RAM ROBOTICS AND MECHATRONICS

UTILIZING TRANSFORMATION CHAIN SYSTEMS FOR ACCURATE POSITIONING AND ORIENTATION OF A 6-DOF ROBOTIC MANIPULATOR

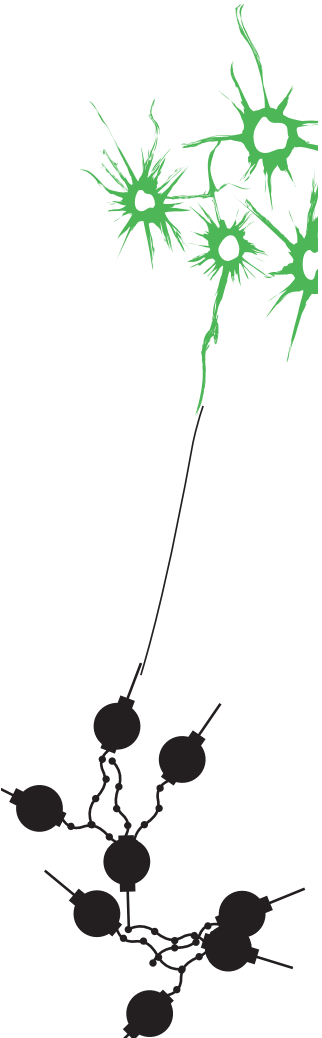
A.S.A.E. (Ahmed) Abdelsayed

BSC ASSIGNMENT

Committee:
dr. ir. M. Abayazid
A. Cordon, MSc
dr. I.S.M. Khalil

July, 2023

039RaM2023
Robotics and Mechatronics
EEMCS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands



UNIVERSITY OF TWENTE. | TECHMED CENTRE UNIVERSITY OF TWENTE. | DIGITAL SOCIETY INSTITUTE

Enhancing Accuracy in Liver Tumor Ablations: Utilizing Transformation Chain Systems for Accurate Positioning and Orientation of a 6-DOF Robotic Manipulator

Ahmed Abdelsayed

***Abstract*—This study introduces an innovative methodology that explicitly exploits chain transformations for registration in a robotic workspace for liver tumor ablation. The research focuses on how chain transformations can be used to accurately calculate the position and orientation of the end-effector concerning the base of a six-degrees-of-freedom (6-DOF) robotic manipulator. The goal is to ensure precise needle alignment concerning tumors for efficient ablation. The study illustrates the feasibility of employing chain transformations and inverse elements by characterizing multiple transformation chains of all the involved systems within the workspace. The study also shows how matrix multiplication and inverse properties are used to calculate undefined variables in the workspace. The obtained transformation matrices are then applied to the robotic manipulator base that achieves optimal needle alignment with the tumor. The findings from this study emphasize the possible application of chain transformations for precision registration in robotic tumor ablation. This method is currently experimental but could serve as a base for robotic tumor ablation. Thus, this research can inform future advancements in precision medicine and robotic interventions, improving treatment outcomes.**

***Keywords*—** Registration, Robotic Manipulator, Liver Cancer, Transformation Chains, Electromagnetic Generator

I. Introduction

Liver cancer is a prevalent and lethal disease worldwide, and unfortunately, it is projected to become even more prevalent in the coming years [1, 2]. The motion of the liver caused by respiration adds a layer of complexity to the treatment of this disease; variations in the amplitude of movement more significant than 5 mm during tumor, treatment could expose healthy liver tissues to a significant risk of radiation-induced damage [3]. Current standard treatment options range from surgery and chemotherapy to radiation therapy [4]. However, the emerging field of robotic manipulators presents an innovative and potentially superior alternative for liver tumor ablation. Compared to traditional surgery, robotic-assisted ablation provides enhanced precision, increased flexibility, superior control, and a less invasive approach, all while minimizing the potential for human error [5].

As reported by Goh *et al.* in [6], robotic surgery represents a progression in minimally invasive procedures across various disciplines. The research focused on the potential of robotics for telepresence surgery, and the US Army funded remote robotic operation systems for wounded soldiers. It was concluded that robotic surgery was a great alternative as it provides visualization and ergonomics advantages, all while being minimally invasive. It is expected that in the future robotic surgery will be adapted globally, which will, in return, enhance accuracy and cost-effectiveness in the future. This paper represents a foundation for possible advancement in medical treatment, specifically for liver tumor ablation, through integrating robotics, transformation matrices, and real-time data gathering. The paper proposes an approach to registering a 6-DOF robotic manipulator using transformation chain properties. The guiding research question for this project is:

"How can transformation chains optimize calculating a 6-DOF robotic manipulator's end-effector position for precise liver tumor ablation?"

The Franka Emika Panda serves as the robotic manipulator for this study. The system is modeled to tackle the research question, and the necessary transformation matrices in the workspace are identified to comprise several transformation chains. Following this introduction in section I, section II outlines the experimental procedure and delves into the analysis and modeling of the workspace. Section III provides an insightful exploration of the results and discussion of results. Section IV presents the conclusion, encapsulating and synthesizing the critical points addressed in the paper and possible future work.

II. Methodology

A. Experimental Setup

The experimental setup involves the Franka Emika robotic arm, an Aurora Electromagnetic (EM) generator (Northern Digital Inc., Waterloo, Canada), a needle, a tumor, and a liver phantom. A 2D sketch, depicted in Fig.1, facilitates a comprehensive understanding of the experimental setup. This illustrates the basic structure and arrangement of the system. Furthermore, a tangible representation is provided by an accompanying photograph of the setup, shown in Fig.2. These images combined offer a detailed overview of the experimental configuration.

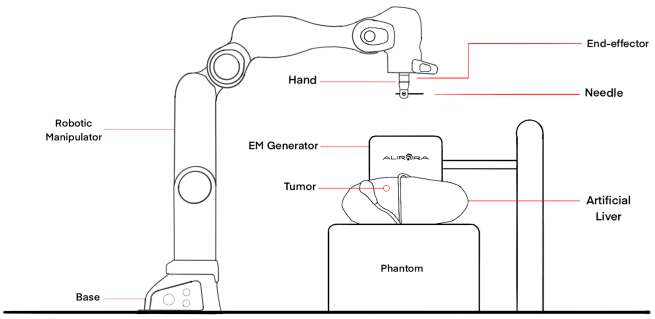


Fig. 1. 2D Sketch of the Experimental Setup

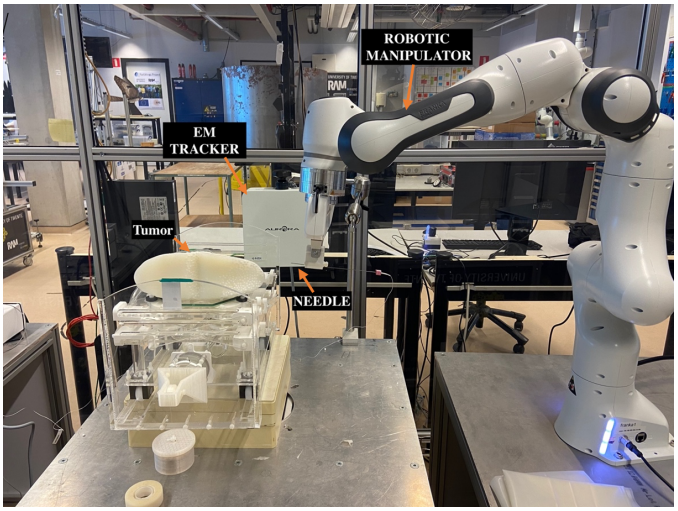


Fig. 2. Photograph of the Actual Experimental Setup

1) ***Robotic Arm (Franka Emika):*** The Franka Emika robot is the main foundation of the setup. It is used for precise manipulation and movement. It is programmed to accurately reach the end-effector (the part of the robot that interacts with the environment) to the designated target.

2) ***End-Effector:*** The end-effector is attached to the robotic arm, and the arm's movements control its position and orientation. Its role is to carry and precisely position the needle toward the target. The hand is an essential component of the workspace; its primary function is to be securely fixated to the end-effector from one side and secure a gripper on the other. In turn, the gripper holds the needle that performs liver tumor ablations.

3) ***Base:*** The base of the Franka Emika robotic arm functions as the reference point for all movements. Translating the tumor's location from the Aurora EM Generator's coordinate system to the robot's base frame is crucial in the experimental setup. This transformation process allows the robot to position its end-effector precisely for optimal needle and tumor alignment. Therefore, the robot base enables exact relative movements derived from the data provided by the EM Generator.

4) ***Electromagnetic Generator (Aurora):*** The electromagnetic (EM) Generator, or the Aurora, plays a vital role in the registration procedure. The EM Generator (Aurora) tracks the position and orientation of the needle and the tumor. The device generates EM waves detected by the trackers embedded within the needle and the tumor. The trackers respond by providing real-time data on their respective positions and orientations. This information is crucial in guiding accurate needle placement and verifying the tumor's position.

5) ***Needle:*** The needle is fixated securely to the hand, which is fixated to the end-effector. The end-effector's position and orientation determine the needle's position in spatial space. The needle performs the precise operation (e.g., liver ablation) on the tumor.

6) ***Tumor (Target):*** The tumor is the target that the needle needs to reach. The aim is to calculate the transformation matrix from base to end-effector, ensuring perfect alignment of the tumor with the needle. The positioning of the needle must be the same as the tumor but only requires a minimal translation along the y-axis (0.01 m) for alignment with the tumor; this way, the needle is ready for the tumor ablation procedure. With this setup, the experiment can provide a way to perform medical procedures involving precise needle placement with high accuracy and safety.

B. Algorithm for Precise Needle Alignment

An algorithm of the experimental details is displayed in Alg.1. A custom Python script that implements the algorithm was created to get the needed transformation matrices, as shown in section V-A. The notation with the general format ${}_{\mathrm{A}}T^{\mathrm{B}}$, that is used in Alg.1, represents the transformation matrix from frame B with respect to frame A. Section II-C, provides a more comprehensive explanation of the notation and its respective elements.

Algorithm 1 Needle Alignment Algorithm

1: Initialise all transformation matrices to None.
2: Model the ${}_{\text{EE}}T^{\text{needle}}$.
3: Obtain the current ${}_{\text{base}}T^{\text{EE}}$ of the Franka.
4: Compute ${}_{\text{Base}}T^{\text{EM}}$ for different positions within the workspace.
5: Compute the average for the samples obtained for ${}_{\text{Base}}T^{\text{EM}}$.
6: **if** ${}_{\text{EE}}T^{\text{needle}}$, ${}_{\text{Base}}T^{\text{EM}}$, and ${}_{\text{EM}}T^{\text{needle}}$ are not None **then**
7: Compute the value for ${}_{\text{base}}T^{\text{EE(opt)}}$
8: **end if**

C. Mathematical Background

*1) **Transformation Matrices**:* According to Craig's book [7], a transformation is the change of position and orientation of a frame attached to a body concerning another frame attached to another body. This paper focuses on spatial (3D) transformation only. A homogeneous transformation matrix combines a translation (position) and rotation (orientation) into one matrix, it is used to describe one coordinate system relative to another. To represent frame B concerning frame A, transformation matrices in spatial space have the following 4x4 general format.

$$ {}_{A}T^{B} = \left[\begin{array}{c|c} {}_{A}R^{B} & \vec{\mathbf{t}} \\ \hline 0\ 0\ 0 & 1 \end{array}\right] \tag{1} $$

*2) **Rotation Matrix**:* According to Lynch *et al.* in their book [8], the notation ${}_{A}R^{B}$ denotes a 3x3 rotation matrix that transforms a point from frame A to frame B. It is an orthogonal matrix, meaning its rows and columns are unit vectors and orthogonal to each other. When a point is multiplied by a rotation matrix, it changes its orientation without changing its position, as rotation is with respect to the frame's origin.

In 3D space, rotations can be performed about the coordinate frame axes. The rotation matrices for rotations about the $\hat{x}$, $\hat{y}$, and $\hat{z}$ axes are given by:

*a) **Rotation about the $\hat{x}$-axis**:* The rotation matrix $R_x(\theta)$ represents a rotation about the $\hat{x}$-axis by an angle θ.

$$ R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{2} $$

*b) **Rotation about the $\hat{y}$-axis**:* The rotation matrix $R_y(\theta)$ represents a rotation about the $\hat{y}$-axis by an angle θ.

$$ R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{3} $$

*c) **Rotation about the $\hat{z}$-axis**:* The rotation matrix $R_z(\theta)$ represents a rotation about the $\hat{z}$-axis by an angle θ.

$$ R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4} $$

To obtain the final rotation matrix, multiply all the previous matrices by each other. Rotation matrices play a crucial role in the calculations used in this study to determine the end-effector's position and orientation relative to the base of the 6-DOF robotic manipulator.

*3) **Translation Vector**:* The notation $\vec{\mathbf{t}}$ denotes a translation vector that moves a point from frame A to frame B. A *three-dimensional translation vector* is a 3x1 vector, which shifts points in the 3D space along the three axes (X, Y, Z) [9]. The translation does not alter the orientation of the point; it only changes its position. When adding a translation vector to a point, it moves the point along the X, Y, and Z axes by the amounts specified in the vector. So, for a translation vector defined as T = [a, b, c], the point P = [x, y, z] will be translated to P' = [x+a, y+b, z+c].

*4) **Using Rotation and Translation Together**:* Often in 3D transformations, both rotation and translation operations are combined. The non-commutative nature of matrix multiplication requires careful ordering of transformations, as different sequences yield different outcomes, like translating before rotating versus rotating before translating. This arises because a transformation matrix does not operate on the object itself but rather on the coordinate system in which the object is situated. To better illustrate the difference in an object's final position and orientation depending on the order of operations, a graphical representation is provided in Fig.3.

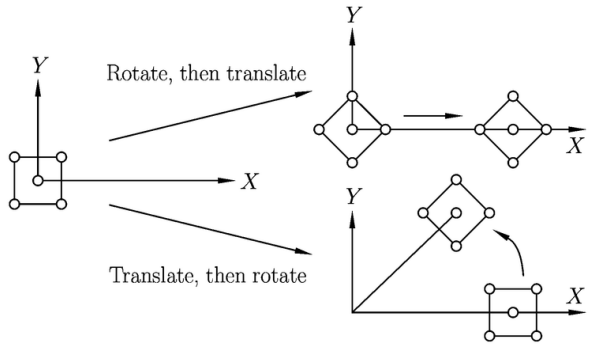


Fig. 3. Outcome from two different orders in transformation [10]

The general form of transformation from frame A to frame B, given a rotation matrix ${}_{A}R^{B}$, and a translation vector $\tilde{\mathbf{t}}$, can be represented as [7]:

$$ \mathbf{p}_B = {}_A R^B \cdot \mathbf{p}_A + {}_{\mathbf{A}}\tilde{\mathbf{t}}^{\mathbf{B}} \tag{5} $$

Here, ${}_AR^B \cdot \mathbf{p}_A$ rotates the point in frame A, and then ${}_A\vec{t}^B$ is added to translate it to frame B. Homogeneous coordinates are used, which allows expressing the two transformations within a single 4x4 matrix; This is done by adding an extra row and column to the rotation matrix and translation vector, forming an augmented matrix as shown in Eq.1 [7].

*5) **Transformation Chains**:* In the registration of the robot, a methodology involving different transformation chains within the workspace and their respective inverse operations was applied when needed. Transformation chains involve a series of transformations that map the state from one reference frame to another. This approach offered an efficient way to relate necessary frames to each other. In the project context, transformation chains are exploited to register a robotic system, specifically a Franka Emika robot with a needle attached to the end-effector, guided by an Aurora Electromagnetic Generator. The first transformation chain formed in the workspace consists of the following transformations:

1) From the robot's base to its end-effector, (${}_{base}\boldsymbol{T}^{ee}$)
2) From the end-effector to the needle, (${}_{ee}\boldsymbol{T}^{needle}$)
3) From the needle to the EM Generator, (${}_{needle}\boldsymbol{T}^{EM}$)

D. Transformation Matrices in Robotic Configuration Space

This section focuses on extracting and interpreting transformation matrices in the context of a robot's configuration space, leveraging data from Robot Operating System (ROS) measurements [11], EM Generator data, and additional independent models. Transformation matrices are pivotal to determining and manipulating the robot's position and orientation within its workspace. ROS provides a variety of measurements linked to the robot's pose, while the electromagnetic (EM) Generator extracts essential tracking data of the needle and tumor. Independent models, such as `EE_TF_needle`, enrich the robotic workspace's understanding. Collectively, these data offer a comprehensive insight into the robot's workspace, a crucial element for the registration process.

*1) **Transformation from base to end-effector**:* The project utilizes ROS along with the Franka Interface to obtain the pose of the robot's end-effector relative to its base frame. The pose includes a Point and a Quaternion representing the position and orientation of the end-effector, respectively. The real-time pose information is obtained by subscribing to the appropriate ROS topic and publishing these messages, providing critical data for the system's registration.

*2) **Transformation from end-effector to needle**:* The needle is securely affixed to the gripper in the system setup, which is further rigidly attached to the robot hand as shown in Fig.1. This sequence is interpreted as a rigid body transformation from the hand to the needle (`hand_TF_needle`), implying that any rotation imparted to the hand directly translates to the needle due to the rigid connections. However, the transformation from the end-effector to the hand (`ee_TF_hand`) does not adhere strictly to a rigid transformation. This transformation involves degrees of rotation, as informed by the Franka manual [12]. The total transformation matrix from the end-effector to the needle is the product of `ee_TF_hand` and `hand_TF_needle`. The transformation matrix for `ee_TF_hand` is denoted as

$$\mathbf{EE\_TF\_hand} = \begin{bmatrix} 0.707 & 0.707 & 0 & 0 \\ -0.707 & 0.707 & 0 & 0 \\ 0 & 0 & 1 & 0.1034 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

On the other hand, `hand_TF_needle` is unknown and should be modeled. In order to do so, the rotation matrix is first computed by relating the frame of the needle concerning the frame of the hand, as shown in Fig.4

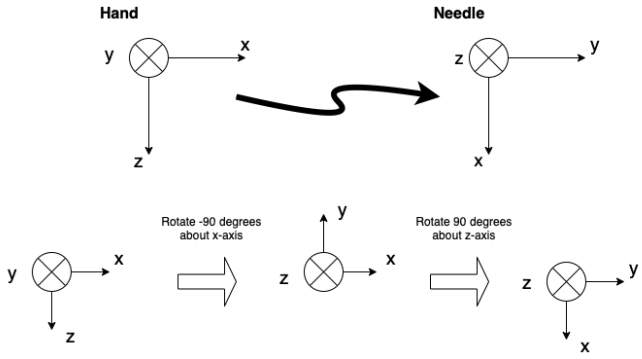


Fig. 4. Relating frame of the needle with respect to the hand

The final rotation matrix from the hand to the needle could be computed using the rotation matrices covered in section II-C2. The translation vector represents the relative positional displacement from the hand's origin to the needle's origin along the x, y, and z axes. Combing both the rotation matrix and translation vector, the following transformation matrix could be shown:

$$\mathbf{hand\_TF\_needle} = \begin{bmatrix} 0 & -1 & 0 & -0.135 \\ 0 & 0 & 1 & 0.0295 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

An additional transformation matrix (`ee_TF_needle`) is computed to elucidate the entire relationship from the end-effector to the needle. This matrix is the product of the end-effector-to-hand transformation matrix (`ee_TF_hand`), obtained from the Franka manual, and the hand-to-needle transformation matrix (`hand_TF_needle`), which is modeled based on the physical parameters of the setup

under the assumption of rigid transformation. This approach simplifies kinematics, providing a practical method for describing the spatial relationship between the end-effector, the hand, and the needle.

*3) **Transformation from needle or tumor to EM Generator**:* The needle and tumor have EM Generators attached to them, which could be identified by the Aurora system, which helps compute their respective transformation matrices concerning the EM Generator. These transformation matrices are obtained using data from the Aurora system and processed using the scipy.spatial.transform function from the scipy library in Python. The Aurora system provides the position and orientation of the EM Generator concerning the needle/tumor in real time at a sampling rate of 40 Hz. The orientation information is given in the form of a quaternion. In Python, the scipy.spatial.transform function converts the quaternion from the Aurora system into a rotation matrix. Once the rotation matrix is obtained, it is combined with the translation vector provided by the Aurora system to form the complete transformation matrix. This matrix captures the spatial relationship between the needle/tumor and the EM Generator, a crucial element of this study.

E. Calculating Precise Needle Alignment

The relationship between the robot's base and the EM generator (`base_TF_EM`) is thoroughly established by calculating it at ten arbitrary positions when the needle is securely attached to the robotic manipulator and within the range of the EM generator. From these multiple measurements, the average is computed and subsequently used as a constant for further calculations. The static nature of the base and the EM generator justifies using a constant transformation matrix. In addition, the standard deviation of `base_TF_EM` is calculated to measure the reliability and consistency of the measurements. Following this, an optimal transformation from the robot's base to the end-effector, `base_TF_EE(opt)`, is calculated by employing a transformation chain that begins with the averaged `base_TF_EM (averaged)`. The equation for the transformation chain is shown in Eq.8.

$$ {}_{base}T^{EE(opt)} = {}_{base}T^{EM(Averaged)} \cdot {}_{EM}T^{Needle} \cdot {}_{Needle}T^{EE} \tag{8} $$

The following link in the chain is `EM_TF_Needle`, the transformation from the EM Generator to the needle. For this transformation, the rotation matrix is modeled perpendicular to the robot's hand and pointing toward the tumor. The translation vector is identical to that obtained from `EM_TF_tumor` but with an adjustment of -0.01 in the y-axis. This adjustment serves as a safety margin, ensuring the needle is precisely targeted to the correct location with an appropriate margin before ablation. The final link in the chain is `Needle_TF_EE`, the inverse of the transformation from the end-effector to the needle, which has been previously modeled in section II-D2. In summary, this method emphasizes the calculation of a more robust `base_TF_EM` through averaging measurements from multiple positions. It uses this transformation to compute an optimal `base_TF_EE` via a transformation chain. This chain incorporates safety measures and pre-modeled transformations to ensure accurate and safe targeting of the liver tumor.

III. Results and Discussion

A. Results

In the study, a series of transformation matrices were computed for the robotic system. These matrices represent the spatial relationships between different parts of the robotic system, including the base, the electromagnetic (EM) generator, the end-effector (EE), and the needle. Initially, the transformation matrix from the robot's base to the EM generator was computed at multiple points, `base_TF_EM`. The average transformation matrix and the standard deviation were calculated for five measurements as shown in Eq.9.

$$ {}_{base}T^{EM_{(\text{Avg5})}} = \begin{bmatrix} 0.2560 & 0.6768 & 0.6852 & 0.7895 \\ 0.2989 & -0.7296 & 0.6117 & 0.0382 \\ 0.9162 & 0.0490 & -0.3911 & 0.0512 \\ 0 & 0 & 0 & 1 \end{bmatrix}, $$

$$ {}_{base}T^{EM_{(\text{Std5})}} = \begin{bmatrix} 0.0650 & 0.0369 & 0.0328 & 0.0230 \\ 0.0367 & 0.0351 & 0.0364 & 0.0612 \\ 0.0112 & 0.0665 & 0.0268 & 0.0337 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{9} $$

The average provides the mean spatial relationship between the robot's base and the EM generator, and the standard deviation quantifies the variability in these measurements. Later, this procedure was repeated with ten measurements, which gave a slightly different average and standard deviation as shown in Eq.10. The average and standard deviation from the ten measurements is more reliable than for five measurements due to the increased number of samples.

$$ {}_{base}T^{EM_{(\text{Avg10})}} = \begin{bmatrix} 0.2468 & 0.6789 & 0.6872 & 0.7941 \\ 0.2880 & -0.7252 & 0.6225 & 0.0493 \\ 0.9222 & 0.0642 & -0.3766 & 0.0480 \\ 0 & 0 & 0 & 1 \end{bmatrix}, $$

$$ {}_{base}T^{EM_{(\text{Std10})}} = \begin{bmatrix} 0.0620 & 0.0361 & 0.0259 & 0.0208 \\ 0.0345 & 0.0348 & 0.0343 & 0.0587 \\ 0.0118 & 0.0516 & 0.0301 & 0.0347 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{10} $$

Using the averaged `base_TF_EM`, an optimal transformation matrix, `base_TF_EE(opt)`, was also calculated. This optimal transformation matrix provides the theoretical best positioning of the needle concerning the robot's base. A comparison of the manually calibrated `base_TF_EE` and the computed optimal `base_TF_EE(opt)` reveals differences between the current configuration of the robotic arm and its optimal configuration as shown in Eq.11. Differences could inform improvements to the modeling of the workspace.

$$
{}_{base}T^{EE} = \begin{bmatrix} 0.999 & 0.0348 & 0.0264 & 0.6169 \\ 0.0350 & -0.9993 & -0.00706 & -0.0247 \\ 0.0261 & 0.0080 & -0.9996 & 0.2946 \\ 0 & 0 & 0 & 1 \end{bmatrix},
$$

$$
{}_{base}T^{EE_{(\text{opt})}} = \begin{bmatrix} 0.9946 & 0.0007 & 0.0713 & 0.6376 \\ 0.0064 & -0.9982 & 0.0116 & 0.0209 \\ 0.0839 & 0.0045 & -0.9946 & 0.3407 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11}
$$

The following video , shows the Franka's movement when the calculated ${}_{base}T^{EE_{(\text{opt})}}$ was published to the base of the robotic manipulator.

B. Discussion of Results

The robotic system's performance analysis revealed significant discrepancies in the spatial relationships between various components. Notably, the average transformation matrix `base_TF_EM`, calculated from five repeated measurements, demonstrated a standard deviation rounded to the nearest two decimal places in the translational vector. This discrepancy corresponds to approximately 1 cm when translated to physical space, considering the units of measurement used in this study. Given the precision and delicacy required for the task, such a level of variance is deemed inadequate. This deviation can be primarily attributed to the method of deriving the average transformation matrix. The straightforward approach of calculating the arithmetic mean of measurements, while being intuitive, is not robust against outliers or skewed distributions. These can lead to an inflated standard deviation and a higher discrepancy in the translational vector. Ideally, the number of samples should be increased until the standard deviation values are in the thousandths, representing millimeters in the workspace. It should be noted that increasing the sample space indefinitely does not guarantee a decrease in the standard deviation. A lower standard deviation does not necessarily imply a more accurate result but indicates greater consistency between the results. The `base_TF_EM` should be validated to ensure it does not cause disparities for the calculated `base_TF_EE(opt)`. A possible validation method for the average of `base_TF_EM` is exploiting closed-loop properties in the robotic workspace. The premise is straightforward, when the following chain `base_TF_EM(avg)`, `EM_TF_needle`, `needle_TF_EE`, and finally `EE_TF_base` is. Multiplied in their respective order, they should produce `base_TF_base`, which should be an identity matrix if `base_TF_EM(avg)` is computed accurately. Any deviation from the identity matrix indicates an inconsistency in the transformations, suggesting a need for fine-tuning. The `base_TF_EM` matrix can be adjusted until the product of the transformation matrices is as close as possible to the identity matrix, effectively closing the loop.

It should be noted that perfectly closing the loop may be challenging due to real-world imperfections and measurement noise. As a result, a minor deviation from the identity matrix is considered acceptable up to 3 decimal places, which is translated to millimeters in the workspace. However, it is essential to note that while this method effectively validates the consistency among the transformation matrices, it does not validate them against an external ground truth. An external validation method, such as optical tracking systems, could be a viable option to be explored for future work to validate the results.

As for the comparison between the manually calibrated `base_TF_EE` and the computed optimal `base_TF_EE(opt)`, it is reasonable to assume that predicting the exact rotation with manual calibration would be challenging. Hence the validation is primarily focused on the translational vector. It is observed that the vectors also display a discrepancy of around two decimal places, which aligns with the standard deviation obtained from the earlier transformation matrices. This could be seen as well in the video as the "needle" is not perfectly aligned with the tumor but significantly close to precise alignment, which emphasizes the importance of using robust methods with more samples to minimize these errors and increase the precision of the robotic system.

IV. Conclusion

In the face of a global health challenge such as liver cancer, an increasingly prevalent and lethal disease, this project introduced the possible registration of a robotic workspace in an innovative and potentially superior alternative for liver tumor ablation by utilizing robotic manipulators. The study provides critical insights into how transformation chains can optimize the calculation of a 6-DOF robotic manipulator's (Franka Emika Panda) end-effector position for precise tumor ablation. This is utilized as a platform for this study. The research demonstrated how transformation chains can be exploited for registration in a robotic workspace to ensure precision in needle placement. This was done by extensively analyzing transformation matrices representing spatial (3D) relationships between different parts in the robotic workspace. Utilizing the transformation chain, the first transformation matrix `base_TF_EM` is computed at ten different positions within the workspace. The average of these matrices aimed to represent a more accurate spatial relationship in the workspace; this method faced certain limitations despite its intuitiveness. One of the main challenges was the variation between the computed average and the actual set of transformations at different points. This variation, quantified as the standard deviation in the translational vector, reached up to roughly 1 cm. Given the high precision expected in medical procedures like tumor ablation, this level of deviation raised concerns. This is an area where future research should yield substantial improvements by maybe having another method to compute a better and more accurate average for `base_TF_EM`. Furthermore, by comparing `base_TF_EE` after manual calibration with calculated `base_TF_EE(opt)`, which seemed to have a very similar standard deviation as well for the translation vector. While the

research revealed some challenges, it ultimately demonstrated the tremendous potential of robotic manipulators for liver tumor ablation. Integrating robotics, real-time data gathering and transformation chain systems could herald a new era of precision and consistency in tumor ablation procedures. The insights derived from this research could prove invaluable in further improving robotic manipulators' precision, flexibility, and control, thereby enhancing the efficacy and safety of tumor ablation procedures. Further studies are required to continue refining the methods used to compute transformation matrices and their associated averages to improve the accuracy and precision of robotic manipulator positioning in clinical applications.

REFERENCES

[1] H. Rumgay, M. Arnold, J. Ferlay, O. Lesi, C. J. Cabasag, J. Vignat, M. Laversanne, K. A. McGlynn, and I. Soerjomataram, "Global burden of primary liver cancer in 2020 and predictions to 2040," *Journal of Hepatology*, vol. 77, no. 6, pp. 1598–1606, Dec. 2022. [Online]. Available: https://doi.org/10.1016/j.jhep.2022.08.021

[2] Rising incidence of cancer. [Online]. Available: https://tribune.com.pk/letter/1865468/rising-incidence-cancer

[3] Y.-L. Tsai, C.-J. Wu, S. Shaw, P.-C. Yu, H.-H. Nien, and L. T. Lui, "Quantitative analysis of respiration-induced motion of each liver segment with helical computed tomography and 4-dimensional computed tomography," *Radiation Oncology*, vol. 13, no. 1, Apr. 2018. [Online]. Available: https://doi.org/10.1186/s13014-018-1007-0

[4] Types of cancer treatment. [Online]. Available: https://www.cancer.org/treatment/treatments-and-side-effects/treatment-types.html

[5] Robotic surgery. [Online]. Available: https://www.mayoclinic.org/tests-procedures/robotic-surgery/about/pac-20394974

[6] E. Z. Goh and T. Ali, "Robotic surgery: an evolution in practice," *Journal of Surgical Protocols and Research Methodologies*, vol. 2022, no. 1, Jan. 2022. [Online]. Available: https://doi.org/10.1093/jsprm/snac003

[7] J. Craig, *Introduction to Robotics: Mechanics and Control*, ser. Addison-Wesley series in electrical and computer engineering: control engineering. Pearson/Prentice Hall, 2005. [Online]. Available: https://books.google.nl/books?id=MqMeAQAAIAAJ

[8] K. Lynch and F. Park, *Modern Robotics: Mechanics, Planning, and Control*. Cambridge Univeristy Press, 2017.

[9] Spatial transformation matrices. [Online]. Available: https://www.brainvoyager.com/bv/doc/UsersGuide/CoordsAndTransforms/SpatialTransformationMatrices.html

[10] X. Wang and Q. Sun, "Mechanical modeling of the bistable bacterial flagellar filament," *Acta Mechanica Solida Sinica*, vol. 24, p. 1, 12 2011.

[11] Stanford Artificial Intelligence Laboratory et al., "Robotic operating system." [Online]. Available: https://www.ros.org

[12] Franka Emika, *FRANKA HAND Product Manual*, 80797 Munich Germany, 2022, https://download.franka.de/documents/220010_Product%20Manual_Franka%20Hand_1.2_EN.pdf.

V. Supporting Material

A. Python Script

```
#!/usr/bin/python3
# -- coding: utf-8 --

import rospy
import numpy as np
import scipy.spatial.transform
from geometry_msgs.msg import Pose, PoseStamped
from ros_igtl_bridge.msg import igtltransform
from franka_msgs.msg import FrankaState


class TransformationOptimizer:

    def _init_(self):
        rospy.init_node('transformation_optimizer')

        self.base_TF_em = None
        self.em_tf_needle_opt = None
        self.tumor_tf_em = None
        self.base_TF_em_std = None
        self.em_tf_tumor = None
        self.em_tf_needle = None
        self.base_TF_tumor = None
        self.ee_tf_hand = np.array([[0.707, 0.707, 0, 0], [-0.707,
                                    0.707, 0, 0], [0, 0, 1, 0.1034], [0,
                                    0, 0, 1]])
        self.hand_tf_needle = np.array([[0, -1, 0, -0.135], [0, 0, 1,
                0.0295], [-1, 0, 0, 0], [0, 0, 0, 1]])
        self.base_TF_ee = None
        self.base_TF_em_measurements = [
            np.array([[0.37301411, 0.65801928, 0.65387306, 0.7791658],
                      [0.23482295, -0.7487801, 0.61956877, 0.07981227],
                      [0.89758415, -0.07757888, -0.43396305,
                      0.00518459], [0., 0., 0., 1.]]),
            np.array([[0.26063612, 0.62639499, 0.7344234, 0.76253417],
                      [0.31810413, -0.77396784, 0.54723167,
                      -0.06884315], [0.91149339, 0.09100739,
                      -0.40111895, 0.08316522], [0., 0., 0., 1.]]),
            np.array([[0.25229139, 0.66675102, 0.70105017, 0.82453963],
                      [0.28291204, -0.74366927, 0.60547139, 0.10756104],
                      [0.92534303, 0.04558383, -0.37637998, 0.0722625],
                      [0., 0., 0., 1.]]),
            np.array([[0.21144681, 0.73475253, 0.64428948, 0.77355042],
                      [0.33815713, -0.67348805, 0.65707132, 0.01751539],
                      [0.91699868, 0.07893902, -0.39099934, 0.07988697],
                      [0., 0., 0., 1.]]),
            np.array([[0.18245945, 0.69785847, 0.6923748, 0.80756695],
                      [0.32071311, -0.70792696, 0.62901629, 0.05508904],
                      [0.92940968, 0.10729438, -0.35310528, 0.01574833],
                      [0., 0., 0., 1.]]),
            np.array([[0.14985626, 0.7356549, 0.66032958, 0.82521958],
                      [0.29493837, -0.6707282, 0.68030459, 0.15135368],
                      [0.94367304, 0.09281368, -0.31759226,
                      -0.02009197], [0., 0., 0., 1.]]),
            np.array([[0.20643994, 0.69897683, 0.68478427, 0.79815017],
                      [0.31741166, -0.69941398, 0.63991378, 0.07387668],
                      [0.92616951, 0.11429072, -0.35978099, 0.03810567],
                      [0., 0., 0., 1.]]),
            np.array([[0.23834924, 0.67996228, 0.69376147, 0.80712529],
                      [0.27344812, -0.72151397, 0.63648194, 0.04001337],
```

```
                    [0.93198473, 0.06361045, -0.35702709, 0.06586284],
                    [0., 0., 0., 1.]]),
            np.array([[0.27817389, 0.65801882, 0.69976813, 0.78821943],
                    [0.26561212, -0.74471013, 0.61287531, 0.05107646],
                    [0.92048479, 0.06686362, -0.38548031, 0.05873695],
                    [0., 0., 0., 1.]]),
            np.array([[0.31580428, 0.63224551, 0.70720099, 0.77442998],
                    [0.23371516, -0.76769813, 0.59724927,
                    -0.01437382], [0.91885105, 0.05960038,
                    -0.39024623, 0.08152537], [0., 0., 0., 1.]]),
            ]

        self.base_TF_em = np.mean(self.base_TF_em_measurements, axis=0)
        print ('Base_TF_EM Average:', self.base_TF_em)
        self.base_TF_em_std = np.std(self.base_TF_em_measurements,
                axis=0)
        print ('Base_TF_EM Standard Deviation:', self.base_TF_em_std)

        rospy.Subscriber('/IGTL_TRANSFORM_IN', igtltransform,
                         self.aurora_callback)
        rospy.Subscriber('/franka_state_controller/franka_states',
                         FrankaState, self.get_base_TF_ee)

        self.output_pose_pub = rospy.Publisher('/equilibrium_pose',
                PoseStamped, queue_size=1)

    def get_base_TF_ee(self, data):
        A = np.array(data.O_T_EE).reshape((4, 4))
        self.base_TF_ee = np.transpose(A)

        rotation_matrix = self.base_TF_ee[:3, :3]
        quaternion = \
            scipy.spatial.transform.Rotation.from_matrix(rotation_matrix).as_quat()
        self.base_TF_ee_quaternion = quaternion

        print ('Base_TF_EE', self.base_TF_ee)

    def aurora_callback(self, data):
        quaternion = [data.transform.rotation.x,
                      data.transform.rotation.y,
                      data.transform.rotation.z,
                      data.transform.rotation.w]
        rotation = \
            scipy.spatial.transform.Rotation.from_quat(quaternion)
        rotation_matrix = rotation.as_matrix()
        translation_vector = np.array([data.transform.translation.x
                / 1000, data.transform.translation.y / 1000,
                data.transform.translation.z / 1000])

        if data.name == 'NeedleToTracker':
            self.em_tf_needle = np.eye(4)
            self.em_tf_needle[:3, :3] = rotation_matrix
            self.em_tf_needle[:3, 3] = translation_vector

            # print ('needle', self.em_tf_needle)

            self.compute_base_TF_EE_opt()
        elif data.name == 'TumorToTracker':
            self.em_tf_tumor = np.eye(4)
            self.em_tf_tumor[:3, :3] = rotation_matrix
            self.em_tf_tumor[:3, 3] = translation_vector

            # print ('tumor', self.em_tumor)
```

```
            self.compute_base_TF_EE_opt()

    def compute_base_TF_EE_opt(self):
        if self.base_TF_em is not None and self.em_tf_tumor is not None:
            self.tumor_TF_em = np.linalg.inv(self.em_tf_tumor)
            self.ee_tf_needle = np.matmul(self.ee_tf_hand,
                    self.hand_tf_needle)
            self.needle_tf_ee = np.linalg.inv(self.ee_tf_needle)
            self.base_TF_EE_opt = np.matmul(np.matmul(self.base_TF_em,
                    self.get_EM_TF_needle_opt()), self.needle_tf_ee)
            print ('Base_TF_EE_opt', self.base_TF_EE_opt)
            pose = Pose()
            pose.position.x = self.base_TF_EE_opt[0, 3] - 0.12
            pose.position.y = self.base_TF_EE_opt[1, 3] - 0.03
            pose.position.z = self.base_TF_EE_opt[2, 3] - 0.01
            pose.orientation.x = self.base_TF_ee_quaternion[0]
            pose.orientation.y = self.base_TF_ee_quaternion[1]
            pose.orientation.z = self.base_TF_ee_quaternion[2]
            pose.orientation.w = self.base_TF_ee_quaternion[3]
            pose_stamped = PoseStamped()
            pose_stamped.header.frame_id = '/panda_link0'
            pose_stamped.pose = pose
            self.output_pose_pub.publish(pose_stamped)

    def get_EM_TF_needle_opt(self):
        rotation_matrix = np.array([[0.9476546, -0.1842600, -0.2607661],
                                    [0.2463202, -0.0977731,  0.9642441],
                                    [-0.2031676, -0.9780024, -0.0472681]])
        translation_vector = np.array([self.em_tf_tumor[0, 3],
                self.em_tf_tumor[1, 3], self.em_tf_tumor[2, 3]])
        em_tf_needle_opt = np.eye(4)
        em_tf_needle_opt[:3, :3] = rotation_matrix
        em_tf_needle_opt[:3, 3] = translation_vector
        return em_tf_needle_opt

if _name_ == '_main_':
    TransformationOptimizer()
    rospy.spin()
```

Listing 1. Python Script called to the launch file

B. Code Development

1) ***Base-TF-EE:*** The `get_base_TF_ee` function processes data from a ROS topic to compute the transformation from the robot's base frame to its end-effector frame. It first reshapes the received 1D array representing the transformation matrix into a 4x4 matrix and transposes it to match Python's column-major format. Once the transformation matrix is computed, the function calls `compute_base_TF_em` to calculate the transformation matrix from the base frame to the Electromagnetic generator frame. This sequence ensures that the computations remain updated with the latest transformations.

2) ***EM-TF-Needle / Tumor:*** The `aurora_callback` function processes incoming transformation data from the Aurora tracking system. The function first extracts the rotation part of the transformation as a quaternion and converts it into a rotation matrix using Scipy's spatial transformations utilities. The translation part of the transformation is then extracted and scaled down by a factor of 1000 to convert it from millimeters to meters. Depending on the type of incoming data - 'NeedleToTracker' or 'TumorToTracker,' the function sets either the needle's or the tumor's transformation from the electromagnetic (EM) generator frame. If the base-to-EM generator frame transformation is already calculated (for the 'TumorToTracker' case), the base-to-tumor transformation is computed. This data processing and classification allow for targeted adjustments based on whether the incoming data pertains to a needle or a tumor.

3) ***Base-TF-Tumor:*** The `compute_base_TF_tumor` function calculates the transformation from the robot base frame to the tumor (`base_TF_tumor`). It checks whether transformations `base_TF_ee` (from the robot base frame to the end-effector frame) and `em_TF_needle` (from the EM generator frame to the needle frame) exist. If so, it computes

`base_TF_tumor` by chaining these transformations together.

4) ***Needle-TF-Tumor****:* The code snippet computes the product of two matrices, `self.em_tf_needle` and `self.em_tf_tumor`. It first calculates the inverse of `self.em_tf_needle` using the *NumPy* function *np.linalg.inv()*, and then performs matrix multiplication using *np.matmul()*. The resulting matrix, `self.needle_tf_tumor`, relates the tumor concerning the needle frame, which is a critical aspect for precise alignment of the needle and tumor.

5) ***Base-TF-EE(optimal)****:* The function `compute_base_TF_EE_opt` is crucial in achieving precise needle and tumor alignment. When the needle and tumor are aligned, the chain of transformations calculates the desired `base_TF_EE`. The matrix is then deconstructed into various elements using the *PoseStamped()* function, which encapsulates the desired coordinates. These coordinates are then published to the Franka as a command for the robot, enabling it to move to the measured coordinates that ensure precise alignment.