A SENSOR SKID FOR PRECISE 3D MODELING OF PRODUCTION LINES

Jan Elseberg, Dorit Borrmann[a], Johannes Schauer[b], Andreas Nüchter[a,*], Dirk Koriath[c], and Ulrich Rautenberg[c]

[a] Informatics VII : Robotics and Telematics, Julius-Maximilians-University Würzburg,
Am Hubland, Wüzburg 97074, Germany
andreas@nuechti.de, http://www.nuechti.de

[b] School of Engineering and Science, Automation Group, Jacobs University of Bremen gGmbH,
Campus Ring 1, Bremen 28759, Germany

[c] K-EFP/V group at Volkswagen AG
Wolfsburg 38436, Germany

ABSTRACT:

Motivated by the increasing need of rapid characterization of environments in 3D, we designed and built a sensor skid that automates the work of an operator of terrestrial laser scanners. The system combines terrestrial laser scanning with kinematic laser scanning and uses a novel semi-rigid SLAM method. It enables us to digitize factory environments without the need to stop production. The acquired 3D point clouds are precise and suitable to detect objects that collide with items moved along the production line.

1 INTRODUCTION

The varieties of manufactured cars constantly increases. For example in Europe of the 1950s Volkswagen produced the Volkswagen Beetle, Karmann-Ghia Typ 14, and the Volkswagen Type 2. By 1990 this changed to the Volkswagen Polo Mk2, Golf Mk2, Jetta Mk2, Passat B3, Scirocco, Corrado, T3, Caddy, Golf Country, and Taro. Nowadays, Volkswagen produces 17 different models: up!, Polo Mk5, Beetle, Golf Mk7, Golf Variant, Golf Cabriolet, Jetta Mk6, Passat B7, CC, Phaeton, Scirocco, EOS, Golf Plus, Touran, Sharan II, Tiguan, Tuareg II. On a global scale the corporate group nowadays produces 280 models. This increase requires that the production facilities are flexible enough to remain competitive. For planning the production lines, a precise digital 3D model is extremely helpful, as it enables one to detect possible collisions of objects in the production line with the surrounding environment.

Terrestrial Light Detection and Ranging (LiDAR) systems have left pre-commercial development and have reached the state of technically mature systems. Unlike several years ago, these systems are made available from a number of vendors. Typical accuracies and precisions of laser scanners are in the range of a few millimeters. Terrestrial laser scanning is the state of the art for digitalization of complex factory environments. When paired with classical surveying, terrestrial LiDAR delivers accurately referenced geo-data, which may be used for topographic purposes. However, many applications exist, for example as-built documentation of industrial plants and factories, ship-building, interior planning, and applications which either do not require the reference to a national grid, or do not require a reference at all.

Terrestrial Laser Scanning (TLS) is a ground based technique to measure the position and dimension of objects in three dimensional space using either pulsed or phase-modulated-continuous laser light. Terrestrial laser scanners use a rotating mirror and a rotating laser head to deflect the laser for measurements. Mobile laser scanning describes terrestrial data acquisition from moving platforms (e.g., boats, trains, road and off-road vehicles) and is also known as kinematic laser scanning. In the mobile scenario, the scanner operates in profiler mode, i.e., they scan a slice of the environment. 3D data is then obtained by moving the system and "unwinding" the obtained measurements.

This paper presents a novel mixture of classical terrestrial and kinematic laser scanning for an industrial inspection application.

*Corresponding author.

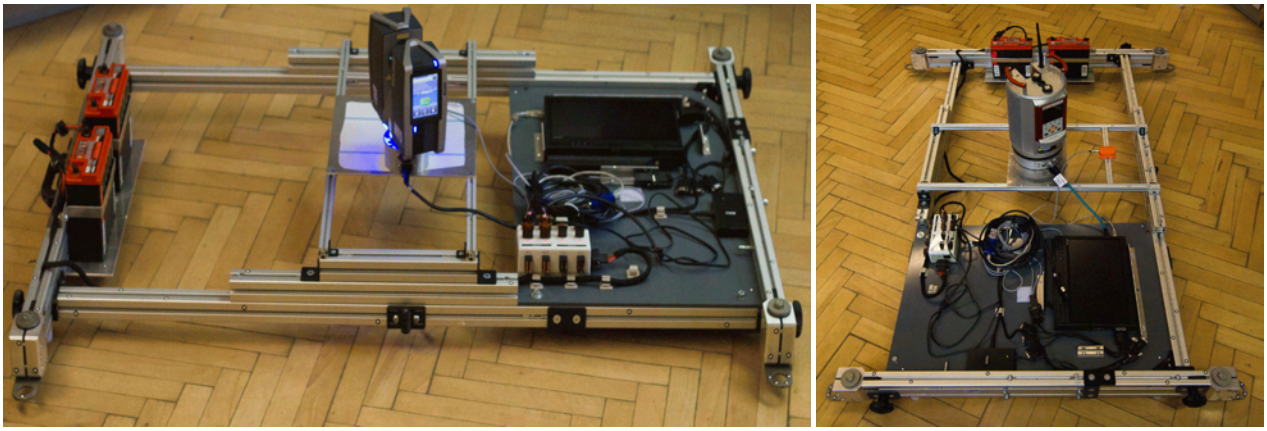

Figure 1: Left: Sensor skid, a.k.a. work-holding fixture, equipped with a FARO Focus^{3D} laser scanner. Right: Skid with a Riegl VZ-400 laser scanner. Both skids are also equipped with a low-price MEMS IMU (xsens MTi).

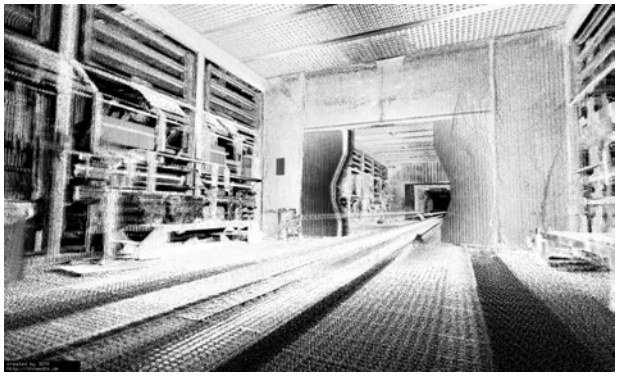

Figure 2: 3D point cloud of the entrance of a dryer tunnel acquired with the FARO Focus^{3D} scanner.

A continuously spinning terrestrial laser scanner is mounted on a skid and moved along a production line, e.g., through a drying chamber. The acquired 3D data is processed with a simultaneous localization and mapping (SLAM) algorithm to obtain precise 3D models. A detailed description of the methods can be found in Elseberg et al. (2013) and Nüchter et al. (2010).

2 CONTINUOUS 3D SCANNING FROM A SENSORSKID

Commonly, conveyor systems are used for carrying skids and supporting products to be assembled along the assembly lines. Skid conveyors are ideally suited for automotive body shops, paint shops and final assembly operations. The base of our system consists of such a skid. We equipped it with a high-precision laser scanner, e.g., FARO Focus^{3D} or Riegl VZ-400, with electronics for connecting either of these scanners to a laptop and with a MEMS IMU (xsens MTi). Fig. 1 shows the setup.

The 3D laser scanner is configured such that the scanner spins around the vertical axis. The minimum rotation rate for the FARO scanner is 1 Hz, for the Riegl scanner it is 0.166 Hz. The field of view (vertical/horizontal) for the FARO Focus^{3D} is $305^\circ \times 360^\circ$, and for the Riegl VZ-400 is $100^\circ \times 360^\circ$. The system is calibrated, i.e., the position and orientation of the sensor is determined on the skid. We acquire time-stamped data while moving through the production line.

3 SEMI-RIGID OPTIMIZATION FOR SLAM

We developed an algorithm that improves the entire trajectory of the skid simultaneously. Unlike previous algorithms, e.g., by Stoyanov and Lilienthal (2009) and Bosse and Zlot (2009), it is not restricted to purely local improvements. We make no rigidity assumptions, except for the computation of the point correspondences. We require no explicit motion model although such information may be incorporated at no additional cost. The algorithm requires no high-level feature computation, i.e., we require only the 3D points themselves.

The motion of the mobile laser scanner between time t_0 and t_n creates a trajectory $T = \{\mathbf{V}_0, \ldots, \mathbf{V}_n\}$, where $\mathbf{V}_i = (t_{x,i}, t_{y,i}, t_{z,i}, \theta_{x,i}, \theta_{y,i}, \theta_{z,i})$ is the 6 degree of freedom (DoF) pose of the skid at time t_i with $t_0 \leq t_i \leq t_n$. Using the trajectory of the skid, a 3D representation of the environment can be obtained by "unwinding" the laser measurements M to create the final map P. However, inaccuracies in the pose estimate which is initially computed using the known velocity estimate, as well as systematic calibration errors degrade the accuracy of the trajectory and therefore the point cloud quality.

The current state of the art developed by Bosse and Zlot (2009) for improving overall map quality of mobile mappers in the robotics community is to coarsely discretize the time. This results in a partition of the trajectory into subscans that are treated rigidly. Then rigid registration algorithms like the ICP and other solutions to the SLAM problem are employed. Obviously, trajectory errors within a subscan cannot be improved in this fashion. Applying rigid pose estimation to this non-rigid problem is also problematic since rigid transformations can only approximate the underlying ground truth. Consequently, overall map quality suffers as a result.

For the sensor skid a fine discretization of the time is employed, i.e., at the level of individual 2D scan slices or individual points. This results in the set of measurements $M = \{\mathbf{m}_0, \ldots, \mathbf{m}_n\}$ where $\mathbf{m}_i = (m_{x,i}, m_{y,i}, m_{z,i})$ is a point acquired at time t_i in the local coordinate system of $\mathbf{V}_i$. In case $\mathbf{V}_i$ represents more than a single point, $\mathbf{V}_i$ is the local coordinate of the first point. All represented points are motion compensated with the best known interpolated trajectory. As modern laser scanners typically operate at a frequency of 100–200 Hz time is discretized to 5–10 ms. Applying the pose transformation $\oplus$ we derive the point $\mathbf{p}_i = \mathbf{V}_i \oplus \mathbf{m}_i = \mathbf{R}_{\theta_{x,i},\theta_{y,i},\theta_{z,i}}\mathbf{m}_i + (t_{x,s}, t_{y,s}, t_{z,s})^T$ in the global coordinate frame and thereby also the map $P = \{\mathbf{p}_0, \ldots, \mathbf{p}_n\}$. Here, $\mathbf{R}_{\theta_{x,i},\theta_{y,i},\theta_{z,i}}$ is the rotation matrix that is defined by Eq. (1).

Given M and T we find a new trajectory $T' = \{\mathbf{V}'_1, \ldots, \mathbf{V}'_n\}$ with modified poses so that P generated via T' more closely resembles the real environment.

The complete semi-rigid registration algorithm proceeds as follows: Given a trajectory estimate, we compute the point cloud P in the global coordinate system and use nearest neighbor search to establish correspondences. Then, after computing the estimates of pose differences and their respective covariances we optimize the trajectory T. This process is iterated until convergence, i.e., until the change in the trajectory falls below a threshold.

To deal with the massive amount of data in reasonable time, we first reduce uniformly and randomly the point cloud by using only a constant number of points per volume, typically to 1 point per 3 cm^3. A compact octree data structure is ideally suited for this type of subsampling and for the computation of the nearest neighbors Elseberg et al. (2012). The octree also compresses the point cloud so it can be easily stored and processed. Furthermore, in initial stages of the algorithm, estimates $\bar{\mathbf{V}}_{i,j}$ are only computed for a subset of poses $\mathbf{V}_0, \mathbf{V}_k, \mathbf{V}_{2k}, \ldots$, with k in the order of hundreds of milliseconds. In every iteration k is decreased so that the trajectory is optimized with a finer scale.

3.1 Pose Estimation

Our algorithm incorporates pose estimations from a rough guess of the velocity of the conveyor. Next, we use the formulation of pose estimates $\bar{\mathbf{V}}_{i,i+1}$ that are equivalent to pose differences:

$$\bar{\mathbf{V}}_{i,i+1} = \bar{\mathbf{V}}_i \ominus \bar{\mathbf{V}}_j. \tag{4}$$

The operator $\ominus$ is the inverse of the pose compounding operator $\oplus$, such that $\mathbf{V}_j = \mathbf{V}_i \oplus (\mathbf{V}_i \ominus \mathbf{V}_j)$. In addition to the default pose estimates that may also be enhanced by separating all pose sensors into their own estimates as well as the proper covariances, we estimate differences between poses via the point cloud P.

For each measurement $\mathbf{p}_i$, we find a closest measurement $\mathbf{p}_j$ via nearest neighbor search with $|t_i - t_j| > \delta$, where δ is again the

$$\mathbf{R}_{\theta_{x,i},\theta_{y,i},\theta_{z,i}} = \begin{pmatrix} \cos\theta_y \cos\theta_{z,i} & -\cos\theta_{y,i}\sin\theta_{z,i} & \sin\theta_{y,i} \\ \cos\theta_{z,i}\sin\theta_{x,i}\sin\theta_{y,i} + \cos\theta_{x,i}\sin\theta_{z,i} & \cos\theta_{x,i}\cos\theta_{z,i} - \sin\theta_{x,i}\sin\theta_{y,i}\sin\theta_{z,i} & -\cos\theta_{y,i}\sin\theta_{x,i} \\ \sin\theta_{x,i}\sin\theta_{z,i} - \cos\theta_{x,i}\cos\theta_{z,i}\sin\theta_{y,i} & \cos\theta_{z,i}\sin\theta_{x,i} + \cos\theta_{x,i}\sin\theta_{y,i}\sin\theta_{z,i} & \cos\theta_{x,i}\cos\theta_{y,i} \end{pmatrix} \tag{1}$$

$$\mathbf{M}_k = \begin{pmatrix} 1 & 0 & 0 & 0 & -m_{y,k} & -m_{z,k} \\ 0 & 1 & 0 & m_{z,k} & m_{x,k} & 0 \\ 0 & 0 & 1 & -m_{y,k} & 0 & m_{x,k} \end{pmatrix} \tag{2}$$

$$\mathbf{H}_i = \begin{pmatrix} 1 & 0 & 0 & 0 & \bar{t}_{z,i}\cos(\bar{\theta}_{x,i}) + \bar{t}_{y,i}\sin(\bar{\theta}_{x,i}) & \bar{t}_{y,i}\cos(\bar{\theta}_{x,i})\cos(\bar{\theta}_{y,i}) - \bar{t}_{z,i}\cos(\bar{\theta}_{y,i})\sin(\bar{\theta}_{x,i}) \\ 0 & 1 & 0 & -\bar{t}_{z,i} & -\bar{t}_{x,i}\sin(\bar{\theta}_{x,i}) & -\bar{t}_{x,i}\cos(\bar{\theta}_{x,i})\cos(\bar{\theta}_{y,i}) - \bar{t}_{z,i}\sin(\bar{\theta}_{y,i}) \\ 0 & 0 & 1 & \bar{t}_{y,i} & -\bar{t}_{x,i}\cos(\bar{\theta}_{x,i}) & \bar{t}_{x,i}\cos(\bar{\theta}_{y,i})\sin(\bar{\theta}_{x,i}) + \bar{t}_{y,i}\sin(\bar{\theta}_{y,i}) \\ 0 & 0 & 0 & 1 & 0 & \sin(\bar{\theta}_{y,i}) \\ 0 & 0 & 0 & 0 & \sin(\bar{\theta}_{x,i}) & \cos(\bar{\theta}_{x,i})\cos(\bar{\theta}_{y,i}) \\ 0 & 0 & 0 & 0 & \cos(\bar{\theta}_{x,i}) & -\cos(\bar{\theta}_{y,i})\sin(\bar{\theta}_{x,i}) \end{pmatrix}. \tag{3}$$

Figure 3: Definition of the rotation matrix and the matrix decompositions.

Figure 4: Left: Before applying our semi-rigid SLAM method. Right: Correction by our semi-rigid SLAM.

minimal amount of time that must have elapsed for the laser scanner to have measured the same point on the surface again. To this end, we alter our search structure for finding closest points to contain time stamps as well. Points are stored in the global coordinate frame as defined by the estimated trajectory T. Closest points are accepted if $|\mathbf{p}_i - \mathbf{p}_j| \leq d_{\max}$. The positional error of two poses $\mathbf{V}_i$ and $\mathbf{V}_j$ is then given by

$$E_{i,j} = \sum_{k=i-N}^{i+N} \left\| \mathbf{V}_i \oplus \mathbf{m}_k - \mathbf{V}_j \oplus \mathbf{m}'_k \right\|^2 = \sum_{i=1}^{m} \left\| \mathbf{Z}_k(\mathbf{V}_i, \mathbf{V}_j) \right\|^2. \tag{5}$$

Here, the $\mathbf{m}_k, \mathbf{m}'_k$ is the pair of closest points in their respective local coordinate system, and N defines a small neighborhood of points taken in the order of hundreds of milliseconds that is assumed to have negligible pose error. Since we will require a linear approximation of this function and $\oplus$ is decidedly non-linear we apply a Taylor expansion of $\mathbf{Z}_k$:

$$\mathbf{Z}_k(\mathbf{V}_i, \mathbf{V}_j) \approx \mathbf{Z}_k(\bar{\mathbf{V}}_i, \bar{\mathbf{V}}_j) - \left(\nabla_{\mathbf{V}_i} \mathbf{Z}_k(\bar{\mathbf{V}}_i, \bar{\mathbf{V}}_j) \Delta\mathbf{V}_i - \nabla_{\mathbf{V}_j} \mathbf{Z}_k(\bar{\mathbf{V}}_i, \bar{\mathbf{V}}_j) \Delta\mathbf{V}_j \right).$$

Here, $\nabla_{\mathbf{V}_i}$ refers to the derivative with respect to $\mathbf{V}_i$. We create a matrix decomposition $\nabla_{\mathbf{V}_i} \mathbf{Z}_k = \mathbf{M}_k \mathbf{H}$ such that the matrix $\mathbf{M}_k$ is independent of $\mathbf{V}_i$. Then, $\mathbf{Z}_k(\mathbf{V}_i, \mathbf{V}_j)$ is approximated as:

$$\mathbf{Z}_k(\mathbf{V}_i, \mathbf{V}_j) \approx \mathbf{Z}_k(\bar{\mathbf{V}}_i, \bar{\mathbf{V}}_j) - \mathbf{M}_k(\mathbf{H}_i \Delta\mathbf{V}_i - \mathbf{H}_j \Delta\mathbf{V}_j).$$

The minimum of this new linearized formulation of the error metric and its covariance is given by

$$\bar{\mathbf{E}}_{i,j} = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{Z}$$
$$\mathbf{C}_{i,j} = s^2(\mathbf{M}^T\mathbf{M}).$$

Here $\mathbf{Z}$ is the concatenated vector consisting of all $\mathbf{Z}_k = \bar{\mathbf{V}}_i \oplus \mathbf{m}_k - \bar{\mathbf{V}}_j \oplus \mathbf{m}'_k$. s is the unbiased estimate of the covariance of the independent and identically distributed errors of the error metric and is computed as

$$s^2 = (\mathbf{Z} - \mathbf{M}\bar{\mathbf{V}}_{i,j})^T (\mathbf{Z} - \mathbf{M}\bar{\mathbf{V}}_{i,j})/(2M - 3). \tag{6}$$

The matrix decomposition $\mathbf{M}_k\mathbf{H}$ depends on the parameterization of the pose compounding operation $\oplus$. Since we employ the Euler parameterization as given in Eq. 1 the matrices $\mathbf{M}_k$ and $\mathbf{H}_i$ are given by Eq. (2) and (3).

3.2 Pose Optimization

We maximize the likelihood of all pose estimates $\mathbf{V}_{i,j}$ and their respective covariances $\mathbf{C}_{i,j}$ via the Mahalanobis distance

$$W = \sum_i \sum_j (\bar{\mathbf{V}}_{i,j} - (\mathbf{V}'_i - \mathbf{V}'_j)) \mathbf{C}^{-1}_{i,j} (\bar{\mathbf{V}}_{i,j} - (\mathbf{V}'_i - \mathbf{V}'_j)),$$

or, with the incidence matrix $\mathbf{H}$ in matrix notation:

$$W = (\bar{\mathbf{V}} - \mathbf{H}\mathbf{V})^T \mathbf{C}^{-1} (\bar{\mathbf{V}} - \mathbf{H}\mathbf{V}). \tag{7}$$

This linear formulation of the problem has been made possible by the linearization of the pose difference equation in the previous section. The minimization of W is accomplished via solving the

following linear equation system:

$$(\mathbf{H}^T\mathbf{C}^{-1}\mathbf{H})\mathbf{V} = \mathbf{H}^T\mathbf{C}^{-1}\bar{\mathbf{V}}. \quad (8)$$

Computing the optimized trajectory is then reduced to inverting a positive definite matrix Nüchter et al. (2010). Since a significant portion of correspondences i, j are not considered due to the lack of point correspondences the matrix is sparse so that we make use of sparse Cholesky decomposition, Davis (2005). A consequence of this formulation is that an arbitrary pose $\mathbf{V}_i$ must remain fixed, otherwise Eq. 8 would be under determined. This fixed pose also incidentally defines the global coordinate system. Thus, for the algorithm the optimization problem is decoupled from determination of global frame coordinates of the point cloud.

4 EXPERIMENTS AND RESULTS

Experiments have been carried out with both skids from Fig. 1 at two different production lines in Wolfsburg, Germany. Fig. 2 presents an obtained point cloud. The system takes as input the approximate velocity of the skid and extracts an initial point cloud using this velocity. Afterwards, we apply our semi-rigid SLAM to correct the whole trajectory of the system. A typical trajectory consists of more than 10000 skid poses.

For evaluating the accuracy of the 3D data points we have also acquired a reference data set in a stop-scan-go mode. These independently acquired 3D scans have been precisely registered and form the reference model. To this reference we compare the 3D point cloud that has been captured, while the skid is in motion. To this end, both data sets are put in the same coordinate frame and point-to-point distances are calculated. Fig. 5 shows two point clouds: on the left the reference point cloud from registered terrestrial 3D scans and in the middle the point cloud while the sensor skid was in motion. The rightmost part of Fig. 6 shows the deviations, i.e., for every point we compute the deviation from the ground truth model. Please note the grey parts: These are 3D points seen by the continuously rotating scanner, i.e., points on the skid, which have not been removed, and points in the environment which turn out to be occluded by terrestrial scanning at discrete poses.

Fig. 6 shows the distribution of the errors in a histogram (left) and its frequencies. 90% of the measured points have a deviation from the reference model below 2.5 cm. 3D points closer to the scanner have usually even higher precision, as scanning is done in a spherical way.

5 COLLISION DETECTION

After acquiring a consistent 3D point cloud of the environment, we check whether a model moving through the environment collides with it, which points in the environment collide and how deep they penetrate the model. To this end we move a point cloud of the model through the point cloud of the environment along a given trajectory. Fig. 7 shows on the upper left the point cloud of the environment in magenta and the point cloud of the model in yellow. The point cloud of the model can either be acquired by measurements with a laser scanner or by sampling the surface area of a given CAD model. The given model represents a Volkswagen Golf scaled to a width of about 3 m to make it collide with the factory environment.

Any point of the environment that is found to be closer than a radius r to any point of the model at any point of the trajectory is classified as "colliding". All points of the environment which are never found to fulfill this criteria are classified as "non-colliding". Fig. 7 shows on the lower left the colliding point cloud in yellow and the non-colliding point cloud in magenta when moving the car model from the right figure through the environment on a straight trajectory along the assembly line. Bigger radii of r allow to implement "safety distances" for objects moving through the environment. In this example, we chose a radius r of 10 [cm].

After the point cloud of the environment has been partitioned in colliding and non-colliding points, we calculate for all colliding points an estimate of their maximum depth of penetration into the model along its path. For each colliding point we find the closest non-colliding point in the environment and take their distance as a heuristic for the depth of penetration. This heuristic works well for the common cases of objects protruding the volume of an assembly line. Fig. 7 shows in the right column the color-coded depth of penetration.

Computation of points within a radius r for the collision detection and of the closest non-colliding points in the environment for calculating the penetration depth are done by creating and searching in a k-d tree of the environment. Using this data structure, every lookup can be done in $O(\log(n))$ time with n being the number of points in the environment. Therefore, collision detection can be done in $O(lm \log(n))$ time with m being the number of points in the model and l being the number of points in the trajectory. The number of points in the environment n is reduced by only considering those points within a bounding sphere of the model. The number of points in the model can only be reduced until points which are neighbors on a surface are a maximum distance of r apart from each other. Using a model of 10000 points, moved through an environment of over 840000 points on a trajectory with 1000 poses (as seen in Fig. 7), collision detection takes about 11 seconds and depth of penetration calculation about 2 seconds on a single threaded desktop machine.

6 SUMMERY, CONCLUSION, AND FUTURE WORK

This paper has presented the essentials of a novel system for precise 3D modeling of production lines. With the built sensor skid we can access and digitize areas which are hard to inspect otherwise. We showed the ability to calculate the application specific depth-of-penetration.

Needless to say a lot of work remains to be done. In future work, we will concentrate on putting the acquired data into a global frame of reference, i.e., geo-referencing, on clustering the colliding point cloud, and on applying a divide-and-conquer methods to cope with longer trajectories. Currently, 100 6D poses are generated per second, which implies that the linear system of equations for solving our SLAM problem contains 600 entries per second.

References

Bosse, M. and Zlot, R., 2009. Continuous 3D Scan-Matching with a Spinning 2D Laser. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '09), pp. 4312–4319.

Davis, T. A., 2005. Algorithm 849: A Concise Sparse Cholesky Factorization Package. ACM Trans. Math. Softw. 31(4), pp. 587 – 591.

Elseberg, J., Borrmann, D. and Nüchter, A., 2013. Algorithmic solutions for computing precise maximum likelihood 3d point clouds from mobile laser scanning platforms. Rem. Sens.

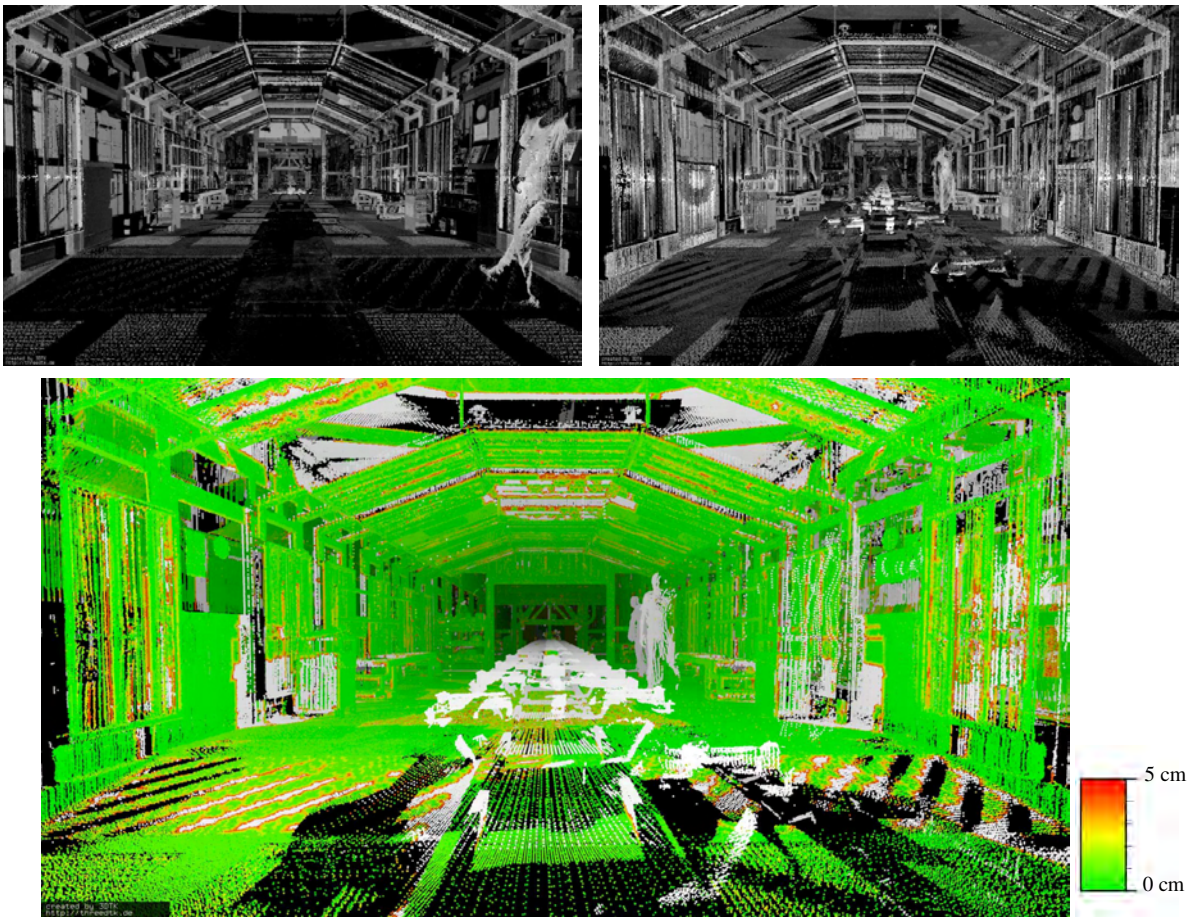

Figure 5: Above left: 3D Point Cloud acquired in a stop-scan-and-go fashion. Above right: Point Cloud acquired in continuous motion. Below: Derivation of the two point clouds in a centimeter scale.

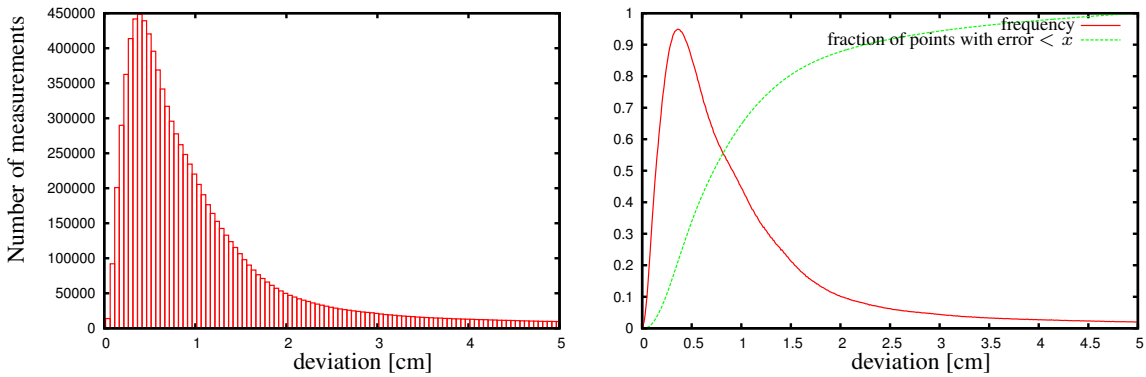

Figure 6: Left: Histogram showing the distribution of the deviations. Right: Frequency of occurrence derivation and the corresponding fraction of points with the error smaller than x.

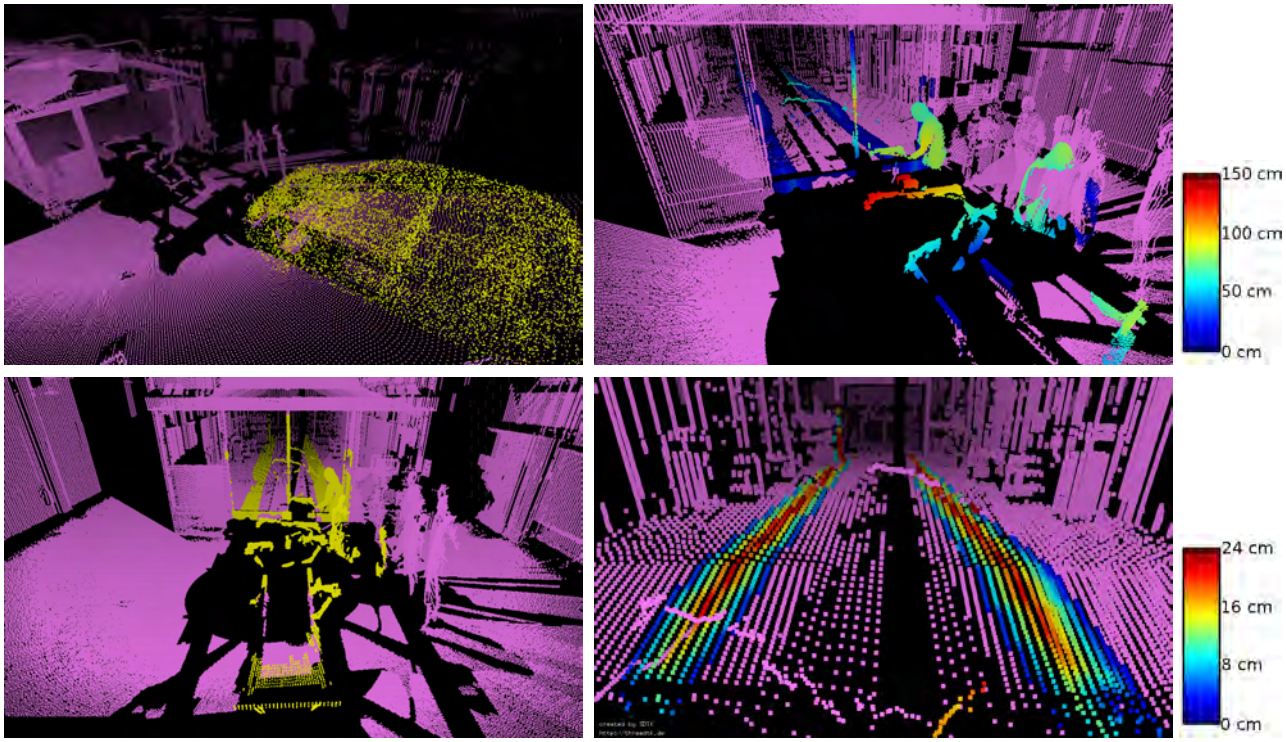

Figure 7: Upper Left: a view of two point clouds (environment and car). Lower Left: two separated point clouds (non-colliding and colliding points). Right Column: Color-coded depths of penetration

Elseberg, J., Magnenat, S., Siegwart, R. and Nüchter, A., 2012. Comparison of Nearest-Neighbor-Search Strategies and Implementations for Efficient Shape Registration. Journal of Software Engineering for Robotics (JOSER) 3(1), pp. 2 – 12.

Nüchter, A., Elseberg, J., Schneider, P. and Paulus, D., 2010. Study of Parameterizations for the Rigid Body Transformations of The Scan Registration Problem. Journal Computer Vision and Image Understanding (CVIU) 114(8), pp. 963–980.

Stoyanov, T. and Lilienthal, A. J., 2009. Maximum Likelihood Point Cloud Acquisition from a Mobile Platform. In: Proceedings of the IEEE International Conference on Advanced Robotics (ICAR '09), pp. 1 – 6.