Clemson University

TigerPrints

All Dissertations

Dissertations

12-2023

Improving Hexapod Platform Pose Accuracy - A Photogrammetry-Based Approach

Sourabh Karmakar
sourabh.karmakar@gmail.com

Follow this and additional works at: https://tigerprints.clemson.edu/all_dissertations

Part of the Electro-Mechanical Systems Commons, and the Tribology Commons

Recommended Citation
Karmakar, Sourabh, "Improving Hexapod Platform Pose Accuracy - A Photogrammetry-Based Approach" (2023). *All Dissertations*. 3500.
https://tigerprints.clemson.edu/all_dissertations/3500

This Dissertation is brought to you for free and open access by the Dissertations at TigerPrints. It has been accepted for inclusion in All Dissertations by an authorized administrator of TigerPrints. For more information, please contact kokeefe@clemson.edu.

Improving Hexapod Platform Pose Accuracy- A Photogrammetry-Based Approach

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Mechanical Engineering

By
Sourabh Karmakar
December 2023

Accepted by:
Prof. Dr. Cameron Turner, Committee Chair
Prof. Dr. Garrett Pataky, Committee Member
Prof. Dr. Hongseok Choi, member
Prof. Dr. Laine Mears, member

ABSTRACT

The aim of this research is to make a newly constructed Stewart-Gough Platform-based test frame Tiger 66.1 operational by developing control software and estimating the error in its pose accuracy. The accuracy of the platform is affected by one source or multiple sources. The typical error sources are kinematic and structural, some of them originate from manufacturing imperfections, assembly deviations, elastic deformations, thermal deformations, and joint clearances which change the expected kinematic behavior of the manipulator. Also, some non-mechanical errors like transmission error, sensor accuracy, algorithm error, and truncation error in calculation contribute significantly in some cases. Using pose deviations as a foundation, this research further aims to develop a calibration method that enhances pose accuracy, leveraging the pose deviations observed during the initial measurements. This research presents a novel calibration method for a Stewart-Gough platform using photogrammetry, a digital image-based technique for 3D measurement and modeling. As part of this research, a new forward kinematic algorithm has been developed to implement an accurate non-contact calibration method that can improve the accuracy and precision of a general Stewart-Gough platform. The Stewart-Gough platform is one of the most popular Parallel Kinematic Machines (PKM). This mechanism, also known as a parallel manipulator or parallel robot, consists of a fixed base and a movable platform connected by six actuators or struts. The Stewart-Gough platform has remained an interesting machine for research due to its flexibility, structural rigidity, high accuracy, and reliability in motion control. The calibration of a Stewart-Gough platform is one of the essential steps in ensuring the accuracy and stability of the moving platform center. This dissertation investigates a forward kinematic calibration method for improving the accuracy of the Stewart-Gough platform-based test platform “Tiger 66.1” which is intended for use in the characterization of additively manufactured parts. The method is not restricted to this platform only and can be extended to any similar platform designed for use in various applications.

For calibrating Tiger 66.1, a new forward kinematic algorithm was developed in this research. The forward kinematics for parallel manipulator generates multiple solutions and they include both feasible and unfeasible solutions. The developed algorithm is a new way of finding unique feasible solution for a platform pose. The proposed algorithm utilizes the high power of modern computing systems and finds a unique solution for a pose through iterations. The advantage of this new algorithm is that the solution obtained from the iterations does not need to be verified manually to check the feasibility in real life and can be directly used as input for any further calculations without stopping the computation process.

The proposed calibration methods used photogrammetry which minimizes the need for manual handling of the platform during the calibration process. Photogrammetry uses images of targets taken by one or more cameras to reconstruct 3D positions and orientations. In this research a high-resolution digital camera has been used to take multiple images of the moving platform center from three different angles for each pose and analyze those images through the commercial software package “Photomodeler”, to measure the pose of the platform in three-dimensional space. This method eliminates the need for any additional measurement instrument to be used directly or indirectly interacting with the Stewart-Gough platform.

The proposed forward kinematic technique is validated through both simulations and experiments on the physical Stewart-Gough platform-based test frame Tiger 66.1. The vision-based approach is highlighted to improve absolute positioning accuracy by up to 25% compared with an uncalibrated platform. The method requires minimal hardware modification and renders it highly suitable for precision application. It provides a practical approach to the calibration of parallel manipulators intended for high-precision tasks. The dissertation concludes by summarizing the contributions of this research, which includes the development of a novel photogrammetry-based calibration method that involves minimal hardware modification and full extrinsic calibration from vision data. The demonstration shows the substantial potential of the proposed method in augmenting the positioning performance of the Stewart-Gough platform tailored for precision applications. Furthermore, this work identifies areas for further work, such as implementing an online calibration method. Overall, this research demonstrates the capabilities of photogrammetry for parallel manipulator calibration while minimizing hardware modifications, thereby presenting a pragmatic approach to the calibration of such systems in the pursuit of high-precision applications.

DEDICATION

To my wife Sujata and my son Sourajit for providing the greatest support that made this Ph.D. journey possible. Their unwavering encouragement throughout this period enabled me to complete this trek.

ACKNOWLEDGEMENTS

I express my deepest gratitude to my advisor, Prof. Dr. Cameron J. Turner, for his exceptional support and for being the driving force behind this research project. His continued support for my work, his inspiring suggestions and guidance steered me in the right direction. I deeply appreciate the opportunity to work with him as his Ph.D. student and the input he provided at various stages of my research and academic growth.

I would like to express my sincere appreciation to my other dissertation committee members, Professor Dr. Pataky and Professor Dr. Choi, and the distinguished external member Professor Dr. Laine Mears for their valuable advice and insightful tips and reviews that shaped the research work.

In this challenging journey the continuous motivations from my wife Sujata and son Sourajit remained a crucial factor for being encouraged to overcome the hardship on the path.

I convey my special thanks to Mechanical Engineering Professor Dr. Martinez-Duarte Rodrigo for his unwavering support at the start of my Ph.D. journey and for motivating me to overcome the hurdles of transitioning from a long industrial stint to academia again.

I had a big opportunity to collaborate with the members of the Clemson Engineering Design Applications and Research (CEDAR) group. The contributions and knowledge sharing by the member professors and students provided refinements of ideas that played a significant role in instigating problem-solving and critical thinking. In this regard I want to mention Dr. Apurva Patel for helping me to organize and declutter the threads of my research.

I am grateful to the Mechanical Engineering Department at Clemson University for providing generous financial support through Teaching Assistantships during my doctoral studies.

The experiences and interaction with the students from teaching in both the laboratory and classroom played a significant role in developing simpler solutions for the problems I came across in the research process. I am especially grateful to Dr. Todd Schweisinger and Dr. Douglas Byrd, my supervisors in undergraduate labs, for their mentorship and support during my teaching tenure.

I acknowledge the help and support from my fellow students, especially my lab mates Sean Taylor Fry and Travis Roberts, for their continuous interactions and discussions. I am also thankful to the other teaching, non-teaching staff of the Clemson Mechanical Engineering Department for their direct and indirect support during my research period.

Last but not the least, I express my appreciation to Clemson University, which financially supported this work, and the United States Naval Research Laboratory in Washington, D.C., which provided technical insight and support for the design and operation of the TIGER 66.1 system through NCRADA-NRL-20-719.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

Stewart platform, also known as Stewart-Gough platform or Gough-Stewart platform or as hexapod platform, is a type of parallel robotic manipulator that has been widely used in various industrial and research applications, such as robotics, aerospace, and manufacturing. A typical sketch is shown in Figure 1.1 [1]. These platforms consist of six linear actuators, or struts, connected to a fixed base and a moving platform, or end effector, in a symmetrical or asymmetrical pattern. The coordinated motion of the struts can generate six degrees of freedom at the moving platform center or end-effector, making the platform capable of precise and flexible positioning in a 6-DOF space.

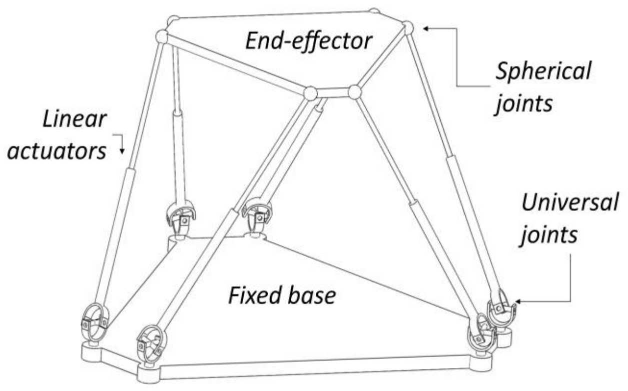


Figure 1.1: A general Stewart-Gough platform

The calibration of a Stewart-Gough platform improves the pose accuracy of its moving platform center. The *pose* of the moving platform center is defined as the position and orientation of the platform center in three-dimensional space. A pose is expressed by 6 parameters: 3 positions and 3 orientation values with respect to a cartesian coordinate frame. Calibration refers to the process of determining the relationship between target pose and the actual pose of the platform center or the end-effector and adjusting them to ensure accuracy and reliability. The accuracy of a

Stewart-Gough platform depends on the calibration of its geometry and kinematics. The length and angle of the struts, as well as the position and orientation of the end effector, need to be carefully measured and adjusted to ensure the platform's accuracy. The accuracy of the platform pose may be improved by inverse kinematics and forward kinematics. In inverse kinematics (IK), the actuator lengths are calculated from the translation and rotation values of the platform center with respect to the home pose. In *home pose* the lengths of all six actuators are equal and the platform center is considered to have no translation and rotations in the three coordinate axes. The forward kinematics (FK) calculates the coordinates and orientations of the moving platform center from the actuator lengths. Inverse kinematics for parallel manipulators is simple and straightforward, but there are several challenging aspects in Stewart-Gough platform calibration using forward kinematics. This research used forward kinematics to implement the calibration strategy.

Measurement of the actual platform pose of the hexapod is a critical part of this calibration process. Traditional measurement methods, such as tape measurement, double ball-bar, or laser scanning, may suffer from measurement noises, errors, and complexities, especially for large or complex platforms. In recent years, photogrammetry has emerged as a promising technique for various measurement processes. Photogrammetry is a method for 3D measurement and modeling based on digital images that can provide highly accurate, non-contact measurements. Images of the platform are captured at different positions and orientations using one or a set of digital cameras. These images are processed with custom software to estimate the 3D coordinates of the platform's joints and end effector and to optimize the calibration parameters.

The objective of this research is to develop, evaluate, and apply a photogrammetric calibration method for the Stewart-Gough platform. This method will improve the accuracy and precision of the platform by measuring the actual 3D spatial position of the end-effector and comparing it with the theoretical (target) positions. The newly developed kinematic algorithm in this research calculates the Denavit-Hartenberg (DH) parameters for each actuator path for the theoretical and actual pose of the platform. The DH parameter sets for theoretical and corresponding actual poses are used to estimate the errors in the platform's geometry and kinematics. From these error values, the required compensation or correction values are calculated and added to the robot’s position control software to improve accuracy. The compensation values are used to estimate the new predicted pose for each target pose of the platform. The predicted poses are used as input to the position control software. The new pose errors are measured with the help of photogrammetry and compared with the uncompensated pose errors to check the accuracy improvement obtained through this calibration process.

The contribution of this research is to provide a novel and effective method for the calibration of the Stewart-Gough platform by forward kinematics, which can significantly impact the performance and applications of these platforms. The photogrammetric method has potential applications in other areas of robotics and manufacturing and can lead to new developments in the field of parallel manipulators.

The complete calibration process demonstrated in this study is divided into three main parts. The first part of this research (Chapter 2) surveys the current state-of-the-art research in Stewart-Gough platform calibration using forward kinematics and the different methods used for it. This chapter has been prepared as a journal article, which at the time of writing this dissertation is in the review process.

The second part of the research (Chapter 3), published as a journal paper, describes the formulation of a new algorithm for forward kinematics. The mathematical formulation of the forward kinematics of the hexapod platform generates multiple polynomial equations that generate multiple feasible and non-feasible solutions for any pose of the moving platform center. The developed new iterative algorithm generates a single, feasible solution for any pose. The simulation results were generated by using this algorithm and are verified as a unique feasible solution for a specific pose.

The next part of the thesis (Chapter 4) introduces the photogrammetry method for calibrating the hexapod test frame "Tiger 66.1" developed in this lab. The pose data obtained from the newly developed algorithm was used for the development of the photogrammetric calibration method. A high-resolution digital camera is used to capture images of the platform center at various positions and orientations. A commercially available software package "Photomodeler" has been used for image processing and to extract the coordinates of the moving platform center. The target data and actual measurements were used to estimate the errors in the platform's pose. Suitable compensation factors were calculated and combined in the custom motion control software to increase the accuracy of the test frame. This chapter is also prepared in the form of a journal article that was submitted at the time of this writing.

In the concluding chapter of this study, a detailed review of this photogrammetry-based calibration method is presented for real-world applications. The outcomes of implementing this technique highlight the significant potential of the photogrammetric calibration approach for enhancing the accuracy and precision of the Stewart-Gough platform. Future research opportunities based on the proposed technique are discussed for diverse applications and the potential to drive further advancements in the field of parallel manipulators with minimal modifications to the actual system.

2 STATE OF ART

Researchers have extensively studied Stewart-Gough platforms, also known as Gough-Stewart platforms or hexapod platforms extensively for their inherent fine control characteristics. Their studies led to the potential deployment opportunities of Stewart-Gough platform in many critical applications such as the medical field, engineering machines, space research, electronic chip manufacturing, automobile manufacturing, etc. These applications require micro and nano-level movement control in 3D space for the motions to be precise, complicated, and repeatable; a Stewart-Gough platform fulfills these challenges smartly. Therefore, the parallel robot must be more accurate than the specified application accuracy level, and thus proper calibration for a parallel robot is crucial. Forward kinematics-based calibration for such hexapod machines becomes unnecessarily complex, and inverse kinematics completes this task with much ease. To experiment with different calibration techniques, various calibration approaches were implemented by using external instruments, constraining one or more motions of the system, and using extra sensors for auto or self-calibration. The literature survey in this chapter focused on the key methodologies used for calibrations, their outcomes, and key details related to forward kinematic-based hexapod platform calibrations. It was observed during this study that the researchers focused on improving the accuracy of the platform position and orientation considering the errors contributed by one source or multiple sources. The error sources considered are kinematic and structural, in some cases, environmental factors also are reviewed; however, those calibrations are performed under no-load conditions. This review aims to study the present state of the art in this field and highlight the processes and errors considered for the calibration of Stewart-Gough platform.

2.1 Introduction

In the world of conventional robots, there are three varieties of mechanisms: (i) Serial robot, (ii) Parallel robot, and (iii) Hybrid robot [2]. Any robot consists of a base and an end-effector. These are connected by multiple links. In a serial robot or manipulator, the links are connected in series. A parallel mechanism, sometimes called a Parallel Kinematic Machine (PKM), is made by linking a moving platform or end-effector, which is generally mounted on a platform or endplate, to a reference body or base through three or more links forming a closed-loop kinematic chain [3]. The base part remains fixed. A hybrid mechanism structure is formed by combining serial links and a parallel robot. Often, the parallel robot is mounted near the end effector of the serial manipulator to provide a high precision adjustment option to the serial system. The rigidity of a parallel robot is higher than a serial manipulator. The hybrid structure increases the workspace of a parallel robot at some cost of the rigidity of the structure.

Parallel robots have received significant attention because of their high dynamic flexibility, structural rigidity, high accuracy due to the closed kinematic loops, no error accumulating characteristics [4], higher load-to-weight ratio, and uniform load distribution capacity compared with the serial manipulators [5]. For any parallel robot, the number of linking elements between the fixed base and movable platform varies between three and six. The link numbers together with the type of connections and the twist of the platform normally determine the degrees of freedom (DOF) of the machine.

One such parallel robot controlled by six links connected between the fixed base and movable platform with 6-degrees of freedom (DOF) is termed Hexapod. The hexapod was first designed by an engineer Gough from the United Kingdom in 1954 for tire testing with six actuators acting as the links between the fixed base and its moving platform. The actuators are prismatic joints. This machine had the structure of an octahedral hexapod [6]. Using the Gough's platform, in 1965 another engineer from the United Kingdom, Stewart developed an articulated 6-DOF flight simulator [7] for the training of pilots. These types of platforms are known as a Stewart-Gough Platform or sometimes Gough-Stewart Platform, or simply as a Stewart Platform. In this article, these terms are used to mean the same machine. The combinations of motions of the six actuators give the platform high precision, high structural stiffness, and high dynamic performance [8]. Stewart-Gough platforms have been employed in many fields. The potential applications of parallel robots include mining machines, walking robots, both terrestrial and space applications including areas like high-speed manipulation, material handling, motion platforms, machine tools, medical fields, planetary exploration, satellite antennas, haptic devices, vehicle suspensions, variable-geometry trusses, cable-actuated cameras, and telescope positioning & pointing devices [9]. They are used in the development of high-precision machine tools by many companies like Giddings & Lewis, Ingersoll, Hexcel, Geodetic, and others [10] [11]. The application options expanded from a simulator to automobile manufacturing, inspection, human-robot collaboration, space telescope, medical tool control (by adding a hexapod at the end-effector point of a serial manipulator) [12]. For the precision and accuracy required for these machines to perform at a specific level of operational characteristics, the platform movements must be precisely controlled. To get the necessary level of accuracy for the moving platform position and orientation, called *pose*, it is essential to understand the various errors related to the machine at the time of its operation and apply suitable compensation. Calibration of the hexapod identifies these errors and adds suitable amounts of compensations to obtain reliable and predictable [13] output data.

This chapter of the dissertation surveys the calibration methods used for hexapod platforms based on Stewart-Gough platforms. Efforts were made to cover most of the key articles published after the year 2000 that focused on forward kinematic calibration techniques. This chapter has been divided into six main sections. The beginning section serves as an introduction. The second section reviews the kinematics of hexapods and the primary error factors that impact their accuracy. Section 3 reviews the calibration techniques and the strategies used for successful calibration thereof. Major calibration methodologies and their outcomes are presented in Section 4. Section 5 discusses and compares the methods previously presented. Finally, section 6 provides a conclusion and recommendations for future work.

2.2 Hexapod Kinematics & Error Factors

A general 6-6 hexapod configuration is shown in Figure 2.1, with the appropriate terms defined using the nomenclature of Tsai [14] and Lee [15].

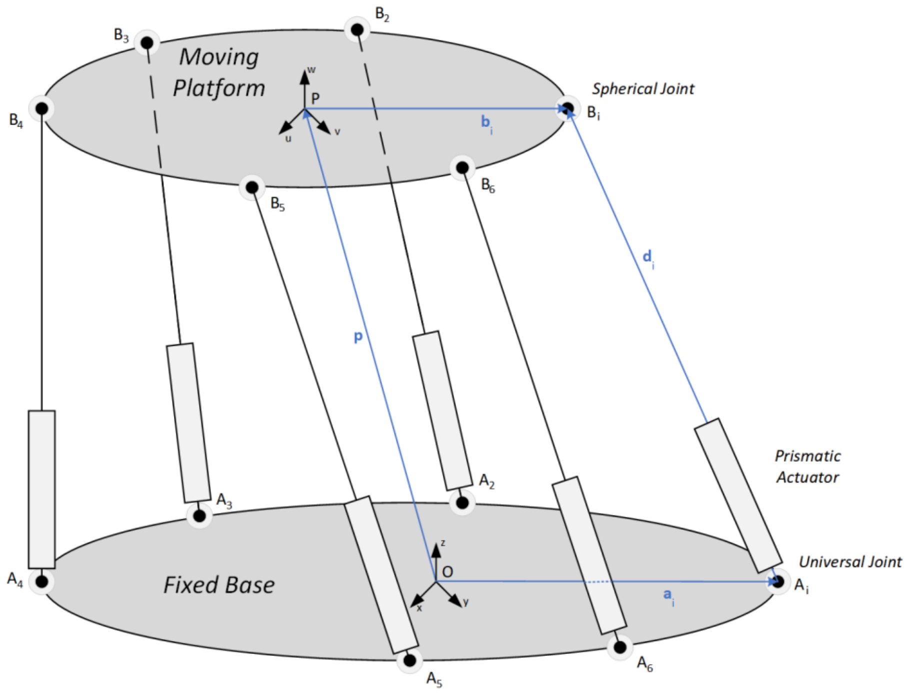


Figure 2.1: Hexapod Schematic with Nomenclature.

In Figure 2.1, the position of the Fixed Base coordinate system (System O) defined at point O compared to the Moving Platform coordinate system (System P) defined at point P is defined through the translation vector $\boldsymbol{p}$, and a rotation matrix ${}^{O}_{P}\boldsymbol{R}$. The rotation matrix can be further defined as in Equation (2.1) as:

$$ {}^{O}_{P}\boldsymbol{R} = \begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix} \tag{2.1} $$

where $\boldsymbol{u}$, $\boldsymbol{v}$, and $\boldsymbol{w}$ are unit vectors along the axes of the moving platform coordinate system, subject to the orthogonal conditions defined in Equations (2.2) through (2.7).

$$ u_x^2 + u_y^2 + u_z^2 = 1 \tag{2.2} $$

$$ v_x^2 + v_y^2 + v_z^2 = 1 \tag{2.3} $$

$$ w_x^2 + w_y^2 + w_z^2 = 1 \tag{2.4} $$

$$ u_x v_x + u_y v_y + u_z v_z = 0 \tag{2.5} $$

$$ u_x w_x + u_y w_y + u_z w_z = 0 \tag{2.6} $$

$$ u_x w_x + u_y w_y + u_z w_z = 0 \tag{2.7} $$

If we define ${}^{O}\boldsymbol{a}_i = [\mathrm{a}_{ix} \quad \mathrm{a}_{iy} \quad \mathrm{a}_{iz}]^T$ as the vector from point O to point A_i (where i = 1, 2, …, 6) in coordinate system O, and in coordinate system P, ${}^{P}\boldsymbol{b}_i = [b_{ix} \quad b_{iy} \quad b_{iz}]^T$ as the vector from point P to point B_i (where i = 1, 2, …, 6) in coordinate system P, this allows to define a vector loop equation for the ith arm of the platform as in Equation (2.8).

$$ \overline{A_i B_i} = {}^{O}\boldsymbol{p} + {}^{O}_{P}\boldsymbol{R}^{P} b_i - {}^{O}\boldsymbol{a}_i = {}^{O}\boldsymbol{d}_i \tag{2.8} $$

Thus, the length of the ith arm controlled by the prismatic actuator i, is defined in Equation (2.9) as:

$$ {}^{O}\boldsymbol{d}_i^2 = \|{}^{O}\boldsymbol{d}_i\| = [{}^{O}\boldsymbol{p} + {}^{O}_{P}\boldsymbol{R}^{P} b_i - {}^{O}\boldsymbol{a}_i]^T [{}^{O}\boldsymbol{p} + {}^{O}_{P}\boldsymbol{R}^{P} b_i - {}^{O}\boldsymbol{a}_i] = {}^{O}\boldsymbol{d}_i \bullet {}^{O}\boldsymbol{d}_i \qquad \forall i = 1,2,\ldots,6 \tag{2.9} $$

Depending on the inputs and outputs to and from the kinematic problem, the solution to the resulting system of equations is defined as either *forward kinematics* or *inverse kinematics* [16]. In forward kinematics, the position and orientation of the moving plate are calculated based on the length of the six actuators, defined by $\boldsymbol{d}_i$ and expressed as ${}^{O}\boldsymbol{p}$ and ${}^{O}_{P}\boldsymbol{R}$.

The opposite calculation is inverse kinematics where the position and orientation of the moving plate, ${}^{O}\boldsymbol{p}$ and ${}^{O}_{P}\boldsymbol{R}$, are known and the required length of each of the *i* actuators, defined by $\|\boldsymbol{d}_i\|$, is to be determined [17].

In both cases, the correctness of the hexapod parameters is dependent on a number of error factors which can be geometric or non-geometric [18] [19]. These error parameters affect the values of all the variables which define the kinematics of the hexapod. Depending on the error factors, the calibration process is classified into three levels [13] [20] [21]:

Level-1 calibration considers only the joint errors that play a critical role in the accuracy of the robot. This is defined as "Joint Level Calibration."

Level-2 calibration, also known as "Kinematic Model Calibration", takes care of the error of the kinematic parameters.

Level-3 calibration, also called as "Non-kinematic calibration" or "Dynamic Model Calibration", captures the errors of non-geometric or quasi-static parameters [22], such as stiffness, geometry of the robot structure, and errors caused by temperature variation [23].

In a hexapod platform, the components of the rigid structure like the base, frame, top platform, and other accessories are fabricated and normally made from metal stocks. Therefore, the accuracy of these components has a direct influence on the accuracy of the entire system. The dimensions of the structure are dependent on the design tolerances or manufacturing deviations, clearances, joint errors [24], thermal deformations [25] [26], and elastic deformations [27]. The actuators or struts are connected to the structure with movable joints; joints are impacted by the errors due to the assembly deviations in the form of joint run-out and ball screw deviations. Also, the mechanical joints are not free from friction, hysteresis, and backlash [4]. If the struts are operated by hydraulic fluids, there are chances of transmission errors and sensor errors [28].

Therefore, the accuracy of a hexapod can be expressed broadly by the function described in equation (2.10).

$$\boldsymbol{E}_{Hexapod} = f\left(\boldsymbol{E}_{manufacturing}, \boldsymbol{E}_{assembly}, \boldsymbol{E}_{transmission}, \boldsymbol{E}_{deformation}, \boldsymbol{E}_{sensor}\right) \tag{2.10}$$

These error parameters are further elaborated in Table 2.1. Each of these errors can affect the position and orientation of the endplate. For instance, $\mathbf{E}_{manufacturing}$, is the resultant error vector due to tolerances and manufacturing accuracy along the kinematic chains that define the hexapod. Conceptually, each degree of freedom in a kinematic chain is defined by four Denavit-Hartenberg (DH) parameters, representing the information necessary to transform the coordinate system across the component providing the degree of freedom (DOF). Therefore, on a 6-DOF kinematic chain defining one limb of a Stewert Platform, there are 4(6) or 24 DH parameters. With one prismatic actuator, there are 23 uncontrolled parameters and 1 controlled parameter in each limb. With six limbs, there are a total of 6 controlled DH parameters and 6 x 23 = 138 uncontrolled DH parameters in the Stewert platform. However, because these parameters are coupled into closed-loop kinematic chains, these parameters are coupled to the other parameters in the system, resulting in a 6-independent DOF for the endplate, and the remaining 138 uncontrolled parameters are dependent upon each other and the 6-independent DOF from the prismatic actuators. Errors originating from manufacturing tolerances and accuracy can have a cumulative effect that alters all uncontrolled DH parameters.

Similarly, assembly errors can also lead to changes in any of these parameters, although some errors may be more likely to be seen in certain parameters (i.e., wear is more likely to affect parameters specifically associated with the joint rather than the links). Similar arguments can be made for the other error components described in Table 2.1.

Table 2.1: Error function illustrations

Error source	**Dependency**	**Remarks**
Manufacturing	Component Tolerances	These components act as the basic structure of the machine and any deviation impacts permanently. This structure bears all the loads generated in the static and dynamic condition of the machine and provides rigidity to the machine.
Assembly	• Assembly Tolerances • System Age • Amount of Usage/Wear	Assembly deviations are controllable and minimized by the replacement of old, worn-out parts with new parts.
Transmission	• Actuation Response Time • Joint Clearance and Backlash • Platform Position • Operation Speed and Lag • Hysteresis Effects	This error depends on the robustness of the system and system configuration. Some default limitations cannot be avoided.
Deformation (Mechanical)	• Material Properties • Applied Loads • Component Geometry • Platform Position	Dependent on the structural materials used and their response properties under load.
Deformation (Thermal)	Working temperature variations	Changes in operating conditions due to temperature may affect structural components, joint tolerances, and actuator performance.
Sensor	• Specification Tolerances/Accuracy • Calibration Drift	Modern sensors tend to be the least inaccurate.

Some of the error sources, in particular the transmission and sensor errors, also can affect the controlled parameters associated with the actuators of each limb. Consequently, the task of calibrating a hexapod involves confirming that each of these 144 DH parameters is known throughout the operating range of the system.

There are more typical sources of errors [29], other than those mentioned here, contribute to the outcome from the system; however, their influences are dependent on the construction philosophy adopted for that particular system, and the type of operations and level of accuracy expected out of those systems.

2.3 ACCURACY IMPROVEMENT STRATEGIES

The purpose of kinematic calibration of parallel robot is to improve the motion and position accuracy of the moving platform by correctly evaluating and calculating the kinematic parameters within its defined workspace. A parallel robot can be calibrated in three ways (Figure 2.2): External calibration, Constrained calibration, and Auto or Self-calibration [30] [31].

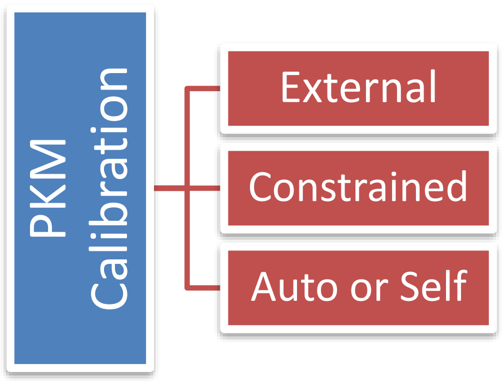


Figure 2.2: Strategies for PKM calibration

External calibration is done by using one or more external instruments such as an electronic theodolite or a laser tracker [32] for measurement of multiple poses of the end-effector. In the Constrained calibration process, some motions of the mechanical elements, usually the movement of a struts of the robot are constrained to gather the error data. This method is comparatively simplest and least expensive [33]. Auto or Self-calibration is one of the most expensive and complex calibration techniques. In this method, the robot itself automatically takes care of the error parameters measured by redundant sensors with the help of the built-in algorithms installed on controllers. The error correction process can take place during normal robot operations. Several extra sensors are installed in the joints and on the links of the robot to gather calibration data continuously. Alternatively, additional sensors can be added to directly measure the end plate motion through extensometers or feeler arms [34] [35]. The number of sensors used in a parallel machine is equal to its number of degrees of freedom. In all these strategies, sensors play an important role and are an essential part of the calibration process; the difference occurs in how these sensors are employed.

Conventionally, the four steps shown in Figure 2.3 are followed in the calibration process of a parallel robot: kinematic modeling, measurement, identification, and implementation [36]–[38].

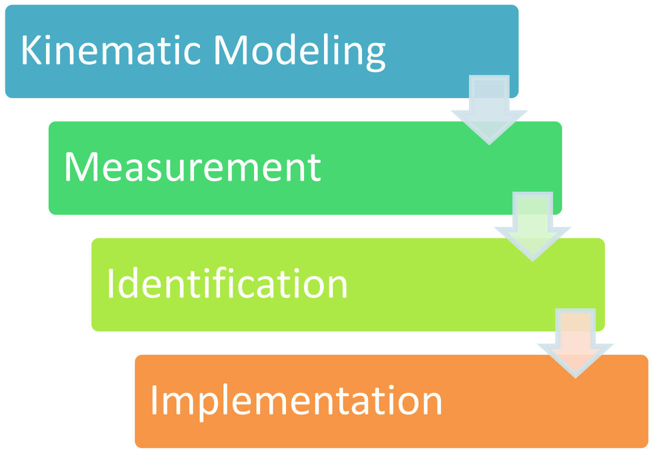


Figure 2.3: Steps in the PKM calibration process

Kinematic modeling of the platform is used to build the relationship between the joint variables and the platform pose, along with the measurement device readings. The result of the modeling phase is a set of analytical equations showing relationships between these parameters as shown in Equations (2.1) - (2.9) [18] [19] [39]. The measurement phase gathers the data related to the actual platform position and orientation with the help of the measurement devices [22]. The identification step will identify the optimal set of unknown parameters based on the kinematic model and measurements to fit the actual behavior of the mechanism considering the major sources of errors for the context [8] [40]. Finally, through implementation, the compensation for the calculated model errors is added in the manipulator controller [38].

The results of the robot calibration process are expressed in terms of the pose errors for a set of positions and orientations [41]. To generate a reliable and accurate result, the mechanical structure of the robot must be defined with an adequate number of parameters without repetition in the calibration model. A "good" calibration model must have three criteria: completeness, equivalence, and proportionality. Completeness refers to the fact that the model must have enough parameters to completely define the motion of the robot. Equivalence means that the derived functional model can be related to any other similar acceptable model. The proportionality property must give the model the ability to reflect small changes in the robot geometry with small changes in the model parameters [20]. A "*good*" calibration model can be established by building relationships between the independent parameters that are used to define the robot system. For a multiloop parallel robot such as the 6-DOF hexapod or Stewart-Gough platform, the number of independent parameters (C) can be calculated as follows [42]:

$$C = 3R + P + S + SS + E + 6L + 6(F - 1) \tag{2.11}$$

where…

- L, number of independent link loops in the robot,
- R, number of unsensed (non-instrumented) revolute joints,
- P, number of prismatic joints,
- S, number of unsensed (non-instrumented) spherical joints,
- SS, number of pairs of S-joints connected by a simple link without any intermediate joint,
- E, number of measurement devices or transducers, and
- F, number of arbitrarily located frames.

For a Stewart-Gough platform with one universal (U), one prismatic (P) and one spherical (S) joints for each actuator considering [UPS ≈ 2RP3R], the minimum total number of independent parameters necessary for the complete calibration model is as follows.

$$C = 3 * 6 * 5 + 6 + 0 + 6 + 6 * 5 + 6(2 - 1) = 138$$

2.4 Calibration Approach

To achieve a high level of kinematic accuracy, it is necessary to develop a robust and reliable calibration method. After the introduction of the hexapod by Gough, and then by Stewart, calibration became the field of interest for researchers. Calibration research is typically performed using either analytical approaches or physical experiments. Analytical approaches are independent of the physical hexapod platform artifacts. Still, they are worth studying to get an idea of the research direction on the analytical hexapod calibration process.

2.4.1 Calibrations through Analytical Approaches

Jing *et al.* [43] considered that the final position and orientation of a hexapod platform depend on the joint radius and angles of the movable platform. In their analysis, they applied an interior-point algorithm. To validate their considerations, analytical techniques were used to reduce the errors of the 6 actuators. Through their analysis, they could reduce the average actuator length error from 0.1 mm to 2 x 10E-7 mm.

In their research, Agheli *et al.* [44] considered the lengths of the actuators as the main sources of error for the moving platform accuracy in the hexapod and accordingly designed the calibration process to minimize the cost function through the simulation of the calibration process. They obtained approximately 50% error reduction in platform position and orientation errors at the workspace boundaries. For their simulations, Agheli *et al.* used the Levenberg-Marquardt algorithm to minimize the cost function obtained through inverse kinematic calculations.

For small numbers (= 10) of measurements Daney *et al.* [45], they used Algebraic variable elimination and monomial linearization to calibrate the platform pose. They used the actuator lengths as the error source and compared their algorithm with classical non-linear least-square methods. They obtained the same results. The advantage of this algorithm is that for small numbers of measurements, there is no need for any hypothesis for noise distribution and no initial estimate of solution is taken.

In [46], Daney *et al.* presented a method of calibration based on interval arithmetic and interval analysis to solve an over-constrained equation. They used Taylor expansion to obtain a linear approximation to determine the kinematic parameters. The least-square method and their developed certification algorithm provide exact kinematic parameters when no errors are considered for the measurements. Considering kinematic errors, the results are comparable with the classical least-square method.

Daney *et al.* [28] used the constrained optimization method and an algorithm named DETMAX for inverse kinematics applications. In this case, they studied the impact of errors at joint positions and actuator lengths. They found that the mean error on kinematic parameters improved from 0.2705 cm to 0.0335 cm for random poses and 0.2705 cm to 0.0023 cm for selected poses.

Wang *et al.* [47] identified that errors from actuator lengths are the most dominant error factor for the overall accuracy of the 6-DOF platform. They considered errors from actuator lengths, ball joint location, and motion errors in their analysis. Based on the sensitivity analysis of the errors, they produced a graphical presentation of the optimal working region for their machine tool used in the experiment.

In another study [48], Daney *et al.* carried out simulation using symbolic variable elimination with numerical optimization. These yielded superior results compared to the classical direct methods and were able to reduce initial pose error by up to 99% near the boundary configurations. With this study, the authors concluded that their algorithm is dependable irrespective of the robot configuration.

A summary of the research on analytical-based calibration is shown in Table 2.2.

Table 2.2: Summary of the analytical work on hexapod calibration

Reference	Error/s considered	Methods	Outcome	Performance	Important points
Jing *et al.* 2020 [43]	Moving platform joint radius and angles	Interior-point algorithm	Average actuator length error reduced from 0.1 mm to 2 x 10E-7	Use of MATLAB optimization toolbox could find local optimal solution	Though other optimization methods may work better, the authors preferred this to tryout.
Agheli *et al.* 2009 [44]	Actuator length	A least-square method based on Levenberg-Marquardt algorithm	Reduction in platform pose error by ~50% at the boundary	The position and orientation errors reduced 500 and 15000 times, respectively.	Maximum Kinematics Parameters errors were observed at the boundary of the workspace.
Daney *et al.* 2004 [45]	Actuator length	Algebraic variable elimination and monomial linearization	Classical nonlinear least-squares method and this method generates exactly same results	A superior method for small numbers (=10) of measurements.	Need no hypothesis for noise distribution and no initial estimate of the solution.
Daney *et al.* 2004 [46]	Actuator length	Interval arithmetic and interval analysis	The new measurement method shows comparable results to the classical least-square method	The interval analysis provided numerically certified result to kinematic calibration problem	The classical least square method may not provide realistic solutions for all cases.
Daney D. 2002 [28]	Joint position and actuator length	Constrained optimization method and DETMAX algorithm	Measurement noise in kinematic parameters reduced by 10 to 15 times	Mean error on kinematic parameters improved from 0.2705 cm to 0.0335 cm for random poses & to 0.0023 cm for selected poses.	The error value decreases steadily with an increase in the number of randomly chosen poses and remains usually constant for carefully chosen configurations.
Wang *et al.* 2002 [47]	Actuator length, location, and motion errors of ball joints	An automated error analysis model of first and second order inverse kinematics	Graphically sensitivity analysis results to select optimal working region	1 mm actuator length (z-axis) error causes platform deviation in x-axis from -140 to -180 μm, y-axis from 520 to 650 μm, z-axis from -150 to -350 μm	The length error (z-direction) of the actuators influences the accuracy of the machine much larger than any other errors.
Daney *et al.* 2001 [48]	Platform pose	Symbolic variable elimination with numerical optimization	More reliable algorithm irrespective of configurations compared the standard direct methods	Initial pose error was reduced by 99%.	An efficient technique to enhance the robustness of the measurement process. Possibility of this method for the self-calibration process.

2.4.2 *External Approach*

External approaches are widely used as a method for hexapod calibration. In this approach, the calibration is done through experiments on the hexapods using additional measuring instruments. Instruments like a double ball bar, laser interferometer, laser tracker, digital cameras, etc. are used [31]. Highlights of external approach based hexapod calibration have been summarized in Table 2.3 and Table 2.4. The system name used for the experiment, instruments used, type of error considered, and kinematics are presented in Table 2.3; whereas the methods, performance, and key findings of the study are presented in Table 2.4.

Table 2.3: Summary of external approaches (part 1 – left side)

Reference	**System Name**	**Instruments used**	**Error considered / measured**	**Kinematics**
Song *et al.* 2022 [49]	-	FARO measuring arm	Joint errors	Inverse
Mahmoodi *et al.* 2014 [50]	-	6 rotary sensors on 6 actuators	Actuator length	Forward/ Inverse
Jáuregui *et al.* 2013 [18]	Secondary mirror of a radio-telescopes.	Laser Interferometer	Actuator length	Inverse
Ren *et al.* 2013 [36]	XJ-HEXA	Biaxial Inclinometer & Laser tracker	Actuator length	Inverse
Nategh *et al.* 2009 [51]	Hexapod table	Digital camera	Platform pose	Forward/ Inverse
Großmann *et al.* 2008 [52]	FELIX	Double Ball Bar	Platform pose	Inverse
Liu *et al.* 2007 [53]	-	3D Laser Tracker	Actuator lengths	Inverse
Ting *et al.* 2007 [54]	Micro-positioning platform	DMT22 Dual Sensitivity Systems with C5 probe	Hysteresis of Piezoelectric actuators	Inverse
Daney *et al.* 2006 [17]	DeltaLab robot	CCD camera.	Joint imperfection and backlash of each actuator.	Inverse
Dallej *et al.* , 2006 [55]	-	Omni-directional camera	Position and orientation of the actuators	Inverse
Daney *et al.* 2005 [56]	DeltaLab "Table of Stewart"	Sony digital video camera	Platform pose	Inverse
Gao *et al.* 2003 [57]	FFCM of FAST	Laser Tracker LTD500	Platform pose	Inverse
Renaud *et al.* 2002 [58]	-	A CCD camera & 1D laser interferometry	Platform pose	Inverse
Week *et al.* 2002 [59]	Ingersoll HOH600	Double ball bar as 7th actuator & a CMM	Actuator length	Inverse
Ihara *et al.* 2000 [60]	-	Telescoping magnetic ball bar (DBB)	Actuator length and joint positions	Inverse

Table 2.4: Summary of external approaches (part 2)

Reference	Methods	Performance	Important points
Song *et al.* 2022 [49]	Artificial Neural Network (ANN) based non-linear functions	Mean pose error reduction from 0.642 mm and 0.184 deg to: Coupled network: 0.076 mm & 0.024 deg respectively; Decouple network: 0.052 mm & 0.018 deg respectively	The coupled and decoupled networks show a comparable results pattern though the optimal numbers of hidden nodes for couple network is 13 and decoupled network is ~6.
Mahmoodi *et al.* 2014 [50]	A new method with 6 rotary sensors	Positional and orientation variances improved by 0.16 m^2 & 0.16 rad^2 respectively.	The new method is less accurate in orientation measurement. This method is better than conventional method for position measurement.
Jáuregui *et al.* 2013 [18]	Simplified method (same error amount considered for all actuators) & comprehensive method (each actuator error is not equal)	Majority of pose deviations fall within 10µm.	The simplified method creates linear relationships and is easy to solve. The comprehensive method is complex, non-linear, but more accurate.
Ren *et al.* 2013 [36]	Keeping any two attitude angles of the end-effector constant.	Position accuracy = 0.1 mm Orientation accuracy = 0.011°	Exempting the need for precise pose measurement and mechanical fixtures. Independent of inclinometer range and accuracy.
Nategh *et al.* 2009 [51]	A least-square approach based on Levenberg-Marquardt algorithm with Singular value decomposition.	The position & orientation errors as per simulation were 0.1 mm and 0.01° respectively and were 1.45 mm and 0.27° as per experiment.	Employed Observability index to find the most visible & optimum number of measurement configurations.
Großmann *et al.* 2008 [52]	Genetic Algorithm based Trajectory optimization.	The deviation of platform pose reduced from 0.7 mm to 0.17 mm.	Genetic algorithms are slow to get the most accurate solution, also rarely improve the solution.
Liu *et al.* 2007 [53]	Genetic Algorithm	After 5000 generations the platform position & orientation errors improved 1.4 & 2.4 times respectively without measurement noise filter.	The genetic algorithm showed good calculation stability, though it is not sensitive to measurement noises.
Ting *et al.* 2007 [54]	Preisach model	Platform accuracy level achieved 1 µm in position and 10 µ deg in orientation.	The convergence of errors for fixed points can happen after several iterations.
Daney *et al.* 2006 [17]	Interval arithmetic and analysis methods	Yielded intervals for the position and orientation, which includes noise and robot repeatability error.	Finds ranges of parameters that satisfy the calibration model.
Dallej *et al.* , 2006 [55]	Linear regression	Experimental validation of the method yielded 0.8 cm median error with respect to the CAD geometry.	An omnidirectional camera overcomes the self-occlusion problem arising in the single perspective camera. No mechanical modification of the robot is necessary.

Reference	Methods	Performance	Important points
Daney *et al.* 2005 [56]	Constrained optimization method with Tabu search	Improvements in accuracy were not as per expectation due to the biasness error of 1.29 mm on the z-axis.	The workspace boundary has a concentration of optimal poses. By maximizing the observability index, the robustness of calibration increases with respect to measurement noise.
Gao *et al.* 2003 [57]	Least Square method	Accuracy improved to 0.2 mm	The method is effective even in lack of measurements. Some false parameters may occur for fewer measurement configurations.
Renaud *et al.* 2002 [58]	Error function minimization.	Precision obtained in camera measurement is in the order of 1 μm in translation and 1x10^{-3} deg in rotations for an axial displacement of 400 mm	Low cost and easy to use compared to the measurements by interferometer. Precision level depends on the camera resolution.
Week *et al.* 2002 [59]	-	Roundness accuracy improved by 3.7x Squareness accuracy improved by 7x.	The redundant actuator can be used to measure and compensate for the deflections due to gravity and thermal error.
Ihara *et al.* 2000 [60]	Fourier transformation	Machine's motion error decreased to ¼.	The measurement is easy and can take care of circularity, absolute radial error, and circle center position error.

The artificial neural network (ANN) was used by Song *et al.* [49] in the calibration process for their Stewart-Gough platform. They corrected the joint variables by embedding the compensations in their numerical control system for online real-time error compensation. They did the experiment with their hexapod and demonstrated that the proposed ANN based robust compensator can enhance static pose accuracy for both coupled and decoupled networks. The ANN approach was implemented with inverse kinematics. The results obtained show that mean pose error reduced from 0.642 mm and 0.184 deg to 0.076 mm & 0.024 deg respectively for coupled network and to 0.052 mm & 0.018 deg respectively for decouple network though the optimal numbers of hidden nodes for couple network is 13, decoupled network is ~6.

Mahmoodi *et al.* [50] proposed a new method of calibration for Stewart platform-based PKM. They used rotary sensors in place of the linear sensors in actuators. The method is not too sensitive to the orientation measurement but showed better results in position measurement for the platform. In this study, 6 rotary sensors were used on 6 actuators to correct the pose of the platform. They used a mix of forward and inverse kinematics for their PKM and observed the positional and orientation variances improved over the conventional methods to 0.16 m^2 and 0.16 rad^2 respectively for both small and moderate movements.

The studies by Jáuregui *et al.* [18] used a laser interferometer as the measuring instrument to calibrate the hexapod. They used inverse kinematics and considered the error related to the actuator length. Their experiment consisted of two methods. In the first method, they considered the error from all actuators to be the same, applying linear relationships. This was simple and easy to solve. In the second method, labeled it as the comprehensive method, each actuator error was measured separately and modeled in a non-linear relationship. As expected, the second method resulted in complex calculations but yielded greater accuracy.

Ren *et al.* [36] did their experiment with their PKM named XJ-HEXA using a biaxial inclinometer with the length precision of 0.002 mm and repeatability for angles of 0.001°. They reached a position and orientation accuracy up to 0.1 mm and 0.01° respectively after calibration of 80 configurations. They kept two of the three orientations defining angles of the moving platform constant during the measurements.

Nategh *et al.* [51] studied their “Hexapod Table” with the use of an image capturing system. In this research, the results obtained by the simulations and experiments matched very closely. The platform position & orientation errors as per the simulation were 0.1 mm and 0.01° respectively, whereas those from the experiment were 1.45 mm and 0.27°, respectively. A least-squares approach based on the Levenberg-Marquardt algorithm was employed in this calibration process with Singular value decomposition.

In another study by Großmann *et al.* [52] used a Double Ball Bar (DBB) to identify and collect the kinematic parameters by moving the platform on a specific trajectory in the 3D workspace. They used a genetic algorithm and simulated measurements to finalize the parameters. Their hexapod named FELIX was designed and manufactured with a focus on simplicity and capacity for compensating the motion errors generated due to the thermal and elastic deformations. Thermal and elastic deformations were considered in the algorithm by incorporating fixed factors. By this method, they measured the error along a trajectory and were able to reduce the initial deviation from 0.7 mm to 0.17 mm after optimizing the trajectory orientation through kinematic calibration.

Liu *et al.* [53] also used a genetic algorithm for calibrating their hexapod using inverse kinematics. They measured the errors coming from the actuator lengths and used a 3D laser tracker for the measurements. The genetic algorithm converged initially fast but gradually became slow little by little. For their experiments, though they obtained an improvement in the error of the platform for the position by 1.4 times and for the orientation by 2.4 times, they found that the genetic algorithm is not sensitive to the measurement noises.

The applications of hexapod did not remain restricted to the dimension level of mm or inch, it has attracted attention for micro-level applications too. Ting *et al.* [54] did their experiment with a 6-dof micro-positioning platform to evaluate the platform error due to the hysteresis of the piezoelectric actuators. By using inverse kinematics with the Preisach model, they achieved a platform accuracy at the level of 1 μm in position and 10 μ deg in orientation after several iterations. For their experiment, they used Lion Precision DMT22 Dual Sensitivity Systems with C5 Probe for measurement.

A popular method of hexapod calibration includes vision-based data collection. Daney *et al.* [17] employed calibration processes using a 1024x768 CCD camera. In their experiments, they used plates with dot marks as visual targets and obtained their images for measurement analysis. The final errors measured by them were large with respect to the length of the actuators and with that they concluded that the kinematic model used was not robust.

Dallej *et al.* [55] used an omnidirectional camera. The measurement of the actuators had been investigated by using images from external cameras. They used an omnidirectional camera which overcomes the self-occlusion problem arising in the single perspective camera. By using the omnidirectional camera, Dallej *et al.* got a median error of approximately 1 cm when compared to the CAD geometry and data obtained from the camera.

Researchers Daney *et al.* [56] did several experiments with hexapod and other PKMs. They used this work on the machine DeltaLab's "Table of Stewart" involved Sony digital video camera (1024 × 768) with a 4.2 mm focal length for measuring the joint positions and actuator lengths. In this case, they used the Constrained optimization method by combining it with the Tabu search, but the results obtained were not satisfying due to the error resulting from the bias of the system along the z-axis in the range of 1.29 mm. They selected 18 and 64 random poses for analyzing the pre- and post-calibration error values.

Gao *et al.* [57] carried out their study and calibration of a Five-hundred-meter Aperture Spherical Telescope (FAST) using inverse kinematics. They used a laser tracker for measurements and controlled the position and orientation of the platform with a Stewart platform-based Fine Feed Cabin Model (FFCM). In their study, they were able to achieve the desired accuracy level of 0.2 mm for the FAST. Here they had not considered any specific error factor except the final pose of the telescope.

In their research, Renaud *et al.* [58] used a CCD camera and an LCD monitor to calibrate a 6-DOF PKM machine tool by using inverse kinematics. They compared the results by comparing the measurement data obtained for the same poses by a laser interferometer. The platform pose of the machine had been measured for the comparison. The final comparison showed that the precision level obtained from the optical measurement is in the order of 1 μm in translation and 1x10^{-3} deg in rotations for an axial displacement of 400 mm.

The findings by Week *et al.* [59] considered the errors on the actuator length. They used a Double Ball Bar (DBB) as a redundant actuator in their Ingersoll HOH600 robot. The accuracy of roundness and squareness in their machining tests was improved by factors of 3.7 and 7.0, respectively. The system of setup they developed could be used to measure the errors due to the thermal and gravity load deflections at the end-effector pose.

An investigation by Ihara *et al.* [60] using a Telescoping magnetic ball bar (DBB) resulted in a reduction in motion error of the platform by 25% after calibration. They used Fourier transformation and included the length error of struts, and position errors of base & platform joints of the platform for optimizing the overall error of the system.

2.4.3 Constraint Approach

The constraint calibration approach has the limitation of practical applications, for this reason this calibration approach is less popular among the researchers. Constraint approach for calibration is implemented by using some mechanical constraints to restrict the motion of one or more joints in the parallel robots [32]. Normally no external measuring instrument is used. The already existing sensors in the system act as the measuring sensor. The applied motion constraint causes a reduction in the degrees of freedom for the end-effector and reduction of workspace of the manipulator. The kinematics parameters also get reduced. The force generated due to constraining the movement may distort the structure and impact the accuracy of the calibration. As the system loses one or more degrees of freedom and the number of sensors becomes more than the active degrees of freedom, the calibration process may be considered as self or auto calibration process of a system with the reduced degrees of freedom.

Ryu *et al.* 2001 [32] used this approach to calibrate a hexapod “Hexa Slide Mechanism (HSM)” by constraining one actuator at a time and repeating the process for all six actuators. Ryu constrained the motion of the platform by restricting the motion of one actuator of the system and the system worked as a 5-DOF system instead of the regular 6-DOF system. There was no extra sensor used, the in-built existing actuator sensors were used for taking the needful measurements. By constraining one actuator, the initial error values of the platform position 8.0E-3 m and orientation 1.4E-2 rad converge to 3.8E-16 m & 1.7E-15 radians, respectively.

Rauf *et al.* 2001 [61] constrained 3 actuators and experimented with 3-DOF in the same hexapod “Hexa Slide Mechanism (HSM)” mentioned in previous paragraph. Here also the final correction values are near zero. For 3-DOF measurements, initial error values of position 1.0E-2 m and orientation 1.7E-2 radians changed to 2.4E-10 m & 1.8E-9 radians, respectively. The final correction values change by very small amounts depending on the value of the measurement noises. The advantage of these methods is that the locking device can be universal and need not be specific for a particular system.

2.4.4 Auto or Self Calibration Approach

Auto or Self-calibration is one of the ways to calibrate Stewart platform-based 6-DOF kinematic machines. This method requires adding one or more redundant sensors to the passive joints or an additional redundant passive limb [31]. The addition of extra increases complexity of the design and manufacturing processes of PKMs. Moreover, the addition of redundant sensors sometimes makes system development more expensive. The auto or self-calibration method also limits the workspace for calibration. Some research studies have been done with this method.

Similarly, Guo *et al.* [34] used a smaller set of four (4) extensometers to directly measure the position of the end plate relative the fixed plate during multimodal loading tests. The data was used in real-time to connect to an ADAMS model of the system, and to generate force feedback information as part of the system control loop. This use of continuous calibration demonstrated that forces could be very precisely controlled during testing.

Mura [35] used a set of wire extensometers to directly measure the position of the moving end plate relative to the fixed plate during testing of flexible automotive components. The testing motions used ensured that the extensometers remained in tension throughout the test and provided accurate positional data while not imparting a significant amount of additional stiffness into the system.

In their study, Chiu *et al.* [62] used a cylindrical gauge block and a commercial trigger probe to do the auto-calibration of the PKM. They made use of the non-linear least square method in their algorithm. The advantage of their method is that the instruments are standard and commercially developed, which makes them easily available and less expensive. Multiple PKM configurations were used to validate their method and the results showed that various levels of accuracy were achieved.

Similarly, in another auto-calibration process, Zhuang *et al.* [63] used a Coordinate Measuring Machine (CMM) with their robot FAU Stewart Platform. Here also the position and orientation of the platform were used to calibrate system errors. They used Levenberg-Marquart algorithm for optimization and got error reduction by 50%. The highlights from their research are that some extra sensors needed to be installed in some of the joints to gather calibration data. Another conclusion they drew is that if the end-effector is a separate attachment on the hexapod platform, then the end effector should be calibrated separately.

Table 2.5: Assessment of auto or self-calibration approaches

Reference	**Robot Name**	**Instruments used**	**Methods**	**Performance**	**Important points**
Guo *et al.* 2016 [34]	MLMTM	Wire extensometer (4)	Direct Kinematics & ADAMS Simulations	Used to measure flexible specimens in multiaxial loading situations. Coupled with ADAMS simulations to provide PID Control information for Force Control.	Integrated into the control loop to provide deflection information to an ADAMS model to provide data to a PID force control loop. The extensometers represent negligible stiffness in comparison to the test specimens.
Mura 2011 [35]	-	Wire extensometer (6)	Direct Kinematics	Used to measure flexible automobile components in dynamic loading conditions such as fatigue situations.	Extensometers continuously measure the platform position and represent a level of stiffness that is negligible compared to the measurement item.
Chiu *et al.* 2003 [62]	-	A cylindrical gauge block and a commercial trigger probe	Nonlinear least squares	Multiple PKM configurations were used to validate the calibration process and different accuracy levels have been obtained.	The instruments used here are standardized and commercialized. The method is comparatively compact and economical.
Zhuang *et al.* 2000 [63]	FAU Stewart Platform	CMM	Levenberg-Marquart algorithm	The average error was reduced by more than 50 percent.	The end-effector required separate calibration since it was not part of the closed-loop kinematic chains. It requires redundant sensors that need to be installed at some of the joints of the machine tool.
Patel *et al.* 2000 [10]	-	Ball-bar	Least square minimization	Simulations suggest that the measuring device accuracy needs to be five to ten times more than the desired calibration accuracy.	The extra actuators can be mounted or unmounted easily. When left with the machine in certain situations, it enables online calibration.

The research done by Patel *et al.* [10], observed that for their calibration algorithm increased error of the platform poses in some cases and for around 90% of observations, they got accuracy improvement from 50% to 100%. They used the least square method for their calibration algorithm. The advantage of their method is that the extra sensor is easily mountable and un-mountable; also, the extra sensor consisting of Ball-Bar can remain with the system during the actual machine operation to allow online calibration.

The details are presented in Table 2.5. It should be noted that for each case, platform pose error was considered.

2.5 Discussion and Comparison

The goal of each calibration method is to make the hexapod machines more accurate, improve precision and obtain correct results at the platform pose during their operations. Based on the literature survey done, it can be inferred that researchers predominantly opt for conducting experiments on their hexapod platforms to validate their calibration procedures. Among these experiments, the majority employed external measurement equipment. The key focus in these studies is the improvement of pose accuracy, primarily targeting platform pose errors, joint errors, and actuator length errors. The extent of error being addressed varies among these studies. In

Table 2.6, the crucial details from each research endeavor were summarized, aiding readers in identifying literature of interest based on the chosen approach, error considerations, methods employed, and the level of error addressed.

It's worth noting that the units used to measure errors differ across various studies, and some literature sources do not specify pose errors before the calibration process; they only report final values after calibration. Additionally, in certain instances, pose error is defined solely by position errors, with no consideration given to orientation errors. Remarkably, none of the experiments focus exclusively on improving accuracy through orientation-related factors.

Table 2.6: Comparison of experimental calibrations mentioned in this review.

Reference	**Calibration approach**	**Instruments used**	**Error considered**	**Methods**	**Pose Error before calibration**	**Pose error after calibration**
Song *et al.* 2022 [49]	External	FARO measuring arm	Joint errors	Artificial Neural Network (ANN) based non-linear functions	0.642 mm & 0.184 deg	0.175 mm & 0.060 deg
Mahmoodi *et al.* 2014 [50]	External	Rotary sensors	Actuator length	A new method	0.1 m & 0.1 rad	Reduced by 8-16 times
Jáuregui *et al.* 2013 [18]	External	Laser Interferometer	Actuator length	Two different methods based on error propagation calculation	0 – 60 μm	0 – 10 μm
Ren *et al.* 2009 [36]	External	Biaxial Inclinometer	Actuator length	A new orientation constraint method	9 mm & 1 deg	0.1 mm & 0.01 deg

Reference	Calibration approach	Instruments used	Error considered	Methods	Pose Error before calibration	Pose error after calibration
		& Laser tracker				
Nategh *et al.* 2009 [51]	External	Digital camera	Platform pose	Least-square approach based on Levenberg-Marquardt algorithm	2.67 mm & 4.6 deg	1.45 mm & 0.27 deg
Großmann *et al.* 2008 [52]	External	Double Ball Bar	Platform pose	Genetic Algorithm based Trajectory optimization.	0.7 mm	0.17 mm
Liu *et al.* 2007 [53]	External	3D Laser Tracker	Actuator lengths	Genetic Algorithm	0.7237 mm & 0.1346 deg	0.1555 mm & 0.0172 deg
Ting *et al.* 2007 [54]	External	DMT22 Dual Sensitivity Systems with C5 probe	Actuator lengths	Preisach model	-	1 µm & 10 µdeg
Daney *et al.* 2006 [17]	External	CCD camera.	Joint errors	Interval arithmetic and analysis methods	1.32 mm & 0.34 deg	1.10 mm & 0.26 deg
Dallej *et al.* 2006 [55]	External	Omni-directional camera	Joint errors	Linear regression	1.2 cm	~1 cm
Daney *et al.* 2005 [56]	External	Sony digital video camera	Platform pose	Constrained optimization method with Tabu search	-	1.45 mm & 0.27 deg
Gao *et al.* 2003 [57]	External	Laser Tracker LTD500	Platform pose	Least Square method	±2 mm	±1 mm
Renaud *et al.* 2002 [58]	External	A CCD camera & 1D laser interferometry	Platform pose	Error function minimization.	-	1 µm & 0.001 deg
Week *et al.* 2002 [59]	External	Double ball bar & CMM	Actuator length	Gravity compensation	-	Improvement (µm): Roundness 3.7x Squareness 7x
Ihara *et al.* 2000 [60]	External	Telescoping magnetic ball bar (DBB)	Actuator length and joint positions	Fourier transformation	Circle center error (-68, 164) µm	Circle center error (-7, -9) µm
Chiu *et al.* 2003 [62]	Self-calibration	Cylindrical gauge block & trigger probe	Platform joints	Nonlinear least squares	±1 µm	±0.001 µm
Zhuang *et al.* 2000 [63]	Self-calibration	CMM	Platform Joints	Levenberg-Marquart algorithm	1 µinch	0.1 µinch
Patel *et al.* 2000 [10]	Self-calibration	Ball Bar	Platform Pose	Least square minimization	±0.1 mm	3% to 100%
Mura 2011 [32]	Self-calibration	Wire Extensometers	Platform Pose	Direct Kinematics	±0.1 mm	±0.005 mm
Guo *et al.* 2016 [33]	Self-calibration	Wire Extensometers	Platform Pose	Direct Kinematics	±10 N, ±0.1 Nm	±0.1 N, ±0.005 Nm

As mentioned earlier, this pose accuracy is dependent on several mechanical and surrounding factors like temperature and load being experienced by the system. The ideal calibration would consider all these factors for any working condition of the machine but achieving that is not only expensive and time-consuming but also potentially unnecessary depending on the application conditions. In experiments where only the actuator lengths have been considered as the sources of error for the accuracy of the platform pose, it should be noted that the motion of the platform is dependent on all the joints which are moving to generate the motion. So, while calibration may include only the actuator length error, it also indirectly includes the error contributed by the joints. Even if all joints are not equipped with individual sensors, their error factors are indirectly accounted for in the calibration process. In general, when the platform pose accuracy is of primary interest, the calibration of it indirectly considers all the error factors existing in the hexapod system but adding appropriate compensation for each error factor becomes difficult unless they are correctly identified and accounted for in the calculations.

2.5.1 Error Identification

From the tables above, it is seen that the level of improvements obtained are varied. The error factors considered are also different in each study. In most cases, the platform pose error due to the actuator length errors remained common and was considered as the primary error contributor in the entire system. Also, considering the error of each actuator separately and equally impacts the overall accuracy of the system. In all these calibration processes, use of external instruments is the most common practice. External instruments such as Double Ball Bar (DBB), laser interferometer, biaxial inclinometer, laser tracker, telescopic magnetic ball bar, and wire extensometers were used. A separate study may be useful to ascertain the effectiveness of each of these instruments for the calibration of a hexapod by inverse kinematics.

Optical calibration methods have also gained importance in recent times. CCD camera, omnidirectional camera, digital video camera, and other image capturing devices were used to achieve an accuracy level in the range of 1.0 cm to 0.1 mm. The advantage of using optical methods is that the parallel robot system does not need modifications to accommodate the measurement equipment for calibration, also it is non-invasive and automatically records the event for future reference. In these cases, the quality of the optical systems and the associated analysis system configuration have a major contribution to the final accuracy attained.

2.5.2 Algorithms

Among the several different algorithms used for the calibration purpose, the least-square method based on the Levenberg-Marquardt algorithm was used most for both optical and non-optical calibration. The genetic algorithm had been used for a couple of studies, but they had a high running time to reach the optimized level and were not sensitive to measuring noises. Constrained optimization method and Tabu search, Algebraic Elimination, non-linear least square methods were used and all of them resulted in various levels of calibration accuracy. The use of quaternions has improved calculation efficiency, though the practical application of hexapod accuracy level achieved by this technique has yet to be evaluated. Different algorithms also yielded different results on the same system for the calibrations done by Daney *et al.* [46]. Therefore, the selection of calibration methods and algorithms plays a key role in obtaining the desired accuracy level for the parallel robot system and application. In all cases, the final platform pose is the guiding parameter to evaluate calibration outcome.

2.5.3 External factors

There are some inherent dimensional errors in the hexapod structure due to the dimensional tolerances of each component used in the fabrication. The cumulative effects of all these tolerances play a significant role in platform accuracy. Apart from these errors, other factors, hysteresis for instance, vary with the change of the operational characteristics. Normally, the effect of temperature is expected to be minimal for the parallel robot system unless the system experiences large temperature variations during its operation. From a practical point of view, such robot systems operate in a controlled environment unless they are deployed in special applications like large field telescope mounting. In these applications where exposure to fluctuating weather conditions is unavoidable, proper calibration factors for thermal deviation must be included.

Likewise, the elastic deformation error factor is not dominant in all cases. If the hexapod platform is subjected to high loads relative to its structure, a factor for elastic deflection is essential. Hexapod platforms reviewed in this chapter can carry loads up to 2000kg though none of them were subjected to such load during the calibration process. As such, the structure can undergo a substantial amount of elastic stress and that may lead to a notable amount of deflection to impact the accuracy of the robot. In these types of cases, the factor of elastic load cannot be ignored. There are potential research opportunities for evaluating the impact of load conditions on parallel robot systems and to add suitable compensation factors for further improving the platform pose accuracy.

2.6 References of standards

While research documents serve as valuable references to grasp the present research landscape, it remains incomplete without highlighting the essential international standards crucial for comprehending the hexapod calibration process. These standards not only provide guideline for practical applications but also establish pathways for industry-wide implementation. So, these are also an important source of reference for Stewart Platform calibration.

A list of some relevant standards is summarized below:

ISO 22958:2021 *Hexapod platforms for industrial applications -- Vocabulary and requirements*: This standard defines the terminology and requirements for hexapod platforms used in industrial applications.

ASTM F3251-10 *Standard Test Method for Measuring the Geometric Errors of Six-Degree-of-Freedom Robot Manipulators*: This standard provides a method for measuring the geometric errors of six-degree-of-freedom robot manipulators, including Stewart platforms. The method is based on the use of a laser tracker to measure the positions and orientations of a reference point on the robot's end-effector.

VDI/VDE guideline 2310-17 *Geometric product specifications (GPS) -- Coordinate systems for robot manipulators*: This guideline provides a set of recommendations for defining coordinate systems for robot manipulators. The recommendations are based on the ISO standard ISO 8434-1:2018 Coordinate systems for robots and manipulators. Part 1: Coordinate systems for manipulators.

ANSI/RIA R15.06-2020 *American National Standard for Industrial Robots and Robot Systems -- Safety Requirements*: This standard provides safety requirements for industrial robots and robot systems.

ISO 230 *Test Code for Machine Tools Package*: This standard provides test codes for the determination of positioning numerically controlled axes, controlled machine tools, thermal effects on machine tools and machines operating under no-load or finishing conditions.

ISO 8373:2021 - *Robotics – Vocabulary*: This document defines the terms used in relation to robotics.

ISO 9283:1998 - *Manipulating Industrial Robots - Performance Criteria And Related Test Methods*.

ISO/TR 13309:1995 - *Manipulating Industrial Robots - Informative Guide*: On Test Equipment And Metrology Methods Of Operation For Robot Performance Evaluation In Accordance With ISO 9283.

ISO 9787:2013 - *Robots And Robotic Devices - Coordinate Systems And Motion Nomenclatures*.

ISO 5725 - *Accuracy (Trueness And Precision) Of Measurement Methods And Results*: The series of Standards in this provide guidelines for the determination of repeatability and reproducibility of measurement processes. These are needed for identifying the precision of a Stewart platform's measurements.

Apart from the above list of standards, the following organization provides a respectable number of references which are extremely useful for Stewart Platform calibration process:

NIST (National Institute of Standards and Technology): An organization in United States of America provides a wealth of calibration and measurement standards and guidelines that can be referenced.

ASTM International (formerly known as American Society for Testing and Materials): Provides a range of standards related to calibration and testing in various industries that may be applicable for Stewart platform calibration.

IEEE (Institute of Electrical and Electronics Engineers): IEEE has standards related to robotics and automation, that are relevant for calibration in the context of robotics, including Stewart platforms.

2.7 Conclusions

Hexapods are used for precise, complex, and repeatable operations in a variety of applications. Small errors in the system can lead to a serious impact on pose accuracy. Therefore, calibration is a critical activity for any hexapod machine to be reliable in each application. There are several sources of errors that negatively impact the accuracy of the hexapod. Selecting an appropriate calibration approach depends on the purpose and constraints of the hexapod system. In some cases, external sensor calibration may be the best solution, while in other cases, constrained or auto-calibration may be a better approach. This article attempts to provide an overview of those methods and draw an outline of the current state of the art in this field and help other researchers to take appropriate notes for the desired methods.

Researchers have explored a variety of calibration methods, each exhibiting varying degrees of accuracy and complexity in implementation. The focus of this review is to identify the methods used to improve the positional accuracy of the hexapod platform or end-effector and the accuracy improvement obtained after calibration. Instruments and techniques utilized to compensate pose errors stemming from various inaccuracies were summarized. Individual researchers searching for an appropriate calibration approach should be cognizant of the level of calibration needed for a particular application and the resulting impact upon the complexity of implementation. Three types of calibration strategies are identified for physical systems: external, constraint, and auto- or self-calibration. Double ball bars, laser trackers, laser interferometers, and optical devices are some of the most used instruments for collecting positional data. The types of errors typically considered in these studies revolve around platform pose, joint variables and actuator length. In addition to the studies on physical systems, analytical-based studies were also discussed, and their specifics were summarized. In general, the authors agree with the sentiment of Merlet [31], who suggested that the emergence of image-based systems offers a great deal of potential improvement. Since the publication of this statement in 2006, significant improvements have been observed in the hardware and software supporting image-based calibration and use of manipulator-image-coordination (i.e., hand-eye-coordination) approaches in biological systems have witnessed great successes.

The literature reviewed in this article covers research conducted since the year 2000 focusing hexapod platforms and on calibration of platform-pose under no-load condition; however, in actual applications, these hexapods may be experiencing heavy working loads. The effect of working load propagates directly to the hexapod structure. Based on structural rigidity, the load causes elastic deformation which can affect the operational accuracy. Therefore, the response behavior and platform accuracy of a hexapod under the influence of working loads remains a subject for further studies.

3 FORWARD KINEMATIC SOLUTION

This chapter presents a method to generate feasible, unique forward-kinematic solutions for a general Stewart-Gough platform. This is done by using inverse kinematics to obtain valid workspace data and corresponding actuator lengths for the moving platform. For parallel kinematic machines, such as the Stewart-Gough platform, inverse kinematics are straight forward, but the forward kinematics are complex and generates multiple solutions due to the closed loop structure of the kinematic links. In this research, a simple iterative algorithm has been used employing modified Denavit-Hartenberg convention. The outcome is encouraging as this method generates a single feasible forward kinematic solution for each valid pose with the solved DH parameters and unlike earlier forward kinematics solutions, this unique solution does not need to be manually verified. Therefore, the forward kinematic solutions can be used directly for further calculations without the need for manual pose verification. This capability is essential for the six degree of freedom materials testing system developed by the authors in their laboratory. The developed system is aimed at characterizing additively manufactured materials under complex combined multiple loading conditions. The material characterization is done by enabling high precision force control on the moving platform via in situ calibration of the as-built kinematics of the Stewart Gough Platform.

3.1 Introduction

The Stewart-Gough platform is one of the most popular parallel kinematic machines (PKMs) [64]. Though there are PKMs with 3, 4, 5, 6 parallel links, 6 parallel linked stationary PKMs are termed as Stewart-Gough platform or Hexapod and are most versatile among the PKMs because of having six degrees of freedom available in a compact machine [65]. A typical hexapod consists of one fixed base plate and one movable plate or platform connected by six actuators. Each of the six parallel actuators generally provides one degree of freedom (DOF) to the machine. 6-DOFs are three translations along *x*, *y*, *z* axes and three rotations about *x*, *y*, *z* axes when a cartesian reference frame is attached to the movable platform center [66]. The hexapod considered here has each prismatic (P) actuator connected to the base with a universal (U) joint and the other end is connected to the moving platform by a spherical (U) joint. Such a hexapod or Stewart-Gough platform is designated as 6-UPS Parallel Kinematic Machine [13]. The movement of the platform in 6-DOF through the movement of six actuators makes the motion control complex compared to other machines.

To provide position and motion control, hexapod platforms use either Inverse Kinematics or Forward Kinematics. The position and orientation of the platform center point is known as the *platform pose* [67]. In inverse kinematics the actuator lengths are calculated based on the platform pose [16]. In forward kinematics the platform pose is calculated for a given set of the actuator lengths and joint angles [68]. The mathematics and solution of inverse kinematics is much easier than forward kinematics for a hexapod. The complexity of forward kinematics, also called direct kinematics, is generated due to highly non-linear kinematic equations with multiple solutions. To solve the forward kinematics, many researchers tried diverse ways to solve the non-linear problem. A 16th-degree univariate polynomial on the 6-3 type PKM was formulated by Innocenti and Parenti-Castelli [69]. Huang, Xiguang, Liao, Qizheng, at el. [70] presented algebraic method for a general 6-6 Stewart-Gough platform that yielded a 20th degree univariate polynomial from the determinant of the 15×15 Sylvester’s matrix. Husty [71] derived a 40th-degree univariate equation for a general 6-6 Stewart-Gough platform, by finding the greatest common divisor of the intermediate polynomials. Domagoj & Leonardo Jelenkovicti [68] used canonical formulation for the forward kinematics and derived 9 equation with 9 unknowns and then solved it by multiple optimization methods. Wang, Yunfeng [72] derived the direct kinematics solutions for calculating the platform pose by increasing the actuator lengths in small amount and then increasing the joint parameters also in small amount utilizing numerical methods. Manuel Cardona [73] calculated the possible poses using Newton-Raphson Method for a set of joint angles and actuator lengths which includes invalid and valid poses for the platform and those need to be checked manually for acceptance. Also, other researchers [74] [75] used Newton-Rapson method to find the forward kinematic solution for parallel robots. The accuracy of convergence obtained in those works are at various levels based on the initial guesses. X. Zhou *et al.* [76] created a pose error model with the help of Denavit-Hartenberg (DH) parameters and then converted it to a constrained quadratic optimization problem. In another research by M. Tarokh [77] used an approach to generate a lookup-table with possible solution space data. This solution space is divided into multiple clusters. For forward kinematic solution, the system looks at lookup-table clusters to get the required data directly or from the fitted curves with the available.

Other research on forward kinematics for Stewart-Gough platform used algebraic elimination [78], interval analysis, multiple optimization techniques, continuation algebraic formulations to generate solutions for a set of nonlinear equations or high degree of polynomials. Some researchers utilized neural network algorithms [79]–[82] and Artificial Intelligence (AI) [83] for improving the accuracy of the hexapod platform solving forward kinematics problem. All these works generated algorithms to obtain a valid solution for several types of PKMs, but finding a single feasible practical solution is still a challenging problem and limited for real-time applications.

The proposed method in this article uses inverse kinematics to solve forward kinematics using modified Denavit-Hartenberg (DH) convention. DH convention is the most popular method for forward kinematics in serial manipulators, but its application remains limited in parallel manipulators due to the closed loop nature of PKMs. The authors adopted this convention because of its simplicity and straight-forward nature of implementation. The proposed algorithm in this article generates a single feasible orientation solution for a general Stewart-Gough platform calculated by forward kinematics from a set of input data. The solution does not need to be validated through manual inspection. This is a simple iterative method that uses the available information from inverse kinematics. The pose data and the corresponding actuator lengths are stored in a database. Then based on the motion limits of each joint, the algorithm generates the DH parameter dataset consisting of joint angles for the platform pose through iterative forward kinematics. The authors tried to exploit the power of the latest generation of computing systems by using a simple iterative method, which is not significantly more time-consuming method in the present days than other efficient optimization methods like Newton-Rapson method, etc. Another advantage of this simple iterative method is that there is no initial guess and no doubt about obtaining a solution (convergence). As long as a pose exists, a solution must be available.

The rest of the article is organized into the following parts. The first part serves as an introduction. The next part elaborates the general mathematical expressions for a Stewart-Gough platform. Section 3 and 4 explain the inverse kinematics and forward kinematics used in the calculations. Section 5 explains the method that has been used in the algorithm to simulate the desired results for a Stewart-Gough platform-based Test frame “Tiger 66.1”. Section 6 contains the result and discussion and finally the piece ends with the conclusion.

3.2 Platform Poses & Workspace

Figure 3.1 shows the typical sketch of a hexapod platform. The circular plates at bottom and top are Fixed base and moving platform, respectively. Two cartesian coordinate frames are attached at the center of each circular plate. The platform frame is defined by $\boldsymbol{P_x}$, $\boldsymbol{P_y}$ and $\boldsymbol{P_z}$ axes with origin $\boldsymbol{O_P}$ and the base coordinate frame is expressed by $\boldsymbol{B_x}$, $\boldsymbol{B_y}$, and $\boldsymbol{B_z}$ axes with origin $\boldsymbol{O_B}$. $\boldsymbol{P_z}$ and $\boldsymbol{B_z}$ are denoting the vertical axes of the respective frames. The orientation of the moving platform is defined by the orientation of the $\boldsymbol{O_P P_x P_y P_z}$ frame with respect to the base coordinate frame $\boldsymbol{O_B B_x B_y B_z}$. The position of the platform center with respect to the base center is defined by vector $\boldsymbol{h}$. In one condition, the base and platform frames remain parallel, the actuators are at their smallest length, and the z-axis are colinear. This orientation is called '*home pose*' [84]. Under this condition, all the six actuators are of the same length. Once the platform is moved by controlling the lengths of the actuators, the resultant motion at the center of the platform is a translation, rotation, or a combination of both.

The position and orientation of the platform center depends on the values of roll, pitch, yaw, and the translation motion along x, y, z axes as per the Euler angle representations [85]. The rotations are expressed by vector Φ and

$$\Phi = (\alpha\ \beta\ \gamma)^T \tag{3.1}$$

where α (roll), β (pitch), γ (yaw) denotes the rotation angles about the x, y, and z axes, respectively.

The translations are expressed by vector $\boldsymbol{d}$ where

$$\boldsymbol{d} = (dx\ dy\ dz)^T \tag{3.2}$$

dx, dy, and dz are the translation values along the x, y, and z axes, respectively.

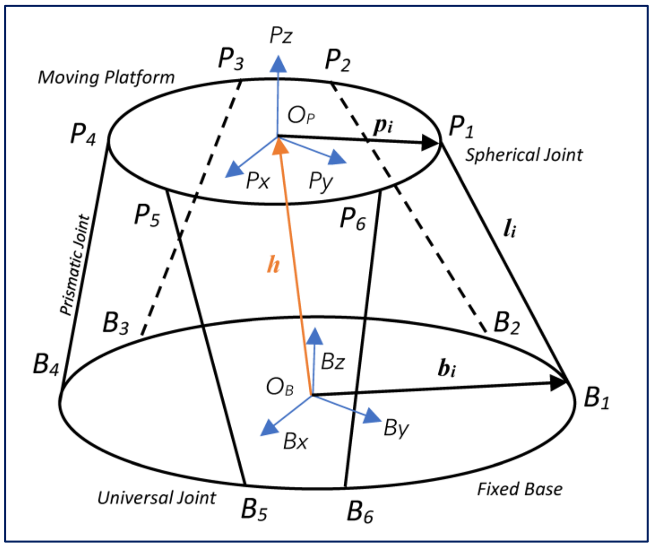


Figure 3.1: A typical hexapod configuration

The relationships between different platform poses and actuator variables are expressed by forward kinematics and inverse kinematics. In forward kinematics, the platform poses are calculated by the length and orientation of the six actuators. It can be expressed by equation (3.3):

$$[dx, dy, dz, \alpha, \beta, \gamma]^T = f(\boldsymbol{q_1}, \boldsymbol{q_2}, \boldsymbol{q_3}, \ldots \ldots, \boldsymbol{q_n}) \tag{3.3}$$

where $dx, dy, dz, \alpha, \beta, \gamma$ are platform pose and $\boldsymbol{q_1}, \boldsymbol{q_2}, \boldsymbol{q_3} \ldots \boldsymbol{q_n}$ are link variables that include joint angles and actuator lengths.

In inverse kinematics, actuator lengths are calculated for a platform pose by Equation (3.4).

$$\boldsymbol{q_i} = f_i(dx, dy, dz, \alpha, \beta, \gamma) \tag{3.4}$$

where $i = 1 \ldots n$ are link numbers. For a Stewart-Gough platform, the value for i = 1 to 6

Mathematically, the rotations about each axis are represented by the following equations as per the Euler angle representation:

$$R_{x,\alpha} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\alpha & -s\alpha \\ 0 & s\alpha & c\alpha \end{bmatrix} \tag{3.5}$$

$$R_{y,\beta} = \begin{bmatrix} c\beta & 0 & s\beta \\ 0 & 1 & 0 \\ -s\beta & 0 & c\beta \end{bmatrix} \tag{3.6}$$

$$R_{z,\gamma} = \begin{bmatrix} c\gamma & -s\gamma & 0 \\ s\gamma & c\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

Here ***s*** and ***c*** represent ***sine*** and ***cosine*** functions, respectively.

Combining all these rotations, the complete rotation of the platform center with respect to its original fixed frame axes is calculated. The combined rotation denoted by $\boldsymbol{R}$ is calculated by pre-multiplying each subsequent rotation.

$$\boldsymbol{R} = R_{z,\gamma}.R_{y,\beta}.R_{x,\alpha}$$

$$= \begin{bmatrix} c\beta c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ c\beta s\gamma & s\alpha s\beta s\gamma + c\alpha c\gamma & c\alpha s\beta s\gamma - s\alpha c\gamma \\ -s\beta & s\alpha c\beta & c\alpha c\beta \end{bmatrix} \tag{3.8}$$

Equation (3.8) represents the final rotation matrix. This is further combined with the *x, y*, and *z* translations, and the Homogeneous Transformation Matrix (HTM) [63] in Equation (3.9) is obtained. The complete HTM is expressed as

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{d} \\ 0 & 1 \end{pmatrix}$$

$$\therefore \ \boldsymbol{H} = \begin{bmatrix} c\beta c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma & dx \\ c\beta s\gamma & s\alpha s\beta s\gamma + c\alpha c\gamma & c\alpha s\beta s\gamma - s\alpha c\gamma & dy \\ -s\beta & s\alpha c\beta & c\alpha c\beta & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

3.3 Inverse Kinematics & workspace

The platform pose with respect to the base center defines the state of the system. In inverse kinematics, the actuator lengths $\boldsymbol{l_i}$ for a particular platform pose are calculated. With the change of vectors $\boldsymbol{\Phi}$ and $\boldsymbol{d}$, the coordinates of the platform center $\boldsymbol{O_P}$, platform joint vectors $\boldsymbol{p_i}$, and vectors $\boldsymbol{h}$, $\boldsymbol{l_i}$ change. As the base is fixed, the base center vector $\boldsymbol{O_B}$, base joint vectors $\boldsymbol{b_i}$ remain unchanged.

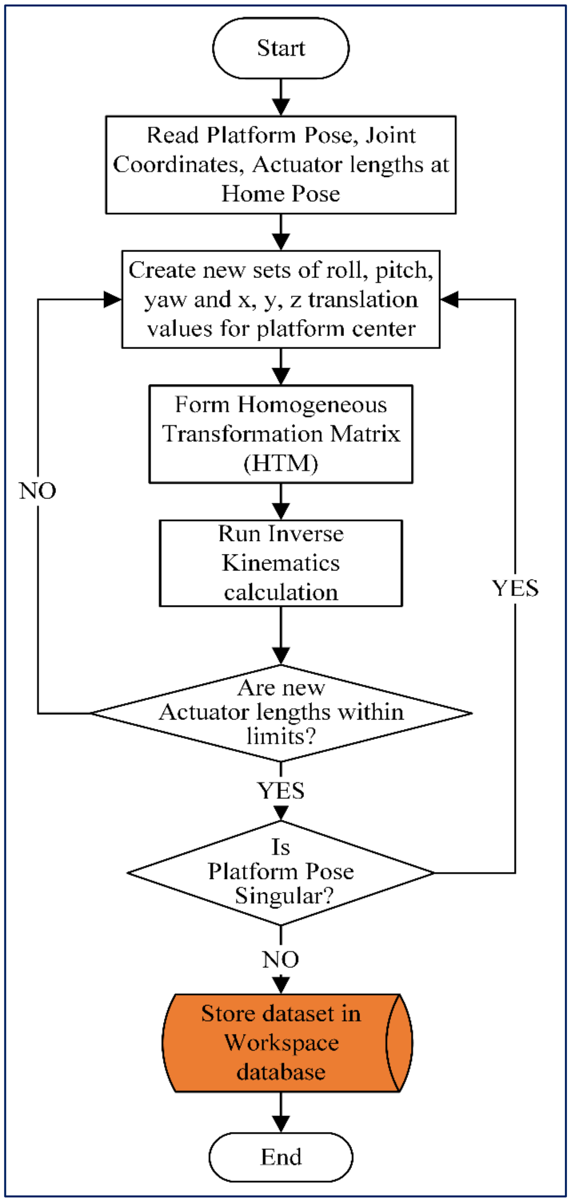

Start
Read Platform Pose, Joint Coordinates, Actuator lengths at Home Pose
Create new sets of roll, pitch, yaw and x, y, z translation values for platform center
Form Homogeneous Transformation Matrix (HTM)
Run Inverse Kinematics calculation
Are new Actuator lengths within limits?
NO
YES
Is Platform Pose Singular?
YES
NO
Store dataset in Workspace database
End

Figure 3.2: Flowchart for Workspace calculation by Inverse Kinematics

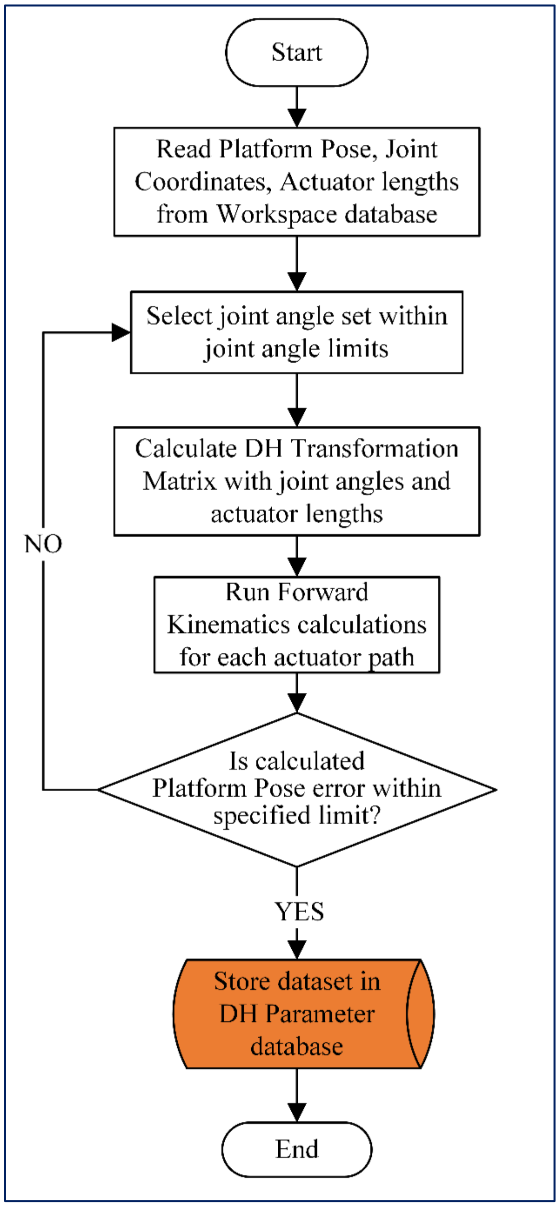

Start
Read Platform Pose, Joint Coordinates, Actuator lengths from Workspace database
Select joint angle set within joint angle limits
Calculate DH Transformation Matrix with joint angles and actuator lengths
Run Forward Kinematics calculations for each actuator path
Is calculated Platform Pose error within specified limit?
NO
YES
Store dataset in DH Parameter database
End

Figure 3.3: Flowchart for DH parameters calculation by Forward Kinematics

The new values of $\boldsymbol{l_i}$ and $\boldsymbol{h}$ are calculated by calculating new values of $\boldsymbol{p_i}$ and $\boldsymbol{O_P}$ by finding value of $\boldsymbol{H}$ in Equation (3.9) and pre-multiply to the respective coordinate values before the change. Then the length of each actuator is obtained by using Equation (3.10) [14]

$$\boldsymbol{l_i} = ||\boldsymbol{h} + {}^{B}\boldsymbol{R_P}.{}^{B}\boldsymbol{p_i} - \boldsymbol{b_i}\,|| \qquad (3.10)$$

where ${}^{B}\boldsymbol{R_P}$ denotes the rotation vector to express the rotation of the platform coordinate frame with respect to the base coordinate frame, and ${}^{B}\boldsymbol{p}_{\boldsymbol{i}}$ is the platform joint vectors expressed with respect to the base coordinate frame.

For each PKM, the actuators operate between fixed length limits. Conforming to the values of $\boldsymbol{l}_{\boldsymbol{i}}$ for each new pose obtained by changing vectors $\boldsymbol{\Phi}$ and $\boldsymbol{d}$ define the feasible movable points for the platform center. All these feasible points form the *working space* for the selected hexapod platform.

Figure 3.2 shows the flowchart of the inverse kinematics calculations used by the authors to generate the valid workspace points. An important aspect of the platform pose is *singularity*. In singular condition, the end-effector gains one or more unwanted instantaneous degrees of freedom (DOF), and the platform pose cannot be determined by unique actuator lengths. In such situation, the PKM becomes out of control and can transitorily make the drive force go infinity [86]. A pose with singularity is not considered as a valid workspace point and discarded from further calculations. The mathematical check for singularity is included inside the calculation code by checking if the determinant of the force Jacobian matrix in that pose is zero or not [87].

3.4 Forward Kinematics and valid poses

The Denavit-Hartenberg (DH) convention [88] is one of the most popular and earliest methods for solving forward kinematics for any serial manipulator in 3D space. For each pair of links, there are four DH parameters that are used in the DH matrix to transfer the coordinate of a point from one coordinate frame to another.

The initial DH convention was introduced in 1955 by Jacques Denavit and Richard Hartenberg. In due course of time, it has been revisited by researchers and a modified DH convention [89] has been introduced. In this article, modified DH convention has been used. The 4 DH parameters, as per modified conventions, are

Table 3.1: DH Parameters descriptions

Modified DH Parameters				
Link (*i*)	$\boldsymbol{a_{i-1}}$	$\boldsymbol{\alpha_{i-1}}$	$\boldsymbol{r_i}$	$\boldsymbol{\theta_i}$
Parameter name	Link length	Link twist	Link offset	Joint angle

Here $\boldsymbol{i}$ denotes the joint in consideration and $(\boldsymbol{i-1})$ is the previous joint. $\boldsymbol{i}$ is always a positive integer $^{\boldsymbol{i-1}}\boldsymbol{T_i}$ describes the transformation matrix for frame $\boldsymbol{i}$ relative to frame $(\boldsymbol{i-1})$. Using these DH parameters, the frame transformation matrix is calculated using Equation (3.11) and (3.12) [90].

$$^{i-1}\boldsymbol{T}_i = R_x(\alpha_{i-1}).\boldsymbol{T}_x(a_{i-1}).R_z(\theta_i).\boldsymbol{T}_z(r_i) \tag{3.11}$$

$$\textbf{Or}, {}^{i-1}\boldsymbol{T}_i = \begin{bmatrix} cos\theta_i & -sin\theta_i & 0 & a_{i-1} \\ sin\theta_i cos\alpha_{i-1} & cos\theta_i cos\alpha_{i-1} & -sin\alpha_{i-1} & -r_i sin\alpha_{i-1} \\ sin\theta_i sin\alpha_{i-1} & cos\theta_i sin\alpha_{i-1} & cos\alpha_{i-1} & r_i cos\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.12}$$

If there are, suppose, 4 joints, they are numbered from 0 to 3. A point $\boldsymbol{P}$ is expressed with respect to the end coordinate frame {3} as $^3\boldsymbol{P} = [P_x\, P_y\, P_z]^T$. The transformation matrix to express frame {3} with respect to the base frame {0} is calculated by Equation (3.13).

$$^0\boldsymbol{T}_3 = {}^0\boldsymbol{T}_1.{}^1\boldsymbol{T}_2.{}^2\boldsymbol{T}_3 \tag{3.13}$$

Point ^3P with respect to the base frame {0} is calculated Equation (3.14).

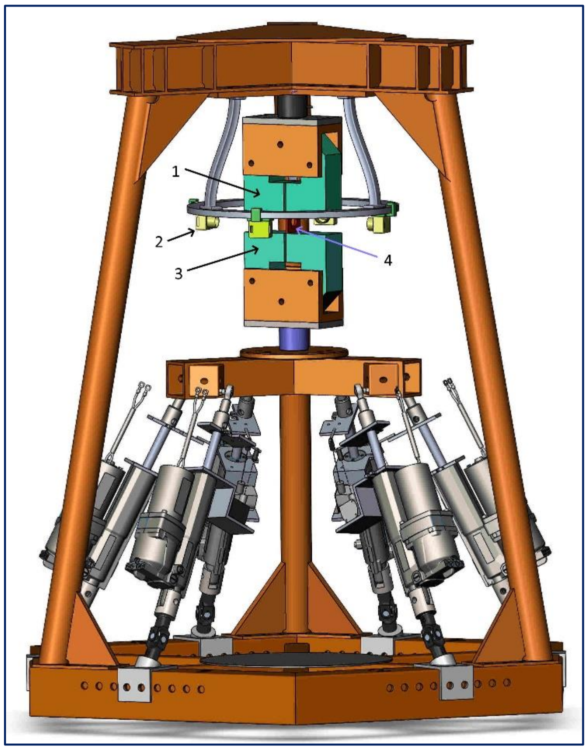
$$^0\boldsymbol{P} = {}^0\boldsymbol{T}_3.{}^3\boldsymbol{P} \tag{3.14}$$

In the workspace calculation by inverse kinematics, the valid actuator lengths are calculated for each valid pose of the platform and these data are stored in the workspace database. For calculating the DH parameters, the MATLAB code follows the flowchart shown in Figure 3.3. The code iterates different combinations of joint angles $\boldsymbol{\theta_i}$ to a reach to the pose coordinate using the corresponding actuator lengths available in the workspace database. The angles $\boldsymbol{\alpha_{i-1}}$ are defined by the cartesian frame attached at each joints following modified DH convention rules and these values are not going to change for the whole process. A predefined error value is set in the calculation for the pose error which measures the distance between the pose in the database and pose calculated through forward kinematics. The forward kinematics calculation is repeated several times by changing $\boldsymbol{\theta_i}$. Once the pose error is achieved below the predefined value, the DH parameter sets are stored in the DH parameter database for the pose. In this process there are some instances when the angle combinations cannot construct an orientation with a pose error below the defined error value, that pose is recorded without a valid DH parameter set.

3.5 Simulation results for "TIGER 66.1"

The above method of DH parameter calculation has been tried out through simulation for the Hexapod Test Frame "Tiger 66.1" developed by the authors in their lab. Tiger 66.1 is a special Stewart-Gough platform-based test frame developed for full-field characterization of additively manufactured specimens. A CAD model of the test frame is shown in Figure 3.4. A partial sketch also shows a couple of critical dimensions. The test process uses photogrammetry, so there are four cameras (2) mounted to capture data from the test zone. This restricts the movements of the platform center and overall workspace. The two green blocks on the upper part of the system are the grippers for holding the material specimen (4) to be tested. The upper gripper (1) is mounted on the fixed frame and the lower gripper (3) is fixed at the center of the hexapod top moving platform. All motions and forces are applied on the test specimen by moving the lower gripper.

Tiger 66.1 uses 6-UPS link combinations. As per the design and construction, the base frame center is situated 71.55 mm about the World frame origin which is located on the center of the circular plate at the lower structure and platform home pose is considered at a point 343.20 mm above the actual platform center point. This dimension does not change. The test frame lower and upper grip centers are considered at this point which is the condition of the system at the beginning of any test. All the calculations were done between the world frame center and this grip center pose. The major dimensions of Tiger 66.1 are shown in Table 3.2.



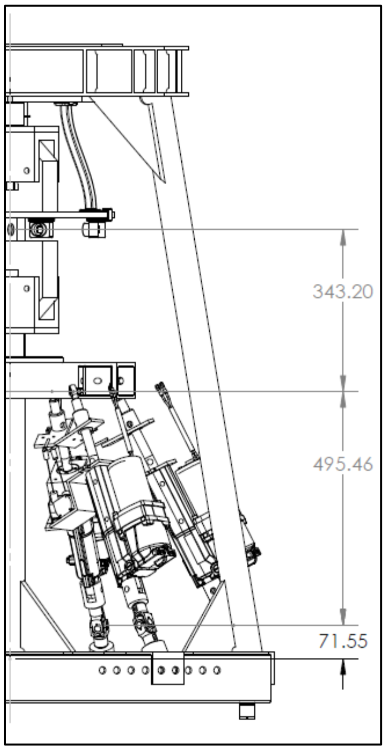


Figure 3.4: CAD model of hexapod Test Frame "Tiger 66.1" with partial sketch

The workspace for Tiger 66.1 has been calculated based on the actuator stroke length limit, translation, and rotation limits of the platform along and about x, y, z axes, respectively. The limits are given (Table 3.3) as per the feasible test conditions. All these motions limits are for pure translation and pure rotations.

The workspace for the test frame is plotted and shown in Figure 3.5. Under given machine limits, the algorithm generated more than 180,000 valid workspace poses. This value may increase or decrease depending on the increment values used for platform's $dx, dy, dz, \alpha, \beta, \gamma$ parameters. The interval steps used in this case for translations along x & y axes = 15 mm, z-axis = 10 mm and +/-10° for rotations about each axis.

Table 3.2: Major dimensions of Tiger 66.1

Sl no.	Parameters	Dimensions
1	Base joint circle radius	477.4 mm
2	Smaller sides of the base	377.9 mm
3	Larger sides of the base	570.4 mm
4	Platform joint circle radius	225.1 mm
5	Smaller sides of platform	178.8 mm
6	Larger sides of platform	268.7 mm
7	Actuator stroke length	203.0 mm
8	Base center to Platform center height at test start	495.46 mm

Table 3.3: Motion limits of Tiger 66.1

Motion no.	Parameters	Limits
1	Rotation about x-axis	+/- 30°
2	Rotation about y-axis	
3	Rotation about z-axis	+/- 55°
4	Translation along x-axis	+/- 158 mm
5	Translation along y-axis	
6	Translation along z-axis	50 mm

In the next part of the calculation, a random 100 poses have been selected (Figure 3.6) for finding the DH parameters for those poses through forward kinematics using DH transformation matrix.

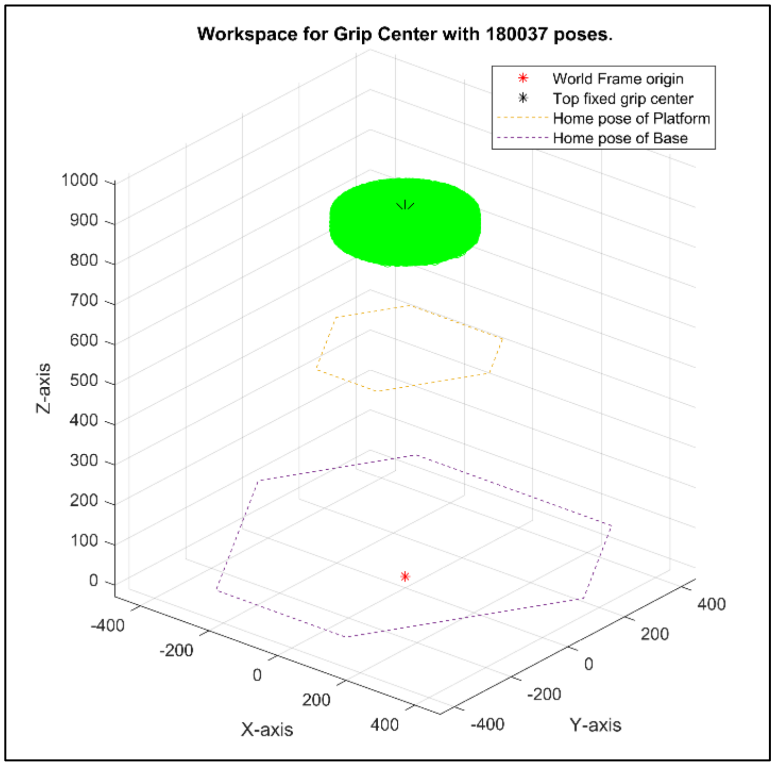


Figure 3.5: Graphical representation of Workspace

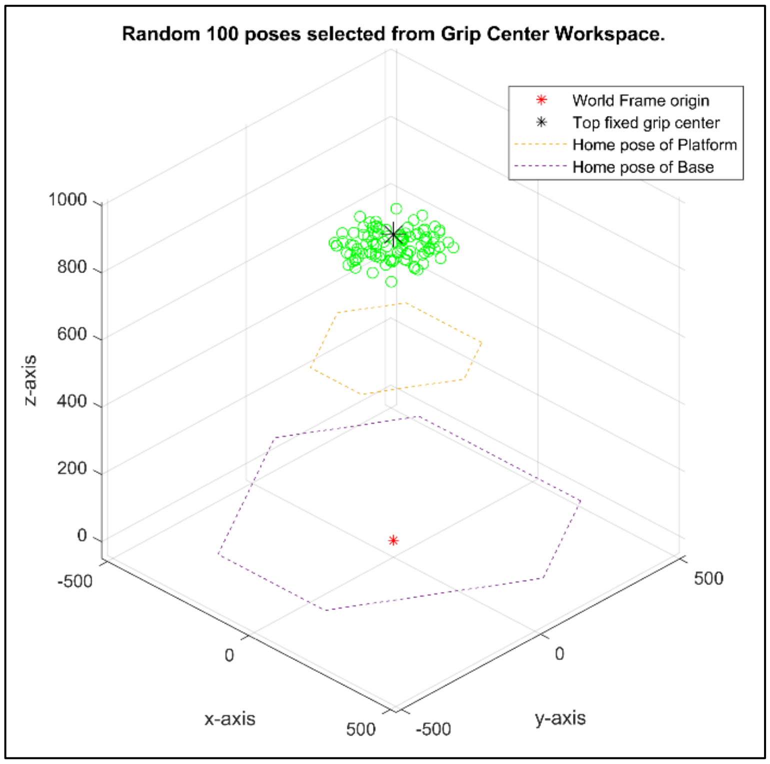


Figure 3.6: Selected 100 random poses from workspace

To use the DH convention, coordinate frames have been assigned to each joint and a DH frame layout has been created as shown in Figure 3.7.

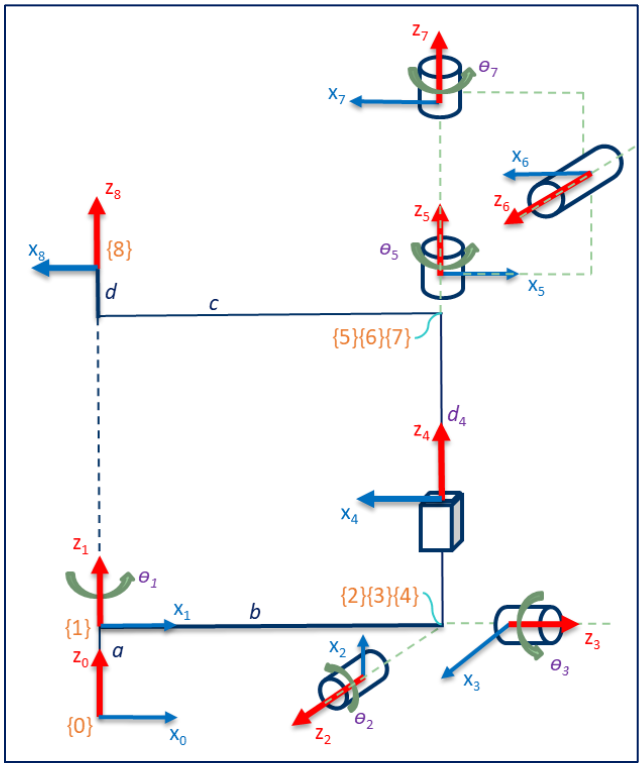


Figure 3.7: DH Frame layout for each actuator path

The layout shows the frames and variables assigned for one actuator path from the world coordinate frame {0} to the moving grip center frame {8} on the platform. The coordinate frames were assigned as per the right-hand rule and the modified DH convention. Frame {1} is the base center, frames {2} & {3} represent the universal joint at the bottom of each actuator, frame {4} denotes the prismatic joint on the actuator, and frames {5}, {6}, {7} represent the spherical joint at the top end of each actuator. As per construction, variables *a*, *b*, *c*, and *d* remain constant and represent the height of the base center, base joint radius, platform joint radius, and grip center height from the platform center, respectively.

Table 3.4 lists the DH parameters for each actuator path. All joint angles are varying except $\boldsymbol{\theta_1}$. The value of $\boldsymbol{\theta_1}$ for each actuator path remains fixed due to the construction of Tiger 66.1.

Table 3.4: Modified DH parameter table

Link (i)	a_{i-1}	α_{i-1}	r_i	θ_i
1	0	0	a	θ_1 *
2	b	π/2	0	θ_2
3	0	π/2	0	θ_3
4	0	π/2	0	$-\pi/2$
5	0	0	d4	θ_5
6	0	π/2	0	θ_6
7	0	π/2	0	θ_7
8	c	0	d	0
* θ_1 is predefined by construction for Tiger 66.1, and the values are 53.4°, 126.6°, 173.4°, 246.6°, 293.4°, 366.6°				

The iterations found valid DH parameter sets for 100 poses out of 100 valid poses. To check the success rate of the algorithm, a random 100 poses were selected for 10 calculation processes and success rate of finding DH parameter sets is 100% for all processes. Table 3.5 shows the DH parameters value ranges as plotted in Figure 8 and the success of finding DH parameters for the number of poses considered.

Table 3.5: DH parameters value ranges and results obtained

Variable	**From**	**to**
θ_2	-83.2°	-38.0°
θ_3	-118.8°	-63.4°
θ_5	90.0°	270.0°
θ_6	92.0°	231.0°
θ_7	-90.0°	90.0°
d_4	465.68 mm	664.68 mm
Number of poses evaluated		100
Number of poses with valid DH parameters		100
Number of poses with Grip Center deviation < 1mm		100

The DH parameter angles and actuator lengths variations for all 6 actuators for 100 poses are shown in Figure 3.8.

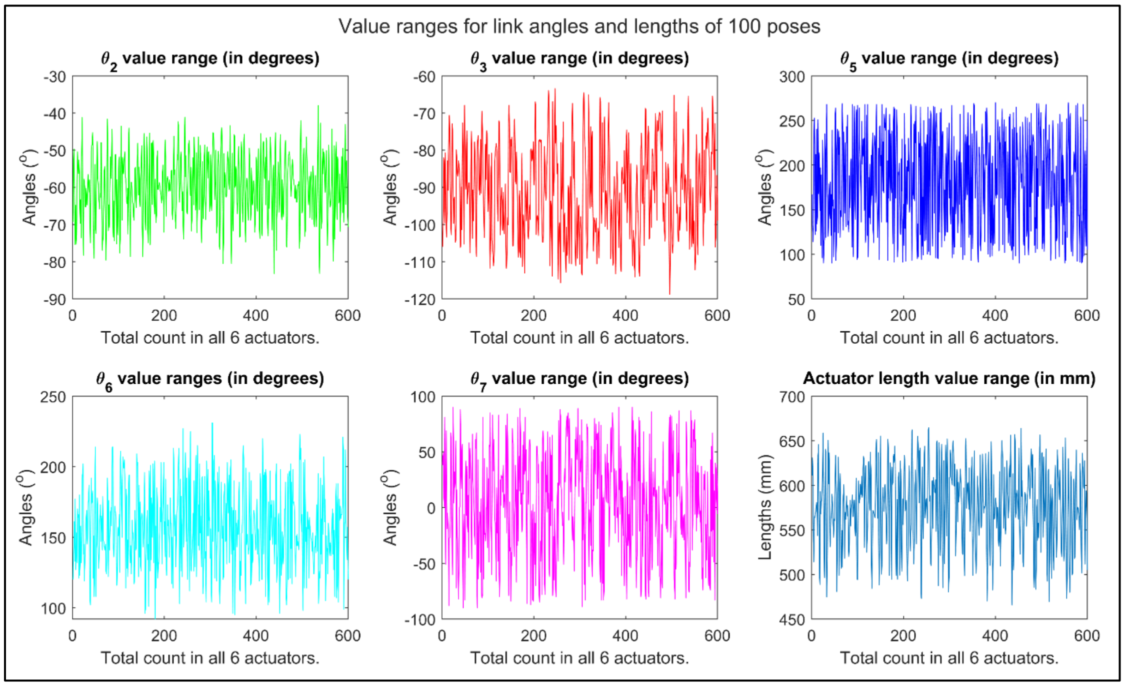


Figure 3.8: Angles and actuator length ranges

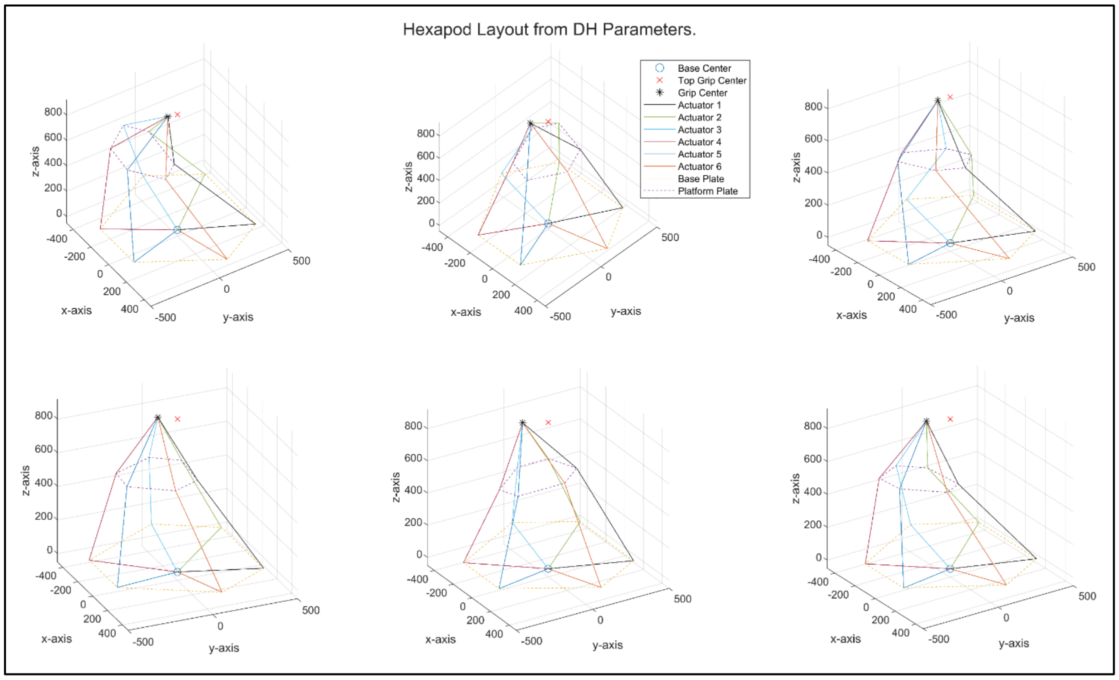


Figure 3.9: Tiger 66.1 orientation layouts through the calculated DH parameters

The orientations of Tiger 66.1 were drawn based on the DH parameters obtained through the simulation. Figure 3.9 shows random 6 numbers of such orientation layout and it is observed that all the orientations are valid and feasible. It has been verified that all DH parameter sets generate unique valid feasible poses every time; however only some of them are shown here due to space limitations.

3.6 Results and discussion

The inverse kinematic and forward kinematic calculations run for Tiger 66.1 in this simulation shows that the unique solutions are achievable using the algorithm. This algorithm does not use any complex calculations and finds solutions through iterations. The pose data points were obtained for the workspace by changing the values of roll, pitch, yaw and translations along *x*, *y*, *z* in each iteration. The number of data points is dependent on the increment step value of each variable. A smaller increment in the value will generate a greater number of poses for the workspace. But the total time of calculation will increase because of an increased number of iterations.

The frame assignments from world frame to grip center frame at the joints through each actuator path can be done in different ways and the initial parameters values may be different. This depends on whether the modified or standard DH conventions are used, but the result will be the same because both DH transformation matrices generate the same unique solutions. For finding the DH parameters for a pose, the variable values are changed by predefined steps in each iteration. The finer the increment steps are, the more accuracy level is achieved, but the calculation time will increase. In the current simulation, DH parameters for 100% valid poses have been found with pose accuracy level < 1 mm. For these iterations, the error limit for moving grip center pose was set as < 1 mm and the angle finding steps used between 0.05° to 1°. In a standard standalone laptop with Intel(R) Core (TM) i7-8550U CPU @1.80GHz and 16GB physical memory, it took an average of 32.09 seconds to solve the DH parameters for one pose. In any multicore modern server, this solution time can be significantly reduced. Trial showed that in a 56-core server, the execution time comes down to 1.28 seconds to get the same solution.

The orientations sketched in Figure 3.9 from the solutions found through forward kinematics indicate that they are feasible and unique. The forward kinematics solutions done earlier by the researchers yielded more than one solution for one pose. Those solutions need to be inspected one by one and validated for the acceptable one. The forward kinematic results from the method explained in this article do not require any manual review to check their feasibility. This gives an option to use the DH parameters for any further calculation without any intermediate stop and manual intervention for the selection of correct results. In section 6.1, one such exercise is executed which is important for the use of Tiger 66.1 in characterization of additively manufactured materials.

3.6.1 Pose deviation due to tolerances & Sensitivity analysis

A real-world Stewart-Gough platform is not free from manufacturing and assembly tolerances. These errors cause the actual platform pose to deviate from the theoretical platform pose for a set of DH parameters. The actual measurement of those DH parameter deviations is not only difficult and time consuming, but also expensive due to proper instrumentation. With various combinations of DH parameter tolerances, a tentative pose deviation can be calculated with this algorithm.

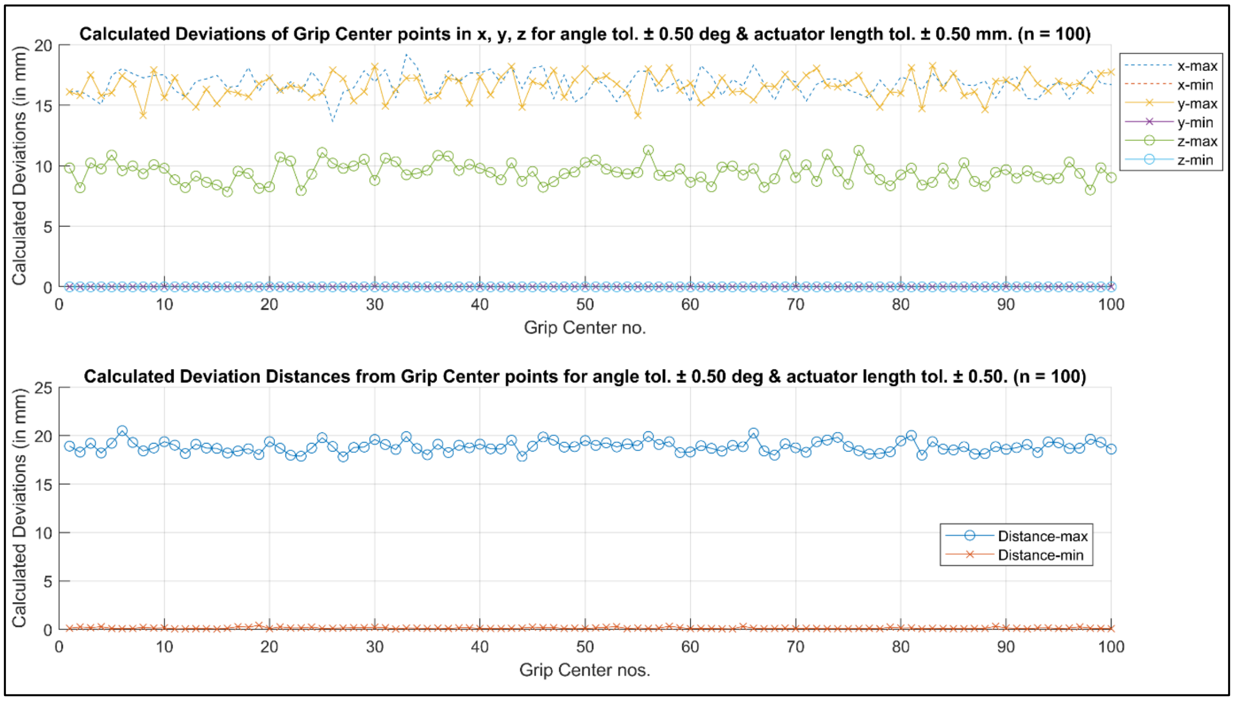


Figure 3.10: Platform pose deviations for DH parameters tolerances of ±0.5 deg & mm

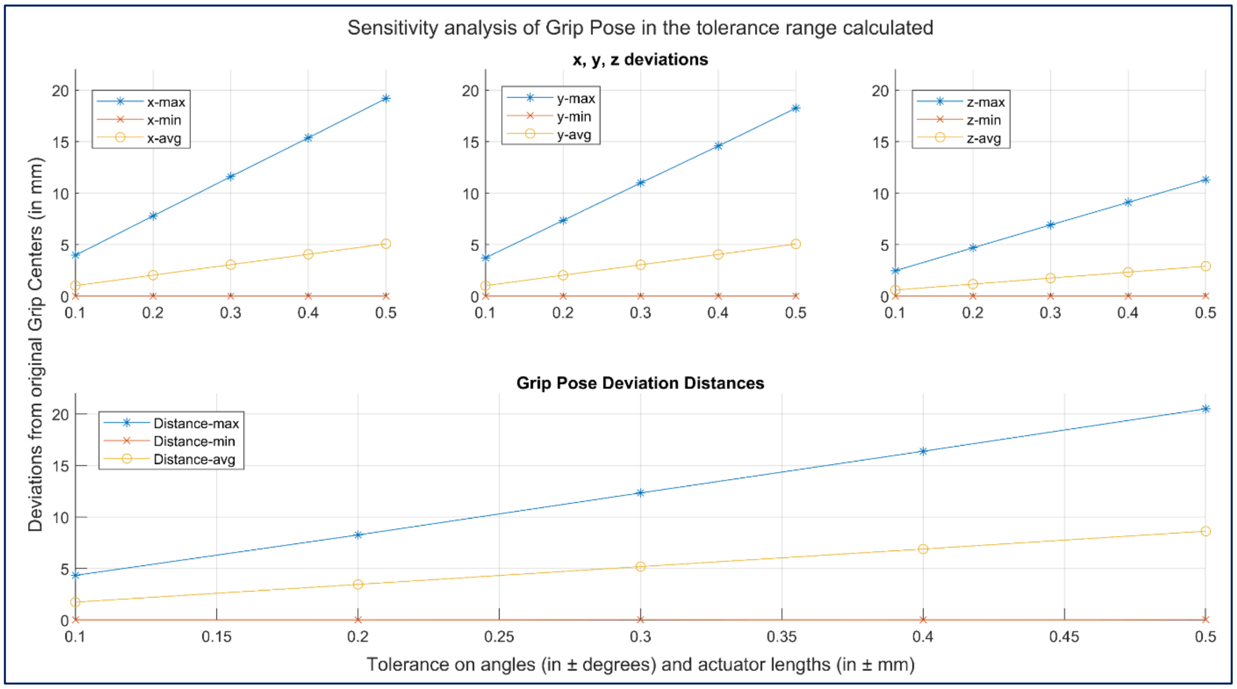


Figure 3.11: Sensitivity of Platform poses due to DH parameters tolerances

Five tolerance values were considered for calculating the pose error. These values are ±0.1, ±0.2, ±0.3, ±0.4 and ±0.5 in degrees for angles and in mm for the actuator lengths. These tolerances were applied to the DH parameters for 100 poses found through the calculations as discussed in the previous section. For ±0.5 deg & mm tolerances on the DH parameters, the platform-pose errors in terms of absolute *x, y, z* values and absolute distance values from the theoretical poses are shown in Figure 3.10. The 0 marked horizontal lines in both the plots indicate the theoretical values.

Similar calculations were done with four other tolerance values. From calculated data the maximum and minimum deviations were plotted to check the sensitivity of the platform poses due to the DH parameters tolerances (Figure 3.11). The maximum and average deviations for each tolerance are shown. In all these cases, the Grip pose denotes the lower moving grip center mounted on the moving platform.

Table 3.6: Pose deviations corresponding to *x, y, z* maximum and minimum values

Tol. value	At	x-deviation	y-deviation	z-deviation	Grip Center deviation	At	x-deviation	y-deviation	z-deviation	Grip Center deviation
±0.1	x-max	**3.96**	0.32	1.01	**4.10**	x-min	**0.00**	2.23	1.10	2.49
	y-max	0.35	**3.72**	0.42	3.76	y-min	0.43	**0.00**	0.14	**0.45**
	z-max	1.75	1.99	**2.47**	3.62	z-min	0.63	1.96	**0.00**	2.06
±0.2	x-max	**7.78**	0.79	1.95	**8.06**	x-min	**0.00**	1.18	0.60	1.33
	y-max	1.59	**7.35**	1.40	7.65	y-min	1.82	**0.00**	0.52	1.89
	z-max	4.80	3.32	**4.69**	7.49	z-min	0.96	0.38	**0.00**	**1.03**
±0.3	x-max	**11.60**	1.28	2.88	**12.02**	x-min	**0.00**	6.59	1.68	6.81
	y-max	2.39	**10.99**	2.05	11.43	y-min	1.82	**0.00**	0.52	1.89
	z-max	4.80	3.32	**4.69**	7.49	z-min	0.96	0.38	**0.00**	**1.03**
±0.4	x-max	**15.37**	1.79	3.79	**15.93**	x-min	**0.00**	1.69	0.27	**1.71**
	y-max	2.55	**14.57**	4.60	15.49	y-min	4.18	**0.00**	1.79	4.54
	z-max	9.56	6.65	**9.10**	14.78	z-min	4.10	4.47	**0.00**	6.07
±0.5	x-max	**19.19**	2.33	4.69	**19.89**	x-min	**0.00**	7.78	4.03	8.76
	y-max	3.11	**18.25**	5.71	19.38	y-min	4.55	**0.00**	2.01	4.97
	z-max	11.95	8.34	**11.30**	18.44	z-min	0.60	0.15	**0.00**	**0.62**

The bold lettered values in the above table represent the maximum and minimum values.

As seen from the plots, the maximum grip center or grip pose deviation occurs when the tolerance value is ±0.5 for the DH parameters. The pose deviation distance may go more than 20 mm. This is of course a worst-case scenario where all tolerance errors constructively combine to magnify the positional error in the end-effector position/orientation. It is far more likely that some tolerance increases the error value, while others decrease the error values. The deviations appear as linearly changing with the change of tolerances, though that has not been verified in this work; but one more point has been found that the maximum and minimum values for the *x*, *y*, *z* coordinates as well as for the grip center deviations are not coincidence. A summary of the observed data is shown in Table 3.6.

3.7 Conclusion

The simulation has been successfully executed for the test-frame Tiger 66.1. Both inverse and forward kinematics are completed using the iteration-based algorithms discussed here. The results indicate that the implementation of the algorithm for real time calculations is sensible. Once the valid workspace data and corresponding DH parameters for the poses are calculated, they can be stored in a database and those data can be used for real-time applications. If any pose-data and related DH parameters are not available in the database, they can be calculated and added to the database to enrich it during the operation and can cover the entire workspace with more precise data. In addition, the algorithm may be refined by reducing the iteration step values to generate valid DH parameter data sets for 100% valid poses with stricter error limits.

In this work, the simulation was run with randomly selected 100 pose points. As the size of the database increases, finding the DH parameter data runtime for a required pose becomes trivial. The algorithm can be refined by adding efficient data searching methods.

There is further scope to validate these data with measurements done on the physical system for practical purposes. The validation is intended to be done on Tiger 66.1 hexapod platform in the next phase of this research through non-invasive methods like Photogrammetry. In photogrammetry the final platform pose can be measured directly without instrumenting and measuring each joint. By doing multiple actual measurements and comparing the values with the calculated values, the in-built construction or fabrication errors of system can be evaluated and applying proper compensation factors for those in-built deviations, the hexapod platform can be guided to the desired pose more accurately. Such a calibration process of a Stewart-Gough platform would make practical application of the machine more useful. In the following chapter, the demonstration of the above method for deriving the DH parameters and their utilization in enhancing the accuracy of the Tiger 66.1 test frame has been presented.

4 CALIBRATION THROUGH PHOTOGRAMMETRY

Accurate calibration of a Stewart-Gough platform is important for their precise and efficient operation. However, the calibration of these platforms using forward kinematics is a challenge for researchers because forward kinematics normally generates multiple feasible and unfeasible solutions for any pose of the moving platform. The complex kinematic relations among the six actuator paths connecting the fixed base to the moving platform further compound the difficulty in establishing a straightforward and efficient calibration method. The Stewart-Gough platform “Tiger 66.1” developed in the author’s lab used for experimenting the calibration strategies described in this chapter. This system became operational upon completion of construction, marking its inaugural use. The author adopted three closely matching calibration strategies for compensating the errors in the system and response of the system to the methods adopted. These strategies leveraged a high-resolution digital camera and off-the-shelf software to non-invasively capture the poses of the moving platform's center. This process is non-invasive and does not need any additional equipment to be attached to the hexapod or any alteration of the hexapod hardware. This photogrammetry-based calibration process involves multiple high-resolution images from different angles to measure the position and orientation of the platform center in the three-dimensional space through photogrammetry. The target poses and actual poses are then compared, and the error compensations are estimated with three methods using the Least-Squared methods to calculate the predicted poses. Results from each of the three compensation approaches demonstrated noticeable enhancements in platform pose accuracies, suggesting room for further improvements. Given that "Tiger 66.1" is based on the general Stewart-Gough platform structure, the proposed calibration method holds promise for extension to machines operating on similar principles where non-invasive calibration is desirable. This study contributes to advancing the field of Stewart-Gough platform calibration, paving the way for more precise and efficient applications in various domains.

4.1 Introduction

Parallel Kinematic Machine (PKMs) constitute a pivotal domain within robotics. Any PKM is characterized by a fixed base and a moving platform. The base and platform are connected by multiple parallel actuators, and the number of actuators can vary between 3 to 6. The actuators are used to control the position and orientation of the platform. The machines with six actuators are called a Stewart Platform, a Stewart-Gough Platform, a Gough-Stewart Platform or more commonly, a hexapod. Hexapod platforms stands out as one of the most prominent and widely adopted parallel kinematic machines [64]. The six actuators add six degrees of freedom at the center of the moving platform [65]. To articulate the dynamics of the moving platform center point, a cartesian coordinate-frame is attached to the platform center. The configuration of the platform center is specified by three translatory or linear displacement along *x*, *y*, *z* axes and three rotations about the same *x*, *y*, *z* axes [66] from a reference position called the *home pose*. The position and orientation of the platform are dependent on the actuators and joints connecting the fixed base with the moving platform. The joints can be spherical (S) or universal (U) while the actuators provide linear motions through prismatic (P) joints. The hexapod, named “Tiger 66.1”, used by the authors has six actuators acting as six prismatic (P) actuated joints, each of them connected to the fixed base with a universal (U) joint at one end and the other end with a spherical joint (S) connected to the moving platform. The combination of joints defines the designation of the hexapod. This configuration classifies Tiger 66.1 as a 6-UPS (total 6 DOF for 1 Universal, 1 Prismatic, and 1 Spherical joint) Parallel Kinematic Machine [13]. This hexapod was built by the authors in their lab and is used in this study to experiment with photogrammetry-based calibration.

Calibration of a Stewart-Gough platform has attracted research interest in the last couple of decades [91]. Various methods were adopted to make the calibration processes simple and straightforward, using both forward and inverse kinematics. The forward kinematics of a hexapod are complex and difficult because of its non-linear kinematic equations and multiple solutions [92]. Moreover, the use of additional equipment and / or modification of the hexapod hardware makes the calibration process more complicated. Various pieces of equipment were used to conduct the calibration procedures. Zhuang *et al.* [40] used commercial electronic theodolite for the calibration of their hexapod platform. Ryu J., and Rauf A. [32] imposed constraint motion on the end-effector by fixing the length of one of the six actuators and using inverse kinematics. In another research effort, Großmann *et al.* [52] used a simple and robust double-ball-bar (DBB) for measurements from a continuously moving hexapod platform under six degrees of freedom. Liu *et al.* [53] innovatively adopted self-calibration, incorporating a three-dimensional laser tracker and a genetic algorithm into their calibration method, which involved both simulated and real measurements.

Digital cameras have assumed a significant role in hexapod calibration research. Daney *et al.* [56] harnessed a Sony digital video camera to measure the joint positions and leg lengths on their hexapod, named “Table of Stewart”. They seamlessly integrated inverse kinematics with data obtained from a digital camera to implement their calibration methodology. A omni-directional camera was employed by Dallej *et al.* [55] in their lab to measure the positions and orientations of the actuators in their hexapod, leveraging inverse kinematics for calibration. A high resolution digital camera was used by Nategh *et al.* [51] for the calibration process. A camera was used to capture images of the moving platform in a vertical downward direction. They developed MATLAB code to extract the platform's pose across various positions and orientations from these images.

With the continuous evolution of digital cameras and the increasing availability of software and hardware support, the integration of image processing into calibration methods is increasing. Notably, while photogrammetry has witnessed substantial adoption across various fields, its application in hexapod calibration remains relatively limited. This present study exclusively relied on photogrammetry. A high-resolution digital camera Nikon D3200 with AF-S DX Nikkor 18-105mm lens has been used. For image processing, a photogrammetry software “Photomodeler 2023” and PTC Creo 9.0.5 student edition have been used.

The methods proposed in this study utilized inverse kinematics to derive forward kinematics solutions for platform poses. From the forward kinematic solutions, the DH parameters for each actuator path were calculated using modified Denavit-Hartenberg (DH) convention. Although the DH convention is the most popular method for forward kinematics in serial manipulators, its application remains very limited in parallel manipulators due to the closed loop nature of PKMs [93] and generation of multiple feasible and unfeasible solutions for any pose of parallel robots. Here modified DH convention was adopted because of its simplicity and straight-forward nature of implementation. This was possible by the new algorithm [94] developed by the authors to find unique, feasible forward kinematic solution for any pose in PKMs. The DH parameters obtained from the target pose and corresponding actual pose were compared and analyzed to develop the calibrated predicted pose for the target configuration. As initial experiments, all the error compensations were calculated with Least-Squared methods.

The subsequent sections of this article are organized as follows. The introduction is presented in this initial section. The second part illustrates the calibration methodology used by the authors to calibrate their “Tiger 66.1” hexapod. The following section explains the experimental setup and data collection methods used. In Section 4, the data collected has been analyzed and findings were documented. Section 5 engages in discussions pertaining to the analyzed data. The concluding remarks are shared in the last section.

4.2 Calibration Methodology

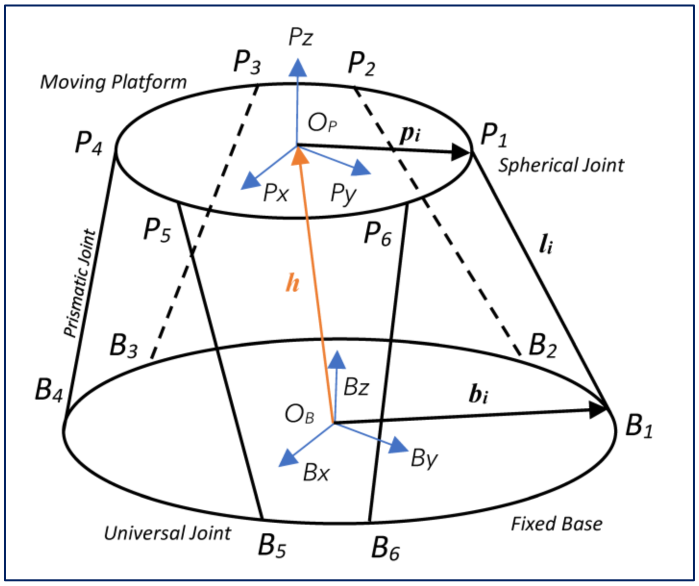


Figure 4.1: A typical hexapod configuration

In Figure 4.1, the typical sketch of a hexapod platform [94] is illustrated. $\boldsymbol{O_B}$ is the center of the fixed base and $\boldsymbol{O_P}$ is the center of the moving platform. Two cartesian coordinate frames $\boldsymbol{O_B B_x B_y B_z}$ and $\boldsymbol{O_P P_x P_y P_z}$ are attached with these center points. The cartesian coordinate frame with $\boldsymbol{O_B}$ as origin has *x*, *y*, *z* axes denoted by $\boldsymbol{B_x}$, $\boldsymbol{B_y}$ and $\boldsymbol{B_z}$ respectively and for moving frame with origin $\boldsymbol{O_P}$ has $\boldsymbol{P_x}$, $\boldsymbol{P_y}$ and $\boldsymbol{P_z}$ axes to indicate *x*, *y*, *z* axes, respectively. $\boldsymbol{B_z}$ and $\boldsymbol{P_z}$ are the vertical axes of the respective coordinate frames. The configuration of the moving platform is defined by the position and orientation of coordinate frame attached at its center with respect to base coordinate frame by 6 parameters: distance along *x*, *y*, *z* axes and rotations about the same three axes.

Figure 4.2 shows the 3D CAD model of Tiger 66.1 and the actual platform used for the experiments. Tiger 66.1 has been developed for characterizing additively manufactured materials under complex loading conditions including tension, torsion, bending and combinations thereof [95].

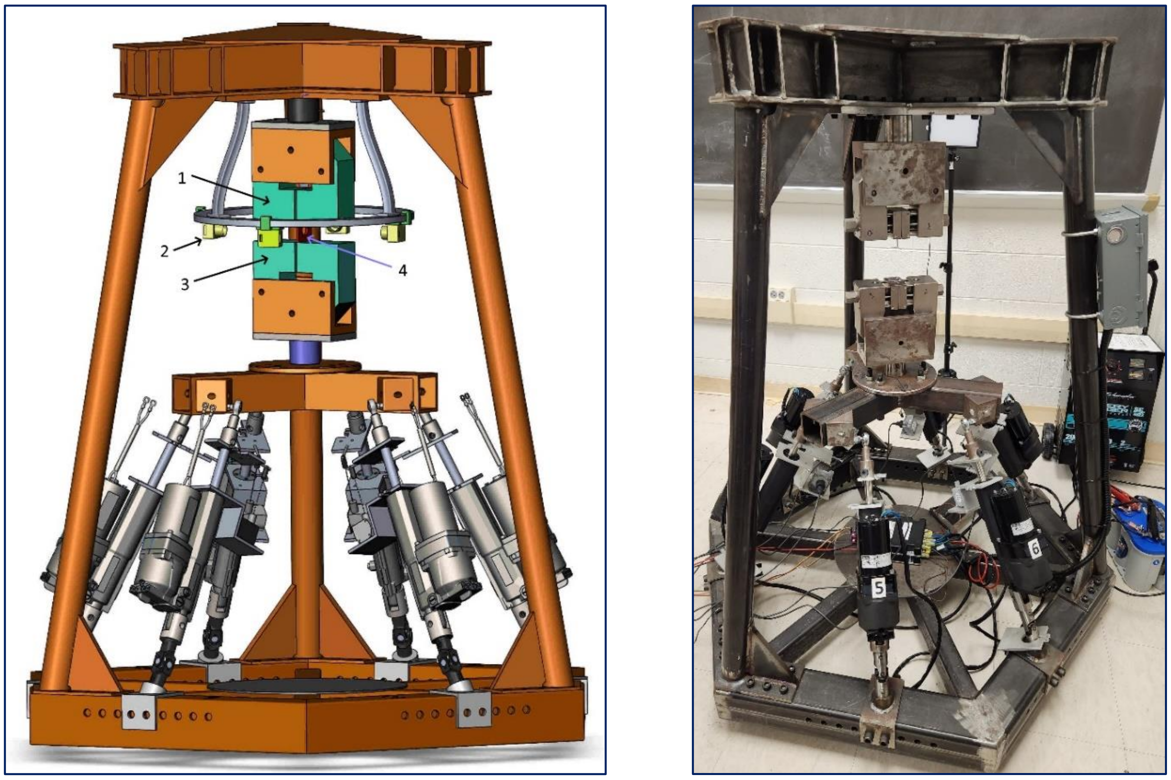


Figure 4.2: CAD model of hexapod Test Frame "Tiger 66.1" & the actual test frame

The configuration of a general Stewart-Gough platform has been modified to make the machine suitable for material testing while keeping the basic principle of Stewart-Gough platform unchanged. The fixed base was extended through rigid structures to move the fixed coordinate frame to the top of the moving platform. This was done to install a fixed gripper in a suitable position for the convenience of material characterization tests. The two green blocks as shown in the CAD model on the upper part of the system are the grippers for holding the material specimen (4) to be tested. The upper gripper (1) is mounted on the fixed frame, and the lower gripper (3) is fixed at the center of the hexapod moving platform. All motions and forces are applied to the test specimen by moving the lower gripper. Photogrammetry has been planned to be used for measurements in this test process, so there are provisions to fix four digital cameras (2) to capture images from the test zone. As the manufactured platform is at the beginning stage of its development, these cameras have not yet been installed. Instead, all images were captured from different angles using an external camera with a suitable field of view, as mentioned previously.

To use photogrammetry, two new coordinate frames have been introduced in Tiger 66.1 for all measurements and calculations: one at the center of the upper grip end plate and another at the center of the lower grip end plate. The new frame configurations are shown in Figure 4.3. $\boldsymbol{O_{UG}}$ is the center of the fixed upper grip end plate and the associated coordinate axes are $\boldsymbol{UG_x}$, $\boldsymbol{UG_y}$ and $\boldsymbol{UG_z}$ for x, y, z axes, respectively. Similarly, $\boldsymbol{O_{LG}}$ is the center of the movable lower grip end plate and the associated coordinate axes are $\boldsymbol{LG_x}$, $\boldsymbol{LG_y}$ and $\boldsymbol{LG_z}$ corresponding to x, y, z axes, respectively. The distance between $\boldsymbol{O_{UG}}$ and $\boldsymbol{O_B}$ are fixed and known and the distance $\boldsymbol{fd}$ between $\boldsymbol{O_{LG}}$ and $\boldsymbol{O_P}$ are also fixed and known. The distance $\boldsymbol{gd}$ between the fixed grip center $\boldsymbol{O_{UG}}$ and moving grip center $\boldsymbol{O_{LG}}$ will change during the operation of Tiger 66.1. The relationships between these frames can be easily established by spatial transformation matrices. In the '*home pose*' [84] all the z-axes remain vertical and colinear while the other axes remain parallel to each other and all x-axes or y-axes remain in one plane. In the home pose the length of all actuators is equal and known. The lower grip center is moved with the moving platform by controlling the lengths of the actuators, and the resultant motion generated at the lower grip center is a translation, rotation, or combination of both.

The new position and orientation of the lower grip center depend on the values of roll, pitch, yaw, and translation motion along x, y, z axes as per the Euler angle representations [85]. These are values measured from $\boldsymbol{O_{LG}}$ with respect to its home pose. In this investigation, the home pose has been specifically established at ***gd*** = 50mm, in accordance with the other requisite home pose conditions. This ***gd*** value is equal to the standard gage length for tensile test of any material.

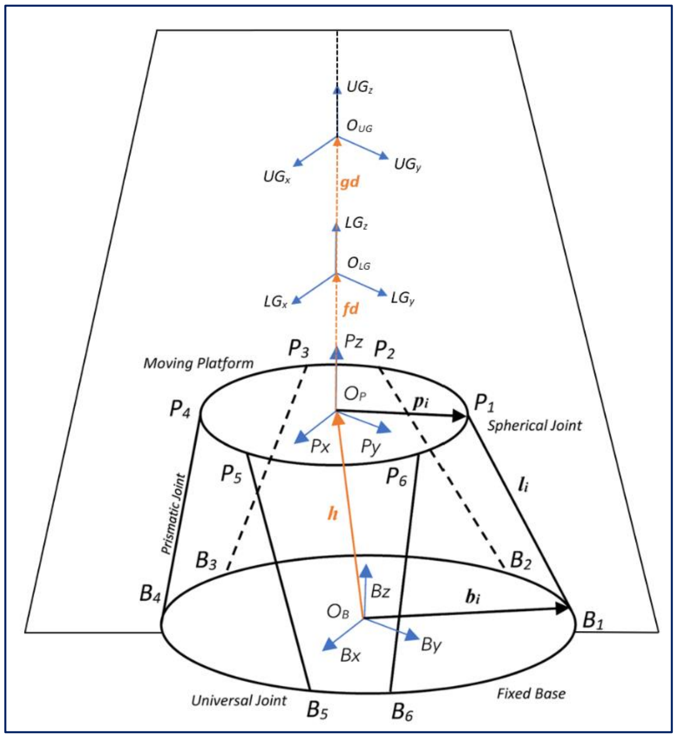


Figure 4.3: Tiger 66.1 coordinate frame configuration

The rotations from the home pose are expressed by vector Φ and

$$\Phi = (\alpha\ \beta\ \gamma)^T \tag{4.1}$$

where α (roll), β (pitch), γ (yaw) denotes the rotation angles about x, y & z axes, respectively.

The translations are expressed by vector ***d*** where,

$$\boldsymbol{d} = (x\ y\ z)^T \tag{4.2}$$

and x, y & z are the translation values from home pose along x, y, and z axes, respectively.

For the calibration process, Tiger 66.1 always starts moving from the home pose to travel to each new pose. In the home pose all actuator lengths are equal. Once the new actuator lengths were calculated by inverse kinematics for each new pose, the platform controller is fed with new actuator lengths. Employing the digital camera, multiple photographs were taken for each pose from suitable angles and are processed through Photomodeler and ProEngineer Creo to get the new position of the lower grip center in the 3D space. The target pose values are subtracted from actual reached pose values to calculate the six error parameters in terms of x, y, z positions and orientations. These error values were now used to calculate the error compensation for Tiger 66.1.

For a pose, if the target pose values = $(x\ y\ z\ \alpha\ \beta\ \gamma)^T$

and the real measurement shows the values = $(x'\ y'\ z'\ \alpha'\ \beta'\ \gamma')^T$,

then the error vector for a pose = $((x'-x)\ (y'-y)\ (z'-z)\ (\alpha'-\ \alpha)\ (\beta'-\beta)\ (\gamma'-\ \gamma))^T$

For an n-number of poses, there will be n-numbers of such error vectors.

The error values are used to calculate the correction values by the Least-Square method. The correction values are then combined with the target pose values to calculate the predicted poses.

The Least-Squares method [96] is a statistical method for fitting a line or curve to a set of data points. It minimizes the sum of the squared residuals, which are the distances between the data points and the fitted line or curve. The mathematical expression for the least squares method:

$$\text{Cost function} = \min \sum (y - f(x))^2 \tag{4.3}$$

where y is the dependent variable, x is the independent variable, $f(x)$ is the fitted line or curve, and $\sum$ is the sum of all the terms.

In other words, the least squares method finds the line or curve that minimizes the total error between the data points and the fitted line or curve.

Three different strategies were adopted by the authors to find the predicted poses after combining them with the correction values. These three strategies are explained in the next three sections.

4.2.1 *Calibration – Option 1: With corrected DH parameters*

In this calibration approach, the Denavit-Hartenberg (DH) parameters for each pose were computed using the new algorithm developed, as explained in [94]. This algorithm enables one to find a unique, feasible solution for each platform pose by using forward kinematics. Using this technique, the DH parameters in each actuator path for each target pose and the corresponding actual pose were determined. The error for each pair of DH parameters were calculated. Using the least squares method, the predicted DH parameters were calculated. The predicted DH parameters for each actuator path were used to determine the predicted platform pose along each actuator path. For each target pose, there are 6 predicted poses calculated with compensated DH parameters through six actuator paths. Computing by matrix averaging technique, the 6 position vectors (through each actuator path) for each pose were unified and a new position vector for the predicted pose has been calculated. The matrix averaging of 6 orientation vectors from each actuator path for a pose did not yield any meaningful outcome, so they have not been treated in the same way as the position vectors. The orientation vectors for the predicted poses were left unchanged. A complete predicted pose vector is defined by combining the unified position vector with the target orientation vector (Euler angles) for that pose. (suffix 'uc' has been used to indicate uncorrected or uncompensated values and suffix 'dc' denotes the corrected or compensated values through DH parameters.)

If a target pose vector = $(x_{uc}\ y_{uc}\ z_{uc}\ \alpha_{uc}\ \beta_{uc}\ \gamma_{uc})^T$

and the predicted position vector after correction = $(x_{dc}\ y_{dc}\ z_{dc})^T$

then the new predicted pose for the target pose = $(x_{dc}\ y_{dc}\ z_{dc}\ \alpha_{uc}\ \beta_{uc}\ \gamma_{uc})^T$ (4.4)

4.2.2 *Calibration – Option 2: With corrected DH parameters & Euler angles*

In this option, the position vector for a pose has been corrected in the same manner as described in the previous section.

However, the orientation vectors were treated differently. For each pose, there are 2 orientation vectors: one for the target pose and the other one for the actual measured pose. For *n* poses, n error values were calculated by subtracting the target orientations from the actual orientations. Using the Least-Square method, the compensation angles values for each pose have been calculated from error values. Now, each predicted pose has been calculated by creating a set of unified position vectors and compensated orientation vectors. Mathematically, it can be expressed as follows (suffix 'lc' denotes least-square method compensated values):

Let, the target pose vector = $(x_{uc}\ y_{uc}\ z_{uc}\ \alpha_{uc}\ \beta_{uc}\ \gamma_{uc})^T$

The predicted translation vector after correction = $(x_{dc}\ y_{dc}\ z_{dc})^T$

The predicted orientation vector after correction as stated above = = $(\alpha_{lc}\ \beta_{lc}\ \gamma_{lc})^T$

then the new predicted pose for the target pose = $(x_{dc}\ y_{dc}\ z_{dc}\ \alpha_{lc}\ \beta_{lc}\ \gamma_{lc})^T$ (4.5)

4.2.3 Calibration – Option 3: With corrected translation vectors & Euler angles

In this option, the predicted pose for a target pose is calculated by correcting both position vectors and orientation vectors using the Least-Squares method.

For *n* number of poses there are *n* numbers of target poses and actual poses. Each pose has 6 parameters, which are a combination of a position vector and an orientation vector. For *n* numbers of poses, *n* error values were calculated by subtracting the target pose parameters from the actual pose parameters. Using the Least-Squares method, the compensation values for each pose parameter were calculated from the error values and applied to the target poses. These compensated pose parameters are used as the predicted pose for each target pose. The main difference between this option and the earlier two options is that in this case, the DH parameters were not considered for determining the predicted poses. The new vector calculation can be expressed in the following way:

Let, the target pose vector = $(x_{uc}\ y_{uc}\ z_{uc}\ \alpha_{uc}\ \beta_{uc}\ \gamma_{uc})^T$

The predicted position vector after correction = $(x_{lc}\ y_{lc}\ z_{lc})^T$

The predicted orientation vector after correction = $(\alpha_{lc}\ \beta_{lc}\ \gamma_{lc})^T$

then the new predicted pose for the target pose = $(x_{lc}\ y_{lc}\ z_{lc}\ \alpha_{lc}\ \beta_{lc}\ \gamma_{lc})^T$ (4.6)

The above three options are summarized in the Table 4.1

Table 4.1: Summary of the experiment options

Option no.	**Position vector correction**	**Orientation vector correction**
1	DH parameters by LSM	No correction
2	DH parameters by LSM	Least Square method
3	x y z coordinates by LSM	Least Square method

After position and orientation error ranges were calculated for uncompensated and compensated poses, the magnitude of errors for position and orientation have been calculated using the "Root Mean Square Error" (RMSE) equation:

$$\text{Magnitude of error or improvement} = \sqrt{\frac{(x\_range^2 + y\_range^2 + z\_range^2)}{3}} \tag{4.7}$$

4.3 Experimental Setup & Data Collection

4.3.1 Selection of the camera for photogrammetry

Selecting a camera for photogrammetry involves considering several key parameters to ensure optimal performance [97]. Here are the main factors to look for [98]:

Resolution: Higher resolution sensors capture more detail, contributing to the accuracy and precision of 3D reconstructions. A camera with higher megapixel counts for better results in photogrammetry.

Sensor Size: Larger sensors generally provide better image quality and low-light performance. Full-frame sensors or APS-C sensors are common choices for photogrammetry applications.

Lens: A wide-angle lens is ideal for photogrammetry, as it will allow you to capture more of the scene in each frame. However, a fisheye lens must be avoided, as they can cause distortion.

Shutter speed: A fast shutter speed is important for capturing sharp images, especially when shooting moving subjects.

ISO Range and Low-light Performance: A low ISO setting will give you less noise in your images. However, ISO setting needs to increase in low light conditions.

Remote Control Capability: Cameras with remote control options are beneficial for integration into automated systems or for capturing images from difficult-to-reach locations. Remote control capability enhances the camera's versatility in photogrammetry applications.

Image Stabilization: In situations where stability is a concern, such as when the camera is mounted on a moving platform, optical or sensor-shift image stabilization can help reduce blur and ensure sharper images.

Dynamic Range: A higher dynamic range allows the camera to capture a wider range of tones in a scene. This is crucial for maintaining detail in both shadow and highlight areas, improving the overall quality of photogrammetric reconstructions.

File Format: Cameras that support the RAW file format provide more flexibility during post-processing, allowing for better control over adjustments and corrections.

Considering these parameters will help you choose a camera that meets the specific requirements of photogrammetry, ensuring reliable and accurate results in 3D reconstruction projects [99].

Here are some specific camera recommendations for photogrammetry: Canon EOS R5, Sony Alpha a7 IV, Nikon Z7 II, Nikon D3200, Fujifilm GFX 100S, Point Grey Research Flea3, and FLIR Blackfly S.

4.3.2 Selecting Nikon D3200

The Nikon D3200's compatibility with a range of lenses is a significant asset for photogrammetry applications. Some important considerations regarding lens compatibility for photogrammetry with the D3200 have been listed below [Nikon website]:

Prime Lenses: Prime lenses with fixed focal lengths are often preferred for photogrammetry because of their optical clarity and lack of distortion. The D3200 is compatible with a variety of Nikon prime lenses, such as the Nikon AF-S NIKKOR 50mm f/1.8G, which can deliver sharp and distortion-free images. In this experiment Nikon AF-S DX Nikkor 18-105mm has been used in a fixed setting condition.

Vibration Reduction (VR) Technology: Lenses equipped with VR technology can be advantageous, especially in dynamic or unstable environments where a robot might introduce vibrations. The Nikon AF-S NIKKOR 18-105mm f/4G ED VR is an example of a versatile zoom lens with VR, providing image stabilization for sharper photos.

Compatibility with DX Format: The D3200 uses a DX-format sensor, and lenses designed for DX-format cameras can maximize the camera's performance. Nikon's DX lenses are tailored for APS-C sensor cameras like the D3200.

High-resolution sensor: The Nikon D3200 has a 24.2-megapixel sensor, which can capture high-resolution images. This is important for robot photogrammetry because it allows the software to accurately reconstruct the 3D model.

Wide-angle lens: The Nikon D3200 comes standard with an 18-55 mm lens, which is a wide-angle lens. This type of lens is ideal for robot photogrammetry because it allows the robot to capture a large field of view in each image. This can reduce the number of images that need to be captured to create a complete 3D model.

Fast shutter speed: The Nikon D3200 has a maximum shutter speed of 1/4000th of a second. This is fast enough to freeze motion blur, which is important for robot photogrammetry because the robot may be moving during the image capture process.

Affordable price: The Nikon D3200 is a relatively affordable camera model. This makes it a good option for researchers and businesses that are on a budget.

Easy to use: The Nikon D3200 is a relatively easy-to-use camera. This makes it a good option for researchers and businesses that do not have much experience with photography.

The Nikon D3200 is considered a viable option for robot photogrammetry, providing the necessary features to capture high-quality images for 3D reconstruction in a variety of environments.

4.3.3 Experimental setup

The experimental setup for these calibration processes did not need any special hardware beyond one high resolution digital camera. This camera is independent from the hexapod test fixture and does not interfere with the operations of the system. A Nikon D3200 digital camera has been used for this purpose. The lens used is Nikon AF-S DX Nikkor 18-105mm. The camera has been used in “manual” mode. This was a requirement by the photogrammetry software to keep the camera calibration and other image parameters uniform throughout the process. The camera settings were kept fixed once the camera calibration was done. Though the zoom value used in the process is not visible on the camera setting windows shown in the Figure 4.4, the zoom settings are also kept fixed to maintain the same values for the lens parameters.

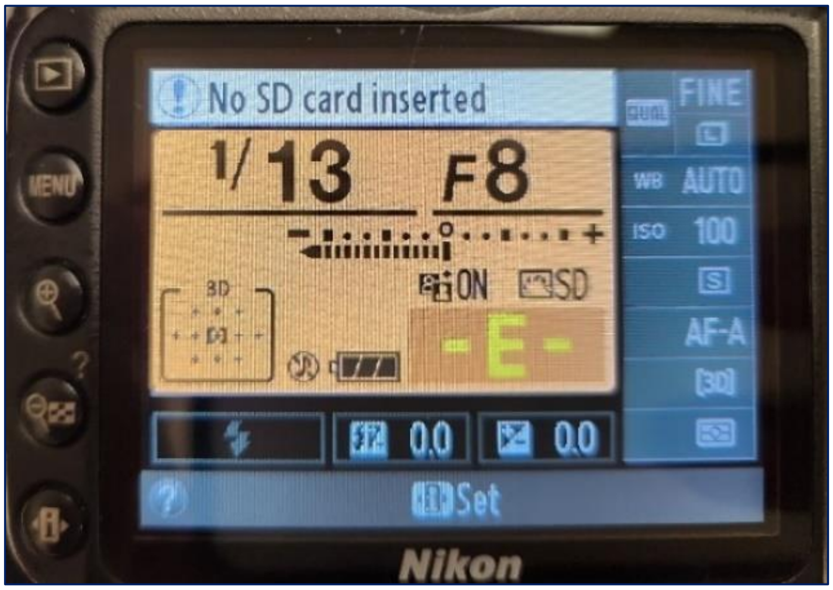


Figure 4.4: Nikon D3200 camera settings for Photogrammetry

In this experimental setup, the initial step involved the calibration of the camera in accordance with the specific calibration procedure prescribed by the photogrammetry software, "Photomodeler". To accomplish this, a series of printed templates, as shown in Figure 4.5, were employed. Subsequently, an automated camera calibration process was executed once these template images were processed through Photomodeler. Upon the successful completion of this calibration procedure, the camera was ready for the project. The completed calibration data were recorded and stored in a file, which was subsequently referenced during the image processing stage for the calibration of Tiger 66.1.

4.3.4 Data collection

In the next step, 34 random poses were selected in the moving gripper’s workspace. These workspaces were free from “singularity” condition and that has been verified in MATLAB code before using in the control software. For each pose three images were taken in three different angles to satisfy the need for photogrammetric measurements. These three images were processed using Photomodeler, as shown in Figure 4.6. Each image was processed manually and integrated to generate a 3D wire frame model (visible in the extreme right-side window of the Figure 4.6).

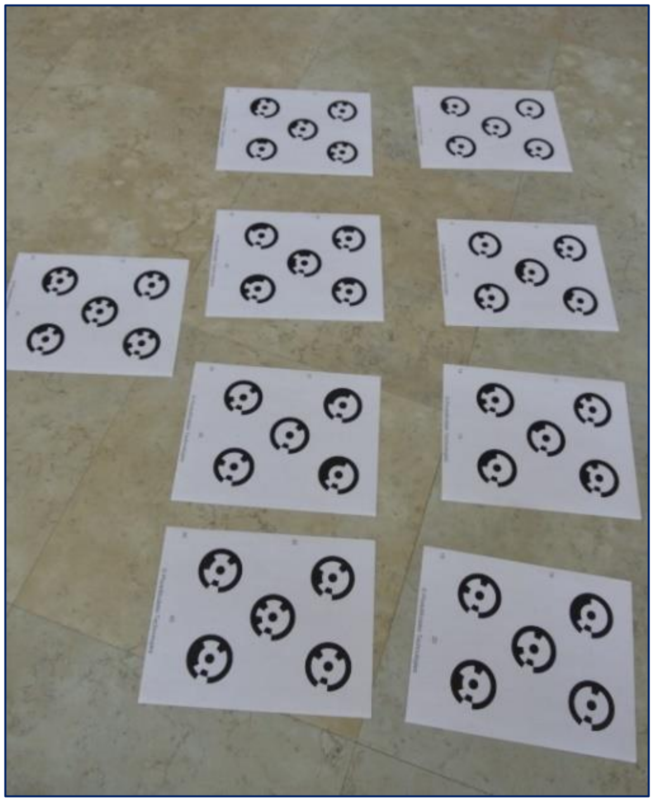


Figure 4.5: Calibration templates

The wire frame model now becomes the input for the ProEngineer Creo software. Each wire frame model is manually analyzed to collect the pose data. A typical screenshot from Creo is shown in Figure 4.7 after the complete processing of a wire frame model in this software package. The outcome from this process is the translation vector and rotation matrix for the actual motion performed by the hexapod. The 3 rows of the last column in the 3x4 matrix shown in Figure 4.7 denotes the position vector and the remaining 3x3 matrix denotes the orientation matrix. By using MATLAB function, the rotation values about the axes are extracted from this 3x3 matrix.

This process is repeated for all poses before and after calibrations following 3 options described in section 2.

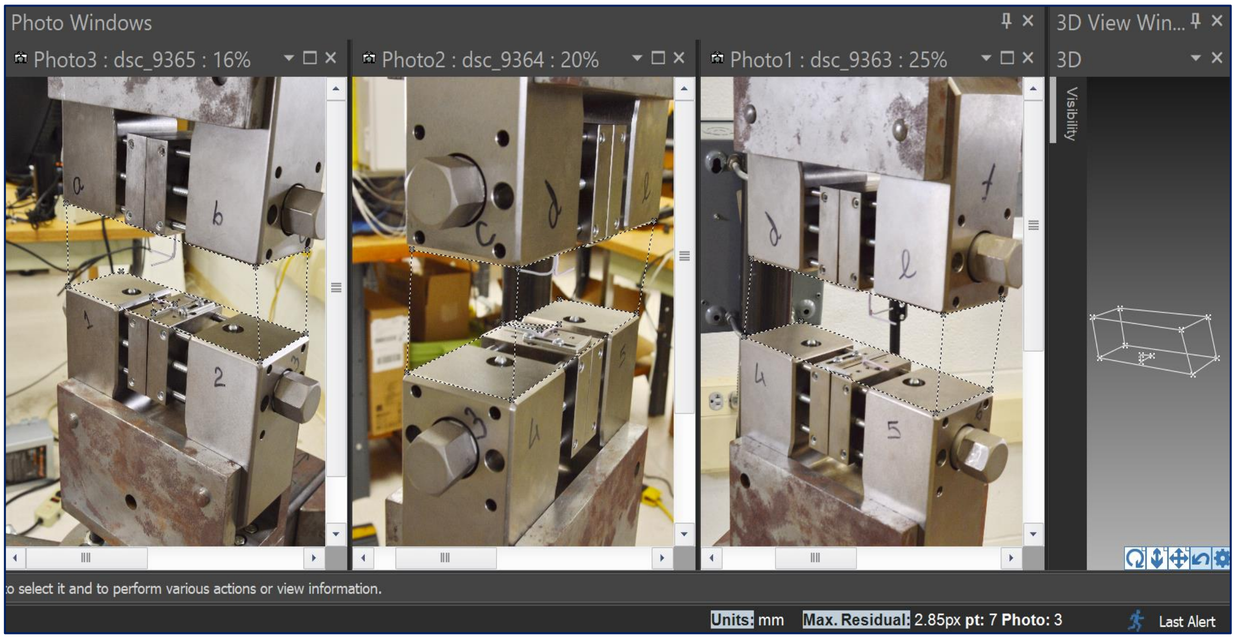
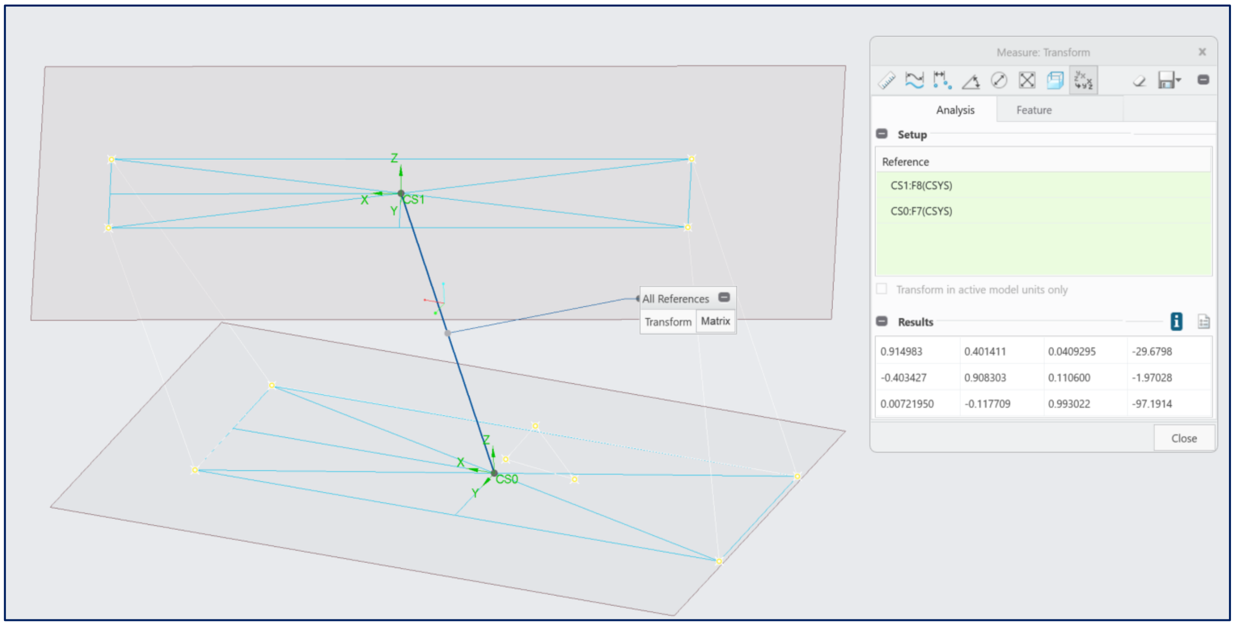


Figure 4.6: Photomodeler user interface for image processing



0.914983	0.401411	0.0409295	-29.6798
-0.403427	0.908303	0.110600	-1.97028
0.00721950	-0.117709	0.993022	-97.1914

Figure 4.7: Manual processing of wire frame model in ProEngineer Creo

4.4 Results and Analysis

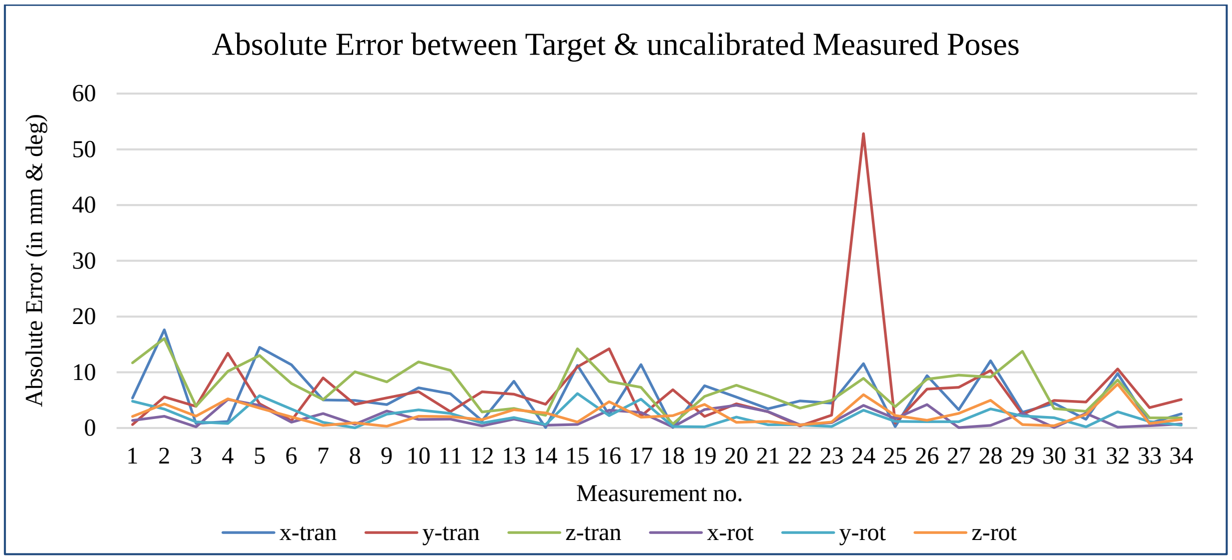


Figure 4.8: Absolute Error between Target & uncalibrated Measured poses

The 6 parameters for all the poses were extracted by photogrammetry and recorded. In the beginning the target poses and actual poses were compared by measuring the differences between the respective pose parameters. The absolute difference values are plotted and are shown in Figure 4.8. A random 34 target poses in the workspace were considered for the experiment and the lower grip center was moved to those poses one by one. The actual poses reached by the lower grip center in this process are uncalibrated poses and obtained from as-build condition. The prefix '-*tran*' has been used to denote the position variables and '-*rot*' has been used to denote the orientation variables.

From the chart in Figure 4.8, it is seen that pose number 24 has one parameter that is significantly out of range compared to the other values. It was determined that while this point is not singular, it is close to a singular point. Therefore, it was designated as an outlier and removed from future calculations. The error range of the uncalibrated poses is shown in Figure 4.9.

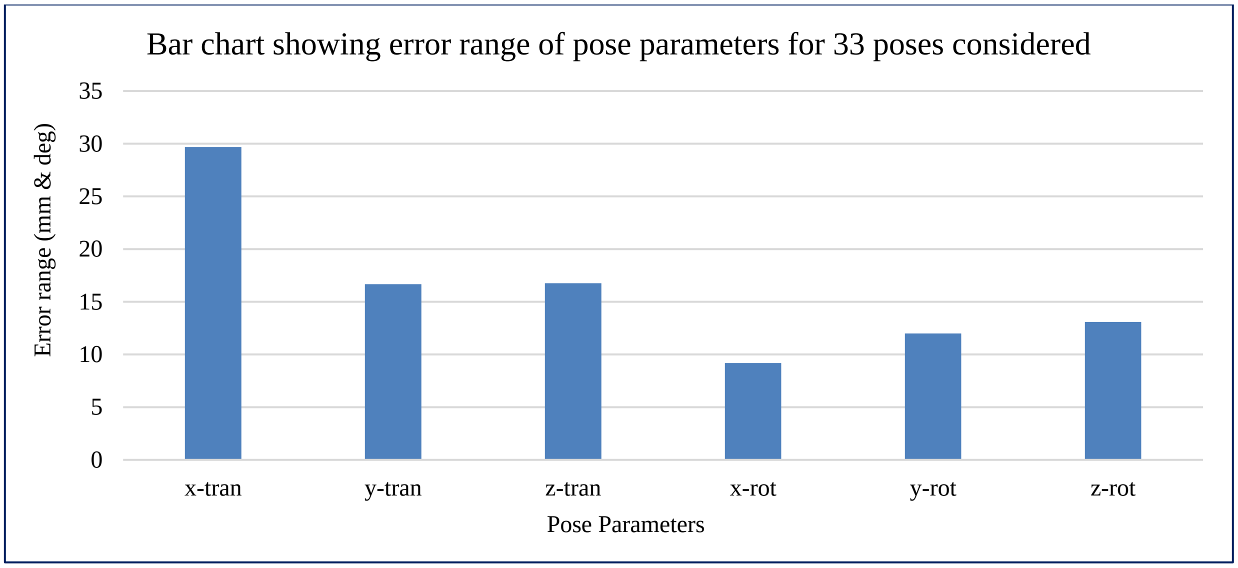


Figure 4.9: Error range of the pose parameters in the uncalibrated condition

4.4.1 Calibration results: Option 1

Calibrating as per the options, the lower grip center has been moved to the new predicted positions for the corresponding target position. During this calibration process, some of the predicted pose parameter values after compensating fell outside the hexapod operating range and were removed from further calculations and analysis. The total valid pose numbers decreased to 27 after removing the out-of-range values. The hexapod has been instructed by the controller to move to the new predicted poses one by one. The actual pose measurements were done through photogrammetry, and the pose data were compared with the corresponding target pose data. The bar chart in Figure 4.10 shows the error range of 6 pose parameters before and after calibration.

There are improvements in pose parameters except position values along y and z axis. For all other parameters various measures of improvements were observed.

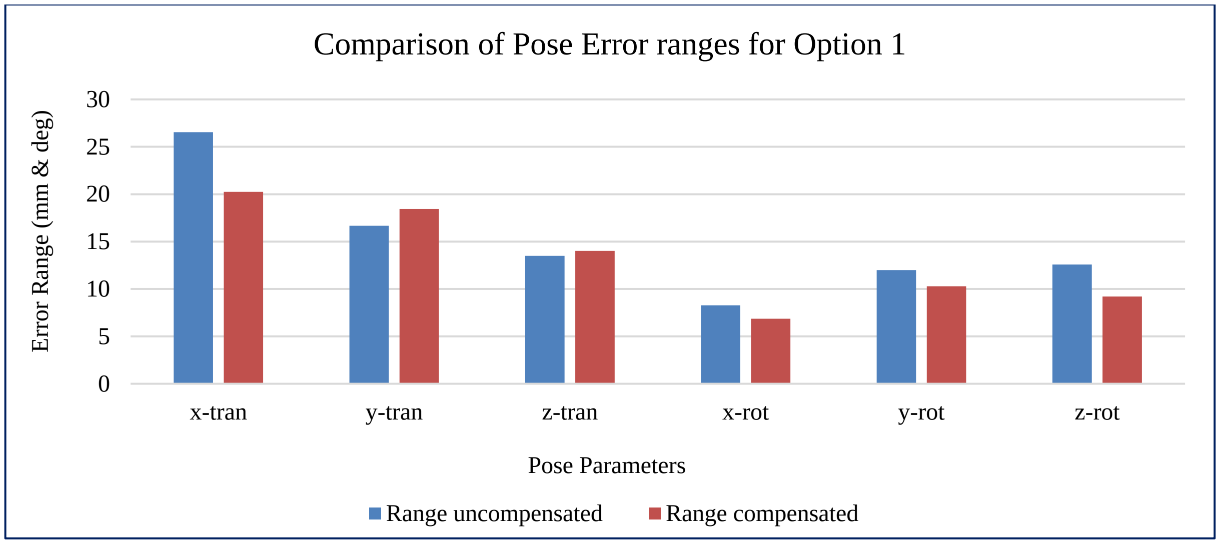
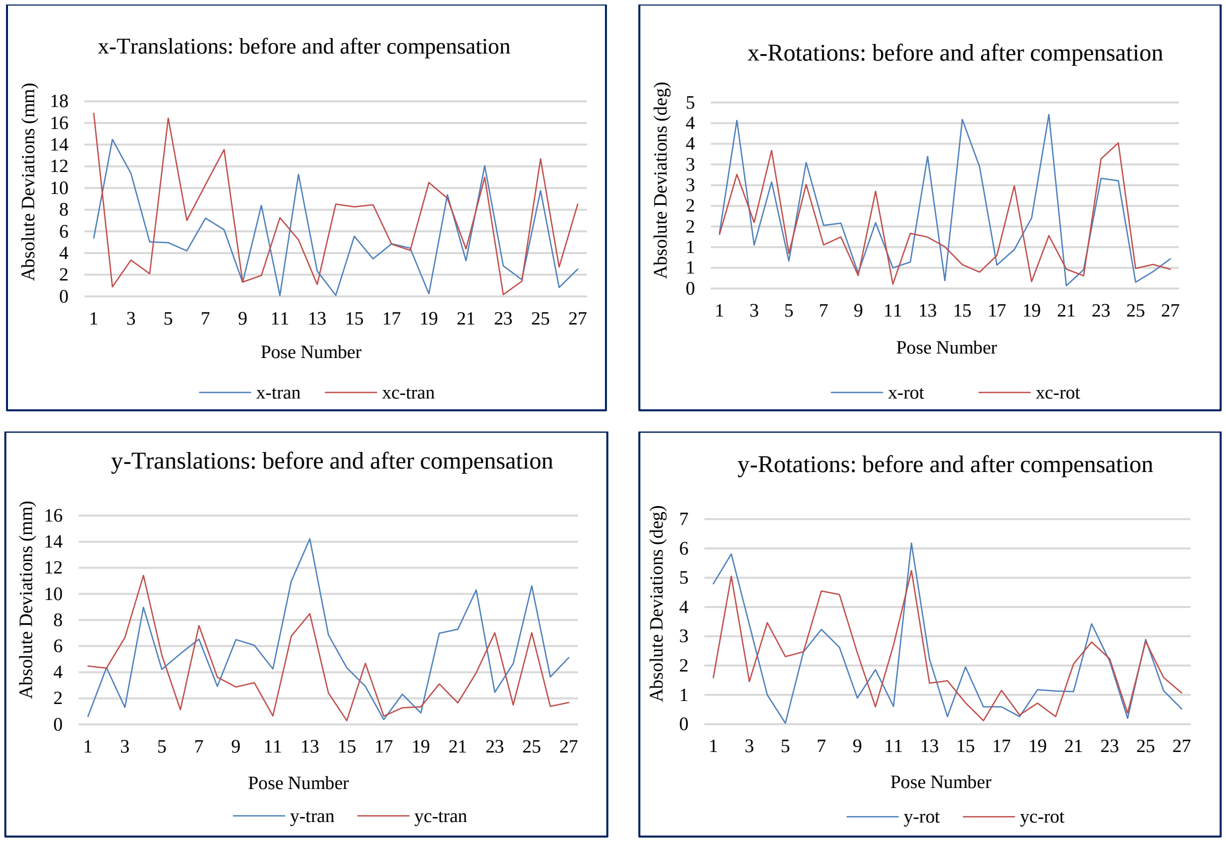
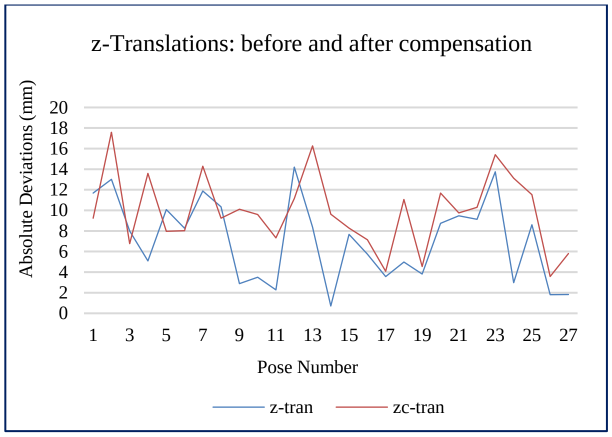


Figure 4.10: Comparison of pose parameters error range after calibration as per Option 1

The absolute deviations of each pose parameters before and after the calibration are shown in Figure 4.11. Here the suffix ‘*c*’ with axis identifier denotes the calibrated values.





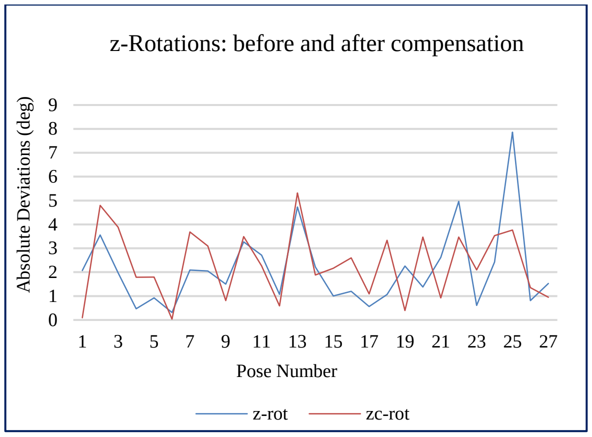


Figure 4.11: Comparison of absolute deviations of all 6 pose parameters for option 1

To better understand the impact of the calibration method used as per option 1, the Magnitude of Errors for position (measured in mm) and orientation parameters (measured in deg) were compared. The data were shown in the Table 4.2. The magnitude of errors for position and orientation values were calculated separately. The results show that the magnitude of error for position improved by 9.8% and for orientation improvement is 19.9%.

Table 4.2: Pose improvement after calibration as per option 1

Parameters	**x-tran**	**y-tran**	**z-tran**	**x-rot**	**y-rot**	**z-rot**
Error range-uncompensated	26.54	16.67	13.49	8.27	11.99	12.59
Magnitude of errors - uncompensated		19.70			11.12	
Error range-compensated	20.25	18.44	14.01	6.86	10.29	9.2
Magnitude of errors - compensated		17.76			8.90	
Magnitude of improvement %		**9.8%**			**19.9%**	

4.4.2 Calibration results: Option 2

In this calibration method, the parameter values for a few of the predicted poses were outside the hexapod’s travel range after compensation. Those poses were not considered for further calculations. After removing those poses, calibration calculations were done with 26 poses. The error range of the uncalibrated and calibrated pose parameters are shown in Figure 4.12.

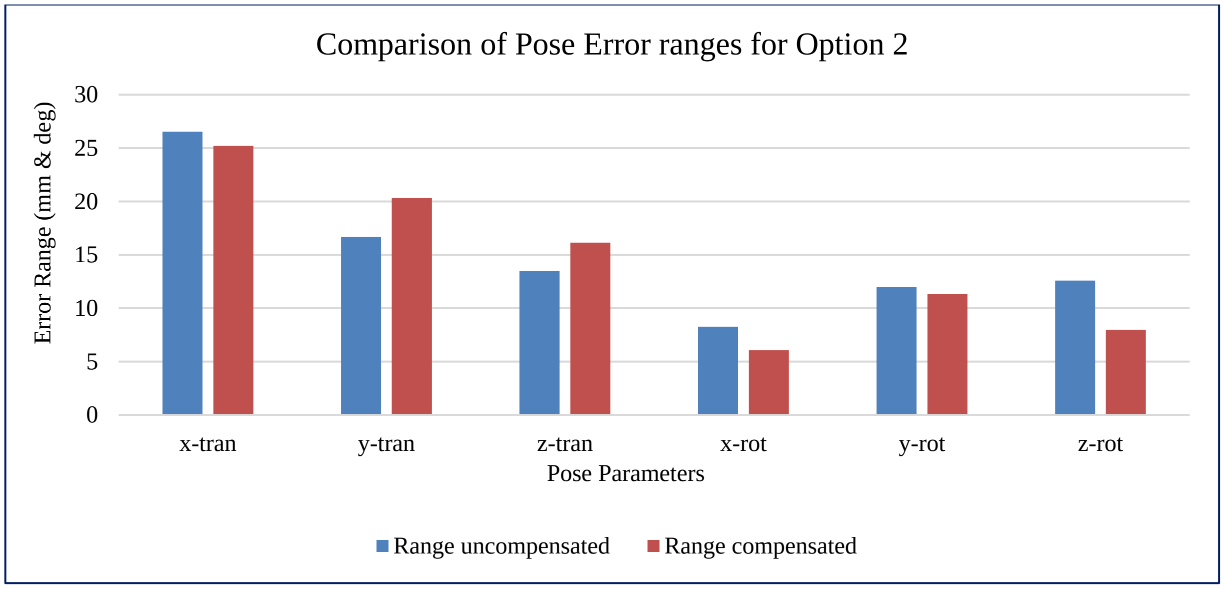
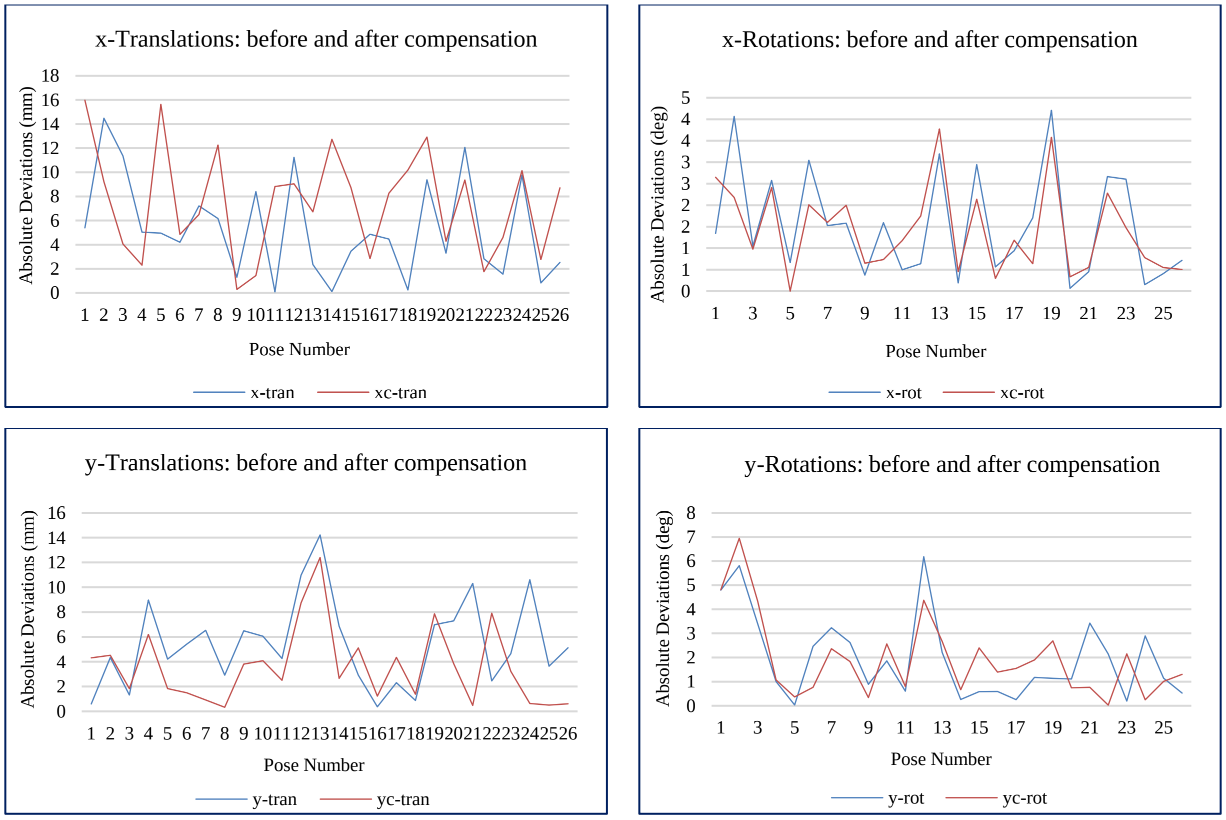
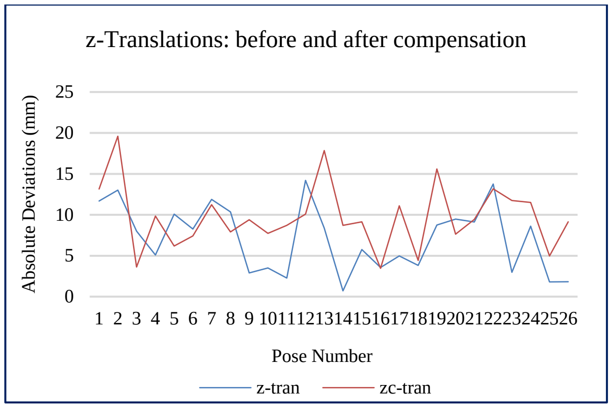


Figure 4.12: Comparison of pose parameters error range after calibration as per Option 2

The individual parameter deviations comparison can be seen in the Figure 4.13.





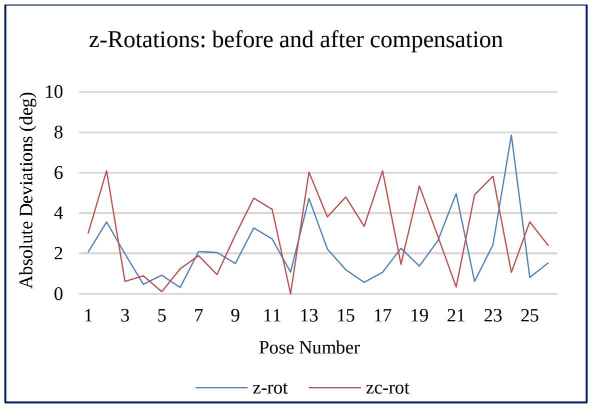


Figure 4.13: Comparison of absolute deviations of all 6 pose parameters for option 2

The pose parameters improvement based on this method of calibration is calculated and shown in Table 4.3.

Table 4.3: Pose improvement after calibration as per option 2

Parameters	**x-tran**	**y-tran**	**z-tran**	**x-rot**	**y-rot**	**z-rot**
Error range-uncompensated	26.54	16.67	13.49	8.27	11.99	12.59
Magnitude of errors - uncompensated		19.70			11.12	
Error range-compensated	25.20	20.31	16.14	6.05	11.32	7.99
Magnitude of errors - compensated		20.88			8.73	
Magnitude of improvement %		**-6.0%**			**21.5%**	

In this case, the magnitude of error for the positional part of the pose deteriorated by 6%, although 21.5% improvement was observed in the orientation part.

4.4.3 Calibration results: Option 3

In the earlier two options, the position parameters were corrected using DH parameter corrections through the Least-Square methods. In this option, both position and orientation parameters were compensated by using only Least-Square methods from the error values obtained from the uncompensated measurements and comparing those values with the target values. After compensating the pose parameters in this method, only 21 poses were found to be inside the valid workspace of the lower grip center of Tiger 66.1. The comparison of the error ranges before and after calibration is shown in the Figure 4.14.

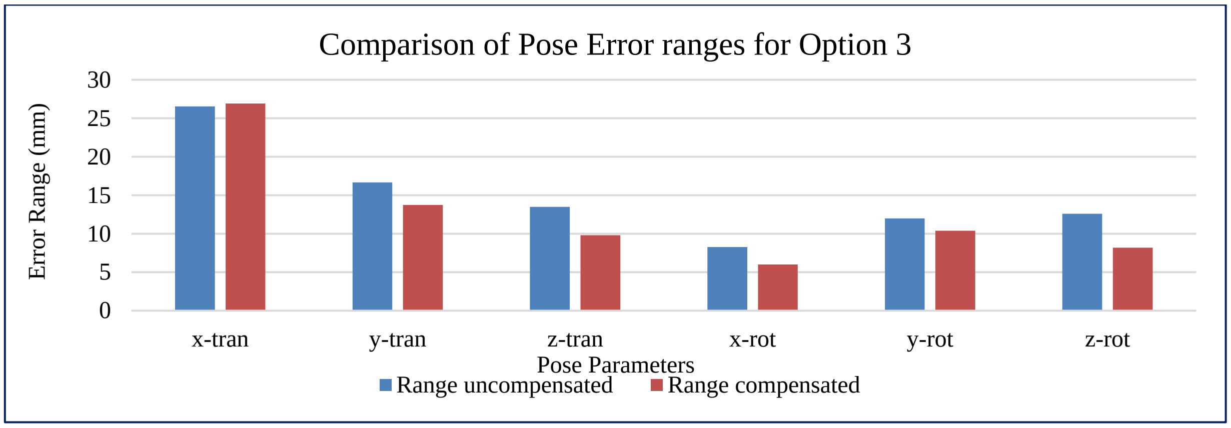
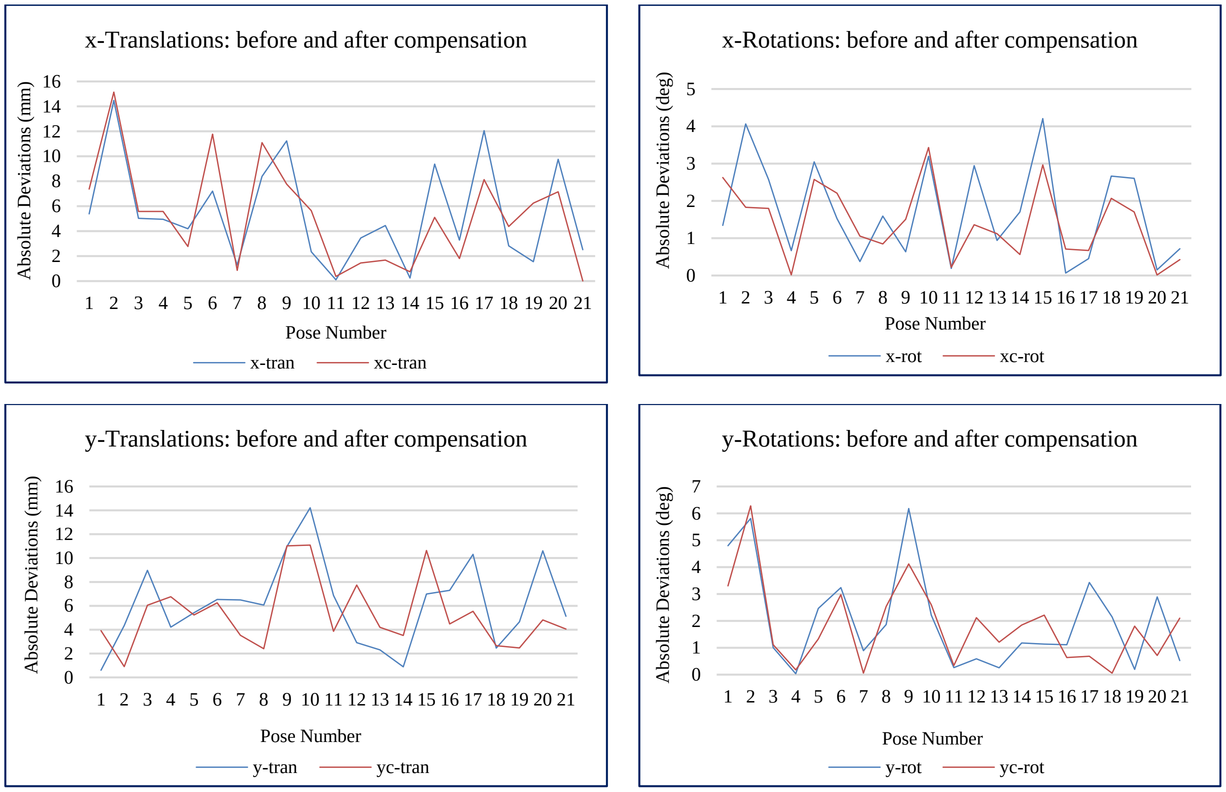
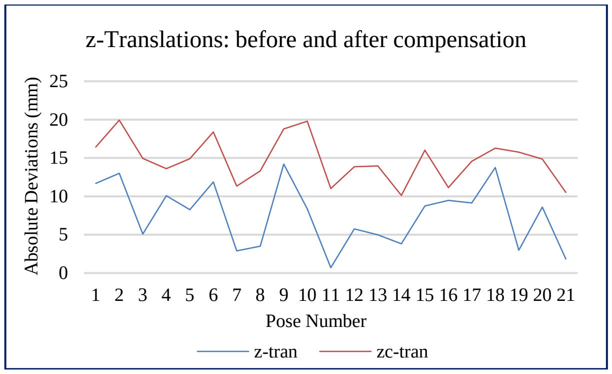


Figure 4.14: Comparison of pose parameters error range after calibration as per Option 3

A comparison of the deviation of each uncompensated and compensated pose parameter with respect to the target pose is presented in the Figure 4.15.





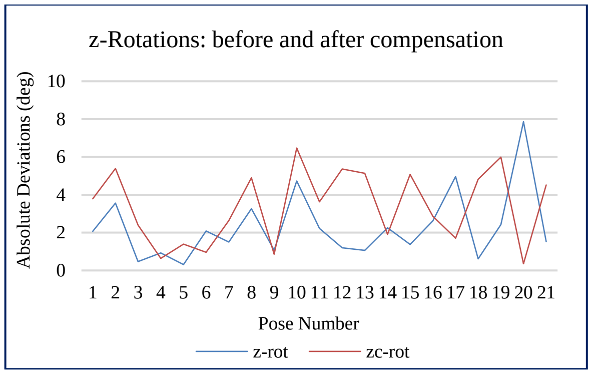


Figure 4.15: Comparison of absolute deviations of all 6 pose parameters for option 3

The comparison of magnitude of error has been captured in Table 4.4.

Table 4.4: Pose improvement after calibration as per option 3

Parameters	**x-tran**	**y-tran**	**z-tran**	**x-rot**	**y-rot**	**z-rot**
Error range-uncompensated	26.54	16.67	13.49	8.27	11.99	12.59
Magnitude of errors - uncompensated	19.70			11.12		
Error range-compensated	26.92	13.74	9.81	6.01	10.39	8.18
Magnitude of errors - compensated	18.35			8.39		
Magnitude of improvement %	**6.9%**			**24.6%**		

The magnitude of error for both position and orientation was observed to be improved and their values were 6.9% and 24.6%, respectively.

4.5 Discussions

The hexapod test-frame Tiger 66.1 was subjected to operation and calibration for the first time after its fabrication. For the first time calibration process, basic methods have been planned to start with. The Least-Square method is one of the basic statistical methods for fitting a line or curve to a set of data points to minimize the error on the data points. The above-mentioned three options have been considered as the starting point before moving to more complex compensation methods.

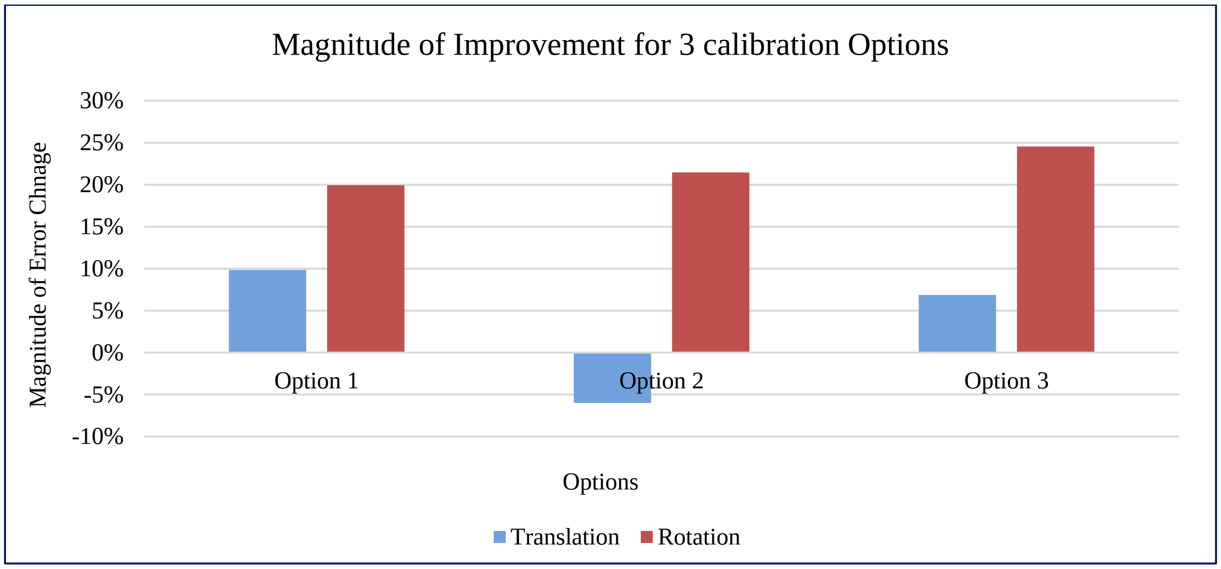


Figure 4.16: Comparison of Magnitude of Improvement for all 3 Calibration strategies

In option 1, the error minimization was done by finding the DH parameters for each pose. But it has been observed during the calculation that the same calculation methods cannot be applied for angular vectors because of the different physical units of measurements. Therefore, the position and orientation vectors of a pose have been treated differently due to the units involved. The target orientation vector used for the calibrated predicted poses without any change in the first option. In the second option the same strategy as option 1 has been followed for position and orientation vectors were compensated by least-square methods. And in the third option, the error corrections for both the vectors were done with least-square methods taking the pose parameter differences into considerations.

The three calibration strategies yielded different results and different amounts of pose-accuracy improvements. The magnitude of improvements for all three options were calculated and presented together Figure 4.16. As can be seen from the plot, option 2 shows a reduction in positional accuracy, whereas option 1 and 3 have shown improvements in both position and orientation accuracy. In option 1 the improvement in both position and orientation accuracy appears to be more balanced (the measure of improvement percentages is closer) than option 3.

When the error ranges are compared between the uncompensated pose values and compensated pose values from all three options, option 1 shows more steady changes than other two options; though the error ranges for y-position and z-position increased marginally after the calibration process based on this method. Option 3 has shown maximum error reduction except for x-position values. The error range comparisons for these options were shown in Figure 4.17.

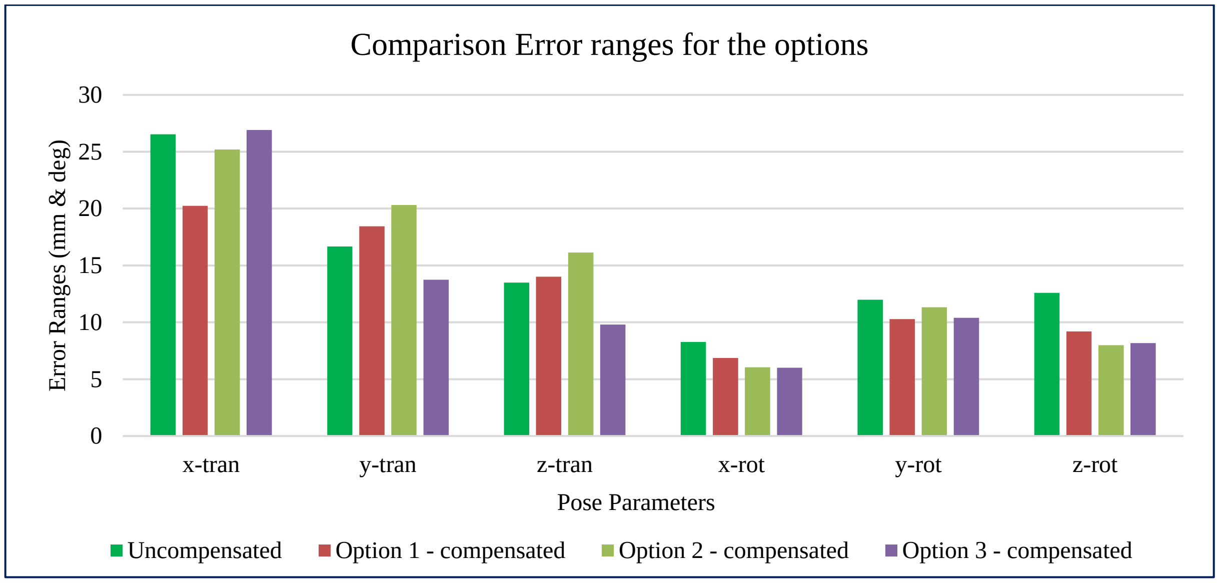


Figure 4.17: Comparison of error ranges for all options

In this research, various calibration methods were diligently applied to enhance the accuracy of hexapod poses. Despite the improvements in the hexapod poses following each of these calibration methods, none of them display a clear indication of which one is best and thereby which method is best suited for this system. It is crucial to acknowledge that there are multiple sources of errors that affect the pose accuracy of a hexapod. For this hexapod, those errors factors might have more complex relationships that is impacting the pose accuracy. These calibration methods were tried on Tiger 66.1 for the first time after it became operational. The control software has also been developed and used with its first version. While definitive conclusions regarding the optimal calibration method remain elusive, it is evident that the hexapod exhibited notable improvements in response to initial calibration processes. Among the strategies explored by the authors, option 1 appears to be a more robust calibration model. If there are any constructional errors in Tiger 66.1, those errors can be taken care of with correction of the DH parameters without going into more detailed measurement of the fabrication errors.

Another key factor to be considered in this experiment is the number of poses considered. The experiment started with 34 random poses, and it was reduced to 21 poses. The efficacy of compensation models relies on the number of data points, and it is expected that considering a greater number of poses for the experiments may help to further refine the calibration results. During the process, some of the pose points had to be eliminated from the calculations due to the newly predicted poses lying outside the workspace and out of the hexapod's motion range. Consequently, a greater number of initial random poses could increase the number of compensated valid poses that can be considered for calibration calculation and analysis. Furthermore, there is considerable merit in considering the poses most likely to be used, i.e., those that would lie along the anticipated load pathways (tension, compression, torsion, bending, and combinations thereof).

It is imperative to acknowledge the potential contribution of software and image processing errors to the calculated results. However, both software platforms employed in these calibration methods are well-established and widely adopted within industrial contexts, thereby minimizing the likelihood of software-related errors. Any residual errors can be primarily attributed to manual processing estimation. However, for image processing in this research, the error limit was restricted to below 5 pixels, which is equal to 0.425mm (for 300dpi resolution 1pixel = 0.085mm), thereby ensuring minimal impact on the overall accuracy of the calculations.

4.6 Conclusion

The target of this project was to estimate pose errors in Tiger 66.1 using a non-invasive, least instrumented approach while exploring simple calibration methods to enhance hexapod pose accuracy. The implementation of photogrammetry for pose measurements is a successful achievement in this effort. It shows promise for further development though the initial iteration yielded modest improvements. The initial controller software efficiently manages the hexapod and incorporates compensations from the three calibration methods employed. Although the accuracy gains are not substantial, these methods demonstrate the possibility of employing non-invasive photogrammetry for hexapod calibration. One of the methods used by the authors involved forward kinematics to derive a unique feasible solution and calculate DH parameters, and the Least-Squares method displayed some error reduction potential, suggesting at the possibility of investigating more complex error compensation models. This successful calibration through photogrammetry not only enhances the hexapod's overall performance but also opens avenues for its versatile deployment across various domains, ranging from industrial automation to advanced research initiatives. This simple, effective calibration process is a significant step toward achieving high quality measurements in diverse applications, thereby contributing to the progress of automation and robotics. The potential for continued refinement and innovation in this field is vast, with photogrammetry emerging as a valuable tool in enhancing the accuracy and reliability of hexapod systems and, by extension, a wide spectrum of robotic and automated processes through non-invasive and minimal instrumented methods.

5 CONCLUSION

The improvement of precision and accuracy in robotic systems has long been a prime objective in various fields, from industrial automation to advanced scientific research. The Stewart-Gough platform, with its versatility and potential for high-precision applications, has been considered a significant player in this research. However, achieving and maintaining the desired levels of accuracy in these systems poses a difficult challenge, because of the complex interactions of multiple error sources.

This dissertation has been centered on addressing this challenge by exploring and implementing a novel calibration approach for Stewart-Gough platform through the utilization of photogrammetry. The study has involved a thorough examination of theoretical formulations, practical implementation, and experimental validation of this methodology. The primary goal was to improve the absolute positioning accuracy of the moving platform center of a general Stewart-Gough platform, enhance its performance for precision applications, and advance the understanding of calibration mechanism for parallel manipulators.

The following sections provide a succinct summary of the key findings, contributions, and implications of this research.

5.1 Key Findings

The key findings from this research are as follows:

- Forward kinematics-based calibration
- Minimal hardware modification
- Cost and time efficient process

- Practical approach for implementation
- Feasibility of photogrammetry
- Expanding the scope for future research

In the next sections these key findings are discussed.

5.1.1 Forward Kinematics-based Calibration

The dissertation aimed to develop and validate a forward kinematic calibration technique. Through simulations and real-world experiments conducted on the physical Stewart-Gough platform test frame, Tiger 66.1, it was demonstrated that this method effectively reduced position errors. Vision-based measurements, in conjunction with forward kinematics, proved to be a powerful tool for calibrating the platform, offering improvements in absolute positioning accuracy of up to 25% compared with the initial system.

5.1.2 Minimal Hardware Modification

An essential feature of the proposed photogrammetry-based calibration method is its minimal hardware modification requirement. This feature not only simplifies the implementation process but also ensures that the core structure and functionality of the Stewart-Gough platform remain intact. This is particularly valuable for applications where structural alterations are costly or impractical and there is a need for continuously monitoring the accuracy of the hexapods during their operations.

5.1.3 Cost and time efficient calibration process

This research uses digital cameras and standard commercial software. There is no need for customization of the standard setup repeatedly once established. This has the potential for substantial cost and time savings by automating the entire process. The digital camera needs calibration in the beginning. Once calibrated, the camera settings remain valid for large numbers of repetitive measurements, moreover all the required software are reusable. Therefore, this process requiress minimal reinvestment of capital resources.

5.1.4 Practical Approach to Calibration

The research highlights the practicality of the photogrammetry-based calibration approach for parallel robots, particularly Stewart-Gough platform by validating it with the Tiger 66.1 system. By minimizing hardware modifications and emphasizing the use of vision-based data, this method offers a pragmatic avenue for enhancing the accuracy of these systems in high-precision applications. This is done using the advantages of the latest high-powered computing systems.

5.1.5 Contributions to Photogrammetry

Beyond its application to Stewart-Gough platform accuracy improvements, this research contributes to the broader field of photogrammetry. The development of a calibration method that employs vision data for full extrinsic calibration serves as an innovative use case for photogrammetric techniques.

5.1.6 Identification of Future Directions

The dissertation also identifies several areas for further exploration. One noteworthy avenue is the potential implementation of an online calibration method, which could continuously adapt and refine the calibration in real-time. Continuous calibration will help to maintain the desired accuracy level of the system throughout its operating period. Additionally, the study highlights the importance of investigating more complex error correction formulas to further enhance the calibration process. More details have been elaborated section 5.4.

5.2 Contributions to Knowledge

This research advances the understanding of calibration methodologies for parallel manipulators, particularly Stewart-Gough platforms. The key contributions to knowledge can be summarized as follows:

- Novel Forward Kinematics algorithm
- Integration to vision-based calibration
- Brining Tiger 66.1 into operation

5.2.1 Novel Forward Kinematics Algorithm

In this research, a groundbreaking forward kinematic algorithm has been developed. The forward kinematics solution for the Stewart platform is inherently intricate, as it consistently yields multiple feasible and unfeasible solutions due to the non-linear and polynomial nature of its kinematic equations. This inherent complexity has traditionally made it necessary to verify each solution manually for practicality, thereby limiting its real-world applicability for any subsequent operations. What sets this algorithm apart is its unique ability to consistently yield a single, feasible solution. This distinct characteristic eliminates the need for manual verification, greatly enhancing its practicality for real-world applications, a feat that previous solutions have not achieved.

Another significant output of this algorithm is the derivation of the Denavit-Hartenberg (DH) parameters for all six actuator paths. These DH parameters have been effectively utilized to enhance the accuracy of the hexapod platform's center, as demonstrated in the experiments conducted in this research. The key advantage of employing DH parameters for accuracy improvement in the hexapod platform's pose lies in the fact that individual error factors need not be measured separately. The DH parameters inherently encapsulate the effects of multiple error factors, rendering the calibration process robust, simple, and efficient.

5.2.2 Integration to vision-based calibration

The effective use of vision-based measurements in the calibration process significantly broadens the scope of photogrammetry in the fields of robotics and automation. One notable benefit of utilizing photogrammetry is that this method seamlessly integrates with the operation of the hexapod platform without any disruption. No adjustments or modifications to the hexapod platform were required during the calibration process, ensuring the safeguarding of its original construction and functionality. Consequently, the original properties of the system remain intact.

5.2.3 Bringing Tiger 66.1 into operation

Simply operationalizing a system holds limited scientific value within any field. However, when the system is purposefully designed for unique and highly specialized functions, it inherently becomes a significant contribution to that domain. Tiger 66.1 has been particularly built for the precise characterization of additively manufactured specimens under complex loading scenarios, encompassing tensile, torsional, and bending forces. Consequently, the development and successful operationalizing of this system holds great significance for the broader scientific community.

5.3 Limitations

While this research presents a promising calibration approach for Stewart-Gough Platform, it is essential to acknowledge its limitations:

- Numbers of poses considered for accuracy improvements
- Hardware compatibility
- Image quality
- Volume (space availability)

5.3.1 Number of poses considered for accuracy improvements

The measurement sample number considered for the experiment in this research is 34 to improve the accuracy of the Tiger 66.1 test frame. When more samples are included in the compensation model, the model performs better. In addition, as the measurement population increases, the statistical error correction model produces improved results.

5.3.2 Hardware Compatibility

The practicality of the photogrammetry-based approach assumes a certain level of compatibility with vision hardware and cameras. The feasibility of implementation may vary depending on the existing hardware infrastructure and compatibility of the image capturing system with the hexapod platform size. In some cases, the camera may not have sufficient capacity for capturing the required quality images of the platform pose and sometimes the operating environment around the platform would not be suitable for image capturing. This may restrict the implementation of this method.

5.3.3 Image quality

Image quality is another factor which may significantly influence the calibration process demonstrated in this research. The camera lens causes distortions in the images and the amount of distortions depends on the position of the subject with respect to the lens center and quality of the lens. So, depending upon the quality of the lens used for image capturing can significantly impact the success of the calibration process. The high quality of image capturing lens will cause less distortion of the images and thereby reduce the error in the image processing and measurement obtained from the image processing.

5.3.4 *Volume (space) availability*

The photogrammetry method is implemented without interfering with the functions of the hexapod platform and the image capturing devices are installed outside the operational periphery of the platform motion. This necessitates an extra space around the hexapod platform's operational space. Availability of such volume may sometimes restrict the implementation of this calibration method.

5.4 Future Scope of work

This research paves the way for several exciting avenues of future exploration:

- Online calibration
- Finding better, complex error models
- Integration with AI-based vision systems
- Compensation model for calibration under loaded condition
- Define poses and workspace for Tiger's application purpose

5.4.1 *Online Calibration*

The development of an online calibration method is an interesting possibility. By continuously adapting to changing conditions and improving accuracy in real-time, such a method could revolutionize the field of Stewart-Gough platform calibration. The continuous input of pose data and their further computations increase the data availability for the compensation model, thereby refining the accuracy of the system from continuous online measurements. By identifying areas for further research, such as online calibration methods and more complex error correction models, this study offers a roadmap for future investigations in the field of parallel manipulator calibration.

5.4.2 Complex Error Models

Investigating more complex error correction models represents a promising area for future research. Understanding the intricate relationships between error sources and developing advanced correction formulas can potentially lead to substantial accuracy improvements. Earlier researches showed ways to compensate errors originating from various sources, combining them together for photogrammetry-based calibration process still to be evolved and an area for future research.

5.4.3 Integration with AI-based Vision Systems

In this research, the method's success relies heavily on capturing accurate images of the platform center to generate precise 3D measurement data. Achieving the correct angle and area coverage in images is essential for this purpose. Manual image capture can be challenging and inconvenient. However, significant improvements can be made by integrating an image capture system that possesses the knowledge to determine the right image specifications for photogrammetry. By combining this image capture system with AI technology, the process can move toward automated, intelligent image capturing-based system. As vision technology evolves, the integration of advanced AI-based vision systems like depth-sensing cameras and LiDAR into the calibration process has the potential to enhance precision and expand the capabilities of Stewart-Gough platforms.

5.4.4 Compensation model for calibration under loaded condition

This research was conducted under a no-load condition, and the error compensation model was specially designed to address errors in this condition. However, it's important to note that the behavior of the platform, both in terms of errors and motion, can be significantly influenced by the type and amount of load applied. Tiger 66.1 has been designed and built for characterization of additively manufactured material specimens which can experience tensile, torsion or bending loads separately or any combination of these. This variability requires adjustments to the compensation model and the application of these compensation values to the control software, introducing complexity. Therefore, there is considerable scope for further research in this area.

5.4.5 Define poses and workspace for Tiger's application

Tailoring calibration methods to specific applications and industries could yield highly specialized and optimized calibration techniques. The poses and working space of the moving platform center may vary depending on the specific application. This is a potential area for future work in the case of Tiger 66.1. Up to this point, the research has focused on conditions without external loads. However, when it comes to material characterization, the system undergoes multiple loading conditions, and the poses and workspace volume can change significantly. Therefore, there is a need to explore calibration techniques tailored to such conditions in the future. Adapting the core concept for various applications offers a wide realm for new experiments.

5.5 Advancing Accuracy in Robotics

This dissertation explores the calibration of a Stewart-Gough platform through the innovative use of photogrammetry. The research has introduced a novel calibration method that leverages vision-based measurements to enhance the absolute positioning accuracy of the platform. With minimal hardware modification requirements, this approach offers practical solutions for industries and applications where precision is paramount.

The findings of this research have practical implications for a wide range of industries, including aerospace, manufacturing, healthcare, and more. They point to the potential for significant cost savings, improved precision, and enhanced versatility.

While this research represents a significant step forward, it is by no means the final word on Stewart-Gough platform calibration. Future research should continue to explore new avenues, such as online calibration, complex error models, and advanced vision systems integration. These efforts will contribute to the advancement of precision in robotics and automation.

It has become increasingly evident that artificial intelligence (AI) will play a pivotal role in advancing the field of robotics, including calibration processes. AI-driven technologies have the potential to revolutionize the way calibration is performed and can enhance the capabilities of the Stewart-Gough platform in unprecedented ways.

One notable avenue for AI integration is the optimization of calibration procedures. Machine learning algorithms can analyze vast datasets generated during calibration experiments, and identify patterns and relationships among various error sources. By automating the calibration process and continuously adapting to changing conditions, AI can potentially streamline and improve calibration accuracy in real-time, thereby reducing the need for manual intervention.

Furthermore, AI can facilitate predictive maintenance for Stewart-Gough platform, alerting operators to potential issues before they impact performance. By analyzing sensor data and historical performance metrics, AI algorithms can detect early signs of wear, fatigue, or component degradation, enabling proactive maintenance and preventing costly downtimes.

Additionally, AI-powered vision systems can enhance the precision and speed of photogrammetry-based calibration. Advanced computer vision algorithms can automatically identify calibration markers, track feature points in real-time, and even perform image processing tasks with exceptional accuracy. This not only reduces the reliance on human intervention but also enhances the overall robustness of the calibration process.

As the processes navigate through this exciting intersection of AI and robotics, it is imperative that researchers and engineers continue to explore and harness the full potential of AI in calibration and beyond. The fusion of these technologies holds the key to further advancements in robotics, automation, and high-precision applications, ultimately propelling us into a future where precision knows no bounds.

In the grand scheme of things, the pursuit of precision & accuracy in robotics is an ongoing journey. With each step forward, it brings processes closer to unlocking new possibilities in industries and applications that rely on highly accurate robotic systems. In the coming days, the quest for precision will continue to drive innovation and shape the future of robotics.

***

6 REFERENCES

[1] A. Ríos, E. E. Hernández, and S. I. Valdez, "A two-stage mono-and multi-objective method for the optimization of general ups parallel manipulators," *Mathematics*, vol. 9, no. 5, pp. 1–20, 2021, doi: 10.3390/math9050543.

[2] K. H. Harib, A. M. M. Sharif Ullah, and A. Hammami, "A hexapod-based machine tool with hybrid structure: Kinematic analysis and trajectory planning," *Int. J. Mach. Tools Manuf.*, vol. 47, no. 9, pp. 1426–1432, 2007, doi: 10.1016/j.ijmachtools.2006.09.021.

[3] H. Shi, Y. She, and X. Duan, "Modeling and measurement algorithm of hexapod platform sensor using inverse kinematics," *2015 Aust. Control Conf. AUCC 2015*, pp. 331–335, 2015.

[4] Zoran PANDILOV and K. Rall, "Parallel Kinematics Machine Tools : Overview - From History To the Future," *Mech. Eng. Sci. J.*, vol. 25, no. 1, pp. 3–20, 2012, [Online]. Available: http://annals.fih.upt.ro/pdf-full/2012/ANNALS-2012-2-16.pdf http://www.mesj.ukim.edu.mk/sites/default/files/Mech-Eng-25-1-2006_0.pdf

[5] M. Jouini, M. Sassi, A. Sellami, and N. Amara, "Modeling and control for a 6-DOF platform manipulator," *2013 Int. Conf. Electr. Eng. Softw. Appl. ICEESA 2013*, 2013, doi: 10.1109/ICEESA.2013.6578432.

[6] V. E. Gough, "Contribution to discussion to papers on research in automobile stability and control and in tire performance," *Proc. Auto Div. Inst. Mech. Eng.*, vol. 171, pp. 392–397, 1957, [Online]. Available: http://ci.nii.ac.jp/naid/10030209099/en/

[7] D. Stewart, "A Platform with Six Degrees of Freedom," *Proc. Inst. Mech. Eng.*, vol. 180, no. 1, pp. 371–386, 1965, doi: 10.1243/pime_proc_1965_180_029_02.

[8] J. F. Wu, R. Zhang, R. H. Wang, and Y. X. Yao, "A systematic optimization approach for the calibration of parallel kinematics machine tools by a laser tracker," *Int. J. Mach. Tools Manuf.*, vol. 86, pp. 1–11, 2014, doi: 10.1016/j.ijmachtools.2014.06.003.

[9] M. Mazare, M. Taghizadeh, and M. Rasool Najafi, "Kinematic analysis and design of a 3-DOF translational parallel robot," *Int. J. Autom. Comput.*, vol. 14, no. 4, pp. 432–441, 2017, doi: 10.1007/s11633-017-1066-y.

[10] A. J. Patel and K. F. Ehmann, "Calibration of a hexapod machine tool using a redundant leg," *Int. J. Mach. Tools Manuf.*, vol. 40, no. 4, pp. 489–512, 2000, doi: 10.1016/S0890-6955(99)00081-4.

[11] Y. D. Patel and P. M. George, "Parallel Manipulators Applications—A Survey," *Mod. Mech. Eng.*, vol. 02, no. 03, pp. 57–64, 2012, doi: 10.4236/mme.2012.23008.

[12] Physik Instrumente (PI) GmbH & Co KG., "Hexapod Parallel Robots Automate Highly Precise Production Processes," *AZoNano*, pp. 1–17, 2018, [Online]. Available: https://www.azonano.com/article.aspx?ArticleID=4524

[13] A. C. Majarena, J. Santolaria, D. Samper, and J. J. Aguilar, "An overview of kinematic and calibration models using internal/external sensors or constraints to improve the behavior of spatial parallel mechanisms," *Sensors (Switzerland)*, vol. 10, no. 11, pp. 10256–10297, 2010, doi: 10.3390/s101110256.

[14] L. Tsai, "Robot analysis: the mechanics of serial and parallel manipulators," *Wiley, New York*. p. 520, 1999.

[15] T. Y. Lee and J. K. Shim, “Forward kinematics of the general 6-6 Stewart platform using algebraic elimination,” *Mech. Mach. Theory*, vol. 36, no. 9, pp. 1073–1085, 2001, doi: 10.1016/S0094-114X(01)00034-9.

[16] J.-P. Merlet, “Still a long way to go on the road for parallel mechanisms,” *Asme*, vol. 64, no. May, pp. 1–19, 2002.

[17] D. Daney, N. Andreff, G. Chabert, and Y. Papegay, “Interval method for calibration of parallel robots: Vision-based experiments,” *Mech. Mach. Theory*, vol. 41, no. 8, pp. 929–944, 2006, doi: 10.1016/j.mechmachtheory.2006.03.014.

[18] J. C. Jáuregui, E. E. Hernández, M. Ceccarelli, C. López-Cajún, and A. García, “Kinematic calibration of precise 6-DOF Stewart platform-type positioning systems for radio telescope applications,” *Front. Mech. Eng.*, vol. 8, no. 3, pp. 252–260, 2013, doi: 10.1007/s11465-013-0249-7.

[19] J. C. Ziegert, B. Jokiel, and C.-C. Huang, “Calibration and Self-Calibration of Hexapod Machine Tools,” *Parallel Kinematic Mach. © Springer-Verlag London Ltd. 1999*, pp. 205–216, 1999, doi: 10.1007/978-1-4471-0885-6_13.

[20] L. J. Everett, M. Driels, and B. W. Mooring, “Kinematic Modelling for Robot Calibration.,” *IEEE*, pp. 183–189, 1987.

[21] Y. D. and L. F. A.Y. Elatta, Li Pei Gen , Fan Liang Zhi, “An Overview of Robot calibration,” *Inf. Technol. J. 3*, pp. 74–78, 2004.

[22] Y. Meng and H. Zhuang, “Autonomous robot calibration using vision technology,” *Robot. Comput. Integr. Manuf.*, vol. 23, no. 4, pp. 436–446, 2007, doi: 10.1016/j.rcim.2006.05.002.

[23] A. Nubiola and I. A. Bonev, “Absolute robot calibration with a single telescoping ballbar,” *Precis. Eng.*, vol. 38, no. 3, pp. 472–480, 2014, doi: 10.1016/j.precisioneng.2014.01.001.

[24] C. Szep, S. D. Stan, V. Csibi, M. Manic, and R. Bălan, “Kinematics, workspace, design and accuracy analysis of RPRPR medical parallel robot,” *Proc. - 2009 2nd Conf. Hum. Syst. Interact. HSI ’09*, pp. 75–80, 2009, doi: 10.1109/HSI.2009.5090957.

[25] F. Bleicher, F. Puschitz, and A. Theiner, “Laser based measurment system for calibrating machine tools in 6 dof,” *Ann. DAAAM Proc. Int. DAAAM Symp.*, pp. 39–40, 2006.

[26] R. Ramesh, M. . Mannan, and A. . Poo, “Error compensation in machine tools — a review. Part I,” *Int. J. Mach. Tools Manuf.*, vol. 40, no. 9, pp. 1257–1284, Jul. 2000, doi: 10.1016/S0890-6955(00)00010-9.

[27] S. Szatmári, “Geometrical errors of parallel robots,” *Period. Polytech. Mech. Eng.*, vol. 43, no. 2, pp. 155–162, 1999.

[28] D. Daney, “Optimal measurement configurations for Gough platform calibration,” *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 1, no. May, pp. 147–152, 2002, doi: 10.1109/robot.2002.1013353.

[29] J. A. Soons, “Error analysis of a hexapod machine tool,” *Trans. Eng. Sci.*, vol. 16, pp. 347–358, 1997.

[30] A. Traslosheros, J. M. Sebastián, J. Torrijos, R. Carelli, and E. Castillo, “An inexpensive method for kinematic calibration of a parallel robot by using one hand-held camera as main sensor,” *Sensors (Switzerland)*, vol. 13, no. 8, pp. 9941–9965, 2013, doi: 10.3390/s130809941.

[31] J.-P. Merlet, “Parallel Robot,” 2006, [Online]. Available: www.springer.com

[32] J. Ryu and A. Rauf, “A new method for fully autonomous calibration of parallel

manipulators using a constraint link," *IEEE/ASME Int. Conf. Adv. Intell. Mechatronics, AIM*, vol. 1, no. July, pp. 141–146, 2001, doi: 10.1109/aim.2001.936444.

[33] A. Traslosheros, J. M. Sebastián, E. Castillo, F. Roberti, and R. Carelli, "A method for kinematic calibration of a parallel robot by using one camera in hand and a spherical object," *IEEE 15th Int. Conf. Adv. Robot. New Boundaries Robot. ICAR 2011*, pp. 75–81, 2011, doi: 10.1109/ICAR.2011.6088594.

[34] J. Guo, D. Wang, R. Fan, W. Chen, and G. Zhao, "Development of a material testing machine with multi-dimensional loading capability," *J. Adv. Mech. Des. Syst. Manuf.*, vol. 10, no. 2, pp. 1–14, 2016, doi: 10.1299/jamdsm.2016jamdsm0017.

[35] A. Mura, "Six d.o.f. displacement measuring device based on a modified Stewart platform," *Mechatronics*, vol. 21, no. 8, pp. 1309–1316, 2011, doi: 10.1016/j.mechatronics.2011.09.001.

[36] X. D. Ren, Z. R. Feng, and C. P. Su, "A new calibration method for parallel kinematics machine tools using orientation constraint," *Int. J. Mach. Tools Manuf.*, vol. 49, no. 9, pp. 708–721, 2009, doi: 10.1016/j.ijmachtools.2009.03.004.

[37] T. Huang, J. Wang, D. G. Chetwynd, and D. J. Whitehouse, "Identifiability of geometric parameters of 6-DOF PKM systems using a minimum set of pose error data," *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 2, no. 1, pp. 1863–1868, 2003, doi: 10.1109/robot.2003.1241866.

[38] H. Zou and L. Notash, "Discussions on the camera-aided calibration of parallel manipulators," *Proc. 2001 CCToMM Symp. Mech. Mach. Mechatronic, Saint-Hubert*, pp. 3–4, 2001.

[39] A. Olarra, D. Axinte, and G. Kortaberria, "Geometrical calibration and uncertainty estimation methodology for a novel self-propelled miniature robotic machine tool," *Robot. Comput. Integr. Manuf.*, vol. 49, no. January 2017, pp. 204–214, 2018, doi: 10.1016/j.rcim.2017.06.011.

[40] H. Zhuang, J. Yan, and O. Masory, "Calibration of Stewart platforms and other parallel manipulators by minimizing inverse kinematic residuals," *J. Robot. Syst.*, vol. 15, no. 7, pp. 395–405, 1998, doi: 10.1002/(SICI)1097-4563(199807)15:7<395::AID-ROB2>3.0.CO;2-H.

[41] J. Santolaria and M. Ginés, "Uncertainty estimation in robot kinematic calibration," *Robot. Comput. Integr. Manuf.*, vol. 29, no. 2, pp. 370–384, 2013, doi: 10.1016/j.rcim.2012.09.007.

[42] P. Vischer and R. Clavel, "Kinematic calibration of the parallel Delta robot," *Robotica*, vol. 16, no. 2, pp. 207–218, 1998, doi: 10.1017/S0263574798000538.

[43] X. Jing, Y. Fang, and Z. Wang, "A Calibration Method for 6-UPS Stewart Platform," *W. Proc. 2019 Chinese Intell. Syst. Conf. CISC 2019, Springer, Singapore.*, vol. 593, no. Lecture Notes in Electrical Engineering, pp. 513–519, 2020, doi: https://doi.org/10.1007/978-981-32-9686-2_58.

[44] M. Agheli and M. Nategh, "Identifying the Kinematic Parameters of Hexapod Machine Tool.," *World Acad. Sci. Eng. Technol.*, vol. 52, pp. 380–385, 2009,

[45] D. Daney and I. Z. Emiris, "Algebraic Elimination for Parallel Robot Calibration," *Proc. 11 World Congr. Mech. Mach. Sci. Tianjin China*, 2004, [Online].

[46] D. Daney, Y. Papegay, and A. Neumaier, "Interval methods for certification of the kinematic calibration of parallel robots," *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 2004, no. 2, pp. 1913–1918, 2004, doi: 10.1109/robot.2004.1308103.

[47] S. M. Wang and K. F. Ehmann, “Error model and accuracy analysis of a six-DOF Stewart Platform,” *J. Manuf. Sci. Eng. Trans. ASME*, vol. 124, no. 2, pp. 286–295, 2002, doi: 10.1115/1.1445148.

[48] D. Daney and I. Emiris, “Variable elimination for reliable parallel robot calibration,” *2nd Work. Comput. Kinemat. - CK’2001 12 p*, p. 13, 2001, [Online]. Available: http://www-sop.inria.fr/coprin/EJCK/Vol1-1/13_daneya.ps

[49] Y. Song, W. Tian, Y. Tian, and X. Liu, “Calibration of a Stewart platform by designing a robust joint compensator with artificial neural networks,” *Precis. Eng.*, vol. 77, no. June, pp. 375–384, 2022, doi: 10.1016/j.precisioneng.2022.07.001.

[50] A. Mahmoodi, A. Sayadi, and M. B. Menhaj, “Solution of forward kinematics in Stewart platform using six rotary sensors on joints of three legs,” *Adv. Robot.*, vol. 28, no. 1, pp. 27–37, 2014, doi: 10.1080/01691864.2013.854455.

[51] M. J. Nategh and M. M. Agheli, “A total solution to kinematic calibration of hexapod machine tools with a minimum number of measurement configurations and superior accuracies,” *Int. J. Mach. Tools Manuf.*, vol. 49, no. 15, pp. 1155–1164, 2009, doi: 10.1016/j.ijmachtools.2009.08.009.

[52] K. Großmann, B. Kauschinger, and S. Szatmári, “Kinematic calibration of a hexapod of simple design,” *Prod. Eng.*, vol. 2, no. 3, pp. 317–325, 2008, doi: 10.1007/s11740-008-0092-6.

[53] Y. Liu, B. Liang, C. Li, L. Xue, S. Hu, and Y. Jiang, “Calibration of a Steward parallel robot using genetic algorithm,” *Proc. 2007 IEEE Int. Conf. Mechatronics Autom. ICMA 2007*, pp. 2495–2500, 2007, doi: 10.1109/ICMA.2007.4303948.

[54] Y. Ting, H. C. Jar, and C. C. Li, “Measurement and calibration for Stewart micromanipulation system,” *Precis. Eng.*, vol. 31, no. 3, pp. 226–233, 2007, doi: 10.1016/j.precisioneng.2006.09.004.

[55] T. Dallej, H. Hadj-Abdelkader, N. Andreff, and P. Martinet, “Kinematic Calibration of a Gough-Stewart platform using an omnidirectional camera,” *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 4666–4671, 2006, doi: 10.1109/IROS.2006.282253.

[56] D. Daney, Y. Papegay, and B. Madeline, “Choosing measurement poses for robot calibration with the local convergence method and Tabu search,” *Int. J. Rob. Res.*, vol. 24, no. 6, pp. 501–518, 2005, doi: 10.1177/0278364905053185.

[57] M. Gao, T. Li, and W. Yin, “Calibration method and experiment of Stewart platform using a laser tracker,” *Proc. IEEE Int. Conf. Syst. Man Cybern.*, vol. 3, pp. 2797–2802, 2003, doi: 10.1109/icsmc.2003.1244309.

[58] P. Renaud, N. Andreff, M. Dhome, and P. Martinet, “Experimental evaluation of a vision-based measuring device for parallel machine-tool calibration,” *IEEE Int. Conf. Intell. Robot. Syst.*, vol. 2, no. October, pp. 1868–1873, 2002, doi: 10.1109/irds.2002.1044028.

[59] M. Week and D. Staimer, “Accuracy issues of parallel kinematic machine tools,” *Proc. Inst. Mech. Eng. Part K J. Multi-body Dyn.*, vol. 216, no. 1, pp. 51–57, 2002, doi: 10.1243/146441902760029384.

[60] Y. Ihara, T. Ishida, Y. Kakino, Z. Li, T. Matsushita, and M. Nakagawa, “Kinematic calibration of a hexapod machine tool by using circular test,” *Proc. 2000 Japan•USA Flex. Autom. Conf. , Ann Arbor, Michigan*, no. January, pp. 1–4, 2000, [Online].

[61] A. Rauf and J. Ryu, “Fully autonomous calibration of parallel manipulators by imposing position constraint,” *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 3, pp. 2389–2394, 2001, doi: 10.1109/robot.2001.932979.

[62] Y. J. Chiu and M. H. Perng, “Self-calibration of a general hexapod manipulator using cylinder constraints,” *Int. J. Mach. Tools Manuf.*, vol. 43, no. 10, pp. 1051–1066, 2003, doi: 10.1016/S0890-6955(03)00082-8.

[63] H. Zhuang, L. Liu, and O. Masory, “Autonomous calibration of hexapod machine tools,” *J. Manuf. Sci. Eng. Trans. ASME*, vol. 122, no. 1, pp. 140–148, 2000, doi: 10.1115/1.538893.

[64] B. Dasgupta and T. S. Mruthyunjaya, “A constructive predictor-corrector algorithm for the direct position kinematics problem for a general 6-6 Stewart platform,” *Mech. Mach. Theory*, vol. 31, no. 6, pp. 799–811, 1996, doi: 10.1016/0094-114X(95)00106-9.

[65] W. Tanaka, T. Arai, K. Inoue, Y. Mae, and C. S. Park, “Simplified kinematic calibration for a class of parallel mechanism,” *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 1, no. May, pp. 483–488, 2002, doi: 10.1109/ROBOT.2002.1013406.

[66] D. Daney, I. Z. Emiris, Y. Papegay, E. Tsigaridas, and J. P. Merlet, “Calibration of parallel robots : on the Elimination of Pose-Dependent Parameters,” *EuCoMeS 2006 - 1st Eur. Conf. Mech. Sci. Conf. Proc.*, pp. 1–12, 2006.

[67] B. Dasgupta and T. S. Mruthyunjaya, “Closed-form dynamic equations of the general Stewart platform through the Newton-Euler approach,” *Mech. Mach. Theory*, vol. 33, no. 7, pp. 993–1012, 1998, doi: 10.1016/S0094-114X(97)00087-6.

[68] D. Jakobović and L. Jelenković, “The forward and inverse kinematics problems for stewart parallel mechanisms,” *CIM 2002 Comput. Integr. Manuf. High Speed Mach. - 8th Int. Sci. Conf. Prod. Eng.*, 2002.

[69] C. Innocenti and V. Parenti-Castelli, “Direct position analysis of the Stewart platform mechanism,” *Mech. Mach. Theory*, vol. 25, no. 6, pp. 611–621, 1990, doi: 10.1016/0094-114X(90)90004-4.

[70] X. Huang, Q. Liao, S. Wei, X. Qiang, and S. Huang, “Forward kinematics of the 6-6 Stewart platform with planar base and platform using algebraic elimination,” *Proc. IEEE Int. Conf. Autom. Logist. ICAL 2007*, no. 104043, pp. 2655–2659, 2007, doi: 10.1109/ICAL.2007.4339029.

[71] M. L. Husty, “An algorithm for solving the direct kinematics of general Stewart-Gough platforms,” *Mech. Mach. Theory*, vol. 31, no. 4, pp. 365–379, 1996, doi: 10.1016/0094-114X(95)00091-C.

[72] Y. Wang, “A direct numerical solution to forward kinematics of general Stewart-Gough platforms,” *Robotica*, vol. 25, no. 1, pp. 121–128, 2007, doi: 10.1017/S0263574706003080.

[73] M. Cardona, “Kinematics and Jacobian analysis of a 6UPS Stewart-Gough platform,” *2016 IEEE 36th Cent. Am. Panama Conv. CONCAPAN 2016*, 2016, doi: 10.1109/CONCAPAN.2016.7942377.

[74] H. Zhang, Q. Gao, M. Zhang, and Y. Yao, “Forward kinematics of parallel robot based on neural network Newton-Raphson iterative algorithm,” no. December 2021, p. 36, 2021, doi: 10.1117/12.2625254.

[75] C. Yang, Q. Huang, P. O. Ogbobe, and J. Han, “Forward kinematics analysis of parallel robots using global Newton-Raphson method,” *2009 2nd Int. Conf. Intell. Comput. Technol. Autom. ICICTA 2009*, vol. 3, pp. 407–410, 2009, doi: 10.1109/ICICTA.2009.564.

[76] X. Zhou, F. Zhou, and Y. Wang, “Pose Error Modeling and Analysis for 6-DOF Stewart Platform,” 2017. doi: doi: 10.1109/CAC.2017.8243943.

[77] M. Tarokh, "Real time forward kinematics solutions for general Stewart platforms," *Proc. - IEEE Int. Conf. Robot. Autom.*, no. April, pp. 901–906, 2007, doi: 10.1109/ROBOT.2007.363100.

[78] J. M. Porta and F. Thomas, "Yet Another Approach to the Gough-Stewart Platform Forward Kinematics," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 974–980, 2018, doi: 10.1109/ICRA.2018.8460900.

[79] R. KumarP and and BBandyopadhyay, *The Forward Kinematic Modeling of a Stewart Platform using NLARX Model with Wavelet Network*. 2013. doi: 10.1109/INDIN.2013.6622907.

[80] N. Fazenda, E. Lubrano, S. Rossopoulos, and R. Clavel, "Calibration of the 6 DOF high-precision flexure parallel robot 'Sigma 6,'" *Proc. 5th Parallel Kinemat. Semin. Chemnitz, Ger.*, pp. 379–398, 2006, [Online]. Available: https://infoscience.epfl.ch/record/117939

[81] J. C. Ziegert and P. Kalle, "Error compensation in machine tools: a neural network approach," *J. Intell. Manuf.*, vol. 5, no. 3, pp. 143–151, 1994, doi: 10.1007/BF00123919.

[82] V. Schmidt, B. Müller, and A. Pott, "Solving the forward kinematics of cable-driven parallel robots with neural networks and interval arithmetic," *Mech. Mach. Sci.*, vol. 15, pp. 103–110, 2014, doi: 10.1007/978-94-007-7214-4_12.

[83] A. Morell, L. Acosta, and J. Toledo, "An artificial intelligence approach to forward kinematics of Stewart Platforms," *2012 20th Mediterr. Conf. Control Autom. MED 2012 - Conf. Proc.*, pp. 433–438, 2012, doi: 10.1109/MED.2012.6265676.

[84] Y. K. Yiu, J. Meng, and Z. X. Li, "Auto-calibration for a parallel manipulator with sensor redundancy," *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 3, pp. 3660–3665, 2003, doi: 10.1109/robot.2003.1242158.

[85] J. Michaloski, "Coordinated Joint Motion Control for an Industrial Robot," *NIST Interagency/Internal Rep. (NISTIR), Natl. Inst. Stand. Technol. Gaithersburg, MD,* no. March, 1988, [Online]. Available: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=820236

[86] T. Charters, R. Enguiça, and P. Freitas, "Detecting Singularities of Stewart Platforms," *Ind. Case Stud. J.*, vol. 1, pp. 66–80, 2009, [Online]. Available: https://cdn.instructables.com/ORIG/FFK/LAIV/I55MRG6M/FFKLAIVI55MRG6M.pdf

[87] B. Li, Y. Cao, Q. Zhang, and Z. Huang, "Position-singularity analysis of a special class of the Stewart parallel mechanisms with two dissimilar semi-symmetrical hexagons," *Robotica*, vol. 31, no. 1, pp. 123–136, 2013, doi: 10.1017/S0263574712000148.

[88] J. Denavit and R. S. Hartenberg, "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices," *Journal of Applied Mechanics*, vol. 22, no. 2. pp. 215–221, 1955. doi: 10.1115/1.4011045.

[89] M. Granja, N. Chang, V. Granja, M. Duque, and F. Llulluna, "Comparison between standard and modified Denavit-Hartenberg Methods in Robotics Modelling," *Proc. World Congr. Mech. Chem. Mater. Eng.*, vol. 1, no. 1, pp. 1–10, 2016, doi: 10.11159/icmie16.118.

[90] John J. Craig, "Introduction to Robotics: Mechanics and Control," *Pearson Prentice Hall*, no. 3e, p. 408, 2004.

[91] S. Karmakar, A. Patel, and C. J. Turner, "Calibration of parallel kinematic machine based on stewart platform - A literature review," in *Proceedings of the ASME Design Engineering Technical Conference*, 2021. doi: 10.1115/DETC2021-71619.

[92] D. Jakobovi, L. Budin, D. Jakobović, and L. Budin, "Forward kinematics of a Stewart

platform mechanism," *J. Mech. Des.*, vol. 115, no. 4, pp. 277–282, 2002, [Online]. Available: https://bib.irb.hr/datoteka/89476.ines2002.pdf%0Ahttp://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.13.580&rep=rep1&type=pdf

[93] S. Kucuk and Z. Bingul, *Robot Kinematics: Forward and Inverse Kinematics*, no. December. 2006. doi: 10.5772/5015.

[94] S. Karmakar and C. J. Turner, "Forward kinematics solution for a general Stewart platform through iteration based simulation," *Int. J. Adv. Manuf. Technol.*, no. 0123456789, 2023, doi: 10.1007/s00170-023-11130-9.

[95] S. T. Fry and C. J. Turner, "Design of a Stewart-Gough Platform for Engineering Material Characterization," *Proc. ASME 2016 Int. Mech. Eng. Congr. Expo.*, pp. 1–9, 2016, doi: 10.1115/imece2016-66669.

[96] Wikipedia, "Least-Square method." Wikimedia Foundation, 2023. [Online]. Available: en.wikipedia.org/wiki/Least_squares_method.

[97] S. Nebel, M. Beege, S. Schneider, and G. D. Rey, "A Review of Photogrammetry and Photorealistic 3D Models in Education From a Psychological Perspective," *Front. Educ.*, vol. 5, no. August, pp. 1–15, 2020, doi: 10.3389/feduc.2020.00144.

[98] A. R. Mosbrucker, J. J. Major, K. R. Spicer, and J. Pitlick, "Camera system considerations for geomorphic applications of SfM photogrammetry," *Earth Surf. Process. Landforms*, vol. 42, no. 6, pp. 969–986, 2017, doi: 10.1002/esp.4066.

[99] M. A. Elhalawani, Z. M. Zeidan, and A. A. A. Beshr, "Implementation of close range photogrammetry using modern non-metric digital cameras for architectural documentation," *Geod. Cartogr.*, vol. 47, no. 1, pp. 45–53, 2021, doi: 10.3846/gac.2021.11269.

7 APPENDIX

7.1 Electrical circuit layout

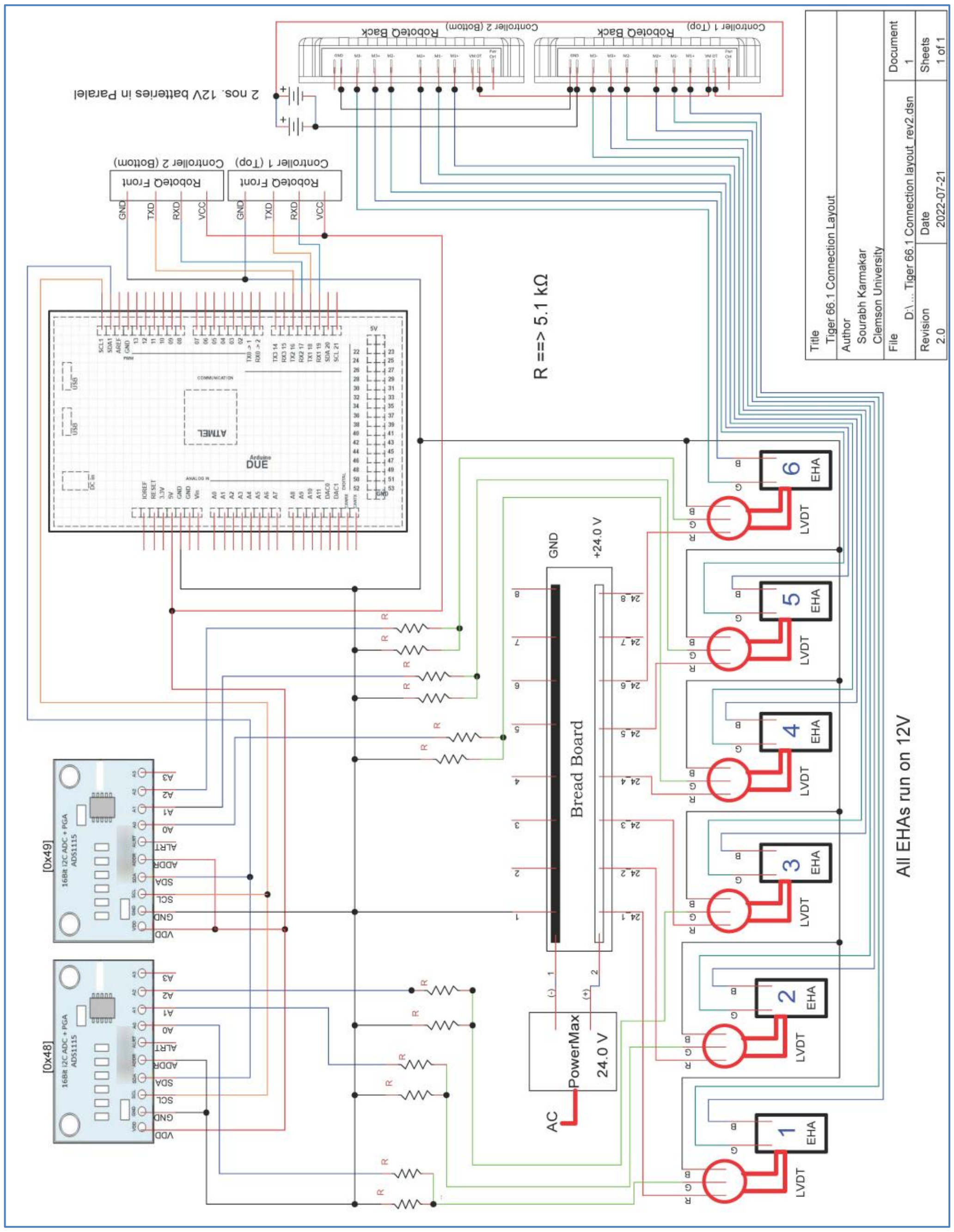


Figure 7.1: Electrical circuit layout

7.2 Arduino codes

The code below is the main control code for Tiger 66.1.

File name: Tiger_66.1_controller.ino

```
// File name: Tiger_66.1_controller.ino
// Created by: SOURABH KARMAKAR
// Creation date: 2021-03-16
// Description: Main code for controlling Tiger 66.1 by Arduino
// Following two Library files are needed for this code to work:
// Adafruit_ADS1X15, MatrixMath

// rev4 : base angle distribution changed as per actual test frame layout
// rev5 : base and platform angle distributions separated and rotation motion
about z-axis is now working

#include <MatrixMath.h>
#include <Wire.h>
#include <Adafruit_ADS1015.h>

#define n (4)

double Tx, Ty, Tz, Rx, Ry, Rz; // User input for new platform center pose

double p_radius = 225.14; // platform joint radius
double b_radius = 477.39; // base joint radius
double to_rad = (2 * PI / 360); // factor to convert deg to rad

// Base joint angles -- from matlab: 6.69, 53.31, 126.69, 173.31, 246.69,
293.31
double angb_rad[6] = { 6.69 * to_rad, 53.31 * to_rad, 126.69 * to_rad,
 173.31 * to_rad, 246.69 * to_rad, 293.31 * to_rad }; // angles for base
joints in home position in radians

// Platform joint angles -- from matlab: 353.38, 66.53, 113.38, 186.53,
233.38, 306.53
double angp_rad[6] = { 353.38 * to_rad, 66.53 * to_rad, 113.38 * to_rad,
 186.53 * to_rad, 233.38 * to_rad, 306.53 * to_rad }; // angles for platform
joints in home position in radians

// limits for translations and rotations in x, y & z
// all linear dimensions are in mm and angles are in degrees unless specified
double x_min = -55.0;
double x_max = 55.0;
double y_min = -55.0;
double y_max = 55.0;
double z_min = -52.5;
double z_max = 0;
double roll_min = -9.0;
double roll_max = 9.0;
double pitch_min = -9.0;
double pitch_max = 9.0;
double yaw_min = -25;
double yaw_max = 25;
double hm_act_len = 96; // actuator length in platform home position, i.e. @
z = 0
double pos_tol = 0.5; // Platform pose tolerance 0.5, i.e +/- 0.5
double hm_pos_upper = hm_act_len + pos_tol;
double hm_pos_lower = hm_act_len - pos_tol;

// actuator length limits : original
double act_len_max_org = 668.02; // = 305.4 + 203 + 83.42 + 76.2; 83.42 mm is
joint dist from top hole center of actuator; 203 mm stroke length
double act_len_min_org = 465.02; // = 305.4 + 83.42 + 76.2; 76.2 mm is joint
dist from bottom hole center of actuator
double act_len_max = 643.02; // 25 mm reduced to be on the safer side. To be
verified and revised.
double act_len_min = 490.02; // 25 mm increased to be on the safer side. To
be verified and revised.

// Initializize ADS1115
Adafruit_ADS1115 ads1(0x48);
Adafruit_ADS1115 ads2(0x49);

int ans = 0; // answer from user
String userAns;
int usrAns;

// EHAs are Initializized without non-zero value
// EHAs are Speed for each actuator
int EHA1 = 1;
int EHA2 = 1;
int EHA3 = 1;
int EHA4 = 1;
int EHA5 = 1;
int EHA6 = 1;
double diff1, diff2, diff3, diff4, diff5, diff6; // difference between
existing and target actuator lengths

int count = 0; // count for the loop

// Variables for writing controller status
String inString1 = "", inString2 = "";
boolean stringComplete1 = false, stringComplete2 = false;
int Controller1_status, Controller2_status;

// Array for controller variables, Actuator lengths variable and voltage
variables
int Controller_status[2];
double Length[6];
double Voltage[6];

// Target actuator lengths
double inputD[6];

// New actuator lengths
double nLeg[6];

// STARTING MATRIX CALCULATIONS FOR LEG LENGTH
// Transformation matrices:
// hwbc -> transformation matrix for base c (local) wrt to world,
// hwpc -> transformation matrix for pc (local) wrt to world,
// htm -> homogeneous transfotmation matrix for new Tx, Ty, Tz, Rx, Ry, Rz
from home pose
// ALL CALCULATIONS ARE DONE wtr TO WORLD FRAME

mtx_type wba1[n][1], wba2[n][1], wba3[n][1], wba4[n][1], wba5[n][1],
wba6[n][1]; // base joint coordinates wrt to World @top home Pose
mtx_type wpa1[n][1], wpa2[n][1], wpa3[n][1], wpa4[n][1], wpa5[n][1],
wpa6[n][1]; // Platform joint coordinates wrt to World @top home Pose
mtx_type gpa1[n][1], gpa2[n][1], gpa3[n][1], gpa4[n][1], gpa5[n][1],
gpa6[n][1]; // Platform joint coordinates wrt to grip center @top home Pose

mtx_type wgc[n][1] = { { 0 }, { 0 }, { 885.21 }, { 1 } }; // height of moving
grip top center wrt world frame
mtx_type n_wgc[n][1], n_ggc[n][1];

mtx_type n_gpa1[n][1], n_gpa2[n][1], n_gpa3[n][1], n_gpa4[n][1],
n_gpa5[n][1], n_gpa6[n][1]; // New platform joint locations wrt moving grip
center
mtx_type n_wpa1[n][1], n_wpa2[n][1], n_wpa3[n][1], n_wpa4[n][1],
n_wpa5[n][1], n_wpa6[n][1]; // New platform joint locations wrt world center

// dimensions at upper home pose, i.e. grippers are gaugue distance (50mm)
double gp = -318.22; // moving grip center to platform center
double gb = -813.66; // moving grip center to base center
double gw = -885.21; // moving grip center to world center
double bp = 495.44; // base to plaform center
double wb = 71.55; // world center to base center
double wp = wb + bp; // world center to platform center

// assign values to the base and platform joints for home pose
// mtx_type is redefining double data type under MatrixMath.h header

// matrices to define base joints
mtx_type ba1[n][1] = { { b_radius * cos(angb_rad[0]) }, { b_radius *
sin(angb_rad[0]) }, { 0 }, { 1 } };
mtx_type ba2[n][1] = { { b_radius * cos(angb_rad[1]) }, { b_radius *
sin(angb_rad[1]) }, { 0 }, { 1 } };
mtx_type ba3[n][1] = { { b_radius * cos(angb_rad[2]) }, { b_radius *
sin(angb_rad[2]) }, { 0 }, { 1 } };
mtx_type ba4[n][1] = { { b_radius * cos(angb_rad[3]) }, { b_radius *
sin(angb_rad[3]) }, { 0 }, { 1 } };
mtx_type ba5[n][1] = { { b_radius * cos(angb_rad[4]) }, { b_radius *
sin(angb_rad[4]) }, { 0 }, { 1 } };
mtx_type ba6[n][1] = { { b_radius * cos(angb_rad[5]) }, { b_radius *
sin(angb_rad[5]) }, { 0 }, { 1 } };

// matrices to define platform joints
mtx_type pa1[n][1] = { { p_radius * cos(angp_rad[0]) }, { p_radius *
sin(angp_rad[0]) }, { 0 }, { 1 } };
mtx_type pa2[n][1] = { { p_radius * cos(angp_rad[1]) }, { p_radius *
sin(angp_rad[1]) }, { 0 }, { 1 } };
mtx_type pa3[n][1] = { { p_radius * cos(angp_rad[2]) }, { p_radius *
sin(angp_rad[2]) }, { 0 }, { 1 } };
mtx_type pa4[n][1] = { { p_radius * cos(angp_rad[3]) }, { p_radius *
sin(angp_rad[3]) }, { 0 }, { 1 } };
mtx_type pa5[n][1] = { { p_radius * cos(angp_rad[4]) }, { p_radius *
sin(angp_rad[4]) }, { 0 }, { 1 } };
mtx_type pa6[n][1] = { { p_radius * cos(angp_rad[5]) }, { p_radius *
sin(angp_rad[5]) }, { 0 }, { 1 } };

mtx_type ggc[n][1] = { { 0 }, { 0 }, { 0 }, { 1 } }; // moving grip center
wrt moving grip center

mtx_type hwgc[n][n] = {
 { 1, 0, 0, 0 },
 { 0, 1, 0, 0 },
 { 0, 0, 1, -gw },
 { 0, 0, 1, 1 }
};
mtx_type hgpc[n][n] = {
 { 1, 0, 0, 0 },
 { 0, 1, 0, 0 },
 { 0, 0, 1, gp },
 { 0, 0, 1, 1 }
};
mtx_type hwbc[n][n] = {
 { 1, 0, 0, 0 },
 { 0, 1, 0, 0 },
 { 0, 0, 1, wb },
 { 0, 0, 0, 1 }
};

// Function to check user input data
double check_InputData(double min_val, double max_val) {
 int ans = 0;
 String userAns;
 double input_data;
 while (ans == 0) {
 userAns = Serial.readStringUntil('\n');
 if (userAns != "") {
 if (userAns.toDouble() >= min_val && userAns.toDouble() <= max_val) {
 ans = 1;
 input_data = userAns.toFloat();
 Serial.print((String) "Value entered = " + input_data + "\n");
 } else {
 ans = 0;
 Serial.print(userAns + " --> is out of range, please enter again!\n");
 }
 }
 }
 return input_data;
}

// Function to read current actautor lengths
void read_curr_act_len_vol(double Length[], double Voltage[]) {
 // Reading present actuator lengths
 auto val1 = ads1.readADC_SingleEnded(0); // reading voltage from the pin
 Voltage[0] = val1 * 0.1875 / 1000.0; // converting voltage in mV units
 Length[0] = 70.81362 * Voltage[0] + 16.97272; // converting mV to lengths in
mm from the regression equation obtained in calibration of the LVDT1

 auto val2 = ads1.readADC_SingleEnded(1); // reading voltage from the pin
 Voltage[1] = val2 * 0.1875 / 1000.0; // converting voltage in mV units
 Length[1] = 73.82455 * Voltage[1] + 11.75907; // converting mV to lengths in
mm from the regression equation obtained in calibration of the LVDT2

 auto val3 = ads1.readADC_SingleEnded(2); // reading voltage from the pin
 Voltage[2] = val3 * 0.1875 / 1000.0; // converting voltage in mV units
 Length[2] = 66.70747 * Voltage[2] + 20.41616; // converting mV to lengths in
mm from the regression equation obtained in calibration of the LVDT3

 auto val4 = ads2.readADC_SingleEnded(0); // reading voltage from the pin
 Voltage[3] = val4 * 0.1875 / 1000.0; // converting voltage in mV units
 Length[3] = 68.76017 * Voltage[3] + 15.22161; // converting mV to lengths in
mm from the regression equation obtained in calibration of the LVDT4

 auto val5 = ads2.readADC_SingleEnded(1); // reading voltage from the pin
 Voltage[4] = val5 * 0.1875 / 1000.0; // converting voltage in mV units
 Length[4] = 69.68864 * Voltage[4] + 15.09707; // converting mV to lengths in
mm from the regression equation obtained in calibration of the LVDT5

 auto val6 = ads2.readADC_SingleEnded(2); // reading voltage from the pin
 Voltage[5] = val6 * 0.1875 / 1000.0; // converting voltage in mV units
 Length[5] = 71.50050 * Voltage[5] + 15.56798; // converting mV to lengths in
mm from the regression equation obtained in calibration of the LVDT6
}

// Function to control actuator speeds
int EHACalc(double differ, double speedFactor) {
 if ((String)differ == "-0.00" || (String)differ == "0.00") {
 differ = 0.0;
 }
 int EHA;

 if (differ >= -pos_tol && differ <= pos_tol) {
 EHA = 0;
 }
 if (differ > pos_tol && differ <= 1) {
 EHA = 105;
 } else if (differ > 1) {
 EHA = 105 + differ * speedFactor;
 }
 if (differ >= -1 && differ < -pos_tol) {
 EHA = -105;
 } else if (differ < -1) {
 EHA = -105 + differ * speedFactor;
 }
 return EHA;
}

// Function to create the platform motion
int travel_steps(int flag, int EHA1, int EHA2, int EHA3, int EHA4, int EHA5,
int EHA6, int count, double inputD[]) {
 while (EHA1 != 0 || EHA2 != 0 || EHA3 != 0 || EHA4 != 0 || EHA5 != 0 || EHA6
!= 0) {

 count = count + 1;

 read_curr_act_len_vol(Length, Voltage);
 // calculates the difference between target extension lengths (inputD) and
current extension lengths (Lengths)
 diff1 = inputD[0] - Length[0];
 diff2 = inputD[1] - Length[1];
 diff3 = inputD[2] - Length[2];
 diff4 = inputD[3] - Length[3];
 diff5 = inputD[4] - Length[4];
 diff6 = inputD[5] - Length[5];

 // considered max speed 300 and minimum speed 100 for difference 100mm to
0.5mm
 // Adjusted after multiple iterations due to unequal responses of the
actuators
 double diff_fact = 6;

 // Sets the actuator motions based on the differnce values
 EHA1 = EHACalc(diff1, diff_fact);
 EHA2 = EHACalc(diff2, diff_fact);
 EHA3 = EHACalc(diff3, diff_fact);
 EHA4 = EHACalc(diff4, diff_fact);
 EHA5 = EHACalc(diff5, diff_fact);
 EHA6 = EHACalc(diff6, diff_fact);

 // Sending signal to controller1 for actuator motions
 Serial1.println("!g 1 " + String(EHA1));
 Serial1.flush();
 Serial1.println("!g 2 " + String(EHA2));
 Serial1.flush();
 Serial1.println("!g 3 " + String(EHA3));
 Serial1.flush();

 // Checks controller availability in each cycle, it is only possible just
after the flushing
 if (Serial1.available() > 0) {
 Controller1_status = 1;
 } else {
 Serial.print("Controller 1 NOT FOUND.\n");
 Controller1_status = 0;
 }

 // Sending signal to controller2 for actuator motions
 Serial2.println("!g 1 " + String(EHA4));
 Serial2.flush();
 Serial2.println("!g 2 " + String(EHA5));
 Serial2.flush();
 Serial2.println("!g 3 " + String(EHA6));
 Serial2.flush();

 // Checks controller availability in each cycle, it is only possible just
after the flushing
 if (Serial2.available() > 0) {
 Controller2_status = 1;
 } else {
 Serial.print("Controller 2 NOT FOUND.\n");
 Controller2_status = 0;
 }
 // Displays reading in serial monitor for the current loop cycle for user
knowledge
 Serial.print((String) "\nReq Length1 = " + inputD[0] + "; Curr Length1 = " +
Length[0] + "; Curr Voltage1 = " + Voltage[0] + "; Length Diff1 = " + diff1 +
"; Speed1 = " + EHA1);
 Serial.print((String) "\nReq Length2 = " + inputD[1] + "; Curr Length2 = " +
Length[1] + "; Curr Voltage2 = " + Voltage[1] + "; Length Diff2 = " + diff2 +
"; Speed2 = " + EHA2);
 Serial.print((String) "\nReq Length3 = " + inputD[2] + "; Curr Length3 = " +
Length[2] + "; Curr Voltage3 = " + Voltage[2] + "; Length Diff3 = " + diff3 +
"; Speed3 = " + EHA3);
 Serial.print((String) "\nReq Length4 = " + inputD[3] + "; Curr Length4 = " +
Length[3] + "; Curr Voltage4 = " + Voltage[3] + "; Length Diff4 = " + diff4 +
"; Speed4 = " + EHA4);
 Serial.print((String) "\nReq Length5 = " + inputD[4] + "; Curr Length5 = " +
Length[4] + "; Curr Voltage5 = " + Voltage[4] + "; Length Diff5 = " + diff5 +
"; Speed5 = " + EHA5);
 Serial.print((String) "\nReq Length6 = " + inputD[5] + "; Curr Length6 = " +
Length[5] + "; Curr Voltage6 = " + Voltage[5] + "; Length Diff6 = " + diff6 +
"; Speed6 = " + EHA6);
 Serial.print("\n");

 Serial.print((String) "\nCycles count " + count + "\n");

 // Execution stops if any controller disconnected.
 if (Controller1_status == 0 || Controller2_status == 0) {
 Serial.print(F("Controller/s failed!\n"));
 break;
 }

 if (EHA1 == 0 && EHA2 == 0 && EHA3 == 0 && EHA4 == 0 && EHA5 == 0 && EHA6 ==
0) {
 if (flag == 0) {
 Serial.print(F("\nReached Home Pose!\n"));
 }
 if (flag == 1) {
 Serial.print(F("Lower Home position reached!\n"));
 }
 if (flag == 2) {
 Serial.print(F("Upper Home position reached!\n"));
 }
 if (flag == 3) {
 Serial.print(F("\nReached point on the Path!\n"));
 }
 break;
 }

 if (Serial.available() > 0) {
 char serialState = Serial.read();
 if (serialState == 's') {
 Serial.print((String) "\nCycles count " + count + "\n");
 Serial.print(F("Processing stopped by user!\n"));
 break;
 }
 }
 }
 return count;
```

```
}

// Calculate new actuator lengths
void calc_new_act_len(double Tx, double Ty, double Tz, double Rx, double Ry,
double Rz, double nLeg[]) {

 double Rx_rad = Rx * to_rad;
 double Ry_rad = Ry * to_rad;
 double Rz_rad = Rz * to_rad;

 mtx_type htm[n][n] = {
 { cos(Ry_rad) * cos(Rz_rad), sin(Rx_rad) * sin(Ry_rad) * cos(Rz_rad) -
cos(Rx_rad) * sin(Rz_rad), cos(Rx_rad) * sin(Ry_rad) * cos(Rz_rad) +
sin(Rx_rad) * sin(Rz_rad), Tx },
 { sin(Rz_rad) * cos(Ry_rad), sin(Rx_rad) * sin(Ry_rad) * sin(Rz_rad) +
cos(Rx_rad) * cos(Rz_rad), cos(Rx_rad) * sin(Ry_rad) * sin(Rz_rad) -
sin(Rx_rad) * cos(Rz_rad), Ty },
 { -sin(Ry_rad), cos(Ry_rad) * sin(Rx_rad), cos(Rx_rad) * cos(Ry_rad), Tz },
 { 0, 0, 0, 1 }
 };

 // Calcualate new Platform joints wrt to plaform center
 Matrixx.Multiply((mtx_type*)htm, (mtx_type*)gpa1, n, n, 1,
(mtx_type*)n_gpa1);
 Matrixx.Multiply((mtx_type*)htm, (mtx_type*)gpa2, n, n, 1,
(mtx_type*)n_gpa2);
 Matrixx.Multiply((mtx_type*)htm, (mtx_type*)gpa3, n, n, 1,
(mtx_type*)n_gpa3);
 Matrixx.Multiply((mtx_type*)htm, (mtx_type*)gpa4, n, n, 1,
(mtx_type*)n_gpa4);
 Matrixx.Multiply((mtx_type*)htm, (mtx_type*)gpa5, n, n, 1,
(mtx_type*)n_gpa5);
 Matrixx.Multiply((mtx_type*)htm, (mtx_type*)gpa6, n, n, 1,
(mtx_type*)n_gpa6);

 // Calcualate new Platform joints wrt to World origin
 Matrixx.Multiply((mtx_type*)hwgc, (mtx_type*)n_gpa1, n, n, 1,
(mtx_type*)n_wpa1);
 Matrixx.Multiply((mtx_type*)hwgc, (mtx_type*)n_gpa2, n, n, 1,
(mtx_type*)n_wpa2);
 Matrixx.Multiply((mtx_type*)hwgc, (mtx_type*)n_gpa3, n, n, 1,
(mtx_type*)n_wpa3);
 Matrixx.Multiply((mtx_type*)hwgc, (mtx_type*)n_gpa4, n, n, 1,
(mtx_type*)n_wpa4);
 Matrixx.Multiply((mtx_type*)hwgc, (mtx_type*)n_gpa5, n, n, 1,
(mtx_type*)n_wpa5);
 Matrixx.Multiply((mtx_type*)hwgc, (mtx_type*)n_gpa6, n, n, 1,
(mtx_type*)n_wpa6);

 // Final actuator lengths
 nLeg[0] = sqrt(sq(n_wpa1[0][0] - wba1[0][0]) + sq(n_wpa1[1][0] - wba1[1][0])
+ sq(n_wpa1[2][0] - wba1[2][0]));
 nLeg[1] = sqrt(sq(n_wpa2[0][0] - wba2[0][0]) + sq(n_wpa2[1][0] - wba2[1][0])
+ sq(n_wpa2[2][0] - wba2[2][0]));
 nLeg[2] = sqrt(sq(n_wpa3[0][0] - wba3[0][0]) + sq(n_wpa3[1][0] - wba3[1][0])
+ sq(n_wpa3[2][0] - wba3[2][0]));
 nLeg[3] = sqrt(sq(n_wpa4[0][0] - wba4[0][0]) + sq(n_wpa4[1][0] - wba4[1][0])
+ sq(n_wpa4[2][0] - wba4[2][0]));
 nLeg[4] = sqrt(sq(n_wpa5[0][0] - wba5[0][0]) + sq(n_wpa5[1][0] - wba5[1][0])
+ sq(n_wpa5[2][0] - wba5[2][0]));
 nLeg[5] = sqrt(sq(n_wpa6[0][0] - wba6[0][0]) + sq(n_wpa6[1][0] - wba6[1][0])
+ sq(n_wpa6[2][0] - wba6[2][0]));
}


void setup() {

 // Calcualate moving grip wrt to World center. THIS CANNOT BE DONE OUTSIDE
setup function
 Matrixx.Multiply((mtx_type*)hwgc, (mtx_type*)ggc, n, n, 1, (mtx_type*)wgc);

 // Calcualate base joints wrt to World center
 Matrixx.Multiply((mtx_type*)hwbc, (mtx_type*)ba1, n, n, 1, (mtx_type*)wba1);
 Matrixx.Multiply((mtx_type*)hwbc, (mtx_type*)ba2, n, n, 1, (mtx_type*)wba2);
 Matrixx.Multiply((mtx_type*)hwbc, (mtx_type*)ba3, n, n, 1, (mtx_type*)wba3);
 Matrixx.Multiply((mtx_type*)hwbc, (mtx_type*)ba4, n, n, 1, (mtx_type*)wba4);
 Matrixx.Multiply((mtx_type*)hwbc, (mtx_type*)ba5, n, n, 1, (mtx_type*)wba5);
 Matrixx.Multiply((mtx_type*)hwbc, (mtx_type*)ba6, n, n, 1, (mtx_type*)wba6);

 // Calcualate platform joints wrt to Moving grip center
 Matrixx.Multiply((mtx_type*)hgpc, (mtx_type*)pa1, n, n, 1, (mtx_type*)gpa1);
 Matrixx.Multiply((mtx_type*)hgpc, (mtx_type*)pa2, n, n, 1, (mtx_type*)gpa2);
 Matrixx.Multiply((mtx_type*)hgpc, (mtx_type*)pa3, n, n, 1, (mtx_type*)gpa3);
 Matrixx.Multiply((mtx_type*)hgpc, (mtx_type*)pa4, n, n, 1, (mtx_type*)gpa4);
 Matrixx.Multiply((mtx_type*)hgpc, (mtx_type*)pa5, n, n, 1, (mtx_type*)gpa5);
 Matrixx.Multiply((mtx_type*)hgpc, (mtx_type*)pa6, n, n, 1, (mtx_type*)gpa6);
 // ==============================================================================

 Serial.begin(115200); // Start serial monitor
 Serial.flush();
 Serial1.begin(115200); // starts controller 1 serial port
 Serial1.flush();
 Serial2.begin(115200); // starts controller 2 serial port
 Serial2.flush();

 // write null speed to 3 output pins to check availability of the controller
1
 Serial1.println("!g 1 0");
 Serial1.flush();
 Serial1.println("!g 2 0");
 Serial1.flush();
 Serial1.println("!g 3 0");
 Serial1.flush();

 Serial.print("\n");
 // if available or unavailable, informs the user
 if (Serial1.available() > 0) {
 while (Serial1.available() > 0) {
 char c_char = Serial1.read();
 inString1 += (char)c_char; // write read string in a single line instead of
writing each character in each line
 if (inString1.indexOf("FS=") < inString1.lastIndexOf('\r')) {
 stringComplete1 = true;
 }
 }
 if (stringComplete1) {
 Serial.print("\nController 1 Feedback: " + inString1);
 inString1 = "";
 }
 } else {
 Serial.print("\nController 1 NOT FOUND.");
 }

 // write null speed to 3 output pins to check availability of the controller
2
 Serial2.println("!g 1 0");
 Serial2.flush();
 Serial2.println("!g 2 0");
 Serial2.flush();
 Serial2.println("!g 3 0");
 Serial2.flush();

 // if available or unavailable, informs the user
 if (Serial2.available() > 0) {
 while (Serial2.available() > 0) {
 char c_char2 = Serial2.read();
 inString2 += (char)c_char2; // write read string in a single line instead of
writing each character in each line
 if (inString2.indexOf("FS=") < inString2.lastIndexOf('\r')) {
 stringComplete2 = true;
 }
 }
 if (stringComplete2) {
 Serial.print("\nController 2 Feedback: " + inString2);
 inString2 = "";
 }
 } else {
 Serial.print("\nController 2 NOT FOUND.");
 }

 // Starts the ADS1115 breakout boards
 ads1.begin();
 ads2.begin();
 ads1.setGain(GAIN_TWOTHIRDS);
 ads2.setGain(GAIN_TWOTHIRDS);
 ads1.setSPS(ADS1115_DR_64SPS);
 ads2.setSPS(ADS1115_DR_64SPS);
 delay(1000);

 read_curr_act_len_vol(Length, Voltage); // Read current actuator length from
potentiometers

 //Display present voltages in the serial monitor
 Serial.print("\nCurrent LVDT voltages (mV): ");
 Serial.print("Actuator 1: " + String(Voltage[0], 2) + "; ");
 Serial.print("Actuator 2: " + String(Voltage[1], 2) + "; ");
 Serial.print("Actuator 3: " + String(Voltage[2], 2) + "; ");
 Serial.print("Actuator 4: " + String(Voltage[3], 2) + "; ");
 Serial.print("Actuator 5: " + String(Voltage[4], 2) + "; ");
 Serial.print("Actuator 6: " + String(Voltage[5], 2) + ";\n");
 //Display present length in the serial monitor
 Serial.print("Current Actuator extensions (mm):");
 Serial.print("Actuator 1: " + String(Length[0], 2) + "; ");
 Serial.print("Actuator 2: " + String(Length[1], 2) + "; ");
 Serial.print("Actuator 3: " + String(Length[2], 2) + "; ");
 Serial.print("Actuator 4: " + String(Length[3], 2) + "; ");
 Serial.print("Actuator 5: " + String(Length[4], 2) + "; ");
 Serial.print("Actuator 6: " + String(Length[5], 2) + ";\n");

 Serial.print(F("\n[z --> upward is +ve]\n"));
 Serial.print(F("At Upper Hoome Position z = 0, Gripper faces are 50mm apart
& all extension lengths are = 96mm.\n"));
 Serial.print(F("At Lower Hoome Position z = -50, all actuators extension
lengths are = 52.5mm.\n"));

 // at z = -50mm actator lengths are 52.5mm
 Serial.print(F("Enter the values in the command line (ENTER 's' TO STOP THE
EXECUTION):\n"));
 Serial.print(F("Go to Lower Home Poisition (1), Upper Home Position (2), No
(0) [0/1/2]:\n"));
 while (ans == 0) {
 userAns = Serial.readStringUntil('\n');
 if (userAns != "") {
 ans = 1;
 usrAns = userAns.toInt();
 }
 }

 if (usrAns == 1) {
 Serial.print("\n");

 // Setting all actuator target length to 52.5mm to reach lower home position
 inputD[0] = (double)52.5;
 inputD[1] = (double)52.5;
 inputD[2] = (double)52.5;
 inputD[3] = (double)52.5;
 inputD[4] = (double)52.5;
 inputD[5] = (double)52.5;

 int flag = 1; // flag to identify the source of the movement
 // Moving the Platform
 travel_steps(flag, EHA1, EHA2, EHA3, EHA4, EHA5, EHA6, count, inputD);
 }

 if (usrAns == 2) {
 Serial.print("\n");

 // Setting all actuator target length to 96mm to reach upper home position
 inputD[0] = hm_act_len;
 inputD[1] = hm_act_len;
 inputD[2] = hm_act_len;
 inputD[3] = hm_act_len;
 inputD[4] = hm_act_len;
 inputD[5] = hm_act_len;

 int flag = 2; // flag to identify the source of the movement
 // Moving the Platform
 travel_steps(flag, EHA1, EHA2, EHA3, EHA4, EHA5, EHA6, count, inputD);
 }

 if (usrAns == 0) {
 // Take position input from the user
 // implement the conditions for Tx Ty Tz Rx Ry Rz
 int count1 = 0;

 Serial.print((String) "Enter x-coordinate from Home Pose (Between " + x_min
+ " to " + x_max + "): \n");
 Tx = check_InputData(x_min, x_max);
 Serial.print((String) "Enter y-coordinate from Home Pose (Between " + y_min
+ " to " + y_max + "): \n");
 Ty = check_InputData(y_min, y_max);
 Serial.print((String) "Enter z-coordinate from Home Pose (Between " + z_min
+ " to " + z_max + "): \n");
 Tz = check_InputData(z_min, z_max);
 Serial.print((String) "Enter rotation about x-axis (Between " + roll_min + "
to " + roll_max + "): \n");
 Rx = check_InputData(roll_min, roll_max);
 Serial.print((String) "Enter rotation about y-axis (Between " + pitch_min +
" to " + pitch_max + "): \n");
 Ry = check_InputData(pitch_min, pitch_max);
 Serial.print((String) "Enter rotation about z-axis (Between " + yaw_min + "
to " + yaw_max + "): \n");
 Rz = check_InputData(yaw_min, yaw_max);

 Serial.print(F("Input values are : Tx = "));
 Serial.print(Tx);
 Serial.print(F("; Ty = "));
 Serial.print(Ty);
 Serial.print(F("; Tz = "));
 Serial.print(Tz);
 Serial.print(F("; Rx = "));
 Serial.print(Rx);
 Serial.print(F("; Ry = "));
 Serial.print(Ry);
 Serial.print(F("; Rz = "));
 Serial.print(Rz);
 Serial.print("\n");

 // Calculating new leg lengths based on the input data
 calc_new_act_len(Tx, Ty, Tz, Rx, Ry, Rz, nLeg);

 Serial.print("New Target Actuator Lengths: Leg1 = ");
 Serial.print(nLeg[0]);
 Serial.print("; Leg2 = ");
 Serial.print(nLeg[1]);
 Serial.print("; Leg3 = ");
 Serial.print(nLeg[2]);
 Serial.print("; Leg4 = ");
 Serial.print(nLeg[3]);
 Serial.print("; Leg5 = ");
 Serial.print(nLeg[4]);
 Serial.print("; Leg6 = ");
 Serial.print(nLeg[5]);
 Serial.print("\n");
 // New extension lengths = total leg length - fixed length (465.02mm) =>
from the Matlab code
 inputD[0] = nLeg[0] - 465.02;
 inputD[1] = nLeg[1] - 465.02;
 inputD[2] = nLeg[2] - 465.02;
 inputD[3] = nLeg[3] - 465.02;
 inputD[4] = nLeg[4] - 465.02;
 inputD[5] = nLeg[5] - 465.02;

 // Sometimes the input data gets deleted from serial monitor due high amount
of text scrolling, so saving the data to display at the end again.
 double o_tx, o_ty, o_tz, o_rx, o_ry, o_rz; // original input data
 double o_leg1, o_leg2, o_leg3, o_leg4, o_leg5, o_leg6; // Final actuator
lengths to be obtained
 double o_ext1, o_ext2, o_ext3, o_ext4, o_ext5, o_ext6; // Final extension
lengths to be obtained

 o_tx = Tx;
 o_ty = Ty;
 o_tz = Tz;
 o_rx = Rx;
 o_ry = Ry;
 o_rz = Rz;

 o_leg1 = nLeg[0];
 o_leg2 = nLeg[1];
 o_leg3 = nLeg[2];
 o_leg4 = nLeg[3];
 o_leg5 = nLeg[4];
 o_leg6 = nLeg[5];

 o_ext1 = inputD[0];
 o_ext2 = inputD[1];
 o_ext3 = inputD[2];
 o_ext4 = inputD[3];
 o_ext5 = inputD[4];
 o_ext6 = inputD[5];

 // Check if the values are within limits
 if (nLeg[0] <= act_len_max && nLeg[1] <= act_len_max && nLeg[2] <=
act_len_max && nLeg[3] <= act_len_max && nLeg[4] <= act_len_max && nLeg[5] <=
act_len_max && nLeg[0] >= act_len_min && nLeg[1] >= act_len_min && nLeg[2] >=
act_len_min && nLeg[3] >= act_len_min && nLeg[4] >= act_len_min && nLeg[5] >=
act_len_min && n_wgc[3][0] <= (wgc[3][0] + 25) && n_wgc[3][0] >= (wgc[3][0] -
50)) {
 // Display New actuator lengths in the serial monitor
 Serial.print(F("New Actuator Extension Lengths (mm)\n"));
 Serial.print("Actuator 1: " + String(inputD[0], 2) + "; ");
 Serial.print("Actuator 2: " + String(inputD[1], 2) + "; ");
 Serial.print("Actuator 3: " + String(inputD[2], 2) + "; ");
 Serial.print("Actuator 4: " + String(inputD[3], 2) + "; ");
 Serial.print("Actuator 5: " + String(inputD[4], 2) + "; ");
 Serial.print("Actuator 6: " + String(inputD[5], 2) + ";");
 Serial.print(F("\n"));
 } else {
 Serial.print(F("\n"));
 Serial.print((String) "Cycles count " + count + "\n");
 Serial.print("Actuator Lengths / Platform center height exceed limits!.\n");
 }

 // if the platform center is not @(x, y, z) o equal to(0, 0, 0) and horz,
bring it to the home position
 if (Length[0] <= hm_pos_lower || Length[1] <= hm_pos_lower || Length[2] <=
hm_pos_lower || Length[3] <= hm_pos_lower || Length[4] <= hm_pos_lower ||
Length[5] <= hm_pos_lower || Length[0] >= hm_pos_upper || Length[1] >=
hm_pos_upper || Length[2] >= hm_pos_upper || Length[3] >= hm_pos_upper ||
Length[4] >= hm_pos_upper || Length[5] >= hm_pos_upper) {
 Serial.print("\n");

 // Setting all actuator target length to 96mm to reach upper home position
 inputD[0] = hm_act_len;
 inputD[1] = hm_act_len;
 inputD[2] = hm_act_len;
 inputD[3] = hm_act_len;
 inputD[4] = hm_act_len;
 inputD[5] = hm_act_len;

 // Going to home pose (0, 0, 0)
 Serial.print(F("\n\nMoving to Home Pose (0, 0, 0) first....\n"));

 int flag = 0; // flag to identify the source of the movement
 count1 = travel_steps(flag, EHA1, EHA2, EHA3, EHA4, EHA5, EHA6, count,
inputD);
 }

 // Path palnning for reaching the destination
 // Divide the destination movements into 10 steps
 // This display in the serial monitor shows more than 10 cycles, the display
shows all
 // moves in between each of the 10 steps
 // As per the current Matlab calculations, based on the intelligent
constructions, there is rarely a singularuty within the workspace
 double Tx10 = Tx / 10;
 double Ty10 = Ty / 10;
 double Tz10 = Tz / 10;
 double Rx10 = Rx / 10;
 double Ry10 = Ry / 10;
 double Rz10 = Rz / 10;

 Serial.print(F("Starting Motion to final point!\n"));

 int flag = 3; // flag to identify the source of the movement

 for (int i = 0; i < 10; i++) {

 if (Serial.available() > 0) {
 char serialState = Serial.read();
 if (serialState == 's') {
 flag = 5;
 Serial.print((String) "\nCycles count " + count + "\n");

 break;
 }
 }

 int j = i + 1;
 Serial.print((String) "\nGoing planned path point " + j + "\n");

 double Txn = Tx10 + i * Tx10;
 double Tyn = Ty10 + i * Ty10;
 double Tzn = Tz10 + i * Tz10;
 double Rxn = Rx10 + i * Rx10;
 double Ryn = Ry10 + i * Ry10;
 double Rzn = Rz10 + i * Rz10;

 calc_new_act_len(Txn, Tyn, Tzn, Rxn, Ryn, Rzn, nLeg);
 // New extension lengths = total leg length - fixed length (465.02mm) =>
from the Matlab code
 inputD[0] = nLeg[0] - 465.02;
 inputD[1] = nLeg[1] - 465.02;
 inputD[2] = nLeg[2] - 465.02;
 inputD[3] = nLeg[3] - 465.02;
 inputD[4] = nLeg[4] - 465.02;
 inputD[5] = nLeg[5] - 465.02;

 count = travel_steps(flag, EHA1, EHA2, EHA3, EHA4, EHA5, EHA6, count,
inputD);
 }
 if (flag == 5) {
 Serial.print(F("Processing stopped by user!\n"));
 } else {
 Serial.print(F("\nReached Target Pose!\n"));
 }
 int tcount = count1 + count;
 Serial.print((String) "\nTotal Cycles count " + tcount + "\n");

 // reprint input data at the end
 Serial.print(F("\nRedisplay critical data:\nInput values are : Tx = "));
 Serial.print(o_tx);
 Serial.print(F("; Ty = "));
 Serial.print(o_ty);
 Serial.print(F("; Tz = "));
 Serial.print(o_tz);
 Serial.print(F("; Rx = "));
 Serial.print(o_rx);
 Serial.print(F("; Ry = "));
 Serial.print(o_ry);
 Serial.print(F("; Rz = "));
 Serial.print(o_rz);
 Serial.print("\n");

 // reprint Final Target Actuator lengths at the end
 Serial.print("Final Target Actuator Lengths: Act1 = ");
 Serial.print(o_leg1);
 Serial.print("; Act2 = ");
 Serial.print(o_leg2);
 Serial.print("; Act3 = ");
 Serial.print(o_leg3);
 Serial.print("; Act4 = ");
 Serial.print(o_leg4);
 Serial.print("; Act5 = ");
 Serial.print(o_leg5);
 Serial.print("; Act6 = ");
 Serial.print(o_leg6);
 Serial.print("\n");

 // redisplay New actuator lengths in the serial monitor
 Serial.print(F("Final Target Actuator Extension Lengths (mm)\n"));
 Serial.print("Actuator 1: " + String(o_ext1, 2) + "; ");
 Serial.print("Actuator 2: " + String(o_ext2, 2) + "; ");
 Serial.print("Actuator 3: " + String(o_ext3, 2) + "; ");
 Serial.print("Actuator 4: " + String(o_ext4, 2) + "; ");
 Serial.print("Actuator 5: " + String(o_ext5, 2) + "; ");
 Serial.print("Actuator 6: " + String(o_ext6, 2) + ";");
 Serial.print(F("\n"));
 }

 Serial.print(F("\n\nExecution ended....\n")); // To show that the all
processing ended
}


// NOTHING IS HERE
void loop(void) {
}
```

7.3 MATLAB codes

7.3.1 tiger_dh_allpose_w_singularity_filter.m

```
% aa_tiger_dh_allpose_w_singularity_filter.m Sourabh Karmakar 2022-05-12
% revised on 2022-09-29 angle distribution matched with physical robot
% 2022-10-22 fully revised to take care of the difference in angles
% in plaform joints and base joints after facing z-rotation issue

% Calculates DH parameters for theoretical poses for Tiger 66.1 Hexapod by
Parallel Processing

% clc
clear all;
close all;
tic

% if the diary file exists delete it
if exist('aa_output_msrd_data.txt', 'file')
 diary off;
 delete('aa_output_msrd_data.txt');
end
diary aa_output_msrd_data.txt;

ncores = feature('numcores');
poolstat = gcp('nocreate'); % If no pool, do not create new one.
```

```
if isempty(poolstat)
 parpool ('local', ncores);
end
fprintf ("\nRunning Matlab on %d cores...\n", ncores);
disp(datetime("now"));
toc

tic
fprintf ("\nFresh time counting started for calculations....\n");

%% Loading all fixed dimensions in mm
az_all_fixed_variables_for_real_check; % Calling script to refresh all fixed
variables
load('fixed_variables_real_check.mat');

wplat_joints = Hwpc*plat_joints; % plat and base joints were wrt to their
centers
wbase_joints = Hwbc*base_joints;

wpa1 = Hwpc*pa1; wpa2 = Hwpc*pa2; wpa3 = Hwpc*pa3;
wpa4 = Hwpc*pa4; wpa5 = Hwpc*pa5; wpa6 = Hwpc*pa6;

wba1 = Hwbc*ba1; wba2 = Hwbc*ba2; wba3 = Hwbc*ba3;
wba4 = Hwbc*ba4; wba5 = Hwbc*ba5; wba6 = Hwbc*ba6;

% Actuator leg lengths in the current condition
leg1 = norm(wpa1(1:3,1) - wba1(1:3,1));
leg2 = norm(wpa2(1:3,1) - wba2(1:3,1));
leg3 = norm(wpa3(1:3,1) - wba3(1:3,1));
leg4 = norm(wpa4(1:3,1) - wba4(1:3,1));
leg5 = norm(wpa5(1:3,1) - wba5(1:3,1));
leg6 = norm(wpa6(1:3,1) - wba6(1:3,1));

exten_len1 = leg1 - dmin_org;
exten_len2 = leg2 - dmin_org;
exten_len3 = leg3 - dmin_org;
exten_len4 = leg4 - dmin_org;
exten_len5 = leg5 - dmin_org;
exten_len6 = leg6 - dmin_org;

fprintf( 'Actuator lengths in Home Position = %4.3f, %4.3f, %4.3f, %4.3f,
%4.3f & %4.3f (in mm).\n', leg1, leg2, leg3, leg4, leg5, leg6);
fprintf( 'Actuator extension lengths in Home Position = %4.3f, %4.3f, %4.3f,
%4.3f, %4.3f & %4.3f (in mm).\n', exten_len1, exten_len2, ...
 exten_len3, exten_len4, exten_len5, exten_len6);
fprintf( ['Tx, Ty, Tz limits are = %0.3f & %0.3f; %0.3f & %0.3f; %0.3f &
%0.3f (in mm);\nRx, Ry, Rz limits are = %0.3f & %0.3f; %0.3f & %0.3f;' ...
 ' %0.3f & %0.3f (in deg).\n'], x_min, x_max, y_min, y_max, z_min, z_max,
roll_min, roll_max, pitch_min, pitch_max, yaw_min, yaw_max);

% get inputs from the user
filename = 'input_theo_data.xlsx';
userdata = readmatrix(filename);
i = size(userdata,1);

% userdata = input('Enter the 6 values for Tx, Ty, Tz, Rx, Ry, Rz within a
[]:');
```

```
tot_calc = i;
fprintf( "\nNumber of checks to run %i.\n", tot_calc);
H_all = sparse(tot_calc, 16);
new_cood = sparse(tot_calc, 54);
jaco_all = [];
jaco_not_0 = [];

%% movement of the platform center --NEVER DELETE THE BELOW PART
% TXdist, TYdist, TZdist => translations along x, y, z axes respectively
% TXrad, TYrad, TZrad => rotations about x, y, z axes respectively

parfor idx = 1:i
 TXdist = userdata(idx,1);
 TYdist = userdata(idx,2);
 TZdist = userdata(idx, 3);
 TXrad = deg2rad(userdata(idx, 4));
 TYrad = deg2rad(userdata(idx, 5));
 TZrad = deg2rad(userdata(idx, 6));

 % from equation 2.63 & 2.64 of Crig's book, rotations are about fixed axes
 H =[cos(TYrad)*cos(TZrad), sin(TXrad)*sin(TYrad)*cos(TZrad)-
cos(TXrad)*sin(TZrad),...
 cos(TXrad)*sin(TYrad)*cos(TZrad)+sin(TXrad)*sin(TZrad), TXdist, ...
 sin(TZrad)*cos(TYrad),
sin(TXrad)*sin(TYrad)*sin(TZrad)+cos(TXrad)*cos(TZrad),...
 cos(TXrad)*sin(TYrad)*sin(TZrad)-sin(TXrad)*cos(TZrad), TYdist, ...
 -sin(TYrad), cos(TYrad)*sin(TXrad), cos(TXrad)*cos(TYrad), TZdist, ...
 0, 0, 0, 1];
 % New platform-end actuator coordinates wrt to base ctr
 h_mat = [H(1:4); H(5:8); H(9:12); H(13:16)];
 new_wpa1 = Hwgc*h_mat*Hgpc*pa1; new_wpa2 = Hwgc*h_mat*Hgpc*pa2; new_wpa3 =
Hwgc*h_mat*Hgpc*pa3;
 new_wpa4 = Hwgc*h_mat*Hgpc*pa4; new_wpa5 = Hwgc*h_mat*Hgpc*pa5; new_wpa6 =
Hwgc*h_mat*Hgpc*pa6;

 % Actuator leg lengths
 leg1 = norm(new_wpa1(1:3,1) - wba1(1:3,1));
 leg2 = norm(new_wpa2(1:3,1) - wba2(1:3,1));
 leg3 = norm(new_wpa3(1:3,1) - wba3(1:3,1));
 leg4 = norm(new_wpa4(1:3,1) - wba4(1:3,1));
 leg5 = norm(new_wpa5(1:3,1) - wba5(1:3,1));
 leg6 = norm(new_wpa6(1:3,1) - wba6(1:3,1));

 exten_len1 = leg1 - dmin_org;
 exten_len2 = leg2 - dmin_org;
 exten_len3 = leg3 - dmin_org;
 exten_len4 = leg4 - dmin_org;
 exten_len5 = leg5 - dmin_org;
 exten_len6 = leg6 - dmin_org;

 % Jacobian of the Stewart Gough Platform
 % Jacobian J^T is calculated as per article "Position-singularity analysis
 % of a special class of Stewart parallel mechanisms with two
 % dissimilar semi-symmetrical hexagons" by Li, Baokun and at el. eq (2b)

 % Unit vectors for each actuator
 s1 = (new_wpa1(1:3,1) - wba1(1:3,1))/leg1;
 s2 = (new_wpa2(1:3,1) - wba2(1:3,1))/leg2;
 s3 = (new_wpa3(1:3,1) - wba3(1:3,1))/leg3;
 s4 = (new_wpa4(1:3,1) - wba4(1:3,1))/leg4;
 s5 = (new_wpa5(1:3,1) - wba5(1:3,1))/leg5;
 s6 = (new_wpa6(1:3,1) - wba6(1:3,1))/leg6;

 s01 = cross(wba1(1:3,1),new_wpa1(1:3,1))/leg1;
 s02 = cross(wba2(1:3,1),new_wpa2(1:3,1))/leg2;
 s03 = cross(wba3(1:3,1),new_wpa3(1:3,1))/leg3;
 s04 = cross(wba4(1:3,1),new_wpa4(1:3,1))/leg4;
 s05 = cross(wba5(1:3,1),new_wpa5(1:3,1))/leg5;
 s06 = cross(wba6(1:3,1),new_wpa6(1:3,1))/leg6;

 jaco_cal = [s1 s2 s3 s4 s5 s6; s01 s02 s03 s04 s05 s06];

 % if det(jaco_cal) = 0, then that pose is singular and invalid
 new_wgc = Hwgc*h_mat*ggc; % new top center of lower (moving) grip

 % for new case the TXdist, TYdist, TZdist limits are valid for the moving
grip center
 lim_rad = sqrt(new_wgc(1)^2 + new_wgc(2)^2);
 if (new_wgc(1) >= x_min && new_wgc(2) >= y_min && new_wgc(3) >= (wmgc(3) -
60) && ...
 new_wgc(1) <= x_max && new_wgc(2) <= y_max && new_wgc(3) <= (wmgc(3) + 25)
...
 && lim_rad <= w_space_rad && leg1 <= dmax && leg2 <= dmax && leg3 <= dmax &&
leg4 <= dmax && ...
 leg5 <= dmax && leg6 <= dmax && leg1 >= dmin && leg2 >= dmin && leg3 >= dmin
&& ...
 leg4 >= dmin && leg5 >= dmin && leg6 >= dmin)
 if (det(jaco_cal) ~= 0)
 new_wpc = Hwgc*h_mat*gpc;
 new_twc = Htwc*new_wgc;
 new_cood(idx,:) = [new_wpa1', new_wpa2', new_wpa3', new_wpa4', ...
 new_wpa5', new_wpa6', new_wpc', new_wgc', leg1, leg2, leg3, leg4, leg5,
leg6, ...
 TXdist, TYdist, TZdist, rad2deg(TXrad), rad2deg(TYrad), rad2deg(TZrad),
exten_len1, exten_len2, ...
 exten_len3, exten_len4, exten_len5, exten_len6, new_twc'];
 H_all(idx,:) = [TXdist, TYdist, TZdist, rad2deg(TXrad), rad2deg(TYrad),
rad2deg(TZrad), exten_len1, exten_len2, ...
 exten_len3, exten_len4, exten_len5, exten_len6, new_wgc'];
 jaco_not_0(idx,:) = det(jaco_cal);
 else
 fprintf( '\nJacobian for data number %i is 0, i.e. Invalid.', idx);
 end
 jaco_all(idx,:) = [s1' s2' s3' s4' s5' s6' s01' s02' s03' s04' s05' s06'];
 else
 fprintf( '\nData number %i is invalid due to out-of-range actuator
lengths.\n', idx);
 end
end

% remove all rows with all zeros, i.e. invalid
h_params_inspace = full(H_all(any(H_all,2),:));
valid_coods_all = full(new_cood(any(new_cood,2),:));
jaco_not_0_valid = full(jaco_not_0(any(jaco_not_0,2),:));
```

```
jaco_all_valid = full(jaco_all(any(jaco_all,2),:));

% Save all data files
save('h_params_inspace.mat', 'h_params_inspace');
save('valid_coods_all.mat','valid_coods_all');
save('jaco_not_0_valid.mat', 'jaco_not_0_valid');
save('jac_all_valid.mat','jaco_all_valid');

%% NEVER DELETE THE ABOVE PART

%% Load exisitng data table

%% Process data
jaco_det_max = max(jaco_not_0_valid(:,1));
jaco_det_min = min(jaco_not_0_valid(:,1));

% count valid poses
v_pos_no = size(h_params_inspace);

fprintf( '\nNo. of valid positions without Singularity Check = %i.',
size(jaco_all_valid, 1));
fprintf( '\nNo. of valid positions with Singularity Check = %i.',
size(jaco_not_0_valid, 1));
fprintf( '\nNo. of Total positions evaluated = %i.\n', tot_calc);

pcx_max = max(valid_coods_all(1:end,25));
pcx_min = min(valid_coods_all(1:end,25));
pcy_max = max(valid_coods_all(1:end,26));
pcy_min = min(valid_coods_all(1:end,26));
pcz_max = max(valid_coods_all(1:end,27));
pcz_min = min(valid_coods_all(1:end,27));

gcx_max = max(valid_coods_all(1:end,29));
gcx_min = min(valid_coods_all(1:end,29));
gcy_max = max(valid_coods_all(1:end,30));
gcy_min = min(valid_coods_all(1:end,30));
gcz_max = max(valid_coods_all(1:end,31));
gcz_min = min(valid_coods_all(1:end,31));

minx = min(h_params_inspace(1:end,1));
maxx = max(h_params_inspace(1:end,1));
miny = min(h_params_inspace(1:end,2));
maxy = max(h_params_inspace(1:end,2));
minz = min(h_params_inspace(1:end,3));
maxz = max(h_params_inspace(1:end,3));

minroll = min(h_params_inspace(1:end,4));
maxroll = max(h_params_inspace(1:end,4));
minpitch = min(h_params_inspace(1:end,5));
maxpitch = max(h_params_inspace(1:end,5));
minyaw = min(h_params_inspace(1:end,6));
maxyaw = max(h_params_inspace(1:end,6));

Leg1_l_min = min(valid_coods_all(1:end,33));
Leg1_l_max = max(valid_coods_all(1:end,33));
Leg2_l_min = min(valid_coods_all(1:end,34));
Leg2_l_max = max(valid_coods_all(1:end,34));
```

```
Leg3_l_min = min(valid_coods_all(1:end,35));
Leg3_l_max = max(valid_coods_all(1:end,35));
Leg4_l_min = min(valid_coods_all(1:end,36));
Leg4_l_max = max(valid_coods_all(1:end,36));
Leg5_l_min = min(valid_coods_all(1:end,37));
Leg5_l_max = max(valid_coods_all(1:end,37));
Leg6_l_min = min(valid_coods_all(1:end,38));
Leg6_l_max = max(valid_coods_all(1:end,38));

minextn1 = min(h_params_inspace(1:end,7));
maxextn1 = max(h_params_inspace(1:end,7));
minextn2 = min(h_params_inspace(1:end,8));
maxextn2 = max(h_params_inspace(1:end,8));
minextn3 = min(h_params_inspace(1:end,9));
maxextn3 = max(h_params_inspace(1:end,9));
minextn4 = min(h_params_inspace(1:end,10));
maxextn4 = max(h_params_inspace(1:end,10));
minextn5 = min(h_params_inspace(1:end,11));
maxextn5 = max(h_params_inspace(1:end,11));
minextn6 = min(h_params_inspace(1:end,12));
maxextn6 = max(h_params_inspace(1:end,12));

fprintf( '\nCoordinate values in mm wrt WC:\n');
fprintf( 'Max of x, y & z-s of Grip center = %0.2f, %0.2f & %0.2f
respectively.\n', gcx_max, gcy_max, gcz_max);
fprintf( 'Min of x, y & z-s of Grip center = %0.2f, %0.2f & %0.2f
respectively.\n', gcx_min, gcy_min, gcz_min);
fprintf( 'Max of x, y & z-s of Platform center = %0.2f, %0.2f & %0.2f
respectively.\n', pcx_max, pcy_max, pcz_max);
fprintf( 'Min of x, y & z-s of Platform center = %0.2f, %0.2f & %0.2f
respectively.\n', pcx_min, pcy_min, pcz_min);

fprintf( '\nValues for the moving Grip Top Centers:\n');
fprintf( 'Maximum & Minimum values of x-translation = %0.2f & %0.2f
respectively.\n', maxx, minx);
fprintf( 'Maximum & Minimum values of y-translation = %0.2f & %0.2f
respectively.\n', maxy, miny);
fprintf( 'Maximum & Minimum values of z-translation = %0.2f & %0.2f
respectively.\n', maxz, minz);
fprintf( 'Maximum & Minimum values of Roll = %0.2f & %0.2f respectively.\n',
maxroll, minroll);
fprintf( 'Maximum & Minimum values of Pitch = %0.2f & %0.2f respectively.\n',
maxpitch, minpitch);
fprintf( 'Maximum & Minimum values of Yaw = %0.2f & %0.2f respectively.\n',
maxyaw, minyaw);

fprintf( ['Maximum & Minimum values of Actuators: A1 = %0.2f & %0.2f, A2 =
%0.2f & %0.2f, A3 = %0.2f & %0.2f,' ...
 ' A4 = %0.2f & %0.2f, A5 = %0.2f & %0.2f, A6 = %0.2f & %0.2f
respectively.\n'], Leg1_l_max, Leg1_l_min, Leg2_l_max, ...
 Leg2_l_min, Leg3_l_max, Leg3_l_min, Leg4_l_max, Leg4_l_min, Leg5_l_max,
Leg5_l_min, Leg6_l_max, Leg6_l_min);
fprintf( ['Maximum & Minimum values of Actuators extensions: xA1 = %0.2f &
%0.2f, xA2 = %0.2f & %0.2f, xA3 = %0.2f & %0.2f,' ...
 ' xA4 = %0.2f & %0.2f, xA5 = %0.2f & %0.2f, xA6 = %0.2f & %0.2f
respectively.\n'], maxextn1, minextn1, maxextn2, ...
 minextn2, maxextn3, minextn3, maxextn4, minextn4, maxextn5, minextn5,
maxextn6, minextn6);

fprintf( 'Maximum & Minimum values of Jacobian determinant = %0.2f & %0.2f
respectively.\n', jaco_det_max, jaco_det_min);

sel_pos_no = 100; % number of points selected for further calculation
if v_pos_no(1) > sel_pos_no
 sel_pos_no_rng = randi_org(v_pos_no(1),[1,sel_pos_no]); % randi_org(max(from
1 to),[rows, cols]) - external function
 valid_coods = valid_coods_all(sel_pos_no_rng, :);
 sele_100_details = h_params_inspace(sel_pos_no_rng, :);
 v_pos_no = sel_pos_no;
 save('valid_coods.mat','valid_coods');
else
 valid_coods = valid_coods_all;
 sele_100_details = h_params_inspace;
end
save('sele_100_details.mat', 'sele_100_details');
save('variables_saved', 'wwc', 'wmgc', 'wplat_joints', 'wbase_joints', 'wbc',
'v_pos_no', 'sel_pos_no')
%% Calculate angles till Platform joint for each set
fprintf('\nStarting angle checks till Platform joints.......');
angles_1 = sparse(v_pos_no(1),84);
parfor i = 1:v_pos_no(1)
 fprintf("\nFirst search no. %i of %i. ", i, v_pos_no(1));
 new_wpa1 = valid_coods(i,1:4)';
 new_wpa2 = valid_coods(i,5:8)';
 new_wpa3 = valid_coods(i,9:12)';
 new_wpa4 = valid_coods(i,13:16)';
 new_wpa5 = valid_coods(i,17:20)';
 new_wpa6 = valid_coods(i,21:24)';

 nleg1 = valid_coods(i,33);
 nleg2 = valid_coods(i,34);
 nleg3 = valid_coods(i,35);
 nleg4 = valid_coods(i,36);
 nleg5 = valid_coods(i,37);
 nleg6 = valid_coods(i,38);

 ang11 = angle_calculation_1(a, b, new_wpa1, nleg1, thetab_rng(1));
 ang12 = angle_calculation_1(a, b, new_wpa2, nleg2, thetab_rng(2));
 ang13 = angle_calculation_1(a, b, new_wpa3, nleg3, thetab_rng(3));
 ang14 = angle_calculation_1(a, b, new_wpa4, nleg4, thetab_rng(4));
 ang15 = angle_calculation_1(a, b, new_wpa5, nleg5, thetab_rng(5));
 ang16 = angle_calculation_1(a, b, new_wpa6, nleg6, thetab_rng(6));
 fprintf("\n");
 angles_1(i,:) = [ang11, ang12, ang13, ang14, ang15, ang16,
valid_coods(i,:)];
end

angles_1 = full(angles_1(any(angles_1,2),:));
ppose_devi = [angles_1(:, 1)' angles_1(:,6)' angles_1(:,11)' angles_1(:, 16)'
angles_1(:,21)' angles_1(:,26)'];
ppose_devi_max = max(ppose_devi);
ppose_devi_min = min(ppose_devi);
valid_angles_1 = size(angles_1);
fprintf('\nPlatform joint devation maximum = %3.4f mm & minimum = %3.4f mm
and found %i valid poses.\n', ...
 ppose_devi_max, ppose_devi_min, valid_angles_1(1));
toc

%% Calculate angles till Grip center for each set
fprintf('\nStarting angle checks till Grip Center .......');
angles_2 = sparse(valid_angles_1(1), 106);

parfor i = 1:valid_angles_1(1)
 fprintf("\nSecond search no. %i of %i.", i, valid_angles_1(1));
 new_wgc = valid_coods(i,29:32)';

 nleg1 = valid_coods(i,33);
 nleg2 = valid_coods(i,34);
 nleg3 = valid_coods(i,35);
 nleg4 = valid_coods(i,36);
 nleg5 = valid_coods(i,37);
 nleg6 = valid_coods(i,38);

 theta2_l1 = angles_1(i,3); theta3_l1 = angles_1(i,4);
 theta2_l2 = angles_1(i,8); theta3_l2 = angles_1(i,9);
 theta2_l3 = angles_1(i,13); theta3_l3 = angles_1(i,14);
 theta2_l4 = angles_1(i,18); theta3_l4 = angles_1(i,19);
 theta2_l5 = angles_1(i,23); theta3_l5 = angles_1(i,24);
 theta2_l6 = angles_1(i,28); theta3_l6 = angles_1(i,29);

 ang21 = angle_calculation_2(a, b, c, d, new_wgc, nleg1, thetab_rng(1),
theta2_l1, theta3_l1);
 ang22 = angle_calculation_2(a, b, c, d, new_wgc, nleg2, thetab_rng(2),
theta2_l2, theta3_l2);
 ang23 = angle_calculation_2(a, b, c, d, new_wgc, nleg3, thetab_rng(3),
theta2_l3, theta3_l3);
 ang24 = angle_calculation_2(a, b, c, d, new_wgc, nleg4, thetab_rng(4),
theta2_l4, theta3_l4);
 ang25 = angle_calculation_2(a, b, c, d, new_wgc, nleg5, thetab_rng(5),
theta2_l5, theta3_l5);
 ang26 = angle_calculation_2(a, b, c, d, new_wgc, nleg6, thetab_rng(6),
theta2_l6, theta3_l6);
 fprintf("\n");
 angles_2(i,:) = [new_wgc', ang21, ang22, ang23, ang24, ang25, ang26,
valid_coods(i,:)];
end
angles_2 = full(angles_2(any(angles_2,2),:));
wgc_pose_devi = [angles_2(:,5)' angles_2(:,13)' angles_2(:,21)'
angles_2(:,29)' angles_2(:,37)' angles_2(:,45)'];
wgc_pose_devi_max = max(wgc_pose_devi);
wgc_pose_devi_min = min(wgc_pose_devi);
valid_angles_2 = size(angles_2);
fprintf('\nGrip Center devation maximum = %3.4f mm & minimum = %3.4f mm and
found %i valid poses.\n', ...
 wgc_pose_devi_max, wgc_pose_devi_min, valid_angles_2(1));
save('angles_2_msrd.mat', "angles_2");

disp(datetime("now"));
toc
diary off;
```

7.3.2 *az_all_fixed_variables_for_real_check.m*

```
% Load all fixed variables.m Sourabh Karmakar 2022-06-05

% revised on 2022-08-29: All references are now calculated wrt moving grip
top center because
% the rotations and translations are specified wrt this point.
% revised on 2022-09-29: theta angles distributions match with physical model

% 2022-10-22 fully revised to take care of the difference in angles in
plaform joints
% and base joints after facing z-rotation issue.
% The angles were obtained from Sean's thesis and then measured in the
% actual model created by Sourabh.

% ** Consider Test starting position with 50mm gauge length as Home Pose.
% And moving platform center is lower Grip top surface center.

% The movement limit for the Tiger Platform, as per Sean's Thesis modified as
below:
% z is the vertical axis (upward +ve)
% z rotation = +/- 55 deg, z --> 50mm (the grippers touches each other),
% x & y rotation = +/- 60 deg, Translation x & y --> +/- 200 mm,
% From Sean's thesis page 76, 77.
% All these are limits are pure translation and pure rotation.
% Actuator close length = 305.4mm; stroke length = 203mm
% All measurements are from the home pose.
% At home position the Euler angles = 0
% Base joint circle radius = 477.39mm
% Sides of the base are = 570.4000mm & 377.8941mm
% Platform joint circle radius = 225.12mm
% Sides of the platform are = 268.6800mm & 178.8067mm

% Actuator length min 464.83mm
% Actuator length max 667.83mm

% WHEN BOTH GRIPPER ARE GAUGE LENGTH 50MM APART, THAT IS TOP HOME POSE.
% As per CAD model z motion possible = 50 (Grippers touches each other) to
% -104mm (all actuator extensions = 0.6mm)??????
% at z = 0, following are the dimensions:
% bc to pc 495.44mm
% wc to bc 71.55
% wc to pc 566.99 495.44 + 51.55
% pc to gc 318.23 gc --> top center of moving (lower) grip
% gc to pc -318.23
% wc to gc 885.22 = 566.99 + 318.23
% wc to tgc 935.22 = 885.22 + 50.00
% gc to bc -813.66
% pc to specimen center 343.23
% Grip face to grip face maximum 154.00????
% Gauge length 50.0 mm
% Grip Movement -104 mm & + 50mm

% New Information Added:
% ABS breaks with max 50% elongation in tensile test, so here z max is
% taken as -50 mm for 50 mm gauge length
% So x & y ranges are -158 mm to +158 mm
% z range changes from 0 mm to -50 mm

%% All fixed dimensions in mm
a = 71.55;
b = 477.39;
c = 225.12;
d = 318.23;

gp = -d; % moving grip center to platform center
gb = -813.67; % moving grip center to base center
gw = -885.22; % moving grip center to world center
bp = 495.44; % base to plaform center
tw = -935.21; % top grip ctr to world center
wb = a; % world center to base center
wp = wb + bp; % world center to platform center
brad = b;   % base joint radius
prad = c; % platform joint radius

dmin_org = 465.02;
% = 305.4 + 83.42 + 76.2; 83.42 mm is joint distance at lower end of actuator
dmax_org = 668.02;
% = 305.4 + 203 + 83.42 + 76.2; 76.2 mm is joint dist at upper end; 203 mm
stroke length

% modified actuator limits by clearing 25mm at both ends
dmin = dmin_org + 25;
dmax = dmax_org - 25;

% Maximum extension length working is <140mm 2023-03-25
% dmax = dmin + 115;

% translation and rotation limits
% Movement of the plaform ctr wrt to base ctr
% roll - x rotation angle in degrees
% pitch - y rotation angle in degrees
% yaw - z rotation angle in degrees

% Limits further changed due to operational issues on 2023-03-25
% Translations: x & y changed to +/-60, z to -60 to 0
x_min = -60; x_max = 60;
y_min = -60; y_max = 60;
z_min = -60; z_max = 0;
w_space_rad = 150;

% Rotations: x & y changed to +/-12, z to +/-30
roll_min = -12; roll_max = 12;
pitch_min = -12; pitch_max = 12;
yaw_min = -30; yaw_max = 30;

thetab_rng = [6.69, 53.31, 126.69, 173.31, 246.69, 293.31];
thetap_rng = [353.38, 66.53, 113.38, 186.53, 233.38, 306.53];

% Base joint coordinates in home position wrt base center frame
ba1 = [brad*cos(deg2rad(thetab_rng(1))); brad*sin(deg2rad(thetab_rng(1))); 0;
1];
ba2 = [brad*cos(deg2rad(thetab_rng(2))); brad*sin(deg2rad(thetab_rng(2))); 0;
1];
ba3 = [brad*cos(deg2rad(thetab_rng(3))); brad*sin(deg2rad(thetab_rng(3))); 0;
1];
ba4 = [brad*cos(deg2rad(thetab_rng(4))); brad*sin(deg2rad(thetab_rng(4))); 0;
1];
ba5 = [brad*cos(deg2rad(thetab_rng(5))); brad*sin(deg2rad(thetab_rng(5))); 0;
1];
ba6 = [brad*cos(deg2rad(thetab_rng(6))); brad*sin(deg2rad(thetab_rng(6))); 0;
1];
base_joints = [ba1 ba2 ba3 ba4 ba5 ba6 ba1];

% Platform joint coordinates in home position in platform center frame
pa1 = [prad*cos(deg2rad(thetap_rng(1))); prad*sin(deg2rad(thetap_rng(1))); 0;
1];
pa2 = [prad*cos(deg2rad(thetap_rng(2))); prad*sin(deg2rad(thetap_rng(2))); 0;
1];
pa3 = [prad*cos(deg2rad(thetap_rng(3))); prad*sin(deg2rad(thetap_rng(3))); 0;
1];
pa4 = [prad*cos(deg2rad(thetap_rng(4))); prad*sin(deg2rad(thetap_rng(4))); 0;
1];
pa5 = [prad*cos(deg2rad(thetap_rng(5))); prad*sin(deg2rad(thetap_rng(5))); 0;
1];
pa6 = [prad*cos(deg2rad(thetap_rng(6))); prad*sin(deg2rad(thetap_rng(6))); 0;
1];
plat_joints = [pa1 pa2 pa3 pa4 pa5 pa6 pa1];

% Vector from global frame to base ctr, platform ctr, grip ctr at test start
condition
% base ctr height 71.55 mm, bc to pc height 566.99-71.55 = 495.44 mm
ggc = [0; 0; 0; 1];
gpc = [0; 0; gp; 1];
gbc = [0; 0; gb; 1];
gwc = [0; 0; gw; 1];

wmgc = [0; 0; wb-gb; 1]; % = 566.99 + 318.22
wtgc = [0; 0; -tw; 1]; % = 885.22 + 50 (bottom center of top fixed grip)
twc = [ 0; 0 ; tw; 1]; % opposite to wtgc
wpc = [0; 0; wb+bp; 1];
wbc = [0; 0; wb; 1];

bpc = [0; 0; bp; 1]; % pc wrt bc
bmgc = [0; 0; -gb; 1]; % gc wrt bc
pgc = [0; 0; -gp; 1]; % gc wrt pc
ppc = [0; 0; 0; 1]; % pc wrt pc
wwc = [0; 0; 0; 1];
bbc = [0; 0; 0; 1];

Hwbc = [1 0 0 0; 0 1 0 0; 0 0 1 wb; 0 0 0 1]; % HTM vector from world ctr to
base ctr in current condition
Hwpc = [1 0 0 0; 0 1 0 0; 0 0 1 wb+bp; 0 0 0 1]; % HTM vector from world ctr
to base ctr in current condition
Hwgc = [1 0 0 0; 0 1 0 0; 0 0 1 -gw; 0 0 0 1]; % HTM vector from world ctr to
moving grip ctr in current condition

% HTM vector from base ctr to platform ctr in current condition
Hbpc = [1 0 0 0; 0 1 0 0; 0 0 1 bp; 0 0 0 1]; % 463.21-71.55
```

```
Hgbc = [1 0 0 0; 0 1 0 0; 0 0 1 gb; 0 0 0 1]; % HTM vector from moving grip
ctr to base ctr
Hgwc = [1 0 0 0; 0 1 0 0; 0 0 1 gw; 0 0 0 1]; % HTM vector from moving grip
ctr to world ctr
Hgpc = [1 0 0 0; 0 1 0 0; 0 0 1 gp; 0 0 0 1]; % HTM vector from moving grip
ctr to pf ctr
Htwc = [1 0 0 0; 0 1 0 0; 0 0 1 tw; 0 0 0 1]; % HTM vector from top grip ctr
to world ctr

save("fixed_variables_real_check.mat");
```

7.3.3 Function getTransformMatrix.m

```
%% Function1: returns the transformation matrix
function TransM = getTransformMatrix(a, alfa, d, theta)
theta = deg2rad(theta);

% from equation 3.6 of Crig's book
TransM = [cos(theta) -sin(theta) 0 a;
 sin(theta)*cos(alfa) cos(theta)*cos(alfa) -sin(alfa) -d*sin(alfa);
 sin(theta)*sin(alfa) cos(theta)*sin(alfa) cos(alfa) d*cos(alfa);
 0 0 0 1];
End
```

7.3.4 Function randi_org.m

```
function R = randi_org(imax, n)
%RANDI_ORG Random (www.random.org) integers from a uniform discrete
distribution.
%
% R = RANDI_ORG returns the quota of available random numbers.
% R = RANDI_ORG(IMAX,N) returns an N-by-N matrix containing random
% integer values drawn from the discrete uniform distribution on 1:IMAX.
% RANDI_ORG(IMAX,[M,N]) returns an M-by-N matrix.
% RANDI_ORG(IMAX,[M,N,P,...]) returns an
% M-by-N-by-P-by-... array.
% RANDI_ORG(IMAX,SIZE(A)) returns an array the same size as A.
%
% R = RANDI_ORG([IMIN,IMAX],...) returns an array containing integer
% values drawn from the discrete uniform distribution on IMIN:IMAX.
%
% Note: The size inputs M, N, P, ... should be positive integers.
%
% Note: the function connects via http to www.random.org. It depends on
% a working internet connection.
%
% Examples:
%
% Generate integer values from the uniform distribution on the set 1:10.
% r = randi_org(10,[100,1]);
%
% See also RAND, RANDN, RANDSTREAM, RANDSTREAM/RANDI,
RANDSTREAM.GETDEFAULTSTREAM.
```

```
% Copyright (c) 2010, Giampiero Salvi
% All rights reserved.
%
% Redistribution and use in source and binary forms, with or without
% modification, are permitted provided that the following conditions are
% met:
%
% * Redistributions of source code must retain the above copyright
% notice, this list of conditions and the following disclaimer.
% * Redistributions in binary form must reproduce the above copyright
% notice, this list of conditions and the following disclaimer in
% the documentation and/or other materials provided with the distribution
%
% THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
% AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
% IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
% ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
% LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
% CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
% SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
% INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
% CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
% ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
% POSSIBILITY OF SUCH DAMAGE.

if nargin == 0
 [result, status] = urlread('http://www.random.org/quota/?format=plain');
 R = str2num(result);
 return
end

if isscalar(imax)
 imin = 0;
else
 if length(imax(:))>2
 error('imax can at most hold two values');
 else
 imin = imax(1);
 imax = imax(2);
 end
end

if any(n<=0)
 error('randi_org:dimOutOfRange', 'n should contain positive integers');
end

if isscalar(n)
 sizes = [n n];
else
 sizes = n;
end

tot = prod(sizes(sizes>0));

% www.random.org restricts queries to 10000 numbers
maxlen = 10000;
```

```
nqueries = ceil(tot/maxlen);
R = zeros(tot,1);
for qu = 1:nqueries
 len = maxlen;
 if qu==nqueries
 len = rem(tot, maxlen);
 end
 places = (qu-1)*maxlen + (1:len);
 [result, status] = urlread(['http://www.random.org/integers/?num='
num2str(len) '&min=' num2str(imin) '&max=' num2str(imax)
'&col=1&base=10&format=plain&rnd=new']);
 if status~=1
 error('randi_org:queryFail', 'command failed with code %d', status);
 end
 R(places) = str2num(result);
end
R = reshape(R, sizes);
```

7.3.5 angle_calculation_1.m

```
%% Function 1: Checking the angles to reach from Base to platform joint
% Calculates the distance by norm function
function angles1_f1 = angle_calculation_1(a, b, nwpa, nleg, theta1)

diff_val_pa = 30;
incre_1 = 1;

theta2_rng = -75:incre_1:-48;
theta3_rng = -115:incre_1:-70;

% calling the inner function
angles1 = angle_calculation_1_inner(a, b, nwpa, nleg, theta1, diff_val_pa,
theta2_rng, theta3_rng);
if isempty(angles1)
 fprintf("\nPartial result!\n");
 angles1_f1 = [0 0 0 0 0];
 return
end

% selects all rows in angles1 where the value in the first column is greater
than or equal to zero
angles_a = angles1(angles1(:,1)>=0,:);
[~,b_idx1] = min(angles_a(:,1));
angles_f1 = angles_a(b_idx1,:);

%% Refining the calculation data
diff_val_pa = 1;
incre_2 = 0.01;

theta2 = angles_f1(1,3);
theta3 = angles_f1(1,4);

theta2_rng = (theta2 - 3 + incre_2):incre_2:(theta2 + 3 - incre_2);
theta3_rng = (theta3 - 3 + incre_2):incre_2:(theta3 + 3 - incre_2);
```

```
angles1_2 = angle_calculation_1_inner(a, b, nwpa, nleg, theta1, diff_val_pa,
theta2_rng, theta3_rng);
if isempty(angles1_2)
 fprintf("\nPartial result!\n");
 angles1_f1 = [0 0 0 0 0];
 return
end

angles_a = angles1_2(angles1_2(:,1)>=0,:);
[~,b_idx2] = min(angles_a(:,1));
angles1_f1 = angles_a(b_idx2,:);
end
```

7.3.6 angle_calculation_1_inner.

```
%% Function 2: repeating loop of angle_calcualtion_1
function angles = angle_calculation_1_inner(a, b, nwpa, nleg, theta1,
diff_val_pa, theta2_rng, theta3_rng)

theta2_rng_len = length(theta2_rng);
theta3_rng_len = length(theta3_rng);

calc_space = [theta2_rng_len, theta3_rng_len];
tot_calc = prod(calc_space);
angles = sparse(tot_calc, 5);

parfor idx = 1:tot_calc
 [i, j] = ind2sub(calc_space, idx);

 theta2 = theta2_rng(i);
 theta3 = theta3_rng(j);

 %dh = a alfa(in radians) d theta(in degrees)
 dh = [0 0 a theta1;
 b pi/2 0 theta2;
 0 pi/2 0 theta3;
 0 pi/2 0 -90;
 0 0 nleg 0];

 % HTM between 2 consecutive frames for F0 to F5 using DH paramaters
 A01=getTransformMatrix(dh(1,1), dh(1,2), dh(1,3), dh(1,4));
 A12=getTransformMatrix(dh(2,1), dh(2,2), dh(2,3), dh(2,4));
 A23=getTransformMatrix(dh(3,1), dh(3,2), dh(3,3), dh(3,4));
 A34=getTransformMatrix(dh(4,1), dh(4,2), dh(4,3), dh(4,4));
 A45=getTransformMatrix(dh(5,1), dh(5,2), dh(5,3), dh(5,4));

 T05 = A01*A12*A23*A34*A45;
 calc_nwpa = T05(1:4,4);
 nwpa_diff = norm(nwpa - calc_nwpa);

 if nwpa_diff < diff_val_pa
 angles(idx,:) = [nwpa_diff, theta1, theta2, theta3, nleg];
 end
end
```

```
% removes any rows in the matrix angles that contain all zeros, and then
assigns the remaining rows to the variable angles.
angles = full(angles(any(angles,2),:));
end
```

7.3.7 *angle_calculation_2.m*

```
%% Function 3: Checking the angles to reach from platform joint to grip
center
function angles3_f3 = angle_calculation_2(a, b, c, d, nwgc, nleg, theta1,
theta2, theta3)

diff_val_pa = 15;
incre_1 = 3;
theta5_rng = 57:incre_1:125;
theta6_rng = 95:incre_1:210;
theta7_rng = -60:incre_1:60;

angles1 = angle_calculation_2_inner(a, b, c, d, nwgc, nleg, theta1, theta2,
theta3, diff_val_pa, theta5_rng, theta6_rng, theta7_rng);
if isempty(angles1)
 fprintf("\nPartial result!\n");
 angles3_f3 = [0 0 0 0 0 0 0 0];
 return
end
angles1_a = angles1(angles1(:,1)>=0,:);
[~,b_idx1] = min(angles1_a(:,1));
angles1_f2 = angles1_a(b_idx1,:);

%% Refining the calculation data
diff_val_pa = 2;
incre_2 = 0.1;

theta5 = angles1_f2(1,6);
theta6 = angles1_f2(1,7);
theta7 = angles1_f2(1,8);

theta5_rng = (theta5 - 3 + incre_2):incre_2:(theta5 + 3 - incre_2);
theta6_rng = (theta6 - 3 + incre_2):incre_2:(theta6 + 3 - incre_2);
theta7_rng = (theta7 - 3 + incre_2):incre_2:(theta7 + 3 - incre_2);

angles2 = angle_calculation_2_inner(a, b, c, d, nwgc, nleg, theta1, theta2,
theta3, diff_val_pa, theta5_rng, theta6_rng, theta7_rng);
if isempty(angles2)
 fprintf("\nPartial result!\n");
 angles3_f3 = [0 0 0 0 0 0 0 0];
 return
end
angles2_a = angles2(angles2(:,1)>=0,:);
[~,b_idx2] = min(angles2_a(:,1));
angles2_f2 = angles2_a(b_idx2,:);

%% Refining the calculation data 2nd stage
diff_val_pa = 1;
```

```
incre_3 = 0.05;

theta5 = angles2_f2(1,6);
theta6 = angles2_f2(1,7);
theta7 = angles2_f2(1,8);

theta5_rng = (theta5 - 0.1 + incre_3):incre_3:(theta5 + 0.1 - incre_3);
theta6_rng = (theta6 - 0.1 + incre_3):incre_3:(theta6 + 0.1 - incre_3);
theta7_rng = (theta7 - 0.1 + incre_3):incre_3:(theta7 + 0.1 - incre_3);

angles3 = angle_calculation_2_inner(a, b, c, d, nwgc, nleg, theta1, theta2,
theta3, diff_val_pa, theta5_rng, theta6_rng, theta7_rng);
if isempty(angles3)
 fprintf("\nPartial result!\n");
 angles3_f3 = [0 0 0 0 0 0 0 0];
 return
end
angles3_a = angles3(angles3(:,1)>=0,:);
[~,b_idx3] = min(angles3_a(:,1));
angles3_f3 = angles3_a(b_idx3,:);
end
```

7.3.8 angle_calculation_2_inner.m

```
%% Function 4: repeating loop of angle_calcualtion_2
function angles = angle_calculation_2_inner(a, b, c, d, nwgc, nleg, theta1,
theta2, theta3, diff_val_pa, theta5_rng, theta6_rng, theta7_rng)

%dh = a alfa(in radians) d theta(in degrees)
dh1 = [0 0 a theta1;
 b pi/2 0 theta2;
 0 pi/2 0 theta3;
 0 pi/2 0 -90;
 0 0 nleg 0];

A01=getTransformMatrix(dh1(1,1), dh1(1,2), dh1(1,3), dh1(1,4));
A12=getTransformMatrix(dh1(2,1), dh1(2,2), dh1(2,3), dh1(2,4));
A23=getTransformMatrix(dh1(3,1), dh1(3,2), dh1(3,3), dh1(3,4));
A34=getTransformMatrix(dh1(4,1), dh1(4,2), dh1(4,3), dh1(4,4));
A45=getTransformMatrix(dh1(5,1), dh1(5,2), dh1(5,3), dh1(5,4));

T05 = A01*A12*A23*A34*A45;

theta5_rng_len = length(theta5_rng);
theta6_rng_len = length(theta6_rng);
theta7_rng_len = length(theta7_rng);

calc_space = [theta5_rng_len, theta6_rng_len, theta7_rng_len];
tot_calc = prod(calc_space);
angles = sparse(tot_calc, 8);

parfor idx = 1:tot_calc
 [i, j, k] = ind2sub(calc_space, idx);
 theta5 = theta5_rng(i);
 theta6 = theta6_rng(j);
 theta7 = theta7_rng(k);

 %dh = a alfa(in radians) d theta(in degrees)
 dh2 = [0 0 0 theta5;
 0 pi/2 0 theta6;
 0 pi/2 0 theta7;
 c 0 d 0];

 % HTM between 2 consecutive frames for F0 to F5 using DH paramaters
 A56=getTransformMatrix(dh2(1,1), dh2(1,2), dh2(1,3), dh2(1,4));
 A67=getTransformMatrix(dh2(2,1), dh2(2,2), dh2(2,3), dh2(2,4));
 A78=getTransformMatrix(dh2(3,1), dh2(3,2), dh2(3,3), dh2(3,4));
 A89=getTransformMatrix(dh2(4,1), dh2(4,2), dh2(4,3), dh2(4,4));

 T59 = A56*A67*A78*A89;
 T09 = T05*T59;
 calc_nwgc = T09(1:4,4);
 nwgc_diff = norm(nwgc - calc_nwgc);

 if nwgc_diff < diff_val_pa
 angles(idx,:) = [nwgc_diff, theta1, theta2, theta3, nleg, theta5, theta6,
theta7];
 end
end
angles = full(angles(any(angles,2),:));
end
```