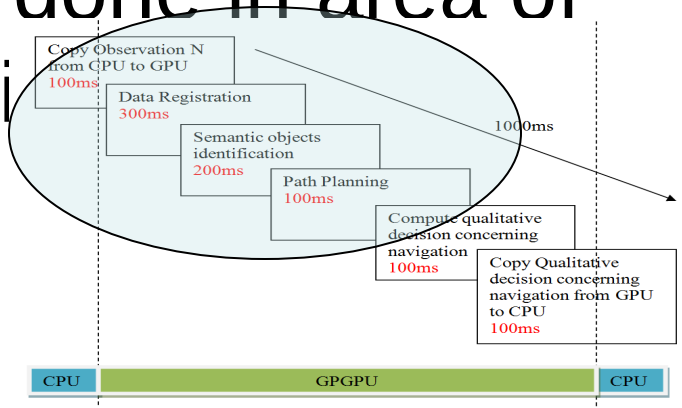
Grid 500x500
path in 148 ms



Grid 1000x1000
path in 570 ms

Conclusions

- The robot functionality using GPGPU computing (ICP, normal vectors, path planning) was presented.
- Still a lot of effort has to be done in area of semantic objects identificati



Future work

Continuation of GPGPU implementation of algorithms such as

-features extraction,

-filtering,

-segmentation

…

Study different NNS approach instead of regular grid decomposition

Thank You

- **Acknowledgements**

The research leading to these results has received funding from the European Community's Seventh Framework Programme

(FP7/2007-2013) under grant agreement n°285417.

GPU TECHNOLOGY CONFERENCE

GPU TECHNOLOGY CONFERENCE 2012
MAY 14-17, 2012 | SAN JOSE, CALIFORNIA