Probe Pose Estimation from Ultrasound Images

A Project Report

Submitted by

Aravind Shankara Narayanan
(roll no. EE14B012)

*in partial fulfilment of the requirements
for the award of the degree of*

BACHELOR OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS

July 2018

THESIS CERTIFICATE

This is to certify that the thesis titled "Probe Pose Estimation from Ultrasound Images", submitted by Aravind Shankara Narayanan, to the Indian Institute of Technology, Madras, for the award of the degree of Bachelor of Technology, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. M Manivannan
Project Guide
Touch Lab
Biomedical Engineering Group
Department of Applied Mechanics
IIT-Madras, 600036

Prof. Kaushik Mitra
Project Co-Guide
Image Processing and Computer Vision Lab
Department of Electrical Engineering
IIT-Madras, 600036

Place: Chennai

Date: 6 July 2018

ACKNOWLEDGEMENTS

I would like to thank my project guide, Dr. M. Manivannan, for all his support, supervision and help during the project work. I would also like to thank Dr. Kaushik Mitra for being the co-guide of our project.

I extend my sincere thanks to the people at the Haptics Lab – Joseph Isaac and the rest of the lab team – and to the people at Merkel Haptics Systems – Tejaswini Yarra, Raj, Prabhu and the rest of the team – for helping me out at different stages of my project work.

Table of Contents

ABSTRACT

The project was initially intended to compute the 3D structure present in an endoscopic video, as well as the camera trajectory. Of the approaches available for this, the SLAM (Simultaneous Localization and Mapping) algorithm based on the extended Kalman filter, was selected based on the good quality of its results. Then a similar approach was proposed to be used in the domain of Ultrasound Probe Pose estimation using only the Ultrasound images as input. The final algorithm was formulated with the introduction of the concept of a virtual camera whose pose is being tracked using said algorithm, and the probe's pose could be calculated from that of the virtual camera.

To evaluate the correctness of the algorithm, we first generated datasets in two ways – phantom data using CT scan images and real ultrasound video along with live pose tracking using HTC Vive trackers. Qualitatively, the algorithms performance was analyzed by generating plots of the computed feature points along with their windows of uncertainty, and by visualizing the obtained trajectory using a custom-made 3D Ultrasound simulator using Unity Engine. Quantitatively, error measures based on the angle between normalized translation and rotation of the estimated trajectory and the ground truth were used.

While all the methods discussed were implemented, the datasets posed challenges in accurate tracking of the Ultrasound probe for various reasons. Future work could help overcome the limitations of this project while making use of the basic algorithm, insights and evaluation methods.

INTRODUCTION

Initial work and Problem Statement

This project initially dealt with endoscopic video-based 3D reconstruction. The problem statement at that time was to compute the 3D structure and camera trajectory given a video of an Endoscopic procedure. This was a challenging problem because of the nature of the images obtained – they are low in texture quality and are normally poorly lit with a considerable amount of noise present. The literature, however, revealed that many methods have been proposed to tackle this problem from various angles and to varying degrees of success. For example, [1] had used a novel method of computing features in the endoscopic image by exploiting the presence of blood vessels. However, we initially went with a method proposed by [2] that used a SLAM algorithm coupled with the extended Kalman filter and novel RANSAC and relocalization algorithms, because it was shown to generate the most promising results.

Motivation for the Ultrasound Problem Statement

Having surveyed the relevant literature, we had observed that the problem of 3D reconstruction in the domain of endoscopy had been solved to a very practical extent. However, there was a need to implement a similar solution in the domain of ultrasound where the literature showed no existing methods to compute the pose of the ultrasound probe (let alone do a 3D reconstruction). The most relevant works found were [3]–[6], all of which used additional input sources apart from the captured ultrasound video, such as optical tracking systems [4] and sensor fusion using IMU sensors [3], [6].

This problem, if solved, would find use in training Ultrasound specialists anywhere because it would need no specialized tracking equipment and would thereby be much more cost effective and easy to use. Further, the problem is an interesting one in the computer vision domain, and a solution to this problem would shed useful insights on many other similar problems of interest in the field as it deals with general image processing problems such as low texture and high noise in images as well as images that are not generated with traditional cameras (which is the case not just with ultrasound, but also with other medical imaging modalities like CT/MRI scans).

Final Problem Statement

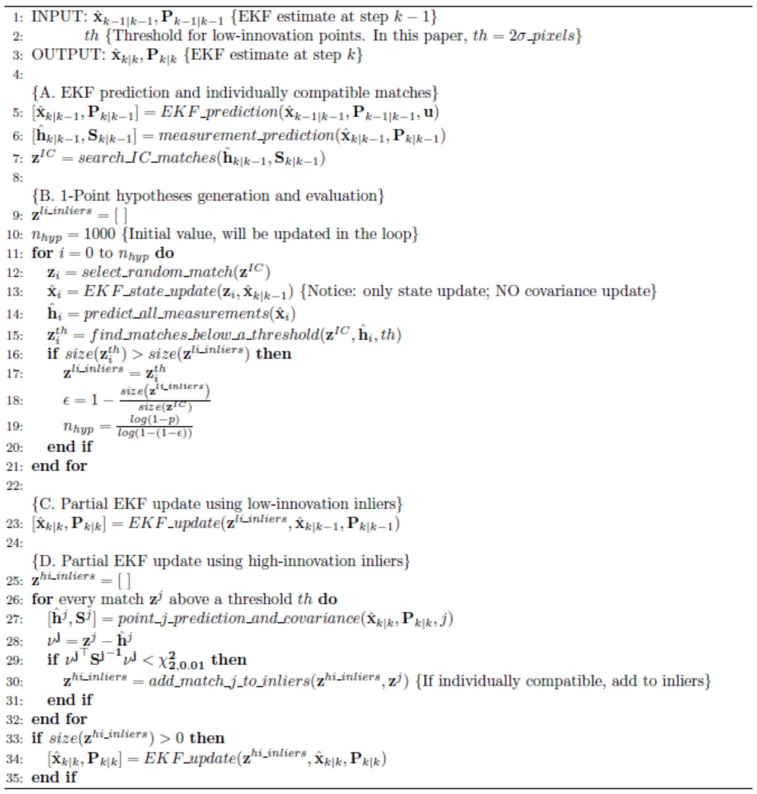
The question then became one of computing the trajectory of ultrasound probe - that is, the position and the orientation of the probe over time – given only the ultrasound images it generates as input.

In the second stage, the obtained trajectory could be used to perform a volume mapping of the ultrasound images, so that a training simulator can be implemented that takes the pose of the ultrasound probe as an input and generates the synthetic ultrasound images corresponding to the pose.

APPROACH

Backbone Algorithm: Monocular EKF-SLAM with 1-Point RANSAC

This is a part of the earlier mentioned algorithm [2] that was used to compute a full 3D reconstruction along with camera trajectory estimation from endoscopic video. This reduced algorithm is described in [7], and is shown in Figure 1.



```
INPUT: $\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{P}_{k-1|k-1}$ {EKF estimate at step $k-1$}
        $th$ {Threshold for low-innovation points. In this paper, $th = 2\sigma\_pixels$}
OUTPUT: $\hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k}$ {EKF estimate at step $k$}

{A. EKF prediction and individually compatible matches}
$[\hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1}] = EKF\_prediction(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{P}_{k-1|k-1}, \mathbf{u})$
$[\hat{\mathbf{h}}_{k|k-1}, \mathbf{S}_{k|k-1}] = measurement\_prediction(\hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1})$
$\mathbf{z}^{IC} = search\_IC\_matches(\hat{\mathbf{h}}_{k|k-1}, \mathbf{S}_{k|k-1})$

{B. 1-Point hypotheses generation and evaluation}
$\mathbf{z}^{li\_inliers} = [\,]$
$n_{hyp} = 1000$ {Initial value, will be updated in the loop}
for $i = 0$ to $n_{hyp}$ do
    $\mathbf{z}_i = select\_random\_match(\mathbf{z}^{IC})$
    $\hat{\mathbf{x}}_i = EKF\_state\_update(\mathbf{z}_i, \hat{\mathbf{x}}_{k|k-1})$ {Notice: only state update; NO covariance update}
    $\hat{\mathbf{h}}_i = predict\_all\_measurements(\hat{\mathbf{x}}_i)$
    $\mathbf{z}_i^{th} = find\_matches\_below\_a\_threshold(\mathbf{z}^{IC}, \hat{\mathbf{h}}_i, th)$
    if $size(\mathbf{z}_i^{th}) > size(\mathbf{z}^{li\_inliers})$ then
        $\mathbf{z}^{li\_inliers} = \mathbf{z}_i^{th}$
        $\epsilon = 1 - \frac{size(\mathbf{z}^{li\_inliers})}{size(\mathbf{z}^{IC})}$
        $n_{hyp} = \frac{log(1-p)}{log(1-(1-\epsilon))}$
    end if
end for

{C. Partial EKF update using low-innovation inliers}
$[\hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k}] = EKF\_update(\mathbf{z}^{li\_inliers}, \hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1})$

{D. Partial EKF update using high-innovation inliers}
$\mathbf{z}^{hi\_inliers} = [\,]$
for every match $\mathbf{z}^j$ above a threshold $th$ do
    $[\hat{\mathbf{h}}^j, \mathbf{S}^j] = point\_j\_prediction\_and\_covariance(\hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k}, j)$
    $\nu^j = \mathbf{z}^j - \hat{\mathbf{h}}^j$
    if ${\nu^j}^\top {\mathbf{S}^j}^{-1} \nu^j < \chi^2_{2,0.01}$ then
        $\mathbf{z}^{hi\_inliers} = add\_match\_j\_to\_inliers(\mathbf{z}^{hi\_inliers}, \mathbf{z}^j)$ {If individually compatible, add to inliers}
    end if
end for
if $size(\mathbf{z}^{hi\_inliers}) > 0$ then
    $[\hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k}] = EKF\_update(\mathbf{z}^{hi\_inliers}, \hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k})$
end if
```

Figure 1 - Snapshot of the 1-Point RANSAC EKF-SLAM algorithm, as outlined in the original paper.

The essential components of the algorithm, as used in our implementation, are described below.

- ***Feature Detection and Matching***

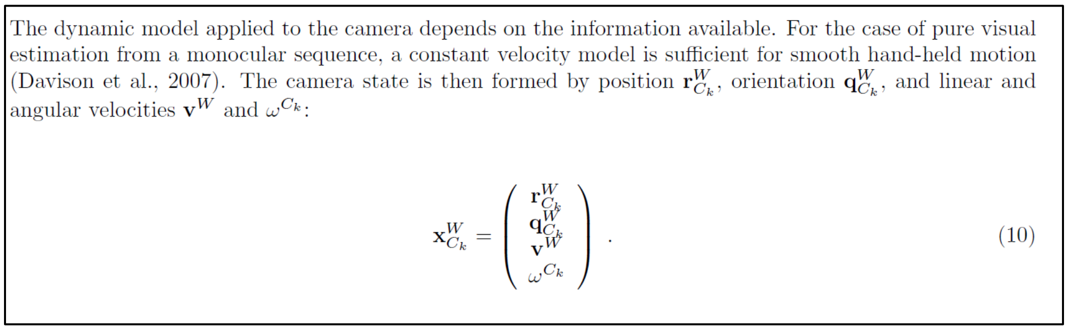
  The algorithm can choose to use any point feature detection method of one's choice – here we used a combination of 3 methods namely, the BRISK detector [8], the SIFT detector [9] and the minimum eigenvalue-based detector proposed by Shi and Tomasi [10]. These three were chosen out of a total of six detectors - the other three being the Harris corner detector [11], the SURF detector [12] and the FAST detector [13] - because they yielded the maximum number of features on a sample ultrasound image from our phantom dataset (described later). The original algorithm did use the FAST detector, but our tests with available implementations of these detectors showed that the selected three were superlative.

  The algorithm defines a random search window of fixed size in the image and selects the strongest detected feature point across all algorithms used within that window. This process of initializing one feature per window is repeated until a desired minimum number of features is initialized – this is done to ensure maximum coverage of features across the image.

  The algorithm encodes the world positions of these point features in its state vector and uses a normalized cross-correlation (NCC) method to match these features between images irrespective of the type of detector used.

- ***State Vector Definition (Figure 2)***

  The EKF algorithm, in essence, combines different sources of information to calculate the values of several variables that constitute its state vector. In this case, the state vector comprised of the 3D world position of the camera, the rotation needed to transform world space to camera space (expressed as a quaternion), the linear and angular velocity vectors of the camera, and the 3D world positions of each point feature that it can reliably detect and track across each input image. The positions of the features are initially encoded in inverse depth coordinates [14] for convenience and later converted to normal Cartesian coordinates to reduce feature vector size.



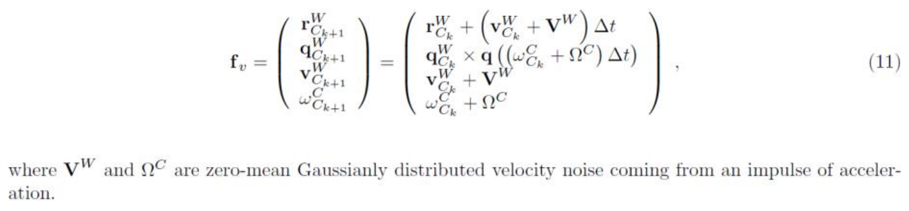
The dynamic model applied to the camera depends on the information available. For the case of pure visual estimation from a monocular sequence, a constant velocity model is sufficient for smooth hand-held motion (Davison et al., 2007). The camera state is then formed by position $\mathbf{r}^W_{C_k}$, orientation $\mathbf{q}^W_{C_k}$, and linear and angular velocities $\mathbf{v}^W$ and ω^{C_k}:

$$\mathbf{x}^W_{C_k} = \begin{pmatrix} \mathbf{r}^W_{C_k} \\ \mathbf{q}^W_{C_k} \\ \mathbf{v}^W \\ \omega^{C_k} \end{pmatrix} . \tag{10}$$

Figure 2 - State vector definition (excluding features), as in [7].

- ***Model Definition (Figure 3, Figure 4)***

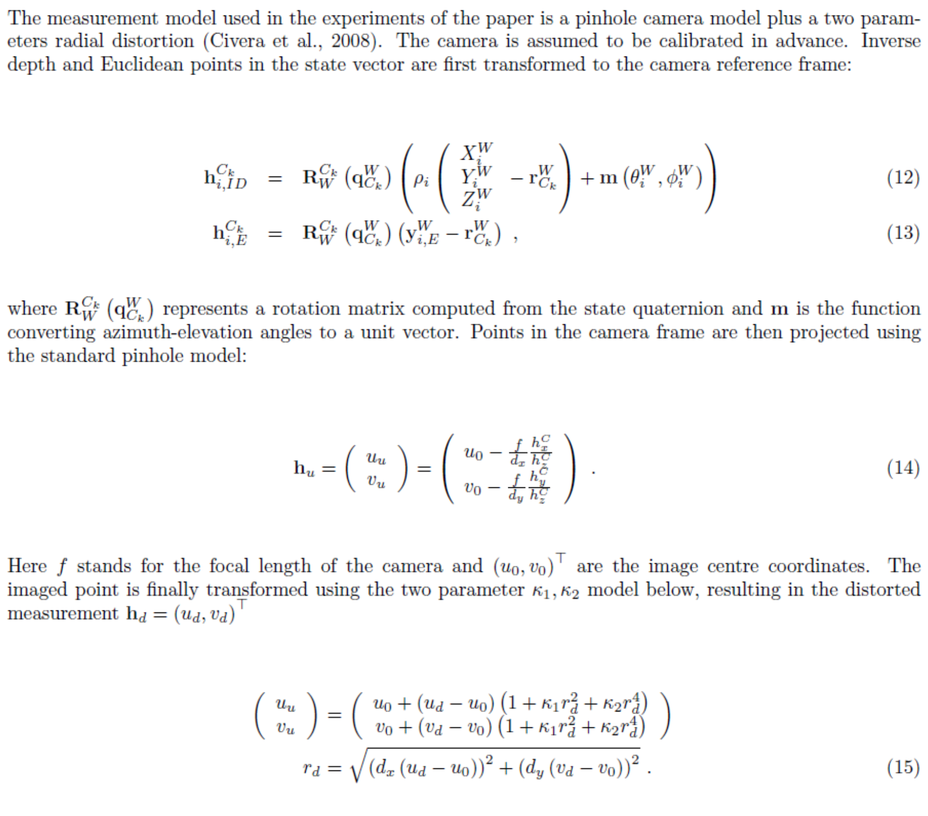
  The dynamic model used by the EKF algorithm is the same as that initially proposed in [7] which is a constant velocity model with impulses of linear and angular acceleration expressed as zero-mean Gaussian noise.



$$\mathbf{f}_v = \begin{pmatrix} \mathbf{r}^W_{C_{k+1}} \\ \mathbf{q}^W_{C_{k+1}} \\ \mathbf{v}^W_{C_{k+1}} \\ \omega^C_{C_{k+1}} \end{pmatrix} = \begin{pmatrix} \mathbf{r}^W_{C_k} + \left(\mathbf{v}^W_{C_k} + \mathbf{V}^W\right)\Delta t \\ \mathbf{q}^W_{C_k} \times \mathbf{q}\left(\left(\omega^C_{C_k} + \Omega^C\right)\Delta t\right) \\ \mathbf{v}^W_{C_k} + \mathbf{V}^W \\ \omega^C_{C_k} + \Omega^C \end{pmatrix} , \tag{11}$$

where $\mathbf{V}^W$ and Ω^C are zero-mean Gaussianly distributed velocity noise coming from an impulse of acceleration.

Figure 3 - Dynamic model for the state.



The measurement model used in the experiments of the paper is a pinhole camera model plus a two parameters radial distortion (Civera et al., 2008). The camera is assumed to be calibrated in advance. Inverse depth and Euclidean points in the state vector are first transformed to the camera reference frame:

$$\mathbf{h}^{C_k}_{i,ID} = \mathbf{R}^{C_k}_W\left(\mathbf{q}^W_{C_k}\right)\left(\rho_i\left(\begin{pmatrix} X^W_i \\ Y^W_i \\ Z^W_i \end{pmatrix} - \mathbf{r}^W_{C_k}\right) + \mathbf{m}\left(\theta^W_i, \phi^W_i\right)\right) \tag{12}$$

$$\mathbf{h}^{C_k}_{i,E} = \mathbf{R}^{C_k}_W\left(\mathbf{q}^W_{C_k}\right)\left(\mathbf{y}^W_{i,E} - \mathbf{r}^W_{C_k}\right) , \tag{13}$$

where $\mathbf{R}^{C_k}_W\left(\mathbf{q}^W_{C_k}\right)$ represents a rotation matrix computed from the state quaternion and $\mathbf{m}$ is the function converting azimuth-elevation angles to a unit vector. Points in the camera frame are then projected using the standard pinhole model:

$$\mathbf{h}_u = \begin{pmatrix} u_u \\ v_u \end{pmatrix} = \begin{pmatrix} u_0 - \frac{f}{d_x}\frac{h^C_x}{h^C_z} \\ v_0 - \frac{f}{d_y}\frac{h^C_y}{h^C_z} \end{pmatrix} . \tag{14}$$

Here f stands for the focal length of the camera and $(u_0, v_0)^\top$ are the image centre coordinates. The imaged point is finally transformed using the two parameter κ_1, κ_2 model below, resulting in the distorted measurement $\mathbf{h}_d = (u_d, v_d)^\top$

$$\begin{pmatrix} u_u \\ v_u \end{pmatrix} = \begin{pmatrix} u_0 + (u_d - u_0)\left(1 + \kappa_1 r_d^2 + \kappa_2 r_d^4\right) \\ v_0 + (v_d - v_0)\left(1 + \kappa_1 r_d^2 + \kappa_2 r_d^4\right) \end{pmatrix}$$

$$r_d = \sqrt{\left(d_x\left(u_d - u_0\right)\right)^2 + \left(d_y\left(v_d - v_0\right)\right)^2} . \tag{15}$$

Figure 4 - Measurement model.

- ***Specialization for 1-Point RANSAC***

  The general 1-Point RANSAC framework proposed in [7] had already been specialized to handle the case of visual SLAM, and hence it also hasn't been modified for this use case. Thus the "prior information" that 1-Point RANSAC algorithm requires is obtained from a state update (without a covariance update) using the state vector and covariance matrix of the previous iteration which reduces the number of model initialization points required by RANSAC to just 1, thereby considerably reducing the number of iterations required for convergence.

Adaptation to Ultrasound – The Virtual Camera

The fundamental difference between the ultrasound images and the general or endoscopic camera images is that ultrasound does not use a camera – the probe is responsible for generating the images by sending ultrasound waves and encoding the time taken for receiving the reflected ultrasound waves from the body as pixel intensities in an image. Therefore, when applying our SLAM algorithm to ultrasound, the question naturally arises as to what we mean by the estimated camera trajectory. The novelty of our proposed approach is that of introducing a virtual camera in the same coordinate frame as the probe and ultrasound images which is assumed to be responsible for capturing the images. The details of how this virtual camera is modelled and how this affects the implementation of our algorithm are explained below.

- **Image Formation using the Virtual Camera**

  The ultrasound probe and the image it generates are all contained in one 3D plane by the very nature of the ultrasound image generation. The virtual camera, however, is assumed to be similar to any normal perspective camera and therefore, it will be generating the same images as the ultrasound probe located at a point outside of this plane and the camera's orientation is such that it will always be looking into the normal of the plane.

The most popular method of ultrasound is curvilinear B-mode which implies that the images generated are not traditional rectangular images but curved images. The virtual camera, therefore, is assumed to be able to capture World points only within the curve defined by the ultrasound probe. The rest of the pixels in the input image which are normally blacked out are not considered at all in the feature detection process and are assumed to be ignored by the virtual camera.

- **Mapping the Virtual Camera to the Ultrasound Probe**

  The ultrasound probe and the virtual camera can be observed to share a "parent-child" relationship – that is, we can observe that the translation of the probe is reflected directly in the camera and the rotation of the probe causes a corresponding rotation along with translation in the virtual camera as though it has been attached to the probe by rigid supports. Therefore, it is theoretically possible to map the pose of the probe to that of the camera by the inverse relationship. These relationships are modelled by standard transform matrices for translation, rotation and scale. The scale of the world coordinate frame is taken to be the same as that of the image frame - that is, all world units are also expressed in pixels.

- **Camera Calibration**

  Since the virtual camera is modelled as a standard perspective camera, its intrinsic parameters need to be determined before it can be used in any SLAM algorithm. However, since this camera is a virtual construct, we have the liberty of fixing its parameters within some constraints and thus spared the trouble of using its own output images to calibrate it.

The intrinsic parameters have been determined as follows:

- ***Near and Far Clip Planes*** – These are defined to be infinitesimally close to each other as the image captured by the virtual camera represents points on only one world plane, which means any 3D volume spanned by the view frustum of the camera must be extremely small to not allow any world points outside of that plane to be captured by the camera. In the implementation, the distance between the planes is set to be the minimum possible but not zero as that causes problems in rendering the image.
- ***Focal Length*** – Since this represents the distance of the camera from the image plane, and the location of the image plane in world coordinates is completely left to our choice, the value of the focal length is completely arbitrary. However, we do need to ensure that the value chosen for focal length is also the same as the position of the near and far clip planes.
- ***Coordinates of the Principal Point*** – This is again left to our choice, but for our convenience, the principal point is defined to be at the center of the image and hence the appropriate pixel coordinates (which are half the respective dimensions in image coordinates)
- ***Field of View*** - Once the focal length (f) and coordinates of the principal point (C_x, C_y) are fixed, the required field of view angle (α) is determined by a formula (assuming the aspect ratio is set to be the same as that of the image):

$$\alpha = 2 * \operatorname{atan}\left(\frac{C_y}{f}\right)$$

- ***Distortion Parameters*** – While the EKF SLAM algorithm in literature allows for computation and usage of radial distortion parameters as well, in our use case we don't need to model distortion as the camera is once again our own virtual construct which we can assume to be distortion free. The ultrasound probe images as known in literature so far, do not exhibit any radial distortion artefacts which makes this an error free assumption.

- **Feature Vector Definition – The Inverse Depth constraint**

  Of the two methods used by the EKF SLAM algorithm to encode features, the Cartesian method is completely unaffected by the change of the camera to a virtual camera. However, the inverse depth method still proves advantageous when detecting and encoding a feature for the first time, and that must undergo a change in this use case. This deals with the value of the inverse depth coded in the feature vector.

  The inverse depth coordinates of a feature point correspond to six numerical values – the three Cartesian coordinates of the camera when it was initially used to capture the feature, the two angles (azimuth and elevation) made by the ray from that camera position to the position of the feature, and the depth of the feature point along that ray coded as its reciprocal. In the normal EKF SLAM algorithm, the inverse depth is assumed unknown and hence given a fixed initial value and initial standard deviation. However, in our use case, the value of the inverse depth can be determined using a direct formula from our estimates of the camera position and the coordinates of the feature point in the image:

  $\rho = 1/\sqrt{\Delta x^2 + \Delta y^2 + f^2}, where\ (\Delta x, \Delta y) = (u - u_0, v - v_0), (u, v) = feature\ point\ coordinates, (u_0, v_0) = principal\ point\ coordinates.$

  This value of the inverse depth cannot, however, be considered as the real inverse depth of the point straightaway, because the camera pose and image feature location that have been used to estimate it are themselves variable. Hence, the formula can be used only to provide a better initial guess of the inverse depth and a fixed non-zero initial standard deviation (here set to 1) is still necessary.

Adaptation to Ultrasound – Preprocessing

Since the input image is generally obtained from a real ultrasound procedure, it will contain many unnecessary artefacts that would get in the way of the SLAM algorithm, such as computer-generated extraneous markings and parameter values along with an overall surrounding of the image that does not contribute to the actual visualization. Further since we use curvilinear B-mode, the region where the virtual camera is assumed to be capturing points must be estimated first, by modelling the obtained ultrasound image as part of a circle whose center is located at the probe.

Therefore, in our implementation of this algorithm, provisions have been made to crop-out the exact region where the ultrasound image is present, cover all extraneous markings using the background color and then compute "curve parameters" which are the pixel coordinates of the center of the circle, the angle between the two extremes of the ultrasound image and the inner and outer radii of the circle, which the image spans. All these processes need to be done manually except the step of finding the curve parameters for which an automatic algorithm based on Sobel edge detection and Hough line detection has been implemented.

Apart from defining the window where the virtual camera is active, the curve parameters also determine one variable that was unknown till now – the position of the probe relative to the virtual camera. While it is obvious that the probe must be on the image plane, the exact position of the probe on the image plane is determined once the circle containing the curve of the image is known – the probe, being the source of the waves, must be at the center of that circle.

EXPERIMENT SETUP

Main algorithm - implementation details

A free online implementation of the original EKF-SLAM algorithm with 1-Point RANSAC was available, as referenced in [7]. This implementation was completely written in MATLAB but used a separate implementation of the FAST corner detection algorithm as its only feature detector.

Our implementation builds on this, so it is also coded in MATLAB, making use of the Image Processing and Computer Vision toolboxes. The Computer Vision toolbox was used to simplify the usage of the five different point feature detectors – Harris, SURF, BRISK, FAST, and minimum eigenvalue (Shi-Tomasi) – and in addition, the SIFT feature detector was thrown into the mix, thanks to an open MATLAB implementation from the VLFeat library. All six of these were tested by running them on a sample ultrasound image from our phantom dataset, from which it was found that the BRISK, SIFT, and minimum eigenvalue detectors were able to find the highest number of features.

All testing was done on a 64-bit Dell Inspiron 5547 model laptop running the Microsoft Windows 10 operating system, with a RAM capacity of 8 GB and using an Intel Core i7-4510U CPU (4-core, 2.00GHz) along with an AMD Radeon R7 M265 GPU (2GB RAM, 400MHz).

Datasets

- **Phantom dataset generated from CT scan data (example - Figure 5)**

  For a major part of the project, we used a Visual Studio C++ implementation of the Field II simulation program (http://field-ii.dk/) to use CT scan images to generate datasets that consist of ultrasound images (just frames, no video) of one single person taken using one probe, at any position and orientation of our choice. The pose information for each frame is thus easily saved as the ground truth, to compare the results of our algorithm with.
  We used this to create different subsets of the data such that each subset posed a different type of restriction on the problem. For example, a few subsets were made such that the ground truth pose of the camera only had one degree of freedom throughout (pure translation or rotation along one of the coordinate axes), while others ensured two or three degrees of freedom, and still others simulated an actual video by randomly picking images taken at poses that were close to each other.

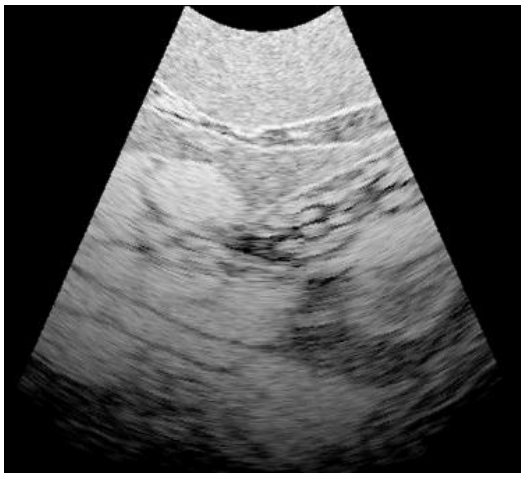
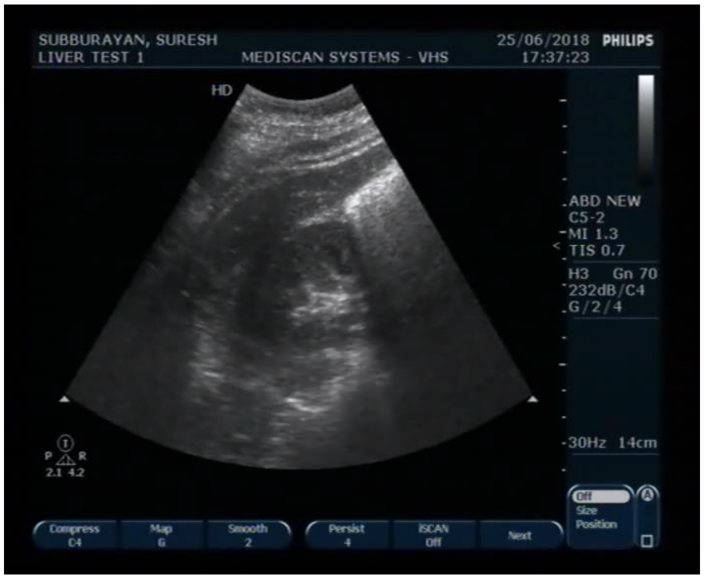


Figure 5 - Sample ultrasound image from the phantom dataset.

- **Real-world dataset, with ground truth trajectory estimated by an external tracking system**

  Towards the end of the project, we were able to use the HTC Vive tracking system to augment a real ultrasound procedure (a sample frame is shown in Figure 6). We fitted a tracker from the Vive tracking unit to the ultrasound probe using supports (as in Figure 7) and set up the rest of the Vive rig as done in any regular Vive unit setup. We used Unity Engine to make a simple virtual environment in which the tracker would move virtually as it is being tracked in the real world, and in that environment, we set up a repeated dump of the estimated pose to a text file at a fixed frame interval (5 fps). Meanwhile, the actual ultrasound video would be recorded from the probe as with any normal ultrasound procedure and would be synchronized with the tracker's dumps with the use of their corresponding timestamps.

Figure 6 - Sample frame from the real-world dataset.



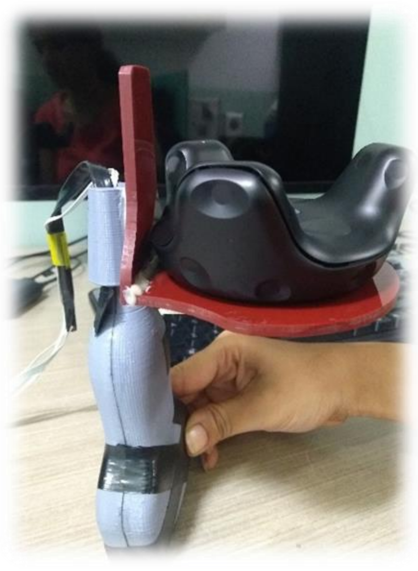


Figure 7 - The Vive tracker attached to an ultrasound probe (dummy).

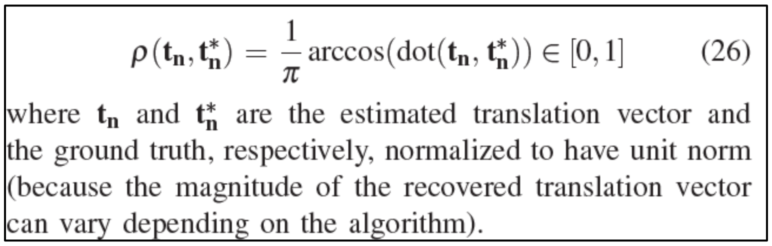
As with the phantom dataset, we ensured that we had separate data subsets available for different degrees of freedom of the probe's pose, by manually asking for the probe to be moved around in fixed ways. We had to keep a few other points in mind as well, however, which are as follows:

- The probe will, in general, have five degrees of freedom – pure translation perpendicular to the patient's body is not useful in real ultrasound.
- While moving the probe, it is important that the patient stays still – if the world points seen by the probe start moving around, the setup needs to be initialized all over again. In other words, the scene is assumed, by the algorithm, to be rigid. However, since it is common for doctors to ask the patient to move about (e.g. lie on one side of the body) in the middle of an ultrasound session, pose estimation would need to be done independently after each such movement. The motion is not continuous though – typically doctors would ask for such movement only around 2-3 times per session – so this is not much of a hindrance.
- Similarly, it is typically asked that the patient holds his/her breath during the scan to get a good probe image, which does favor our algorithm as well – but it must be ensured that the pose estimation is done entirely only when the breath is held or not held, not over a period where the patient keeps holding and releasing breaths.
- In our model, once an input frame is processed, the virtual camera is completely calibrated, and the intrinsic parameters so found stay constant throughout the session. This assumption also needs to hold, which means that typical operations done during real ultrasound sessions such as zooming in and out, adjusting other parameters like gain and curve angles and suchlike are not to be performed while carrying out pose estimation.
- Further, interfering directly with the output images by adding screen overlays (like highlighting rectangles and labels) will also be detrimental to the algorithm, and thus needs to be avoided, despite being common operations doctors perform.

Validation

- **Error in normalized vector angles (Quantitative)**

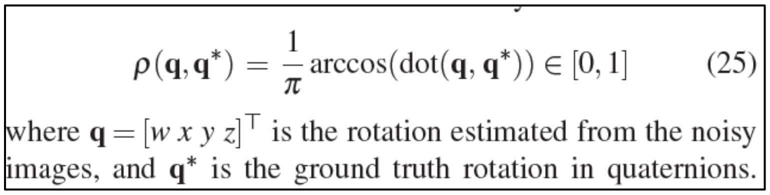
This is an error measure taken from [15], calculated as shown in Figure 8 and Figure 9:



$$\rho(\mathbf{t_n}, \mathbf{t_n^*}) = \frac{1}{\pi} \arccos(\mathrm{dot}(\mathbf{t_n}, \mathbf{t_n^*})) \in [0, 1] \qquad (26)$$

where $\mathbf{t_n}$ and $\mathbf{t_n^*}$ are the estimated translation vector and the ground truth, respectively, normalized to have unit norm (because the magnitude of the recovered translation vector can vary depending on the algorithm).

Figure 8 - Translation error definition.



$$\rho(\mathbf{q}, \mathbf{q}^*) = \frac{1}{\pi} \arccos(\mathrm{dot}(\mathbf{q}, \mathbf{q}^*)) \in [0, 1] \qquad (25)$$

where $\mathbf{q} = [w\ x\ y\ z]^\top$ is the rotation estimated from the noisy images, and $\mathbf{q}^*$ is the ground truth rotation in quaternions.

Figure 9 - Rotation error definition.

Since the absolute pose values that we obtain varies depending on our choice of coordinate frame and scale, it only makes sense to output the translation and rotation of the camera between frames, normalized to unit norm (to remove scale discrepancy), and validate the correctness of that data against ground truth translation and rotation (normalized again). So, the above error measure is a direct and robust indicator of correctness for our use case. In our implementation, provisions were made to calculate this error between every two frames (e_{fi} for i running across consecutive frame pairs), and to calculate the average cumulative error across the pairs of frames ($e_{ci} = (\frac{\sum_{j=1}^{i} e_{fi})}{i})$). These would be calculated independently for translation and rotation. All the error values computed across pairs of frames would also be plotted at the end; the trends so observed would be helpful in gaining insights about the stability of the implementation (i.e. how the EKF-SLAM correctness improves or worsens with more input).

- **Unity Engine-based ultrasound simulator (Qualitative)**

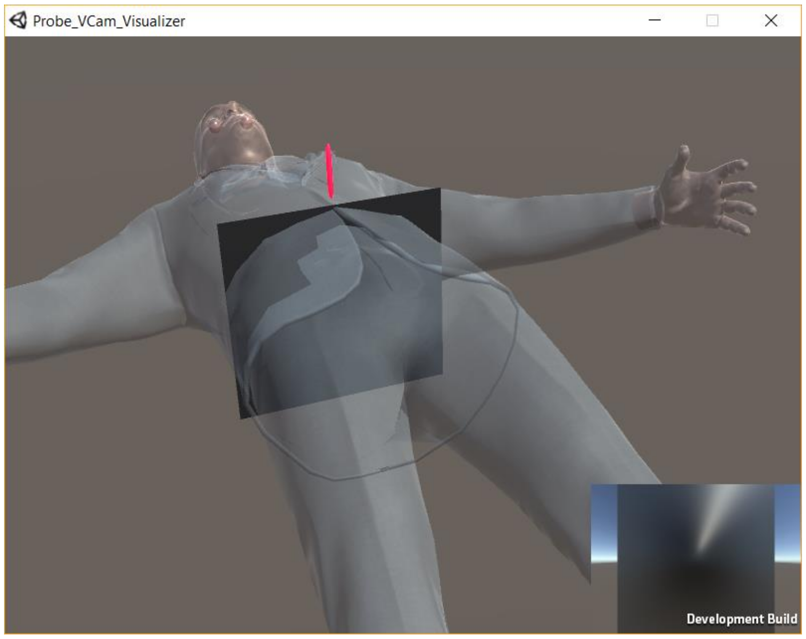


Figure 10 - Screenshot of Unity-based ultrasound visualizer.

In cases where ground truth data may not be present, or to intuitively visualize the estimated trajectory and get an “eyeball” estimate of correctness in general, a program that takes the output trajectory as its input and creates a simulated video of the probe moving around in that trajectory over a dummy human body would be very helpful. We made a simulator (shown in Figure 10) that does just that, using Unity Engine, and it communicates via TCP/IP with the MATLAB EKF-SLAM implementation to get live updates of the estimated pose as the main algorithm processes each frame. We can choose to see this on the fly, or we can see the video later as it takes photos of the simulation and lets the MATLAB code convert it into a video at the end of the algorithm.

- **Plot of feature points with estimated uncertainties (Qualitative)**

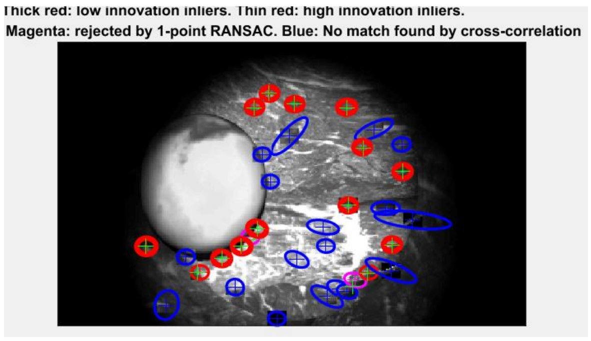

Thick red: low innovation inliers. Thin red: high innovation inliers.
Magenta: rejected by 1-point RANSAC. Blue: No match found by cross-correlation

Figure 11 - Feature point plot example (the image is an endoscopy phantom).

While the camera (and probe) pose are our intended outputs, plotting the feature points stored in the state vector along with their individual regions (ellipses) of uncertainty would yield useful insights about how the algorithm is able to function – after all, those points are its landmarks in calculating the trajectory, so a reliable estimation of their locations is key to the correctness of the algorithm. Our implementation uses the original implementation's method and plots the detected points projected into each new input frame, as shown in Figure 11. It saves these plots as images and makes a video from them too, thereby also making another useful visualization of the feature points and their uncertainties as they are being tracked across images.

OBSERVATIONS AND ANALYSIS

Output of algorithm on datasets (both real and phantom)

Initially, there was a version of the code which used the EKF-SLAM implementation of [7] as-is, but on a real ultrasound dataset that straightened the curvilinear image. It ended up with the following type of trajectory output:

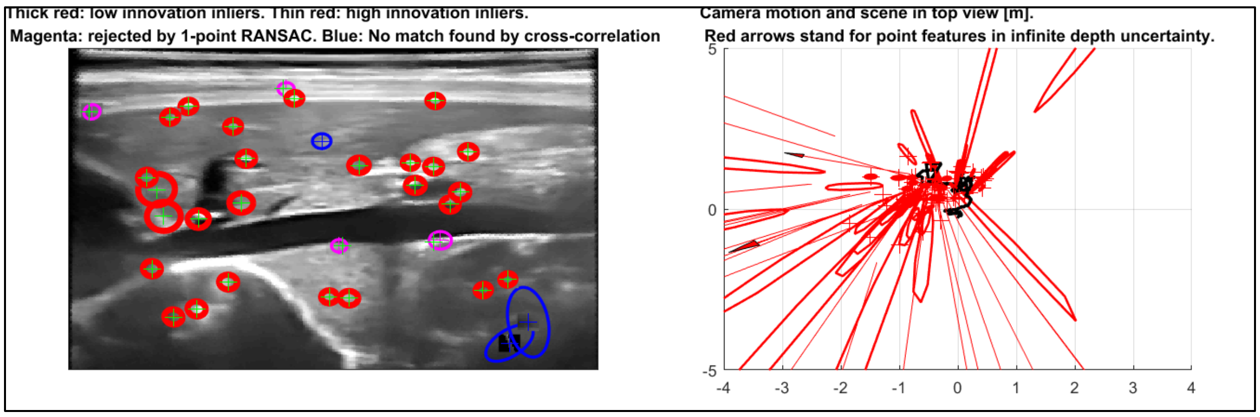


Figure 12 - Output of original algorithm of [7] *on curve-straightened real ultrasound dataset.*

Clearly, there is noticeable error in the path – the dataset was taken for a 1-degree-of-freedom straight line motion of the probe.

Hence, we made corrections to the algorithm and ended up with the one we have now. We then observed the following after running our algorithm on the first two frames of each dataset:

- The three chosen feature detectors (BRISK, SIFT and Shi-Tomasi) were able to detect point features very well in the first frame. Even when the minimum number of features required in the image was raised to 100 from the initial value of 25, that number of features could still be found within a restricted window of the image (85% of the area defined by the curve parameters). The feature detection methods were used on such a restricted window to avoid boundary points (i.e. points on the border defining the curvilinear ultrasound image) being detected as features.

- However, correlating two frames still proved to be a challenge. Depending on what was done to relax the conditions on the feature matching step, the following effects were observed:
    - Without any modifications to the algorithm compared to what had been explained above, the code crashed immediately after reading the second image and trying to start the 1-Point RANSAC model estimation process.
    - This was because the algorithm could not find any individually compatible matches (i.e. NCC-based matches of patches surrounding each feature point) across images – that is, not a single match was available to carry out the 1-Point RANSAC step.
    - Upon investigating the cause of this, we found that the correlation coefficient threshold had no effect on this – it was a problem that prohibited the feature from being matched in the first place, which meant that the correlation coefficient was never computed.
    - The problem was found to occur during the measurement prediction stage of the EKF update – the measurement covariance estimated by the algorithm was simply too large (i.e. the eigenvalues of the covariance matrix were well above the threshold of 100 defined by the original algorithm as a sanity check), thereby signaling that there was a problem in the numerical estimation of the measurement vector and covariance matrix.
    - This meant that there had to be a practical difficulty during the EKF update – possibly due to wrong initialization of velocity or covariance values, or the choice of pixel units in world space as well (which might have led to higher residual values – it is a known fact that normalization of coordinates is an essential error-reduction step in other computer vision algorithms based on linear algebra, for example).

    - The eigenvalue-based sanity check, however, turned out to be an essential step – since the image regions where matches were searched in depended on the values of the measurement covariance matrix, removing this check meant that the window where matches were searched in became too large, and therefore too many "matching patches" were returned. This in turn prevented the computation of the correlation coefficient, as memory was not enough to store and process the huge matrices arising from the computation.
- This behavior (i.e. crashing during the 1-Point RANSAC step after reading the second frame) persisted irrespective of the type of dataset used (real or phantom). Hence, it can be inferred that the problem has more to do with the setup of the algorithm itself. Due to time constraints, this problem could not be investigated further.

Analysis

Although our experiments couldn't produce any tangible results to validate, we have been able to gain several useful insights into the problem statement and the algorithm proposed over the course of the project. These are summarized below and could prove to be handy for improving further research in this domain.

- To the human eye, figuring out the exact movement of the virtual camera (and hence the ultrasound probe) is a difficult task, and it is not immediately clear that one can detect motion across all degrees of freedom simply by seeing the ultrasound video. While translation along the image plane (and hence rotation along an axis perpendicular to it) are clearly visible, motions out of the image plane are not directly reflected in the ultrasound images. This is simply because the image plane changes between frames, and since only those world points that lie on it are captured, we cannot say that points matched between two frames by conventional matching methods correspond to the same world point in general.
- That said, this would only be an issue if we use methods such as triangulation or Structure-from-Motion to estimate world point locations from feature matches. We can

avoid this problem by noting that once the virtual camera's intrinsic parameters are fixed, knowledge of the camera's world pose and feature point coordinates (in the image) is enough to reconstruct every image point in 3D from a single image (which means matching is not necessary for reconstruction). This information has been captured by the inverse depth constraint introduced earlier, and any computer vision algorithm that operates on ultrasound images would find this observation very useful.

- Since the images will be low-texture and high-noise and would also have a considerable amount of shadowing present, any feature detection method would have to be coupled with an algorithm that takes all this into account and performs robust estimation – such as RANSAC (even better, 1-Point RANSAC as used by us).
- Since motions of an ultrasound probe are smooth and no sudden jerks are usually present, the constant velocity model (as used by us) will suffice to model the motion of the probe in general, which also makes EKF a prime candidate for application to this domain. However, the initial values of velocities and their covariances are likely to play a role – in our case, they might have been the cause of the entire problem.
- The tracker was attached to the top of the probe handle during data collection for the real-world dataset. The point that is being modelled as the probe virtually, on the other hand, lies at the center of the curve circle – which doesn't lie at the top of the real probe in general. The distance between the modelled probe point and the actual point being tracked introduces errors with respect to the "ground truth" because the modelled point's pose is what the algorithm estimates while the tracked point's pose is what has been recorded as the ground truth. This discrepancy hasn't been resolved yet.

CONCLUSION

We have implemented an algorithm, based on the existing EKF-SLAM with 1-Point RANSAC algorithm [7], to compute the trajectory of the ultrasound probe over the duration of an ultrasound video, using only the video as input – for which the initial inspiration came from related work on endoscopic video. In doing so, we have made several useful observations and contributions regarding the problem statement – the formulation of the virtual camera, the inverse depth constraint, the evaluation measures and all the insights gained over the course of this work – which can be instrumental in guiding future research in this domain. Due to time constraints, the problems surrounding the application of the algorithm to this problem remain unsolved, but work will continue, and solutions to the problems will be investigated.

However, the algorithm as it stands now has a few shortcomings. The rigidity constraint enforced throughout can be prohibitive in mapping the internals of a human body (as done in ultrasound) because there are many kinds of moving entities inside it. The requirement that the patient's body must remain still (and even maintain the same state of breath) is also a slight issue, as is the requirement that the ultrasound probe's parameters (gain, zoom etc.) should also stay constant. The running time of the algorithm is also quite high, especially when a higher number of features are requested to be initialized, which makes it unsuitable for online applications – though our use case wasn't intended to be online.

In future, it would be prudent to explore new methods of feature detection and matching, perhaps unconventional ones tailored to the ultrasound domain, and analyze their impact on the algorithm. These new methods could possibly factor in the information about the organ being scanned and use specialized algorithms accordingly (e.g. use a specific feature type to detect parts of the liver in a liver ultrasound scan). The feature matching stage may need modifications to accommodate the different constraints imposed by the ultrasound problem statement. The existing algorithm and detectors also provide good scope for customization, with various parameters that can be tweaked to all for detection of more (or better) features;

the impact of these parameters on the algorithm could be studied to gain more insight into it. Further, the input images could be preprocessed more, perhaps by image adjustment for contrast, so that the subsequent steps can yield more reliable results (better features, better matching etc.) – although one must take care not to influence what the camera sees too much, as that may cause deviations from the ground truth. Still other approaches could try and avoid the topic of features altogether – since, as we have seen, feature detection and matching remain a challenge - and operate on the images “as a whole” without omitting any information in them. This has been done for a different computer vision problem by [16], using a method called Large-Scale Direct SLAM (LSD-SLAM); investigation into methods of integration of those concepts with the virtual camera and other such ultrasound-specific details could prove to be very useful.

However, it can be safely said that the virtual camera concept and the inverse depth constraint, together with the other insights gained during this work, would be very important for computer vision algorithms on ultrasound images going forward. Other types of unconventional image sources (where the image is not a typical camera render, as is the case with different types of medical imaging, for example) could be analyzed using a similar approach (i.e. by introducing a virtual camera looking into the image); those systems would introduce different, specific constraints but the general approach would stay the same.

REFERENCES

[1] B. Lin, “Visual SLAM and Surface Reconstruction for Abdominal Minimally Invasive Surgery,” 2015.

[2] O. G. Grasa, J. Civera, and J. M. M. Montiel, “EKF monocular SLAM with relocalization for laparoscopic sequences,” in *Proceedings - IEEE International Conference on Robotics and Automation*, 2011.

[3] U. Pardasani, “Neurosurgical Ultrasound Pose Estimation Using Image-Based Registration and Sensor Fusion - A Feasibility Study,” no. November, 2016.

[4] D. G. Gobbi, R. M. Comeau, and T. M. Peters, “Ultrasound Probe Tracking for Real-Time Ultrasound/MRI Overlay and Visualization of Brain Shift,” pp. 920–927, 1999.

[5] R. Mebarki, A. Krupa, and F. Chaumette, “2-D ultrasound probe complete guidance by visual servoing using image moments,” *IEEE Trans. Robot.*, vol. 26, no. 2, pp. 296–306, 2010.

[6] P. J. Stolka, H. J. Kang, M. Choti, and E. M. Boctor, “Multi-DoF probe trajectory reconstruction with local sensors for 2D-to-3D ultrasound,” *2010 7th IEEE Int. Symp. Biomed. Imaging From Nano to Macro, ISBI 2010 - Proc.*, pp. 316–319, 2010.

[7] J. Civera, O. G. Grasa, A. J. Davison, and J. M. M. Montiel, “1-Point RANSAC for EKF Filtering. Application to Real-Time Structure from Motion and Visual Odometry,” *J. F. Robot.*, vol. 27, no. 5, pp. 609–631, 2010.

[8] S. Leutenegger, M. Chli, and R. Y. Siegwart, “BRISK: Binary Robust invariant scalable keypoints,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2011.

[9] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *Int. J. Comput. Vis.*, 2004.

[10] Jianbo Shi and Tomasi, “Good features to track,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR-94*, 1994.

[11] C. Harris and M. Stephens, “A Combined Corner and Edge Detector,” in *Procedings of the Alvey Vision Conference 1988*, 1988.

[12] H. Bay, T. Tuytelaars, and L. Van Gool, “SURF: Speeded up robust features,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2006.

[13] E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2006.

[14] J. Civera, A. J. Davison, and J. M. M. Montiel, “Inverse Depth Parameterization for Monocular {SLAM},” *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 932–945, 2008.

[15] K. Fathian, J. P. Ramirez-Paredes, E. A. Doucette, J. W. Curtis, and N. R. Gans, “Quaternion Based Camera Pose Estimation From Matched Feature Points,” 2017.

[16] J. Engel, T. Schöps, and D. Cremers, “LSD-SLAM: Large-Scale Direct monocular SLAM,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014.