

e.g., searching for the 'closest' search-point to the query-point (after transformation) that upholds the continuous function of the reference representation, i.e., $f(x, y, z)$. Consequently, this tangent search criteria process upholds the following constraints, by implementing Least Squares target function minimization:

1. The search-point upholds a surface function existing in a local plane in the reference representation (a TIN mesh was used in this research, e.g., triangular-plane, represented by heights Z_1, Z_2 , and Z_3); and,
2. Query-point after transformation $g^t(x, y, z)_i$ and search-point $f(x, y, z)_i$ form a vector that has to be a tangent to the local plane function (shortest path transformed from query-point to local plane). Thus, first order derivative in both x (2nd constraint) and y (3rd constraint) are generated.

The schematics of this point-to-plane (tangent) strategy are given in Figure 1. It can be described as if these criteria of constraints shift the vector between the points $g^t(x, y, z)_i$ (dark-grey) and $f(x, y, z)_i$ (black) over the local triangle plane towards its final position, thus minimizing the Least Squares target function until converging to the final positioning of $f(x, y, z)_i$.

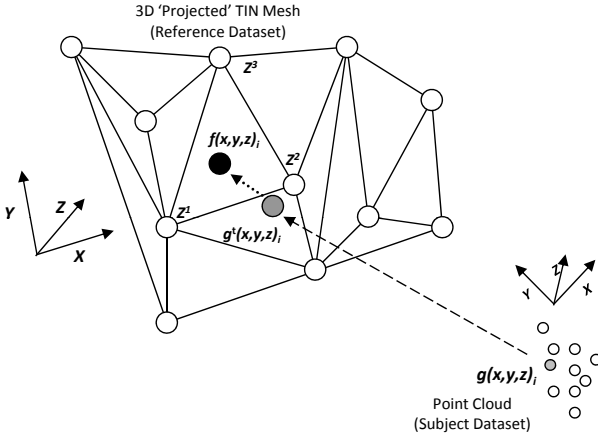


Figure 1: Schematic explanation of Least Squares target function minimization via three constraints

For each query-point i in dataset g a nearest search-point from dataset f is paired-up as long as the three constraint criteria are fulfilled. Consequently, with all the point pairs extracted, a rigid transformation is achieved. The ICP process is carried out iteratively until predefined statistical criteria are achieved, which are based on:

1. Value differences of the n -parameter in each consecutive iteration (can also be denoted as spatial difference);
2. Number of iterations - a large number of iterations indicates that both representations functions are different, i.e., matching might be impossible. This number is derived by several factors, such as: number of points, congruency percentage of models, etc.; and,
3. Number of homologous counterpart points - the larger the number, the more robust the matching. A number smaller than a certain threshold indicates that a solution might not be reliable enough.

3.3 Point Cloud TIN Mesh

In order to determine the mutual points correspondences based on the point-to-plane strategy a 3D surface of the reference model has to be established, based on the scan points. The most obvious

solution might suggest generating the complete 3D surface of the reference model in advance and to use it during the process. In various experiments we have found that there is no fast - and more importantly - reliable surface reconstruction method for arbitrary point-based laser scans. The mesh created is not an explicit one and existing data outliers, even a small number, produced an unusable 3D surface meshes. Therefore we have decided to create local meshes from the kNN points in the reference model for each query-point. The local meshes are created on demand during the process. At first, the kNN of the query point has to be found in the reference model. It was determined that $k = 32$ neighbors produced good results; this number ascertains that a reliable triangle mesh is constructed, representing the scanned surface: wide enough for the projection task, while avoiding harming computation time. Since the reference dataset is represented by an Octree, the kNN are determined as follows:

1. Find the Octree cell c containing the query-point q and at least k points, while the child s of c (which also contains q) contains less than k points ($\mathcal{O}(1)$); and,
2. Do a linear kNN-search for all points contained in cell c ($\mathcal{O}(k)$).

The second step entails the projection of the kNN points found in c from 3D to 2D space. For that, we compute an oriented bounding box of these points, where the box axes are the eigenvectors of the covariance matrix of the k points. As shown in Figure 2, once the oriented bounding box is computed, the points are projected onto the largest plane of that bounding box, while the correspondences between the original and projected points are stored. The 2D points are triangulated using an ordinary Delaunay triangulation algorithm. Finally, the triangulation is transformed (projected backward) from 2D to 3D space by using the previously stored correspondences. The result is a local 3D surface mesh of the given k nearest neighbor points.

Finally, as described in 3.2, the query-point is projected orthogonally onto the plane of each of the mesh's triangles that were generated. Only if the projected point falls into the bounds of a certain triangle while validating the three constraints, it can be considered valid. Overall, three cases might exist, where the query-point q :

1. Cannot be projected on any of the triangles of the mesh;
2. Can be projected on exactly one triangle; and,
3. Can be projected on several triangles.

In the first case, the query-point is skipped for the current iteration. The second case represents a unique transformation, while the third case is somewhat ambiguous. The ambiguity is solved by choosing the projection with the smallest Euclidean distance between the query-point and the projected point (plane). In order to avoid meshes with outliers, which might be expressed by large triangles, a median triangle area threshold is applied, which is defined by all triangles containing search-points found within the first iteration.

3.4 Caching

Since the process described in 3.3 is computationally intensive, we propose a caching strategy of the already built 3D meshes. Once a mesh is created, it is stored within the Octree, or more precisely within the Octree cell c originally used for the kNN determination. Thus, the next time before a kNN search of a specific query-point is executed, the cache of the corresponding

Figure 2: Schematic explanation of 3D 'projected' TIN mesh creation

Octree cell is searched for existing meshes the query-point can be projected on. If the query-point can be projected onto a mesh from the cache the correspondence is stored. Otherwise, if no usable cached mesh is found, all steps described in 3.3 are carried out and a new mesh is built, which will be cached for further reuse.

Due to memory limitations, the size of the cache is limited to a fixed number of meshes per Octree cell. In our experiments, we used a maximum cache size of ten meshes, while new meshes are preferred over old ones. Once the cache for a particular cell contains ten meshes, and a new mesh is required to be cached, the oldest mesh from the cache is removed, while the new one is stored. Since the reference points and their corresponding meshes are not subject to transformations, the cache is preserved over consecutive iterations.

3.5 SIMD Strategy

To reduce computation time we introduce an advanced Octree data structure to achieve fast kNN determination as well as caching for the reuse of existing meshes. As shown in (Eggert and Paelke, 2010), the kNN determination as one of the major computational bottlenecks can be handled via SIMD-based strategies, such as GPGPU (General-Purpose computing on Graphics Processing Units) or utilizing multiple CPU cores in parallel. Thus, we propose a Single Instruction, Multiple Data (SIMD) strategy to exploit modern hardware capabilities. In contrast to the Single Instruction, Single Data (SISD) architecture, SIMD applies the same operation(s)/instruction(s) to not only one data unit, but to multiple ones. Therefore, the processed data has to be independently processable. Moreover, the processing itself has to be designed for parallel execution. As in the proposed ICP process, the transformation for each query-point is determined independently to finally derive a global one. Hence, the process, as well as the data, is suitable for a parallel processing strategy, such as the strategy proposed here. As shown in Figure 3, all instructions involving a single query-point are executed in parallel:

1. kNN determination
2. Oriented Bounding Box (OBB) determination
3. 3D -> 2D point projection
4. 2D Delaunay triangulation
5. 2D -> 3D triangulation projection
6. Point-to-plane projection
7. Transformation calculation

After all individual transformations are calculated, a global transformation is derived from all individual ones. Since all intensive computational parts are parallelized, the process easily scales in respect to the number of available processing units. Meaning that doubling the number of processing units almost cuts the needed computation time in half.

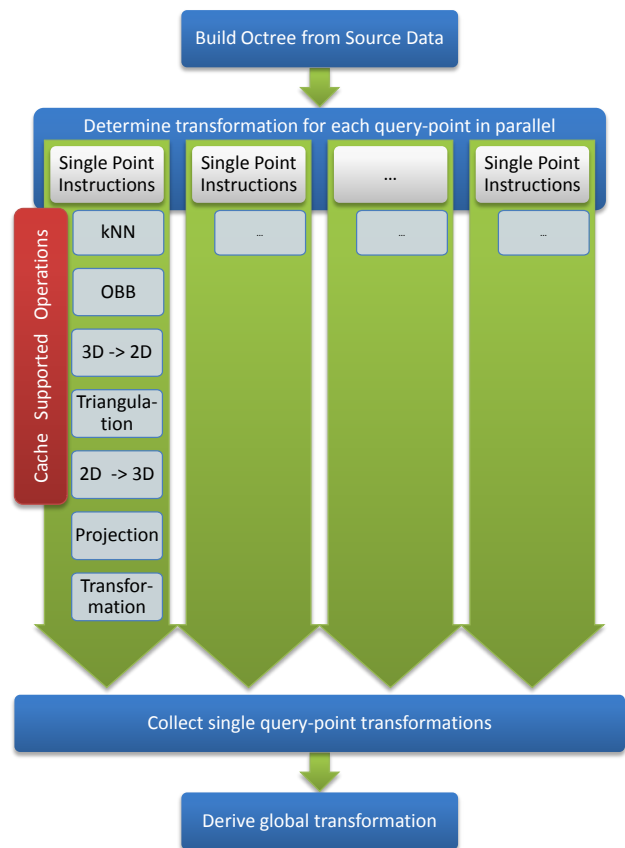


Figure 3: SIMD strategy

Nonetheless, using a SIMD-based processing implementation involves the use of suitable (thread-safe) data structures. While read only operations are usually thread safe, all operations altering the data structure (e.g., adding or removing elements) are not. Since the Octree containing the reference points, as well as the point list containing the subject dataset, are not altered during the process, no precautions have to be taken. In contrast, the cache is extensively altered by adding new and removing old meshes. As the experiments were carried out using a Java-based implementation, a *ConcurrentLinkedQueue* has been used as a thread-safe cache data structure. The other crucial altering data structure is the map storing the single transformation for each point during an iteration. Here, Java provides the thread safe *ConcurrentHashMap*, which serves our needs.

4 EXPERIMENTAL RESULTS AND DISCUSSION

Our experiments were carried out using several different datasets: the well known Stanford bunny model and the Welfenschloss model, which was scanned both by terrestrial and mobile laser scanner. The first experiment presented here involved the registration and alignment of the Stanford bunny model. Since the strategy presented here is designed for an ICP workflow for the registration of a dense as well as a coarse point datasets, we simplified the bunny model, originally containing 36,000 vertices, to a model containing 18,000 vertices. The second experiment presented here involved the registration and alignment of the Welfenschloss model from different scans, each presenting different point-of-view, density and LOD: the mobile laser scan is a more coarse model containing 32,000 vertices, while the terrestrial one is much denser with 168,000 vertices.

The proposed algorithm was implemented using the Java programming language, while SIMD was realized using Java's fork/join framework. The experiments were conducted on an Intel core i7 870 machine with four physical and eight logical cores running at 2.93 GHz.

Since the two bunny datasets are virtually the same model with different LOD, an arbitrary synthetic shift was applied to one of the models. The applied shift was in the range of 20% on the x -axis, 10% on the y -axis, and 0.5% on the z -axis in respect to the models' bounding box. This demonstrates quite a large displacement of the models, which enable to evaluate the robustness of the proposed methodology in converging to the right alignment. The starting point constellation of the models is depicted in Figure 4 (top-left). The proposed registration and alignment algorithm converged after 17 iterations, with a spatial displacement difference threshold of 0.0002m between consecutive iterations (less than 0.2% of the average bounding box axis length). The registration and alignment result, depicted in Figure 4 (bottom-right), achieved an accuracy of 98.5% on the x - and y -axis, and 97.5% on the z -axis in respect to the synthetic shift applied. Iterations 6 and 14 are also depicted in Figure 4 (top-right) and (bottom-left), respectively.

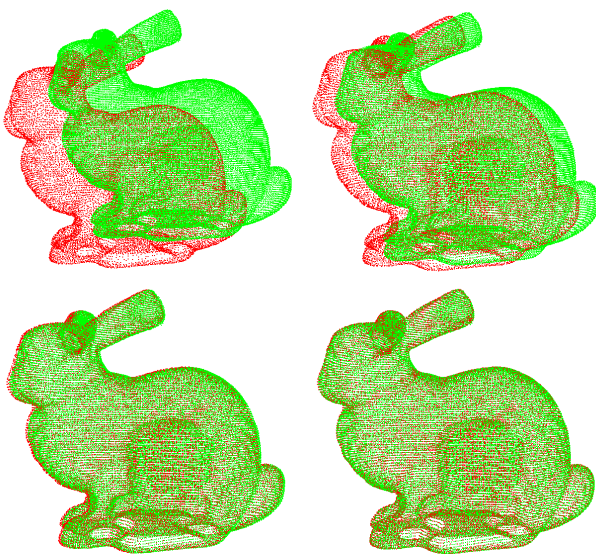


Figure 4: ICP process applied to the Stanford bunny point cloud with coarse subject (red) and dense reference (green)

In contrast to the bunny datasets, the two Welfenschloss datasets present a different origin, and thus a different point of view.

Hence, there is no reference to which the registration and alignment result can be compared to. First, the two models are manually roughly registered to serve as start point, as depicted in Figure 5 (top). The registration process converged after 30 iterations, with a spatial displacement difference threshold of 0.01m between consecutive iterations (being less than 0.0001% of the average bounding box axis length). The registration and alignment result is depicted in Figure 5 (bottom).

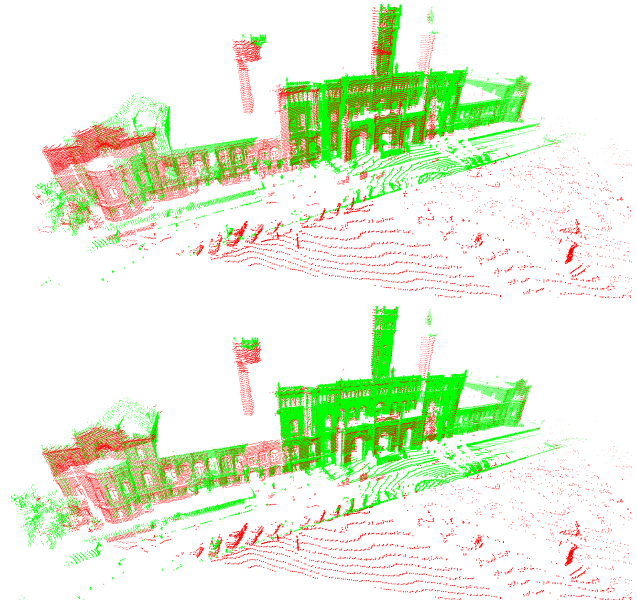


Figure 5: ICP process applied to the Welfenschloss point cloud with coarse subject from the mobile laser scanner (red), and dense reference from terrestrial laser scanner (green)

The statistics and figures clearly show that the proposed algorithm is robust and reliable. To present advantages of the proposed performance enhancements, namely Octree data structure, caching, and parallel processing, the ICP process was also implemented in its straight-forward form (referred to here as list-based, i.e., "classic" ICP implementation with point-to-plane strategy), with and without mesh caching, and with various number of processing threads. The duration of all processes are depicted in Figure 6 for the bunny model (left), and for the Welfenschloss model (right). The graphs show the runtime in seconds of all processes in the y -axis in respect to the number of used threads (SIMD level) in the x -axis (both in logarithmic scale). The list-based ICP processes with caching disabled (red) and caching enabled (green) are depicted in respect to the proposed methodology with caching disabled (blue) and caching enabled (pink), respectively. First, it is clear that caching accelerates running time for both processes, suggesting efficiency without harming the process reliability. Second, it is clear that our proposed methodology performs best, suggesting an improvement of up to 65 times faster than the straight forward "classic" implementation (list-based, non-cached, single threaded), as in the Welfenschloss experiment, when optimizing the number of threads to 8 SIMD. It is worth noting that when the number of threads was increased to 16, all processes showed some decline in their running time, which is the result of an overhead with existing configuration of our working environment (providing 8 processing cores). A more powerful computing system (providing a larger number of processing cores) will suggest even better results. It should be noted that the list-based process with caching enabled did not always converge. The reason might be the insufficient caching strategy: since there is

no way to georeference the cached meshes a global cache has to be used, which might introduce a large number of false point-to-plane correspondences. In contrast, an Octree based data structure, as used in the proposed methodology, enables local (georeferenced) mesh caching. In terms of number of iterations and matching results all processes (except for the list-based cached) terminated with similar values.

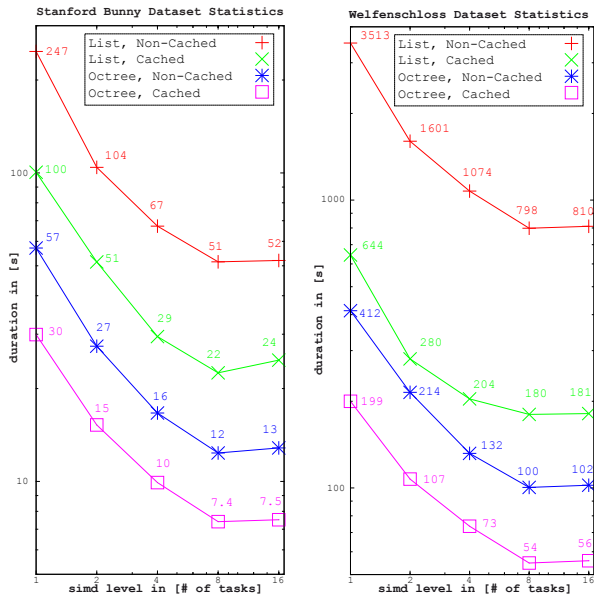


Figure 6: Duration statistics

5 CONCLUSIONS AND FUTURE WORK

This paper proposed an ICP methodology for the registration and alignment of 3D point clouds. The computational complexities introduced by the straight forward ICP are tackled by the application of specific performance enhancements, such as the use of an Octree data structure, caching strategies for the reuse of existing data, as well as the SIMD parallelization scheme. The overall results of the experiments showed a sharp decrease in running times - up to 65 times faster than the straight forward algorithm, while maintaining a reliable and robust solution. This methodology present a step forward towards an ICP process that is much more efficient with near real-time computing capabilities.

Our future work will involve the extension of the SIMD feature to make use of the GPGPU capabilities of modern graphic card - and not merely the use of multiple CPU cores. Furthermore, the caching strategy is planned to be fine tuned in order to increase the robustness of the solution and further improve the computation speed.

ACKNOWLEDGEMENTS

This joint research project was financially supported by the State of Lower-Saxony and the Volkswagen Foundation, Hannover, Germany.

REFERENCES

Akca, D. and Gruen, A., 2005. Recent advances in least squares 3d surface matching. In: *Optical 3-D Measurement Techniques VII*, pp. 197–206.

Besl, P. J. and McKay, N. D., 1992. A method for registration of 3-d shapes. *IEEE Trans. Pattern Analysis and Machine Intelligence* 14(2), pp. 239–256.

Dalley G., F. P., 2002. Pair-wise range image registration: A study in outlier classification. *Computer Vision and Image Understanding* 87(1–3), pp. 104–115.

Eggert, D. and Paelke, V., 2010. Relevance-driven acquisition and rapid on-site analysis of 3d geospatial data. In: *Proceedings of the Joint International Conference on Theory, Data Handling and Modelling in GeoSpatial Information Science*, Vol. 38number 2, pp. 118–123.

Fitzgibbon, A. W., 2003. Robust registration of 2d and 3d point sets. *Image Vision Comput* pp. 1145–1153.

Greenspan, M. and Yurick, M., 2003. Approximate k-d tree search for efficient icp. In: *3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings. Fourth International Conference on*, pp. 442–448.

Gruen, A., 1996. Least Squares Matching: a Fundamental Measurement Algorithm. Whittles, chapter Close Range Photogrammetry and Machine Vision, pp. 217–255.

Gruen, A. and Akca, D., 2004. Least squares 3d surface matching. In: *IAPRS, 34(5/W16)*.

Guehring, J., 2001. Reliable 3d surface acquisition, registration and validation using statistical error models. In: *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pp. 224–231.

Horn, B. K. P., Hilden, H. and Negahdaripour, S., 1988. Closed-form solution of absolute orientation using orthonormal matrices. *Journal of the Optical Society America* 5(7), pp. 1127–1135.

Langis, C., Greenspan, M. and Godin, G., 2001. The parallel iterative closest point algorithm. In: *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pp. 195–202.

Mount, D. M., Netanyahu, N. S. and Moigne, J. L., 1999. Efficient algorithms for robust feature matching. *Pattern Recognition* pp. 17–38.

Plackett, R. L., 1950. Some theorems in least squares. *Biometrika* 37(1/2), pp. 149–157.

Qiu, D., May, S. and Nüchter, A., 2009. Gpu-accelerated nearest neighbor search for 3d registration. In: *Proceedings of the 7th International Conference on Computer Vision Systems: Computer Vision Systems, ICVS '09*, pp. 194–203.

Rusinkiewicz, S. and Levoy, M., 2001. Efficient variants of the icp algorithm. In: *Proceedings of the Third Intl. Conf. on 3D Digital Imaging and Modeling*, pp. 145–152.

Strand, M., Erb, F. and Dillmann, R., 2007. Range image registration using an octree based matching strategy. In: *Mechatronics and Automation, 2007. ICMA 2007. International Conference on*, pp. 1622–1627.

Szeliski, R., 1996. Matching 3-d anatomical surfaces with non-rigid deformations using octree-splines. *International Journal of Computer Vision* 18, pp. 171–186.

Wang, J. and Shan, J., 2005. Lidar data management with 3-d hilbert space-filling curve. In: *ASPRS Annual Meeting, Baltimore, March 7-11*.

Zhang, Z., 1994. Iterative point matching for registration of free-form curves and surfaces.