December 2015

Adaptation of Moiré Phase Tracking to a Mobile Device for Field 3D Data Collections

Amar Nikhanj
University of Wisconsin-Milwaukee

ADAPTATION OF MOIRÉ PHASE TRACKING TO A MOBILE DEVICE FOR FIELD 3D DATA COLLECTIONS

by

Amar Nikhanj

A Thesis Submitted in

Partial Fulfillment of the

Requirements for the Degree of

Master of Science

in Engineering

at

The University of Wisconsin-Milwaukee

December 2015

ABSTRACT

ADAPTATION OF MOIRÉ PHASE TRACKING TO A MOBILE DEVICE FOR FIELD 3D DATA COLLECTIONS

by

Amar Nikhanj

The University of Wisconsin-Milwaukee, 2015
Under the Supervision of Professor Brian Armstrong

The accelerating technologies of mobiles devices such as tablets and phones provide an ability to perform high intensity calculations all while obtaining precise data with increasingly more accurate sensors. Image metrology and 3D motion tracking can take advantage of these improvements as they require both significant processing power and camera controls not seen in mobile devices until very recently. This thesis discusses the development of 3D motion tracking using Google Nexus tablets and applying the technology to the Rapid Upper Limb Assessment (RULA) to demonstrate the technology's application to field measurements. RULA analysis determines the stress placed on a human body by lifting, turning, and twisting, among other factors and is normally estimated by trained personnel. However, this low-cost and mobile image metrology system inherently calculates angles and motion, allowing for a quicker and more precise instrumented RULA versus simply an observation analysis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Brian Armstrong, who has taught me vast amounts of knowledge during this adventure and from all of the classes he's taught that I was fortunate enough to be enrolled in. Thanks to Professor Naira Campbell and Madiha Ahmed for the assistance with the RULA study and the three UWM students who volunteered to be our subjects. I would also like to thank my family in supporting me in all of my endeavors. And to my smart, sassy, and beautiful wife Kristen: thank you for all the support you give me, especially when I need you the most.

1 Introduction

There are a number of 3D motion tracking technologies that exist in the world today. A wide variety of industries, such as biomechanics, automation, and entertainment currently benefit from this technology and many others are looking to integrate 3D motion tracking into their products and processes. These systems are getting smaller, cheaper, and faster just like all other technologies. Most would agree that one industry in particular excels at making devices that are smaller, cheaper, and faster: the mobile device industry. Combining 3D motion tracking with a mobile device would therefore be a paradigm shift in 3D motion tracking technology. A method to do just that is described in this thesis.

1.1 Moiré Phase Tracking (MPT)

Many 3D motion tracking systems that exist require multiple cameras to perform the motion tracking, such as in Figure 1.

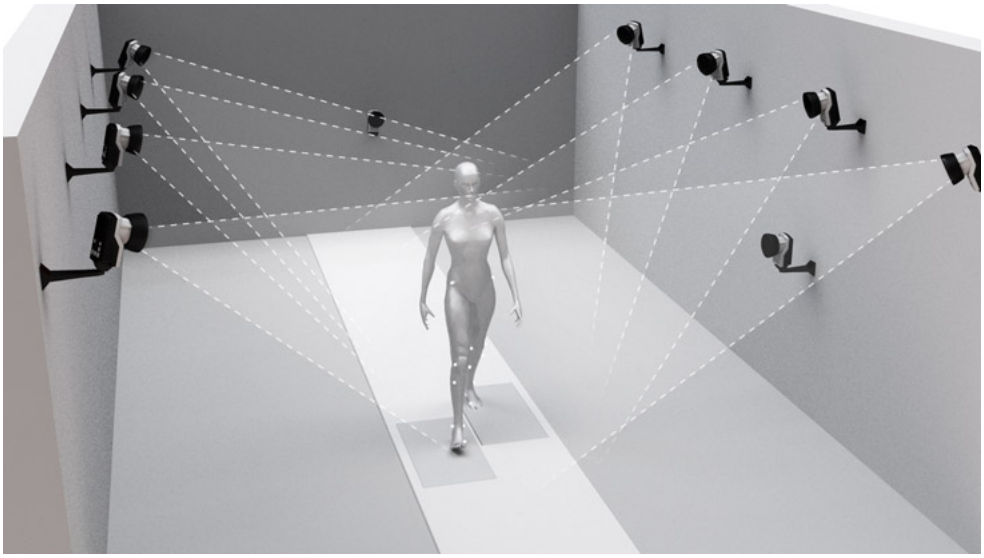

Figure 1: Motion tracking of gait analysis (courtesy of Qualysis, Inc.). Notice the multiple cameras and reflective balls - making it a poor choice for 3D measurements in the field.

If this system were to be adopted on a mobile device, certainly more than one device would be required due to the multiple cameras which immediately increases the cost and complexity. Some motion tracking systems, similar to the one seen in Figure 2, are sensitive to other objects in the field of view that may emit or reflect light, such as jewelry, and confuse it with the reflective balls.

The edges of the reflective balls must also be detected and, as such, contrast must exist between the balls and anything behind the balls from the cameras' perspective. In short, many of these technologies require *control of their optical environment*. If 3D motion tracking is desired in the field, there is a high likelihood

Figure 2: Physical therapy lab (courtesy of University of California, Merced). Again, multiple cameras and the system must be calibrated each time it is set up.

it will be in an environment that is not easily controlled.

Moiré Phase Tracking (MPT) is a 3D motion tracking technology developed in the Armstrong laboratory that mitigates the issues with other 3D tracking technology [1]. MPT is immune to reflections and other sources of light and requires merely one camera to track markers, shown in Figure 3, in 3D space. The markers are engineered such that their six degrees of freedom (6-DOF) pose can be calculated [2].

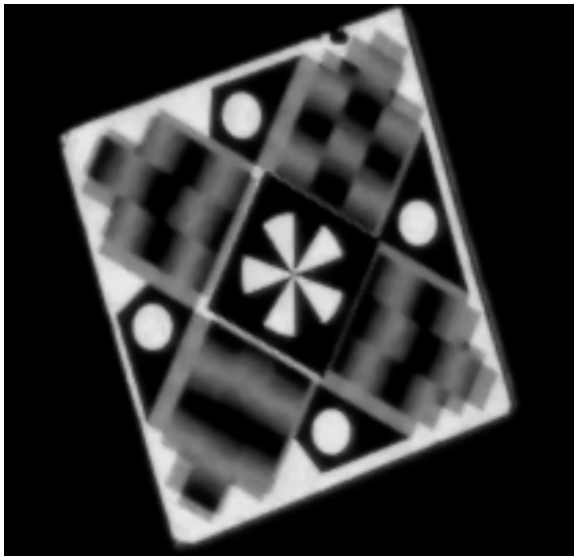

Figure 3: An example of an MPT marker. The design allows for all 6-DOF pose data to be calculated.

The object in the center of the motion tracking marker is called a *starburst landmark*. This landmark, each of the four circles along the perimeter, and the periodic Moiré patterns are all used in the processing algorithms to calculate the X, Y, Z of the marker (the location of the center of the marker with respect to

the camera) in addition to the pitch, roll, and yaw *rotation* angles of the marker. The X, Y, Z, and pitch, roll, and yaw are the 6-DOF pose. The relative simplicity of MPT makes it ideal for porting to a mobile device due to requiring only one camera and one camera calibration.

MPT has already been utilized and studied. For example, in a Magnetic Resonance Imager (MRI), motion of the subject can be render the results unusable. MPT has been used within MRI scans to detect the motion and subsequently correct the blur it produces [3]. MPT has also been used to assess susceptibility to injury [4].

The successful history of MPT in studying biomechanics merged with the mobility provided by a mobile device provides a powerful combination of possibilities. One possibility is the Rapid Upper Limb Assessment.

1.2 Rapid Upper Limb Assessment (RULA)

The human body is an extraordinary machine. It has evolved for millennia, adapting to the environmental changes throughout history that created a versatile and complex system. This complexity exists both internally and externally. Part of the external complexity is the range of motion of the body's joints, where limbs can rotate via multiple joints that couple to give complex rotations. Measuring these simultaneous rotations, decomposing them, and then applying them to risk assessment can be a daunting task. But risk analyses do exist, such as the Rapid Upper Limb Assessment (RULA). The RULA is a survey method developed to perform risk analysis of upper limb injury in the workplace [5]. The standard RULA is to analyze the worst-case posture, or in some cases multiple postures, in a motion and estimate the joint angles. The final score determines one of four results:

- The motion is acceptable
- The motion requires further investigation and changes may be necessary
- An investigation and changes to the motion are necessary soon
- An investigation and changes to the motion are required immediately

RULA has been used to measure musculoskeletal risk, improve workstation designs, increase productivity, and educate workers on improving their motion to reduce their risk of injury [6]. RULA has been applied to many industries including rubber sheet production, knitting postures, truck driving, and weaving machine operators [7]. The bio-mechanical aspect along with the field difficulties make RULA a strong candidate for validation of a mobile 3D motion tracking system.

1.2.1 Standard RULA

RULA in the most basic form can be completed visually and was developed to do so. The observer performing RULA uses his or her best judgment in determining the worker's pose during the motion that is analyzed. This pose could either be the position that is held the longest or is the highest risk. A worksheet developed by Dr. Alan Hedge allows for a relatively simple flow down of joint rotations versus RULA scores (Figure 4). The angles of concern for RULA are upper arm flexion or extension, upper arm abduction, elbow flexion or extension, wrist flexion or extension, wrist radial or ulnar deviation, neck flexion or extension, and trunk flexion or extension.

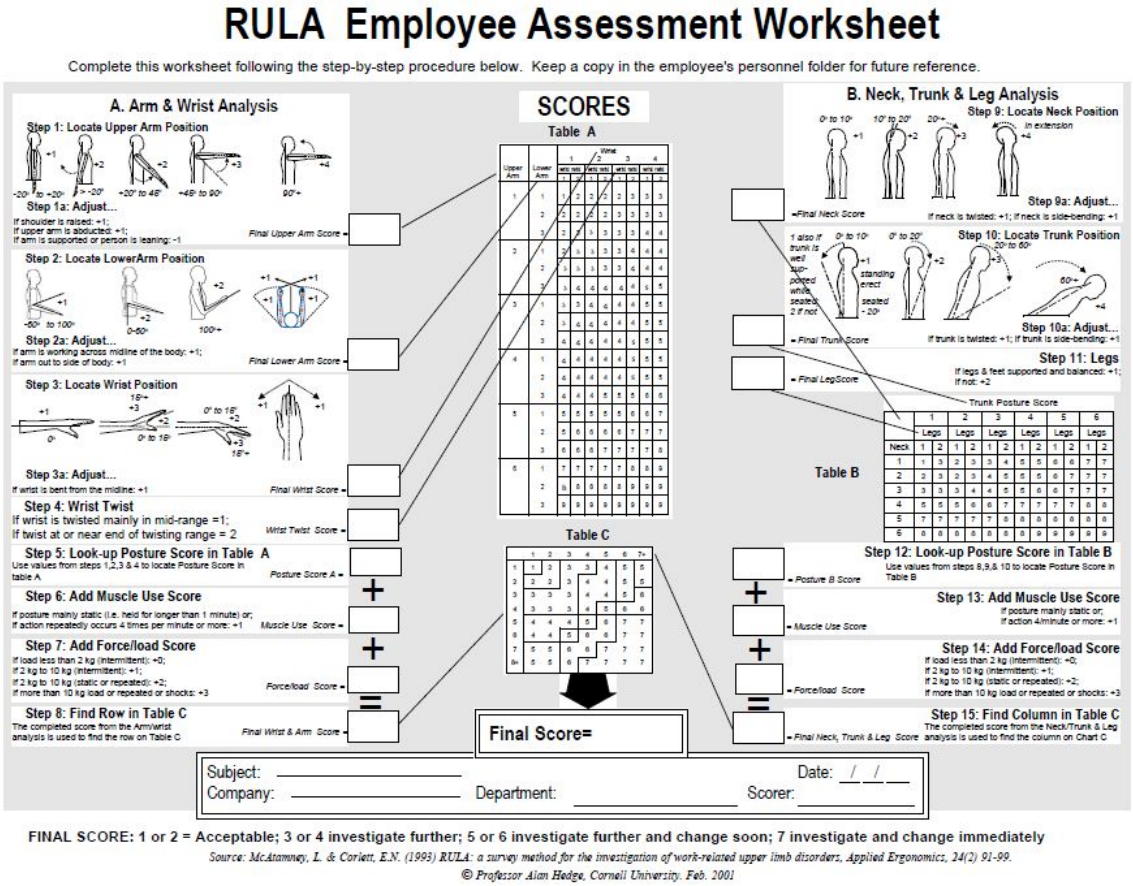

Figure 4: Example of an Rapid Upper Limb Assessment (RULA) worksheet. Trained personnel can use this to determine musculoskeletal risk of a worker's motion.

Important to this thesis are the coordinates frames that each limb's rotation is measured with respect to:

- The trunk, or torso, is measured with respect to the room coordinates (Y-Axis aligned with gravity)
- The neck rotation is with respect to the torso
- The upper arm rotation is with respect to the torso

- The lower arm rotation is with respect to the upper arm
- The wrist rotations, including wrist twist, are with respect to the lower arm

Non-angular variables are load and leg balancing which are included in the RULA calculation.

1.2.2 Alternative Methods to Conduct RULA

The standard RULA measurement has several downfalls:

- The observer performing RULA must use his or her judgment on which posture to use
- The observer must *estimate* the angles through which each limb is rotating

Requiring the observer to use discretion can certainly increase error in the analysis. However, the rapid increase in technology has given rise to more autonomous and precise methods to perform RULA. A study to perform RULA on palm oil harvesters juxtaposed an image of the worker with a digital representation of the worker's posture through Digital Human Modeling Software (DHMS) [8]. The angles generated with the DHMS are used with CATIA software to generate the RULA score. The DHMS software presumably returns accurate angles, however, the study still had to determine the posture to analyze, and each posture was remodeled in the DHMS software. This certainly increases error in the angles as human discretion is required.

Using Siemens Jack, a DHMS package, RULA is performed on a simulation video of the installation of fog lamps in an automotive assembly line [9], thus allowing all postures throughout the movement to be analyzed. This eliminates the discretion required for the worst-case posture used in the analysis. However, models of all the tools used, the human subject, and the environment has to be created - adding substantial time to the analyses. Also, care has to be taken to ensure the models in the simulation are accurate.

A novel solution is to use a Microsoft Kinect that is already engineered to detect body movements [10]. Using Microsoft's Kinect Software Development Kit (SDK), this eliminates the need to determine posture as RULA scores occur in real-time and uses voxels to estimate angles. However, certain limb angles cause the angle estimation algorithm to become unstable, including crossing arms and various trunk rotations that are essential to the RULA calculation. The Microsoft Kinect also currently does not track the hand and therefore cannot detect wrist rotations which are also necessary for RULA.

1.3 A Portable Optical 3D Motion Capture System

RULA is a good candidate for 3D motion tracking: it is information in 3D space that must be decomposed into angles. Unfortunately, many 3D motion tracking technologies require more than one camera and have

a complicated calibration process [11, 12, 13]. RULA is also typically performed in the field in a potentially messy, enclosed, or dangerous environment - where having a complicated multi-camera system may not be realistic.

1.3.1 MPT Mobility

The current use of MPT requires a calibrated camera and comes with a computer containing the algorithms to calculate the 6-DOF pose data (Figure 5). In terms of RULA, this is still disadvantageous as the computer and camera must be brought into a typically hostile environment.

Figure 5: Figure of the current MPT system. Although MPT fundamentally is a good choice for field measurements, the physical set up still has drawbacks due to the multiple components required.

Thus, MPT ported to a mobile device then applied to RULA immediately validates the benefit of mobile 3D motion tracking technology. The work of developing photogrammetry on mobile devices is already being implemented [14, 15, 16] so there is reason to believe MPT can also be ported to a mobile device. The

development requires three major components: processing capability (Section 2), camera control (Section 3), and camera calibration (Section 4). Section 5 describes how the data returned by MPT is analyzed to calculate the RULA angles, and Section 6 summarizes a RULA case study.

2 Developing MPT Processing on the Android Platform

The Nexus device must have the resources to process an image otherwise the mobility aspect is lost. The validation that processing is working correctly is done processing a known good image and comparing the results to the validated MPT system.

2.1 MPT Processing Development Introduction

MPT can theoretically be ported to the Apple (iOS) or Google (Android) platform, although it benefits greatly from intimate knowledge of the camera and its internal workings. As such, Android, as the open-source platform, was decided to be the superior choice.

The majority of processing development occurred on the Nexus 10 tablet running Android KitKat with Eclipse Integrated Development Environment (IDE). The image processing algorithms were originally written in MATLAB and was subsequently translated into C++ using the MATLAB code generation feature targeting the ARM processor. This posed the problem of adding complexity considering the default language to program an Android device is Java, but the MATLAB coder does not translate to Java.

The Android platform contains a Standard Development Kit (SDK) accessible in Java, but the developers are aware that many people have an interest in programming with native languages, such as C or C++. To use these native languages, the Native Development Kit (NDK) is available and contains the necessary libraries for compiling to an Android device. It also contains GDB for runtime debugging of a *native* process, or NDK-GDB.

A mobile application, or app, is needed to:

- interface to the hardware and user
- pass image pixel data to the native domain
- perform processing
- camera control (see Section 3)

This app is developed for the Nexus devices and is called *MPT App*.

At this point, the major processing functions should be defined: `ProcessRGRImageToBarcodes()` is the native function that searches the image for markers and identifies them with their unique ID code, and `ProcessRGRBarcodeToPose()` calculates the 6-DOF pose of each marker with a detected ID code. `ProcessRGRImageToBarcodes()` will be abbreviated to `ImageToBarcodes()` and `ProcessRGRBarcodeToPose()` to `BarcodeToPose()`, respectively. These majors functions are in the native

domain and are accessed through the *Java-Native Interface*, or JNI. The JNI is a specialized interface that allows data to pass from the java domain to the native domain and vice versa. The final processing architecture is shown in Figure 6.

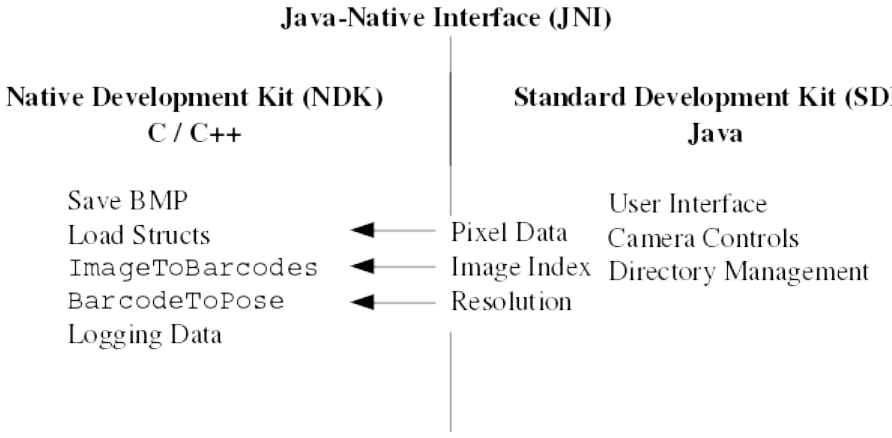

Figure 6: Overview of MPT Architecture on Android. Camera controls and user interface are in the Java domain. The JNI is used as little as possible due to complexity. MPT Processing occurs in C++, or the Native domain.

The Java-Native interface, or JNI, is an important step in the processing architecture and defining what data passes through the interface is helpful early in development.

2.2 Java-Native Interface

MPT requires defining various structs for the analyses. These include `StarburstControl`, `Circle`, `SineFitControl`, `AccIllumControl`, `CosineAmbiguityRecoveryControl`, `CameraModel`, and `TargetModel`. A possible solution is to make the JNI "porous", that is, all of the structs required for processing would be loaded in the Java domain and passed through the JNI to the native domain, and the native domain would strictly be used for processing. However, due to major architectural differences in memory management of C++ and Java, the most effective solution is using JNI as little possible. JNI becomes highly complex when user-defined data types, such as classes in Java, must be passed from Java to native. Thus, the simplest solution is only pass data through the JNI when there are no other options - such as camera pixel data that is exclusively retrieved from the Java domain.

2.3 Java Domain

MPT is designed to work with the Bitmap file type as it requires uncompressed images. Fortunately, the Android Application Programming Interface (API) SDK contains a `Bitmap` class to easily manipulate and load bitmap files. In order to process the pixels of an image, the first step in development is to process a

known image that has been successfully processed on the validated MPT system. The image used is shown in Figure 7.

Figure 7: The test image used for processing development. It is a known good image that was previously processed on a validated MPT system.

Black space is added to force the image to be the correct resolution of the Basler A404k camera model which is 2352 x 1726. The bitmap is loaded into a `Bitmap` type with Java method `BitmapFactory.decodeFile([Image Path])`, and subsequently the pixels are extracted with method `Bitmap.getPixels()`. This method places 32-bit `int` pixel data into an `int` array in the format Alpha-Red-Green-Blue (ARGB), each 8-bits, that is of length `ImageHeight` * `ImageWidth`. For MPT, the only channel that is required is the Alpha channel that defines pixel intensity. The pixel array of type `int` is then converted to `bPixel` array of type `byte[]` by taking only the alpha channel of 8-bits and placing it into the corresponding index of the array of bytes, each `byte` also 8 bits. This is to easily recast the pixel data to `unsigned char`, also an 8-bit type, required by the MPT algorithm in the native domain. Once the pixels are in the `bPixel` array, the call to the native domain commences, of course, after successfully compiling the native code.

2.4 Native Domain

2.4.1 Compilation for ARM Target

MPT Processing was originally developed in MathWorks MATLAB as it is highly mathematical in nature. With the MATLAB coder functionality, users are able to compile runnable C++ code to various markers, including the ARM processor used by the Nexus devices. MPT C++ code was generated targeting the ARM processor, producing a very large .cpp file along with header and .cpp helper files. The helper files define MATLAB types and various ways to handle said types. Header files defining the MPT API were also created.

To determine the required libraries, compilation is attempted and then typically fails until the library header files are included and their libraries are also compiled. Each source file is individually added until the compiler successfully creates the shared objects. The libraries required by MPT include:

- OpenCV, an open-source library containing helpful image manipulation functions
- LogRGRMessage, used to print logging information to a ring buffer
  - The Android architecture does not have a ring buffer, so logging information was printed to a text file

There are several more, however, the ones listed above are relevant to MPT processing development. The interface function, called `RGRWrapper()`, contains the JNI, creates the processing pthread (see Section 2.4.3), and performs all of the setup prior to calling `ImageToBarcodes()` and subsequently `BarcodeToPose()`. The final software package used in the RULA study contains a total of fifteen shared objects (.so) that are required for the MPT App to run.

2.4.2 Runtime Issues

Initially, running `ImageToBarcodes()` resulted in a segmentation fault (or SEGFAULT). Interestingly, the process did not always SEGFAULT at the same line of code. It was identified that the default `pthread` stack size was insufficient for `ImageToBarcodes()`. A new `pthread` was required such that the stack size could be increased to 3MB. This allowed the processing to finish without throwing a SEGFAULT.

2.4.3 MPT Test Processing Sequence

To process the image in Figure 7 in the native domain, the following steps occur to achieve ID code and 6-DOF pose data when the Java domain passes pixel data:

- Cast `jbyte` (the type Java `byte`, where the pixel data is stored, cast to a corresponding type in C++) to an `unsigned char`.
- Place image width, image height, image stride, index, and pixel data in `pthread_arg_struct` struct. This data all passes through the JNI.
- Setup the new pthread with 3MB of stack data.
- Start the thread and pass `pthread_arg_struct` as an argument to the new child thread.
- Populate the data structs needed by `ImageToBarcodes()` and `BarcodeToPose()`. These structs are of type `TargetModel_t` (populated with the generic marker model for `ImageToBarcodes()`), `CameraModel_t`, `StarburstControl_t`, `Circle_t`, `SineFitControl_t`, `AccIllumControl_t`, and `CosineAmbiguityRecoveryControl_t`.
  - The image shown in Figure 7 was taken with a Basler A404k Camera with a Schneider 28mm lens, and therefore that Camera Model is loaded into `pCameraModel` of type `CameraModel_t`. This is strictly for processing development. To process images taken by the Nexus camera, the Nexus camera model will be loaded which is generated from camera calibration. Camera calibration is discussed in Section 4.
- Allocate `ProcessingSpace`, a struct that is on the heap. This struct contains the working data used throughout the MPT algorithm.
- Call `ImageToBarcodes()`.
- For all MPT markers `found by ImageToBarcodes()`, calculate pose data for each marker by calling `BarcodeToPose()`.
- Print results to Results.txt.

2.4.4 Bulk Processing

Processing of multiple images must be done serially due to memory management considerations, as the resources are limited on an embedded system such as a Nexus device [17]. A single 13 Megapixel image from the Nexus 6 requires 238MB of processing space on the heap, so parallel processing could be detrimental until a a memory management scheme is constructed. The Nexus 6 is determined to be the best Nexus device currently available for RULA, as discussed further in Section 3.2. Memory considerations are also taken by saving the images first to flash memory before serially processing the images. A check-box option,

discussed in Section 3.2.4, can be used to either take bulk images, process bulk images, or both. All bitmap images located in `/sdcard/MPTProject/Images` are processed if bulk image processing is selected. Once bulk processing has begun, a progress bar appears as seen in Figure 8.

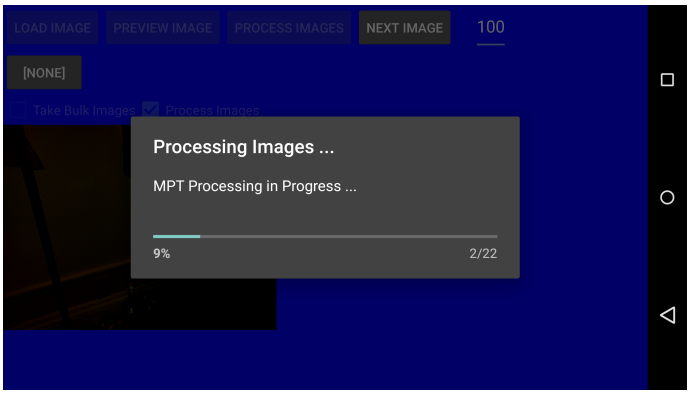

Figure 8: The Nexus 6 processing bulk images. It will save all 6-DOF pose data to a text file located on the device.

Text file `/sdcard/MPTProject/LogRGRMessage/Results.txt` contains 6-DOF pose return data from MPT, along with the corresponding ID code of each marker analyzed and image index.

2.4.5 Single Image Processing

For debugging purposes, a single image can be selected from the device for processing and is picked from a window as seen in Figure 9.

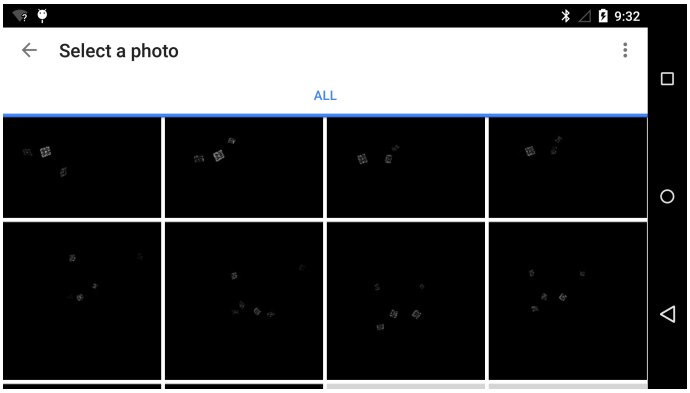

Figure 9: Screen for selecting a single image for processing (versus bulk processing). It saves the 6-DOF pose data to a text file with an image index of -1.

The result of the processing is also saved in Results.txt but with an image index of -1 instead of a number from 1 to `nImages`. The image also loads to the screen for viewing (Figure 10).

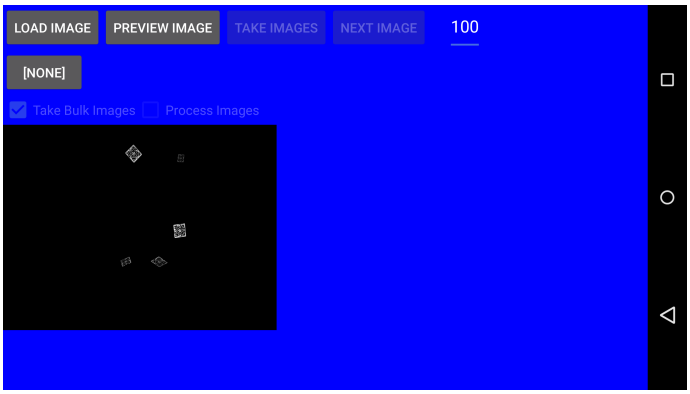

Figure 10: Once an image has been selected for processing, it will display on the screen and calculate the 6-DOF pose data.

2.5 MPT Processing Results

MPT returns the same 6-DOF pose data either with the 6 explicit values `PoseHat2.tSScknot` or a 4 x 4 homogeneous transform matrix `PoseHat2.ctT`. The characteristic of the homogeneous transform is shown in Table 5 and is further explained in Section 5.1. The position vector X, Y, and Z are in meters and are the same in PoseHat2.tSScknot and PoseHat2.ctT.

The image processed by the Nexus 6 in Figure 7 has the results shown in Table 1 and are juxtaposed with the results processed with the validated MPT system.

PoseHat2.ctT (Lab)			
-0.8048	-0.5872	-0.0856	0.3738
-0.5153	0.6188	0.5930	0.5029
-0.2946	0.5218	-0.8006	2.8499
0	0	0	1

PoseHat2.ctT (Nexus 6)			
-0.8048	-0.5872	-0.0865	0.3738
-0.5153	0.6188	0.5930	0.5029
-0.2946	0.5218	-0.8006	2.8499
0	0	0	1

Table 1: PoseHat2.ctT processing results from Figure 7. They are identical, thus proving processing capability on the Nexus device.

2.6 Nexus 6 Hardware Specifications

Relevant to processing, the Nexus 6 device used in this thesis has the following specifications:

- Qualcomm 2.7GHz Quad-Core Krait Processor (32-bit)
- 3GB LPDDR3 RAM
- Performance: 3.51 DMIPS / MHz
- Sony IMX214 Rear Camera
- 64 GB Flash Memory

2.7 Processing Development Conclusion

Processing is a major component of MPT. If the Nexus does not have the resources to perform such high intensity calculations or the libraries required to run MPT, MPT would not be ready to be ported to a mobile device. However, this section shows Nexus processing gives identical to results to the expected and validated results. However, the processing is not helpful if images from the camera cannot be passed to the MPT. Thus, the next important step is controlling the camera.

3 Android Camera Control for MPT

The goal of camera control is to be able to obtain unmodified pixel data from the camera and save and/or process them. Mobile devices, including Android devices, were not designed to have precise, data-collecting cameras. The intention is to make images *aesthetically appealing* - contrasted with MPT which requires *accura*cy. In this project, camera control development occurred on the original Android Camera API, but the original API has insufficient camera controls to meet the requirements of MPT. Development was migrated to the new Camera2 API.

3.1 Original Camera API

The infrastructure to calibrate the camera begins with the original Camera API which is supported on the Nexus 10. The MPT App is designed to take an image every 2 seconds, up to a user-defined number of images, grab the pixels, and save them to bitmap files.

3.1.1 Focus and Exposure

The Camera API is very basic and, for all intents and purposes, was designed to be used with the Autofocus feature. But to accurately calibrate the camera, the focus must be held *constant* to ensure repeatability. With the Camera API, focus is set to `FOCUS_MODE_INFINITY`, which blurs objects within a reasonable RULA distance and is therefore undesirable. Additionally, there are no exposure controls with the original Camera API other than exposure compensation.

3.1.2 Frame Data Format

MPT, as described in Section 2.3, requires only the alpha (intensity) channel of a bitmap. The cameras that are used with MPT are black and white digital cameras. However, the Android camera itself is a color camera. The RGB must be converted to gray scale.

Fortunately, the Nexus 10 preview format is NV21 encoding format which is a YUV type, and is determined by:

```
Parameters camParams;
camParams = mCamera.getParameters();
imageFormat = camParams.getPreviewFormat();
```

Fundamentally, YUV contains the luma data, in the 8-bit Y component, and the chroma data, located in the UV components, where every component is independent of each other [18]. The Y component of luma and the intensity component alpha of the bitmap format thus give the same information and are both 8-bits.

At every instance where preview frame data is needed, the method `setOneShotPreviewCallback()` is called. Once the camera returns with the preview frame data, `onPreviewFrame()` is executed and the MPT App stores the the YUV frame data in `byte[] FrameData`. The first `nWidth` * `nHeight` bytes are passed to the native function `SaveBMPWrapper()`, or Save Bitmap Wrapper, where `nWidth` and `nHeight` are the width and height of the image, respectively. This only passes the luma, or Y channel, data.

3.1.3 Save Bitmap Wrapper

The MPT library includes `ImageUtilities` that uses the open source library OpenCV for many image related functions. This suite is specifically used in this project to save the bitmap files from the frame data through the JNI function `SaveBMPFile()`. `ImageUtilities` takes the 8-bit pixel data and various header information to save the Bitmap to a file described by a file descriptor.

The wrapper `SaveBMPWrapper()` contains the JNI and makes the call to `SaveBMPFile()` within the `ImageUtilities` library. The frame data in the Java domain is passed to `SaveBMPWrapper()` along with the image width, height, stride, and index, and also the location to save the Bitmap. This data is wrapped into a struct of type `RGRImage_t` which contains basic information about the image and includes the pixel data. This struct is then passed to `SaveBMPFile()` for hard disk storage.

Care must be taken when taking into account the row stride, visualized in Figure 11 which is strictly dependent on the resolution of the preview data. If one assumes the row stride is equal to the image length, which is sometimes the case, the image will end up distorted. The row stride is returned by `mImage.getPlanes()[0].getRowStride()` in the Camera2 API.

3.1.4 Nexus 10 Device Calibration

After camera control was implemented for the Nexus 10, calibration was undertaken. The role of camera calibration in photogrammetry is discussed in Section 4. However, after several calibration attempts, camera calibration for the Nexus 10 was deemed unsuccessful. This is discussed further in Section 4.2.1.

More control over the camera is absolutely necessary for MPT, such as manual control of the exposure and focus distance. Around the time of unsuccessful Nexus 10 camera calibration, the new Camera2 API was released by Google.

3.2 Camera2 API

Through numerous calibration attempts, the original Camera API was discovered to be insufficient for MPT. Several new Nexus devices contain the new Camera2 API that allows for manual focus, exposure, and image

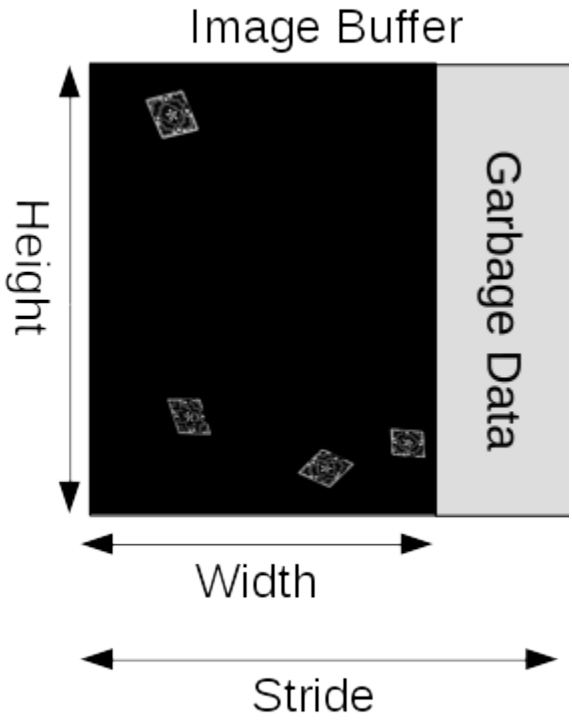

Figure 11: A visual of image stride. If the programmer is not careful, the garbage data can accidentally be included in an image.

processing pipeline control - all characteristics discovered to be necessary for MPT on a mobile device. The Nexus 9 and then the Nexus 6 are acquired as they both support the Camera2 API. The Nexus 6 phone is ultimately used for RULA (see Section 3.2.3).

The camera effort with the original Camera API discussed in Section 3.1 has been overhauled. The Camera2 API contains a suite of new features: focus control, exposure control, post-processing ability, full control of the image-processing pipeline, obtaining RAW data format, and more. It also increases complexity, as the Camera2 API runs *asynchronously* to the Nexus clock. Thus, image data has to be formally requested with a `CaptureRequest` to the camera controller.

3.2.1 Obtaining Images for MPT

The method `openCamera()` is called which in turn calls `setUpCameraOutputs()` and `configureTransform()` and directly interfaces with the API to request to open the camera. The method `setUpCameraOutputs()` defines the resolution and the image format (in this case `ImageFormat.YUV_420_888`, a YUV format similar to NV21 discussed in Section 3.1.2) along with defining the orientation of the returned image. The method `configureTransform()` simply rotates the image data if the Nexus is in landscape mode, which MPT App is by default.

Once the camera is opened, the method `createCameraPreviewSession()` is called. This is a one-time call that builds the camera preview request to the camera via `createCaptureRequest()`, adds the SurfaceTexture (the texture that holds the preview images) as a target for the incoming data, and uses `CaptureRequest.Builder` to set the focus (in diopters), exposure, and a slew of other parameters related to the preview mode defined in the API. Preview mode can be seen in Figure 12.

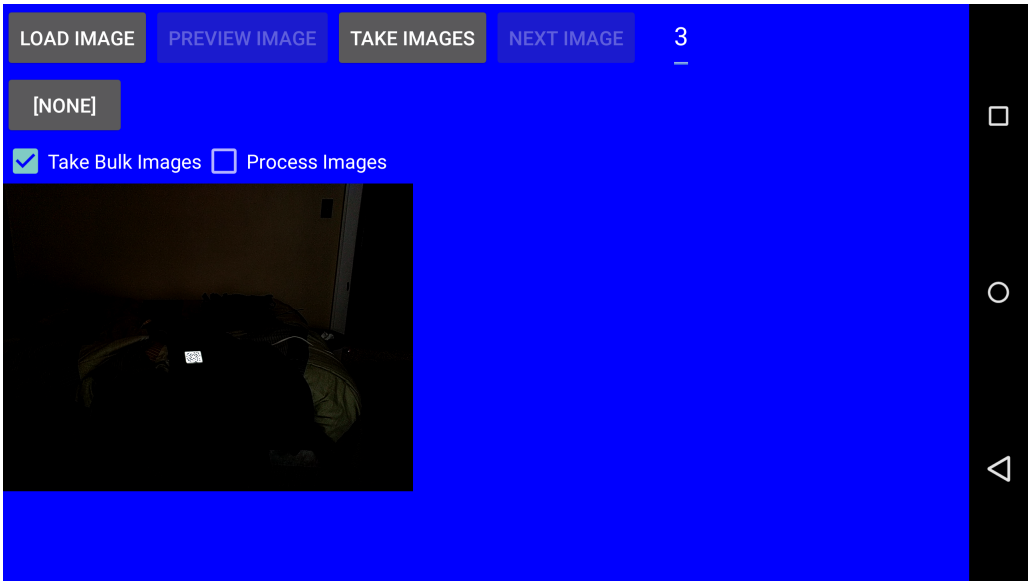

Figure 12: The Nexus 6 MPT App in Image Preview mode. Pressing TAKE IMAGES will begin image collection.

A *still* image request is similar to a preview request, except the builder is notified that the type of request is `CameraDevice.TEMPLATE_STILL_CAPTURE` versus `CameraDevice.TEMPLATE_PREVIEW`. Simply put, `CameraDevice.TEMPLATE_STILL_CAPTURE` prioritizes quality over frame rate, where `CameraDevice.TEMPLATE_PREVIEW` prioritizes frame rate over quality. It is at this point that the focus, exposure, and flash mode are set for a still image capture. Post-processing algorithms are turned off at this time (see Section 3.2.2).

The request is sent to the camera through `CameraCaptureSession.capture()` and awaits a callback when frame data becomes available. `OnImageAvailableListener()` is called and `ImageReader.acquireLatestImage()` grabs the data from the camera buffer. The method `Image.getPlanes()[0].getBuffer()` returns the Y component of the YUV frame data. The Y component is again passed to SaveBMPWrapper in the native domain as described in Section 3.1.3.

3.2.2 Camera2 API Image Post-Processing

The Camera2 API includes a host of post-processing algorithms to cater to consumer desires in their photos: noise-reduction, soft-edges, auto-white balance, and more. For MPT, raw pixel data is ideal. Under certain circumstances, the post-processing introduces artifacts in the images of motion tracking markers as seen in Figure 13. The exact nature requires more investigation, but the CaptureRequest.NOISE_REDUCTION_MODE has the largest impact on distorting the markers and must be turned off.

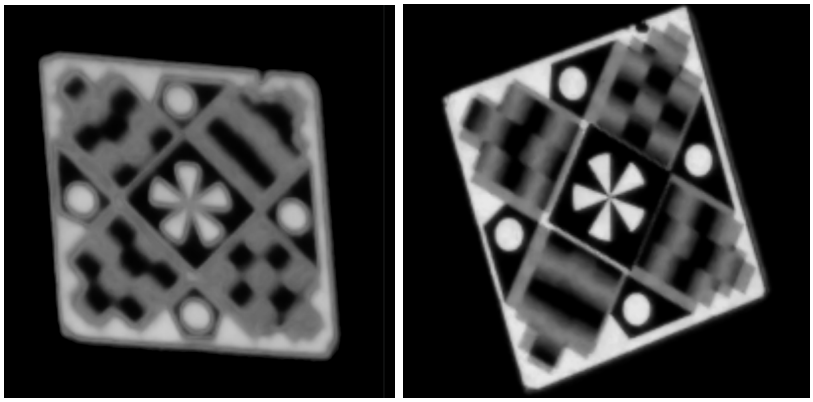

Figure 13: A comparison of post-processing. Some markers in various images result in the left image if post-processing is enabled. If post-processing is disabled, the distortion artifacts disappear (right image).

Certainly, the MPT algorithm cannot detect the motion tracking marker shown in Figure 13 (left). All available post-processing functions listed in the API are turned off manually for each `CameraDevice.TEMPLATE_STILL_CAPTURE` request:

- `CaptureRequest.CONTROL_EFFECT_MODE`
- `CaptureRequest.CONTROL_AWB_MODE`
- `CaptureRequest.COLOR_CORRECTION_ABERRATION_MODE`
- `CaptureRequest.EDGE_MODE`
- `CaptureRequest.HOT_PIXEL_MODE`
- `CaptureRequest.NOISE_REDUCTION_MODE`
- `CaptureRequest.SHADING_MODE`
- `CaptureRequest.STATISTICS_LENS_SHADING_MAP_MODE`

3.2.3 Getting Camera Technical Data

The Camera2 API allows for retrieval of camera characteristics through the `CameraCharacteristics.get()` method. This allows for a broad array of camera, camera controller, and image pipeline properties to be retrieved through a running app. Therefore an app can determine the capabilities of the camera before allowing or disallowing certain functionality. For example:

- `CameraCharacteristics.SENSOR_INFO_EXPOSURE_TIME_RANGE`
- `CameraCharacteristics.LENS_INFO_MINIMUM_FOCUS_DISTANCE`
- `CameraCharacteristics.LENS_INFO_AVAILABLE_FOCAL_LENGTHS`
- `CameraCharacteristics.LENS_INFO_FOCUS_DISTANCE_CALIBRATION`

There are substantially more `CameraCharacteristics` that can be retrieved and can be viewed in the API. A characteristic of importance to MPT, `CameraCharacteristics.LENS_INFO_FOCUS_DISTANCE_CALIBRATION`, reports the ability of the lens to return to the same focus distance at different points in time by returning:

- `LENS_INFO_FOCUS_DISTANCE_CALIBRATION_APPROXIMATE`
  - Focus distance repeatability may depend on device orientation, age, or temperature. The actual focus distance is approximate.
- `LENS_INFO_FOCUS_DISTANCE_CALIBRATION_CALIBRATED`
  - Focus distance repeatability has good accuracy and corresponds to the real distance to the in-focus plane.
- `LENS_INFO_FOCUS_DISTANCE_CALIBRATION_UNCALIBRATED`
  - Focus distance repeatability is poor and the diopters setting does not correspond to real distance to the in-focus plane.

These descriptions are listed in the Android API. The exact nature of why an android device's camera is calibrated, approximated, or uncalibrated requires an investigation but appears to be at the hardware level. The Nexus 9 camera returns APPROXIMATED. This is cause for concern due to the dependence on precise camera calibration of MPT. A quick investigation reveals a different device, the Nexus 6, returns *CALIBRATED*.

Characteristic	Nexus 9	Nexus 6
Image Format	JPEG, RAW_SENSOR, YV12, YUV_420_888	RAW10, JPEG, RAW_SENSOR, YUV_420_888
Supported Hardware Level	Limited	Full
Available Apertures	1.3	2.0
Available Focal Lengths	3.097	3.82
Focus Distance Calibration	Approximated	Calibrated
Exposure Time Range (nanoseconds)	[16379, 811608704]	[20660, 674969300]
Available Capabilities	Backwards Compatible, Manual Sensor	Backwards Compatible, Manual Sensor, Manual Post Processing, Read Sensor Settings, Burst Capture, RAW

Table 2: Nexus 9 versus Nexus 6 camera comparison. These are characteristics relevant to MPT and the Nexus 6 is better suited than the Nexus 9 for MPT due to the improved camera control capabilities.

The `CameraCharacteristics.get()` method allowed for a complete juxtaposition of the Nexus 9 versus the Nexus 6 camera functionality, several examples that are relevant to MPT are seen in Table 2.

The Nexus 6, per this comparison of a handful of camera characteristics, is better suited to run MPT due to the high levels of manual capability, and thus the project moved to the Nexus 6 platform to improve confidence in camera calibration and MPT operation going forward.

3.2.4 Using the MPT App on the Nexus 6

MPT App, the Android software developed for this thesis, did not require modification for the Nexus 6 as the Camera2 API and its ARM processor are both compatible with the Nexus 9. When the MPT App is first opened on the Nexus 6, the screen appears in Figure 14. The background is blue so the preview `SurfaceTexture` edges can easily be seen. The LOAD IMAGE button is to perform an MPT calculation on a single image as described in Section 2.4.5.

The button that says [NONE] can be pressed and scrolled through to apply a label to the image file name that will be taken. The button will update with the label value as seen in Figure 15. For example, if a label other than [NONE] is selected, all images will have the form ImageXXX_LABEL.bmp, where XXX is the sequential image number starting with 001 and LABEL is a hard coded Java String to insert into the image file name. If [NONE] is selected, the image will simply be ImageXXX.bmp with no underscore.

Labels to the image file names are absolutely necessary for RULA. The short exposure, discussed in Section 3.2.5, makes it difficult to see objects other than the retroreflective markers in the image unless those objects are creating their own light. An example of an image taking by the Nexus 6 in the actual RULA

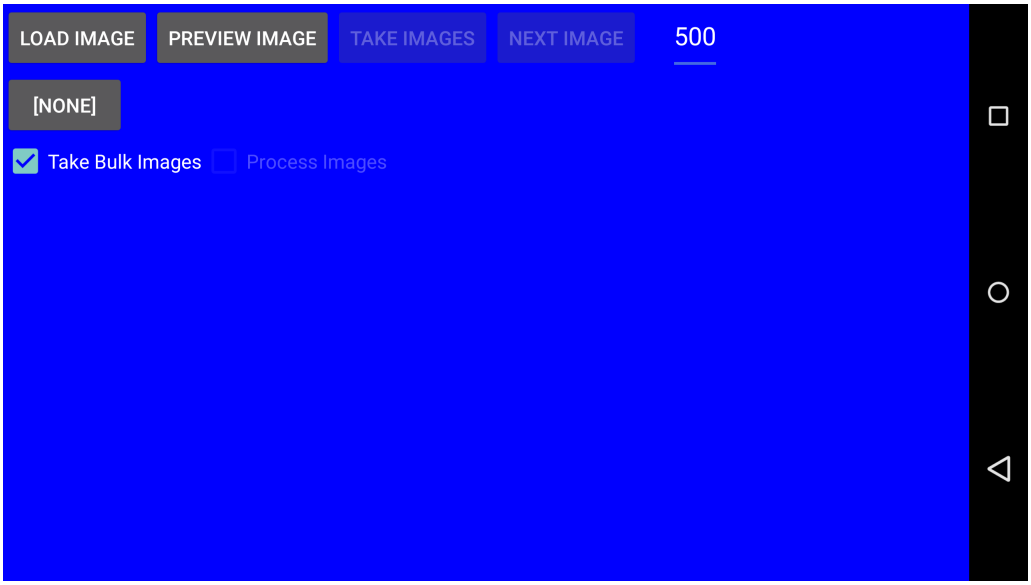

Figure 14: The MPT App screen, on the Nexus 6, right after it is launched.

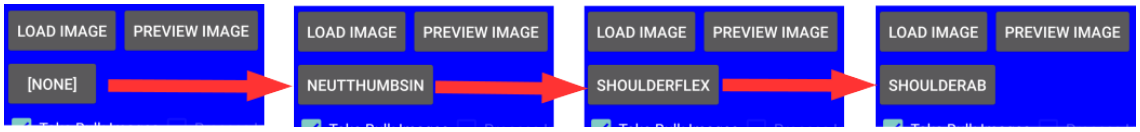

Figure 15: Pressing this button cycles through names relevant to RULA such that the image names are useful.

study is shown in Figure 16.

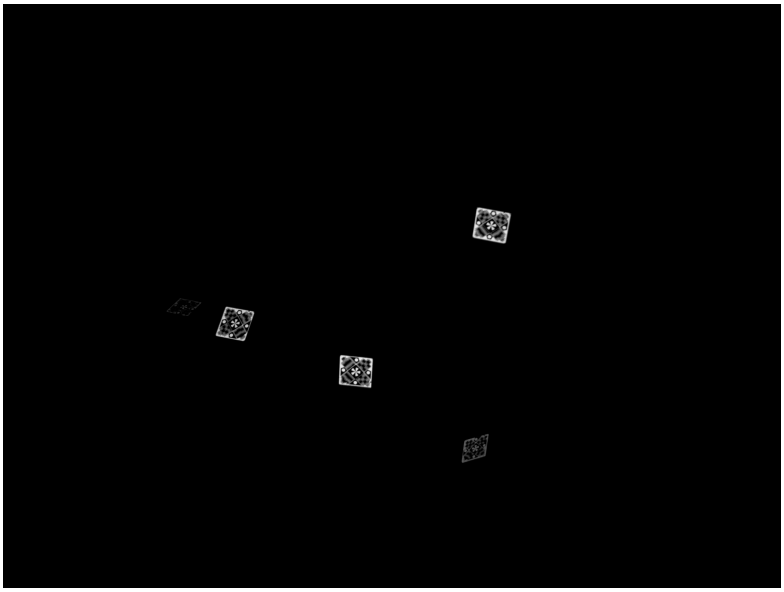

Figure 16: A RULA study image taken on the Nexus 6. The file name must have text indicating the pose, as the pose cannot be determined simply by looking at the image.

It is very difficult to deduce the pose of the subject just by viewing the image alone thus labels to the image file names are needed to understand the angles.

To take an image, the PREVIEW IMAGE button is pressed after the MPT App is launched to open the camera and setup for image collection. This results in the screen shown in Figure 12. The number of maximum images is set in the `EditText` box. While in this mode, the user can decide to perform the following via the check boxes just above the camera preview:

- Image collection only (Figure 12) and press the TAKE IMAGES button
- Both image collection and processing (Figure 17) and press TAKE/PROCESS IMAGES

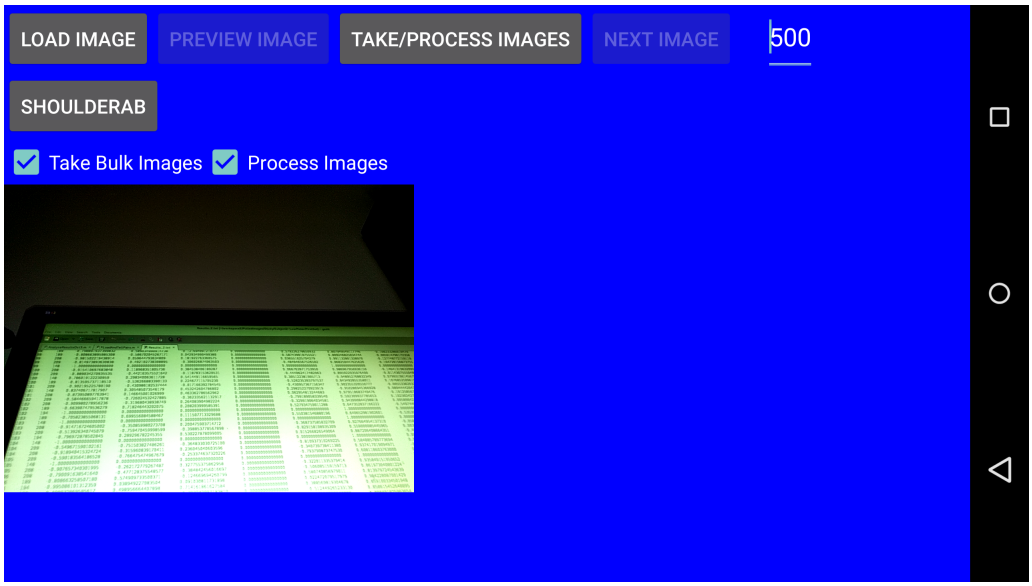

Figure 17: The Nexus 6 MPT App in TAKE/PROCESS IMAGES Mode (both check boxes selected). This will first collect images then subsequently process them.

- Processing of images only (Figure 18) that are already in the working directory.

Once image collection begins after either pressing TAKE IMAGES or TAKE/PROCESS IMAGES, the first image is taken after a two second delay to assist in stabilization. Every subsequent image requires pressing NEXT IMAGE to take the next image, as seen in Figure 19.

The images are saved in the /sdcard/MPTProject/Images/ directory. The image collected as seen in Figure 19 can be seen in Figure 20.

3.2.5 Nexus 6 Camera Physics

The Nexus 6 Camera sensor is a Sony IMX214 Exmore R CMOS with a rolling shutter. A rolling shutter will expose rows of pixels, depending on architecture, sequentially [19]. The standard MPT system contains a

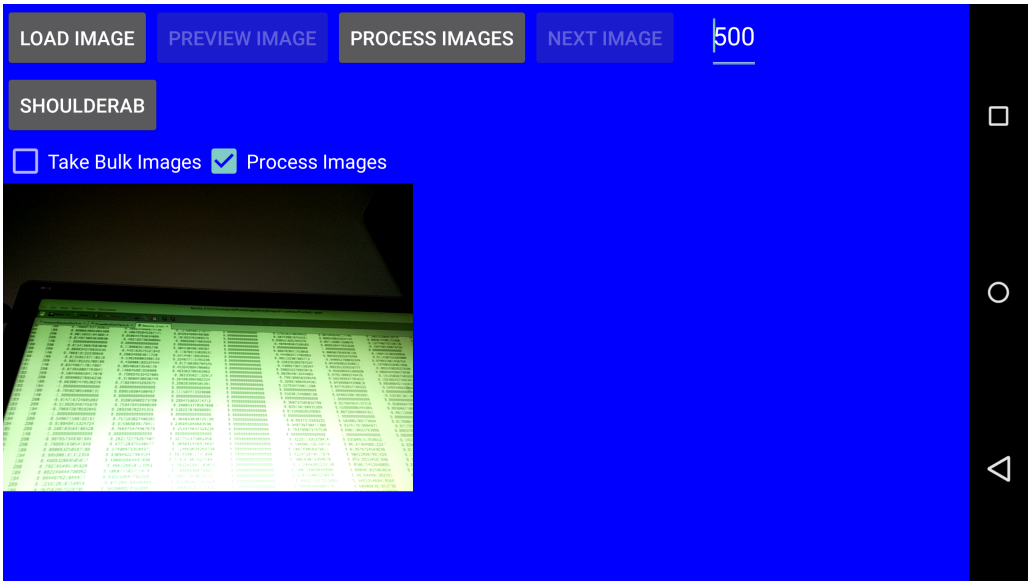

Figure 18: The Nexus 6 MPT App in PROCESS IMAGES Mode. When PROCESS IMAGE is pressed, all images in the working directly will be processed.

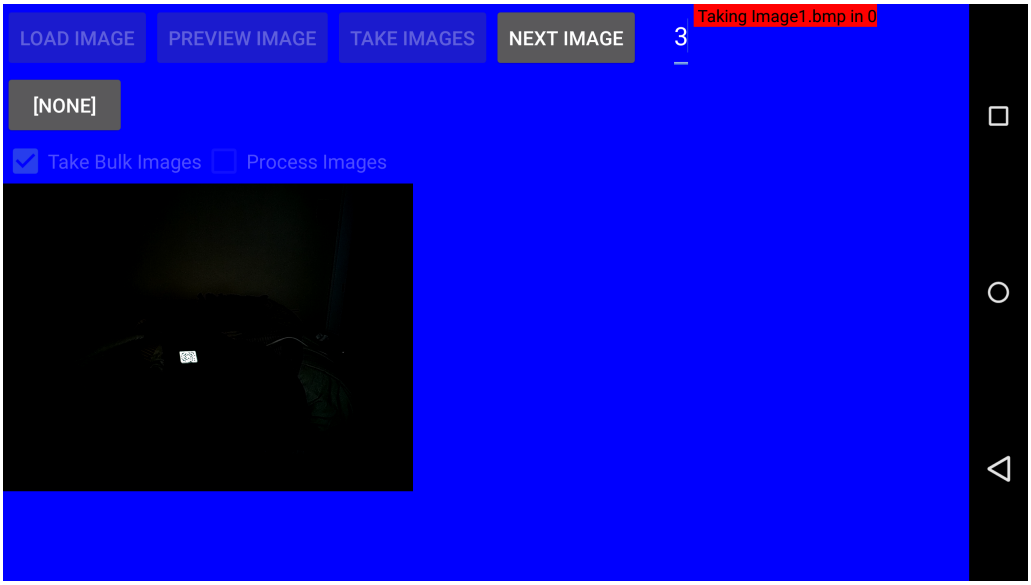

Figure 19: The Nexus 6 MPT App collecting images. The button NEXT IMAGE must be pressed before the next image will be taken.

camera with a global shutter which exposes all pixels at the same time. Rolling shutters have the drawback of exposing image regions at different instants in time [20]. For a system that is very sensitive to image distortions, such as MPT, this can pose the the problem if either the camera and/or the markers are moving. Reducing the amount of time required for frame data to be capture is therefore ideal and is the simplest method to reduce rolling shutter distortions - although rolling shutter compensation methods do exist [21].

The flash setting can either be off, torch mode, or single flash. Torch mode means the flash is on

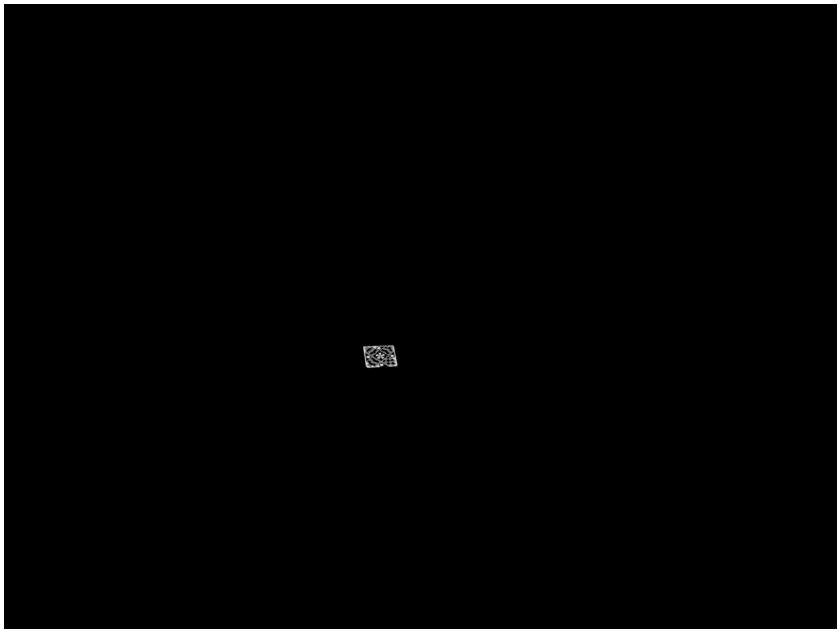

Figure 20: The resultant Image from Figure 19.

continuously but at a lower brightness. Single flash is the brightest, and therefore was used with a very quick exposure to reduce the effects of the rolling shutter. This helps reduce the motion blur if the motion tracking marker and/or camera is in motion during image capture. The markers and camera calibration tool themselves are made of retro reflective material that require little incident light to be detected. The exposure rate is set to 750µsec and 300µsec for use with the calibration tool and markers, respectively. The exposure time must be slightly higher for the calibration tool due to the its lower reflective capacity. If the exposure rate is too high, saturation of the marker can occur (Figure 21).

The edge of the starburst landmark and gradient of the Moiré pattern are indiscernible to MPT and the image would not be processable.

Focus is set in units of diopters (or $^1/_m$) and is set to 0.8 which gives an in-focus distance of approximately 1.25 meters - an appropriate distance for RULA.

3.2.6 Nexus 6 Camera Control Conclusion

Camera control is accomplished which is required by both camera calibration, such that images can be taken and analyzed for distortion components, using MPT and ultimately capturing images for RULA.

Figure 21: An example of marker saturation. If exposure is too high, saturation can occur and the marker cannot be processed.

4 Camera Calibration

Camera calibration is required for MPT and general photogrammetry due to nonlinear geometric properties of the camera lens [22]. The Nexus device lens has a specific distortion to that must be compensated for to achieve reliable RULA results.

The two main types of lens distortion are radial distortion and tangential distortion. Radial distortion is defined as outward distortion with respect to the image center, while tangential distortion is defined as error that flows in one direction [23, 24]. Mobile device cameras similar to the Nexus 6 have successfully been calibrated in the past [25, 26, 27].

Radial distortion, according to Brown [23], is characterized by Equation 1 and tangential distortion, or decentering, is characterized by Equation 2 [24, 28].

$$\eth \mathrm{r} = K_1 \cdot r^3 + K_2 \cdot r^5 + K_3 \cdot r^7 + K_4 \cdot r^9 + K_5 \cdot r^{11} \cdots \tag{1}$$

$$\eth^j P_a\_decen = \begin{bmatrix} \eth x \\ \eth y \end{bmatrix} = \begin{bmatrix} \left(P_1(r^2 + {}^jX_a^2) + 2P_2{}^jX_a{}^jY_a\right)\left(1 + P_3 r^2\right) \\ \left(P_2(r^2 + {}^jY_a^2) + 2P_1{}^jX_a{}^jY_a\right)\left(1 + P_3 r^2\right) \end{bmatrix} \tag{2}$$

Camera calibration will return the odd-ordered polynomial coefficients K_1, K_2, $K_3, \cdots$, K_n and P_1, P_2, and P_3 as lens distortion compensation parameters.

4.1 Typical MPT Camera Calibration

The standard camera calibration technique for MPT is the holding of the calibration tool (Figure 22) while it is facing the camera and moving it around the field of view of the camera during image collection. Areas of importance are the edges of the imager as the distortions are at the highest magnitudes. The calibration technique described in [28] can identify the landmarks and their actual distance from each other to return K_1, K_2, and K_3, and $P_1$and P_2 (P_3 is not used in MPT) which are loaded at runtime of `ImageToBarcodes()` and `BarcodeToPose()` for distortion compensation within the `CameraModel` struct. It is sufficient for standard MPT cameras for the radial distortion polynomial to go to $K_3 \cdot r^7$ as they are not wide angle lenses. Due to the very quick exposure along with the global shutter of the standard MPT cameras, motion of the calibration tool during collection of calibration images does not interfere with the calibration process.

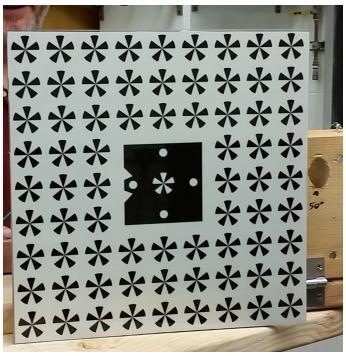

Figure 22: The calibration tool used for camera calibration. There is a cylindrical handle behind it such that it can be moved around by hand in the camera field of view.

4.2 Camera Calibration for Android

4.2.1 Nexus 10 Camera Calibration

The technique described in Section 4.1 returns wildly varying results on the Nexus 10 with the original Camera API. Due to lack of control of basic camera features, there is substantial motion blur and often saturation of the calibration tool, similar to Figure 21. Having manual control over the camera is essential to MPT and the camera technology in the Nexus 10 is, simply put, insufficient for MPT.

The Nexus 6 with the Camera2 API (See Section 3.2) provides more control and allows for a much faster exposure time and increases confidence in successful camera calibration.

4.2.2 Nexus 6 Calibration Technique

Now with control over exposure, to help reduce motion blur and saturation, and focus, camera calibration is attempted again via the same mechanism of physically moving the calibration tool shown in Figure 22 during image collection. Inconsistent but processable results are returned since there is neither visual motion blur nor saturation. The rolling shutter is determined to be of consequence, as is described by Su [20]. A diopters setting of 2.5 $^1/_m$ is used for camera calibration which gives an approximate focus distance of 0.8 meters.

An option is to perform rolling shutter compensation although that is a daunting task in its own right due to the lack of public information about the Nexus 6 Camera. A second option is to immobilize the camera and calibration tool while collecting calibration images to negate the rolling shutter effects. To immobilize the calibration tool, a holder was assembled, as shown in Figures 23 through 27. The holder is weighted down by two bricks that can be seen in Figure 23 to prevent the tool from tipping and to dampen any vibrations which could exacerbate the rolling shutter effects.

The holder sits on a plank held up by two ladders, one on each side, such that that holder and calibration

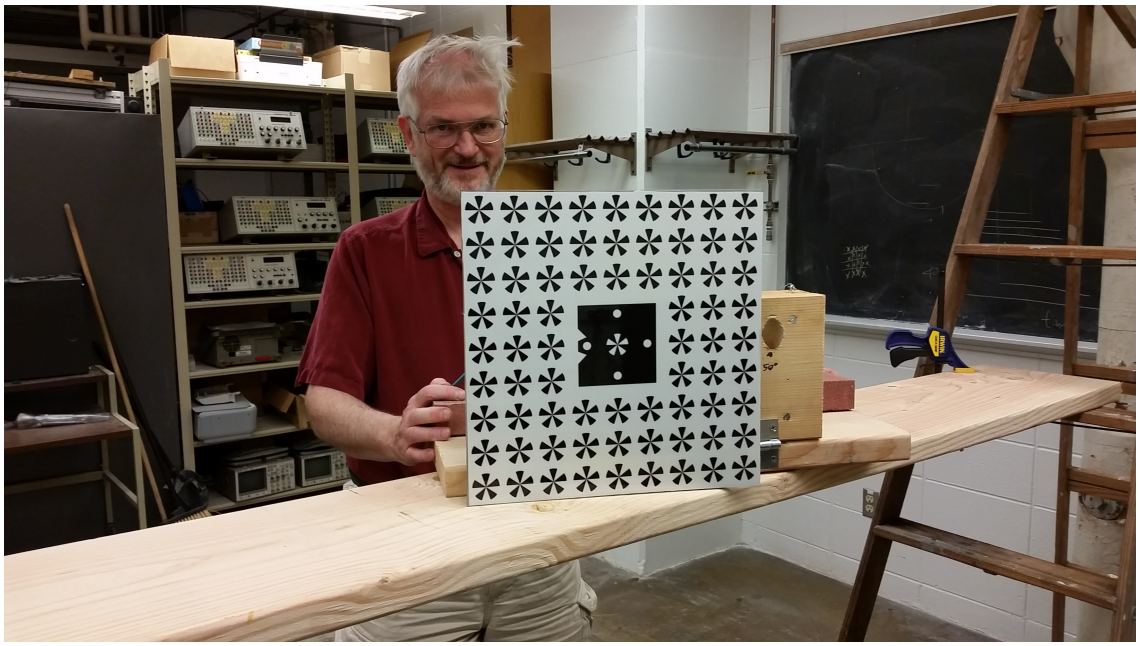

Figure 23: The calibration tool must be immobile to mitigate rolling shutter effects. Thus, a weighted assembly is built to keep the calibration tool immobile.

tool can move in the X direction as seen in Figures 23 and 24. The tool can also move in the Y-direction, or up and down, by moving the plank to different ladder steps (Figure 24). Z-direction movement, or linear distance from the Nexus 6 camera, is achieved by physically moving the entire assembly, ladders included, forward or backward.

The calibration tool's cylindrical handle fits into a drilled hole in the calibration tool holder that allows for Z-Axis rotations (Figure 25).

Another hole was drilled in the calibration tool holder such that there is upwards rotation about the X-Axis as seen from Figure 25 to Figure 26. Hinges in the calibration tool holder can be seen in Figure 26 to allow for downwards rotation about the calibration tool's X-Axis.

Again, rotations about the calibration tool's Z-Axis can occur even with a upward rotation about the X-Axis due to the cylindrical handle of the calibration tool fitted in the drilled holes, as seen from Figure 26 to Figure 27. The difference can be seen by the two triangles in the black region of the center of the calibration tool.

Y-Axis rotations can occur by pulling one of the calibration tool holder's edges forward or backward. Pose diversity of the calibration tool is necessary during image collection for robust calibration results, hence the many forms of linear and angular movement of the calibration tool - all while maintaining tool and camera immobility for each image.

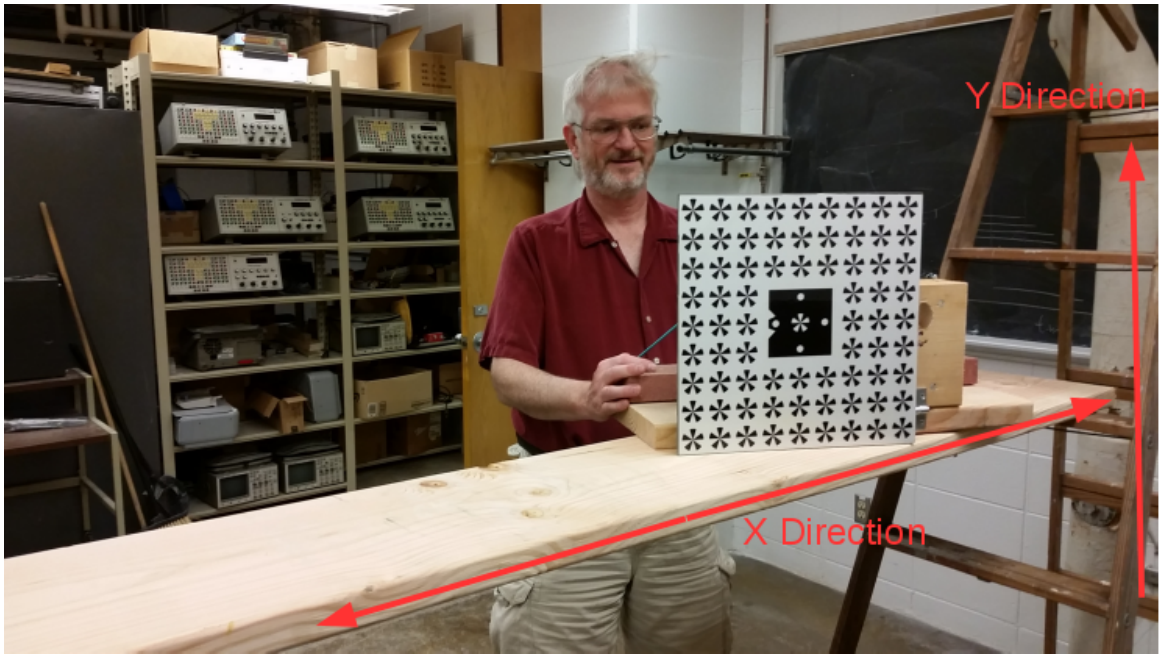

Figure 24: The calibration tool assembly can slide in the X direction by moving along the plank, in the Y direction by moving up or down the ladder steps, or in the Z direction by moving everything forward or backward (ladders included).

Figure 25: Z-Axis rotations can occur due to the drilled holes and calibration tool's cylindrical handle.

4.2.3 Nexus 6 Calibration Sequence

Six calibration sets are performed to compare variations in results due to:

Figure 26: X-Axis rotations can occur upward with drilled holes that are at an angle, or downward with the hinges seen in the figure.

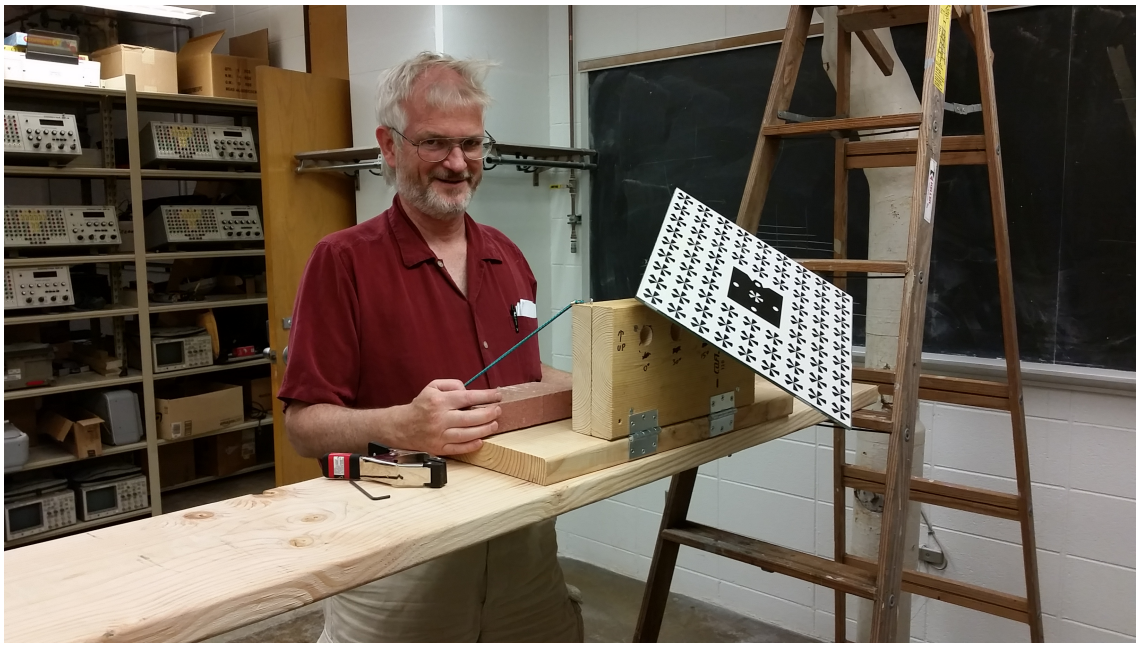

Figure 27: Again, the calibration tool can rotate in the Z-direction even with an X-axis rotation.

- Changing diopters setting to 0 $^1/_m$ (infinite focus distance) and then back to 2.5 $^1/_m$
- Allowing 24 hours to pass and collecting another set
- Restarting the MPT App

- Creating a mechanical disturbance to the Nexus 6 by picking up and setting the device down while the MPT App is running

Camera calibration in progress can be seen in Figure 28.

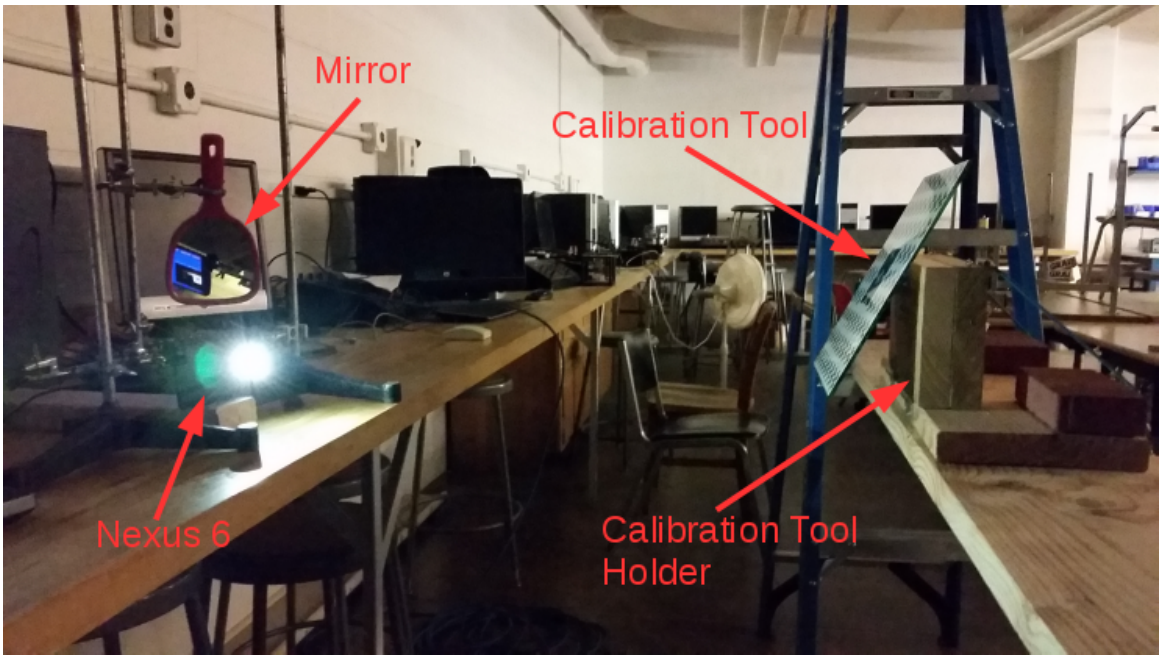

Figure 28: The actual camera calibration in progress. The calibration tool and the holder assembly are on the right, with the Nexus 6 on the left.

A mirror is placed behind the Nexus 6 to allow the operator to see the preview image such that the calibration tool does not go out of the camera's field of view. Each take has the following sequence:

- Take several pictures at or near the center of the imager with the calibration tool rotating about all angles (pitch, roll, and yaw). This occurs at approximately 1.5 meters from the Nexus 6 (or 1.5 meters in the Z-direction) and is referred to as Z-Center. A subset of these images can be seen in Figure 29.

- Sweep at the Z-Center position around the field of view of the camera. A subset of such images can be seen in Figure 30.

- Sweep at the Z-Close position, defined as approximately 0.75 meters. A subset of Z-Close images can be seen in Figure 31.

- Sweep at the Z-Far position, defined as approximately 2.5 meters. A subset of Z-far images can be seen in Figure 32.

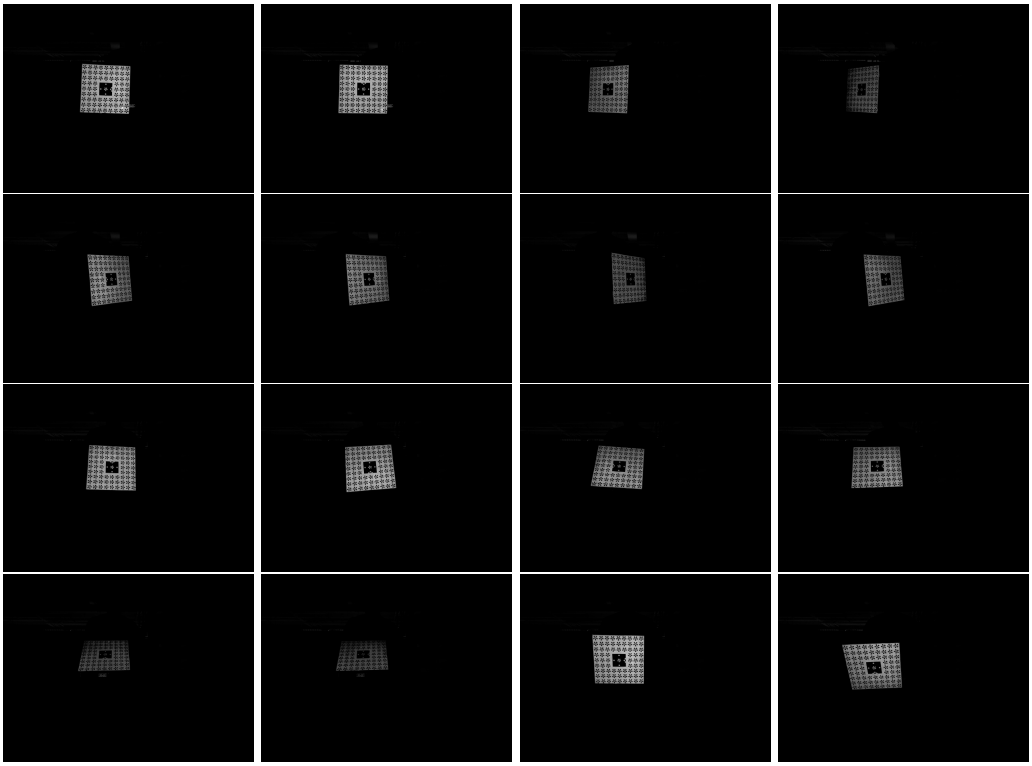

Figure 29: A subset of calibration images at the Z-Center distance (1.5 meters). The calibration tool is rotated about all axes near the center of the imager.

Each camera calibration take creates approximately 400 images. The majority of images are created from the Z-far position since the further away the calibration tool is in the Z-Axis, the more images that are required for a full field-of-view sweep.

4.2.4 Nexus 6 Camera Calibration Data

The camera calibration process returns data on the camera characteristics and the images processed. This is helpful to determine if the calibration technique is successful. For example, the detection of landmarks for the first calibration attempt (Take 1) at various locations in the field of view as seen in Figure 33.

Certainly there is significant coverage in the field of view. The focus distance is also calculated and can be seen in Figure 34, along with the distance of the calibration tool from the Nexus 6 camera center in the images analyzed.

These plots show this set of images provide a large amount of data points and provides confidence for an accurate calibration.

4.2.5 Comparative Nexus 6 Camera Calibration Results

Six camera calibration takes are analyzed with the following relationships between the takes:

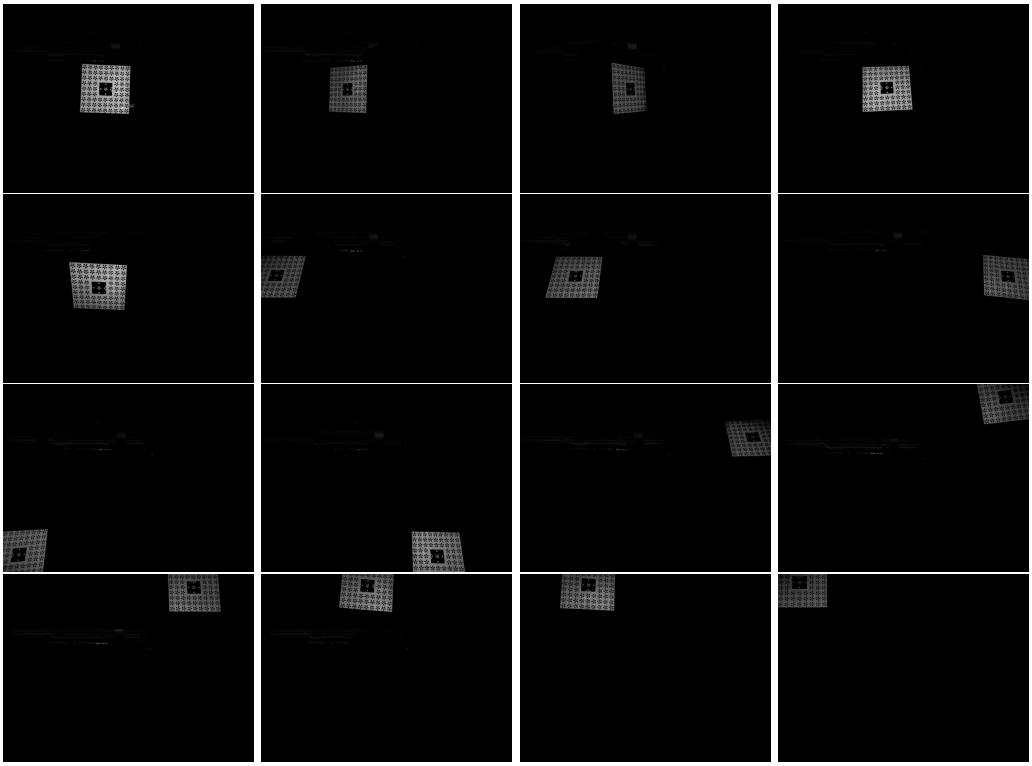

Figure 30: A subset of calibration images at the Z-Center distance with sweeps around the field of view of the Nexus 6 camera with Z-Axis rotations.

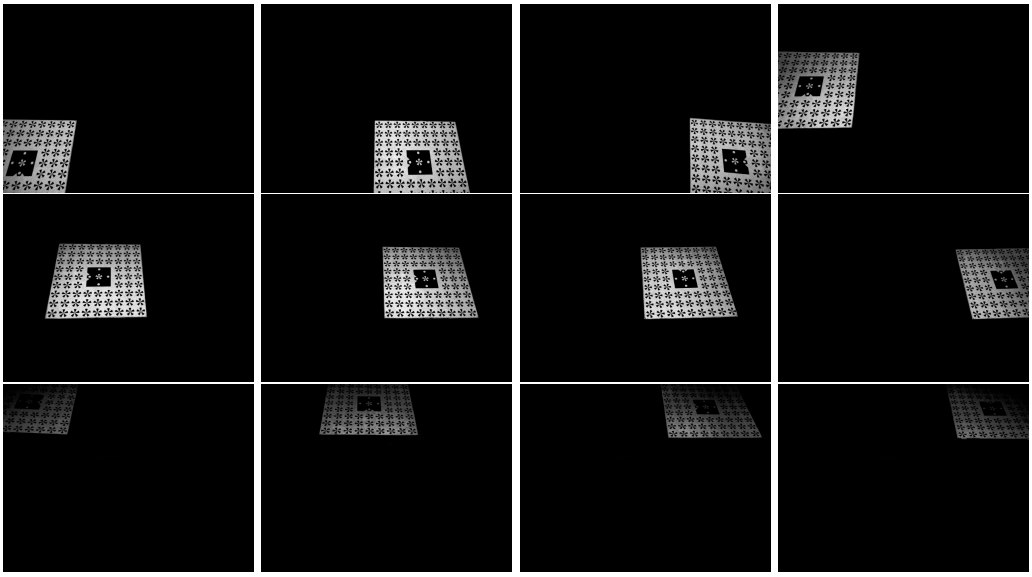

Figure 31: A subset of calibration images at the Z-Close distance (0.75 meters). The tool is swept around the field of view with Z-Axis rotations.

- Take 1 and Take 2: Changing the diopters setting to 0 $^1/_m$ (infinite focus distance) and then back to 2.5 $^1/_m$ without restarting the MPT App

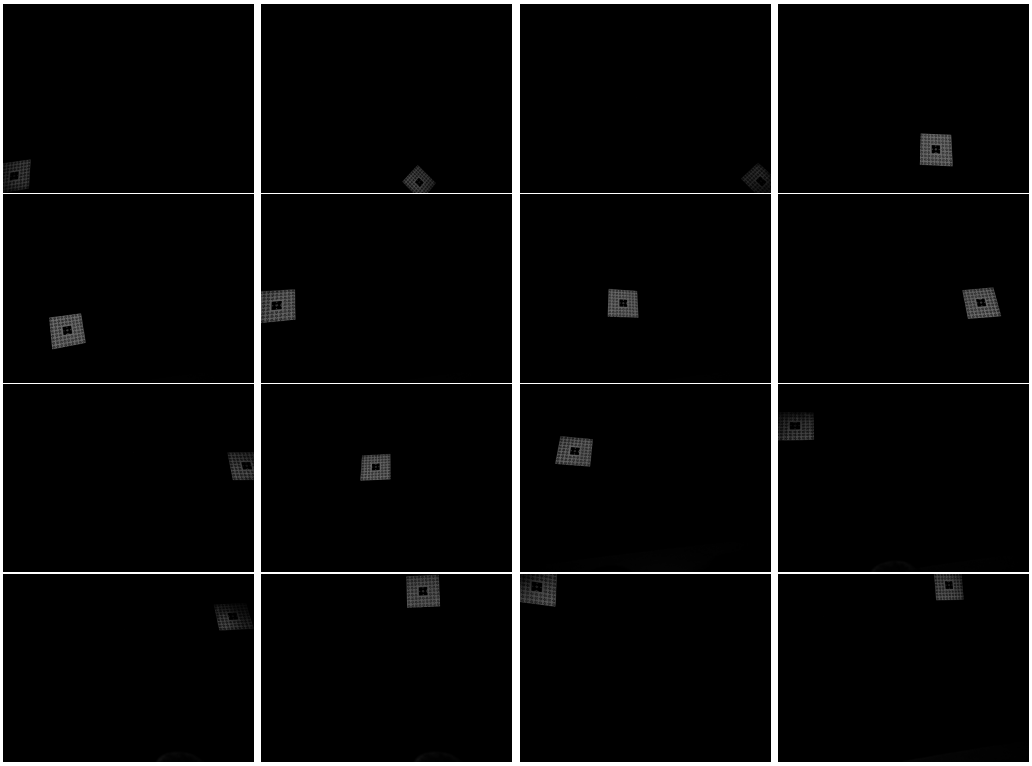

Figure 32: A subset of calibration images at the Z-Far distance (2.5 meters). Again, the tool is swept around the field of view with Z-Axis rotations.

- Take 2 and Take 3: 24-hour pass, allowing for the Nexus 6 to cool down and power off, re-charge, then turn on and warm-up
- Take 3 and Take 4: Restarting the MPT App
- Take 4 and Take 5: Another 24-hour pass
- Take 5 and Take 6: Mechanically disturbing the Nexus 6 by picking it up and setting it crudely on the table

The camera calibration results can be seen in Table 3.

Values K1, K2, K3, K4, and K5 are all radial distortion polynomial coefficients, as described in Equation 1, P1 and P2 are tangential distortion components per Equation 2, Cp is the calculated focal length in millimeters, kx and ky are the total pixels per millimeter in the x and y direction, respectively, and k0 and y0 are the optical center of the imager in pixel coordinates.

As described in Section 4.1, MPT typically calibrates the radial distortion out to $K_3 \cdot r^7$, or three terms. However, smart phones and tablets, such as the Nexus 6, have wide-angle lenses [29] which results in more dominant high order terms [30]. Therefore, camera calibration calculated the radial distortion to $\cdots K_5 \cdot r^{11}$,

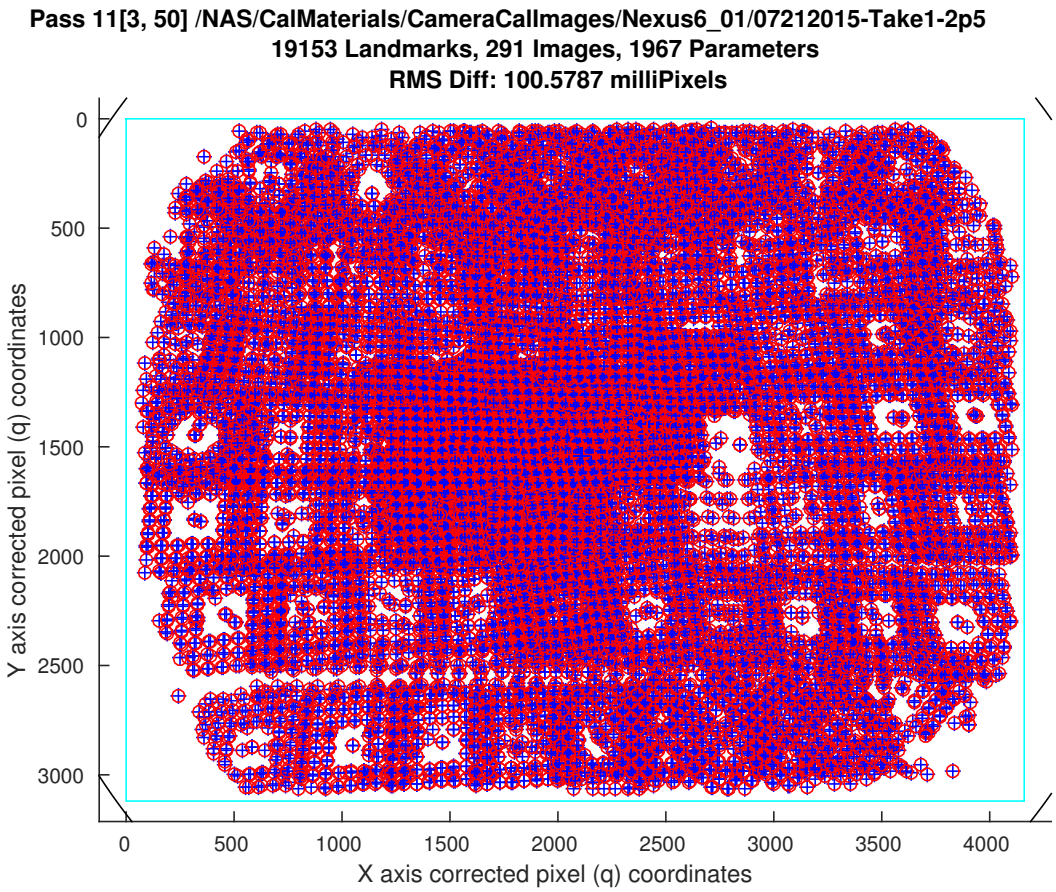

Figure 33: All of the starburst landmarks found on the calibration tool during the Take 1 image set. Notice there is significant coverage.

Values	Take 1	Take 2	Take 3	Take 4	Take 5	Take 6
Cp [mm]	3.826	3.8293	3.8313	3.8275	3.8263	3.8269
kx [px/mm]	892.86	892.86	892.86	892.86	892.86	892.86
ky [px/mm]	892.96	892.69	893.11	893.22	893.08	893.13
ktheta	0	0	0	0	0	0
k0	2101.0	2102.1	2102.4	2101.7	2102.3	2100.9
y0	1548.4	1549.3	1546.7	1549.7	1549.6	1547.4
K1	-0.016836	-0.016346	-0.017536	-0.018044	-0.017239	-0.016991
K2	0.0064702	0.005986	0.0073076	0.0075864	0.0067895	0.0066888
K3	-0.00090557	-0.00071541	-0.0012688	-0.0013162	-0.0010153	-0.0010028
K4	6.5752e-06	-2.5542e-05	7.3291e-05	7.3256e-05	2.5306e-05	2.4759e-05
K5	4.7685e-06	6.7294e-06	3.9243e-07	7.8278e-07	3.529e-06	3.555e-06
P1	-1.8013e-05	-2.5667e-05	-2.9771e-05	-4.5542e-05	3.8751e-06	-9.4155e-06
P2	-1.7762e-05	-8.4995e-05	-7.4124e-05	-5.9278e-05	-6.7234e-05	-2.7227e-05

Table 3: The parameters returned by camera calibration for all six takes: focal length (Cp), pixels/mm (kx and ky in x and y directions, respectively), optical center (k0 and y0, in pixel coordinates), radial distortion polynomial coefficients (K1 through K5), and tangential distortion coefficients (P1 and P2).

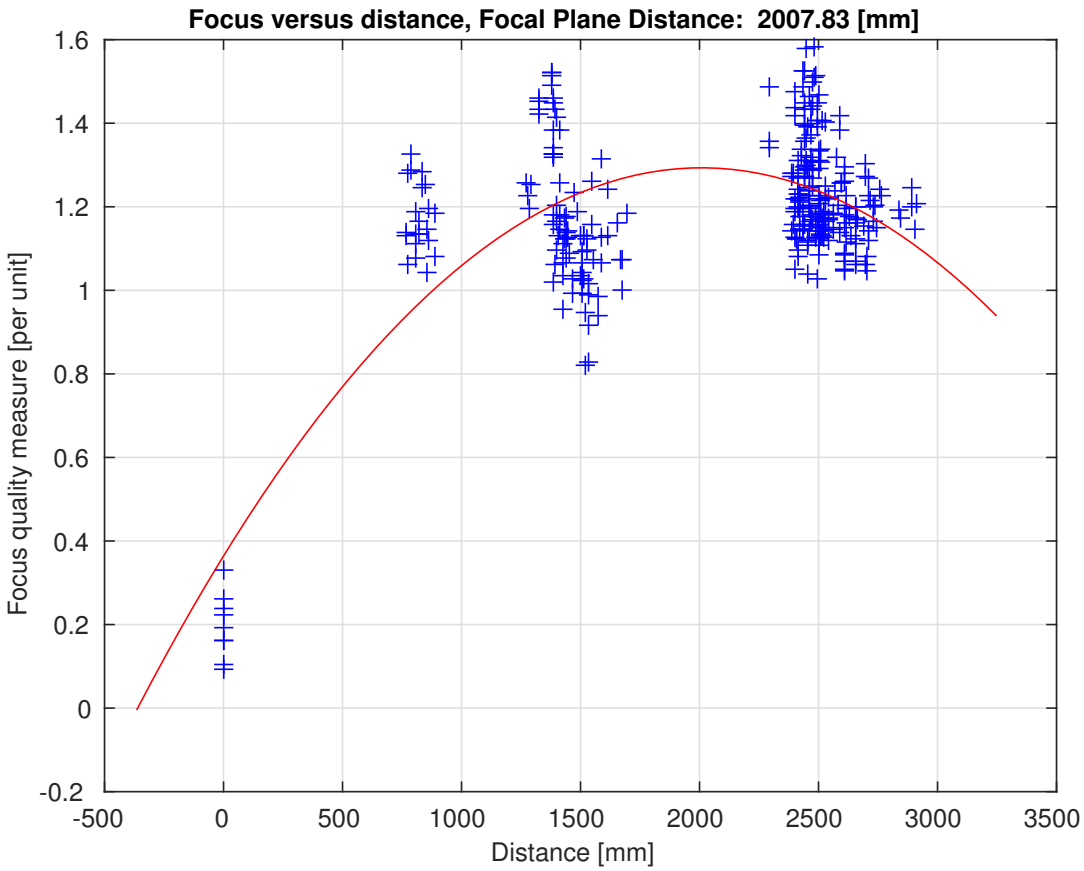

Figure 34: Plot of the Nexus 6 camera focal plane and the calibration tool at Z-Close, Z-Center, and Z-Far.

or five terms. Although K4 and K5 are very small numbers, the fact their term is of a higher order means they have more of a dramatic effect at larger radii. For example, the radial distortion components K1 through K5 of Take 1 can be seen in Figure 35.

The K5 term shown in cyan has very little weight on the distortion until approximately 2000 pixels.

It should be noted that in Table 2, the Nexus 6 does return a 3.82 mm focal length when requested through the Camera2 API, and Cp in Table 3 is also approximately 3.82mm.

Table 4 shows the RMS pixel differences in distortion correction between each of the six takes.

	Take 2	Take 3	Take 4	Take 5	Take 6
Take 1	0.89	8.11	8.38	11.41	17.15
Take 2		11.92	12.27	16.30	23.96
Take 3			0.25	1.04	2.76
Take 4				0.86	2.59
Take 5					1.73

Table 4: The amount of RMS pixel differences after the distortion correction of each camera calibration is applied.

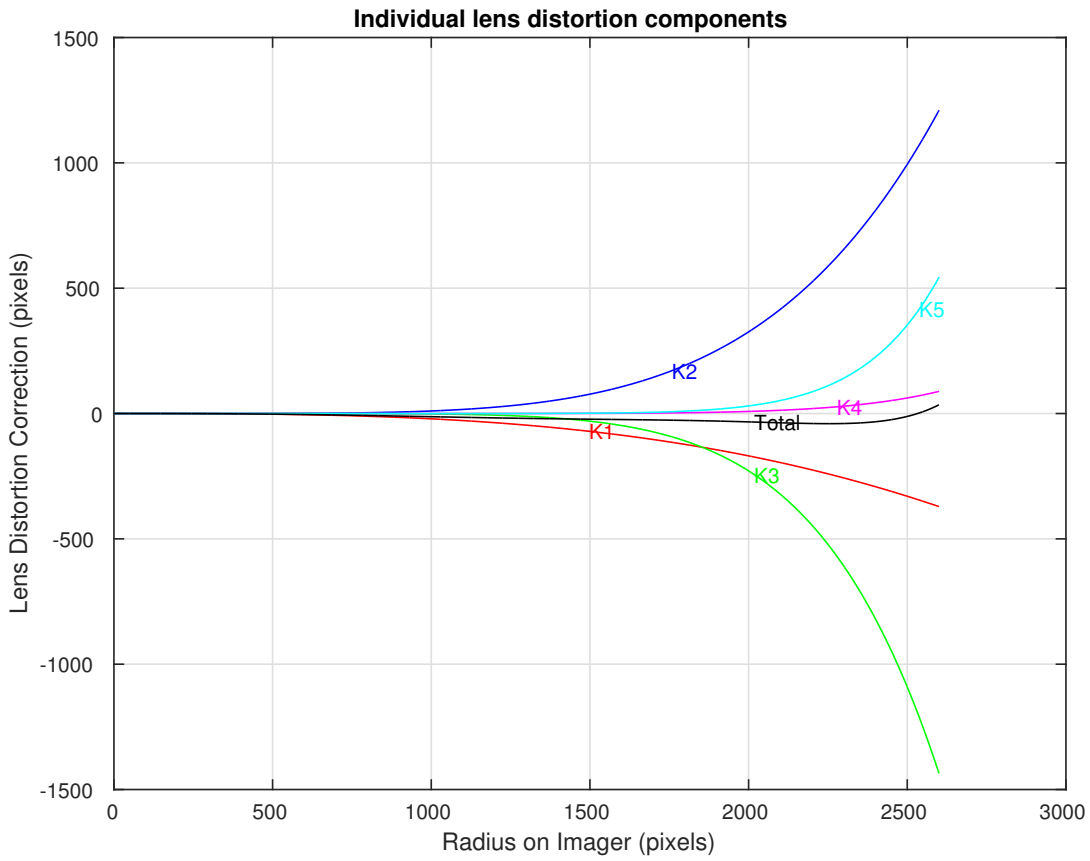

Figure 35: The radial distortion individual component contributions are shown here. Component K5 is not negligible >2000 pixel radius.

The data can be interpreted as the difference between two calibrations that are looking at the same point in space, in RMS pixels.

4.2.6 Camera Calibration Conclusion

The successful camera calibration allows for reliable 6-DOF pose results that can be used for the RULA angles. However, the homogeneous transform returned by MPT requires analyses to apply the transforms to RULA which is discussed in Section 5.

5 Algorithm to Calculate Limb Angles

The raw MPT return values do not directly provide the RULA angles. This section describes the process to calculate the angles from the homogeneous transform.

5.1 Homogeneous Transform

The homogeneous transform matrix is shown in Table 5.

PoseHat2.ctT			
Rotation Matrix R (3x3)			X
			Y
			Z
0	0	0	1

Table 5: The PoseHat2.ctT / Homogeneous Transform definition. The position vector is in the 4th column, while the rotation matrix is the first 3x3.

The values come in the form of the array `PoseHat2.ctT[16]` which is returned by MPT for each motion tracking marker that is analyzed. The information in the transform can be visualized by Figure 36.

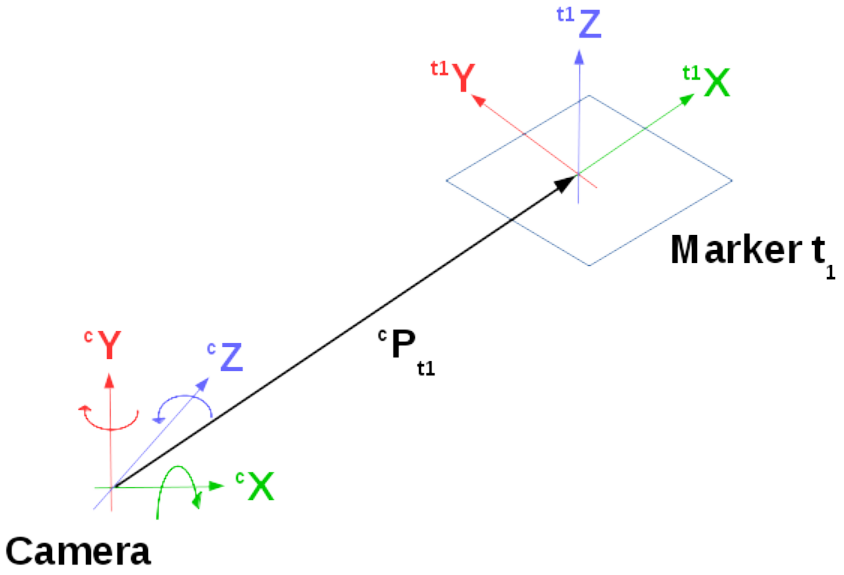

Figure 36: The visualization of the homogeneous transform ${}^{c}_{t_1}T$ returned by MPT. It can be described as the marker t_1 in camera c coordinates.

Three of the six degrees of freedom are the ${}^{c}X$, ${}^{c}Y$, and ${}^{c}Z$ coordinates, that is, the X, Y, and Z linear distances to the marker center in the camera coordinate frame. This is referred to as the *position vector*. The other three of the 6-DOF data are the rotations about the ${}^{c}X$, ${}^{c}Y$, and ${}^{c}Z$ axes, or the pitch, yaw, and roll angles, respectively. All six values are within the 4 x 4 matrix described in Table 5

called a *Homogeneous Transform*. The position vector is explicit in the homogeneous transform (Column 4, Rows 1 through 3). However, the pitch, roll, and yaw angles are not explicitly included in the homogeneous transform and require a mathematical decomposition of rotation matrix R to obtain each one individually (See Section 5.1.2). The homogeneous transform has the mathematical notation ${}^{c}_{t}T$, and describes the pose of motion tracking marker t in camera coordinates c.

The homogeneous transform can use simple linear algebra equations to determine other characteristics. For example, the pose of the camera in target coordinates is given as

$$ {}^{t}_{c}T = ({}^{c}_{t}T)^{-1} \tag{3} $$

Conceptually, Equation 3 states the pose of the camera in the marker frame is equal to the the *inverse* of the pose of the marker in the camera frame.

5.1.1 Transforming the Coordinate Frame

Taking this concept a step further, consider an image that has two markers, t_1 and t_2 as shown in Figure 37.

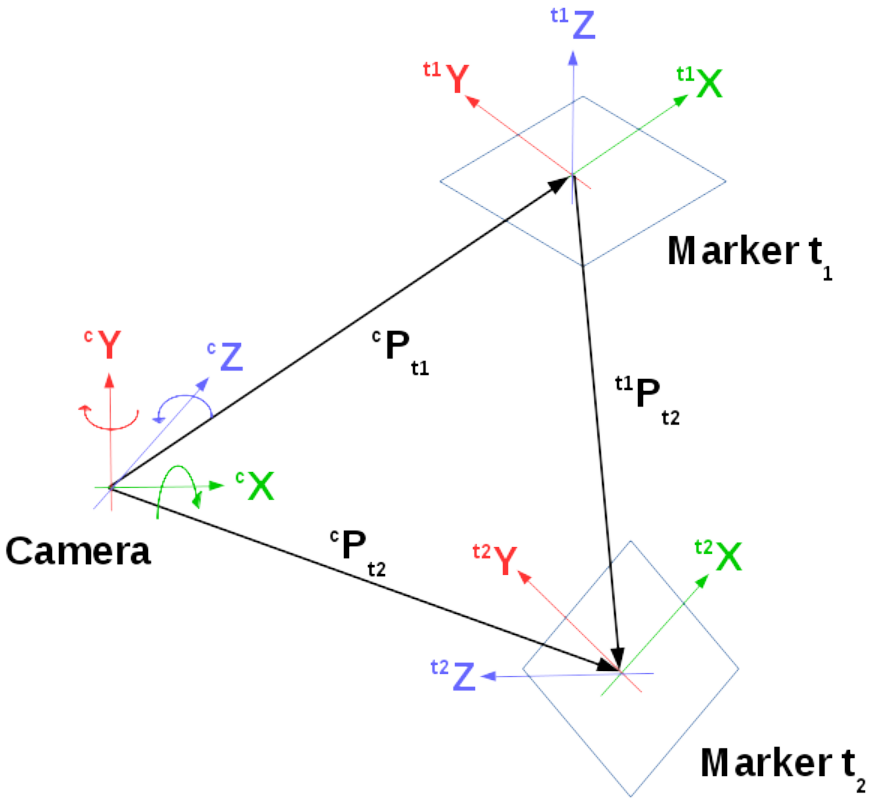

Figure 37: Using linear algebra, ${}^{t_1}_{t_2}T$ can be found from ${}^{c}_{t_1}T$ and ${}^{c}_{t_2}T$ as shown in Equation 4.

MPT will return both ${}^{c}_{t_1}T$ and ${}^{c}_{t_2}T$. Then, the homogeneous transform of marker t_2 in the t_1 coordinate

frame is calculated by:

$$ {}^{t_1}_{t_2}T = \left({}^{c}_{t_1}T \right)^{-1} {}^{c}_{t_2}T \tag{4} $$

The transformation shown in Equation 4 describes the center of t_2 located at ${}^{t_1}X$, ${}^{t_1}Y$, and ${}^{t_1}Z$ (or ${}^{t_1}P_{t_2}$ in Figure 37) and the pitch, roll, and yaw angles about the axes in the marker t_1 coordinate frame. A coordinate frame aligned in marker coordinates is shown in Figure 38.

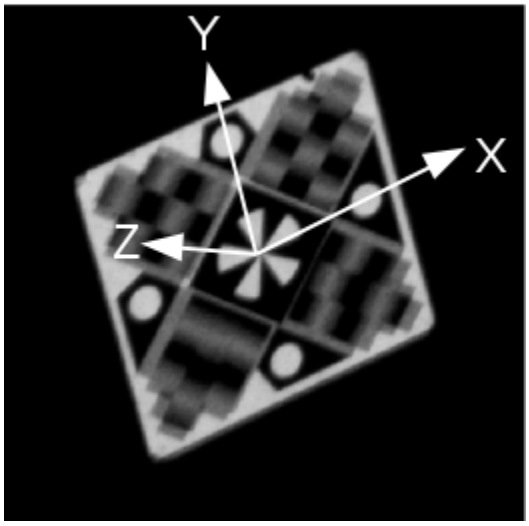

Figure 38: The surface of the marker is the XY plane while the Z-axis is protruding through the center, orthogonal to both the X-Axis and Y-Axis.

Notice the Z-Axis protrudes from the center of the starburst landmark.

5.1.2 Cardan Angle Decomposition

To find the Cardan angles about each axes, the following equations are used on the 3 x 3 rotation matrix R of the homogeneous transform:

$$ \begin{aligned} E_x &= \arcsin(R(2,3)) \\ E_y &= \arctan 2(-R(1,3),\ R(3,3)) \\ E_z &= \arctan 2(-R(2,1),\ R(2,2)) \end{aligned} \tag{5} $$

5.2 Apply Homogeneous Transform to RULA

The MPT system intrinsically measures the motion of markers in camera coordinates. Using this data, the motion between markers can be determined according to Equation 4. This is not sufficient to extract anatomically correct motion. In order to take motions in marker coordinates and access them as motions

measured in anatomically relevant coordinates, it's necessary to rotate each segment's marker coordinate frame into an anatomically-aligned body coordinate frame. The data determines the rotation from marker coordinates *into* body coordinates is obtained by taking an image collection in neutral pose.

5.2.1 Neutral Pose Normalization

When the subject wears the markers, all marker coordinate frames are based on the plane to which they are attached, as seen in Figure 39.

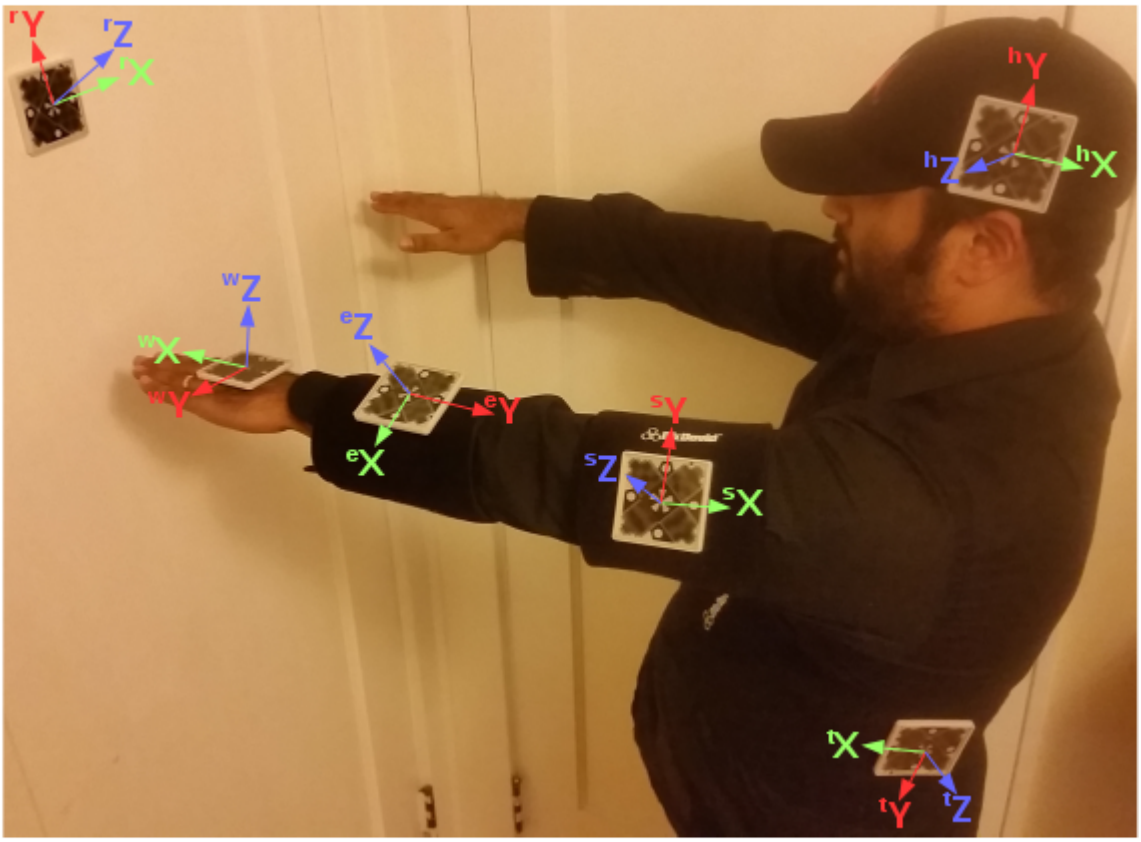

Figure 39: The markers' coordinates frames are akimbo and need to be aligned with anatomical coordinates.

The coordinate frame of each marker has no physical meaning since it is rotated arbitrarily in 3D space. That is, if a segment is rotated about an arbitrary marker axis, it has no physical meaning. Each marker's rotation is only relevant if it is with respect to *anatomical coordinates*. By placing a marker on the wall with known coordinates (Y-Axis aligned with gravity, Z-Axis protruding through marker center, and X-Axis orthogonal to both which can be seen as ${}^{r}Y$, ${}^{r}Z$, and ${}^{r}X$, respectively, in Figure 39) and having the subject squarely face the marker, anatomical coordinates are *aligned*. This establishes the body coordinates frame.

To achieve this end, each marker in the neutral pose images is rotated into room coordinates by Equa-

tion 4, where t_1 is the room marker and t_j iterates through all limb markers. The subject must be facing the wall marker to align the anatomical axes. The resulting homogeneous transform's position vector is set to [0, 0, 0]. This aligns all segment targets' coordinate frame to the room coordinates but keeps their origin position in free space. Now, for example, shoulder abduction is strictly a rotation about the ${}^{b_n}Y$-Axis, and shoulder flexion is strictly about the ${}^{b_n}X$-Axis. The resulting homogeneous transforms are ${}^{b_n}_{t_n}T$, where t_n is the segment marker n in its own coordinate frame and b_n is the segment marker in anatomical coordinates. These rotations can be seen in Figure 40.

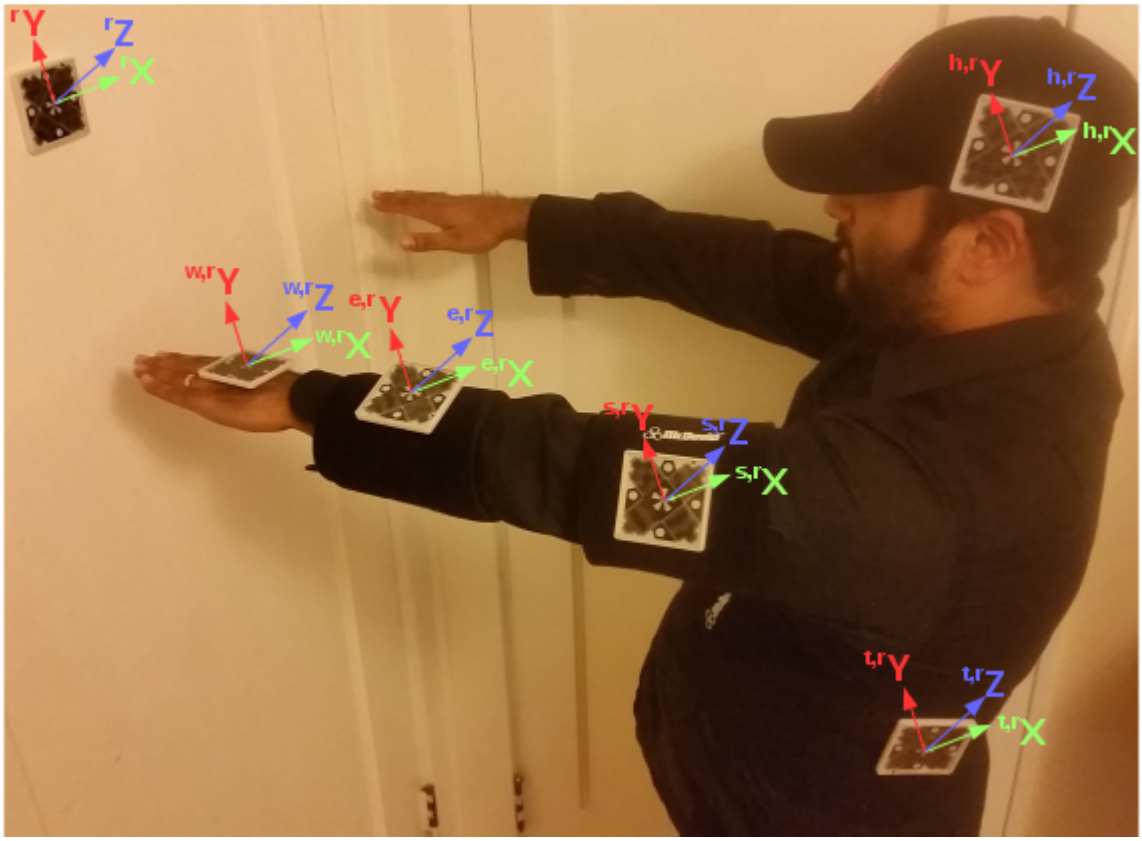

Figure 40: When the subject is square to the wall, anatomical coordinates can be established (X-Axis through the shoulders, Y-Axis through the head, Z-Axis out of the chest) using the room marker t_r. Notice all body marker coordinate frames are aligned with t_r but keep their same origin.

Note that all marker coordinate frames are at their own origin but rotated to be aligned with the room marker coordinate frame.

5.2.2 Motion Comparison

For every marker pair in every image of a motion under RULA analysis, the following equation is applied:

$$ {}^{b_n}_{b_j}T \;=\; {}^{b_n}_{t_n}T \;\left({}^{c}_{t_n}T\right)^{-1} \;{}^{c}_{t_j}T \left({}^{b_j}_{t_j}T\right)^{-1} \tag{6} $$

Equation 6 finds ${}^{t_n}_{t_j}T$ in the motion image but then *rotates* it via the homogeneous transforms ${}^{b_n}_{t_n}T$ and ${}^{b_j}_{t_j}T$ which aligns the marker coordinate frames of $t_n$and t_j, respectively, with the body coordinates.

5.2.3 Extracting the Desired Homogeneous Transforms

Section 1.2.1 describes the rotation each segment's is measured with respect to. Each segment under analysis has its own marker along with the marker it rotates about. These relationships with the reference designators are seen in Table 6, where r is room, t is torso, u is upper arm, l is lower arm, w is wrist, and h is head.

Segment	Designation	Description
Torso	${}^{b_r}_{b_t}T$	Room Coordinates, Y Axis aligned with gravity
Upper Arm	${}^{b_t}_{b_u}T$	Upper arm rotation with respect to torso
Lower Arm	${}^{b_u}_{b_l}T$	Lower arm rotation with respect to torso
Wrist	${}^{b_l}_{b_w}T$	Wrist rotation with respect to lower arm
Head	${}^{b_t}_{b_h}T$	Head rotation with respect to torso

Table 6: These are the final homogeneous transforms required to calculate RULA - all in body (anatomical) coordinates.

Each homogeneous transform in Table 6 is calculated by Equation 6. The physical representation of each segment and its reference frame is shown in Figure 41.

The arrow head of the line points to the reference coordinate in which the moving segment rotates. For example, the head rotation is with respect to the torso pose in anatomical coordinates.

5.2.4 RULA Neutral Pose Chaining

It is generally not possible to get a single image that includes *all* of the markers. As such, a process described as *chaining* allows for relationships to be built in the neutral pose between the markers. For example, if the lower arm marker t_l is not in an image with room marker t_r but *is* with the upper arm marker t_u, the analysis will align t_l through t_u if t_u exists in another neutral stance image with t_r. The assumption is the subject has no voluntary motion and minimal involuntary motion during the multiple image captures while in neutral stance.

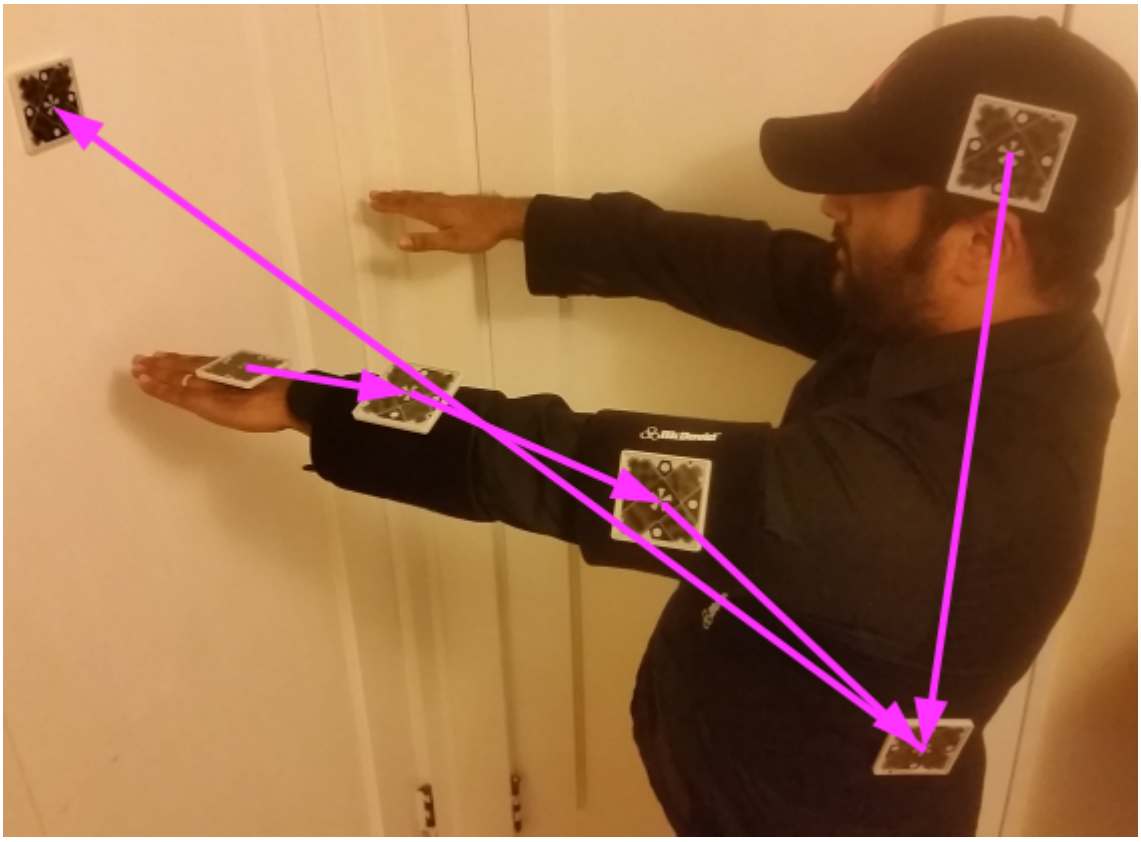

Figure 41: For RULA, each segment marker's angular rotation must be measured with respect to a different marker's coordinate frame. This figure visualizes the marker pairs used for RULA.

5.3 RULA Calculation Conclusion

Once each homogeneous transform in Table 6 is calculated, it simply needs to be extracted for the individual Cardan angles (Section 5.1.2) for RULA.

6 RULA Study

IRB #16.137 approval was received to perform this study.

6.1 Materials and Method

The RULA study requires a total of six motion tracking markers for the transforms shown in Table 6: t_r room marker, t_t torso marker, t_u upper arm marker, t_l lower arm marker, t_w wrist marker, and t_h head marker. Room marker t_r is placed such that it is in camera view and the Y-Axis is aligned with gravity.

6.1.1 RULA Setup

To carry out the RULA procedure, each subject is required to perform the following steps:

1. Put on the motion tracking markers at the anatomical locations as shown in Figure 41.
2. Stand at neutral pose (Figure 42) such that a reference frame can be established as described in Section 5.2.1, square with t_r. Images are collected.
3. The subject performs various elementary motions such as shoulder flexion and abduction during image collection. This is to verify the resulting data makes sense.
4. The subject uses the pipe wrench on a disconnected gas meter at an extended arm pose, known as Movement 1 for image collection (see Figure 43).
5. Each subject moves the nut with the pipe wrench to an approximate middle pose, known as Movement 2 for image collection (see Figure 44).
6. Each subject moves the nut with the pipe wrench to a close pose, known as Movement 3 for image collection (see Figure 45).
7. Steps 2, 4, 5, and 6 are completed two more times at the current gas meter nut height.
8. Steps 2, 4, 5, and 6 are completed three times total at at the other gas meter nut height (either 36" or 72" - whichever has not yet been completed).

The 36" gas meter nut height is shown in Figure 46.

Each subject completes 6 sets of data with one of those sets containing the elementary motions: three at a 36" gas meter nut height, and three at a 72" gas meter nut height. Each set consists of the three motions (Motion 1, Motion 2, and Motion 3). Thus, each subject has a total of 6 takes and 18 motions.

Figure 42: Subject #2 at neutral pose stance. There is a 90° shoulder flexion in neutral stance because it provides an aligned pose that is repeatable across subjects.

Figure 43: Motion 1 on a disconnected gas meter with a nut height of 72". The arms are extended.

6.1.2 RULA Study

Three subjects are chosen for the study giving 18 total sets and 54 total motions.

Figure 44: Motion 2 on a disconnected gas meter with a nut height of 72”. The arms are slightly pulled in.

Figure 45: Motion 3 on a disconnected gas meter with a nut height of 72”. The arms are fully pulled in.

These assumptions are made for image collection:

- The wall target’s Y-Axis is precisely aligned with gravity
- The subjects hold each pose steadily

Figure 46: Subject #1 on a disconnected gas meter with a nut height of 36".

- The subjects are square with wall marker t_r during neutral pose data collection

A neutral stance of 90 degree shoulder flexion is chosen due to varying natural abduction / flexion between subjects when they are standing normally with their arms at side. This requires an addition of 90 degrees on upper arm about X-Axis rotation while scoring RULA from the returned Cardan angles.

An example of a Nexus 6 image taken during the study is shown in Figure 47.

Note that the image is far too dark to see anything else in the field of view besides the markers due to the short exposure.

6.1.3 RULA Pilot Testing

A pilot study was performed prior to the IRB approved study and processed 11 out of 12 poses successfully on the newest MPT software build.

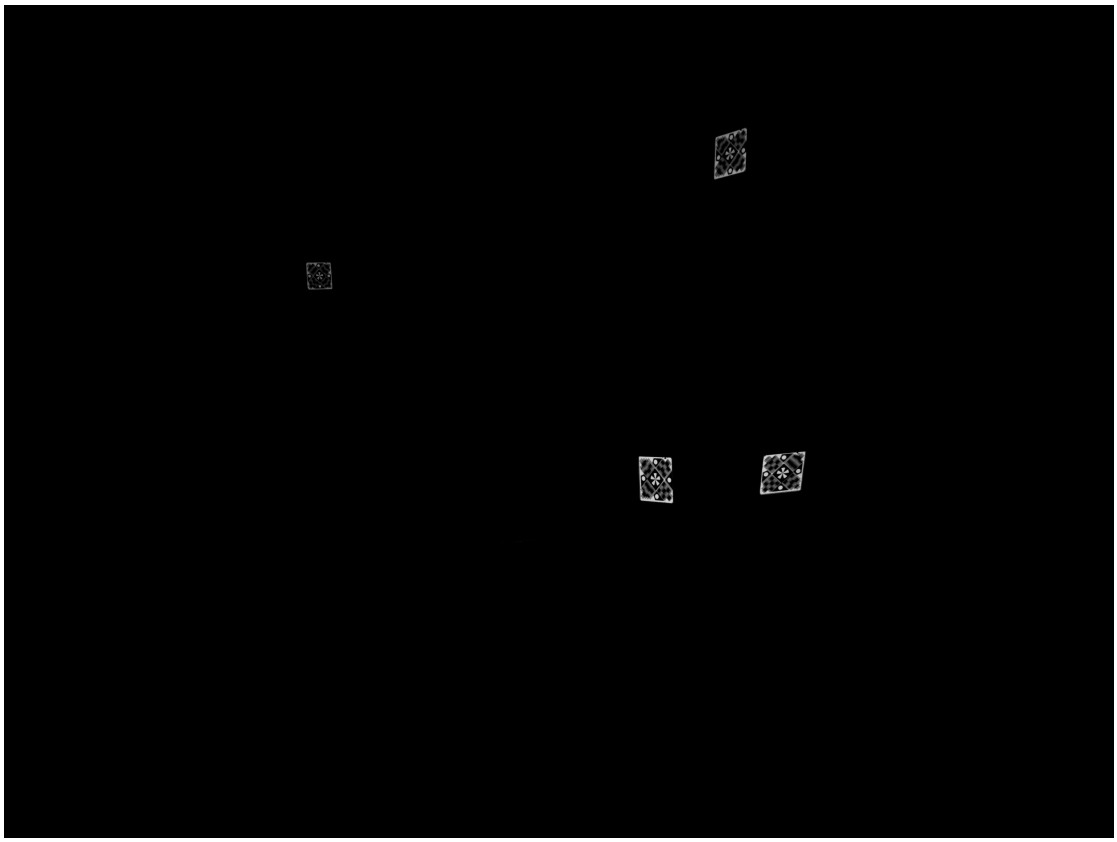

Figure 47: An actual Nexus 6 Image in RULA Study (Subject #2, Low Meter, Neutral-At-Side Pose). MPT markers incorporate retroreflective material, making it possible to capture MPT images with very short exposure. When MPT App captures an image, it is too dark to see the subject's pose.

6.1.4 RULA Data Collection and Mobility

A goal of this thesis is to demonstrate the mobility of MPT on an Android device. As stated in Section 1.1, many of the commercial 3D motion tracking systems require complicated setups and multiple cameras. This mobility can be seen in the RULA data collection images shown in Figure 48.

Figure 48 shows that there are no complicated set ups or several tripods with expensive and specialized cameras on them. The image collector has movement to move around, as space allows, and can take images from several angles.

All RULA angles are calculated on the Nexus 6. This includes the homogeneous transforms from MPT and the subsequent RULA analysis (described in Section 5) calculated using Octave on the Nexus 6.

Figure 48: Images of RULA image collection in process. There is significant mobility with little set up and no camera calibration required in the field.

6.2 RULA Results

6.2.1 RULA Data Collection Efficiency

One RULA was calculated with the Nexus 6 out of 18 sets total: Subject #1 at low (36") meter nut height (Figure 46), second set, motion 1. The low count of RULA analyses is attributed to:

1. Insufficient wrist markers in the neutral pose position. The images were not taken from the correct angle (not anterior enough) to effectively detect the wrist marker. A stricter protocol and real-time processing would all be effective methods in obtaining the wrist marker. Real-time processing would alert the operator if there are not enough markers in the collection of images.

2. The outdated build of the MPT software on the Nexus 6 is less robust at detecting markers and

calculating 6-DOF pose data. An update to the source could would remedy this issue. However, the current software is sufficient to achieve demonstration, as a software upgrade would require an extensive effort.

6.2.2 RULA Results

The Cardan angles for Subject #1, low meter nut height, second set, motion 1 are calculated and shown in Table 7.

	Axis of Rotation		
Segment Rotation	X-Axis	Y-Axis	Z-Axis
Upper Arm Rotation $({}^{b_t}_{b_u}T)$	**-41.7014°***	**-13.9910°**	-0.6597°
Lower Arm Rotation $({}^{b_u}_{b_l}T)$	**0.9475°**	-8.7144°	-1.4733°
Wrist Rotation $({}^{b_e}_{b_w}T)$	**10.1340°**	**3.0979°**	**-9.8731°**
Head Rotation $({}^{b_t}_{b_h}T)$	**-18.3969°**	-1.9872°	-2.2952°
Torso Rotation $({}^{b_r}_{b_t}T)$	**-4.79°**	90.4606°	-1.9245°

*Note: Due to neutral pose having a 90° shoulder flexion (Figure 39), 90° is added to X-Axis upper arm rotation in RULA worksheet.

Table 7: Cardan angles of the homogeneous transforms in body coordinates with the numbers in bold used in the RULA.

The values shown in **bold** are used for RULA. Additionally, the lower arm rotation is given by the angulation between the Y-Axis of torso body coordinates and the Z-axis of lower arm body coordinates given by:

$$\theta_l = \cos^{-1}\left(\left\langle {}^{u}\bar{Z},\ {}^{t}\bar{Y}\right\rangle\right) \tag{7}$$

Applying the worksheet in Figure 4, Table 8 uses the angles to calculate the RULA score with Step 2 calculated from Equation 7:

The final score of 6 indicates the motion should be investigated and changes to the motion are required soon.

6.2.3 Comparing RULA Analyses

Three observed RULAs are completed by trained personnel with each giving a total RULA score of four for the same motion - less than the 6 calculated by MPT. The angles observed, compared with the MPT calculation, are shown in Table 9. Some of the steps in RULA do not use an actual angular range but MPT angles are used to determine that step's score in Table 8. For example, Step 2a requires a +1 if the arm is working across midline of the body - which can be seen as a Y-Axis upper arm rotation. These are not included in Table 9 as the angle estimated by the RULA observer is unknown. All non-angular variables are relatively consistent between all analyses.

Section	Step	Step Description	Value	Angle Score	Step Score	Table Score
A	Step 1	Locate Upper Arm Position	-41.7° + 90° = 48.3°	+3	3	4
	Step 2	Locate Lower Arm Position	50.7°	+2	3	
	Step 2a	Location of Arm	-14.0°	+1		
	Step 3	Wrist (X-Axis Rotation)	10.1° Downwards	+2	3	
		Wrist Midline (Y-Axis Rotation)	3.1°	+1		
	Step 4	Wrist Twist	-9.8°	+1	1	
	Step 5	Table Score from Section A			4	
	Step 6	Add Muscle Use Score	Action repeated 4 / min	+1	1	
	Step 7	Add Force / Load Score	2kg to 10kg (inter.)	+1	1	
	Step 8	Add Step 5, Step 6, Step 7			6	
B	Step 9	Neck Position	18.4° (Flexion)	+2	2	2
	Step 10	Trunk Position	4.8° (Flexion)	+1	1	
	Step 11	Supported / Balanced Legs	Yes	+1	1	
	Step 12	Table Score from Section B			2	
	Step 13	Add Muscle Use Score	Action repeated 4 / min	+1	1	
	Step 14	Add Force/Load Score	2 kg to 10kg (inter.)	+1	1	
	Step 15	Find Column in Table C			4	
		Final Score			**6**	

Table 8: Actual RULA results of Subject #1, low gas meter nut height, Second Set, Motion 1. The final score of a 6 indicates there should be an investigation and changes to the motion soon.

Angle Under Observation	MPT Angle	Observer 1 Angle	Observer 2 Angle	Observer 3 Angle
Upper Arm (Shoulder) Rotation	48.3°	15 - 45°	15 - 45°	15 - 45°
Lower Arm (Elbow) Angulation to Torso	50.7°	60 - 100°	60 - 100°	60 - 100°
Wrist Rotation	10.1° Downwards	0° to 15° Downwards	0° to 15° Downwards	0° to 15° Upwards
Neck Rotation	18.4° Flexion	10° to 20° Flexion	0° to 10° Flexion	0° to 10° Flexion
Torso Rotation	4.8° Forward	0° - 20° Forward	0° (Erect)	0° (Erect)

Table 9: RULA angle comparison with three observed RULAs on the same motion. All of the angles are comparable.

All of the angles are reasonably similar. Most angles are within 15° of each other.

6.3 RULA Data Results Conclusion

The RULA angles calculated by MPT for certain steps are quantitative as opposed to observational. For example, the Torso Rotation in Table 9 has a small angle that is normally rounded to 0° by two of the observers performing RULA.

Although the MPT RULA score is higher than the observed RULA scores, the majority of angles between the RULA results are comparable which is relevant to this thesis.

The pilot study results show a more experienced operator and a newer MPT build produces a high level of success.

7 Main Lessons Learned

A successful RULA measurement is captured in this thesis via mobile 3D motion tracking. It can be seen in Figure 48 that since the data capturing source can be mobile, this 3D motion tracking technology is suited for field use and does *not* need significant control of its environment. A framework for 3D motion tracking on a Nexus device has been constructed and its application demonstrated.

7.1 Nexus Camera

Camera calibration efforts with several mobile devices resulted in the knowledge of camera control requirements for MPT. Manual focus and exposure control is absolutely necessary for MPT and ideally the settings are repeatable with a very high accuracy.

An image request is sent to the camera controller with post-processing turned off and a callback is executed when image data becomes available. The Y channel, or luma channel, of the YUV data is passed across the JNI within `SaveBMPWrapper()`. This saves the pixel data into a format that is MPT friendly (gray scale uncompressed Bitmap).

7.2 Processing Effort

When a bitmap is retrieved and the pixels are loaded into memory in the Java domain from the flash memory, the pixel data is transferred across the JNI through `RGRWrapper()` where the main MPT functions are called.

MPT requires a substantial amount of memory and processing effort by the device. For the current MPT code on the Nexus 6, which has not been optimized for the 32-bit multicore platform, a single image of 13 megapixels by the Nexus 6 takes approximately 45 seconds to process and uses 238MB of heap memory. Although mobile devices are advancing, it is still a non-negligible load on the mobile device's resources. The standard `pthread` stack size is insufficient and an increase in the stack is required for the process.

Care must also be taken in declaring variables. For example, the `long` data type disparities on an Intel versus an ARM processor caused errors during the porting of MPT to the Nexus devices.

MATLAB coder builds MPT into C++ and thus the Native Development Kit (NDK) is required although the standard Android API is in Java (SDK). This is not necessarily a problem as the architecture of C++ is better suited for intense processing than Java (direct memory management, for example). This does, however, add for complexity as a scheme for the Java-Native Interface must be designed.

7.3 Camera Calibration

Control over the camera allows for an appropriate setting for focus distance required for RULA. Exposure control reduces motion blur and marker saturation. The Nexus 10 motion blur during the calibration attempts were visible to the eye, but increasing the exposure speed with the Camera2 API in the Nexus 6 eliminated the motion blur.

The radial distortion polynomial had non-negligible high-order components due to the wide angle lens. The quality of calibration increased when the higher order components were added to the radial distortion compensation.

7.3.1 Rolling Shutter Effects

Focus and exposure control alone is still insufficient to perform standard calibration - that is, calibrate from a moving calibration tool. The rolling shutter prevents consistent calibration results from the Nexus devices. Thus, a method to keep the calibration tool *and* Nexus device stable eliminates rolling shutter distortion artifacts allowing for consistent calibrations.

7.4 Application to RULA

Field data collection for RULA has been demonstrated. The location of data collection did not require major modifications to the environment and subjects were not required to remove jewelry or any other reflective objects. A stricter protocol and real time processing will certainly improve data collection efficiency. The pilot study showed that an experienced operator and new build of software will also improve data collection effectiveness.

7.5 Thesis Impact

No optical 3D system that is well adapted to field data collections exists today. So taking a robust 3D motion tracking system (MPT) that inherently needs little control of its environment which is then ported to a mobile device demonstrating a quick and easy 3D measurement proves its worth. Future research of this technology will certainly improve accuracy and speed of the system.

7.6 Future Investigations

Turning a mobile device into a 3D motion tracking system has unearthed areas where improvements of performance of MPT on the Nexus device can be achieved.

7.6.1 RAW Data Format and Post-Processing

Android devices perform behind-the-scene image post-processing by default but nearly full control over the image pipeline is possible. Similar to requesting the YUV image type for a `CaptureRequest`, the RAW data format can also be requested which is the raw image sensor data, and is a Bayer pattern encoded image. This data is free of all behind-the-scenes signal processing. An assumption in this thesis is that all post-processing is turned off (see Section 3.2.2), but due to limited information, this is not guaranteed.

Performing MPT processing in the RAW data format can also pose a challenge. First and foremost, the raw image is *color* data - where MPT analyzes gray scale.

7.6.2 Rolling Shutter Compensation

As discussed in [21], rolling shutter compensation methods do exist. Care had to be taken during camera calibration that the Nexus device and the calibration tool were motionless. If a method is devised to compensate the rolling shutter effects, the system would increase robustness and require less control of the environment during field measurements.

7.6.3 Mobile Device Resources Scheme

A goal of this thesis is to demonstrate 3D motion tracking on a mobile device. MPT is implemented such that it works on a mobile device but it has not been optimized specifically for the Nexus 6.

A more intimate knowledge of the Nexus hardware will allow an optimized processing scheme to be developed. This would allow real-time processing to occur and MPT App would not be required to finish image collection before image processing begins. For example, real time processing would allow MPT App to alert the operator if insufficient markers are detected during a RULA analysis.

7.6.4 K1 through K5 Image Processing

Camera calibration returned radial distortion all the way to the fifth term in the polynomial in Equation 1 due to the wide-angle nature of the lens. However, the MPT build for the Nexus 6 uses only K1, K2, and K3, or three terms, causing inaccuracies at the edge of the imager (greater than about 2000 pixels from the image center).

In a future MPT build for a mobile device, MPT with radial distortion compensation capabilities out to K5 will provide more accurate 6-DOF pose calculations.

8 Conclusion

3D motion tracking on a mobile device in the field has been demonstrated. The technology is significant given the other 3D motion tracking systems that exist today. This lays the foundation for future research to improve upon accuracy, speed, and robustness of MPT on a mobile device. Much of the ground work has been established, with the basic challenges solved in this thesis such as processing, camera control, and camera calibration. Tracking motion in 3D space on a device that can fit in a pocket truly is a paradigm shift.

References

[1] Brian Armstrong, Thomas Verron, Lee Heppe, Jim Reynolds, and Karl Schmidt. RGR-3D: Simple, cheap detection of 6-DOF pose for tele-operation, and robot programming and calibration. In *Proc. 2002 Int. Conf. on Robotics and Automation*, pages 2938–2943. IEEE: Washington, 2002.

[2] Kristian M O'Connor, Brian S. R. Armstrong, Joshua Weinhandl, Todd P. Kusik, and Robb T. Barrows. Validation of a single camera 3D motion tracking system. In *Proc. Annual Meeting of the American Society of Biomechanics*, page 980. ABS: State College, Penn., Aug 2009.

[3] Julian Maclaren, Brian S. R. Armstrong, Robert T. Barrows, K. A. Danishad, Thomas Ernst, Colin L. Foster, Kazim Gumus, Michael Herbst, Ilja Y. Kadashevich, Todd P. Kusik, Qiaotian Li, Cris Lovell-Smith, Thomas Prieto, Peter Schulze, Oliver Speck, Daniel Stucht, and Maxim Zaitsev. Measurement and correction of microscopic head motion during magnetic resonance imaging of the brain. *PLoS ONE*, 7(11):e48088, 11 2012.

[4] Brian Armstrong, Michael Botum, Mustafa Farrah, Kristian O'Connor, and Stephen Watts. An innovative diagnostic tool for reducing traumatic knee injuries. In *Proc. Annual Meeting of the American Society of Biomechanics*. ABS: Stanford, CA, aug 2007. Poster Session P9-9.

[5] Lynn McAtamney and E Nigel Corlett. RULA: a survey method for the investigation of work-related upper limb disorders. *Applied Ergonomics*, 24(2):91–99, 1993.

[6] Neville Stanton, Alan Hedge, Karel Brookhuis, Eduardo Salas, and Hal Hendrick, editors. *Handbook of Human Factors and Ergonomic Methods*. CRC Press, 2004.

[7] P. Wintachai and N. Charoenchai. The comparison of ergonomics postures assessment methods in rubber sheet production. In *2012 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 1257–1261. IEEE, 2012.

[8] B. M. Deros, N. K. Khamis, D. Mohamad, N. Kabilmiharbi, and D. D. I. Daruis. Investigation of oil palm harvesters' postures using RULA analysis. In *2014 IEEE Conference on Biomedical Engineering and Sciences*, pages 287–290. IEEE, 2014.

[9] L. H. Ran J. W. Niu, X. W. Zhang. Investigation of ergonomics in automotive assembly line using Jack. In *2012 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 1381–1385. IEEE, 2010.

[10] H. Haggag, M. Hossny, S. Nahavandi, and D. Creighton. Real time ergonomic assessment for assembly operations using kinect. In *2013 UKSim 15th International Conference on Computer Modelling and Simulation*, pages 495–500. IEEE Computer Society, 2013.

[11] Z. Kertesz and I Lovanyl. 3D motion capture methods for pathological and non-pathological human motion analysis. In *Information and Communication Technologies, 2006. ICTTA '06*, pages 1062 – 1067. Damascus, 2006.

[12] S Mihradi, Ferryanto, T Dirgantara, and A.I. Mahyuddin. Development of an optical motion-capture system for 3D gait analysis. In *2011 International Conference on Intsrumentation, Communication, Information Technology and Biomedical Engineering*, pages 301 – 303. Bandung, 2011.

[13] Victor W. Zhou, Andre Z. Kyme, Steven R. Meikle, and Roger R. Fulton. Event-by-event motion compensation for small animal PET. In *2007 IEEE Nuclear Science Symposium Conference Record*, pages 3109 – 3114. Honolulu, 2007.

[14] Heesung Jun Jaeyoung Kim. Implementation of image processing and augmented reality programs for smart mobile device. In *Strategic Technology (IFOST), 2011 6th International Forum on*, pages 1070–1073. IEEE, 2011.

[15] Dianyuan Han and Hui Dong. The study on standing tree measurement based on image processing embedded in a smartphone. In *Image Analysis and Signal Processing (IASP), 2011 International Conference on*, pages 252–256. IEEE, 2011.

[16] U. Kallavus V. Sinivee, L. Kurik. Mobile photogrammetry for positioning measuring sensors. In *Electronics Conference, 2008. BEC 2008. 11th International Biennial Baltic*, pages 227–230. IEEE, 2008.

[17] Yi-Shin Chen Kuei-Chung Chang, Chia-Yang Kao and Guang-Yu Chen. Memory behavior profile for Android applications. In *Consumer Electronics (GCCE), 2014 IEEE 3rd Global Conference on*, pages 634–635. IEEE, 2014.

[18] Zhongshui QU and Jianwei Wang. A color YUV image edge detection method based on histogram equalization transformation. In *2010 Sixth International Conference on Natural Computation (ICNC 2010)*, pages 3546–3549. IEEE, 2010.

[19] Shang-Hong Lai Yen-Hao Chiao, Tung-Ying Lee. Rolling shutter correction for video with large depth of field. In *Image Processing (ICIP), 2013 20th IEEE International Conference on*, pages 900–904. IEEE, 2013.

[20] Shuochen Su and Wolfgang Heidrich. Rolling shutter motion deblurring. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 1529–1537. IEEE, 2015.

[21] Laurent Kneip Roland Siegwart Luc Oth, Paul Furgale. Rolling shutter camera calibration. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1360–1367. IEEE, 2013.

[22] Roger Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf tv cameras and lenses. *Robotics and Automation, IEEE Journal of*, 3(4):323–344, 1987.

[23] D.C. Brown. Close-range camera calibration. *Photogrammetric Engineering*, 37(8):855–866, 1971.

[24] D.C. Brown. Decentering distortion of lenses. *Photogrammetric Engineering*, (3):444–462, 1965.

[25] Abd Manan Samad Ismail Ma'arof Noor Aniqah Mohd Azhar, Annuar Ahmad and Khairil Afendy Hashim. Comparative geometric and radiometric evaluation of mobile phone, compact and dslr cameras. In *Signal Processing and its Applications (CSPA), 2013 IEEE 9th International Colloquium on*, pages 353–358. IEEE, 2013.

[26] Adrian Goral. Sparse 3D reconstruction on a mobile phone with stereo camera for close-range optical tracking. In *Instrumentation and Measurement Technology Conference (I2MTC), 2015 IEEE International*, pages 2032–2037. IEEE, 2015.

[27] Philip Saponaro and Chandra Kambhamettu. Towards auto-calibration of smart phones using orientation sensors. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference on*, pages 20–26. IEEE, 2013.

[28] Qiaotian Li. Enhancement to camera calibration: Representation, robust statistics, and 3D calibration tool. *UWM Theses and Dissertations*, Paper 469, 2014.

[29] Seong-Chan Byun Junhee Park and Byung-Uk Lee. Lens distortion correction using ideal image coordinates. *Consumer Electronics, IEEE Transactions on*, 55(3):987–991, 2009.

[30] Taufiqur Rahman and Nicholas Krouglicof. An efficient camera calibration technique offering robustness and accuracy over a wide range of lens distortion. *Image Processing, IEEE Transactions on*, 21(2):626–637, 2012.