IMEKO International Conference on
Metrology for Archaeology and Cultural Heritage
Lecce, Italy, October 23-25, 2017

ICV and fine-registration algorithms for an efficient merging of point clouds

Domenica Costantino[1], Maria Giuseppa Angelini[1], Francesco Settembrini[2]

[1] DICATECh – Politecnico di Bari, V. le del Turismo, 8 – 74121 Taranto
(domenica.costantino, mariagiuseppa.angelini)@poliba.it
[2] DICAR – Politecnico di Bari, via A. Orabona, 4 – 70126 Bari
francesco.settembrini@poliba.it

***Abstract* – One of most important and frequents needs that arise when we have to handle the point clouds is to merging them in an efficient and precise manner so that we can obtain more complex and bigger 3D models by chunks, seamlessly. ICV, Intelligent Cloud Viewer, is a software developed in-house by AESEI (spin-off of Polytechnic of Bari) that contains many tools and filters for point cloud processing. In this paper we illustrate the fine-registration algorithm implemented in ICV, one of the countless variants of ICP methods (Iterative Closets Points).**

The fine-registration method can be usefully adopted to estimate the differences between point clouds of the same subject executed in different times or for a comparison of point clouds of same subject obtained with different acquisition techniques, for example a laser-scanner and a photogrammetric technique.

Furthermore, ICV integrates the M3C2 module for a robust error analysis, where by error is intended the differences or distances between similar point clouds.

I. INTRODUCTION

We define as scan-matching the determining, given two point-sets, of a rigid transform that leads to matching of them.

Usually a point-cloud is a result of an instrumental acquiring like an active, e.g. a laser-scanner, or a passive survey, e.g. a photogrammetric survey.

A common situation in which applies a scan-matching is in aligning of two point-clouds of which one, named P_{ref}, is taken as reference while the other, named *model-set* (P_{set}), will be superimposed to the first by mean of an affine transformation.

The purpose of scan-matching is in determining exactly the difference in position and asset between P_{set} and P_{ref} so that we can make the right and precise aligning of two scans.

Therefore, will occur to find a rotation R and a translation t so that we can define a measure of the distance between two scans and find a properly rigid transformation (translation, rotation and, eventually, scaling) that minimize this distance.

Normally, the spaces in which are carried out this operations have dimension 2 (e.g. in imaging registration) or else 3 (e.g. TLS point-cloud registration).

The perfect aligning of two real consecutive scans of the same subject is practically impossible since rarely the areas identified in the first scan are visible in the next one, due both to a limited field of view of a laser-scanner and to the presence of holes caused by occlusions. For that reason, by the years, research has led to a formulation of numerous algorithms of scan-matching, each of them aimed to improve the aligning precision as well as the robustness and the computing speed.

The Iterative Closest Point (ICP) is one of the algorithms of scan-matching more used. It is based on the association of matches point-to-point by mean of an iterative procedure in which first it calculate the matches between the scans to be compared and then it minimizes the distance error of their differences.

The least-squares solution come from a minimization of the following distance function, which is defined on *n* pairs of homologue points:

$$e^2(R,t) = \sum_{i=1}^{n} \|Rx_i + t - y_i)\|^2 \qquad (1)$$

with $i=1,2,...,n$, where x_i is the reference point-cloud while y_i is the model-set cloud.

ICP is, as the acronym says, an iterative method and, as all iterative methods, it can't converge if the initial conditions are not optimal that is if the initial positions and especially the attitudes of two clouds are too dissimilar.

The need for more and more robust and fast scan-matching techniques has resulted, over the years, in many variants and innovations in basic ICP algorithms.

Many of these variants were aimed at improving accuracy, rather than robustness (i.e. sensitivity in the presence of outliers) or running speed. Some of them are based on a point cloud pre-processing aimed at extracting key points and features, so concentrating the search for homologous points on certain regions with higher probability of matches (homologous points): this actually allows to reduce, also dramatically, the processing times.

To date, computational improvement in terms of speed rates can be obtained by indexing point-clouds in kd-trees.

Adopted for the first time in Bentley [4] kd-trees structure is particularly suitable for point-clouds as it is specifically designed for the organization of k-dimensional points (in our case the dimension is 3).

The kd-trees can be thought of as a binary tree in which each inner node implicitly generates a plane aligned with the axes that divide the space into two sub-spaces. The points on the right side of the plane belong to the right underside, the points on the left side of the plane belong to the left sub-tree.

The fig. 1 shows the basic algorithm for building a kd-trees.

```
Algorithm BuildKdTree($P, depth$)
1.  if $P$ contains only one point
2.     then return a leaf storing this point
3.     else if $depth$ is even
4.             then Split $P$ with a vertical line $\ell$ through the
                  median $x$-coordinate into $P_1$ (left of or
                  on $\ell$) and $P_2$ (right of $\ell$)
5.             else Split $P$ with a horizontal line $\ell$ through
                  the median $y$-coordinate into $P_1$ (below
                  or on $\ell$) and $P_2$ (above $\ell$)
6.          $v_{\mathrm{left}} \leftarrow$ BuildKdTree($P_1, depth+1$)
7.          $v_{\mathrm{right}} \leftarrow$ BuildKdTree($P_2, depth+1$)
8.          Create a node $v$ storing $\ell$, make $v_{\mathrm{left}}$ the left
            child of $v$, and make $v_{\mathrm{right}}$ the right child of $v$.
9.          return $v$
```

Fig. 1. Basic kd-trees algorithm

For each element of the point cloud reference, the search for the nearest point in the other point-cloud occurs by visiting the hierarchical reorganization of the latter in a kd-trees (but also octree and *r*-tree structures are equally good for this purpose) allowing it to immediately discard all the countless combinations that would surely give distances above the minimum; this leads to a significant increase in efficiency and, therefore, a significant reduction in processing time.

II. ICV SOFTWARE

Intelligent Cloud Viewer (ICV) is a software for point-clouds visualization and processing developed in-house by AESEI, a spin-off of Polytechnic of Bari.

Entirely written in C++ it adopt an object-oriented paradigm and a template metaprogramming.

It uses both standard libraries to manage advanced data structures such as lists, maps, kd-trees (STL [12], BOOST [8]), as well as open-source libraries for numerical computing (EIGEN [10], BLAS [11]) for computational geometry (CGAL [9]) and for computer graphics (SGI OpenInventor [13]).

Taking full advantage of modern multi-processor and multithreaded 64-bit architectures, ICV can handle point-clouds of also several tens of millions of points even without requiring particularly powerful computing systems.

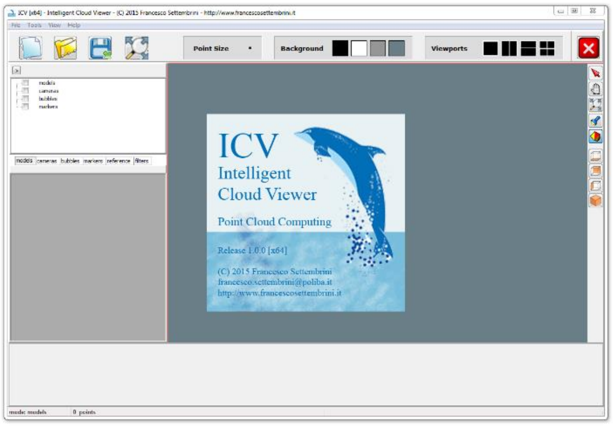


Fig. 2. ICV application interface

In order to speed up computational and graphic operations only a part of the set of points is displayed on the screen while all the calculations, in floating point and in double precision, are performed internally on the entire point cloud.

ICV processes point clouds from generic acquisitions such as TLS (Terrestrial Laser Scanner), LiDAR (Light Detection and Ranging) or photogrammetric survey and can read and save point clouds in popular formats such as PLY, TXT, LAS, XYZ, PTS, etc.

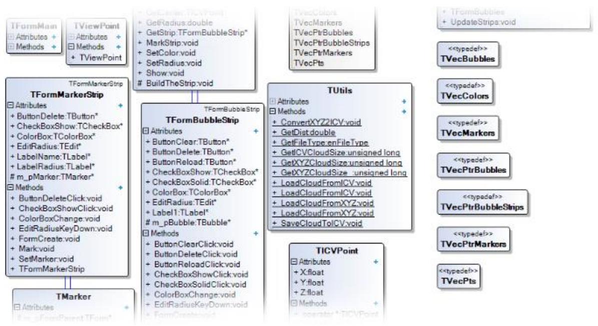


Fig. 3. UML class diagram for ICV components.

A number of filters have been implemented for point-clouds processing such as:

- ✓ cropping, useful for eliminating unwanted regions;

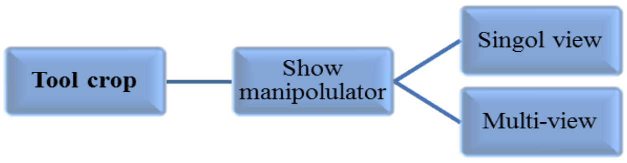


Fig. 4. Tool Crop

- ✓ decimation and simplify, for reorganizing (e.g. in voxel-grid) and reducing point-clouds density (fig. 5);
- ✓ cloning, for point-cloud replications;

- ✓ merging, for merging two or more point clouds into one;
- ✓ color, for selecting and processing portions of clouds that satisfy particular chromatic criteria (fig. 6);
- ✓ transform, for usual geometric transformations such as rototranslation and scaling (fig. 7).

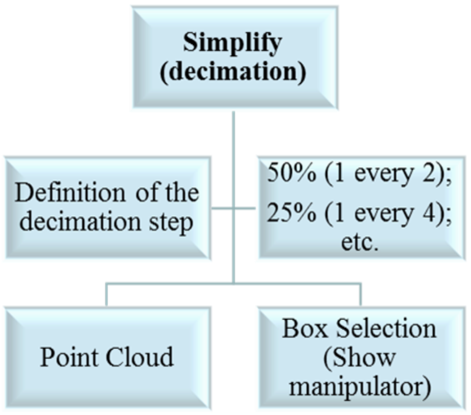

Simplify (decimation)
Definition of the decimation step
50% (1 every 2); 25% (1 every 4); etc.
Point Cloud
Box Selection (Show manipulator)

Fig. 5. Tool Simplify

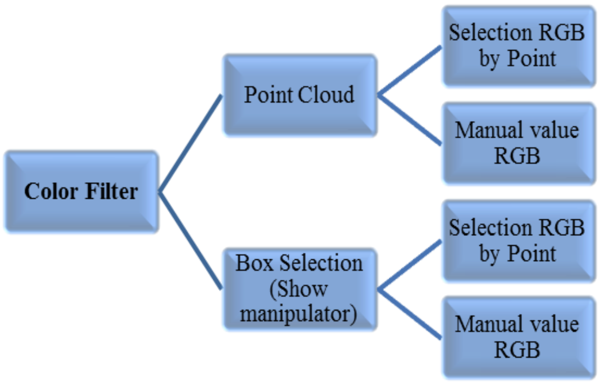
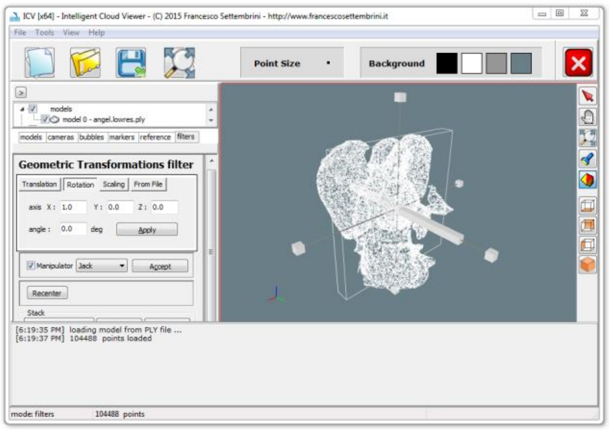


Fig. 6. Color filter


ICV [x64] - Intelligent Cloud Viewer - (C) 2015 Francesco Settembrini - http://www.francescosettembrini.it
File
Tools
View
Help
Point Size
Background
models
model 0 - angel.lowres.ply
models
cameras
bubbles
markers
reference
filters
Geometric Transformations filter
Translation
Rotation
Scaling
From File
Apply
Manipulator
Accept
Recenter
Stack

axis X : 1.0 Y : 0.0 Z : 0.0

angle : 0.0 deg

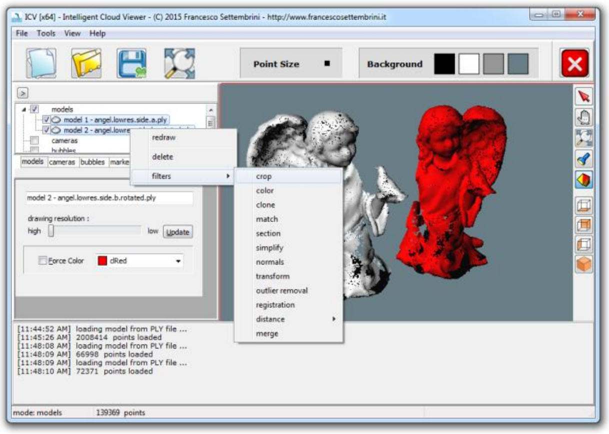
Manipulator: Jack

```
[6:19:35 PM]  loading model from PLY file ...
[6:19:37 PM]  104488  points loaded
```

mode: filters   104488 points

Fig. 7. Transform filter

And, last but not least, the very useful filters for coarse and fine-registration, those for cloud distance computing such as Cloud-to-Cloud (C2C) and M3C2 (Multiscale Model to Model Cloud Comparison [2]) for sections extraction (with the possibility of exporting geometries in standard CAD formats such as DXF), for outliers removal, etc. (fig. 8).



model 2 - angel.lowres.side.b.rotated.ply

Force Color: dRed

```
[11:44:52 AM]  loading model from PLY file ...
[11:45:26 AM]  2008414  points loaded
[11:48:08 AM]  loading model from PLY file ...
[11:48:09 AM]  66998  points loaded
[11:48:09 AM]  loading model from PLY file ...
[11:48:10 AM]  72371  points loaded
```

mode: models   139369 points

Fig. 8. Context-menu for filters available in ICV.

III. SCAN MATCHING IN ICV

For scan-matching algorithms implemented in ICV we have, substantially, two computational steps.

In the first one, so-called *coarse-registration*, we want to get a coarse alignment for point-clouds, usually by providing explicitly (i.e. giving *by hand*) the pairs of homologue points and then by applying [1].

In the next one, continuing from the previous step, we try to find, iteratively, an alignment between clouds as fine as possible, that is the *fine-registration*.

A. Coarse registration

One of the most common requirements when managing point clouds is to be able to assemble several instrumental acquisitions (often associated with different scale factors) of an object to obtain a more detailed and/or more complete results, as well evidenced in the example of data acquisition platform described in [8].

So, given two point patterns x_i an y_i with i=1,2,...,n in a m-dimensional space (m=3 in our examples), we want to find the affine transformation parameters (R: rotation, t: translation, c: scaling) that minimizes the root mean square of distance $e^2(R, t, c)$:

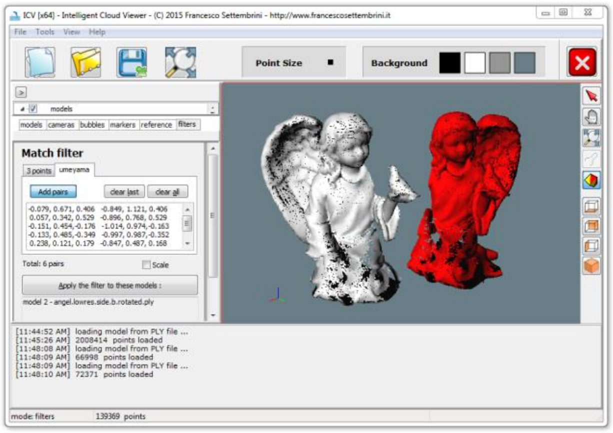
$$e^2(R,t,c) = \frac{1}{n}\sum_{i=1}^{n}\|y_i - (cRx_i + t)\|^2 \qquad (2)$$

For more details and a mathematical proof you can refer to [1].

ICV implement this method in the *match* filter.

Therefore, by mean of match filter, it is possible to align two point-clouds by identifying, manually, pairs of homologue points on each of two point sets.

For example, wanting to superimpose the *model* point-cloud on the right in fig. 9 (in red) to the *reference* point cloud on the left (in white) we select pairs of homologue points by clicking first on the reference point-cloud and then by clicking the homologue point on the *model*; so the algorithm calculates the rigid rototranslation parameters for the best superimposition.

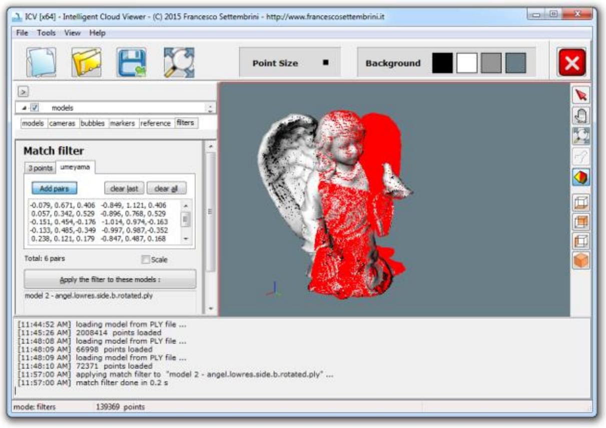

ICV [x64] - Intelligent Cloud Viewer - (C) 2015 Francesco Settembrini - http://www.francescosettembrini.it
File Tools View Help
Point Size
Background
models
models cameras bubbles markers reference filters
Match filter
3 points umeyama
Add pairs
clear last
clear all
Scale
Apply the filter to these models :

```
-0.079, 0.671, 0.406   -0.849, 1.121, 0.406
 0.057, 0.342, 0.529   -0.896, 0.768, 0.529
-0.151, 0.454,-0.176   -1.014, 0.974,-0.163
-0.133, 0.485,-0.349   -0.997, 0.987,-0.352
 0.238, 0.121, 0.179   -0.847, 0.487, 0.168
Total: 6 pairs
model 2 - angel.lowres.side.b.rotated.ply
```

```
[11:44:52 AM]  loading model from PLY file ...
[11:45:26 AM]  2008414  points loaded
[11:48:08 AM]  loading model from PLY file ...
[11:48:09 AM]  66998  points loaded
[11:48:09 AM]  loading model from PLY file ...
[11:48:10 AM]  72371  points loaded
```

```
mode: filters        139369  points
```

Fig. 9. Coarse-Registration filter settings.


ICV [x64] - Intelligent Cloud Viewer - (C) 2015 Francesco Settembrini - http://www.francescosettembrini.it
File Tools View Help
Point Size
Background
models
models cameras bubbles markers reference filters
Match filter
3 points umeyama
Add pairs
clear last
clear all
Scale
Apply the filter to these models :

```
-0.079, 0.671, 0.406   -0.849, 1.121, 0.406
 0.057, 0.342, 0.529   -0.896, 0.768, 0.529
-0.151, 0.454,-0.176   -1.014, 0.974,-0.163
-0.133, 0.485,-0.349   -0.997, 0.987,-0.352
 0.238, 0.121, 0.179   -0.847, 0.487, 0.168
Total: 6 pairs
model 2 - angel.lowres.side.b.rotated.ply
```

```
[11:44:52 AM]  loading model from PLY file ...
[11:45:26 AM]  2008414  points loaded
[11:48:08 AM]  loading model from PLY file ...
[11:48:09 AM]  66998  points loaded
[11:48:09 AM]  loading model from PLY file ...
[11:48:10 AM]  72371  points loaded
[11:57:00 AM]  applying match filter to  "model 2 - angel.lowres.side.b.rotated.ply" ...
[11:57:00 AM]  match filter done in 0.2 s
```

```
mode: filters        139369  points
```

Fig. 10. Coarse Registration output.

It's possible to give an arbitrary number of points (but four at least) and ICV calculates the affine transformation for best fit the latter model on the first.

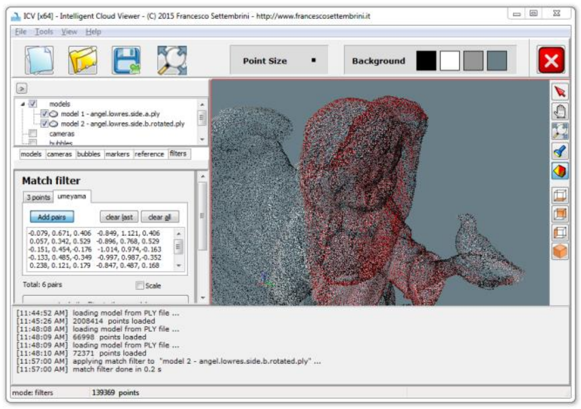
Generally, giving the points in input with sufficient accuracy, you can already get a good registration of point-clouds.

B. *Fine registration*

The coarse registration is often a preparatory step for a more accurate, i.e. *fine*, registration.

Because the fine-registration process is normally based on purely iterative methods in some cases, especially if the initial conditions are not favorable (e.g. in cases of very dissimilar point-clouds attitudes), the iterations may not converge to the solution.

For this, normally, the fine-registration step must be proceeded by the coarse-registration one.


ICV [x64] - Intelligent Cloud Viewer - (C) 2015 Francesco Settembrini - http://www.francescosettembrini.it
File Tools View Help
Point Size
Background
models
model 1 - angel.lowres.side.a.ply
model 2 - angel.lowres.side.b.rotated.ply
cameras
bubbles
models cameras bubbles markers reference filters
Match filter
3 points umeyama
Add pairs
clear last
clear all
Scale

```
-0.079, 0.671, 0.406   -0.849, 1.121, 0.406
 0.057, 0.342, 0.529   -0.896, 0.768, 0.529
-0.151, 0.454,-0.176   -1.014, 0.974,-0.163
-0.133, 0.485,-0.349   -0.997, 0.987,-0.352
 0.238, 0.121, 0.179   -0.847, 0.487, 0.168
Total: 6 pairs
```

```
[11:44:52 AM]  loading model from PLY file ...
[11:45:26 AM]  2008414  points loaded
[11:48:08 AM]  loading model from PLY file ...
[11:48:09 AM]  66998  points loaded
[11:48:09 AM]  loading model from PLY file ...
[11:48:10 AM]  72371  points loaded
[11:57:00 AM]  applying match filter to  "model 2 - angel.lowres.side.b.rotated.ply" ...
[11:57:00 AM]  match filter done in 0.2 s
```

```
mode: filters        139369  points
```

Fig. 11. Coarse-Registration (detail).

The main difference between those is that, while in the coarse-registration the homologue point pairs are explicitly given (since matching is known), in the fine-registration is the algorithm for itself (e.g. ICP [3]) that try to find the homologue points (taking into account some situations, e.g. outlier presence due to noise in instrumental acquiring , partial correspondence between clouds, etc.) and, iteratively, it proceed to the search of the affine transformation (rototranslation with, optionally, the scaling) that minimizes the differences between the two point clouds.

Given each point on the reference point-cloud a somewhat elementary (but also ineffective) method to find its counterpart on the second point-cloud is to iterate on all points of the latter in search of the closest point (usually the nearest point is just the homologue point sought).

Once funded the highest number of homologous points for two point-clouds, it search the rigid transformation (rototranslation with, eventually, the scaling) that minimizes a cost-function (generally as a cost-function is considered the average quadratic deviation of the euclidean distances but we can also consider the point normals compatibility, etc.).

However, since point clouds can also be composed of millions of points, we understands that such approach, which we might say *brute-force*, would normally be inapplicable because the search for homologous points would require no less than M*N operations ($O(M*N) \approx O(N^2)$).

Instead, you recur to a far more sophisticated method that rely on a point-cloud reorganization in particular hierarchical tree data structures, typically of the kd-trees type. The purpose of this reorganization is to allow, by taken a point on the cloud A, to search as quickly as possible the nearest point (nearest-neighbor) on cloud B.

The use of kd-trees reduce the computational complexity in the searching of nearest-neighbor from

*O(M*N)* to only *O(M*log(N))* and this lead to a big difference in computations with enormous increments in terms of speed.

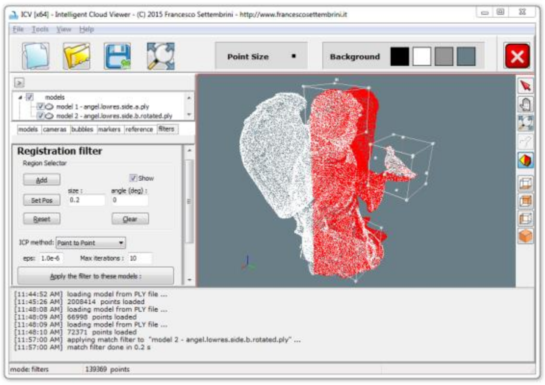
For the fine-registration, ICV implements a variant of the ICP algorithm proposed by Arun et al. [3], which is based on finding the least squares solution for *R* and *t*, deriving this from the transformation matrix 3x3 reduced to singular values (SVD, Singular Value Decomposition).

In one of the early implementations of the algorithm (Costantino et al. [5], [6]) we considered the two point-clouds in their entirety, with little variants [7].

Currently, to further speed up the fine-registration procedures, ICV uses a simple but effective optimization technique that we can define as of *supervised* type: it is the operator that indicate the matching features by positioning bounding boxes to focus the search for homologous points only in those areas (figg. 12-13).

In this way it reduce considerably the calculation times and can be rapidly aligned point-clouds even of many tens of millions of points such as that derived by high density TLS acquisitions or that obtained by aerial photogrammetric restitutions of wide geographical areas.

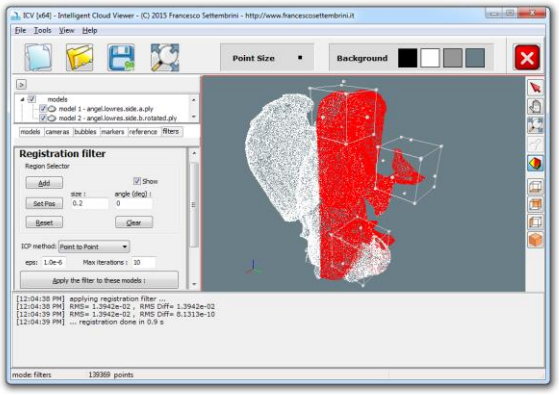
For the fine-registration of our example model, i.e. to superimposing accurately the point cloud on the right in fig. 12 (the model-set, in red) on to that on the left (the reference-set, in white) we have selected three key regions marked by the relative bounding-boxes.


ICV [x64] - Intelligent Cloud Viewer - (C) 2015 Francesco Settembrini - http://www.francescosettembrini.it
Point Size
Background
model 1 - angel.lowres.side.a.ply
model 2 - angel.lowres.side.b.rotated.ply
Registration filter
Region Selector
Add
Set Pos
Reset
Clear
Apply the filter to these models :
size : 0.2 angle (deg) : 0
ICP method: Point to Point
eps: 1.0e-6 Max iterations : 10

```
[11:44:52 AM]  loading model from PLY file ...
[11:45:26 AM]  2008414  points loaded
[11:48:08 AM]  loading model from PLY file ...
[11:48:09 AM]  66998  points loaded
[11:48:09 AM]  loading model from PLY file ...
[11:48:10 AM]  72371  points loaded
[11:57:00 AM]  applying match filter to  "model 2 - angel.lowres.side.b.rotated.ply" ...
[11:57:00 AM]  match filter done in 0.2 s
```

mode: filters 139369 points

Fig. 12. Fine-Registration filter setup.


ICV [x64] - Intelligent Cloud Viewer - (C) 2015 Francesco Settembrini - http://www.francescosettembrini.it
Point Size
Background
model 1 - angel.lowres.side.a.ply
model 2 - angel.lowres.side.b.rotated.ply
Registration filter
Region Selector
Add
Set Pos
Reset
Clear
Apply the filter to these models :
size : 0.2 angle (deg) : 0
ICP method: Point to Point
eps: 1.0e-6 Max iterations : 10

```
[12:04:38 PM]  applying registration filter ...
[12:04:38 PM]  RMS= 1.3942e-02 ,  RMS Diff= 1.3942e-02
[12:04:39 PM]  RMS= 1.3942e-02 ,  RMS Diff= 8.1313e-10
[12:04:39 PM]  ... registration done in 0.9 s
```

mode: filters 139369 points

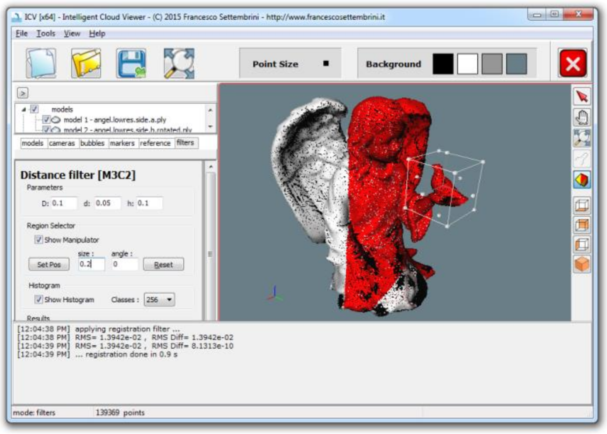
Fig. 13. Fine Registration.

IV. POINT CLOUDS DISTANCES

To evaluate the goodness of fine-registration process we have done an analysis of the differences (distances) between the point-clouds in the overlapping regions.

In addition to the simple C2C (cloud-to-cloud) method for distance calculation between two point-clouds, in ICV has been also implemented the M3C2 method [2], normally more robust and usually much more accurate than C2C. Since in the examples of this article the two point-clouds have been obtained from cropping of the same model it is expected that the fine-registration process, if really valid, will lead to a nearly null error.

To speed-up the processing, once again, we decided to limiting the operations on the sample region by mean of bounding-box, as illustrated in fig.14.


ICV [x64] - Intelligent Cloud Viewer - (C) 2015 Francesco Settembrini - http://www.francescosettembrini.it
Point Size
Background
model 1 - angel.lowres.side.a.ply
Distance filter [M3C2]
Parameters
Region Selector
Show Manipulator
Set Pos
Reset
Histogram
Show Histogram
Results
D: 0.1 d: 0.05 h: 0.1
size : 0.2 angle : 0
Classes : 256

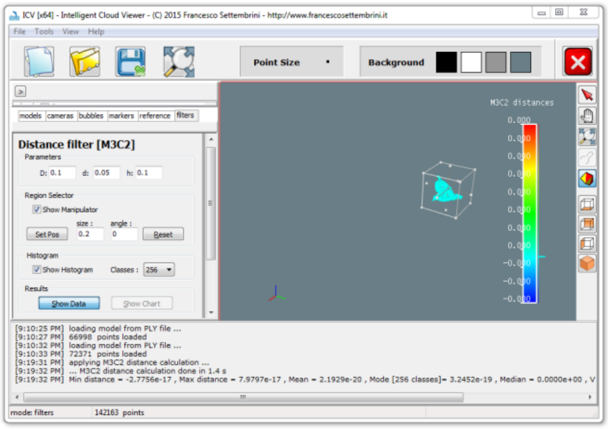
```
[12:04:38 PM]  applying registration filter ...
[12:04:38 PM]  RMS= 1.3942e-02 ,  RMS Diff= 1.3942e-02
[12:04:39 PM]  RMS= 1.3942e-02 ,  RMS Diff= 8.1313e-10
[12:04:39 PM]  ... registration done in 0.9 s
```

mode: filters 139369 points

Fig. 14. Point-Clouds distances setup (M3C2).

By analyzing the results of the M3C2 operation it is possible to see how the differences in overlapping regions of the two point-clouds are practically equal to zero, with values for average, mean, median and RMS in the order of between 10^{-18} and 10^{-20} (figg. 15-16).

Therefore, we can deduce that, after the fine-registration process, the two clouds match perfectly.

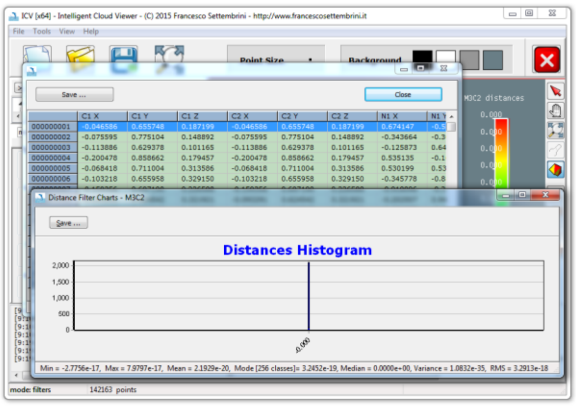


D: 0.1 d: 0.05 h: 0.1
size : 0.2 angle : 0
Classes : 256

```
[9:10:25 PM]  loading model from PLY file ...
[9:10:27 PM]  66998  points loaded
[9:10:32 PM]  loading model from PLY file ...
[9:10:33 PM]  72371  points loaded
[9:19:31 PM]  applying M3C2 distance calculation ...
[9:19:32 PM]  ... M3C2 distance calculation done in 1.4 s
[9:19:32 PM]  Min distance = -2.7756e-17 , Max distance = 7.9797e-17 , Mean = 2.1929e-20 , Mode [256 classes]= 3.2452e-19 , Median = 0.0000e+00 , V
```

mode: filters 142163 points

Fig. 15. Point-Clouds distances calculation (M3C2).


ICV [x64] - Intelligent Cloud Viewer - (C) 2015 Francesco Settembrini - http://www.francescosettembrini.it
File Tools View Help
Point Size
Background
Save ...
Close
M3C2 distances
Distance Filter Charts - M3C2
Save ...
Distances Histogram
2,000
1,500
1,000
500
0
0.000
mode: filters

	C1 X	C1 Y	C1 Z	C2 X	C2 Y	C2 Z	N1 X	N1 Y
000000001	-0.046586	0.655748	0.187199	-0.046586	0.655748	0.187199	0.674147	-0.5
000000002	-0.075595	0.775104	0.148892	-0.075595	0.775104	0.148892	-0.343664	-0.3
000000003	-0.113886	0.629378	0.101165	-0.113886	0.629378	0.101165	-0.125873	0.64
000000004	-0.200478	0.858662	0.179457	-0.200478	0.858662	0.179457	0.535135	-0.1
000000005	-0.068418	0.711004	0.313586	-0.068418	0.711004	0.313586	0.530199	0.53
000000006	-0.103218	0.655958	0.329150	-0.103218	0.655958	0.329150	-0.345778	-0.8

Min = -2.7756e-17, Max = 7.9797e-17, Mean = 2.1929e-20, Mode [256 classes]= 3.2452e-19, Median = 0.0000e+00, Variance = 1.0832e-35, RMS = 3.2913e-18

142163 points

Fig. 16. M3C2 charts and tables.

V. CONCLUSIONS AND FUTURE PROSPECTS

With the adopted implementations it has been possible to obtain, even in this software pre-release, good processing speeds for the scan matching step without sacrificing the robustness and the precision of the method.

ICV is a work-in-progress: although, at present, it implements several tools for point cloud management it is still far from being a complete product.

There are many possible improvements: from massive parallelism to increase computational performance to the use of more powerful and/or smart fine-registration algorithms (e.g. with feature-detection techniques), to the integration of image-processing algorithms for the processing, for example, of geo-tagged orthophotos mapped on to point-clouds (texture mapping), to the geometries extraction for automated generation of whole 3D urban areas. More transversally the use of ICV could be tested to obtain an accurate 3D thermal map useful for electronic device characterization improving some actual 2D mapping techniques [9].

REFERENCES

[1] S.Umeyama, “Least-squares estimation of transformation parameters between two point patterns”, IEEE Transactions On Pattern Analysis And Machine Intelligence, vol. 13, No. 4, 1991, pp. 376–380.

[2] D.Lague1, N.Brodu, J.Leroux, “Accurate 3D comparison of complex topography with terrestrial laser scanner: application to the Rangitikei canyon (N-Z)”, Journal of Photogrammetry and Remote Sensing, vol. 82, 2013, pp. 10-26.

[3] K.S.Arun, T.S.Huang, S.D.Blostein, “Least square fitting of two 3-d point sets”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 9, No. 5, 1987, pp. 698-700.

[4] J.L.Bentley, “K-d trees for semidynamic point sets”, Proceedings of the sixth annual symposium on Computational geometry, 1990.

[5] D.Costantino, M.G.Angelini, F.Settembrini, “Point cloud management through the realization of the intelligent cloud viewer software”, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. 42, No. 5W1, 2017, pp. 105-112.

[6] M.Caprioli, D.Costantino, F.Mazzone, et al., “Point clouds from different photographic sensors for cultural heritage surveying” - Conference: 14th International Forum of Studies Location: World Heritage and Degradation: Smart Design, Planning and Technologies Book Series: Fabbrica della Conoscenza, 2016, pp. 253-261.

[7] D.Costantino, A.Guarnieri, A.Vettore, M.Camarda, “Automatic Registration of Large Range Datasets with Spin-Images”, Journal of Cultural Heritage, vol. 12, 2011, pp. 476-484.

[8] G.Andria, F.Attivissimo, A.Di Nisio, A.M.L. Lanzolla, A.Pellegrino, “Development of an automotive data acquisition platform for analysis of driving behavior”, Measurement: Journal of the International Measurement Confederation, vol. 93, November 2016, pp. 278-287.

[9] F.Attivissimo, A. DiNisio, C.Guarnieri Calò Carducci, M.Spadavecchia, “Fast Thermal Characterization of Thermoelectric Modules Using Infrared Camera”, IEEE Transactions on Instrumentation and Measurement, vol. 66, n. 2, 2017, pp. 305–314.

[10] http://www.boost.org

[11] http://www.cgal.org

[12] http://eigen.tuxfamily.org

[13] http://www.netlib.org/blas

[14] https://www.sgi.com/tech/stl

[15] http://oss.sgi.com/projects/inventor