Measurement Uncertainty Analysis of a Robotic Total Station Simulation

Christoph Klug
VRVIS Research Center
Vienna, Austria
klug@vrvis.at

Clemens Arth
Graz University of Technology
Graz, Austria
arth@icg.tugraz.at

Dieter Schmalstieg
Graz University of Technology
Graz, Austria
schmalstieg@icg.tugraz.at

Thomas Gloor
Hilti Corp.
Schaan, Liechtenstein
thomas.gloor@hilti.com

***Abstract*—The design of interactive algorithms for robotic total stations often requires hardware-in-the-loop setups during software development and verification. The use of real-time simulation setups can reduce the development and test effort significantly. However, the analysis of the simulation uncertainty is crucial for proper design of simulation setups and for the interpretation of simulation results. In this paper, we present a real-time simulation method for modern robotic total stations. We provide details for an exemplary robotic total station including models of geometry, actuators and sensors. The simulation uncertainty was estimated analytically and verified by Monte Carlo experiments.**

***Index Terms*—Simulation, Vision-Based Robots, Virtual Reality Systems**

I. Introduction and Related Work

A robotic total station (RTS) is commonly used for measuring 3D points with high precision and accuracy [1]. Designing and testing interactive algorithms for these robotic devices usually requires hardware-in-the-loop (HIL) approaches during various software development phases. An efficient alternative is to use simulator-in-the-loop (SIL). However, special care must be taken in the RTS simulation design to guarantee meaningful simulation results. Numerical imprecision of the simulation, especially when using game engines tuned for speed rather than precision, can effect the outcome. In particular, finite precision of floating-point arithmetic may not be negligible, and therefore has to be carefully examined.

We are not the first to investigate the benefits of game engines for robot design and simulation. Mattingly *et al.* [2] describe the use of Unity3D as their primary authoring tool. They show how one can easily build and simulate a skeletal structure with rigid-body kinematics.

Andaluz *et al.* [3] present a teleoperation simulator in Unity3D and MATLAB, based on their earlier mathematical model of the robotic manipulator [4]. Their work focus on simulating the robotic arm along with a low-level closed-loop control systems.

Meng *et al.* [6] introduced *ROSUnitySim*, a simulator based on the Robot Operation System (ROS) and Unity3D, which can simulate multiple unmanned aerial vehicles, including image sensors and light detection and ranging (LIDAR) sensors. The authors demonstrated an autonomous search task for multiple drones, including simultaneous localization and mapping (SLAM) and 3D path planning. A more complete description of the system is provided in another publication [7], but no details about sensor and device modeling are included.

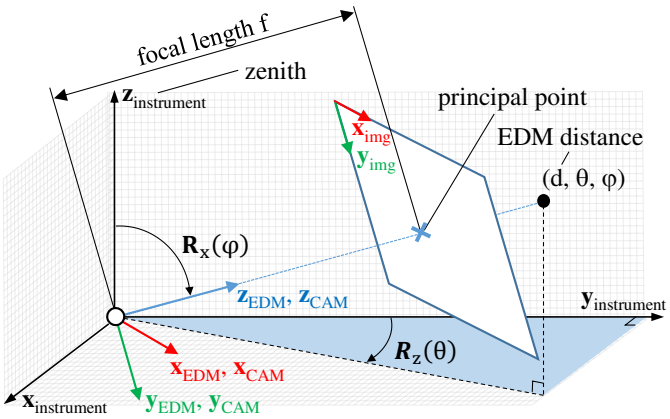


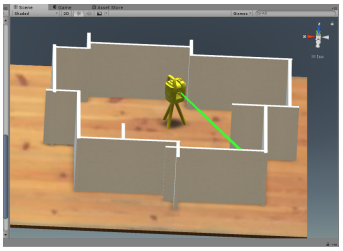
Fig. 1. Simple geometric model of an RTS with a camera module, as used by Klug *et al.* [5].

To the best of our knowledge, we are the first to consider an uncertainty analysis in real-time simulation of robotic systems. None of the mentioned previous approaches consider this aspects probably because the accuracy of measurements is not a major success factor for the applications considered in previous work.

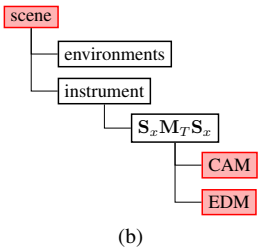
The contribution of this work is the description of a RTS simulator in Unity3D for the PLT 300 and a methodology for measurement uncertainty analysis. We determine a conservative uncertainty characterization of the simulator and verify the results using Monte Carlo (MC) experiments. Both approaches are useful for distinct purposes. The analytical equations can be applied prior to the design of computer-aided design (CAD) simulation scenes, whereas the MC experiments can easily be added to particular simulation. Our methodology is not limited to the PLT 300 device, but can be generalized to any RTS and other devices with similar hardware and sensor combinations.

II. Geometric Hardware Model

An RTS is an electrical theodolite which consists of an optical telescope, an electronic distance meter (EDM) and one or more cameras [1]. Modern devices support teleoperation; hence, the telescope has become optional and is commonly replaced by a camera module without any eyepiece. For example, the PLT 300 [8] contains no optical telescope, but



(a)



(b)

Fig. 2. (a) RTS simulation scene. (b) Simplified Unity3D scene graph.

a single camera, which supports RGB and infrared video streaming, and one EDM. In this work, we use the spherical geometric RTS model described by Klug *et al*. [5], shown in Fig. I. The stationary frame is referred as *instrument frame* and serves as reference frame for measured 3D points. The end-effector of the robotic device is called the *telescope frame*; it is virtually aligned with the *camera frame* and the *EDM frame*.

While this model does not consider nonlinear properties or inaccuracies in manufacturing and device calibration, it is sufficient for basic RTS simulation. We assumes that systematic errors are compensated through factuatory calibration, whereas remaining errors are treated as measurement uncertainties. This assumption reflects realistic RTS scenarios, where the incomplete knowledge of systematic effects or insufficient corrections add to the random variations of the observations. The simulation and analysis of systematic effects is beyond the scope of this work [1]. In the following, a short overview of the geometric relationship between sensor data and 3D points is provided.

Let $\{\theta, \varphi, d\}$ be a measurement tuple, where θ, ϕ describe the horizontal and vertical angle of the RTS and d is the corresponding EDM measurement. The simple geometric model for all actuators and sensors of the RTS is given as follows:

$$\tilde{\mathbf{x}}_I = \mathbf{M}_T \tilde{\mathbf{x}}_T, \;\; \tilde{\mathbf{u}} = \mathbf{P}\mathbf{M}_T^{-1}\tilde{\mathbf{x}}_I, \;\; \mathbf{M}_T = \mathbf{R}_z(-\theta)\mathbf{R}_x(-\varphi) \quad (1)$$

Here, $\tilde{\mathbf{x}}_I$ is a homogeneous 3D point; $\mathbf{M}_T$ is a 4×4 matrix, which describes the pose of the telescope frame with respect to the instrument frame; $\tilde{\mathbf{x}}_T$ in the corresponding 3D point with respect to the telescope frame, $\mathbf{P}$ is the camera projection matrix, and $\tilde{\mathbf{u}}$ is the corresponding image coordinate. An EDM distance d is measured along the z-axis of the telescope frame; it can be written as $\tilde{\mathbf{x}}_T = \begin{bmatrix} 0 & 0 & d & 1 \end{bmatrix}$.

An RTS without explicit optical telescope is mainly controlled by image-based targeting. The conversion of 2D image coordinates to spherical coordinates delivers the required servo control parameters $\{\theta, \varphi\}$, which can be extracted from re-projected image rays and Eqn. 1 [5].

III. Unity3D as Simulator Front-End

The game engine Unity3D was used as a front-end for a real-time simulation. It conveniently allows creating the measurement setup as a scene graph consisting of nodes for sensors, actuators and measurement targets with attached *behavior* scripts for faithfully simulating the firmware behavior, environmental influences and timing constraints. Fig. 2a shows an exemplary simulation scene, and Fig. 2b shows the corresponding scene graph. Unity3D uses a left-handed coordinate system, while most RTS models are based on right-handed coordinate systems [1], [9]. Hence, kinematic models and simulation results of the Unity3D module must be converted accordingly. The related axis conversions are shown in Fig. 3.

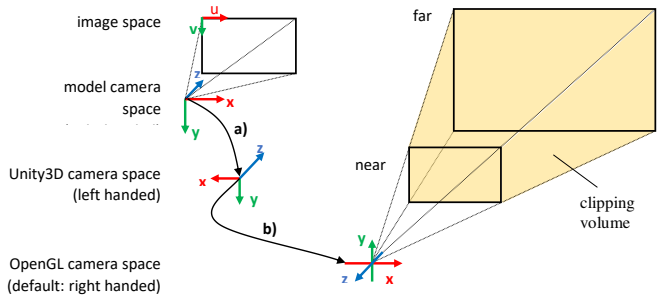


Fig. 3. Transformation between RTS model, Unity3D and OpenGL: a) convert camera space RTS model to Untiy3D, b) convert camera space from Untiy3D to OpenGL.

A. Modeling RTS sensors, actuators and targets

The geometric environment of the RTS was modeled as triangle meshes in a scene graph (Fig. 2b): The *instrument node* represents the instrument frame of the RTS, which can be freely positioned within the scene. Telescope frame, camera sensor and EDM were modeled as child nodes of the instrument node. The *environment node* is a placeholder for different measurement targets. Environments may include CAD models of reflective and non-reflective measurement targets, individual rooms, complete buildings or urban areas. The EDM was determined as the smallest distance d between the ray origin and the closest ray intersections with a scene object.

The reader is referred to the Unity3D User Manual [10] for implementation details.

B. Modeling sensor uncertainties

The analysis and report of measurement uncertainties of physical sensors is crucial for assessing simulation and measurement results. The JCGM 100:2008 Guide to the Expression of Uncertainty (GUM) [11] standardizes the evaluation and report of measured physical quantities, using measurement uncertainties to guarantee reliable and repeatable experiments. We follow GUM in the analysis of uncertainty given in the following.

RTS manufacturers specify sensor uncertainties for normal distributed random variables in following general form:

$$p(|x' - x| \leq k u_c(x')) = CI_k \quad (2)$$

Here, x is the measured quantity, $u_c(x')$ is the combined standard uncertainty of the measurement result x', k is the coverage factor, and CI_k is the corresponding confidence interval.

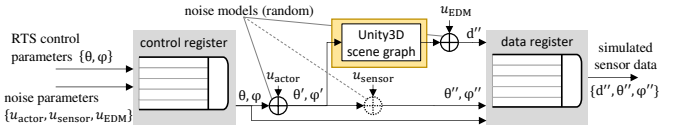


Fig. 4. Sensor hardware uncertainty model of the RTS simulator. The noise parameters can be controlled externally which allows simulation of different hardware specifications and measurement setups; $u_{actuator}$ and u_{sensor} are the actuator and sensor uncertainties respectively; u_{EDM} is the distance measurement uncertainty.

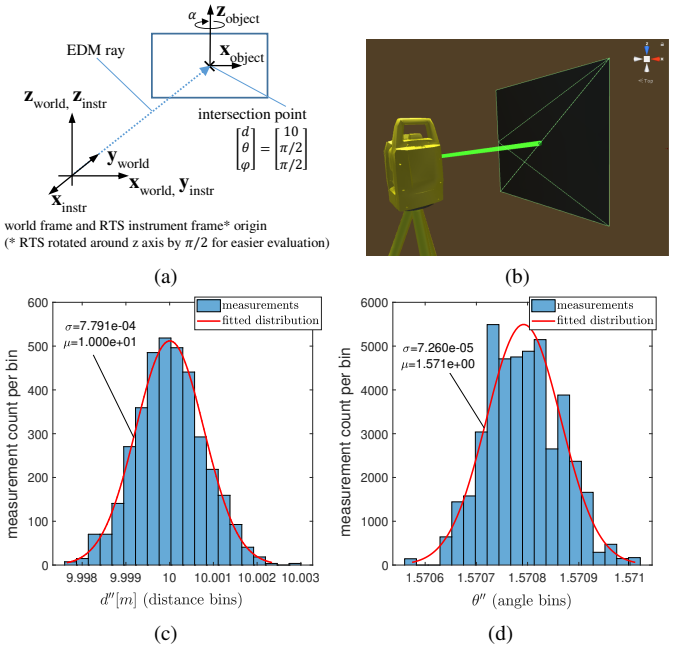


Fig. 5. Sensor hardware noise simulations. (a) Uncertainty verification concept. (b) Unity3D setup for simulation uncertainty verification. (c) EDM noise: $u_1(x') = 0.75{\times}10^{-3}$m, $u_2(x') = 10{\times}10^{-6}$. (d) angle sensor noise: $u_1(x') = 5'' \approx 24.241{\times}10^{-6}$rad, $u_2(x') = 0$.

For RTS sensors, one can estimate the combined standard uncertainty according as follows:

$$u_c(x') \approx \sqrt{u_1(x') + (xu_2(x'))^2} \tag{3}$$

Here, $u_1(x')$ is the bias and $u_2(x')$ is the scale, both provided by the device manufacturers or through sensor calibration. Usually, the true value x is not known, and the current measurement x' is used instead in Eqn. 3 to estimate the combined standard uncertainty $u_c(x')$. In this work, the error model given in Eqn. 2 was applied to simulate EDM and angular sensor uncertainties. The measurement uncertainty distribution was assumed to have normal distribution and zero mean. In addition, the servo actuator and angular sensor were split, thereby increasing the flexibility of simulation. Fig. 4 shows the sensor uncertainty model of the RTS simulator, Fig. 5 shows the verification setup for the simulation uncertainty. We used the Box-Muller transform [12] to generate the sensor reading x' from uniformly distributed random values a, b and the simulated true value x:

$$x' = x + \sqrt{-2\ln(a)}\cos(2\pi b u_c(x')) \tag{4}$$

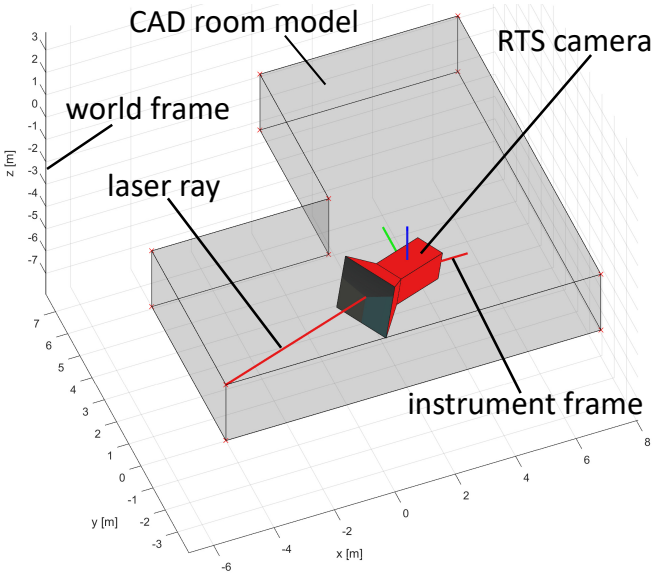


Fig. 6. Simple scene setup for the uncertainty analysis of floating-point effects.

Here, the desired standard uncertainty $u_c(x')$ is taken from Eqn. 3.

IV. Simulator Uncertainties

In Section III-B, a simple model for sensor uncertainty simulation was provided, used for simulating realistic hardware sensors. However, the limitations of the real-time simulation itself have not been considered yet.

We will derive a simple a-priori estimate of the uncertainty of a particular simulation setup. For this particular investigation, we assumed the ideal geometric model presented in Section II without any systematic modeling error and with ideal sensors. Hence, all sensor uncertainties discussed in Section III-B were set to zero. As a result, only the variability of the simulation in Unity3D itself is considered here. Unless otherwise stated, arithmetic rules, naming convention and format specification conform to the IEEE 754-2008 standard for floating-point arithmetic [13] and follow GUM [11].

A. Uncertainty sources of the RTS simulator

Unity3D allows the placement of scene objects anywhere in the coordinate system. Unlike CAD tools, game engines rely on fast single-precision (32-bit) floating-point representations for handling geometric entities.

In this work, we used 64-bit floating-point format when working with the real device and as a data interchange format, but relied on the 32-bit floating-point format of Unity3D for scene graph simulation and rendering. Hence, we must consider rounding errors of numerical operations on mesh vertices and scene objects with large distances to the world frame origin. Further uncertainty sources are data input and format conversions of meshes and RTS control parameters.

B. Analytical uncertainty estimation

Fig. 6 shows an example simulation setup with metric scale and realistic object positions. The simulator follows a graphics

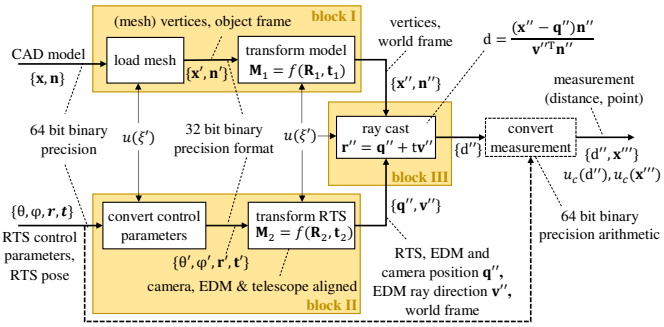


Fig. 7. Uncertainty propagation in the simulator pipeline, partitioned into three processing blocks.

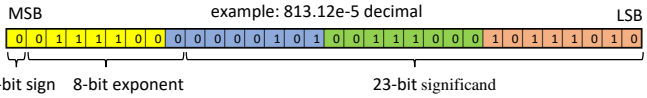


Fig. 8. Memory layout and bit allocation of 32-bit floating-point numbers according to IEEE 754-2008

pipeline [14] with three major blocks of uncertainty (Fig. 7): *Block I* transforms CAD vertices and normals $\{\mathbf{x}, \mathbf{n}\}$ to the word frame; *Block II* transforms the RTS object to the world frame; *Block III* calculates the EDM distance using ray casting. The ray casting result was interpreted as spherical coordinate vector with optional conversion to the Euclidean space. The latter was carried out with 64-bit floating-point arithmetic, which we regarded as having negligible error for our purposes. For better readability, the enumeration indices of vertices and normals were omitted in the following.

Input vertices are originally stored at 64-bit precision, but loaded in *block I* with 32-bit floating-point precision. The expected rounding error is mainly influenced by the distance between vertex and origin and the limits of the data format. Fig. 8 shows the memory layout of the 32-bit binary floating-point format, including 1-bit sign, an 8-bit exponent, and a 23-bit *mantissa* plus one implicit leading bit. If the *format precision* p denotes the maximum number of digits at radix (base) 10 which can be represented, vertices and normals in Unity3D have a format precision of $p = 7$.

Let ξ be an input number at radix 10. The rounded floating-point format ξ' and the round-off error e_ξ are given as follows:

$$\xi' = \lfloor \xi \cdot 10^{N_{\text{frac}}} \rceil \cdot 10^{-N_{\text{frac}}} \qquad e_\xi = \xi - \xi' \tag{5}$$

$$N_{\text{frac}} = p - N_{\text{int}} \qquad N_{\text{int}} = \begin{cases} \lfloor \log_{10} |\xi| \rfloor + 1, & \xi \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

Here, N_{int} and N_{frac} are the number of integer and fractional digits, respectively. If the true value ξ is not known, the uncertainty bounds $a_+ - a_- = 2a$ of the rounded vertex ξ' are given by a rectangular distribution:

$$\xi - a_- \leq \xi' \leq \xi + a_+ \qquad a = 0.5 \cdot 10^{-(N_{\text{frac}}+1)} \tag{7}$$

Here, a_- and a_+ are the lower and upper limit, respectively. The standard uncertainty of a simulated vertex without any applied transformation is then given as follows [11]:

$$u(\xi') \approx \frac{0.5}{\sqrt{3}} \cdot 10^{-(N_{\text{frac}}+1)} \tag{8}$$

Let $\mathbf{x}$ be an input vertex $\mathbf{x}$ with corresponding normal $\mathbf{n}$, given in 64-but. Let $\mathbf{x}'$ and $\mathbf{n}'$ be the corresponding entities in 32-bit floating-point precision. Then, the uncertainties for each element of the vertex and normal are given as follows:

$$u(x') \approx u(\xi')|_{\xi = x_b} \qquad u(n') \approx u(\xi')|_{N_{\text{frac}}=7} \tag{9}$$

Here, $u(x')$ defines the element-wise uncertainties of a vertex $\mathbf{x}'$, and $u(n')$ defines the element-wise uncertainties of the normal $\mathbf{n}'$. A conservative approximation for all vertices is given by $\xi = x_b$, where x_b is the maximum absolute value of all components of the scene bounding box. Assuming $||\mathbf{n}|| = 1$, the fractional digit count for the normals in Eqn. 9 is given by $N_{\text{frac}} = 7$.

To reduce the calculation complexity, we used following approximations: Scene transformations were reduced to Euclidean transformations to avoid homogeneous matrix operations. Where feasible, concatenated transformations as a single transformation only.

The quadratic variance propagation for a single output and multiple input variables was used to combine the uncertainty sources, which can be written in the following form [15], [16]:

$$u_c(\chi) = \sqrt{\mathbf{g}^T \mathbf{V} \mathbf{g}} \qquad \mathbf{g} = \vec{\nabla}\chi = \left[\frac{\partial \chi}{\partial \beta_1} \dots \frac{\partial \chi}{\partial \beta_N} \right]^T \qquad \boldsymbol{\beta} = \{\beta_i\} \tag{10}$$

Here, $u_c(\chi)$ is the combined standard uncertainty of some function $\chi(\boldsymbol{\beta})$, and $\mathbf{V}$ is the covariance matrix for N input variables β_i with $i \in \{1 \dots N\}$.

The transformation of a model space vertex $\mathbf{x}'$ and normal $\mathbf{n}'$ to the corresponding world space vertex $\mathbf{x}''$ and normal $\mathbf{n}''$ is given as follows:

$$\mathbf{x}'' = \mathbf{R}'\mathbf{x}' + \mathbf{t}' \qquad \mathbf{n}'' = \mathbf{R}\mathbf{n}' \tag{11}$$

Here, $\mathbf{R}'$ is the 32-bit floating-point representation of a 3×3 rotation matrix $\mathbf{R}$ and $\mathbf{t}'$ is the 32-bit floating-point representation of a 3×1 translation vector $\mathbf{t}$. To reduce the uncertainty transfer function output for block I to a single variable, the same scalar uncertainty $u_c(x'')$ was used for the x, y and z component of vertex $\mathbf{x}''$, and all components were treated as independent. Analogously, the translational standard uncertainty $u(t') \approx u(x')$ was assumed to be equal for all axes. The fractional digit count N_{frac} in Eqn. 8 can be estimated using $|x| = x_b$.

Let r'_{ij} be the elements of the rotation matrix $\mathbf{R}'$, $u(r_{ij})$ be the element-wise standard uncertainty, and r_{ij} the elements of the true rotation matrix $\mathbf{R}$. Rotation matrix rows are normalized to one, which leads to $|r_{ij}| \leq 1$. Consequently, the uncertainty value for each matrix element r'_{ij} can be approximated by $u(r'_{ij}) \approx u(\xi)$ with $N_{\text{frac}} = 7$, as defined in Eqn. 8. Without known rotation parameters, a liberal approximation for the variance propagation of the rotational part is given by substituting the identity matrix as

rotation matrix $\mathbf{R}$. However, a conservative approximation is preferable, substituting $|r_{ij}| = 1$ for all matrix elements[1]. Consequentially, the standard uncertainty for each element of $\mathbf{x}''$ can be estimated from the first component of $\mathbf{x}''$ only, which is given by the first row of Eqn. 11 according to

$$x_1'' = r_{11}'x_1' + r_{12}'x_2' + r_{13}'x_3' + t_1' \qquad \mathbf{x}' = \begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix} \qquad \mathbf{t}' = \begin{bmatrix} t_1' \\ t_2' \\ t_3' \end{bmatrix} \tag{12}$$

where x_1'' is the first component of vertex $\mathbf{x}''$.

According to Eqn. 10, the partial derivatives of x_1'' with respect to $\boldsymbol{\beta}$ can be calculated as follows:

$$\mathbf{g} = \vec{\nabla} x_1'' \qquad \boldsymbol{\beta} = \{x_1' \dots x_3', r_{11}' \dots r_{13}', t_1'\} \tag{13}$$

Here, $\boldsymbol{\beta}$ is the set of 7 scalar variables. The corresponding $[7 \times 7]$ covariance matrix $\mathbf{V}$ is given as follows:

$$\mathbf{V} = diag(\mathbf{1}_3^T u(x'), \mathbf{1}_3^T u(r_{ij}'), \mathbf{1}_3^T u(t')) \tag{14}$$

Here, $\mathbf{1}_3^T$ is a 1×3 vector with all elements equal one. All input variables are assumed to be independent which makes the covariance matrix diagonal. By substituting $|x_i| = x_b$, $|r_{ij}| = 1$, the estimation of $u_c(x'')$ can be reduced to the following:

$$u_c(x'') \approx \frac{1}{k}\sqrt{3u(r_{ij}')^2 x_b^2 + 3u(x')^2 + u(t')^2} \tag{15}$$

Here, the regularization term k takes the conservative assumptions into account. Therefore, it can be interpreted as *coverage factor*. Using $k = 3$ avoids overestimation of uncertainty effects and turns the expanded uncertainty of Eqn. 15 into the standard uncertainty as defined in GUM [11].

The element-wise combined standard uncertainty $u_c(n'')$ of a transformed vertex normal $\mathbf{n}''$ can be directly derived from Eqn. 15. From $||\mathbf{n}'|| \approx 1$ follows that the bounding box for all normal elements in Eqn. 15 can be set to $x_b = 1$. Normals are not affected by translation, hence $u(t')$ in Eqn. 15 is zero, and $u_c(n'')$ can be estimated as follows:

$$u_c(n'') \approx \frac{1}{k}\sqrt{3u(r_{ij}')^2 + 3u(n')^2} \tag{16}$$

Block II models the EDM by transforming the RTS control parameters to a ray in world space. Let the instrument space ray be defined by the ray origin $\mathbf{q}$ and the ray direction $\mathbf{v}$. We approximated the chain of RTS transformations by a single Euclidean transformation as done in *block I*[2]. Assuming the same bounding box for all vertices, scene objects and the RTS position, the combined uncertainty values for the transformed world space ray origin $\mathbf{q}''$ and the transformed world space ray direction $\mathbf{v}''$ can be approximated by the combined uncertainty values calculated for block I:

$$u_c(q'') \approx u_c(x'') \qquad u_c(v'') \approx u_c(n'') \tag{17}$$

[1]This leads to an invalid rotation matrix, but the results are more trustworthy for conservative approximations without explicit knowledge of the rotation parameters.

[2]A more detailed, but also more complex approximation of the variance propagation would consider all control parameters in block II as shown in Fig. 7.

Here, $u_c(q'')$ and $u_c(v'')$ are the element-wise combined standard uncertainties for the transformed ray origin and ray direction, respectively.

For the analysis of *Block III*, the EDM ray casting was simplified to plane-ray intersection. The simulated distance d'' between the ray origin $\mathbf{q}''$ and the scene intersection point can be calculated as follows:

$$d'' = \frac{(\mathbf{x}'' - \mathbf{q}'')^T \mathbf{n}''}{\mathbf{v}''^T \mathbf{n}''} = \frac{(\mathbf{x}'' - \mathbf{q}'')^T \mathbf{n}''}{||\mathbf{v}''||\ ||\mathbf{n}''|| \cos\alpha} \tag{18}$$

Here, d'' is the ray length, calculated with finite precision arithmetic, $\mathbf{n}''$ is the plane normal, $\mathbf{x}''$ is a point on the plane, and α is the angle of incidence between the ray direction and the plane normal. The propagated combined standard uncertainty $u_c(d'')$ can be estimated by solving Eqn. 10 with $\chi = d''$, using the input parameters $\boldsymbol{\beta}$:

$$\boldsymbol{\beta} = \{\mathbf{x}'', \mathbf{n}'', \mathbf{q}'', \mathbf{v}''\} \tag{19}$$

Here, $\boldsymbol{\beta}$ is interpreted as set of 12 scalar variables. The corresponding $[12 \times 12]$ covariance matrix $\mathbf{V}$ is given as follows:

$$\mathbf{V} = diag(\mathbf{1}_3^T u_c(x''), \mathbf{1}_3^T u_c(n''), \mathbf{1}_3^T u_c(q''), \mathbf{1}_3^T u_c(v'')) \tag{20}$$

The denominator of Eqn. 18 shows a significant dependency between the combined uncertainty of the distance measurement, $u_c(d'')$, and the incident ray angle α. A first or second order Taylor approximation of the denominator could be used for angles close to zero, but would be insufficient for larger angles. As an alternative, a rough estimation of $u_c(d'')$ can be derived by substituting an independent variable g with zero variance as denominator. Hence, Eqn. 18 can be written in the following form:

$$d'' = \frac{(\mathbf{x}'' - \mathbf{q}'')^T \mathbf{n}''}{g} \qquad ||\mathbf{v}''|| \approx ||\mathbf{n}''|| \approx 1 \Rightarrow g \approx \cos(\alpha) \tag{21}$$

Using the same considerations as for blocks I and II, $u_c(d'')$ can be approximated by solving Eqn. 10 with $\chi = d''$, where $\boldsymbol{\beta}$, $\mathbf{V}$ and d'' are given in Eqns. 19 - 21, respectively. The conservative substitutions of $|n_i''| = 1$ for all elements of $\mathbf{n}$ and $\mathbf{x}'' - \mathbf{q}'' = \begin{bmatrix} 2x_b & 2x_b & 2x_b \end{bmatrix}^T$ lead to the following form:

$$u_c(d'') \approx \frac{1}{|\cos(\alpha)|}\sqrt{12u_c(n'')^2 x_b^2 + 3u_c(x'')^2 + 3u_c(q'')^2} \tag{22}$$

Here, x_b denotes the maximum absolute element of the scene bounding box.

Optionally, the simulated distance measurement d'' and the angle control parameters $\{\theta, \varphi\}$ can be converted to an Euclidean point $\mathbf{x}'''$ using Eqn. 1 with 64-bit floating-point arithmetic. If the norm of all columns of an Euclidean rotation matrix equals one, the combined uncertainty for each element of the measured point can be estimated as follows:

$$u_c(x''') \approx m\frac{u_c(d'')}{\sqrt{3}} \qquad 1 \le m \le \sqrt{3} \tag{23}$$

Here, m is a correction factor which avoids underestimation for special scene setups. For example, when measuring far distances along a single axis, the combined standard uncertainty $u_c(x''')$ of the significant axis can be analyzed with $m = \sqrt{3}$.

V. Experimental Results

We evaluated our setup with MC experiments based on random placements of the simulated RTS and associated targets.

A. Uncertainty estimation using Monte Carlo experiments

The uncertainty propagation functions developed in Section IV-B can be verified using MC experiments. Let $\mathbf{y}_i$ be the true intersection point of a laser distance measurement and $\mathbf{q}_i$ the laser ray origin, both calculated with 64-bit arithmetic. The standard uncertainty $u(d'')$ for the simulated distance measurement can be estimated by N repeated measurements, using the following:

$$u(d'') \approx \sqrt{\frac{\sum_{i=1}^{N} (d_i'' - d_i)^2}{N}} \qquad d_i = ||\mathbf{y}_i - \mathbf{q}_i|| \qquad (24)$$

The simulator setup for the MC experiments is shown in Fig. 9. For each experiment, the CAD model, RTS pose and measurement target pose were randomly generated. The CAD model of each experiment consists of a single triangle, the measurement target, with arbitrary rotation, a circumradius of one, and the centroid placed randomly on the bounding box. In the world space frame, the RTS was placed on the negative and the measurement target was placed on the positive $y = 0$ plane of the bounding box. Each experiment was evaluated using the simulator and compared with results of an 64-bit precision arithmetic model.

B. Uncertainty estimation results

We estimated the element-wise combined standard uncertainties $u_c(x''')$ of the intersection point x''' for the distances $d \in \{0.5, 1, 10^1, 10^2, 10^3\}$ meter. The world space bounding box was assumed to be $x_b \approx \pm\frac{d}{2}$ in x, y and z direction, respectively. The RTS was placed near the border of the bounding box as shown in Fig. 9. The rays' incident angles were set to $\alpha \in \{0, \frac{\pi}{12}, \frac{\pi}{6}, \frac{\pi}{4}, \frac{\pi}{3}, \frac{2\pi}{5}, \frac{9\pi}{20}, \frac{9\pi}{40}, \frac{9\pi}{1000}\}$. The Euclidean transformations $\mathbf{M}_{\text{instr}}$ and $\mathbf{M}_{\text{mesh}}$ describe the world space transformation for the RTS and the measurement target, respectively. For each experiment, both matrices were randomly generated, placing the RTS and the measurement target near the scene bounding box. The CAD model for each experiment consisted of single triangle, the measurement target, randomly placed within the CAD bounding box. Each triangle was created by randomly picking three points near the CAD bounding box. In addition, $\mathbf{M}_{\text{instr}}$ and $\mathbf{M}_{\text{mesh}}$ include jitter for the poses of each experiment; the jitter was created using random translations with ± 0.5m in x,y and z direction and random rotations with ± 0.05rad for each Euler rotation angle. In addition, a local RTS rotation around the z axis of the instrument frame was applied, using randomized angles between 0 and 2π. Fig. 10 shows the resulting uncertainty

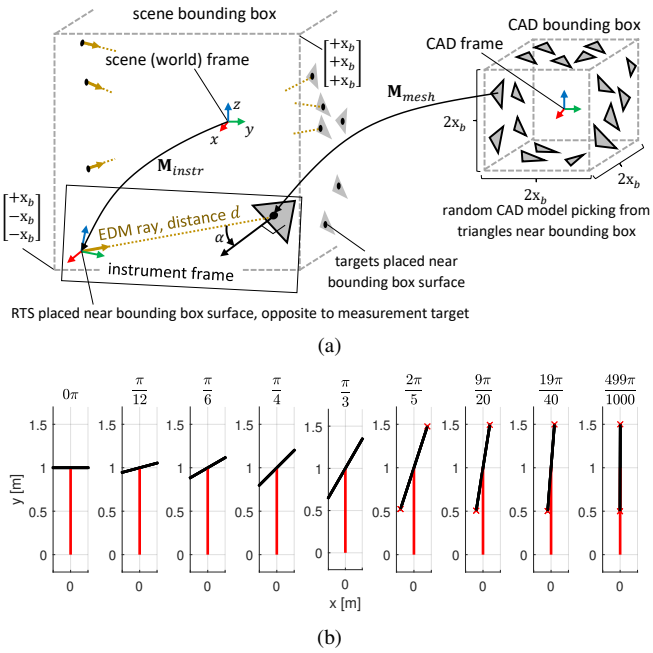


Fig. 9. Setup of MC uncertainty experiments for analyzing influences of floating-point arithmetic. (a) object frames and transformations. (b) Varying incident angle experiments (top view, EDM ray and measured plane, instrument frame).

TABLE I
Analytical and simulated (SIM) uncertainty results for the realistic RTS simulator setup. The scene bounding box was defined using $x_b \approx 10$m. The simulator application programming interface (API) is derived from the original driver and does not provide access to intermediate variables, hence $u(n')$, $u(x')$ and $u(n'')$ where not accessible (NA).

Method	$u(n')$ [m]	$u(x')$ [m]	$u_c(n'')$ [m]	$u_c(x'')$ [m]	$u_c(d'')$ [m]
Analytical $\alpha = 0.33\pi$	2.89×10^{-8}	5.00×10^{-7}	2.36×10^{-8}	3.44×10^{-7}	1.87×10^{-6}
MC $\alpha = 0.33\pi$	NA	NA	NA	NA	2.41×10^{-6}
Analytical $\alpha = 0.45\pi$	2.89×10^{-8}	5.00×10^{-7}	2.36×10^{-8}	3.44×10^{-7}	6.00×10^{-6}
MC $\alpha = 0.45\pi$	NA	NA	NA	NA	7.34×10^{-6}
MC, realistic $\alpha \approx 0.46\pi$	NA	NA	NA	NA	5.82×10^{-6}

characterization curves, estimated analytically and by using MC experiments. The run-time of the analytical uncertainty evaluation was $4.2\times10^{-3}s$, implemented and executed with MATLAB 2017, under Microsoft Windows 10 on an Intel Core i7 processor with 64GB RAM. The MC simulation took $2.5\times10^{3}s$ ($\approx 0.7h$), using 100 samples for each configuration.

In addition, we estimated the simulation uncertainty for the realistic measurement setup shown in Fig. 6. We assumed a scene bounding box with $x_b \leq 10$m and expected incident ray angles of $\alpha \approx \{\frac{\pi}{3}, 0.45\pi\}$ in Eqn. 22. The uncertainty results of the MC experiments with the realistic setup showed an average incident angle of 0.46π and an average EDM distance of 7.2 m. Results are provided in Tab. I.

The classification curves provided in Fig. 10 allow uncertainty estimations prior to the design of simulation setups and related CAD models. The analysis shows a linear rela-

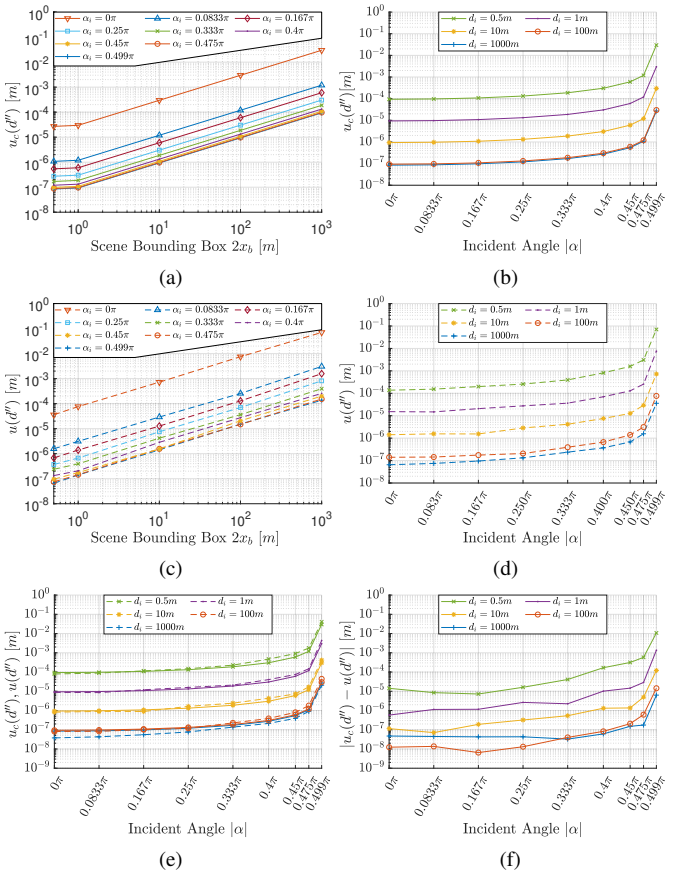


Fig. 10. Uncertainty estimations of EDM intersection points for the RTS simulator. Analytical uncertainty: with respect to the scene bounding box (a), with respect the the incident angle (b). Uncertainty using MC experiments: with respect to the measured scene bounding box (c), with respect to the incident angle (d). Analytical uncertainty $u_c(d'')$ (continuous lines) compared with MC experiments (dashed lines), plotted with respect to the incident angle (e,f).

tionship between the logarithm of the simulation uncertainty and the logarithm of the scene bounding box; furthermore, a significant influence of the angle of incidence is observable. Fig. 10e shows the overlay of the analytic approach and the MC experiment; Fig. 10f shows the difference between the two methods. Both diagrams indicate that the analytical uncertainty estimation is a reasonable prior approximation. With increasing incident angle, the difference between the two methods increases.

VI. Discussion and Conclusions

In this work, we have presented a Unity3D based RTS simulator for graphical and non-graphical algorithm development, test and verification, including a simple geometric models for the PLT 300, actors, sensors and uncertainties. We provided an analytical a-priori method for assessing simulation uncertainty, which is useful for the design of simulation setups. We validated the results with MC experiments, which provide more detailed uncertainty estimations for a particular setup, but at the cost of increased execution time. We verified the results using a realistic simulation setup. With our concept, real-time algorithms and workflows can be designed without hardware-in-the-loop setups; custom simulation environments can be implemented, and various measurement effects can be simulated.

While the simplified geometric models in this work are sufficient for many setups, detailed sensor, actuator and environment models could be used to simulate further physical effects. The simulator uncertainty could be reduced by increasing the numerical precision of certain geometric operations. This may include scaling the nominal world units, splitting and scaling object space or image space regions into multiple parts, or implementing methods like ray casting explicitly using higher precision arithmetic. Furthermore, physical and behavioral models of human operators could be included in the scene graph. More realistic scenes and comparison with RTS measurements of laboratory setups could further increase the reliability of the simulation and uncertainty evaluation.

References

[1] J. Uren and B. Price, *Surveying for Engineers*. Basingstoke England New York: Palgrave Macmillan, 2010.

[2] W. A. Mattingly, D. j. Chang, R. Paris, N. Smith, J. Blevins, and M. Ouyang, "Robot design using Unity for computer games and robotic simulations," in *2012 17th Int. Conf. Comput. Games*, Jul. 2012, pp. 56–59.

[3] V. H. Andaluz, F. A. Chicaiza, C. Gallardo, W. X. Quevedo, J. Varela, J. S. Sánchez, and O. Arteaga, "Unity3D-MATLAB simulator in real time for robotics applications," in *Augment. Reality, Virtual Reality, Comput. Graph. Third Int. Conf. AVR 2016, Lecce, Italy, June 15-18, 2016. Proceedings, Part I*, L. T. De Paolis and A. Mongelli, Eds. Cham: Springer International Publishing, 2016, pp. 246–263.

[4] V. H. Andaluz, P. A. Canseco, A. Rosales, F. Roberti, and R. Carelli, "Multilayer scheme for the adaptive cooperative coordinated control of mobile manipulators," in *IECON 2012 - 38th Annu. Conf. IEEE Ind. Electron. Soc.*, Oct. 2012, pp. 2737–2743.

[5] C. Klug, D. Schmalstieg, and C. Arth, "Measuring human-made corner structures with a robotic total station using support points, lines and planes," in *Proc. 12th Int. Jt. Conf. Comput. Vision, Imaging Comput. Graph. Theory Appl. - Vol. 6 VISAPP, (VISIGRAPP 2017)*, INSTICC. SciTePress, 2017, pp. 17–27.

[6] W. Meng, Y. Hu, J. Lin, F. Lin, and R. Teo, "ROS+Unity: An efficient high-fidelity 3D multi-UAV navigation and control simulator in GPS-denied environments," in *IECON 2015 - 41st Annu. Conf. IEEE Ind. Electron. Soc.*, Nov. 2015, pp. 2562–2567.

[7] Y. Hu and W. Meng, "ROSUnitySim: Development and experimentation of a real-time simulator for multi-unmanned aerial vehicle local planning," *Simulation*, vol. 92, no. 10, pp. 931–944, Oct. 2016.

[8] Hilti Corporation, "PLT 300," https://www.hilti.com/measuring-systems/optical-tools/r4728599, 2017, [Online; Accessed 28 July 2017].

[9] J. Awange and E. W. Grafarend, *Solving Algebraic Computational Problems in Geodesy and Geoinformatics: The Answer to Modern Challenges*. Springer Berlin Heidelberg, 2005.

[10] Unity Technologies, "Unity manual," https://docs.unity3d.com/Manual, 2017, [Online; Accessed 28 July 2017].

[11] "JCGM 100:2008 - Evaluation of measurement data - Guide to the expression of uncertainty in measurement," Int. Organ. Stand. Geneva ISBN, Standard, 2008.

[12] G. E. P. Box and M. E. Muller, "A note on the generation of random normal deviates," *Ann. Math. Stat.*, vol. 29, no. 2, pp. 610–611, 1958.

[13] "IEEE Std 754-2008: IEEE Standard for Floating-Point Arithmetic (Revision of IEEE Std 754-1985)," IEEE Computer Society, Standard, 2008.

[14] M. Segal and K. Akeley, "The OpenGL graphics system: A specification. version 4.5 (core profile)," https://www.khronos.org/registry/OpenGL/specs/gl/glspec45.core.pdf, Jun. 2017, [Online; Accessed 28 July 2017].

[15] R. Barlow, *Statistics: A Guide to the use of Statistical Methods in the Physical Sciences*. Chichester, England New York: Wiley, 1989.

[16] J. Tellinghuisen, "Statistical error propagation," *J. Phys. Chem. A*, vol. 105, no. 15, pp. 3917–3921, 2001.