DOI 10.5162/ettc2022/5.3

Automated Registration of Large Assemblies with Robot Guided 3D Sensors and CAD-based Planning

Steffen Sauer[1], *Daniel Sopauschke*[1], *Thomas Dunker*[1], *Michael Heizmann*[2], *Dirk Berndt*[1]
[1] *Fraunhofer Institute for Factory Operation and Automation IFF, Magdeburg, Germany,*
[2] *Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany*
steffen.sauer@iff.fraunhofer.de

Abstract:
We address the problem of automated inspection of rivets and brackets in aircraft fuselage shells with a robot guided 3D-sensor. High variety of such assemblies requires that an inspection automatically compares acquired 3D data to the CAD model representing the assembly. For this purpose, a precise sensor pose with respect to the fuselage shell coordinate system is needed. In this industrial context, the current set-up can deviate from the model by few centimeters which is too much for inspection. We present an automated registration method, which does not need further sensors for tracking. This method reduces the deviation between measured points and the CAD model to few tenth of a millimeter in regions with no assembly defect. Therefore, we find automatically measurement positions for registration which cover all degrees of freedom. The algorithm for registering the 3D points to the CAD model avoids the assignment of points to faces which cannot be reached by the sensor, and parts to be inspected. In addition, local registration can deal with slight deformation of the fuselage shell.

Key words: robot guided inspection, CAD model, automated viewpoint selection, point cloud registration, simulation

Motivation

We consider inspection of rivets and brackets inside an aircraft fuselage shell. As assembly errors may cause high costs in downstream processes or could lead to dangerous malfunctions, a reliable and automated inspection solution is preferred to the frequent visual inspection by a second person. Such solution must be able to inspect new part configurations without time consuming preparation effort. Digital models of assembly and work shop equipment provide necessary information for model-based inspection. Robot guided sensors assure the needed flexible acquisition of camera images and 3D point clouds.

CAD-based approaches to this inspection task need an estimate of the sensor position with respect to the coordinate system of the CAD model in order to compare an image or point cloud to the nominal model. Using the estimated sensor position, one can map the acquired data to the model. The remaining deviation between model and data of error free parts should be much smaller than the smallest deviations caused by assembly defects.

Fig. 1: Robot guided sensor inspecting a fuselage shell

We use a 3D-sensor with a measurement volume of about 160 x 100 x 110 mm^3 and a working distance of 160 mm, see Fig. 1. A hand-eye calibration showed a sufficiently small 6D uncertainty of the robot deviation on the target below one millimeter. The robot is able to inspect a volume of about 2.8 x 2.5 x 2.5 m^3, which is only a part of the entire fuselage shell. Whenever the fuselage shell position changes with respect to the robot, a local registration of the reachable

region of the fuselage shell is needed. This can be achieved by registering the point cloud from four measurements in the corners of the inspection volume. This approach of reusing the inspection sensor saves e.g. further wide-angle cameras for tracking like in [1]. Besides the skin, we need to capture stringers and frames in order to bind all 6 degrees of freedom. There are two challenges: Well suited measurement positions binding all 6 degrees of freedom need to be generated automatically. The point cloud registration to the CAD model, representing mostly metal sheet parts, should not assign points to false faces that cannot be reached by the sensor.

Because of its own weight, the fuselage shell, which is held by the processing fixture, may be slightly deformed compared to the CAD model. That is why an additional local registration of single inspection measurements may be necessary.

Summarizing, our contributions presented in the sequel are:

- A solution for a viewpoint selection problem using rendering-based measurement simulation and an objective function including analysis of the surface normal distribution and
- a fast point cloud to CAD model (triangle mesh) registration using acquisition direction and surface orientation in order to avoid finding the closest point on a surface not visible for the sensor.

Related work

Survey [2] gives an overview of early work in the domain of CAD-based inspection. The papers [3, 4] discuss methods to compare camera images to rendered CAD data.

It appears natural to combine this CAD based approach with flexible robot guided image [5] and point cloud [1] acquisition by choosing viewpoints which allow to distinguish assembly errors from correct assemblies best. For reviews on viewpoint selection see [6, 7]. A recent technique to find good viewpoints is to simulate the measurement and feed it to an objective function, which measures the “goodness”. An optimization algorithm samples over the feasible viewpoints, see e.g. [5] for rendering and [8] for ray tracing.

The registration between a measured point cloud and a corresponding CAD model is usually performed with the Iterative Closest Point (ICP) algorithm [9, 10]. This algorithms is based on the detection of nearest neighbors of the points on the CAD model and the subsequent computation of a rigid transformation [11], which reduces the distances between the correspondences. This two-step process is repeated until it converges to a local optimum. In order to perform the ICP efficiently, a fast, spatial filtering technique for the point cloud is required to accelerate the costly nearest neighbor searches. The most common data structures for this purpose are the Octree [12, 13] and the k-d tree [14, 15]. Both employ a recursive, spatial subdivision method to significantly reduce the number of candidates for the nearest neighbor. Our work focuses on the k-d tree, since it enables efficient nearest neighbor searches [16] and can be implemented using a favorable memory layout [17], which is utilized in our approach.

Numerous extensions to the original ICP algorithm exist, which improve the matching procedure and increase the chance of reaching the global optimum. The correspondence detection can be extended from simple geometric distances to more sophisticated compatibility measurements [18], that take additional point cloud features into account. Common choices are intensity values obtained from 3D-scanners [19], colors [20], or surface features and contour lines [21]. Taking the normal vectors of the sampled surface into consideration helps to avoid local optima, where local surface orientations are not compatible, see [22].

Sensor simulation and view point selection

The global sensor registration is based on matching measured 3D points towards a CAD model of the given assembly. As the typically used ICP algorithm is vulnerable to local minima, the acquired 3D points must cover all six degrees of freedom in order to reach a global maximum. Thus, for the use of automatic inspection systems it is important to know well suited target locations that should be used to acquire registration data. One can see easily that a maximum spread of 3D point locations within the measuring range is not sufficient. Instead the underlying surface orientation has a major impact on the ICP algorithm result.

Therefore, we propose a new optimization technique to automatically find good sensor poses, which selects poses such that resulting 3D points are acquired from the most orthogonal surfaces around a target location. We call the evaluation criterium *orientation coverage*. It denotes the ability to cover all six degrees of freedom within a 3D point cloud.

This method uses a simulation which was already introduced in [23]. Based on sensor parameters and an assembly CAD model, the simulation framework is able to compute an

approximately expected 3D point cloud generated by a fringe projection system (see Fig. 2). For a given target coordinate on the surface of an assembly, we wish to compute a sensor pose that maximizes the orientation coverage. The algorithm follows a generate and test strategy [24] and performs the following steps:

1. Generate equally distributed poses on a sphere using a radius equal to the sensor working distance.
2. Eliminate those poses that cannot be reached by the sensor due to robot movement limitations.
3. Eliminate those poses which would cause collision between sensor and assembly or robot and assembly.
4. For the remaining poses perform a sensor simulation and evaluate the orientation coverage.
5. Select the pose resulting in the highest coverage.

The number of initially generated poses depends very much on the assembly complexity. In the airplane shell scenario, we started using 4.000 poses which were typically reduced to a third, caused by collisions and robot limitations.

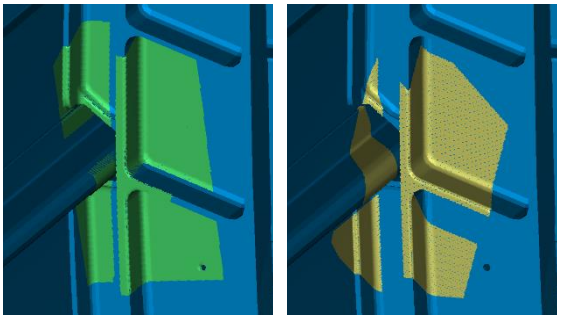

Fig. 2: Simulated point clouds. Left: a poor simulation covering only two space axes, right: more sophisticated points, covering all three axes.

To compute the *orientation coverage*, we use the surface normals within a simulated 3D view. Given a set of simulated 3D points $\boldsymbol{P}$, we call $\boldsymbol{N} = \{\boldsymbol{n}_1 \dots \boldsymbol{n}_m\}$ the set of normals that correspond to each coordinate of $\boldsymbol{P}$. The normal can be derived from the CAD model during simulation. By visualizing $\boldsymbol{N}$ as coordinates on a unit sphere, we see a characteristic distribution depending on the scene geometry (see Fig. 3).

Furthermore, the set $\widetilde{\boldsymbol{N}} = \{\boldsymbol{n}_1 \dots \boldsymbol{n}_m, -\boldsymbol{n}_1 \dots -\boldsymbol{n}_m\}$ describes the set $\boldsymbol{N}$ united with the inverse of $\boldsymbol{N}$. Thus, using a principal component analysis (PCA), we can estimate the normal principal axes and their variance. As the mean of $\widetilde{\boldsymbol{N}}$ is $(0,0,0)^T$ the variance indicates the density of normals with respect to each axis.

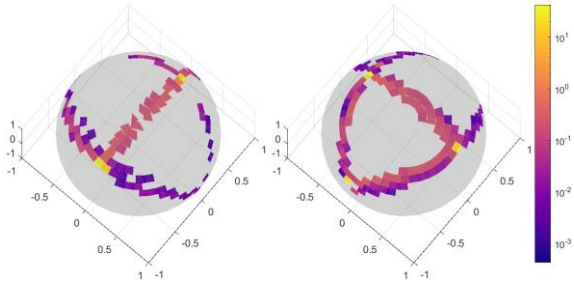

Fig. 3: Density of binned simulated surface normals on a unit sphere: left and right results from point clouds shown in Fig. 2.

Given $\widetilde{\boldsymbol{N}}$, the covariance matrix $\Sigma_{\widetilde{\boldsymbol{N}}}$ is defined as

$$\Sigma_{\widetilde{\boldsymbol{N}}} = \begin{bmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{bmatrix}. \quad (1)$$

The Eigenvalues λ_k of $\Sigma_{\widetilde{\boldsymbol{N}}}$ can be computed by solving

$$(\Sigma_{\widetilde{\boldsymbol{N}}} - \lambda_{\mathrm{k}}\boldsymbol{I})\boldsymbol{v}_{\mathrm{k}} = 0, \quad \mathrm{k} = 1 \dots 3. \quad (2)$$

Furthermore, let $F(\widetilde{\boldsymbol{N}})$ be a function which selects the smallest Eigenvalue of $\Sigma_{\widetilde{\boldsymbol{N}}}$:

$$\mathrm{F}(\widetilde{\boldsymbol{N}}) = \min \mathrm{eig}(\Sigma_{\widetilde{\boldsymbol{N}}}). \quad (3)$$

The smallest Eigenvalue is used as an indicator for the orientation coverage. Fig. 4 visualizes $\mathrm{F}(\widetilde{\boldsymbol{N}})$ for a set of sensor pose candidates:

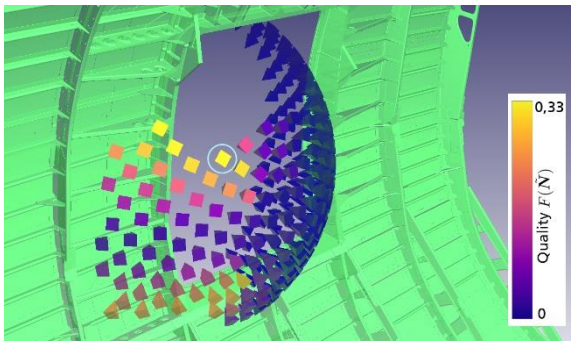

Fig. 4: Thinned result of a generate and test simulation. Evaluated sensors positions are shown as pyramids directed to a target on the surface. The color encoding visualizes each test quality. The best pose is highlighted.

Given sets of simulated normals from different sensor poses $\boldsymbol{N_i}, i = 1 \dots n$, we finally select the dataset of index i^* that maximizes $F(\widetilde{\boldsymbol{N}}_{\boldsymbol{i}})$:

$$i^* = \arg\max_i F(\widetilde{\boldsymbol{N}}_{\boldsymbol{i}}). \quad (4)$$

Based on the candidates shown in Fig. 4, the automatically selected sensor pose is visualized in Fig. 5.

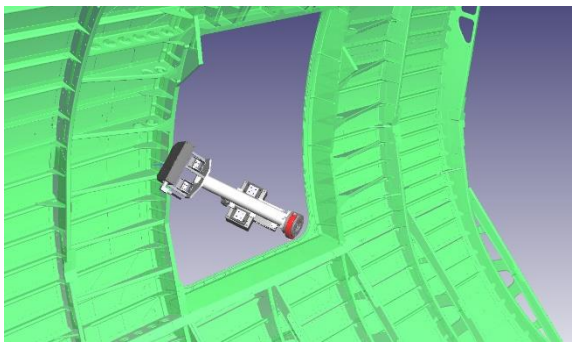

Fig. 5: A final sensor position, which is selected by using the best orientation coverage.

This method is very independent of the assembly shape itself and can be applied to arbitrary target points on the object surface. Fig. 6 shows the results of a systematic sampling of an airplane shell at distances of 100 mm. As expected the resulting qualities between the frames are extremely low, as the geometry in those regions is almost completely flat. However, in more complex areas and especially at stringer frame crossings, the expected quality reaches its maximum.

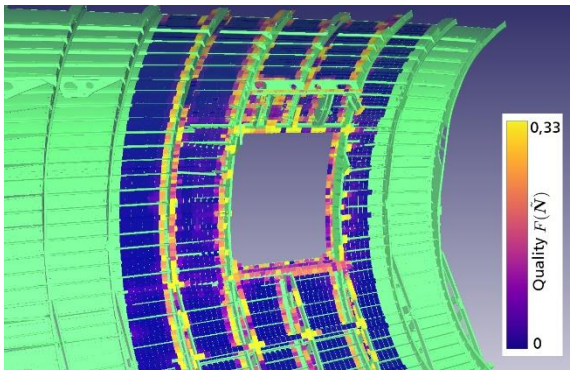

Fig. 6: Systematic surface sampling and evaluation of referencing areas. Flat areas result in low qualities whereas strongly structured geometries provide almost ideal results.

Point cloud registration

The measured point clouds are recorded in the sensor coordinate system and need to be transformed and aligned to the aircraft CAD model in order to analyze them. Usually this is done by computing a registration using the ICP algorithm. However, a direct application of this strategy is not feasible in our use case.

Since the aircraft shell is very thin the ICP, which is based on iterative nearest neighbor detections, could assign a point of the measured point cloud to the backside of the CAD model. This could result in a transformation that places the entire scan on the backside of the model, which complicates further analysis steps. Secondly, the CAD model of the aircraft does not represent the real-world situation accurately. We found out that the shell is slightly deformed, due to its own weight and supporting mounting frame. Furthermore, brackets or other parts of the shell might be missing or are mounted incorrectly, which also distracts the ICP.

To address these challenges, we subdivide the registration process into two parts: In the first step a coarse registration is computed, that aligns measurements at the corners of the inspection area with the aircraft shell as a whole. The computed transformation is used as an initial alignment afterwards, which is refined by the second registration step. This step aligns each scan individually to its local surroundings and handles the challenges created by missing or misplaced parts, while the first step ignores these issues.

Different strategies can be applied to solve the thin shell challenge. One could remove all triangles facing in the opposite direction prior to the registration. However, this approach is not feasible on larger scales, as it requires a view dependent adaption of the triangle mesh, which results in a much more complex data handling. We use a modified ICP variant, that incorporates information from normal vectors into its nearest neighbor search procedure. Only neighboring triangles, for which the scalar product between their normal vector and the one of the search point is larger than a threshold, are considered during the registration. Therefore, measurement points cannot find neighbors on the backside of the shell. This strategy only works reliably if the initial alignment of the two objects does not contain very large deviations.

Since this normal compatible alignment strategy is performed in both registration steps, computing the normal vectors of a point cloud is a vital step of the processing pipeline that should be performed with high speed. The common naïve approach (see [25]) did not fit these requirements. We therefore propose a new method for computing normal vectors of point clouds in an approximation fashion that results in low latencies while preserving enough quality to not interfere with subsequent steps.

The basic idea of our method is to extract a density dependent subset of the whole point cloud and only compute normal vectors for this subset. Afterwards, the normal vectors of this set are distributed to their local neighborhood. As a result, small patches of the point cloud will share the same normal vectors. If these patches are small enough, the subsequent steps are not or only imperceptibly impaired.

Our method is based on a k-d tree, whose structure was presented in detail in [17]. To be most efficient, our method utilizes the specific node types of the tree and their memory layout, which are shown in Fig. 7.

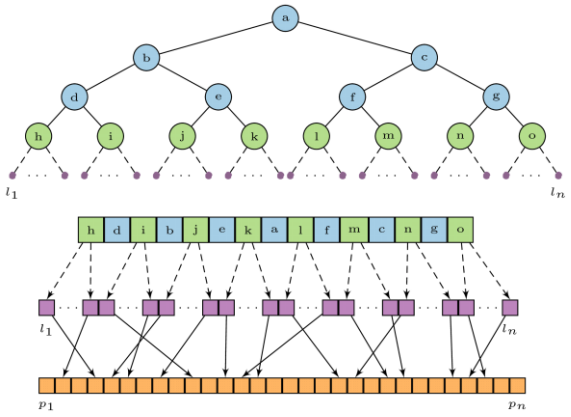

Fig. 7: Top: tree structure and node types (internal blue, pre-leaf green, leaf purple), bottom: memory layout with 3 arrays (internal and pre-leaf nodes, leaf nodes and points)

The tree consists of internal, pre-leaf and leaf nodes, whereby the internal and pre-leaf nodes are stored contiguously in an array in their in-order traversal order. The leaf nodes and actual points are stored in separate arrays.

We use the internal nodes of a specific tree level as the subset of points, for which the normal vectors are computed. This selection satisfies all requirements: Since the underlying k-d tree is constructed using a recursive median of the longest axis splitting method, the corresponding points are spread across the whole point cloud and their distribution reflects the local densities. The size of the subset grows quadratically with each tree level, so the degree of approximation can be tuned with a simple parameter.

After collecting the subset, its k-nearest neighbors of each point of the set are found. The k-d tree is used to perform this search efficiently. Afterwards a plane is fitted into the neighbor set using a least squares approximation method. The normal vector of the plane is used as the point normal.

The normal vectors of the subset are distributed to all of their children as shown in Fig. 8. Due to the special memory layout of the k-d tree, collecting the relevant child nodes can be performed very efficiently. For each subset point the leftmost leaf node of the leftmost child and the rightmost leaf node of the rightmost child are determined. These leaf nodes form a contiguous range, with all leaf nodes in between belonging to the original subset point. Traversing the tree to determine these leaf nodes isn't really required, since the leftmost and rightmost internal node can be determined by subtracting or adding an offset in the internal nodes array. Furthermore, the required computations can be performed in parallel since all subset points and their children are completely independent from another.

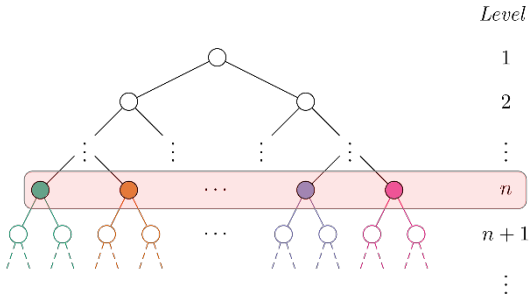

Fig. 8: Approximated normal vector distribution

After the normal vectors have been computed and distributed, a consistent alignment must be ensured, since the least squares plane fitting results in inconsistent normal vector directions. In some scenarios this inconsistency might be ignored, but our workflow relies on correctly oriented normal vectors to perform the described modified ICP algorithm. Different approaches exist to solve this issue, either by modifying the underlying eigenvector computation directly like in [26, 27], or by trying to recreate consistent alignment afterwards by propagating aligned directions through the point cloud [25, 28, 29]. However, our scenario allows to employ a simpler and faster solution. The points are measured in sensor coordinates and only points located on surfaces pointing towards the sensor have been generated. Therefore, a simple dot product test between the computed normal vector and the direction vector towards the sensor origin is enough to realign the normal vectors consistently across the point cloud.

The first coarse alignment step uses the approximated normal vectors to perform the modified ICP. It first collects the corner measurements and combines them into a single point cloud, taking the information about their corresponding robot orientation into account to preserve their relative positions and orientations. This cloud is registered against the CAD model as a whole, supported by an initial guess of its orientation. The modified ICP ensures that the cloud is aligned to the front side of the model.

Again, a k-d tree is used to speed up the nearest neighbor searches, performed by the ICP. The k-d tree uses the same structure as described before, but stores triangles instead. We found that a preprocessing of the triangle mesh of the CAD model was necessary to achieve acceptable processing speeds. The CAD model contains very long triangles, that can span

almost the entire length of the aircraft shell. These elongated triangles prevent the k-d tree from creating tight bounding volumes, which otherwise account for the fast neighbor detection. Therefore, we subdivide all triangles recursively until all edges are shorter than a threshold value of 200 mm. This increases the total number of triangles by a significant amount, but nevertheless leads to an improved performance due to tighter bounding volumes.

After performing the coarse registration there are two main sources contributing to the remaining alignment error. Due to the uncertainty of the robot positions, the relative positions of the corner measurements contain a small error. The second is the deviation between CAD model and the deformed real aircraft shell, which cannot be eliminated as the cloud is transformed as a whole. The measurement uncertainty of the 3D sensor of about 50 µm can be neglected.

To reduce the remaining error, a second alignment step is performed. This fine registration is applied to each scan individually, to account for local conditions. Again, the ICP with the compatible normal nearest neighbor search is employed, to ensure that the current alignment to the front side of the aircraft shell is retained. In contrast to the coarse registration, the fine registration is performed against a simulated point cloud. Using an artificial reference cloud has several advantages. The nearest neighbor searches can be performed at higher speeds, as only a relatively small point cloud has to be searched instead of the triangle mesh of the whole CAD model. Secondly the registration is focused on the relevant parts of the model, which leaves fewer opportunities for misalignments.

Only the so-called background of the scene is simulated. No points are simulated on brackets or other parts, whose existence and placement should be checked in a following analysis. Without this exclusion the ICP could transform points of a misplaced bracket to its actual position on the CAD model. A following analysis then might overlook this assembly error due to its close alignment. Computing a registration against the background implies that a maximum distance for the nearest neighbor search of the ICP has to be defined, so only background points of the measurement are used during the alignment. Since the coarse registration ensures a decent initial alignment, enough background points are close to the CAD model and a tight threshold can be chosen.

Results

Our new method for computing approximated normal vectors can be applied to any kind of point cloud, therefore we tested it on generic data sets as well as measurements from the aircraft scenario. The results are shown in Fig. 9 and Fig. 10. The blue line depicts the runtime, displayed is the median over 10 runs. Each step of the x-axis corresponds to a specific level of nodes in the k-d tree, that results in a distribution of one computed normal to 2^x points. The red line shows the median of the overall normal quality, measured as dot product between the ground-truth normal and the approximated one. The shaded red areas display quantile ranges of the normal quality.

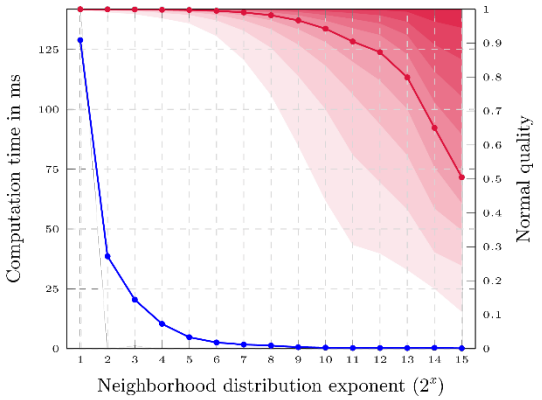

Fig. 9: Results of the approximated normal vectors computation for a generic point cloud; runtime in blue; quality of normal vectors in red; red areas show quantile boundaries in 10% steps

The runtime decreases almost quadratically with each level. The large reduction between the first two steps is explained by an implementation artifact, as the approximation is only used when a normal is distributed to more than two points. The first measurement therefore represents the base runtime with no approximation.

The quality of normal vectors remains very high and stable for the first exponents. Up to an exponent of 6, which corresponds to a distribution of one normal to 128 points, more than 90% of the points have a normal quality of 0.9 or more. The distribution patches remain small and localized enough to not impair the overall normal quality significantly. Afterwards the quality starts to drop faster with each level. The aircraft shell showed even better results, since more points retain a good normal quality for higher exponents. This is explained by the large number of flat or gently curved parts of the shell, in which a normal is very similar to its neighboring ones. Only if the distribution patches cover larger surface areas, the normal quality starts to drop significantly.

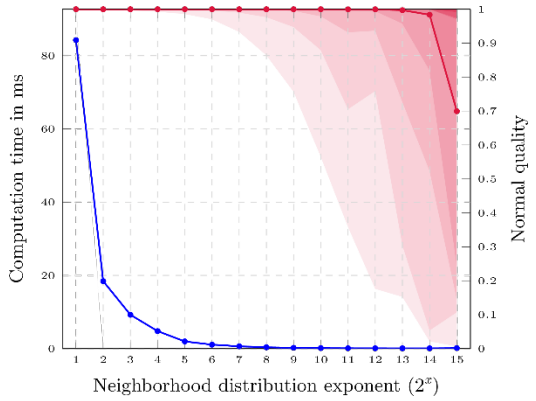

Fig. 10: Results of the approximated normal vectors computations for a point cloud from the airplane shell

In summary, choosing an exponent of four or five seems to represent a good tradeoff between high processing speeds, with reductions to 8% resp. 4% of the original runtime, and preserving a high quality of the resulting normal vectors.

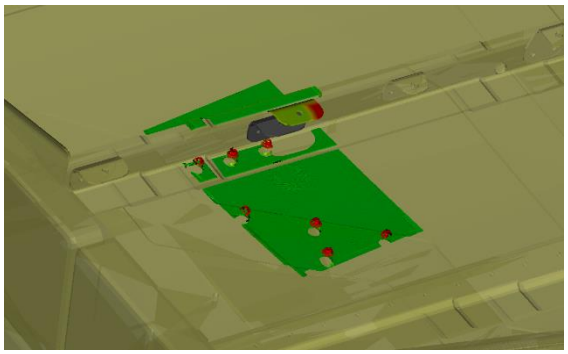

Fig. 11: Fine registration result with a misplaced bracket, colors represent distances to CAD model

The two-step registration process results in stable and close alignments of the measured point clouds with the CAD model. Due to the faster normal vector computation the overall registration time can be reduced, too. Deviations of the scans from the real CAD model are distributed across all scans.

The fine registration is able to eliminate the remaining deviations and aligns the scans very closely to the model. Especially the registration against a simulated background point cloud proves to be beneficial. Fig. 11 shows the alignment result of a misplaced bracket. A registration against the whole mesh would have resulted in a translation that places the bracket at is actual model position, making it challenging to detect this assembly error afterwards.

Summary

In this article we presented a novel, innovative approach for the automatic registration of 3D point clouds in challenging inspection tasks. Therefore, we use a CAD model based, two step registration. In the first step, we compute a coarse registration which estimates a 3D sensor location roughly. Suitable sensor poses for the required data acquisition are computed in preface by simulation and automatic view point selection. In the second step, a fine registration is performed that accounts for local deviations from the CAD model and improves the prior alignment. Both registration computations use a modified ICP algorithm, which registers measured 3D point clouds to simulated point clouds efficiently.

References

[1] H.B. Abdallah, J.-J. Orteu, B. Dolives, and I. Jovančević, 3D point cloud analysis for automatic inspection of aeronautical mechanical assemblies, *Fourteenth International Conference on Quality Control by Artificial Vision*, 209–217 (2019), doi: 10.1117/12.2521715.

[2] T.S. Newman and A.K. Jain, A Survey of Automated Visual Inspection, *Computer Vision and Image Understanding* 61 (2), 231–262 (1995), doi: 10.1006/cviu.1995.1017.

[3] I. Viana, J.-J. Orteu, N. Cornille, and F. Bugarin, Inspection of aeronautical mechanical parts with a pan-tilt-zoom camera: an approach guided by the computer-aided design model, *J. Electron. Imaging* 24 (6), 61118 (2015), doi: 10.1117/1.JEI.24.6.061118.

[4] S. Sauer, E. Trostmann, D. Berndt, and G. Holtmann, Optische Messsimulation zur flexiblen, automatischen Montageprüfung, *19. Anwendungsbezogener Workshop zur Erfassung, Modellierung, Verarbeitung und Auswertung von 3D-Daten (3D-NordOst 2016)*, 123–132 (2016).

[5] H. Ben Abdallah, I. Jovančević, J.-J. Orteu, and L. Brèthes, Automatic Inspection of Aeronautical Mechanical Assemblies by Matching the 3D CAD Model and Real 2D Images, *Journal of Imaging* 5 (10), 81 (2019), doi: 10.3390/jimaging5100081.

[6] S. Chen, Y. Li, and N.M. Kwok, Active vision in robotic systems: A survey of recent developments, *The International Journal of Robotics Research* 30 (11), 1343–1377 (2011), doi: 10.1177/0278364911410755.

[7] R. Zeng, Y. Wen, W. Zhao, and Y.-J. Liu, View planning in robot active vision: A survey of systems, algorithms, and applications, *Comp. Visual Media* 6 (3), 225–245 (2020), doi: 10.1007/s41095-020-0179-3.

[8] T. Kot, Z. Bobovský, D. Heczko, A. Vysocký, I. Virgala, and E. Prada, Using Virtual Scanning to Find Optimal Configuration of a 3D Scanner Turntable for Scanning of Mechanical Parts, *Sensors (Basel, Switzerland)* 21 (16), 5343 (2021), doi: 10.3390/s21165343.

[9] P.J. Besl and N.D. McKay, Method for registration of 3-D shapes, *Sensor fusion IV: control paradigms and data structures*, 586–606 (1992), doi: 10.1117/12.57955.

[10] Y. Chen and G. Medioni, Object modelling by registration of multiple range images, *Image and Vision Computing* 10 (3), 145–155 (1992), doi: 10.1016/0262-8856(92)90066-C.

[11] W. Kabsch, A solution for the best rotation to relate two sets of vectors, *Acta Cryst A* 32 (5),

922–923 (1976), doi: 10.1107/S0567739476001873.
[12] D. Meagher, Geometric modeling using octree encoding, *Computer Graphics and Image Processing* 19 (1), 85 (1982), doi: 10.1016/0146-664x(82)90128-9.
[13] M. Schütz, S. Ohrhallinger, and M. Wimmer, Fast Out-of-Core Octree Generation for Massive Point Clouds, *Computer Graphics Forum* 39 (7), 155–167 (2020), doi: 10.1111/cgf.14134.
[14] J.L. Bentley, Multidimensional binary search trees used for associative searching, *Commun. ACM* 18 (9), 509–517 (1975), doi: 10.1145/361002.361007.
[15] R.A. Brown, Building a Balanced k-d Tree in O(kn log n) Time, *Journal of Computer Graphics Techniques (JCGT)* 4 (1), 50–68 (2015), doi: 10.48550/arXiv.1410.5420.
[16] J.H. Friedman, J.L. Bentley, and R.A. Finkel, An Algorithm for Finding Best Matches in Logarithmic Expected Time, *ACM Trans. Math. Softw.* 3 (3), 209–226 (1977), doi: 10.1145/355744.355745.
[17] D. Sopauschke, C. Teutsch, E. Trostmann, and D. Berndt, A parallel memory efficient outlier detection algorithm for large unstructured point clouds, *Optical Measurement Systems for Industrial Inspection XII*. *21-25 June 2021, online only, Germany*, 68 (2021), doi: 10.1117/12.2592299.
[18] S. Rusinkiewicz and M. Levoy, Efficient variants of the ICP algorithm, *Proceedings Third International Conference on 3-D Digital Imaging and Modeling* (null), doi: 10.1109/im.2001.924423.
[19] G. Godin, M. Rioux, and R. Baribeau, Three-dimensional registration using range and intensity information, *Videometrics III*, 279–290 (1994), doi: 10.1117/12.189139.
[20] M. Korn, M. Holzkothen, and J. Pauli, Color Supported Generalized-ICP, *Proceedings of the 9th International Conference on Computer Vision Theory and Applications* (2014), doi: 10.5220/0004692805920599.
[21] J. Fan, L. Ma, and Z. Zou, A method of registering point cloud with CAD model based on line contour matching, *Optics and Photonics for Information Processing XIV*, 55–64 (2020), doi: 10.1117/12.2570494.
[22] K. Pulli, Multiview registration for large data sets, *Proceedings / Second International Conference on 3-D Digital Imaging and Modeling, October 4 - 8, 1999, Ottawa, Ontario, Canada*, 160–168 (1999), doi: 10.1109/IM.1999.805346.
[23] S. Sauer, T. Dunker, and M. Heizmann, Ein Framework zur Simulation optischer Sensoren, *20. GMA/ITG-Fachtagung*. *Sensoren und Messsysteme 2019*, 35–42 (2019), doi: 10.5162/sensoren2019/1.1.1.
[24] K.W. Khawaja, A.A. Maciejewski, D. Tretter, and C.A. Bouman, Camera and light placement for automated assembly inspection, *Proceedings 1996 IEEE International Conference on Robotics and Automation*, 3246–3252 (1996), doi: 10.1109/ROBOT.1996.509207.
[25] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, Surface reconstruction from unorganized points, *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, 71–78 (1992), doi: 10.1145/133994.134011.
[26] R. Bro, E. Acar, and T.G. Kolda, Resolving the sign ambiguity in the singular value decomposition, *J. Chemometrics* 22 (2), 135–140 (2008), doi: 10.1002/cem.1122.
[27] F. Tombari, S. Salti, and L. Di Stefano, Unique Signatures of Histograms for Local Surface Description, 356–369 (2010), doi: 10.1007/978-3-642-15558-1_26.
[28] H. Xie, J. Wang, J. Hua, H. Qin, and A. Kaufman, Piecewise C/sup 1/ continuous surface reconstruction of noisy point clouds via local implicit quadric regression, *IEEE Visualization 2003*. *Seattle, Washington, October 19 - 24, 2003; VIS 2003; proceedings*, 91–98 (2003), doi: 10.1109/VISUAL.2003.1250359.
[29] N. Schertler, B. Savchynskyy, and S. Gumhold, Towards Globally Optimal Normal Orientations for Large Point Clouds, *Computer Graphics Forum* 36 (1), 197–208 (2017), doi: 10.1111/cgf.12795.