Robust Object Pose Tracking for Augmented Reality Guidance and Teleoperation

David Black, Septimiu Salcudean

***Abstract*—For many augmented reality guidance, teleoperation, or human-robot interaction systems, accurate, fast, and robust 6 degree of freedom object pose tracking is essential. However, current solutions easily lose tracking when line-of-sight to markers is lost. In this paper we present a tracking system which matches or improves on current methods in speed and accuracy, achieving 1.77 mm and 1.51 degrees accuracy at 22 Hz. Reflective markers are segmented in infrared images using contour detection before using the known marker geometry to perform point correspondence and pose computation using novel approaches. At the same time, a new square-root unscented Kalman filter is introduced which improves accuracy and flexibility by tracking the markers themselves rather than the computed pose, and enables fusion of an external inertial measurement unit. This reduces noise and makes the tracking robust to brief loss of line-of-sight. The algorithms and methods are described in detail with pseudo-code for ease of reproduction. The system is implemented in simulation and on a Microsoft HoloLens 2 using Unity for ease of entry and integration into graphical projects. The code is made available open source. Tests of the system are described, and the results analyzed.**

***Index Terms*—Augmented Reality, Tracking, Computer Vision, Kalman filter, Teleoperation, Human Computer Interaction**

I. Introduction

Mixed reality (MR) headsets are being used increasingly for guidance of manual tasks. This includes MR guidance of device assembly [1], maintenance and inspection [2], [3], physiotherapy and rehabilitation [4]–[6], needle biopsies and interventions [7], [8], surgical guidance [9], and even sports instruction [10], [11]. Direct "human teleoperation" of people through a tightly coupled hand-over-hand guidance system is also being developed [12], [13]. By fusing virtual content with the real environment, MR enables seamless instruction and guidance while completing tasks, and allows users to "see inside" patients by overlaying medical images.

In most of these cases, the user of the interface manipulates a tool or device with their hand. For example, in human teleoperation for tele-ultrasound, a novice user moves an ultrasound probe with hand-over-hand guidance from a remote expert. To provide feedback in any of these systems, whether visual feedback for the user to aid real-time correction of motions, recording of performance for later evaluation, or even pose feedback to a remote expert in teleoperation systems, it is essential to track the pose of the tool. Additionally, sensors on the tool often require pose information. In orthopedic surgery, the tools are commonly used to mark locations for drilling or cutting, where accuracy is key [14], [15]. For instrumentation with force sensors, the tool orientation is necessary to transform measured forces and torques to the world frame [16]. Pose information can also be used in 3D ultrasound, to stitch several 2D ultrasound images from different planes into a 3D volume [17].

For teleoperation in particular, fast, accurate, and robust pose tracking is essential. To provide force feedback in a mixed reality teleoperation system, a 2-channel force-position architecture could be used [18]–[20] where the expert commands a pose, which is matched by the novice, resulting in an applied force on the patient. The force is measured and sent to the expert, where it is applied back to the expert through a haptic device [21]. In this case, force and thus also pose sensing is required on the user's tool. However, this architecture has problems if there are time delays [22]. To address this issue, more sophisticated four-channel architectures are also used, in which force, velocity, or wave variables are communicated bilaterally [23]–[26]. In all of these cases, pose tracking is required to transform the forces and directly to feed back the poses. To maintain stability and transparency, the tracking must be robust - i.e. rarely losing track of the object pose - and accurate.

To this end, different approaches are possible. Common methods involve external optical trackers from vendors including NDI (Waterloo, Ontario) and Atracsys (Puidoux, Switzerland). These are fast and accurate but require line of sight, which is easily lost in tasks requiring larger rotations, such as ultrasound. Alternatively, electromagnetic tracking systems are also available, for example the NDI Aurora. However, these suffer from drift in the presence of ferromagnetic materials in the workspace. Kalman filtering techniques have been employed to improve the performance of these devices, including for optical [27] and electromagnetic tracking [28]–[30]. In this context, many different types of Kalman filters have been described. Extended Kalman filters (EKFs) have been used to fuse optical and inertial pose measurements [31], but unscented Kalman filters (UKFs) are known to be more accurate as they do not linearize the dynamics, and they have equal computational complexity [32], [33]. For passive target tracking, comparative studies have showed improved performance using a UKF over an EKF [34], [35]. Further developments have also been described, including iterative UKFs [36] which boast improved convergence, robustness, and tracking accuracy. Square-root UKFs address the issue of numerical instability by guaranteeing positive definiteness of the covariance matrix in every time step [32]. Enayati et al. present an excellent review of object tracking systems with various filters, and themselves implement a UKF for fusion of optical and inertial information for object pose tracking [37].

These methods, however, use the output pose of a commer-

cial optical sensor. This requires external hardware in addition to the MR headset which is often not desirable. Object pose tracking using the HoloLens 2 (Microsoft, Redmond, Washington) has thus recently been explored. Many groups have used fiducial marker tracking like ArUco markers [38]–[40], but these are not sufficiently accurate [41]. Kunz et al used the infrared (IR) reflectivity and time of flight (ToF) depth sensors of the original HoloLens to track individual infrared markers with an accuracy of 0.76 mm and update rate of 22-30 Hz [42]. The accuracy of the actual 6 degree of freedom (6-DOF) pose calculated from the individual markers is not reported and it uses 8 IR markers which is not practical for most hand-held applications. The method includes no provision for occlusion of markers or temporary loss of tracking. Gsaxner et al. achieved tracking accuracy in non-static scenarios of 1.9 mm and $1.18°$ at rates of 20-30 Hz using the HoloLens 2 stereo cameras and a single-constraint-at-a-time Extended Kalman Filter [41]. Their approach does not use the IR cameras and relies instead on placing an extra light source on the HoloLens. Loss of tracking due to occlusion of one or more markers is not considered.

Iqbal et al. again used the IR cameras to track spherical markers, co-registered with an external optical tracker, with an accuracy of 2.03 mm and $1.12°$ [43]. The measurement rate is not reported. Furthermore, the HoloLens system is not robust to marker occlusion, though the external tracker can compensate if it sees the markers. Finally, Martin-Gomez et al. described a similar approach and present a very thorough analysis of the stability and accuracy [44]. Their implementation includes the ability to define tool geometries on the fly and track multiple tools. The approach achieves accuracies of 0.09-0.42 mm and $0.80°$ on limited, single-axis motions, and a measurement rate of 172.37 Hz when run offline. In practice, the sensor rate is up to 45 Hz, which limits the measurements. The tracking algorithm is run externally on a very powerful PC, not on the HoloLens 2 itself, which can be advantageous in operating rooms, but is often not practical for remote guidance or teleoperation scenarios. Some of the presented approaches to marker recognition and point matching would thus be difficult with the limited computational power of a HoloLens 2. Robust tracking through IMUs or other approaches is not presented.

To our knowledge, no paper has combined the most modern Kalman filtering approaches with the practical utility of the MR headset-based tracking, nor addressed the concern of marker occlusion in HoloLens-based tracking. Furthermore, the filtering approaches described above use the output pose from a commercial pose sensor while in the HoloLens 2 only the individual marker locations are available. Before computing the pose, point correspondences must be determined, and phantom points caused by noise or missing points caused by occlusion must be handled. While point correspondence algorithms are explored in stereo matching [45], [46], these rely on having relatively unique descriptors such as SIFT [47], and large numbers of points for approaches such as RANSAC [48], [49]. Point cloud registration similarly uses large numbers of points for various statistical methods [48], [50]–[52]. However, we have few points and all are identical, so these approaches are not applicable. A method based on the comparison of inter-point distances among the measured points and among the known geometry is mentioned but not described in [43], [53]. Gsaxner et al. describe a similar approach using minimum-weight matching [41]. No fall-back is presented in these cases, in case the initial approach fails.

We therefore set out to develop a HoloLens 2-based 6-DOF pose tracking system that is more robust to marker occlusion, matches previously reported accuracies, and is independent of external hardware. To achieve this, we use the IR cameras of the HoloLens 2 and describe in detail an algorithm for more robust point correspondence. This is also based on inter-point Euclidean distance comparison, but uses a novel voting-based method that is robust to noise and can assign just the subset of points that is confidently known, unlike optimization-based approaches [41]. Several subsequent steps and back-up methods are presented which can fill in the remaining points and find matches even if the initial attempt failed. A square-root UKF is used for pose estimation and optionally to fuse data from an inertial measurement unit (IMU) which may be installed on the tracked tool. In many cases, such as in point-of-care ultrasound (POCUS) probes or devices instrumented with force sensors, IMUs are already included so no additional hardware is required. This gives the tracking system robustness. Furthermore, the described SR-UKF model is to our knowledge novel for this application as it tracks not the pose directly, but rather the individual marker positions. In this way, it is more suited to systems like the HoloLens 2 where the marker positions are the input instead of the pose itself. The pose that is subsequently computed from these measurements is guaranteed to satisfy the constraint that the markers form a rigid body. In the case of occluded markers, the measurement update step can make use of a subset of the markers rather than losing the whole measurement frame because no pose could be computed directly. This is only possible if point correspondences exist.

In the following sections, the measurement system is described from hardware and sensor acquisition to pre-processing, and finally the basics of the SR-UKF are explained and applied to our system. Mathematical details are expanded upon in the Appendix. In Sections II-F and III respectively, tests are described to evaluate the system, and the results are presented. Finally, simulation code and the HoloLens 2 implementation are available in Section VI.

II. Methods

A. Pose Tracking Pipeline

The overall processing pipeline for HoloLens 2-based pose tracking is enumerated below. Each step and algorithm is detailed in the following sections.

1) Determine potential marker positions with the HoloLens 2 sensors (Section II-B)
2) Remove outliers or phantom markers and find clear point correspondences (Section II-D, Algorithm 2)
3) Find remaining point correspondences (Section II-D, Algorithm 3)

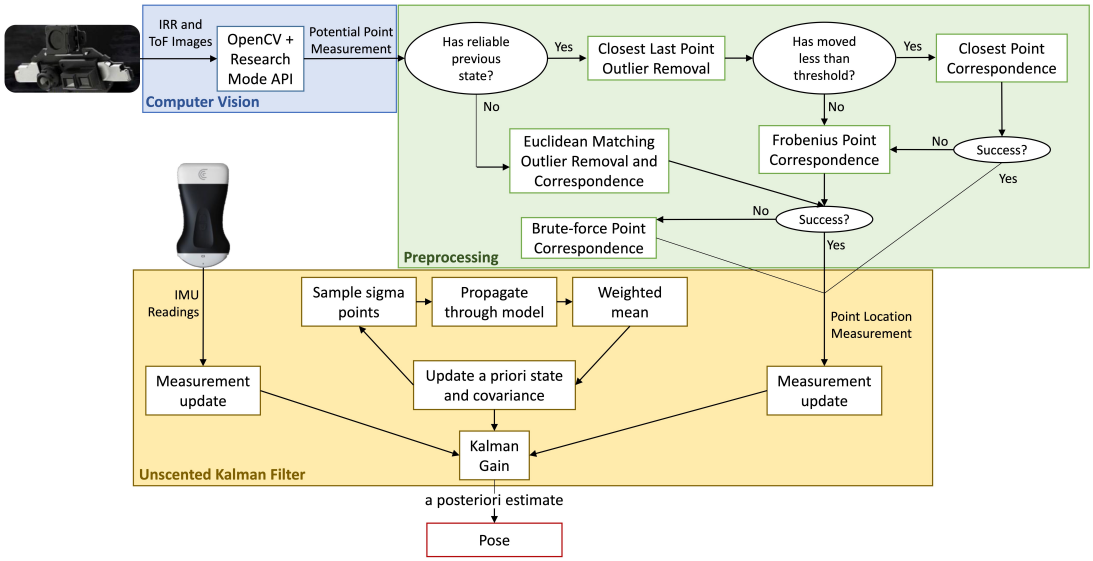

Fig. 1. Flowchart of pose estimation algorithm. Most of the steps are outlined in one of the below algorithms. In our case, however, the question of reliable previous state was always negative as the frame rate was not high enough to match to previous points with good accuracy.

4) SR-UKF measurement update (Section II-E, Algorithm 5)
5) Compute the pose from the SR-UKF state (Section II-C, Algorithm 4)

In parallel, the SR-UKF is running and performing state updates at a higher rate. Additionally, the IMU readings come in at a higher rate and perform their own measurement updates. The entire process is illustrated in Fig. 1.

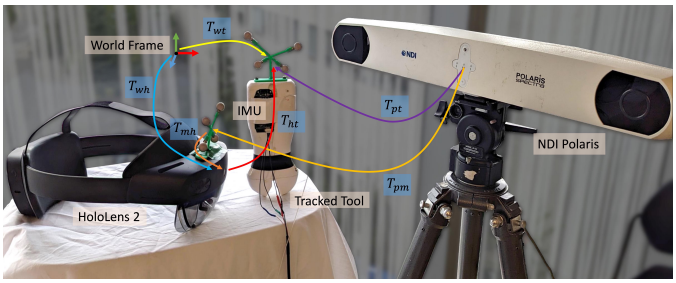

Fig. 2. Hardware setup for tracking and evaluation. The relevant transforms are shown, as is NDI Polaris Spectra optical tracker, the HoloLens 2 with IR markers for registration with the Polaris, and the tracked tool, in this case a dummy ultrasound probe, with IMU and IR markers. The world frame is held stationary in the environment by the HoloLens SLAM functionality.

B. Hardware and Measurement

The only hardware required for this system is the Microsoft HoloLens 2 and an arrangement of IR-reflective markers. For this paper, four IR markers were arranged spatially in a rigid body such that the distance between each pair of markers was unique. The reason for this is explained in Section II-D. For pose calculation, at least three markers are required, but more may be used for better results. The rigid body was designed in SolidWorks CAD software and 3D-printed, and the known geometry from the CAD model was used in the tracking algorithm. In our application of interest, we need accurate tracking of an ultrasound transducer, so, without loss of generality of the approach, the markers were attached to a 3D-printed dummy ultrasound probe in the shape of a Clarius C3HD 3, as shown in Fig. 2.

The HoloLens 2 has two pairs of stereo greyscale cameras for SLAM as well as a front-facing depth camera. The depth camera can operate in two modes: articulated hand tracking (AHAT - 45FPS, up to 1m from device), or long-throw (1-5 FPS, for spatial mapping). Due to its better rate and accuracy, AHAT mode was used. The depth camera returns a ToF depth image and an IR reflectivity image, both of which can be accessed via the HoloLens 2 Research Mode APIs [54]. Custom extrinsic parameters were computed separately after finding that the ones in the API were inaccurate for our device.

For testing of the IMU fusion, a custom printed circuit board for an ultrasound probe force sensor [16] with built-in IMU was placed on the dummy probe. The gyroscope readings were communicated to a desktop PC via a UART to USB converter, and sent to the HoloLens over a WebSocket connection with minimal delay (< 10 ms) at approximately 60 Hz (the frame-rate of the Unity application on the HoloLens 2).

The primary steps of the marker recognition and 3D position extraction are shown in Fig. 3. Binary thresholding followed by contour detection is used to find the markers. The detected contours are filtered for circularity by determining the contour area, a, and circumference, c, and computing $k = 4\pi a/c^2$. A k value close to 1 is a perfect circle, and a threshold is set to remove non-circular contours. The marker locations in the image plane are determined as the centroids of the closed contours. The depth values are then obtained by querying those coordinates in the depth image. Using the camera intrinsic parameters and the depth values, the image plane coordinates are transformed to 3D points, and finally to world-frame coordinates through the extrinsic parameters and HoloLens 2 SLAM capabilities. The method was found to be very robust, though occasional "phantom" markers are also measured, due

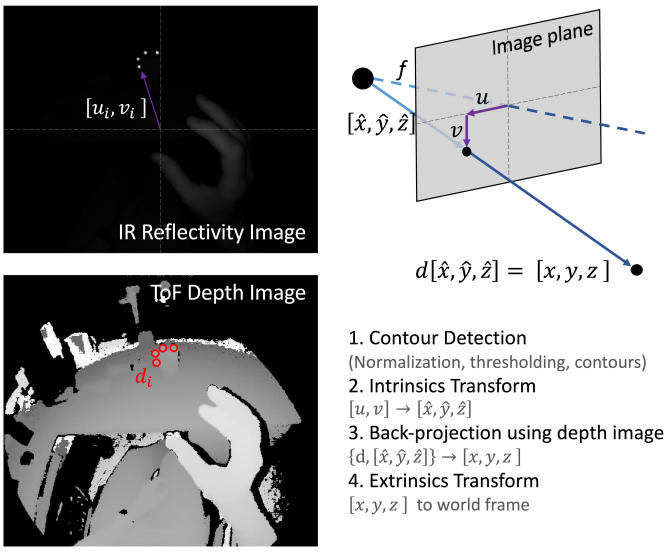

Fig. 3. Computer vision process, showing IR reflectivity and ToF depth images, and the steps for extraction of 3D point locations in the world frame from the two images. The contour detection is carried out using OpenCV on the reflectivity image, before the depth values from those $[u, v]$ coordinates are extracted from the depth image.

to reflection from random items in the environment. These phantom markers are later removed in the pre-processing.
In the next sections, the following notation is used.

$$a = \text{Scalar}$$
$$\boldsymbol{a} = \text{Column vector}$$
$$A = \text{Matrix}$$
$$a_k = a \text{ at time step } k$$

C. Pose Computation

To calculate the tool's pose from the measured points, $\{\boldsymbol{r}^i, i \in [1, n]\}$, we solve the constrained orthogonal Procrustes problem using the Kabsch-Umeyama algorithm [55], shown in Algorithm 4 in the Appendix. This is computationally efficient and guarantees that the output is a rotation matrix. It relies on knowing the geometry of the tracked points a priori, and calculates the least squares solution which best maps the measured points to the known ones. To do this, however, we must first find point correspondences, as described below.

D. Pre-processing and Point Correspondence

In order to calculate 6-DOF pose from the marker locations, we find the homogeneous transform that most closely aligns the known rigid geometry with the measured marker positions. However, to do so, we must first know which known point corresponds to which measured point. This process is called finding point correspondences and is non-trivial as it is not possible to differentiate between the spherical IR markers. Additionally, there may be points missing, if a marker is occluded, or there may be extra phantom points as explained above, or likely some combination of the two. Therefore, the first step is to remove phantom points and perform point correspondence.

There are several options for outlier point removal:

- Random sample concensus (RANSAC) [56]; This relies on a relatively large number of points to make the random sampling effective. However, in our case there may not be many more inliers than outliers (we have about 2-4 inliers and usually 0-2 outliers). Also, this does not take advantage of our knowledge of rigid body geometry.
- Brute Force: try every permutation of the measured points, calculate the pose, determine the error, and choose the permutation with the minimum error. This is slow and does not take advantage of any prior knowledge we may have of the the last state of the markers, from which they presumably have not moved much if the measurement rate is sufficiently high.
- Closest Last Point: Compare the current measured points to the previous state, and keep the closest matches. This assumes the motion is slow relative to the measurement rate. Points that are further away than threshold from the previous points can be marked as outliers.
- Euclidean Distance Comparison: Calculate the distances between all the measured points, and compare them to the distances between the known points. This relies on designing the geometry so each inter-point distance is unique. This is shown in Algorithm 2.

Closest Previous Point Matching

The naïve closest previous point method operates on the assumption that since the last accurate state update, the object has undergone a small enough motion that a given point is still closer to its own previous position than to the previous position of any of the other points. If this is not true, this algorithm easily assigns the same index to more than one measured point. We can make the algorithm more robust by following an optimization-based approach. We first assume that all outlier points have been removed (see Algorithm 4 for example), and the measured points are placed in the columns of a matrix $X \in \mathbb{R}^{3\times k}$. Let $Y \in \mathbb{R}^{3\times k}$ be a matrix whose columns are the previously measured points. To determine point correspondences, we can solve the following optimization problem:

$$P_y = \underset{P_y}{\text{argmin}}\{\|X - YP_y\|_F^2\} \quad (1)$$
$$\text{s.t. } P_y \text{ is a permutation matrix,} \quad (2)$$

where $\|\|_F$ is the Frobenius norm. Solving this optimization problem analytically using the Frobenius inner product (See Appendix), we arrive at Algorithm 1.

Though effective in simulation, this method is unfortunately ineffective in practice in our system as the depth sensor frame rate is sometimes not sufficiently fast relative to the tool motions, so the current and previous pose do not correspond closely. It could, however, be effective with faster hardware.

Euclidean Distance Matching

Closest-previous-point methods were found to be relatively ineffective with the HoloLens 2. Sometimes, also, the algorithm

Algorithm 1 Frobenius norm-based Point Correspondence

Generate a list of all permutations of $\{1, 2, \ldots, k\}$ (do this offline beforehand)
Given the set of measured points, $\{\boldsymbol{r}^i\}$, and previous points, $\{\boldsymbol{y}^i\}$ for $i \in [1, n]$
$X \leftarrow \begin{bmatrix}\boldsymbol{r}^1 & \boldsymbol{r}^2 & \ldots & \boldsymbol{r}^k\end{bmatrix}$, and $Y \leftarrow \begin{bmatrix}\boldsymbol{y}^1 & \boldsymbol{y}^2 & \ldots & \boldsymbol{y}^k\end{bmatrix}$
$Z \leftarrow X^\top Y$
for each permutation $\boldsymbol{p} = \begin{bmatrix}i_1 & i_2 & \ldots & i_k\end{bmatrix}$ **do**
  $e \leftarrow \sum_{j=1}^{k} Z[j, i_j]$
  if e is biggest so far **then**
    Keep permutation $\boldsymbol{p}$
  end if
end for
The point correspondence is the best permutation

may lose track of the points completely, and upon startup of the application, the algorithm has no previous pose to fall back on. For these cases, we need a robust method that does not rely on a priori knowledge. Algorithm 2 presents a very effective Euclidean distance voting-based method of removing outliers which runs in $O(n^2m^2)$ time. Considering that there are four points and the inter-point distance matrices are symmetric, this is only around 128 computations.

The Euclidean method is likely similar to what is mentioned in references [41], [43], [53]. As noted in the Introduction, however, Algorithm 2 is based on voting which adds robustness. Even if, by a highly unlikely coincidence, a phantom point and a measured point have the same distance as a pair of known points, that constitutes only one vote for the phantom point, and it is still unlikely to be included. In practice the phantom points are generally far away and this method works well. Additionally, it has the ability to remove outliers reliably, and assign only the correspondences that are clear. Rather than forcing questionable correspondences through an optimization-based assignment, these can be considered separately after further processing. For example, brute force methods to identify the remaining two or three points become very simple and low-cost.

Brute Force Matching

After much experimentation with slicker solutions, we determined that the brute force method is best in this case, as shown in Algorithm 3. If the noise is small enough, it is guaranteed to find the correct solution. There are $m!$ combinations, and each trial involves carrying out an SVD which is $O(3m^2)$. Since $m = 2,\ 3$, or 4, the $O(m!m^2)$ cost is not very large. This computation is only carried out relatively infrequently if all else fails.

These methods requires at least three points to be measured to provide a unique match and subsequently to compute the pose. For better numerical stability and robustness in the case of partial occlusion, we use four markers. However, with the unscented Kalman filter described below, it is possible to provide measurement updates with less than three measured markers if correspondences can be determined. This would require the markers to be individually identifiable, for example

Algorithm 2 Euclidean Distance Voting Point Correspondence

Given the set of measured points, $\{\boldsymbol{r}^i, i \in [1, n]\}$, and known geometry points, $\{\boldsymbol{y}^i, i \in [1, m]\}$

Form a matrix of inter-point Euclidean distances (D)
for i in 1:n **do**
  for j in $(i+1)$:n **do**
    $D[j, i] = D[i, j] \leftarrow \|\boldsymbol{r}^i - \boldsymbol{r}^j\|_2^2$
  end for
end for

Form the same matrix (Y) for the m known points (do this offline beforehand)

Define $V \in \mathbb{Z}^{m \times n}$ with every element equal to 0
Vote on likely matching pairs of points:
for i in 1:m **do**
  for j in $(i+1)$:m **do**
    $(k, l) = \underset{(k,l)}{\text{argmin}}\{|D[k, l] - Y[i, j]|\}$
    $V[i, k]$ += 1, $V[i, l]$ += 1, $V[j, k]$ += 1, $V[j, l]$ += 1
  end for
end for

for i in 1:n **do**
  if $sum(V[:, i]) < threshold$ **then**
    Mark measured point i as an outlier
  else
    $j \leftarrow \text{argmax}\{V[\ell, i]\}$ and $c \leftarrow \max\{V[:, i]\}$
    if $c > \underset{\ell}{threshold}$ **then**
      Measured point i likely corresponds with geometry point j
    else
      Flag measured point i for further processing
    end if
  end if
end for

Algorithm 3 Brute Force Point Correspondence

Given the set of measured points, $\{\boldsymbol{r}^i, i \in [1, n]\}$, and known geometry points, $\{\boldsymbol{y}^i, i \in [1, m]\}$
Generate a list of all permutations of $\{1, 2, 3, \ldots, n\}$ (do this offline beforehand)
$X \leftarrow \begin{bmatrix}\boldsymbol{r}^1 & \boldsymbol{r}^2 & \ldots & \boldsymbol{r}^n\end{bmatrix}$, and $Y \leftarrow \begin{bmatrix}\boldsymbol{y}^1 & \boldsymbol{y}^2 & \ldots & \boldsymbol{y}^m\end{bmatrix}$
for each permutation **do**
  Rearrange the columns of Y according to the permutation to create Y'
  $R, \bar{\boldsymbol{r}} \leftarrow kabsch(X, Y')$ (See Algorithm 4)
  $e \leftarrow \sum_{i=1}^{n} \|\bar{\boldsymbol{r}} + R\boldsymbol{\ell}^i - \boldsymbol{r}^i\|_2^2$
  if e is smallest so far **then**
    Keep permutation
  end if
end for
The point correspondence is the best permutation

by making them active with IR LEDs. With the current markers and point correspondence methods described above, at least three markers are required.

E. Kalman Filter

To improve the tracking, both by reducing noise, interpolating between point measurements for higher readout rates, and for integrating external IMU data, a Kalman filter or particle filter can be used. Due to its Monte Carlo sampling, the particle filter is more computationally intensive, so we use a Kalman filter. As the dynamics are nonlinear, an extended (EKF) or unscented Kalman filter (UKF) is required. EKFs rely on linearization of the dynamics about the operating point, which is less accurate and requires explicit computation of a Jacobian matrix [57], [58]. We therefore present an SR-UKF for the pose tracking.

In order to track the object, we follow the n markers rather than the pose of the object itself. This alleviates the issues associated with propagating orientation. Euler angles and similar representations have singularities, and rotation matrices quickly lose orthonormality. Quaternions are effective, but tracking marker positions also lets us enforce the rigid body constraint, and give partial measurement updates, as described in the Introduction. Thus, the marker positions, $\boldsymbol{r}^i$, are tracked, as are the angular and linear velocity of the object's centroid, $\boldsymbol{c}$: $\boldsymbol{\omega}$ and $\dot{\boldsymbol{c}}$. The state vector, $\boldsymbol{x}_k$ is then:

$$\begin{aligned}\boldsymbol{x}_k &= \begin{bmatrix}(\boldsymbol{r}_k^1)^\top & \dots & (\boldsymbol{r}_k^n)^\top & \boldsymbol{\omega}_k^\top & \dot{\boldsymbol{c}}_k^\top\end{bmatrix}^\top \in \mathbb{R}^{N\times 1}, \\ N &= 3n+6\end{aligned} \tag{3}$$

The marker locations are measured using the computer vision system. The angular velocity can also be measured using an IMU on the object, as can the linear acceleration, and thus through numerical integration, the linear velocity. However, the latter tends to drift significantly, so only the angular velocity is used. The output, $\boldsymbol{y}_k$, is thus simply:

$$\boldsymbol{y}_k = \begin{bmatrix}(\boldsymbol{r}_k^1)^\top & \dots & (\boldsymbol{r}_k^n)^\top & \boldsymbol{\omega}_k^\top\end{bmatrix}^\top \tag{4}$$

The state and output equations are given by nonlinear, vector-valued functions $\boldsymbol{f}(\boldsymbol{x}_k)$ and $\boldsymbol{h}(\boldsymbol{y}_k)$ respectively, with additive process and measurement noise vectors $\boldsymbol{v}_k$ and $\boldsymbol{w}_k$ respectively:

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k) + \boldsymbol{v}_k \tag{5}$$

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{w}_k \tag{6}$$

The state's evolution is given by:

$$\begin{cases}\boldsymbol{r}_{k+1}^i & = \boldsymbol{r}_k^i + dt\left(\dot{\boldsymbol{c}}_k + [\boldsymbol{\omega}_k]_\times R_k\boldsymbol{\ell}^i\right) + \\ & \left\{\frac{1}{2}dt^2\left[\ddot{\boldsymbol{c}}_k + ([\boldsymbol{\omega}_k]_\times^2 + [\boldsymbol{\alpha}_k]_\times)R_k\boldsymbol{\ell}^i\right]\right\} \text{ for } i\in[1,n] \\ \boldsymbol{\omega}_{k+1} & = \boldsymbol{\omega}_k + \{dt\boldsymbol{\alpha}_k\} \\ \dot{\boldsymbol{c}}_{k+1} & = \dot{\boldsymbol{c}}_k + \{dt\ddot{\boldsymbol{c}}_k\}\end{cases} \tag{7}$$

Where R_k is the rotation matrix, described below, and $\boldsymbol{\ell}^i$ is the position of the i^{th} marker in the rigid body frame. This state equation is derived in the Appendix and is specifically formulated to explicitly enforce the rigid body constraint. Rather than computing the marker positions based on individual velocities, they are computed from the velocity of the rigid body and the known relative transforms of the markers from the centroid. In this way we maintain a constant rigid geometry.

In the above expression, the terms in curly brackets are not explicitly calculated, but rather are modeled as noise by placing them in $\boldsymbol{v}_k$ and assuming that the linear and rotational acceleration are small and varying normally, as is commonly done. i.e.:

$$\ddot{\boldsymbol{c}}_k \sim \boldsymbol{N}(0,\Sigma_c^2) \tag{8}$$

$$\boldsymbol{\alpha}_k \sim \boldsymbol{N}(0,\Sigma_\alpha^2) \tag{9}$$

Assuming the axes of the accelerations are independent and identically distributed (iid), $\Sigma_c = \sigma_c I_3$ and $\Sigma_\alpha = \sigma_\alpha I_3$. In this way, we define the state equation as:

$$\boldsymbol{f}(\boldsymbol{x}_k) = \begin{cases}\boldsymbol{r}_{k+1}^i & = \boldsymbol{r}_k^i + dt\left(\dot{\boldsymbol{c}}_k + [\boldsymbol{\omega}_k]_\times R_k\boldsymbol{\ell}^i\right) \text{ for } i\in[1,n] \\ \boldsymbol{\omega}_{k+1} & = \boldsymbol{\omega}_k \\ \dot{\boldsymbol{c}}_{k+1} & = \dot{\boldsymbol{c}}_k\end{cases} \tag{10}$$

And the process noise is:

$$\boldsymbol{v}_k = \begin{bmatrix}\frac{1}{2}dt^2\left[\ddot{\boldsymbol{c}}_k + ([\boldsymbol{\omega}_k]_\times^2 + [\boldsymbol{\alpha}_k]_\times)R_k\boldsymbol{\ell}^1\right] \\ \vdots \\ \frac{1}{2}dt^2\left[\ddot{\boldsymbol{c}}_k + ([\boldsymbol{\omega}_k]_\times^2 + [\boldsymbol{\alpha}_k]_\times)R_k\boldsymbol{\ell}^n\right] \\ dt\boldsymbol{\alpha}_k \\ dt\ddot{\boldsymbol{c}}_k\end{bmatrix} \tag{11}$$

We assume that the measurements of the point positions are all iid normally distributed with variance $\Sigma_r^2 = \sigma_r^2 I_3$, as are the gyroscope measurements from the IMU: $\Sigma_\omega^2 = \sigma_\omega^2 I_3$. Then the measurement noise vector is

$$\boldsymbol{w}_k = \begin{bmatrix}(\boldsymbol{N}(0,\Sigma_r)^1)^\top & \dots & (\boldsymbol{N}(0,\Sigma_r)^n)^\top & \boldsymbol{N}(0,\Sigma_\omega)^\top\end{bmatrix}^\top \tag{12}$$

In the above, the rotation matrix, R_k, is used though it is not included in the state. Rather, the computation of the rotation matrix is part of the state update, and it is recomputed every step to avoid bad scaling. This is completed using Algorithm 4, as described in Section II-C.

From these noise vectors, covariance matrices are computed, and the presented state update and output equations are used to propagate the state and covariance in the UKF. Measurements from the marker tracking and an IMU provide periodic measurement updates. The details of this unscented Kalman filter and implementation on the HoloLens 2 are given in the Appendix.

F. Testing

To develop and test the efficacy of the Kalman filter, two methods were used. First, the system was simulated in MATLAB using Simulink. A series of random accelerations and angular accelerations was applied to the known geometry points and integrated to obtain a realistic motion. Simulated IMU and marker position measurements were extracted from these signals and zero-mean Gaussian noise was added to

match the real system. Phantom points were added with probability P_p to each measurement, and markers were removed ("occluded") with probability P_o. The algorithms were applied to the simulated data, sampling the IMU signal at 2 to 3 times the rate of the point measurements, with probability P_2 and $P_3 = 1 - P_2$, respectively. This reflects the difference in the computer vision frame rate and IMU measurement rate experienced in our real system.

The second test method was using the actual hardware (Fig. 2). A Microsoft HoloLens 2 running a Unity application performed all the tracking and processing, obtaining the homogeneous transforms T_{ht} in Fig. 2, while an external NDI Polaris Spectra optical tracker was used for the ground truth (T_{pt}). An ultrasound probe-shaped dummy with 3D-printed structure holding four IR reflective markers was held within view of both the HoloLens and the Polaris. Further, a structure with four IR markers was placed on the front of the HoloLens, where it faced the Polaris device, to compute the registration to the Polaris (T_{pm}). The transform from HoloLens frame to world frame (T_{wh}) is enabled by the HoloLens spatial mapping and localization. To compare the measurements, T_{ht}, and ground truth, T_{pt}, it is necessary to compute the transform from the HoloLens frame to the markers placed on the HoloLens, indicated as T_{mh}. Using the relation $T_{ht} = T_{pt}T_{pm}^{-1}T_{mh}^{-1}$, where T_{ht}, T_{pm}, and T_{pt} are measured, T_{mh} was determined using least squares using a measurement of 5000 pose samples. This matrix was subsequently used for all tests.

With the systems thus set up to be as similar as possible, the algorithms were designed in simulation and adjusted and tested on the physical setup. It was found that both systems had nearly identical responses. However, all results shown below are data recorded from the physical system. All data was recorded with timestamps to allow alignment and direct comparison between the HoloLens and Polaris. After aligning the start of the data, it was re-sampled so the HoloLens and NDI measurements aligned in every sample. The error between the two was computed by taking the root-mean-square (RMS) of the element-wise difference between the two signals.

To compute lag between the signals, the Polaris data was read in programmatically while the same program periodically exchanged timestamps over a WebSocket connection with the HoloLens, as described previously [59]. This enabled clock synchronization between the HoloLens and the PC connected to the Polaris. The lag of the HoloLens measurements could then be computed by finding the time delay that maximized the normalized cross correlation between the synchronized HoloLens and Polaris signals.

III. Results

Without Kalman Filter

First the pose tracking system was tested alone, without a Kalman filter. Compared to the NDI optical tracker, the mean positional RMS error was 3.01 mm and the mean RMS Euler angle representation of the rotation error was 1.75°. Both position and rotation have a mean error of 0.00 mm in all axes. The position and quaternion tracking is shown in Fig. 4.

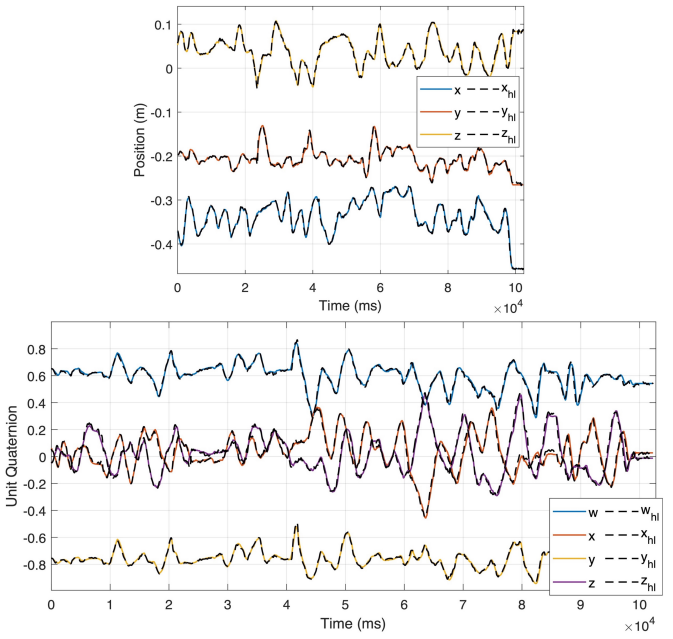

Fig. 4. Tracking without Kalman filter: Upper plot shows position (x, y, z) and lower plot shows rotation (quaternion w, x, y, z) with very close correspondence.

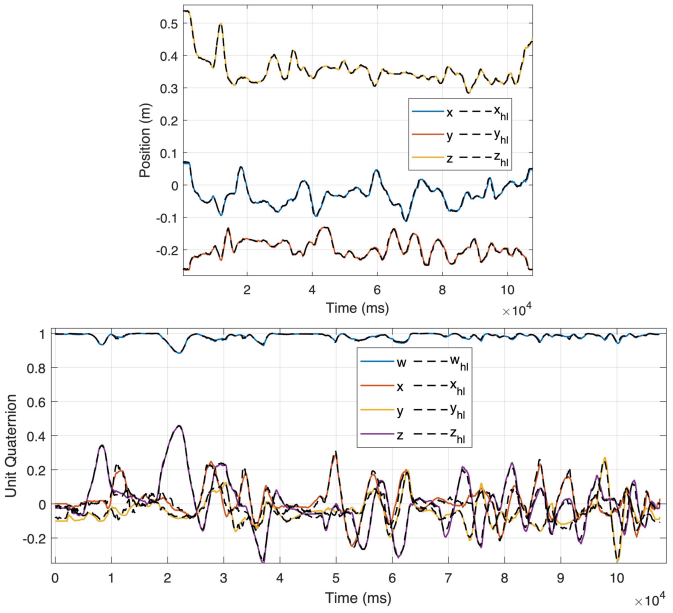

Fig. 5. Tracking with UKF: Upper plot shows position (x, y, z) and lower plot shows rotation (quaternion w, x, y, z) with very close correspondence.

Kalman Filter without IMU

The Kalman filter was then tested in isolation, without receiving IMU data. A series of random translations and rotations was applied by hand, tracked, and recorded as explained above. Compared to the NDI optical tracker, the HoloLens tracking achieved a positional RMS error of 1.77 mm. The RMS angular error using the Euler representation of the quaternions was 1.51°. Both position and rotation have mean error of 0.00 mm in all axes. The position and quaternion tracking is shown in Fig. 5.

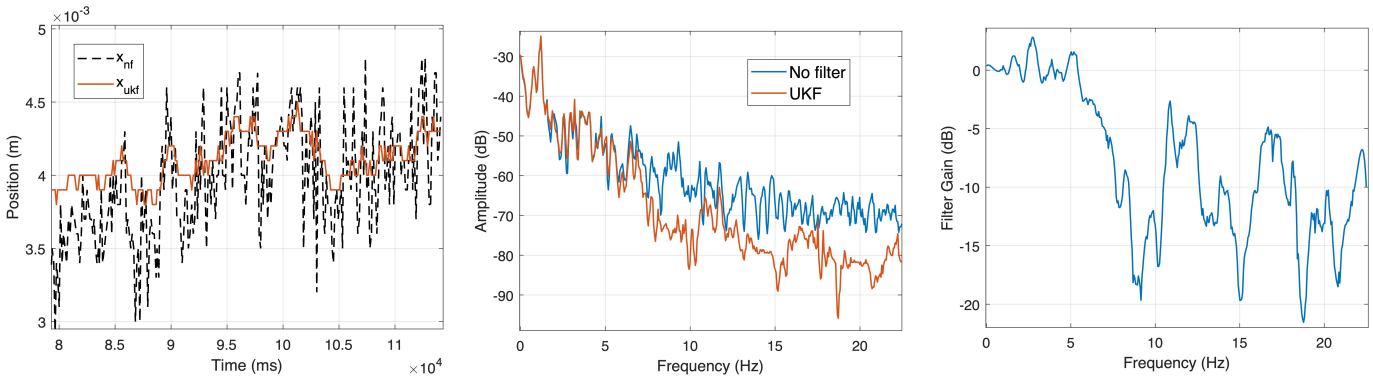

Fig. 6. Noise characteristics of Kalman filter. (Left) Zoomed in signal showing noise rejection characteristics of Kalman filter. x_{nf} is the x position without filter, and x_{ukf} is with the unscented Kalman filter. (Middle) Discrete Fourier transform of filtered and unfiltered signals, showing decreased amplitude in higher frequencies (noise). (Right) Gain between filtered and unfiltered signals, showing low-pass characteristics.

This reflects a lower variance in the error distribution than without the filtering, both centred about zero. From observation, it was clearly more noisy without the Kalman filter, though also much less lag was present. The Kalman filter allows us to interpolate between successive measurements using the dynamic model, so we can have a higher measurement rate which is important for teleoperation, but the smoothing introduces lag. The lag values are measured and discussed later. We can change the smoothing by tuning the σ values, with a trade-off between noise and lag. The noise rejection characteristics of the filter are shown in Fig. 6, from a zoomed-in portion of signal in Fig. 5.

Beyond accuracy, a key aspect of the tracking system is robustness to motion of the object. It should maintain tracking even if the object is rotated or translated substantially or the markers are partially or briefly occluded. To test this, a series of large motions and many full 360° rotations in all axes was carried out, with markers periodically occluded briefly. The measured poses are plotted in Fig. 7. During the test, the vision system successfully found the correct pose in 98.63% of the 10000 measured frames. The test was repeated with one of the markers removed, and the correct pose was found in 96.04% of the 10000 measured frames. These results indicate that the tracking is very robust to occlusion during any motion, even without IMU fusion.

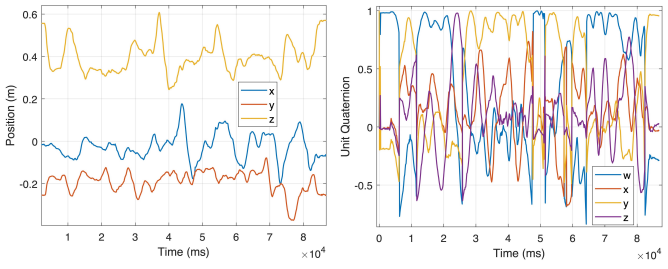

Fig. 7. Large motions and especially rotations used while obtaining 98.63% successful calculation of pose.

Kalman Filter with IMU

As seen from the results so far, no gyroscope is needed during normal operation. The vision-based tracking with Kalman filter works well enough on its own. Even occlusions bad enough to make the vision-based system lose tracking are very rare and usually only lead to a few dropped frames at most. However, in case the operator has to hold the probe in a position that results in such an occlusion for an extended period of time, having a gyroscope can add robustness as it will continue to track. Furthermore, the IMU sampling rate can be higher than the computer vision system, allowing for more frequent measurement updates. This is the case in our system, which has two to three IMU readings during each computer vision frame.

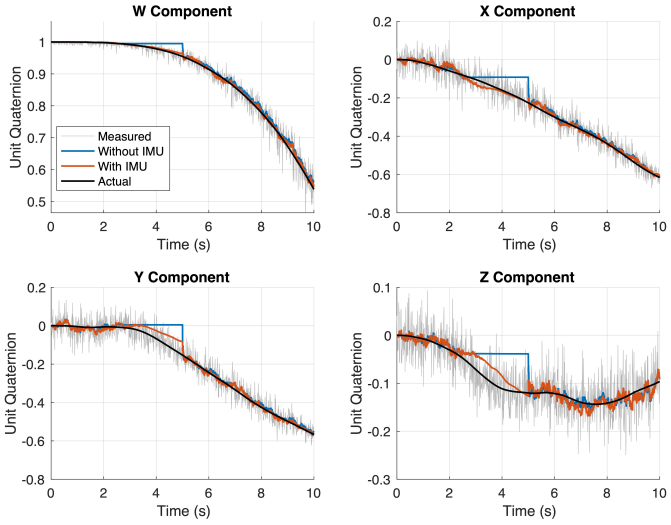

Fig. 8. Orientation tracking with and without IMU. Though the tracking is not noticeably better with IMU, it allows approximate orientation tracking to continue when the markers are occluded. This is seen between 3 and 5 seconds, when the markers were purposely occluded.

From tests it was found that typical hand motion during tracking was slow enough that the sampling rate of the vision system was sufficient. Indeed, the human hand bandwidth for applying forces is approximately 7 Hz [60], and the vision system runs at 22 Hz (see below). The vision system is thus beyond the Nyquist rate for hand motions. As a result, adding IMU data at a higher rate does not noticeably improve

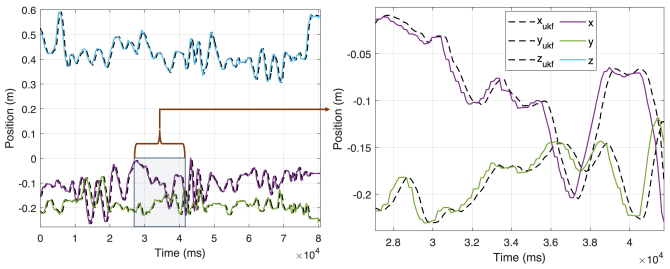

Fig. 9. UKF lag between the UKF signal (x_{ukf}, y_{ukf}, z_{ukf}) and the unfiltered signal (x,y,z). The full signal (left), and a zoomed in section (right) are shown. The average lag is 199.50 ± 4.65 ms with the chosen UKF parameters.

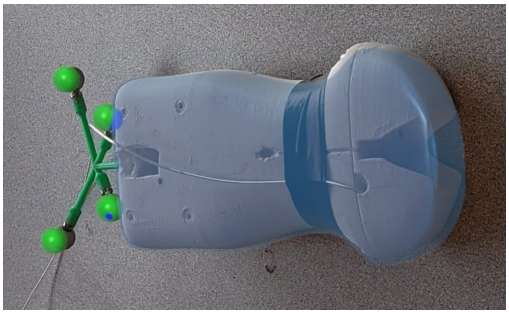

Fig. 10. MR Capture from HoloLens 2 of the dummy ultrasound probe with IR markers showing successful pose tracking. The green spheres are the measured marker positions while the blue ones are the output of the UKF, and the blue virtual probe shows the resultant calculated pose. This preview can be disabled during actual operation.

tracking accuracy or speed. On the other hand, when the markers are occluded manually for an extended time, the IMU fusion drastically improves tracking of orientation during the occlusion. This is illustrated in Fig. 8.

Time Complexity and Lag

Each image and subsequent point measurement undergoes substantial processing before a pose can be estimated. The average time taken for the major steps of this process are outlined in Table I. In total, poses are captured from the CV system every 45.98 ± 7.67 ms, or at 21.75 ± 3.11 Hz. As explained above, this is well above the maximum bandwidth for hand forces. At the chosen uncertainty values, σ, the lag between the UKF and the no-filter tracking was 199.50 ± 4.65 ms. This is shown in Fig. 9. The lag of the no-filter tracking was determined by moving the probe along a track in a pre-programmed manner, so its position was known at all times. Exchanging timestamps over a WebSocket server to synchronize the clocks, the lag of the filter-less tracking with respect to the object's motion was found to be 25.36 ± 6.51 ms. In total, then, the UKF lags object motion by between 25.36 ± 6.51 ms (with no smoothing) and 224.86 ± 11.16 ms (with the smoothing presented in this paper), or more with more smoothing. The exact value depends on the priorities of the application.

TABLE I

TIMING OF KEY SUBROUTINES, NOT IN CHRONOLOGICAL OR HIERARCHICAL ORDER. THE PRE-PROCESSING ROW INCLUDES THE ENTIRE PRE-PROCESSING PIPELINE BETWEEN THE COMPUTER VISION STEP AND FEEDING THE POINTS TO THE KALMAN FILTER, INCLUDING GEOMETRIC TRANSFORMS, OUTLIER REMOVAL, AND POINT CORRESPONDENCE. THE VARIATION IS DUE TO THE INFREQUENT USE OF BRUTE FORCE MATCHING, WHICH TAKES LONGER. BRUTE FORCE MATCHING IN ITSELF VARIES IN SPEED DEPENDING ON HOW MANY MARKERS ARE BEING MATCHED.

Subroutine	Average Time (ms)
Pre-processing	0.104 ± 0.293
Euclidean Matching	0.018 ± 0.005
Brute Force Matching	0.305 ± 0.270
Kabsch Algorithm	0.055 ± 0.005
UKF Measurement Update	2.231 ± 0.623
UKF State Update	8.856 ± 1.974
Contour detection	12.765 ± 5.545

IV. DISCUSSION

In this paper we have introduced an IR marker-based 6-DOF pose tracking system for the HoloLens 2. The system includes new, efficient point correspondence methods and a novel SR-UKF with IMU fusion which together give accurate, fast, robust tracking. The pose tracking is shown in Fig. 10. The primary results are compared to previous work in Table II.

For the simulated and physical systems, all the exact same algorithms and parameters were used, written once in MATLAB and once in C#. Implemented in C#, however, we found that some of the methods could be numerically unstable. Even when the MATLAB implementation produced a reasonable output given the same inputs, C# sometimes gave NaN or Inf values in the Cholesky update steps. To handle these cases, the failed Cholesky update was simply ignored, and the value from before the update was used. We found that this did not affect the performance of the filter. In 25000 up/downdate computations, it failed 16 times (0.06%).

The approach of tracking points also allows for more intelligent measurement updates in the case of multiple occluded markers. If the markers were individually identifiable, for example by comparing their positions between frames with a higher frame rate, or by using different colors or different frequency active markers, etc., the measurement update step could still be performed with just one or two markers visible, thus giving strong constraints on the probe position and orientation. Unfortunately, with the current markers and point correspondence methods, at least three markers are required.

The IMU fusion can maintain approximate orientation tracking during extended occlusion of two or more markers. However, it cannot improve position tracking without using accelerometer data. This is notoriously inaccurate for position tracking due to the two integrations required. However, in some respects orientation tracking is most important. For sensors such as force sensors on the tracked device, only the orientation is needed to convert the forces to a base coordinate system.

Future work will include improving the measurement rate of the system. Currently, the HoloLens application is implemented on Unity, which runs at only 60 Hz, thus imposing a hard limit on the tracking speed and adding a large overhead.

TABLE II
COMPARISON OF PERFORMANCE BETWEEN HOLOLENS-BASED OBJECT TRACKING SYSTEMS. OUR SYSTEM HAS SIMILAR ACCURACY, HIGHER SPEED, BETTER ROBUSTNESS, AND DOES NOT REQUIRE MODIFYING THE HOLOLENS.

Paper	Pose Accuracy	Speed (Hz)	Robust	Hardware
Kunz et al. [42]	Not reported	22-30 Hz	No	HoloLens 1
Gsaxner et al. [41]	1.9mm, 1.18°	20-30 Hz	No	HoloLens 2 + IR LEDs
Iqbal et al. [43]	2.0mm, 1.12°	Not Reported	With external optical tracker	HoloLens 2
This paper	1.8mm, 1.51°	22-75 Hz	With IMU	HoloLens 2

Speed could be improved by writing the whole application in C++. Another limitation of the HoloLens application is that the Research Mode APIs are only compatible with an ARM64 processor, while some useful libraries for MR applications, such as WebRTC for efficient communication, are only available for 32-bit ARM processors. This discrepancy should be rectified if possible.

An alternative method for the computer vision step is simple blob detection, with filtering for convexity, area, circularity, and brightness of the blobs with various thresholds. This was equally effective as contour detection, but took $50-80$ ms for the detection step while contour detection took $5-13$ ms. Previous works have used blob detection [41], [43]. These works reported update rates based on the speed of the subroutines rather than the system as a whole. Using this approach, the worst-case running time of one pose computation would be contour detection + pre-processing + Euclidean matching + brute force matching + the Kabsch algorithm, which from Table I sums to 13.247 ms, or an update rate of 75.49 Hz. This is of course not achieved, as the depth sensor frame rate is only up to 45 Hz, and intermediate steps of extracting the camera pixels, etc. add to the latency.

The accuracy of the results, and especially the computed rotations, could likely be improved by utilizing a better marker geometry. Optimization-based methods for generating the geometry exist [61] and may improve the results. Furthermore, the cameras are unsynchronized, and the NDI Polaris Spectra "ground truth" that we used also has an error of approximately 1 mm and 1°.

V. CONCLUSION

In this paper we have presented a 6-DOF object pose tracking algorithm and implementation using IR markers and sensors that achieves accuracies of 1.77 mm and $1.51°$, and a practical measurement rate of 21.75 Hz. The theoretically achievable rate is 75.49 Hz with faster camera hardware, and specific means to improve the accuracy are outlined for future work. The algorithm is resistant to loss of tracking, with 98.63% success rate in calculating pose from measurements taken during large motions. In the event of extended loss of visual tracking, IMU fusion is included with an unscented Kalman filter to continue tracking the orientation. The accuracy, speed, and robustness of the method enable mixed reality-based teleoperation and high-fidelity feedback for remote guidance or human-robot interaction systems.

VI. SUPPLEMENTARY MATERIAL

The simulation code and HoloLens 2 implementation are available here: github.com/dgblack/hl2ObjTracking

VII. APPENDIX

We introduce the following notation:

$\boldsymbol{a}[i] = i^{th}$ element of $\boldsymbol{a}$ (indexing starting at 1)

$$[\boldsymbol{a}]_\times = \begin{bmatrix} 0 & -\boldsymbol{a}^3 & \boldsymbol{a}^2 \\ \boldsymbol{a}^3 & 0 & -\boldsymbol{a}^1 \\ -\boldsymbol{a}^2 & \boldsymbol{a}^1 & 0 \end{bmatrix}$$

$A[:,i] = i^{th}$ column of A, and $A[i,:] = i^{th}$ row of A

$$A + \boldsymbol{b} = \begin{bmatrix} A[:,1] + \boldsymbol{b} & A[:,2] + \boldsymbol{b} & \dots & A[:,n] + \boldsymbol{b} \end{bmatrix}$$

$$\boldsymbol{f}(A) = \begin{bmatrix} \boldsymbol{f}(A[:,1]) & \boldsymbol{f}(A[:,2]) & \dots & \boldsymbol{f}(A[:,n]) \end{bmatrix}$$

$a_k = a$ at time step k

$I_n = n \times n$ identity matrix

$0_n = n \times n$ zero matrix

A. Pose Computation from Points:

The Kabsch-Umeyma algorithm for pose computation from a set of points is given by Algorithm 4.

Algorithm 4 Kabsch-Umeyama Algorithm for Pose Calculation

Given a set of measured points $\{\boldsymbol{r}^{1:n}\}$ in the world frame, and corresponding known points $\{\boldsymbol{y}^{1:n}\}$ in the rigid body frame

Compute and subtract the centroids

$kabsch(\boldsymbol{r}^{1:n}, \boldsymbol{y}^{1:n})$:

$\bar{\boldsymbol{r}} \leftarrow \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{r}^i$, and $\bar{\boldsymbol{y}} \leftarrow \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{y}^i$

$$P \leftarrow \begin{bmatrix} (\boldsymbol{y}^1 - \bar{\boldsymbol{y}})^\top \\ \dots \\ (\boldsymbol{y}^n - \bar{\boldsymbol{y}})^\top \end{bmatrix} \in \mathbb{R}^{n\times 3}$$

$$Q \leftarrow \begin{bmatrix} (\boldsymbol{r}^1 - \bar{\boldsymbol{r}})^\top \\ \dots \\ (\boldsymbol{r}^n - \bar{\boldsymbol{r}})^\top \end{bmatrix} \in \mathbb{R}^{n\times 3}$$

$H \leftarrow P^\top Q$ (Compute cross-covariance)

$[U, \Sigma, V] \leftarrow \text{SVD}(H)$ (Perform SVD)

$d \leftarrow \text{sign}(\det(VU^\top))$ (Coordinate system handedness)

$$R \leftarrow V \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & d \end{bmatrix} U^\top \text{ (Compute rotation matrix)}$$

Return R, $\bar{\boldsymbol{r}} - \bar{\boldsymbol{y}}$

end $kabsch(\boldsymbol{r}^{1:n}, \boldsymbol{y}^{1:n})$

B. Frobenius Norm-Based Matching

Recall, we first assume that all outlier points have been removed (see Algorithm 4 for example), and the measured points are placed in the columns of a matrix $X \in \mathbb{R}^{3\times k}$. Let $Y \in \mathbb{R}^{3\times k}$ be a matrix whose columns are the previously measured points. To determine point correspondences, we can solve the following optimization problem:

$$P_y = \underset{P_y}{\operatorname{argmin}}\{\|X - YP_y\|_F^2\} \tag{13}$$

$$\text{s.t. } P_y \text{ is a permutation matrix} \tag{14}$$

Where $\|\|_F$ is the Frobenius norm. A permutation matrix is one which rearranges the columns of matrix Y; i.e. each row and column has exactly one 1 and is otherwise 0. We expand this definition to allow columns that are entirely 0, to allow for missing measurement points in X. Note that in fact we should be using $TX - YP_y$, where T represents the rotation and translation of the points since the last measurement, but we assume here that this transform is close to identity because the object has moved very little since the last frame and ignore it. Using the Frobenius inner product, we can expand Equation 13:

$$P_y = \underset{P_y}{\operatorname{argmin}}\{\|X\|_F^2 + \|YP_y\|_F^2 - 2\langle X, YP_y\rangle_F\}$$

Assuming Y and X have the same number of columns (we will revisit this below), P_y has no columns that are all zero, so all it does is rearrange the columns of Y. Hence, $\|YP_y\|_F^2 = \|Y\|_F^2$. This and $\|X\|_F^2$ are constants, so they do not affect the minimization. Thus we are left with:

$$P_y = \underset{P_y}{\operatorname{argmax}}\{\langle X, YP_y\rangle_F\}$$

By the definition of the Frobenius norm, this simplifies further to

$$\begin{aligned} P_y &= \underset{P_y}{\operatorname{argmax}}\{\operatorname{tr}(X^\top YP_y)\} \\ &= \underset{i_1,i_2,\ldots,i_k}{\operatorname{argmax}} \left\{\sum_{j=1}^{k}(X^\top Y)_{ji_j}\right\} \quad \text{s.t. } i_j \neq i_k \forall j,k \end{aligned} \tag{15}$$

Which is to say, we choose one element from each row of $X^\top Y$ such that no two choices come from the same column, and their sum is maximized. This can be achieved in $O(k!)$ time, which, considering $k = 3$ or 4, is faster than k^2 or k^3 respectively. It also makes sense as we are effectively just maximizing the cross-covariance of X and YP_y. A simple solution is shown in Algorithm 1. This method works if there are no outliers, but we also assumed that X and Y have the same dimension, k. This is not the case if X is missing a marker. To handle this scenario, we simply perform the presented optimization repeatedly, each time removing a different column of Y, and keep the best result. Thus we arrive at Algorithm 1. If the measurement rate is too slow relative to the speed of motion, T in $TX - YP_y$ is no longer close to identity, so the method fails.

C. State Equation

The state equation found in Eqn. 7 is simple to derive. The position of one of the markers is

$$\boldsymbol{r}_k^i = \boldsymbol{c}_k + R_k\boldsymbol{\ell}_k^i$$

Where $\boldsymbol{c}$ is the origin of the tracked object's coordinate system expressed in the world frame, and $\boldsymbol{\ell}^i$ is the position of the marker in the object frame. The state update is then

$$\boldsymbol{r}_{k+1}^i = \boldsymbol{r}_k^i + dt\dot{\boldsymbol{r}}_k^i + \frac{1}{2}dt^2\ddot{\boldsymbol{r}}_k^i$$

From these two equations we can determine the update law by taking the derivative twice.

$$\begin{aligned} \boldsymbol{r}_{k+1}^i = \boldsymbol{r}_k^i + dt\left(\dot{\boldsymbol{c}}_k + [\boldsymbol{\omega}_k]_\times R_k\boldsymbol{\ell}^i\right) + \\ \left\{\frac{1}{2}dt^2\left[\ddot{\boldsymbol{c}}_k + ([\boldsymbol{\omega}_k]_\times^2 + [\boldsymbol{\alpha}_k]_\times)R_k\boldsymbol{\ell}^i\right]\right\} \text{ for } i \in \{1,\ldots,n\} \end{aligned}$$

D. Square-Root Unscented Kalman Filter

The state, output, and noise vectors, and the state update and output equations are given in Section II-E. In the following, the implementation of the SR-UKF using this model is described. Additionally, the covariance matrices are derived from the noise vectors.

The UKF is similar to a particle filter in that it uses sampling to make an a priori estimate. However, unlike random Monte Carlo sampling, UKFs use the unscented transform with deterministic sampling at a few well-chosen points called sigma points. By choosing these correctly, they maintain the correct mean, variance, and moments of the distribution after being transformed through the nonlinear model. In addition to updating the state every time step, the UKF updates the process covariance matrix, P_{proc}. To create the sigma points, however, the matrix square root of P_{proc} is used. It is computationally relatively expensive to compute the matrix square root every iteration, and it fails if P_{proc} is not positive definite. Unfortunately, this happens easily due to numerical error in the computations. Therefore, a modified UKF algorithm called the square root UKF (SR-UKF) is used in which the matrix square root, S_k, of the covariance matrix is propagated rather than the matrix itself [32]. The SR-UKF algorithm is shown in Algorithm 5.

This makes use of a few powerful linear algebra techniques which are described in detail in reference [32]. Briefly, the Cholesky factorization is used for the matrix square root. This expresses a symmetric positive definite (SPD) matrix A as $A = LL^\top$, where L is upper triangular. To efficiently compute this, we use the QR decomposition, which expresses matrix $A = QR$ where Q is orthogonal and R is upper triangular. Taking just the triangular part of R gives us the Cholesky factor, L. Since this works only for SPD matrices and sometimes $\boldsymbol{w}_c[1] < 0$, we initially include only columns 2 to $2N + 1$ (see Algorithm 5), and then perform a rank one Cholesky update. This is a useful tool for efficiently computing the Cholesky factorization of a matrix $A + \gamma\boldsymbol{x}\boldsymbol{x}^\top$ when the Cholesky factor of A is already known. In Algorithm 5, this is expressed as $cholupdate(A, \boldsymbol{x}, \gamma)$. If γ is negative, this is known as a Cholesky downdate. Both the up and downdate are

Algorithm 5 Square Root Unscented Kalman Filter

Set $1e-4 < \alpha < 1$, $\beta = 2$, $\kappa = 3 - N$, and $\lambda = \alpha^2(N + \kappa) - N$
Initialize $\boldsymbol{x}_0$ using a good measurement
Initialize $S_0 = choleksy(P_{proc})$
$\boldsymbol{w}_c[1] \leftarrow \dfrac{\lambda}{N+\lambda} + 1 - \alpha^2 + \beta$, and $\boldsymbol{w}_m[1] \leftarrow \dfrac{\lambda}{N+\lambda}$,
$\boldsymbol{w}_c[2:2N+1] = \boldsymbol{w}_m[2:2N+1] \leftarrow \dfrac{1}{2(N+\lambda)}$

Calculate $2N+1$ sigma points:
$X_{k-1} \leftarrow \left[\boldsymbol{x}_{k-1} \quad \boldsymbol{x}_{k-1} + \sqrt{N+\lambda}S_k \quad \boldsymbol{x}_{k-1} - \sqrt{N+\lambda}S_k\right]$

State Update:
$X_{k|k-1} \leftarrow \boldsymbol{f}(X_{k-1})$
$\boldsymbol{x}_k^- \leftarrow X_{k|k-1}\boldsymbol{w}_m$
$S_k^- \leftarrow qr\left\{\left[\sqrt{\boldsymbol{w}_c[2]}X_{k|k-1}[:, 2:2N+1] \ \sqrt{P_{proc}}\right]\right\}$
$S_k^- \leftarrow cholupdate(S_k^-, X_{k|k-1}[:,1] - \boldsymbol{x}_k^-, \boldsymbol{w}_c[1])$
$Y_{k|k-1} \leftarrow \boldsymbol{h}(X_{k|k-1})$
$\boldsymbol{y}_k^- \leftarrow Y_{k|k-1}\boldsymbol{w}_m$

Measurement Update:
$S_y^- \leftarrow qr\left\{\left[\sqrt{\boldsymbol{w}_c[2]}Y_{k|k-1}[:, 2:2N+1] \ \sqrt{P_{mes}}\right]\right\}$
$S_y \leftarrow cholupdate(S_y, Y_{k|k-1}[:,1] - \boldsymbol{y}_k^-, \boldsymbol{w}_c[1])$
Find cross-covariance
$P_{xy} \leftarrow \sum_{i=1}^{2N+1} \boldsymbol{w}_c[i](X_{k|k-1}[:,i] - \boldsymbol{x}_k^-)(Y_{k|k-1}[:,i] - \boldsymbol{y}_k^-)^\top$
Compute Kalman gain
$K_k \leftarrow (P_{xy}/S_y^\top)/S_y$
Update state and covariance
$\boldsymbol{x}_k \leftarrow \boldsymbol{x}_k^- + K_k(\boldsymbol{y}_k^{mes} - \boldsymbol{y}_k^-)$
$S_k \leftarrow cholupdate(S_k^-, K_k S_y, -1)$

built-in functions in MATLAB, and algorithms for computing them are described in [62]–[65]. In C# we implemented these functions using Alglib.

Finally, to compute the Kalman gain, since S_y is an upper triangular Cholesky factor, we can use efficient backsubstitution twice rather than matrix inversion. Specifically, $P_{xy} = (K_k S_y)S_y^\top$, so we can solve for $(K_k S_y)$ using backsubstition once on $S_y^\top$, then solve for K_k by applying backsubstitution on S_y to the result.

The measurement update step is very flexible in that it is possible to update just the parts of the state that were measured in a given sample. In particular, the IMU and image-based marker data arrive at very different rates, so there can be frequent updates of IMU data and less frequent marker position updates. When marker updates do arrive, these may not include all the markers. To support this flexibility, we temporarily set the uncertainty (σ) associated with the missing data to a very large number before computing S_y and K_k. In this way, those values are effectively ignored.

Parameters α, β, and κ are chosen based on the application. A value of $\beta = 2$ is optimal for Gaussian distributions, and κ is usually set to $3 - N$, where N is the dimension of the state.

The other important parameters to tune the performance of the Kalman filter include the variances of the process and measurement noise, from Equations 8, 9, and 12. The resulting covariance matrices are used in the state and measurement update steps of Algorithm 5. The measurement covariance is given in Equation 16.

$$P_{mes} = \begin{bmatrix} \sigma_r^2 I_n & 0 \\ 0 & \sigma_\omega^2 I_3 \end{bmatrix} \tag{16}$$

On the other hand, the process covariance is given by Equation 17.

$$\begin{aligned} P_{proc} &= \mathbb{E}\left\{\boldsymbol{v}_k \boldsymbol{v}_k^\top\right\} \\ &= \begin{bmatrix} E_{rr} & E_{r\alpha} & E_{rc} \\ E_{r\alpha}^\top & dt^2\sigma_\alpha^2 I_3 & E_{c\alpha} \\ E_{rc}^\top & E_{c\alpha}^\top & dt^2\sigma_c^2 I_3 \end{bmatrix} \in \mathbb{R}^{(3n+6)\times(3n+6)} \end{aligned} \tag{17}$$

Where, for $i, j \in [1, n]$,

$$E_{rr}^{ji} = \frac{1}{4}dt^4 \mathbb{E}\left\{\ddot{\boldsymbol{r}}_k^j (\ddot{\boldsymbol{r}}_k^i)^\top\right\} \tag{18}$$

$$E_{r\alpha}^{i} = \frac{1}{2}dt^3 \mathbb{E}\left\{\boldsymbol{\alpha}_k (\ddot{\boldsymbol{r}}_k^i)^\top\right\} \tag{19}$$

$$E_{rc}^{i} = \frac{1}{2}dt^3 \mathbb{E}\left\{\ddot{\boldsymbol{c}}_k (\ddot{\boldsymbol{r}}_k^i)^\top\right\} \tag{20}$$

where $\ddot{\boldsymbol{r}}_k^i = \left[\ddot{\boldsymbol{c}}_k + ([\boldsymbol{\omega}_k]_\times^2 + [\boldsymbol{\alpha}_k]_\times)R_k \boldsymbol{\ell}^i\right]$

$$E_{c\alpha} = dt^2 \mathbb{E}\left\{\ddot{\boldsymbol{c}}_k \boldsymbol{\alpha}_k^\top\right\} \tag{21}$$

Since both random variables, $\ddot{\boldsymbol{c}}_k$ and $\boldsymbol{\alpha}_k$, are iid and zero mean, we can eliminate any degree 1 or cross terms.

$$E_{r\alpha}^{i} = \frac{1}{2}dt^3 \mathbb{E}\left\{\boldsymbol{\alpha}_k \left(\boldsymbol{\ell}^i\right)^\top R_k^\top [\boldsymbol{\alpha}_k]_\times^\top\right\} \tag{22}$$

$$E_{rc}^{i} = \frac{1}{2}dt^3 \mathbb{E}\left\{\ddot{\boldsymbol{c}}_k \ddot{\boldsymbol{c}}_k^\top\right\} = \frac{1}{2}dt^3\sigma_c^2 I_3 \tag{23}$$

$$E_{rc} = \begin{bmatrix} E_{rc}^1 \\ \vdots \\ E_{rc}^n \end{bmatrix} \in \mathbb{R}^{3n\times 3} \tag{24}$$

$$E_{c\alpha} = 0_3 \tag{25}$$

If we let the rows of R_k be vectors $\boldsymbol{R}_1^\top$, $\boldsymbol{R}_2^\top$, and $\boldsymbol{R}_3^\top$, then Equation 22 can be written as

$$E_{r\alpha}^{i} = \frac{1}{2}dt^3 \mathbb{E}\left\{\boldsymbol{\alpha}_k \begin{bmatrix} \alpha_k^y \boldsymbol{R}_3^\top \boldsymbol{\ell}^i - \alpha_k^z \boldsymbol{R}_2^\top \boldsymbol{\ell}^i \\ \alpha_k^z \boldsymbol{R}_1^\top \boldsymbol{\ell}^i - \alpha_k^x \boldsymbol{R}_3^\top \boldsymbol{\ell}^i \\ \alpha_k^x \boldsymbol{R}_2^\top \boldsymbol{\ell}^i - \alpha_k^y \boldsymbol{R}_1^\top \boldsymbol{\ell}^i \end{bmatrix}^\top\right\}$$

Since $\alpha_x,\ \alpha_y,\ \alpha_z$ are iid:

$$= \frac{1}{2}dt^3\sigma_\alpha^2 \begin{bmatrix} 0 & -\boldsymbol{R}_3^\top \boldsymbol{\ell}^i & \boldsymbol{R}_2^\top \boldsymbol{\ell}^i \\ \boldsymbol{R}_3^\top \boldsymbol{\ell}^i & 0 & -\boldsymbol{R}_1^\top \boldsymbol{\ell}^i \\ -\boldsymbol{R}_2^\top \boldsymbol{\ell}^i & \boldsymbol{R}_1^\top \boldsymbol{\ell}^i & 0 \end{bmatrix} \tag{26}$$

$$E_{r\alpha} = \begin{bmatrix} E_{r\alpha}^1 \\ \dots \\ E_{r\alpha}^n \end{bmatrix} \in \mathbb{R}^{3n\times 3} \tag{27}$$

This is easily calculated during each time step. Finally, E_{rr} remains to be simplified. We can first expand Equation 18 by temporarily using the notation $\ddot{R} = ([\boldsymbol{\omega}_k]_\times^2 + [\boldsymbol{\alpha}_k]_\times)R_k$ for convenience.

$$E_{rr}^{ji} = \frac{1}{4}dt^4\mathbb{E}\left\{\ddot{\boldsymbol{c}}_k\ddot{\boldsymbol{c}}_k^\top + \ddot{\boldsymbol{c}}_k(\boldsymbol{\ell}^j)^\top\ddot{R}^\top + \ddot{R}\boldsymbol{\ell}^i\ddot{\boldsymbol{c}}_k^\top + \ddot{R}\boldsymbol{\ell}^i(\boldsymbol{\ell}^j)^\top\ddot{R}^\top\right\}$$

The middle two terms disappear in the expectation since $\ddot{\boldsymbol{c}}_k$ and $\boldsymbol{\alpha}_k$ (in $\ddot{R}$) are iid and zero mean. The first term also simplifies trivially to $\sigma_c^2 I_3$. The last term expands to

$$\mathbb{E}\left\{\ddot{R}\boldsymbol{\ell}^i(\boldsymbol{\ell}^j)^\top\ddot{R}^\top\right\} = [\boldsymbol{\omega}]_\times^2 R_k\boldsymbol{\ell}^i(\boldsymbol{\ell}^j)^\top R_k^\top[\boldsymbol{\omega}]_\times^2 + \mathbb{E}\left\{[\boldsymbol{\alpha}]_\times R_k\boldsymbol{\ell}^i(\boldsymbol{\ell}^j)^\top R_k^\top[\boldsymbol{\alpha}]_\times^\top\right\}$$

Two additional terms are first order in $\boldsymbol{\alpha}_k$ so their expectations equal 0. After extensive ugly simplification, and again using the rows of R_k, the second term becomes:

$$\mathbb{E}\left\{[\boldsymbol{\alpha}]_\times R_k\boldsymbol{\ell}^i(\boldsymbol{\ell}^j)^\top R_k^\top[\boldsymbol{\alpha}]_\times^\top\right\} = G^{ij}\sigma_\alpha^2 \tag{28}$$

$$\text{Where } G^{ij} = \begin{bmatrix} g_3^{ij} + g_2^{ij} & 0 & 0 \\ 0 & g_3^{ij} - g_1^{ij} & 0 \\ 0 & 0 & g_1^{ij} + g_2^{ij} \end{bmatrix}$$

$$\text{And } g_l^{ij} = (R_l^\top\boldsymbol{\ell}_i)^\top(R_l^\top\boldsymbol{\ell}_j)$$

The process covariance of the marker locations is then

$$E_{rr} = \begin{bmatrix} E_{rr}^{11} & \dots & E_{rr}^{1n} \\ \vdots & \ddots & \vdots \\ E_{rr}^{n1} & \dots & E_{nn} \end{bmatrix} \in \mathbb{R}^{3n\times 3n} \tag{29}$$

Where $E_{rr}^{ji} = \frac{1}{4}dt^4\left(\sigma_c^2 I_3 + [\boldsymbol{\omega}]_\times^2 R_k\boldsymbol{\ell}^i(\boldsymbol{\ell}^j)^\top R_k^\top[\boldsymbol{\omega}]_\times^2 + G^{ij}\sigma_\alpha^2\right)$

Thus, finally, we obtain the process covariance matrix:

$$P_{proc} = \begin{bmatrix} E_{rr} & E_{r\alpha} & \frac{1}{2}dt^3\sigma_c^2 I_3 \\ E_{r\alpha}^\top & dt^2\sigma_\alpha^2 I_3 & 0_3 \\ \frac{1}{2}dt^3\sigma_c^2 I_3 & 0_3 & dt^2\sigma_c^2 I_3 \end{bmatrix} \tag{30}$$

Where E_{rr} is given in Equation 29 and $E_{r\alpha}$ is given in Equation 27. Knowing this covariance matrix is important for accurate state updates.

REFERENCES

[1] A. C. C. Reyes, N. P. A. Del Gallego, and J. A. P. Deja, "Mixed reality guidance system for motherboard assembly using tangible augmented reality," in *Proceedings of the 2020 4th International Conference on Virtual and Augmented Reality Simulations*, pp. 1–6, 2020.

[2] N. Gavish, T. Gutiérrez, S. Webel, J. Rodríguez, M. Peveri, U. Bockholt, and F. Tecchia, "Evaluating virtual reality and augmented reality training for industrial maintenance and assembly tasks," *Interactive Learning Environments*, vol. 23, no. 6, pp. 778–798, 2015.

[3] S. Henderson and S. Feiner, "Exploring the benefits of augmented reality documentation for maintenance and repair," *IEEE transactions on visualization and computer graphics*, vol. 17, no. 10, pp. 1355–1368, 2010.

[4] R. Tang, X.-D. Yang, S. Bateman, J. Jorge, and A. Tang, "Physio@ home: Exploring visual guidance and feedback techniques for physiotherapy exercises," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 4123–4132, 2015.

[5] Y. A. Sekhavat and M. S. Namani, "Projection-based ar: Effective visual feedback in gait rehabilitation," *IEEE Transactions on Human-Machine Systems*, vol. 48, no. 6, pp. 626–636, 2018.

[6] C. Leuze, G. Yang, B. Hargreaves, B. Daniel, and J. A. McNab, "Mixed-reality guidance for brain stimulation treatment of depression," in *2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pp. 377–380, IEEE, 2018.

[7] M. Rosenthal, A. State, J. Lee, G. Hirota, J. Ackerman, K. Keller, E. D. Pisano, M. Jiroutek, K. Muller, and H. Fuchs, "Augmented reality guidance for needle biopsies: an initial randomized, controlled trial in phantoms," *Medical Image Analysis*, vol. 6, no. 3, pp. 313–320, 2002.

[8] L. Groves, N. Li, T. M. Peters, and E. C. Chen, "Towards a first-person perspective mixed reality guidance system for needle interventions," *Journal of Imaging*, vol. 8, no. 1, p. 7, 2022.

[9] J. Cartucho, D. Shapira, H. Ashrafian, and S. Giannarou, "Multimodal mixed reality visualisation for intraoperative surgical guidance," *International journal of computer assisted radiology and surgery*, vol. 15, pp. 819–826, 2020.

[10] T. Nozawa, E. Wu, F. Perteneder, and H. Koike, "Visualizing expert motion for guidance in a vr ski simulator," in *ACM SIGGRAPH 2019 Posters*, pp. 1–2, 2019.

[11] T. N. Hoang, M. Reinoso, F. Vetere, and E. Tanin, "Onebody: remote posture guidance system using first person view in virtual environment," in *Proceedings of the 9th Nordic Conference on Human-Computer Interaction*, pp. 1–10, 2016.

[12] D. Black, Y. Oloumi Yazdi, A. H. Hadi Hosseinabadi, and S. Salcudean, "Human teleoperation-a haptically enabled mixed reality system for teleultrasound," *Human–Computer Interaction*, pp. 1–24, 2023.

[13] D. Black and S. Salcudean, "Mixed reality human teleoperation," in *2023 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pp. 375–383, IEEE, 2023.

[14] H. Iqbal, F. Tatti, and F. R. y Baena, "Augmented reality in robotic assisted orthopaedic surgery: A pilot study," *Journal of Biomedical Informatics*, vol. 120, p. 103841, 2021.

[15] J. T. Verhey, J. M. Haglin, E. M. Verhey, and D. E. Hartigan, "Virtual, augmented, and mixed reality applications in orthopedic surgery," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 16, no. 2, p. e2067, 2020.

[16] D. Black, A. H. Hadi Hosseinabadi, N. Rangga Pradniyawira, P. Maxime, M. Nogami, and S. Salcudean, "Towards differential magnetic force sensing for ultrasound teleoperation," in *IEEE World Haptics Conference*, IEEE, 2023.

[17] M. H. Mozaffari and W.-S. Lee, "Freehand 3-d ultrasound imaging: a systematic review," *Ultrasound in medicine & biology*, vol. 43, no. 10, pp. 2099–2124, 2017.

[18] B. Hannaford and R. Anderson, "Experimental and simulation studies of hard contact in force reflecting teleoperation," in *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, pp. 584–589, IEEE, 1988.

[19] R. Daniel and P. R. McAree, "Fundamental limits of performance for force reflecting teleoperation," *The international journal of robotics research*, vol. 17, no. 8, pp. 811–830, 1998.

[20] C. Reboulet, Y. Plihon, and Y. Briere, "Interest of the dual hybrid control scheme for teleoperation with time delays for proceeding of iser'95," in *Experimental Robotics IV*, pp. 498–506, Springer, 1997.

[21] C. R. Flatau, "Sm-229: a new compact servo master-slave manipulator," in *Proceedings of the 25th conference on remote systems technology*, 1977.

[22] K. J. Kuchenbecker and G. Niemeyer, "Induced master motion in force-reflecting teleoperation," 2006.

[23] K. Hashtrudi-Zaad and S. Salcudean, "Analysis and evaluation of stability and performance robustness for teleoperation control architectures," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 4, pp. 3107–3113 vol.4, 2000.

[24] K. Hashtrudi-Zaad and S. E. Salcudean, "Transparency in time-delayed systems and the effect of local force feedback for transparent teleoperation," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 1, pp. 108–114, 2002.

[25] S. E. Salcudean, K. Hashtrudi-Zaad, S. Tafazoli, S. P. DiMaio, and C. Reboulet, "Bilateral matched impedance teleoperation with application to excavator control," *IEEE Control Systems Magazine*, vol. 19, no. 6, pp. 29–37, 1999.

[26] A. Aziminejad, M. Tavakoli, R. V. Patel, and M. Moallem, "Transparent time-delayed bilateral teleoperation using wave variables," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 3, pp. 548–555, 2008.

[27] K. Dorfmüller-Ulhaas, "Robust optical user motion tracking using a kalman filter," 2007.

[28] Y. Qi, H. Sadjadi, C. T. Yeo, K. Hashtrudi-Zaad, and G. Fichtinger, "Electromagnetic tracking performance analysis and optimization," in *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 6534–6538, IEEE, 2014.

[29] A. Lang, P. Mousavi, G. Fichtinger, and P. Abolmaesumi, "Fusion of electromagnetic tracking with speckle-tracked 3d freehand ultrasound using an unscented kalman filter," in *Medical Imaging 2009: Ultrasonic Imaging and Signal Processing*, vol. 7265, pp. 399–410, SPIE, 2009.

[30] A. Vaccarella, E. De Momi, A. Enquobahrie, and G. Ferrigno, "Unscented kalman filter based sensor fusion for robust optical and electromagnetic tracking in surgical navigation," *IEEE Transactions on Instrumentation and Measurement*, vol. 62, no. 7, pp. 2067–2081, 2013.

[31] C. He, P. Kazanzides, H. T. Sen, S. Kim, and Y. Liu, "An inertial and optical sensor fusion approach for six degree-of-freedom pose estimation," *Sensors*, vol. 15, no. 7, pp. 16448–16465, 2015.

[32] R. Van Der Merwe and E. A. Wan, "The square-root unscented kalman filter for state and parameter-estimation," in *2001 IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No. 01CH37221)*, vol. 6, pp. 3461–3464, IEEE, 2001.

[33] S. Julier and J. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[34] C. Vijayakumar and R. Rajagopal, "Passive target tracking by unscented filters," in *Proceedings of IEEE International Conference on Industrial Technology 2000 (IEEE Cat. No.00TH8482)*, vol. 2, pp. 129–134 vol.1, 2000.

[35] Y. Xu and L. Liping, "Single observer bearings-only tracking with the unscented kalman filter," in *2004 International Conference on Communications, Circuits and Systems (IEEE Cat. No.04EX914)*, vol. 2, pp. 901–905 Vol.2, 2004.

[36] R. Zhan and J. Wan, "Iterated unscented kalman filter for passive target tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 43, no. 3, pp. 1155–1163, 2007.

[37] N. Enayati, E. De Momi, and G. Ferrigno, "A quaternion-based unscented kalman filter for robust optical/inertial motion tracking in computer-assisted surgery," *IEEE Transactions on Instrumentation and Measurement*, vol. 64, no. 8, pp. 2291–2301, 2015.

[38] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[39] P. Kriechling, R. Loucas, M. Loucas, F. Casari, P. Fürnstahl, and K. Wieser, "Augmented reality through head-mounted display for navigation of baseplate component placement in reverse total shoulder arthroplasty: a cadaveric study," *Archives of Orthopaedic and Trauma Surgery*, vol. 143, no. 1, pp. 169–175, 2023.

[40] F. Müller, S. Roner, F. Liebmann, J. M. Spirig, P. Fürnstahl, and M. Farshad, "Augmented reality navigation for spinal pedicle screw instrumentation using intraoperative 3d imaging," *The Spine Journal*, vol. 20, no. 4, pp. 621–628, 2020.

[41] C. Gsaxner, J. Li, A. Pepe, D. Schmalstieg, and J. Egger, "Inside-out instrument tracking for surgical navigation in augmented reality," in *Proceedings of the 27th ACM Symposium on Virtual Reality Software and Technology*, pp. 1–11, 2021.

[42] C. Kunz, P. Maurer, F. Kees, P. Henrich, C. Marzi, M. Hlaváč, M. Schneider, and F. Mathis-Ullrich, "Infrared marker tracking with the hololens for neurosurgical interventions," *Current Directions in Biomedical Engineering*, vol. 6, no. 1, 2020.

[43] H. Iqbal and F. R. y Baena, "Semi-automatic infrared calibration for augmented reality systems in surgery," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4957–4964, IEEE, 2022.

[44] A. Martin-Gomez, H. Li, T. Song, S. Yang, G. Wang, H. Ding, N. Navab, Z. Zhao, and M. Armand, "Sttar: surgical tool tracking using off-the-shelf augmented reality head-mounted displays," *IEEE Transactions on Visualization and Computer Graphics*, 2023.

[45] L. Di Stefano, M. Marchionni, and S. Mattoccia, "A fast area-based stereo matching algorithm," *Image and vision computing*, vol. 22, no. 12, pp. 983–1005, 2004.

[46] A. Geiger, M. Roser, and R. Urtasun, "Efficient large-scale stereo matching," in *Asian conference on computer vision*, pp. 25–38, Springer, 2010.

[47] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2, pp. 1150–1157, Ieee, 1999.

[48] H. Chui and A. Rangarajan, "A new point matching algorithm for non-rigid registration," *Computer Vision and Image Understanding*, vol. 89, no. 2-3, pp. 114–141, 2003.

[49] G. Shi, X. Xu, and Y. Dai, "Sift feature point matching based on improved ransac algorithm," in *2013 5th International Conference on Intelligent Human-Machine Systems and Cybernetics*, vol. 1, pp. 474–477, IEEE, 2013.

[50] C. Yuan, X. Yu, and Z. Luo, "3d point cloud matching based on principal component analysis and iterative closest point algorithm," in *2016 International Conference on Audio, Language and Image Processing (ICALIP)*, pp. 404–408, IEEE, 2016.

[51] S. Oomori, T. Nishida, and S. Kurogi, "Point cloud matching using singular value decomposition," *Artificial Life and Robotics*, vol. 21, pp. 149–154, 2016.

[52] Z. Gojcic, C. Zhou, J. D. Wegner, and A. Wieser, "The perfect match: 3d point cloud matching with smoothed densities," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5545–5554, 2019.

[53] B. Schwald, "A tracking algorithm for rigid point-based marker models," 2005.

[54] D. Ungureanu, F. Bogo, S. Galliani, P. Sama, X. Duan, C. Meekhof, J. Stühmer, T. J. Cashman, B. Tekin, J. L. Schönberger, *et al.*, "Hololens 2 research mode as a tool for computer vision research," *arXiv preprint arXiv:2008.11239*, 2020.

[55] J. Lawrence, J. Bernal, and C. Witzgall, "A purely algebraic justification of the kabsch-umeyama algorithm," *Journal of research of the National Institute of Standards and Technology*, vol. 124, p. 1, 2019.

[56] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[57] M. I. Ribeiro, "Kalman and extended kalman filters: Concept, derivation and properties," *Institute for Systems and Robotics*, vol. 43, no. 46, pp. 3736–3741, 2004.

[58] S. Yang and M. Baum, "Extended kalman filter for extended object tracking," in *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 4386–4390, IEEE, 2017.

[59] D. Black, D. Andjelic, and S. Salcudean, "Evaluation of communication and human response latency for (human) teleoperation," *TechRxiv*, 2022.

[60] D. Black and S. Salcudean, "Human-as-a-robot performance in mixed reality teleultrasound," *International Journal of Computer Assisted Radiology and Surgery*, pp. 1–8, 2023.

[61] F. Steinicke, C. P. Jansen, K. H. Hinrichs, J. Vahrenhold, and B. Schwald, "Generating optimized marker-based rigid bodies for optical tracking systems.," in *VISAPP (2)*, pp. 387–395, Citeseer, 2007.

[62] G. W. Stewart, *Matrix algorithms: volume 1: basic decompositions*. SIAM, 1998.

[63] T. A. Davis and W. W. Hager, "Modifying a sparse cholesky factorization," *SIAM Journal on Matrix Analysis and Applications*, vol. 20, no. 3, pp. 606–627, 1999.

[64] M. Seeger, "Low rank updates for the cholesky decomposition," tech. rep., 2004.

[65] O. Krause and C. Igel, "A more efficient rank-one covariance matrix update for evolution strategies," in *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*, pp. 129–136, 2015.