Chair for Computer Aided Medical Procedures & Augmented Reality
Department of Computer Science
Technical University of Munich

# Robotically-Guided Ultrasound System for 3D Liver Reconstruction

Final Report of Master Praktikum Surgineering WS 2022/2023

**Christian Engel** ✉, **Mei-Ling Fang** ✉, **and Chengzhi Shen** ✉

Department of Computer Science, Technical University of Munich

✉ christian.engel@tum.de
✉ julie.fang@tum.de
✉ chengzhi.shen@tum.de

August 8, 2023

**Abstract** —
Hepatocellular Carcinoma (HCC) is a prevalent and lethal type of cancer, with Transcatheter Arterial Chemoembolization (TACE) being the standard treatment option. Fluoroscopy is commonly used as the intra-operative imaging modality for guidance during TACE. However, it results in significant exposure to radiation for both patients and medical personnel. To address this challenge, we propose an automatic robotically-guided Ultrasound (US) system that acquires optimal image during scanning, which can be used to construct high quality 3D compounding after scanning. Our approach can be primarily divided into three distinct parts: initial point detection, trajectory planning and robot control, and image shadow detection. As occlusions can lead to undesired shadows during US scan, we further develop a shadow-aware trajectory planning approach that dynamically adjust the predefined trajectory according to current image quality. Geometrical and physical constraints are integrated in the pipeline to estimate the best ultrasound acquisition trajectories with respect to the available acoustic windows. We evaluate the developed method on a tissue-mimicking phantom and demonstrate the acquired US image quality outperforms native plannings. Our code is available at the GitLab repository[1].

## 1 Introduction

Liver disease is a significant global health issue affecting millions of people worldwide. Early detection and accurate diagnosis of liver diseases are crucial for effective treatment and management. Ultrasound imaging is one of the most widely used imaging modalities for liver diagnosis due to its non-invasive, safe, and cost-effective nature. However, conventional ultrasound imaging techniques rely on manual scanning, which can result in operator-dependent variability and inconsistency in image quality.

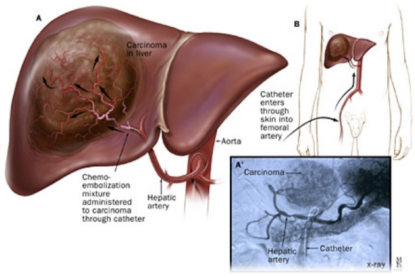



**Figure 1** Illustration of TACE procedure. A catheter is inserted to block the blood supply as well as to release chemotherapeutic agent near the tumor site.

In this report, we present the development and evaluation of the liver scanning system based on the previous work of the students. We optimized the design of the system and tested it on a phantom liver to evaluate its performance. Additionally, we outline our individual contributions to the project, providing a breakdown of the tasks we completed.

The results of our study demonstrate that our liver scanning system can provide high-quality ultrasound images with minimal operator-dependency. This system has the potential to improve liver disease diagnosis and management, ultimately leading to better patient outcomes.

## 2 Methods & Contribution

Throughout the project, each team member made significant contributions to the development and evaluation of the system. Below provides details on our proposed approach, as well as the individual contribution of each member.

[1] https://gitlab.lrz.de/computational-surgineering/rus_liver_scan

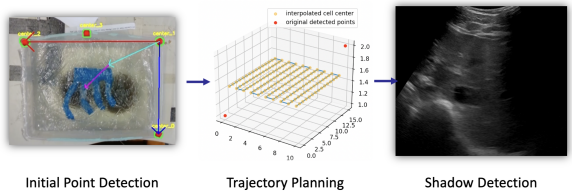



**Figure 2** The technical breakdown of our proposed pipeline.

## 2.1 Initial Point Detection

To realize autonomous liver scanning, the robot manipulator must initially perceive its surroundings using the sensors mounted on its end-effector or fixed around the robotic base. Afterwards, a scan trajectory can be planned after locating the target tissue, which may have anatomical markers or markers manually attached. In this project, Mei-Ling implemented three sequential processes - *eye-in-hand calibration*, *color sticker detection*, and *initial scan point determination* - to gather perception information about the target tissue. These processes were essential for the downstream task of scan trajectory planning. The code can be found on the abovementioned GitLab repository, and they require the *RealSense SDK* 2.0, *OpenCV* 4.5.4, and *ROS Noetic* to run successfully under Ubuntu 20.04. These processes are elaborated below step-by-step.

### 2.1.1 Hand-Eye Calibration

In robotics, the hand-eye calibration problem is to determine the homogeneous transformation matrix between a robot end-effector and a camera sensor, or between a robot base and the world coordinate system. It is a fundamental but crucial task done offline. Hand refers to the robotic arm, and eye the camera. There are two types of hand-eye calibration depending on where the camera is fixed:

- Eye-in-hand: the camera is mounted on the robot
- Eye-to-hand: the camera is stationarily mounted next to the robot

In our setting, a realsense camera D435 is mounted near the end-effector of the robotic arm, therefore, eye-in-hand calibration is used. Let us denote the $\{\mathcal{B}\}$, $\{\mathcal{F}\}$, $\{C\}$, $\{O\}$ as the coordinate frames of the robotic base, robotic flange/final link, the camera and the calibration board, respectively.

Eye-in-hand calibration is a process for identifying the relative position and orientation of $\{C\}$ with respect to $\{\mathcal{F}\}$, usually presented as a homogeneous matrix ${}^{\mathcal{F}}\mathbf{T}_C$, where ${}^{i}\mathbf{T}_j$ stands for transformation from $\{i\}$ to $\{j\}$. The calibration is usually done by capturing a set of images of a fixed calibration board under different robot joint configurations results in different poses of $\{C\}$ with respect to $\{\mathcal{B}\}$. The calibration board and the robotic base are fixed during calibration. The mathematical description of the hand-eye relationship:

$$ {}^{\mathcal{B}}\mathbf{T}_{\mathcal{F}_2}\,{}^{\mathcal{F}_2}\mathbf{T}_{C_2}\,{}^{C_2}\mathbf{T}_O = {}^{\mathcal{B}}\mathbf{T}_{\mathcal{F}_1}\,{}^{\mathcal{F}_1}\mathbf{T}_{C_1}\,{}^{C_1}\mathbf{T}_O \tag{1} $$

where the lower right subscripts of $\mathcal{F}$ and $C$ represents the transformation matrices sampled under two different robot joint configurations. After shifting the matrices, we get:

$$ {}^{\mathcal{B}}\mathbf{T}_{\mathcal{F}_1}^{-1}\,{}^{\mathcal{B}}\mathbf{T}_{\mathcal{F}_2}\,{}^{\mathcal{F}_2}\mathbf{T}_{C_2} = {}^{\mathcal{F}_1}\mathbf{T}_{C_1}\,{}^{C_1}\mathbf{T}_O\,{}^{C_2}\mathbf{T}_O^{-1} \tag{2} $$

It then takes the following form since and ${}^{\mathcal{F}_1}\mathbf{T}_{C_1} = {}^{\mathcal{F}_2}\mathbf{T}_{C_2}$ remains constant since the $\{C\}$ is rigidly mounted with respect to $\{\mathcal{F}\}$.

$$ \mathbf{AX} = \mathbf{XB} \tag{3} $$

where $\mathbf{A} = {}^{\mathcal{B}}\mathbf{T}_{\mathcal{F}_1}^{-1}\,{}^{\mathcal{B}}\mathbf{T}_{\mathcal{F}_2}$, $\mathbf{B} = {}^{C_1}\mathbf{T}_O\,{}^{C_2}\mathbf{T}_O^{-1}$, and $\mathbf{X} = {}^{\mathcal{F}}\mathbf{T}_C$ is the calibration result. Thus, ($\mathbf{A}$, $\mathbf{B}$) constitutes one sample for the calibration. Fig. 3 presents

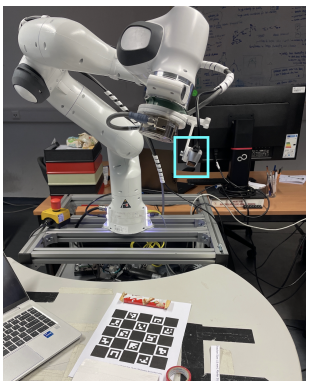


**Figure 3** A realsense camera D435 (marked in cyan color) is rigidly attached to the robotic flange using a 3D printed mounting fixture, which is designed and printed in the IFL lab. A `5x5` ChArUco board is used for calibration.

the calibration setup where the samples are collected by changing the joint configuration of the robot manipulator while at the same time detecting the pose of the calbration board expressed in $\{C\}$. The default calibration algorithm that solves the augmented Eq. (3) is set to be `Tsai-Lenz` [7], a long standing eye-in-hand configuration technique for computing position and orientation of a camera relative to the last joint of a robot manipulator. Other algorithms are available and listed below: `Park` for robot sensor calibration [5],

Horaud for hand-eye calibration [3], Andreff for on-line hand-eye calibration [1] and Daniilidis for hand-eye calibration using dual quaternions [2]. Users can specify the algorithm under the same ros package when launching the calibration GUI. The method was first tested on the Franka robotic arm, and then on the Kuka arm after minor adjustment.

This step was crucial in establishing an accurate relationship between the camera frame and the robotic arm (i.e. link 7 on Kuka) coordinate systems, which ensured precise scanning. It is advised to sample at least 20 points per calibration. The positions of the robotic end-effector should be as different as can be, in order to ensure high accuracy of calibration result. After sampling, press compute button in the GUI (see figure 4) to calculate the final calibration matrix. One can also publish the result to ros topic for downstream task to subscribe.

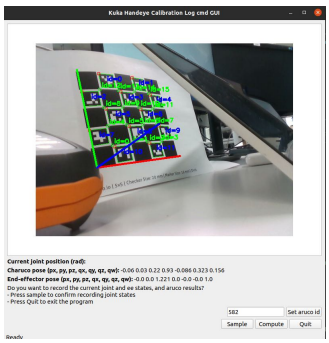



**Figure 4** A GUI from point sampling app using ChArUco

As for calibration board, calibration can be performed based on ArUco markers corners or ChArUco corners using the ArUco module from OpenCV. Both methods are implemented and available under ros package franka_handeye_cali. Traditionally, chessboard is used to perform calibration. However, calibrating using ArUco is much versatile than traditional chessboard patterns, since it allows occlusions or partial views. Furthermore, it is highly recommended using the ChArUco corners approach (see figure 5) since the combination of both provide much more accurate result in comparison to the marker corners alone.

One way to judge whether the calibration is accurate enough is to visualize the result in the ros plugin rviz. Place an ArUco marker on the table, then move the end-effector of the robotic arm to different positions to detect the marker. Through observing the changes of the ArUco marker position with respect to the base position of the robot, one can conclude that the calibration is accurate if the variance remains low

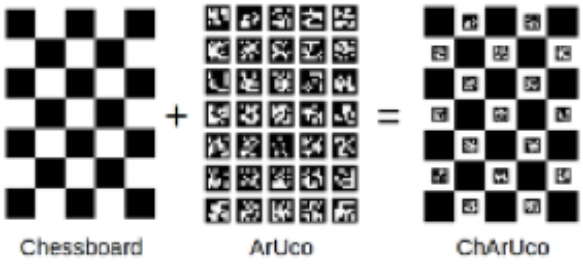



**Figure 5** ChArUco board is a combination of both chessboard and ArUco marker corners.

enough. Note that OpenCV version 4.5.4 is used in the pipeline, but a different version 4.7.0-dev can also be found in the code comment.

In general, calibration has to be performed as long as the position of the camera has shifted. In addition, ChArUco board with different configuration size can be used, but users have to manually specified the board information such as square size, length, marker size, in the rs2_charuco_detector.py under franka_handeye_cali package. One can use the online pattern generator[2] to produce a desired ChArUco board.

To conclude this part of work, we aim to retrive the transformation matrix between Kuka link 7 (end-effector) and the camera frame, see the calibration result in figure 6.

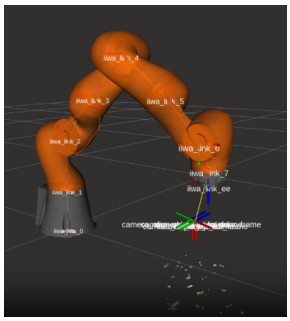


**Figure 6** A visualization in rviz after hand-eye calibration performed on Kuka robotic arm.

### 2.1.2 Marker Detection

The final outcome of this section is to detect manually placed markers on the tissue-mimicking phantom, in order to construct a coordinate system on the camera detected 2D images and decide an initial point to start ultrasound scanning. To start, 4 red stickers are placed in different positions on the edge of the phantom. Through color segmentation and morphological

[2]https://calib.io/pages/camera-calibration-pattern-generator

transformations, the stickers are detected and circled. To avoid noises, the detected area are sorted by ascending order and then only first N largest points are returned.

Since the distance between stickers remains static, it is fairly simple to calculate the relative positions between points and define a coordinate system.

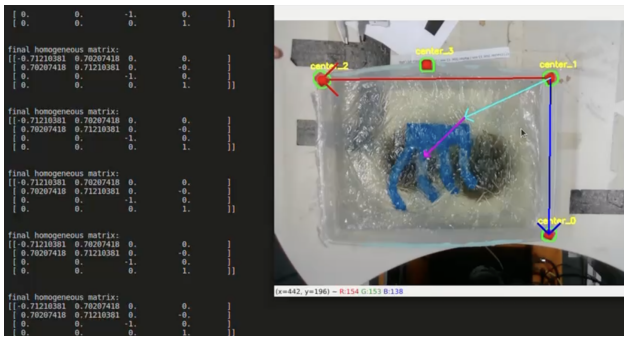


**Figure 7** On the right hand side, a phantom coordinate system is shown. Red vector denotes $x$ coordinate and blue $y$. Cyan vector points to the starting point for scanning, and the purple vector suggests the depth. On the left hand side, a homogeneous matrix is calculated and printed on the console.

In our final setup, only 2 points are required to divide the phantom for downstream trajectory planning. However, the function is flexible and easily configurable if more markers are needed. It can be revised in the `initial_point_detection.py` under `iiwa_core` package. Since the pixel points $(u,\ v)$ are in 2D coordinates, one needs to convert it to a 3D point $(x_b,\ y_b,\ z_b)$ expressed in $\{\mathcal{B}\}$ (see Fig. 8) using the depth image $\mathcal{I}_{depth}$ subscribed from the `ros` topic, the eye-in-hand calibration result ${}^{\mathcal{F}}\mathbf{T}_C$ obtained offline and the forward kinematics ${}^{\mathcal{B}}\mathbf{T}_{\mathcal{F}}$ subsribed from the `ros tf tree`.

**1). Convert 2D pixel coordinates to 3D points in $\{C\}$:**

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = z \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{4}$$

where $z = \mathcal{I}_{depth}(u,\ v)$ denotes the depth value at the $(u,\ v)$ image coordinate, $f_x$, $f_y$, $u_0$ and $v_0$ are the camera's intrinsic parameters. $(x_c,\ y_c,\ z_c)$ denotes the 3D points expressed in $\{C\}$.

**2). 3D points transformation from $\{C\}$ to $\{\mathcal{B}\}$:** the 3D point $(x_b,\ y_b,\ z_b)$ in $\{\mathcal{B}\}$ is eventually calculated by the following equation:

$$\mathbf{p}_b = {}^{\mathcal{B}}\mathbf{T}_{\mathcal{F}}{}^{\mathcal{F}}\mathbf{T}_C\mathbf{p}_c \tag{5}$$

where $\mathbf{p}_b = [x_b,\ y_b,\ z_b,\ 1]^T$ and $\mathbf{p}_c = [x_c,\ y_c,\ z_c,\ 1]^T$.

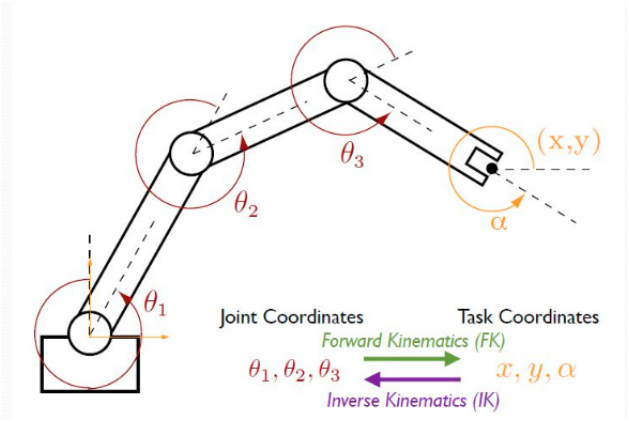



**Figure 8** An illustration of forward and inverse kinematics, highlighting the transformation relationship between coordinates.

## 2.2 Robot Control Pipeline and Trajectory Planning

The robot control pipeline is at the center of our proposed system, which manages all the communication among major functionalities via ROS. Beside the basic robot movement, it is highly beneficial to develop protocols based on request-subscribe via ROS that executes other back-end applications upon request, instead of constantly running them all the time, which causes system overload and application crashes. Furthermore, both automatic and human-involved safeguards are necessary to maintain the safety while executing the pipeline.

Chengzhi implement the robot controller and all basic robot movements, including camera position, trajectory planning, scanning protocol, sweeping motion, and safeguards. He is also responsible for force adaptation and integration of the whole pipeline.

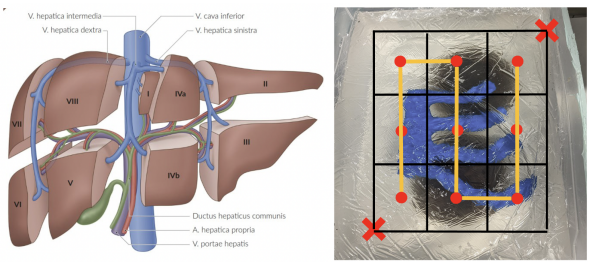


**Figure 9** The approximation of medical scan protocol (left) with grid cells (right).

### 2.2.1 Camera Position

At the beginning of the robot control pipeline, the robot end effector is set to a high position while maintaining a perpendicular orientation to the target. This is denoted as the Camera Position. It allows a better view in the RGBD camera for initial point detection and depth estimation, as well as a standardalized location with reproducibility. The Cartesian pose of the camera position is empirically set to $x = 0.5$, $y = 0$, and $z = 0.45$. The unit is in meters.

### 2.2.2 Trajectory Planning and Scanning Protocol

When performing a medical liver scan, doctors typically divide the liver into regions to better analyze and diagnose any potential issues. Since the pipeline is primarily developed on the phantom, we approximate the doctor's workflow by dividing the scanning area into regions as shown in Figure9.

Specifically, given the Cartesian position of two detected points in diagonal $p_1 = (x_1, y_1, z_1)$, $p_2 = (x_2, y_2, z_2)$, the scan region is divided into a 3x3 grid in the x-y plane using $(x_1, y_1)$ and $(x_2, y_2)$. Once the grid is established, the center points of each cell are extracted, which serve as the anchor points for constructing further trajectories. Note that for all center points $p_i = (x_i, y_i, z_i)$, $z_i = \frac{z_1+z_2}{2} + offset$, that is the average estimated depth of two points plus the offset. As the phantom is relatively flat, the initial trajectory is preferred to have the same depth (values in z-dimension). Moreover, the offset is applied for safety reasons to ensure that the initial trajectory will always be above the phantom surface. Eventually, the end effector will move down to find an optimal contact with the surface thanks to the force adaptation mechanism, as described in Section 2.2.3.

To form a higher-resolution trajectory, an interpolation is applied between the anchor points using a fixed step-size of $1cm$. All trajectory points are ordered into a zig-zag form and sent to the robot.

Such scanning protocol allows for a more structured and standardized approach to robotic liver scan analysis. In future developments, advanced scanning protocols can be developed that incorporate various domain-specific knowledge and adapt to different scanning targets.

### 2.2.3 Force Adaptation

To ensure safe and effective US scanning, robotic systems need to adapt their contact force and movement in response to changes in the scanning target. This requires keeping the contact force within an optimal range to maximize scanning quality while avoiding harm to the patient. Figure 10 illustrates the tradeoff between contact force and quality of the US image acquired at the same location. Small contact force may lead to undesired shadow, while large contact force may cause danger. Hence, we need to find an optimal contact force range that offers good image quality while maintaining safety.

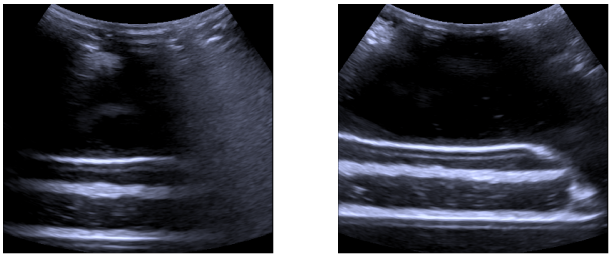


**Figure 10** The tradeoff between contact force and image quality. Small contact force leads to poor image quality (left), while optimal contact force gives good image quality (right.)

In practice, the optimal force range was determined to be between $F_{min}$ and $F_{max}$, with the target force $F_T$ set to the average of the range. We empirically set $F_{min} = 1N$, $F_{max} = 2N$, and $F_T = 1.5N$. We constantly monitor the robot's contact force by subscribing to `iiwa_msgs.msg.CartesianWrench` and adjust the end effector's position to approach $F_T$ at each trajectory point $p_i$.

The force adaptation mechanism is realized by adjusting the Cartesian pose at the z-axis with a fixed step-size. In the beginning, a large step-size (e.g. $1cm$) is adopted to enable faster movement of the robot arm. However, large step-size may overshoot the force range, making it hard for the force adaptation algorithm to converge. Therefore, we log the motion status of the end effector and further define the oscillation in the form of [down, up, down]. An oscillation-aware mechanism is developed that reduces the step-size to half when an oscillation is detected. Such a mechanism allows finer control, as large step-size can lead to overshooting issues.

### 2.2.4 Sweeping Motion

During the US liver scan, there are undesired shadows caused by ribs that leads to poor scanning quality. To tackle this challenge, we propose to obtain more information at the non-shadow location by sweeping the

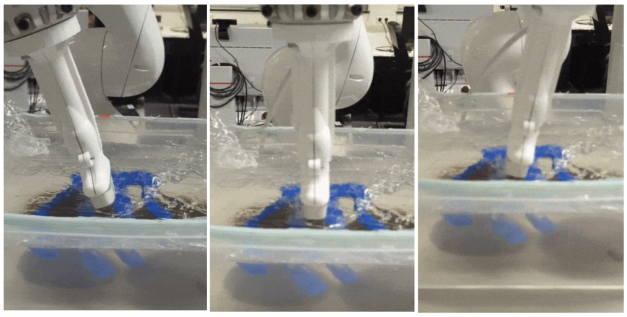


**Figure 11** The sweeping motion of the probe at fixed point

end-effector at different angles, which can be helpful to 3D compounding after the scanning.

It is notable that the US probe should sweep around the fixed contact point, in this case, the tool tip, as shown in Figure11. Thus, an extra tool transformation is needed that maps the end-effector coordinate to the tool tip coordinate.

ROS supports adding the tool transformation by running the command `rosparam set /iiwa/toolName <Name>` , where `<Name>` is the corresponding config name for the tool. Note that the `rosparam` command should be run after roscore, but before starting the SmartServo application on the robot control panel.

To implement the sweep motion, a quaternion $Q_t$ is first read by subscribing to the CartesianPose, which is converted to a rotational matrix $R_t$ that describes the current orientation of the tool. Then a dot product is taken between $R_t$ and the new rotational matrix $R_{rot}$ with desired rotational angle. The final result $R_{sweep}$ is converted back to quaternion $Q_{sweep}$ and published to ROS. Mathematically, this process can be described as:

$$R_{sweep} = R_{rot} R_t \tag{6}$$

$$Q_{sweep} = Oper(R_{sweep}) \tag{7}$$

where $Oper$ is the operator that converts rotational matix to quaternion.

### 2.2.5 Sweeping Protocol

Additionally, a sweeping protocol is implemented to decide the location for sweeping, namely the first clear trajectory point after escaping the shadow area. At each trajectory point, a shadow detection request is sent to ROS, and wait for the detection result. If there is a shadow, then the robot moves to the next trajectory point and gets the shadow detection result again. This process is executed until the point that receives the non-shadow prediction (i.e. the shadow region is escaped). Finally, the sweeping motion is performed.

### 2.2.6 Robot Control Pipeline

The robot control pipeline is illustrated in Figure12. Note that different modules are communicated via publishers and subscribers to ROS. Several request-subscribe protocols are designed to optimize the application's efficiency.

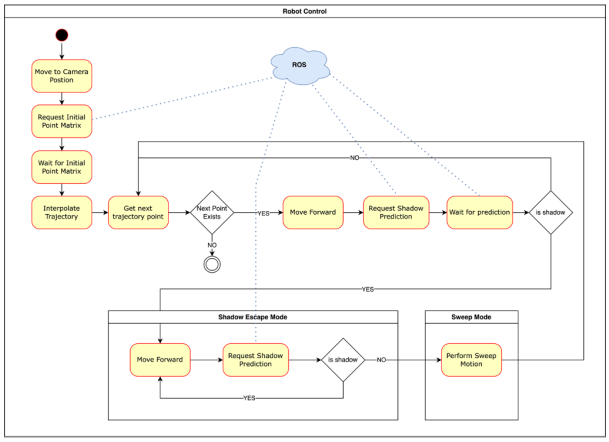



**Figure 12** A state diagram illustrating the workflow of the robot control pipeline

**initial point request publisher** (msg-type: Bool): after moving to the initial camera position, the pipeline publishes a request to the initial point detection module that asks for the detection result. Note that we only need the initial point detection once, thus the request is sent for only one time.

**initial point subscriber** (msg-type: numpy_msg(Floats)): once the initial points are detected and published, the pipeline subscribes the results as a numpy array in floats. Note that the subscribed message is received only once.

**pose subscriber** (msg-type: CartesianPose): get the cartesian pose of the robot's end effector. The CartesianPose message contains both position and orientation.

**pose publisher** (msg-type: CartesianPose): publish the target pose of the robot's end effector to move the robot along the trajectory. shadow prediction request publisher (msg-type: Bool): once the end-effector reaches the point in trajectory, the pipeline publishes a request to the shadow detection module. This allows a higher efficiency of the application as we do not require the GPU to constantly run the deep learning model unless requested.

**shadow prediction subscriber** (msg-type: Bool): get the shadow detection result. This message is used

in the pipeline to escape shadow areas and perform sweeping motions.

**force subscriber** (msg-type: CartesianWrench): get the current force of the robot's end-effector. This message is used in the force adaptation module.

### 2.2.7 Safeguards

It is very important to have safe guards in the robot control pipeline to prevent any undesired damages. We adopt the following safe guards in our pipeline:

**range check**: the published target position should stay within the safe range, x in [0.4, 0.7], y in [-0.7, 0.7], and z in [-0.2, 0.5].

**reach target check**: check if the end effector's Cartesian pose has reached the desired position, by comparing and different between current pose and target pose with error $\epsilon$. This prevent the robot from skipping some trajectory points due to the high rate in ROS command and slow motion speed.

**human in the loop:** the result of initial point detection and depth estimation is visualized via OpenCV and printed in logs. Human decision like "Press Space to Continue" are required before running the pipeline

**the sleep mechanism:** the rospy.sleep() allows us to pause the execution of a ROS node for a specified amount of time, which is necessary to create delays between publishing or subscribing to messages on topics. Without the sleep mechanism, the pipepline may suffer from unsuccessful initialization, out of time error, or other crashes.

### 2.2.8 Other work by Chengzhi

Chengzhi provide additional support to Mei-Ling for implementing the ROS connection, transformations from pixel coordinate to robot base coordinate, and marker detection. Chengzhi also works on the 3D compounding part as demonstrated in Section 3

## 2.3 Real-Time Shadow Detection

The third primary component in our pipeline encompasses a real-time shadow detection application, which provides feedback to dynamically modify the trajectory and prompt the probe to execute a sweep in the subsequent shadow-free region. To accomplish this, Christian collected multiple datasets of phantom liver scans and implemented as well as assessed four main classification strategies, some of which employed deep-learning techniques.

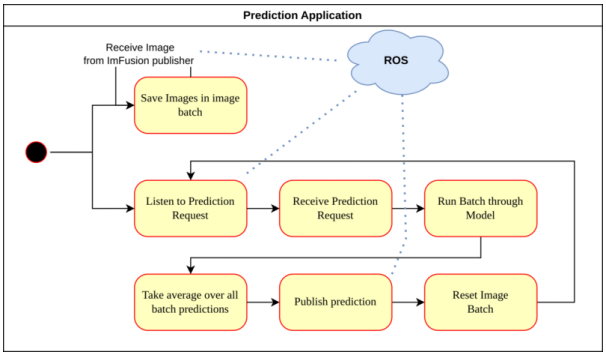



**Figure 13** A state diagram illustrating the workflow of the implemented prediction app

### 2.3.1 Histogram-based Classification

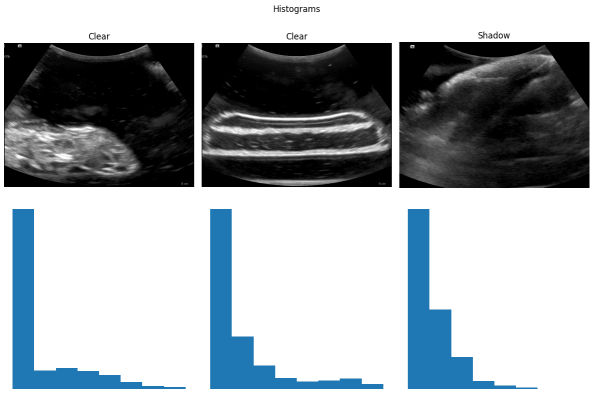



**Figure 14** Comparison of Histograms for Shadow and Non-Shadow Images

The first method employed a histogram-based classification technique. This approach involved using two template histograms, one corresponding to shadow images and the other to non-shadow images, to compare the histograms of a new image with these templates by employing a distance metric. The Euclidean distance served as the chosen metric.

$$d(H_1, H_2) = \sqrt{\sum_{i=1}^{n} (H_{1,i} - H_{2,i})^2}$$

where $H_{1,i}$ and $H_{2,i}$ are the bin counts of bin $i$ in histograms $H_1$ and $H_2$, respectively.

However, as illustrated in Figure 14, the histograms of shadow and non-shadow images (particularly the second and third images) lacked a distinct separation, rendering this method unsuccessful.

### 2.3.2 Pseudo-Label Selection-based Classifier

Christian subsequently explored deep learning techniques, such as Resnet-based classifiers, for the

classification task. However, these models require large quantities of labeled data for training or fine-tuning. One way to address this issue is to use a semi-supervised learning approach known as pseudo-labeling. In the approach proposed by Rizve et al. [6], a small portion of the entire dataset is initially manually labeled, and a pre-trained classifier is trained using this subset. The classifier predicts labels for the remaining dataset in each iteration. A subset of these pseudo-labels is then chosen based on prediction confidence and prediction uncertainty, which is calculated using Monte Carlo sampling using ten stochastic forward passes. The classifier is trained on an increasingly more significant dataset subset in each iteration. The ultimate objective is to train the classifier on the full dataset with reliable labels.

**Figure 15** Shadow or Non-Shadow?

Upon implementing and evaluating this method in a binary classification context, it became evident that, while early iterations demonstrated robust training, instability emerged in later iterations, resulting in diminished performance on the validation set over time. An examination of sample predictions revealed that the classifier struggled with ambiguous frames containing shadowy and non-shadowy regions. Figure 15 provides an example of an ambiguous frame. Despite modifying the implementation to accommodate multi-label predictions, the classifier could finally not perform well on these frames, leading to the abandonment of this approach.

### 2.3.3 Refinement using Confidence Maps

The third method employed a refinement of the pseudo-labeling technique. Rather than directly classifying ultrasound images, an initial step involved computing a confidence map of each image. This aimed

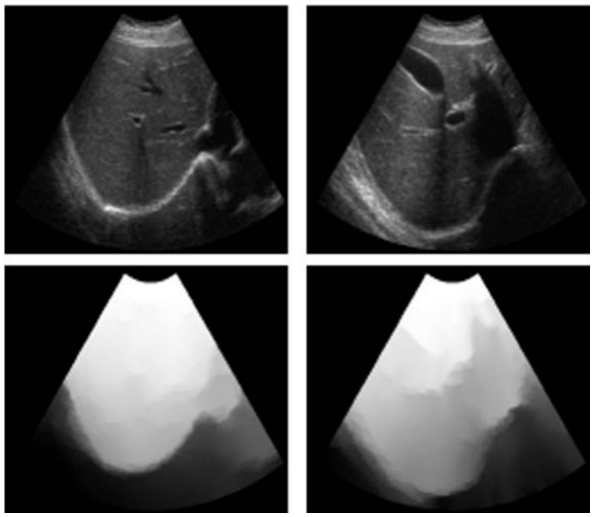


**Figure 16** Confidence Maps on Ultrasound images [4]

to reduce noise and enhance interpretability for the classifier. Confidence maps serve as a means to assess uncertainty in ultrasound images, resulting from shadows and other attenuation artifacts [4]. An exemplar of a confidence map is depicted in Figure 16.

Implementing this approach was realized using an API call to ImFusionSuite for applying confidence maps to image batches. This augmented the classifier's overall efficacy; however, it continued to underperform on ambiguous frames in subsequent iterations. Moreover, the computation of confidence maps proved resource-intensive, preventing a real-time application. Specifically, the generation of confidence maps for a batch of only 10 images required over 40 seconds on our workstation. Consequently, this method was disregarded.

### 2.3.4 Embedding-Based Clustering

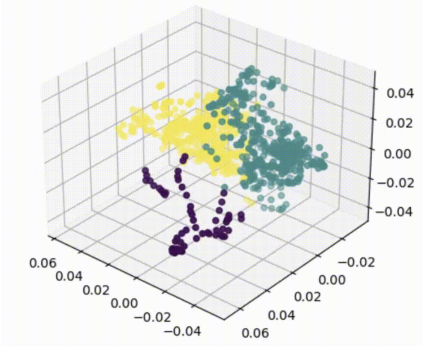



**Figure 17** Clustering Outcome

Christian subsequently investigated an embedding-based clustering method as an alternative to previous techniques. In this strategy, a fully convolutional au-

toencoder was initially trained to derive meaningful embeddings from ultrasound images. Following the initial training, the encoder was frozen, and embeddings were extracted from the training images. Dimensionality reduction was then accomplished by applying PCA with 32 components to the embeddings. K-means clustering was subsequently utilized to categorize images based on the reduced embeddings. While clustering with $k = 2$ still exhibited ambiguity, clustering with $k = 3$ proved successful. A figure depicting the clustering outcome is shown in Figure 17.

This approach proved to be successful, exhibiting both reliability on the validation set and a compact model size for the encoder. Concrete implementation details can be found in the supplied Jupyter notebooks and python source code files. This enabled real-time classification of large image batches containing up to 512 images on our workstation.

#### 2.3.5 Implementation Details

The final implementation employed the embedding-based clustering method and a ROS connection interface. The application subscribes to two topics as illustrated in Figure 13. The first topic continuously publishes ultrasound images captured using a frame grabber, while the second topic encompasses prediction requests made by the trajectory application. The prediction application maintains a window of 100 images in memory.

Upon receiving a prediction request, the application initially extracts the embeddings of the images in the current batch. These embeddings are then reduced to 32 dimensions using PCA and fitted to precomputed clusters. Based on the cluster labels, the application determines whether the current batch contains a majority of shadow or non-shadow i mages and subsequently publishes the boolean result to another topic to which the trajectory application subscribes.

#### 2.3.6 Other work by Christian

Christian provided additional support to Mei-Ling for implementing the ROS connection and the 2D-to-3D transformation process and to Chengzhi for developing the force adaptation mechanism, the sweeping protocol, and the control pipeline for the robot.

## 3 3D Compounding

3D compounding of US scans can provide a better view for medical analysis. We record the US image together with the robot pose during the robotic scanning and leverage the off-the-shelf 3D compounding algorithm provided in ImFusion to construct the 3D model. It is very import to reset the scan size to world-size according to the scanning configs to correct the 3D stack of US images from deformation and overlapping, as a prerequisite for meaningful compounding results. This is demonstrated in Fig.18. The 3D compounding results is shown in Appendix 20. The 3D compounding part is done by Chengzhi.

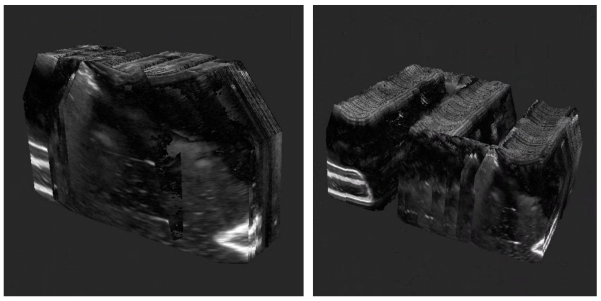


**Figure 18** The 3D stack of US images. Left: before correction, the images are overlapping, causing error for the compounding. Right: after correction, the images are correctly stacked in the zig-zag trajecory.

## 4 Conclusion & Future Work

In conclusion, our team successfully developed and assessed a robotically-guided ultrasound system for 3D liver reconstruction, which exhibited high versatility in initial point finding on our liver phantom, dynamic and reliable trajectory planning and adjustment, and dependable shadow predictions, all with minimal operator dependency. The system's workflow, which encompasses hand-eye calibration, initial point detection, trajectory definition and adjustment, sweep motions, and shadow predictions, proved effective in acquiring high-quality and consistent ultrasound images for 3D compounding. The contributions of each team member played a crucial role in the development and optimization of the system, with individual members focusing on distinct aspects and supporting each other.

In future work, the developed application can serve as a solid foundation for transitioning from phantom scans to real liver scans on humans. By building upon the success of the current system, coming students can further refine and adapt the techniques employed to address the unique challenges and complexities associated with human liver scans. This progression could advance the field of ultrasound imaging significantly, ultimately enhancing the quality and consistency of

liver scans while minimizing operator dependency, thereby improving patient care and diagnostic capabilities.

## Acknowledgement

We thank Dani Velikova, Bi Yuan, Dianye Huang, Ardit Ramadan, and Thomas Wendler for their consistent support throughout the project.

## References

[1] N. Andreff, R. Horaud, and B. Espiau. On-line hand-eye calibration. *Second International Conference on 3-D Digital Imaging and Modeling (Cat. No.PR00062)*, pp. 430–436, 1999.

[2] K. Daniilidis. Hand-eye calibration using dual quaternions. *The International Journal of Robotics Research*, 18:286 – 298, 1999.

[3] F. Dornaika and R. Horaud. Simultaneous robot-world and hand-eye calibration. *IEEE Trans. Robotics Autom.*, 14:617–622, 1998.

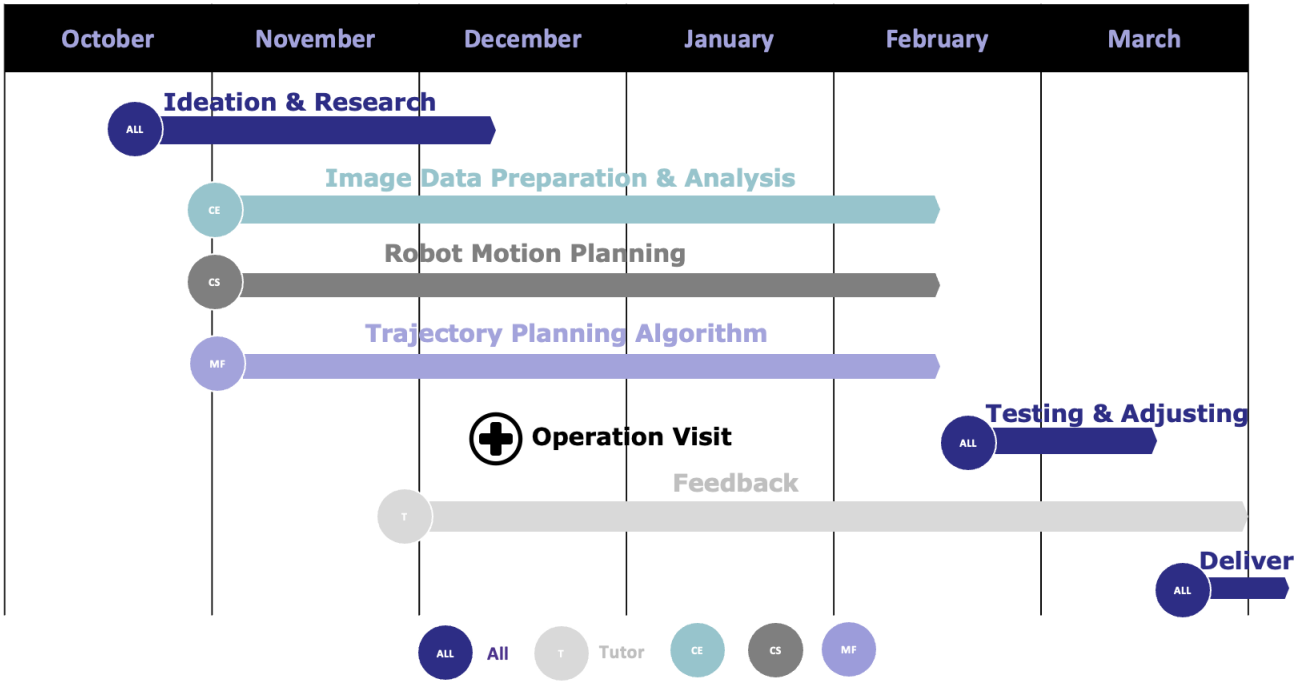
[4] A. Karamalis, W. Wein, T. Klein, and N. Navab. Ultrasound confidence maps using random walks. *Medical Image Analysis*, 16(6):1101–1112, Aug. 2012. 96 citations (Crossref) [2023-01-31]. doi: 10.1016/j.media.2012.07.005

[5] F. C. Park and B. J. Martin. Robot sensor calibration: solving ax=xb on the euclidean group. *IEEE Trans. Robotics Autom.*, 10:717–721, 1994.

[6] M. N. Rizve, K. Duarte, Y. S. Rawat, and M. Shah. In Defense of Pseudo-Labeling: An Uncertainty-Aware Pseudo-label Selection Framework for Semi-Supervised Learning, Apr. 2021. arXiv:2101.06329 [cs]. doi: 10.48550/arXiv.2101.06329

[7] R. Y. Tsai and R. K. Lenz. A new technique for fully autonomous and efficient 3d robotics hand/eye calibration. *IEEE Trans. Robotics Autom.*, 5:345–358, 1988.

## A Project Timeline




## B 3D Compounding Results

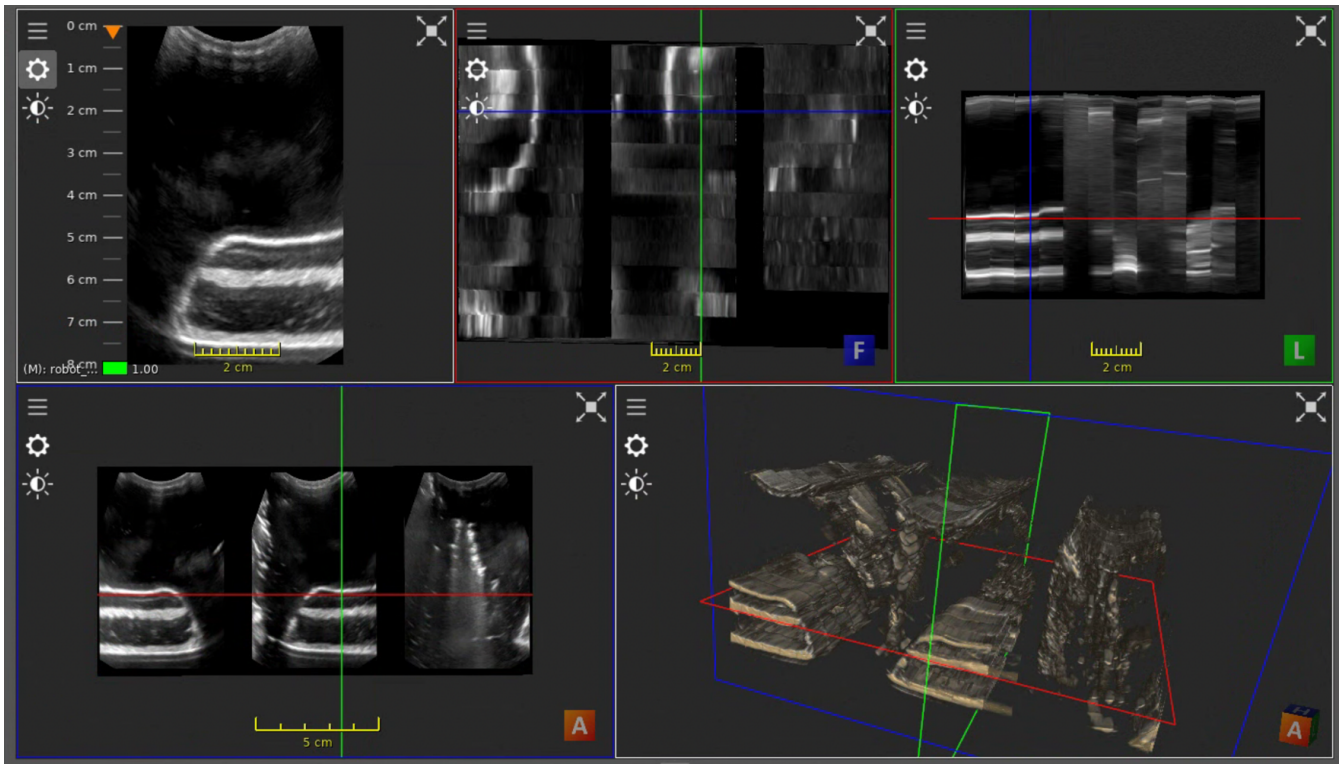



**Figure 19** 3D compounding results of the robotic scan.

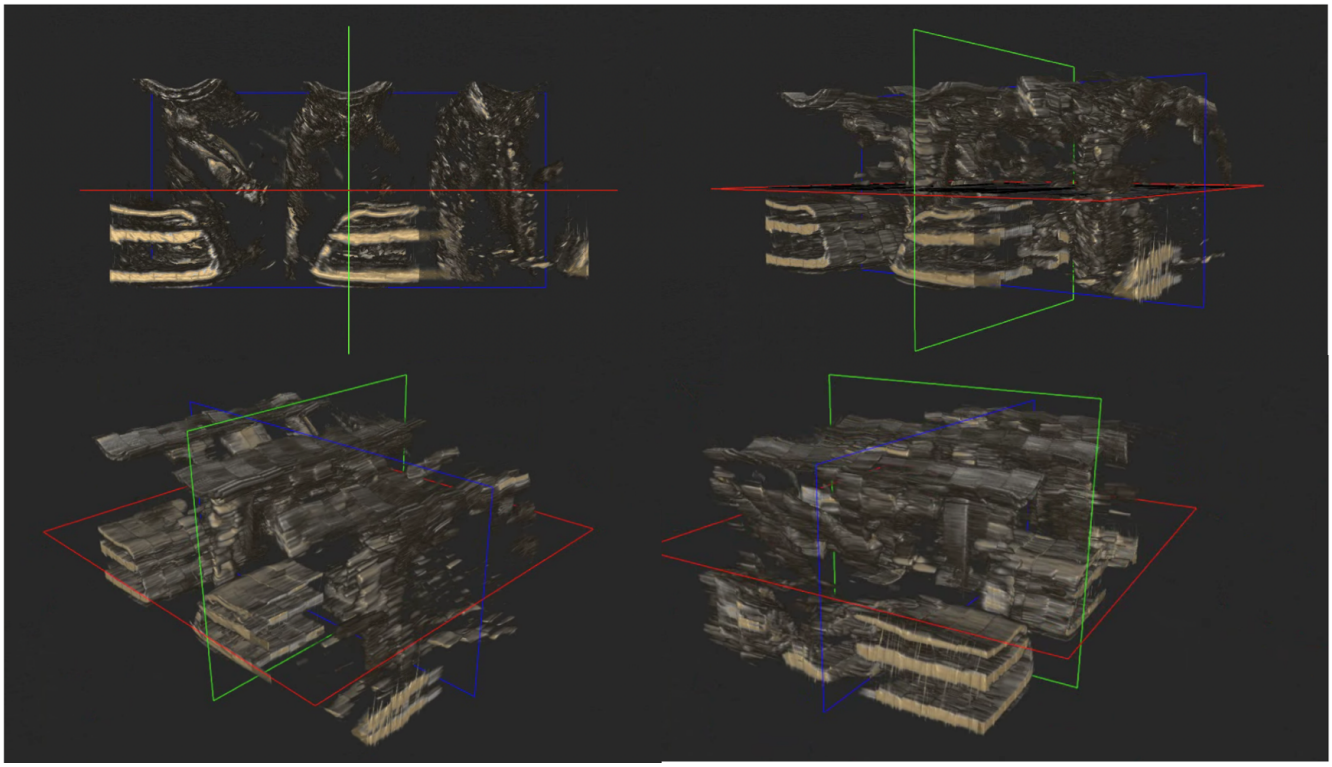


**Figure 20** 3D compounding from different viewing angels.