Optical Tracking in Medical Physics

Advanced Lab Course Experiment at the Department of Medical Physics

Index

1. Introduction

Optical tracking systems use infrared light to determine the position of a target via active (transmits its own infrared light) or passive (reflects infrared light supplied by an external illumination source) markers. The most common active targets are infrared (IR) light-emitting diodes (LEDs). Passive targets are generally spheres or disks coated with a highly reflective material. Various detectors can be used to determine the positions of an optical target; however, charged couple device (CCD) cameras are used most often. Each CCD camera provides a 2D image of a scene, as viewed from the specific camera angle. An array of cameras provides a number of different views of the same scene, each from a different perspective. The multiple views of the same scene can be used to reconstruct accurate 3D locations of each object in the scene.

This is how an individual point object is tracked. However, the ability to track the location of a single object by itself is not sufficient to track the location of a target (point) inside the patient. Such a goal can be achieved by correlating the position of external marker to the 3D motion of internal targets, relying on dedicated imaging data. In this way, optical tracking systems can be used to provide stereotactic localization, allowing quasi real-time localization to implement motion management strategies.

An alternative approach is to detect surface information from single or multiple camera systems, relying on infrared light projected by an illuminator. Such an approach has the advantage to avoid the placement of markers, and directly uses the skin surface of the patient for stereotactic localization. The use of such systems is rapidly spreading for medical physics applications in external beam radiotherapy.

2. Principles of optical tracking

Optical tracking provides 3D localization of specific points (markers) as a function of time, relying on synchronized views from at least two points of view (cameras). Optical tracking systems based on IR cameras are the actual standard in medical applications for the following main reasons:

- computer vision techniques provide accurate, real-time and 3D point (marker) localization
- The use of IR light minimizes disturbances due to visible light.

Visibility of the markers to be tracked needs to be ensured in order to achieve proper 3D reconstruction. Different markers can be used, as schematized in Figure 1:

- Active (cabled or wireless) markers, where the marker emits IR light to be localized by multiple cameras
- Passive markers, that reflect the IR light emitted by appropriate illumination LEDs, which operate synchronously with respect to the acquisition frequency of the cameras.

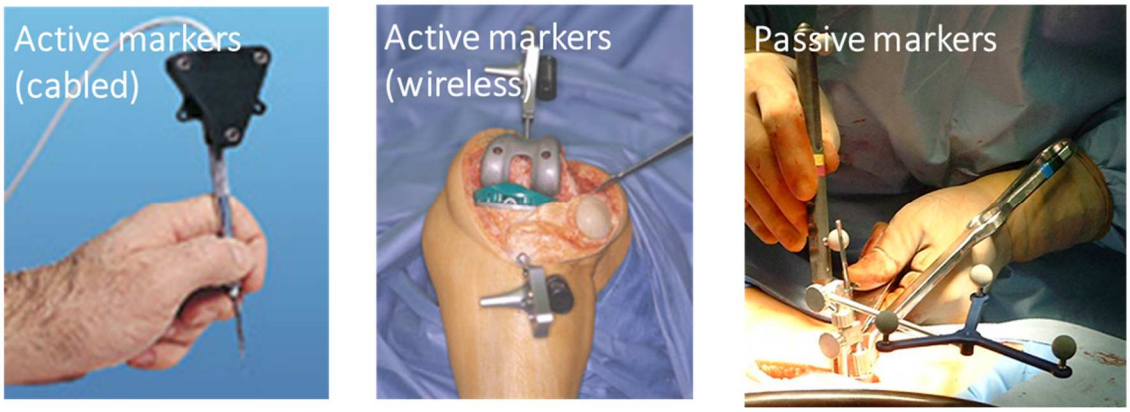

Figure 1. Markers utilized by IR optical tracking systems.

Cameras are objects that map the 3D space (scene) to a set of light-sensitive sensor elements via an optical system, from which a 2D image is derived. To exploit the image content of multiple cameras for reconstructing 3D objects, we need a mathematical model that approximates the physical reality of the projection to a sufficient extent. The geometric model of the projection of points and lines in 3D into the image generated by a real camera has to take into account:

(1) the pose of the camera, i.e., the spatial position and attitude of the camera during exposure
(2) the projection through the optical lenses
(3) the effects of lens distortion
(4) the effects of refraction, especially for large distances between the camera and the scene.

The projection of points in the 3D scene on the camera can be described with a process that entails 4 main steps, as visualized in Figure 2.

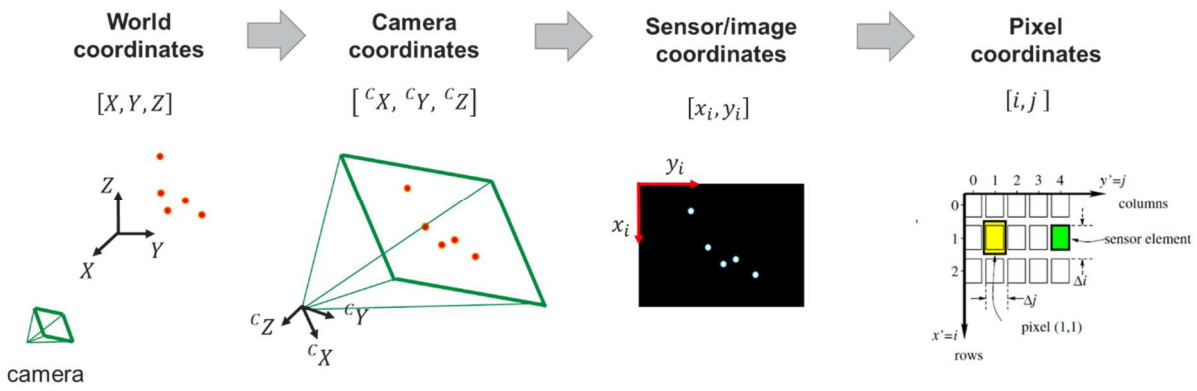

Figure 2. Graphical representation of the workflow required to project 3D points in camera sensor coordinates.

2.1 Camera models

A geometric camera model allows us to invert the projection process by inferring the spatial direction to 3D points and lines from their observed images. We use such directions to determine the spatial position of the camera and the 3D position of the observed points and lines.

The parameters that describe the model of a camera can be distinguished into extrinsic and intrinsic parameters. Extrinsic parameters describe the pose of the camera in 3D space, namely:

- 3 coordinates of the projection centre, or the translation of the camera from the origin to its position during the acquisition ($\boldsymbol{Z} = [X_O, Y_O, Z_O]^T$)
- 3 parameters describing the rotation, e.g., as rotation angles around the three camera axes ($\omega,\ \pi,\ \kappa$). The overall rotation can then be expressed with a $3x3$ rotation matrix $\boldsymbol{R} = \boldsymbol{R}(\omega,\ \pi,\ \kappa)$.

Given $\boldsymbol{X} = [X,\ Y,\ Z]$ the world coordinates of a given point in 3D space, the corresponding coordinates ${}^{C}\boldsymbol{X}$ in the camera reference system can be calculated as follows:

$$ {}^{C}\boldsymbol{X} = \boldsymbol{R}(\boldsymbol{X} - \boldsymbol{Z}) $$

or alternatively, in homogeneous coordinates as:

$$ \begin{bmatrix} {}^{C}\boldsymbol{X} \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & 0 \\ \boldsymbol{0}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{I_3} & -\boldsymbol{Z} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{X} \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & -\boldsymbol{RZ} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{X} \\ 1 \end{bmatrix} $$

Conversely, the intrinsic parameters are all parameters necessary to model the geometry and the physics of the camera in order to be able to infer the direction of the projection ray towards an object point given an image point and the exterior (i.e. extrinsic) orientation. This part of the model can be implemented at various complexities, thus providing different camera models, as depicted in Figure 3.

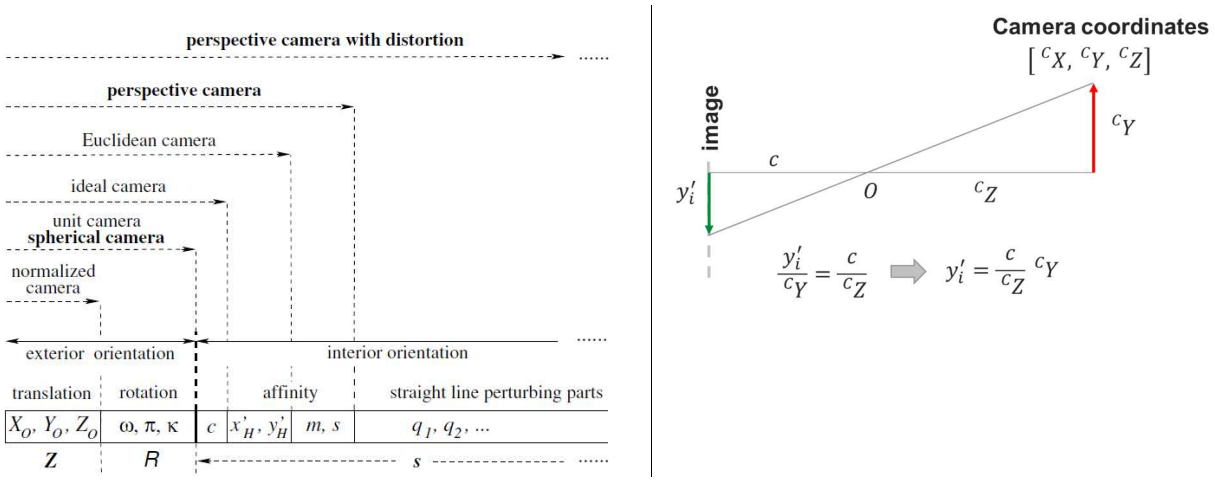

Figure 3. Overview of different camera models (left panel) and projection in an ideal perspective camera (right panel).

When modeling the interior orientation, the following parameters shall be considered:

- c is the focal distance of the camera
- $[x'_H, y'_H]$ are the coordinates of the principal point $\mathcal{H}$(point in the image plane closest to the projection centre of the camera lenses)
- s and m are the shear or scale difference in x' and y' (i.e. the pixel coordinates) with respect to the physical size
- q_i describe the image errors which perturb straight lines, especially due to lens distortion.

When considering an ideal perspective camera (i.e. the only effect of the focal distance c), the projection of the coordinates can be written as (Figure 3, right panel):

$$\frac{y'_i}{{}^cY} = \frac{c}{{}^cZ} \Rightarrow y'_i = \frac{c}{{}^cZ}\,{}^cY$$

We can represent the projection image coordinates $[x_i, y_i]$ with 3 elements $[x'_i, y'_i, z'_i]$ under the assumption that:

$$x_i = \frac{x'_i}{z'_i} \quad y_i = \frac{y'_i}{z'_i}$$

We then have in matrix form:

$$\begin{bmatrix} x'_i \\ y'_i \\ z'_i \end{bmatrix} = \begin{bmatrix} c & 0 & 0 & 0 \\ 0 & c & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} {}^cX \\ {}^cY \\ {}^cZ \\ 1 \end{bmatrix}$$

Remembering that ${}^{C}\boldsymbol{X} = \boldsymbol{R}(\boldsymbol{X} - \boldsymbol{Z})$, we obtain:

$$\begin{bmatrix} {}^{c}X \\ {}^{c}Y \\ {}^{c}Z \end{bmatrix} = \boldsymbol{R}\left(\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} - \begin{bmatrix} X_O \\ Y_O \\ Z_O \end{bmatrix}\right)$$

We therefore obtain the so-called collinearity equations, by calculating the matrix product explicitly and normalizing the coordinates ($x_i = x_i'/z_i'$ and $y_i = y_i'/z_i'$) as:

$$x_i = c\frac{r_{11}(X - X_o) + r_{12}(Y - Y_o) + r_{13}(Z - Z_o)}{r_{31}(X - X_o) + r_{32}(Y - Y_o) + r_{33}(Z - Z_o)}$$

$$y_i = c\frac{r_{21}(X - X_o) + r_{22}(Y - Y_o) + r_{23}(Z - Z_o)}{r_{31}(X - X_o) + r_{32}(Y - Y_o) + r_{33}(Z - Z_o)}$$

where r_{ij} are the elements of the rotation matrix $\boldsymbol{R}$.

The model for a distortion-free perspective camera can instead be written as:

$$\begin{bmatrix} x_i' \\ y_i' \\ z_i' \end{bmatrix} = \begin{bmatrix} c & cs & x_H' & 0 \\ 0 & c(1+m) & y_H' & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} {}^{c}X \\ {}^{c}Y \\ {}^{c}Z \\ 1 \end{bmatrix}$$

In order to model distortion of the lenses, we can hypothesize small corrections (within ~10% of the image size), so that distortion itself can be model in additive manner with respect to the undistorted coordinates $\overline{\boldsymbol{x}}_i'$:

$$\boldsymbol{x}_i' = \overline{\boldsymbol{x}}_i' + \Delta\boldsymbol{x}' \ (\overline{\boldsymbol{x}}_i', \boldsymbol{q})$$

Assuming that the effects on s and m are small, we obtain the approximation:

$$\begin{bmatrix} x_i' \\ y_i' \\ z_i' \end{bmatrix} = \begin{bmatrix} c & cs & x_H' + \Delta x'(\overline{\boldsymbol{x}}_i', \boldsymbol{q}) & 0 \\ 0 & c(1+m) & y_H' + \Delta y'(\overline{\boldsymbol{x}}_i', \boldsymbol{q}) & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} {}^{c}X \\ {}^{c}Y \\ {}^{c}Z \\ 1 \end{bmatrix}$$

For radial distortion, the distortion is 0 at the (optical) center of the imager and increases as we move toward the periphery (Figure 4). In practice, this distortion is small and can be characterized by the first few terms of a Taylor series expansion around $r = 0$.

$$\Delta x' = \bar{x}_i'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

$$\Delta y' = \bar{y}_i'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

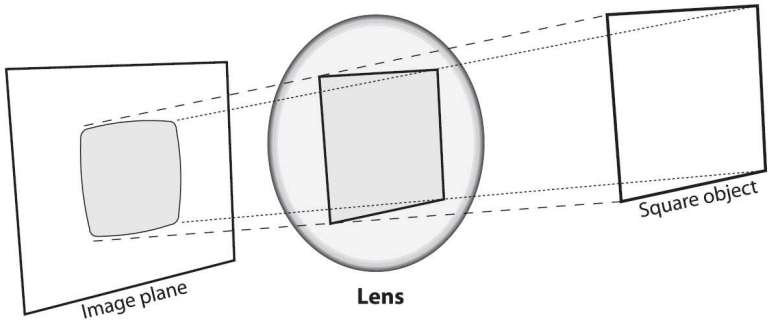

Figure 4. Radial distortion effects

The second-largest common distortion is tangential distortion. This distortion is due to manufacturing defects, resulting from the lens not being exactly parallel to the imaging plane (Figure 5). Tangential distortion is minimally characterized by two additional parameters, p_1 and p_2, such that:

$$\Delta x' = 2p_1 \bar{x}_i' \bar{y}_i' + p_2\left(r^2 + 2\bar{x}_i'^2\right)$$

$$\Delta y' = p_1\left(r^2 + 2\bar{y}_i'^2\right) + 2p_2 \bar{x}_i' \bar{y}_i'$$

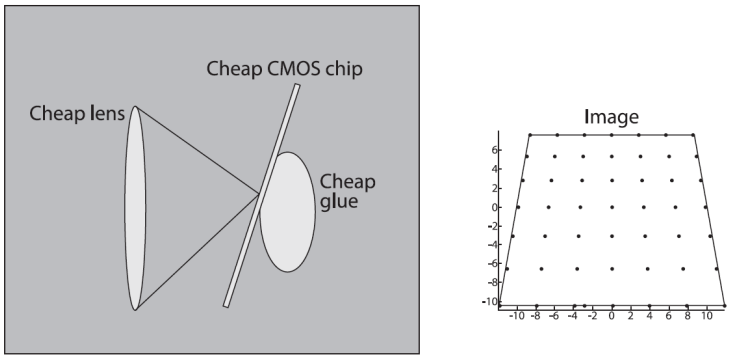

Figure 5. Tangential distortion effects (right panel), as resulting from manufacturing (left panel)

The extrinsic and intrinsic parameters can therefore be combined in a single camera matrix as follows:

$$\begin{bmatrix} x_i' \\ y_i' \\ z_i' \end{bmatrix} = \underbrace{\begin{bmatrix} c & cs & x_H' + \Delta x'(\bar{x}_i', \boldsymbol{q}) & 0 \\ 0 & c(1+m) & y_H' + \Delta y'(\bar{x}_i', \boldsymbol{q}) & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\boldsymbol{K_{3x4}}} \begin{bmatrix} {}^{C}X \\ {}^{C}Y \\ {}^{C}Z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} {}^{C}\boldsymbol{X} \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & 0 \\ \boldsymbol{0}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{I_3} & -\boldsymbol{Z} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{X} \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{R} & -\boldsymbol{RZ} \\ \boldsymbol{0}^T & 1 \end{bmatrix}}_{[\boldsymbol{R}|\boldsymbol{Z}]_{4x4}} \begin{bmatrix} \boldsymbol{X} \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x_i' \\ y_i' \\ z_i' \end{bmatrix} = \boldsymbol{K}_{3x4}[\boldsymbol{R}|\boldsymbol{Z}]_{4x4} \begin{bmatrix} \boldsymbol{X} \\ 1 \end{bmatrix} = \underbrace{\boldsymbol{M}_{4x4}}_{\text{Camera matrix}} \begin{bmatrix} \boldsymbol{X} \\ 1 \end{bmatrix}$$

$$x_i = \frac{m_{11}X_i + m_{12}Y_i + m_{13}Z_i + m_{14}}{m_{31}X_i + m_{32}Y_i + m_{33}Z_i + m_{34}}$$

$$y_i = \frac{m_{21}X_i + m_{22}Y_i + m_{23}Z_i + m_{24}}{m_{31}X_i + m_{32}Y_i + m_{33}Z_i + m_{34}}$$

2.1 Camera calibration and 3D reconstruction

Calibration implies the minimization of projection errors (i.e. the difference between calculated and acquired coordinates) as a function of calibration parameters, in order to determine the parameters that best match the acquired data. Calibration is performed either with 2D calibration object (such as a simple chessboard grid, as shown in Figure 6) or more complex 3D geometries (Figure 7). In both cases objects are moved inside the field of view of the camera in different positions (see Figure 8) in order to span an adequate calibrated volume.

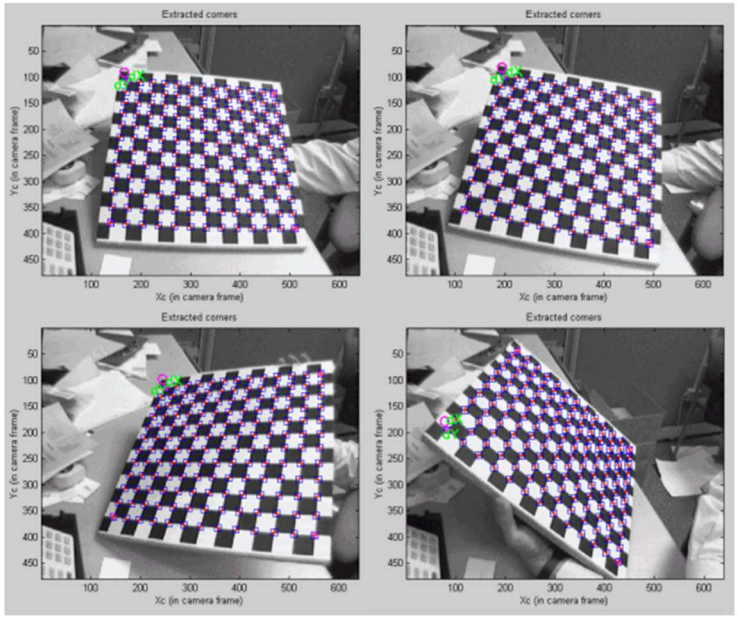

Figure 6. Camera calibration with multiple views of a chessboard grid

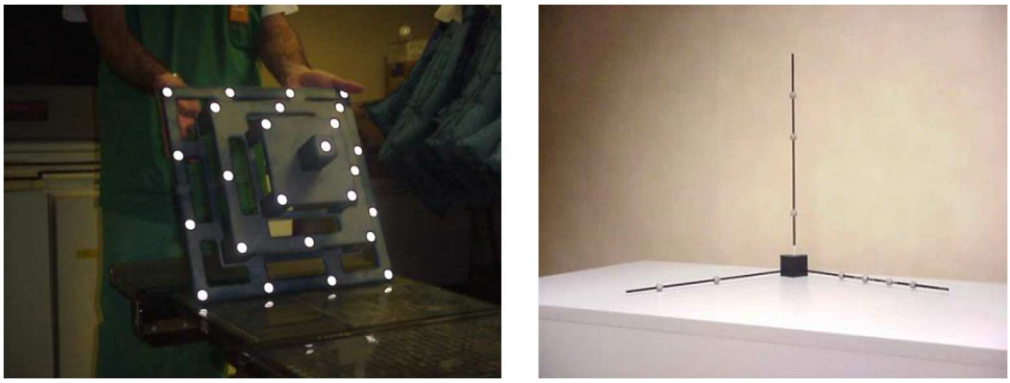

Figure 7. 3D calibration objects: pyramid (left panel) and three axes system (right panel)

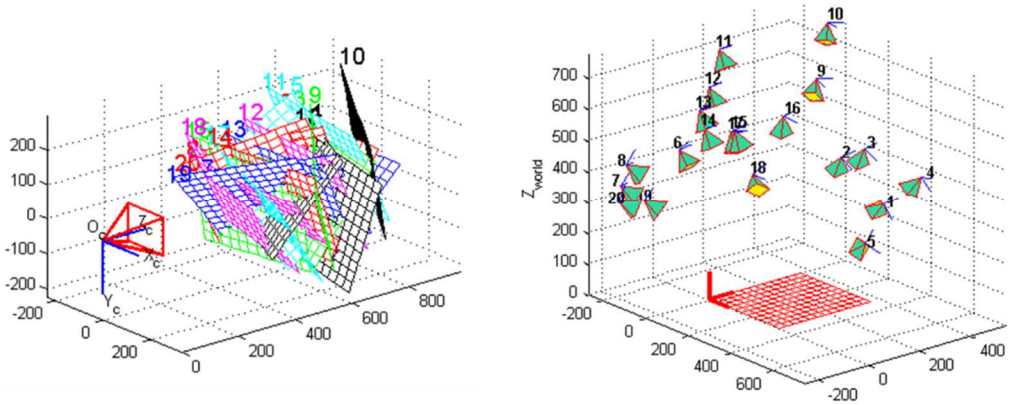

Figure 8. Camera calibration with the chessboard in different positions, shown in the camera reference system (left panel) and with respect to a chessboard-mounted reference system (right panel).

The elements of the camera matrix m_{ij} are determined by minimizing the back-projection error. This can be implemented either as a linear system from a set of redundant points at given 3D coordinates $[X, Y, Z]$ and measured camera projections (x_i, y_i):

$$\begin{cases} x_i \ (m_{31}X_i + m_{32}Y_i + m_{33}Z_i + m_{34}) = m_{11}X_i + m_{12}Y_i + m_{13}Z_i + m_{14} \\ y_i \ (m_{31}X_i + m_{32}Y_i + m_{33}Z_i + m_{34}) = m_{21}X_i + m_{22}Y_i + m_{23}Z_i + m_{24} \end{cases} \Rightarrow \quad \boldsymbol{Q}\,\boldsymbol{m} = \boldsymbol{Q}\begin{bmatrix} m_{11} \\ \cdots \\ m_{34} \end{bmatrix} = \boldsymbol{0}$$

or by minimizing the non-linear cost function:

$$\sum_{i=1}^{N}\left(x_i - \frac{m_{11}X_i + m_{12}Y_i + m_{13}Z_i + m_{14}}{m_{31}X_i + m_{32}Y_i + m_{33}Z_i + m_{34}}\right)^2 + \left(y_i - \frac{m_{21}X_i + m_{22}Y_i + m_{23}Z_i + m_{24}}{m_{31}X_i + m_{32}Y_i + m_{33}Z_i + m_{34}}\right)^2$$

Once camera parameters are known for each camera, the system can detect and reconstruct the 3D coordinates of any point in the 3D object space, relying on the same equations that are solved for [X, Y,

$Z]$, where all other parameters have been determined through the calibration procedure. Collinearity equations define, for each camera, a line in 3D space, connecting the 3D unknown point, the focal point of the camera and the projection of the 3D point on the camera sensor, as shown schematically in Figure 9. The availability of at least 2 cameras allow the calculation of 3D coordinates by intersection of these lines, based on stereoscopic triangulation. Due to numerical errors, these lines might not intersect in reality: in this case, the segment at minimal distance between the 2 lines is considered, and the middle point is assumed as the intersection. In case multiple cameras are available, the calculation is more reliable, and 3D reconstruction can be achieved with reduced residual errors.

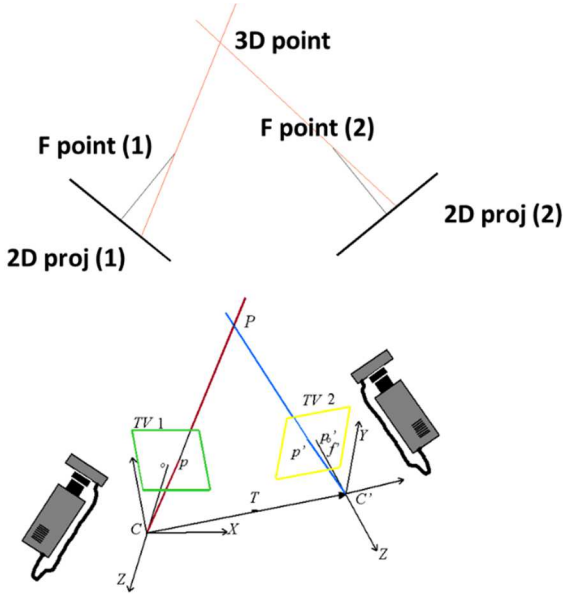

Figure 9. Three-dimensional reconstruction, based on stereoscopic triangulation, with a 2 cameras optical tracking system

3. Principles of surface detection

As an alternative to marker-based localization, different technologies can be used to provide surface detection. There are mainly three approaches to solve the surface acquisition problem, all derived by industrial or commercial applications.

Laser scanner

The reconstruction of the surface is based on triangulation (Figure 10): the spatial position of a point in space is determined by knowing the geometrical characteristics of the laser projector (point or line projection) and its relative position and orientation with respect to the reflected light receiver (commonly a CCD television camera). This kind of scanners as usually fast and accurate, but have the serious drawback of looking at the object (or the patient) from a single point of view. Actually the acquisition of the patient body surface requires the scanning of hidden areas like the lateral sides. Even if the acquisition is quite

fast, the acquisition of the whole surface requires a few seconds, which makes the system not suitable to monitor breathing in real-time. If a breath-hold maneuver is executed, there is no guarantee that the patient will keep the same breathing level for each acquired patch. Monitoring of breathing motion can be achieved by restricting the acquisition to a smaller area, which then allows motion detection at sufficient frequency to capture the breathing cycle.

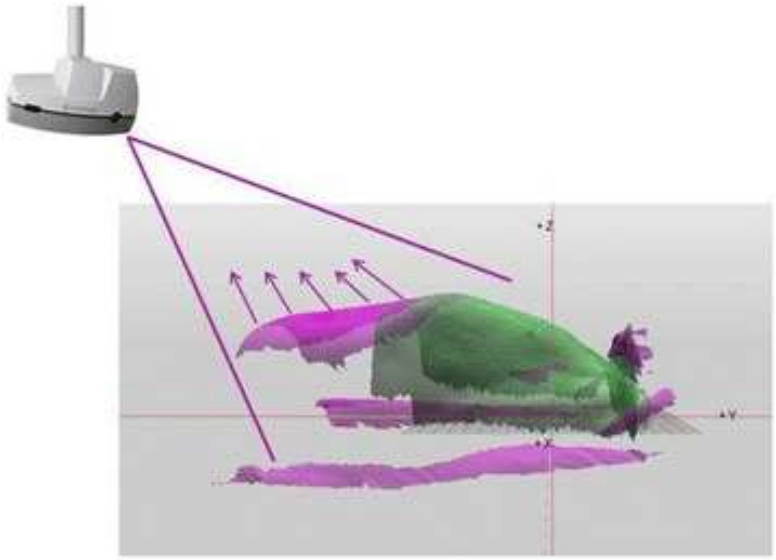

Figure 10. Example of a laser scanner for radiotherapy applications. The projected laser light is reconstructed in 3D relying on known relative geometry between the projector and the sensor.

Fringe-pattern projector

Fringe pattern scanners (also called structured light scanners or Moiré fringes projectors) historically have been the first qualitative tool used for surface morphology acquisition. Their first use was to define asymmetries in body shape to quantify, for example, scoliosis level. Technological development has led to systems able to acquire the same surface portion projecting fringe patterns at increasing spatial frequency, in order to iteratively calculate spatial localization of the subject surface (Figure 11).

This kind of scanners have mainly the same problems of the static laser scanners, as their position relative to the acquired object must be kept constant during the scanning procedure. Anyway, a particular drawback affects this technology is that the color of the acquired surface is a fundamental parameter in spatial tracking: a black or dark spot on a white background, or vice versa, is seen as a difference in depth with respect to the acquisition field.

Figure 11. Fringe projector (A) and fringe projected on a complex surface (B)

Stereo-cameras

The term "stereo-cameras" commonly defines a system made up by two common photographic cameras, rigidly connected to each other and calibrated, that shoot a picture at the same object from two different points of view. These systems typically rely on an infrared pattern projector, which generates a pseudo-random pattern on the surface (Figure 12). The pattern is then analyzed in the two pictures, and correspondent points are detected via a template matching procedure. The 3D positions of such points is then reconstructed relying on stereoscopic principles, similar to what described for marker-based optical tracking (Figure 12). An additional RGB camera can be integrated for texture detection, which provides a realistic representation of the detected surface.

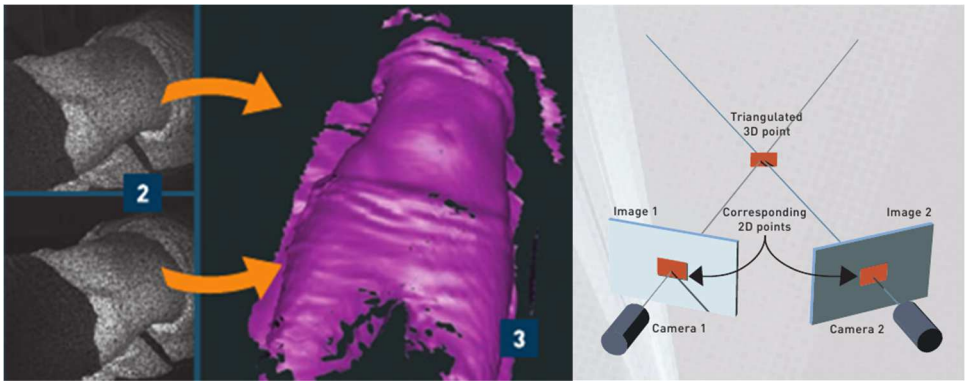

Figure 12. Stereo-cameras system, showing the infrared pattern (left panels), the reconstructed geometrical surface (central panel) and the stereoscopic triangulation principle (right panel)

Stereo-camera systems based on stereoscopic triangulation are now widespread for medical physics applications in external beam radiotherapy. An alternative approach with stereo-cameras, where cameras are positioned in a parallel arrangement is also possible (Figure 13). Such a configuration has been originally proposed for gaming, and has an intrinsically lower 3D reconstruction accuracy. The reconstruction of the patient's surface is made through a so-called disparity map which is used to define

the 3D position of each pixel on the basis of the difference in its 2D position on the two acquired images (see Figure 13). The disparity map is defined as the distance in pixels between each point on the left image and the corresponding one on the right image, after an image rectification process. In order to calculate this distance, a kernel around the considered pixel of the left image is taken and is iteratively superimposed on the right one till the correct match is found. The figure to be minimized is usually the sum of squared distances (SSD), calculated as:

$$SSD_{x,y} = \sum_{i=x-R_H}^{x+R_H} \sum_{j=y-R_V}^{y+R_V} [L(i,j) - R(i-d,j)]^2$$

Where R_H and R_V are the horizontal and vertical kernel dimensions, L and R are the left and right images, and d is the disparity.

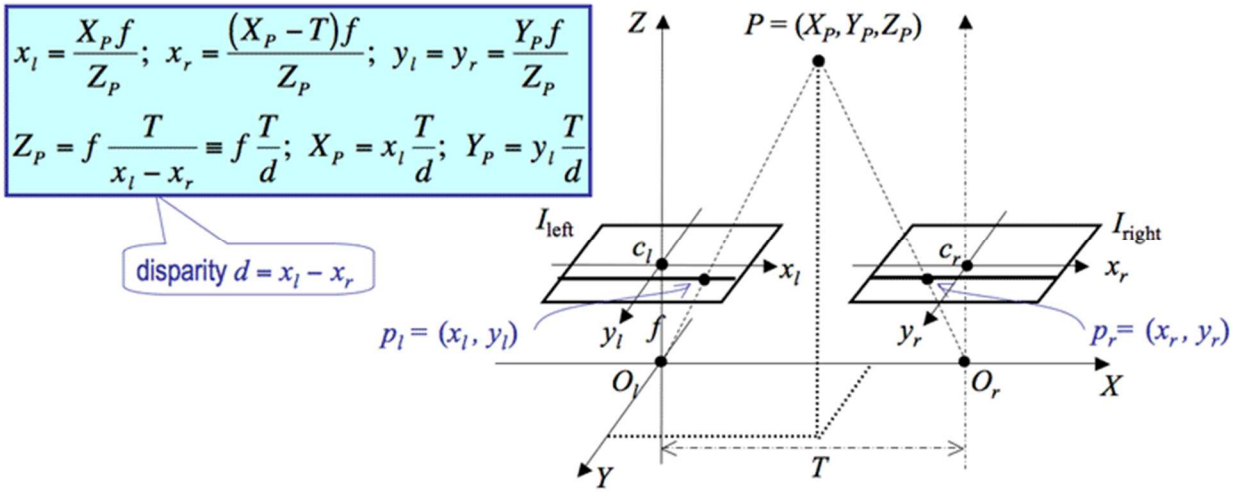

Figure 13. Schematic of the stereoscopic reconstruction based on disparity. The depth of point P with respect to the cameras is calculated as:

$$D = f\frac{T}{x_l - x_r} = f\frac{T}{d}$$

where f is the cameras focal length and $d = x_l - x_r$ is the disparity

4. Experimental methods and materials

4.1 Personal safety: read me first

The Hybrid Polaris Spectra position sensor (described later) is equipped with a Class2 laser that is used to verify the center of the calibrated volume. Pay attention at not looking into the laser aperture directly when pressing the button to activate the laser (Figure 14).

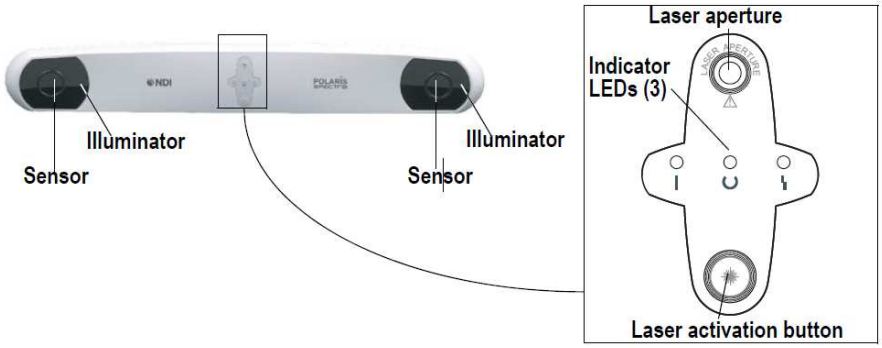

Figure 14. Frontal view of the Hybrid Polaris Spectra position sensor showing the laser activation button and the laser aperture

The following warning is reported on the Hybrid Polaris Spectra device to warn the user (Figure 15).

Figure 15. Laser radiation warning on the Hybrid Polaris Spectra

Please pay attention when using the system, and carefully read the following safety instructions before operating the Hybrid Polaris Spectra system:

Warning!

Do not look directly into the laser-emitting aperture. The Class 2 laser module on the Position Sensor emits radiation that is visible and may be harmful to the human eye. Direct viewing of the laser diode emission at close range may cause eye damage.

Take precautions to ensure that people with restricted movement or reflexes (for example, patients undergoing medical procedures) do not look directly into the laser-emitting aperture. Patients undergoing medical procedures may be restricted in the availability of adverse-effects reflexes (turning away eyes and/or head, closing eyes) due to pharmaceutical influences and/or mechanical restraints. The Class 2 laser module on the Position Sensor emits radiation that is visible and may be harmful to the human eye. Direct viewing of the laser diode emission at close range may cause eye damage.

Laser Activation Button Press this button to activate the positioning laser. The laser will remain on only while the button is pressed.

Warning!

Use of laser controls or adjustments or performance of laser-related procedures other than those specified herein may result in hazardous radiation exposure.

4.2 Hybrid Polaris Spectra

The Hybrid Polaris Spectra system consists of a Position Sensor, a System Control Unit (SCU), Strobers (optional, not used in this lab exercise) and cables, as shown in Figure 16.

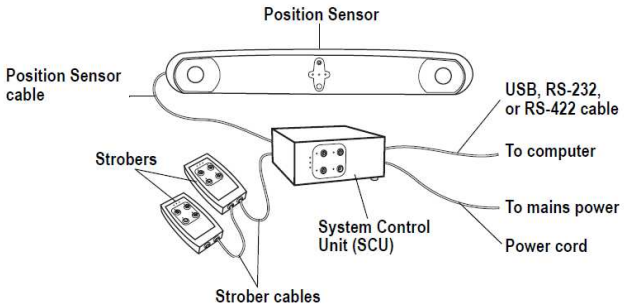

Figure 16. Hybrid Polaris Spectra system setup

The Position Sensor is the main component of the system. An overview of its operation is as follows:

- The Position Sensor emits infrared (IR) light from its illuminators, similar to the flash on a conventional camera. The IR light floods the surrounding area and reflects back to the Position Sensor off passive sphere markers (on passive tools) or triggers markers to activate and emit IR light (on active wireless tools)
- The SCU and/or Strobers activate the markers on the active tools, causing them to emit IR light
- The Position Sensor then measures the positions of the markers, and calculates the transformations (position and orientation) of the tools to which the markers are attached.
- The Position Sensor transmits the transformation data, along with status information, to the host computer for collection, display, or further manipulation.

When connected to the SCU, the Position Sensor can track three types of tools: passive tools, active wireless tools (not used in this praktikum), and active tools.

4.3 Calibrated volume

The Hybrid Polaris Spectra system provides 3D reconstruction in a reference axis system centered on the Position sensor, as schematized in Figure 17.

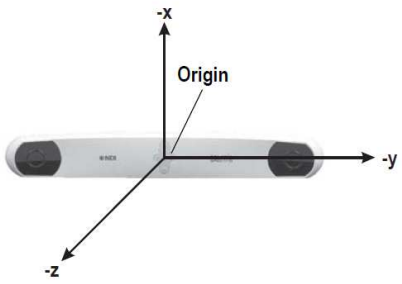

Figure 17. Global coordinates system of the Hybrid Polaris Spectra system

The corresponding calibrated volume depends on system configuration. The Hybrid Polaris Spectra system used for this lab activities features the pyramid volume as effective calibrated volume. The size of such a calibrated volume are shown in Figure 18.

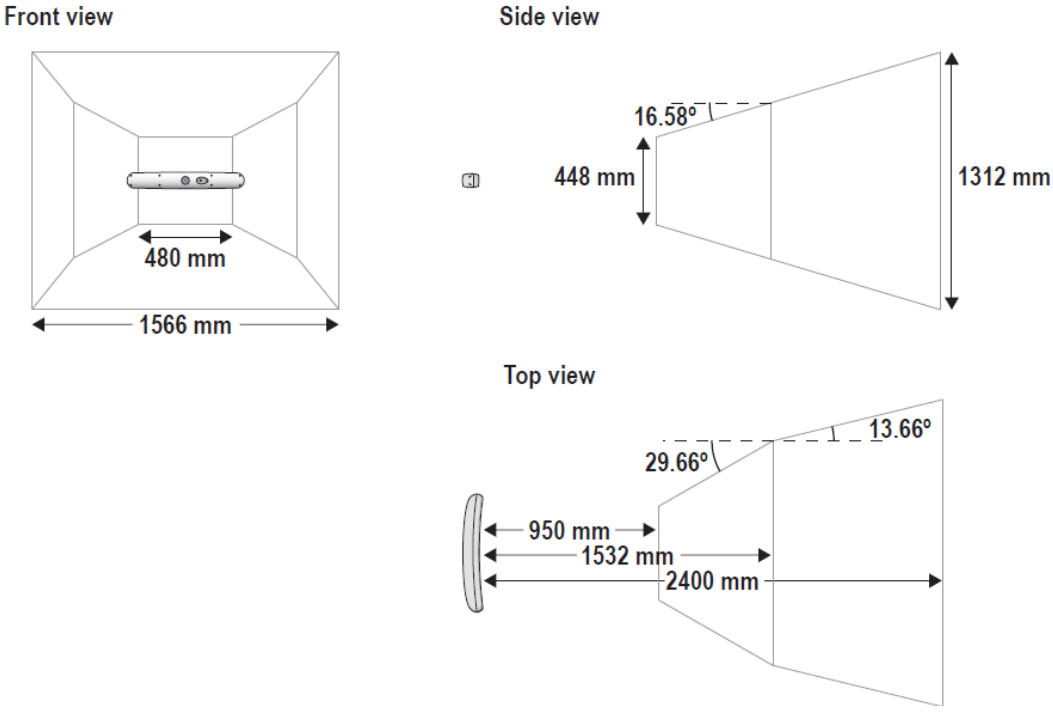

Figure 18. Calibrated volume (pyramid volume) for the Hybrid Polaris Spectra system

4.4 Tools and definition files

A tool is a rigid structure on which three or more markers are fixed so that there is no relative movement between them.

Figure 19. Example of a passive tool

Passive tools (see Figure 19) are equipped with markers, which consist of plastic spheres with a reflective IR coating, able to reflect back the IR light coming from the system cameras. Passive markers need to be handled with care, and using gloves as depicted in Figure 20 in order to maintain the reflective coating properties.

Figure 20. Handling of passive markers

Active tools (see Figure 21) are equipped with active LEDs, and do not require illumination from the Position sensor to be tracked.

Figure 21. Example of an active tool

The system can track passive, active wireless and active tools simultaneously, with the following restrictions:

- The system can simultaneously track up to six passive tools and one active wireless tool, with the following constraint: a maximum of 32 passive and 32 active markers, including stray markers, can simultaneously be in view of the Positions Sensor. Additional markers in view may affect the speed of the system and its ability to return transformations.
- Up to nine active tools can be tracked simultaneously (three connected to the SCU and three connected to each Strober).

Each passive tool has a tool definition file (formatted as **.rom**) to describe it to the system. A tool definition file must be loaded into the system before the system can track the associated tool. Tool definition files can be created and/or edited in **NDI 6D Architect**, whose main window in shown in Figure 22.

The corresponding executable file can be retrieved at:
C:\Program Files (x86)\Northern Digital Inc\NDI 6D Architect 3\6DArchitect.exe

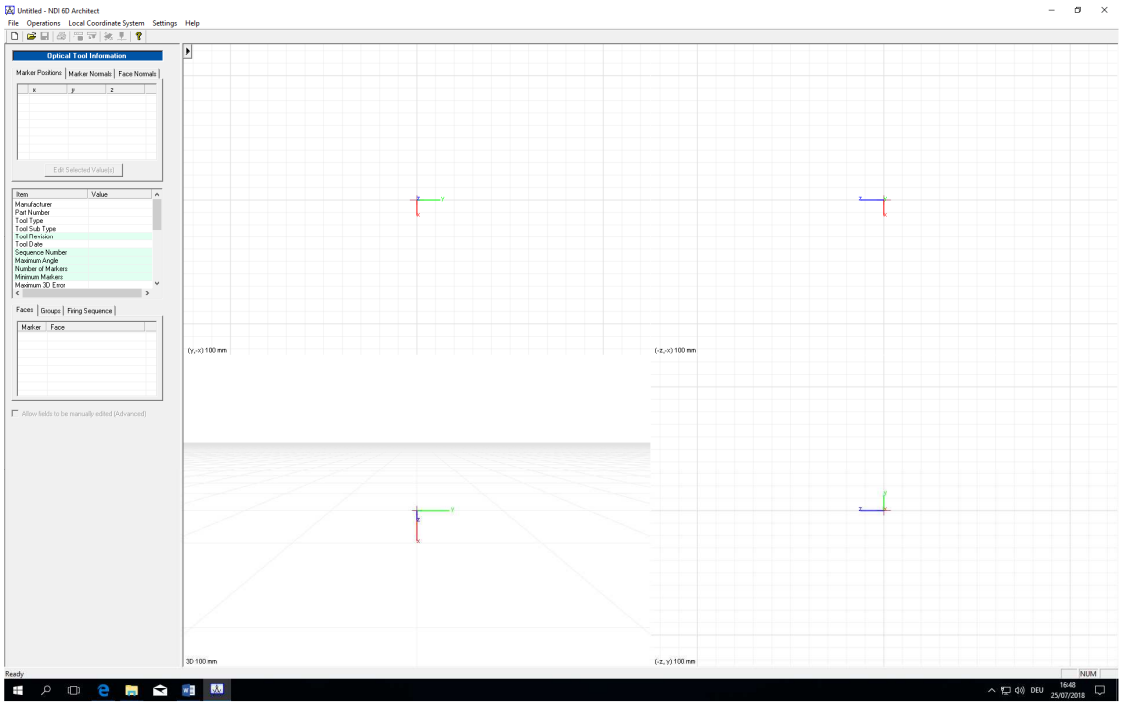

Figure 22. Main window of the NDI 6D Architect software

The following procedure should be used to edit a given tool definition file for changing the local reference system:

- Click on **File → Open** and select the .rom definition file to be edited
- Click on **Local Coordinate System → Define Local Coordinate System**, the window dialog depicted in Figure 23 will appear

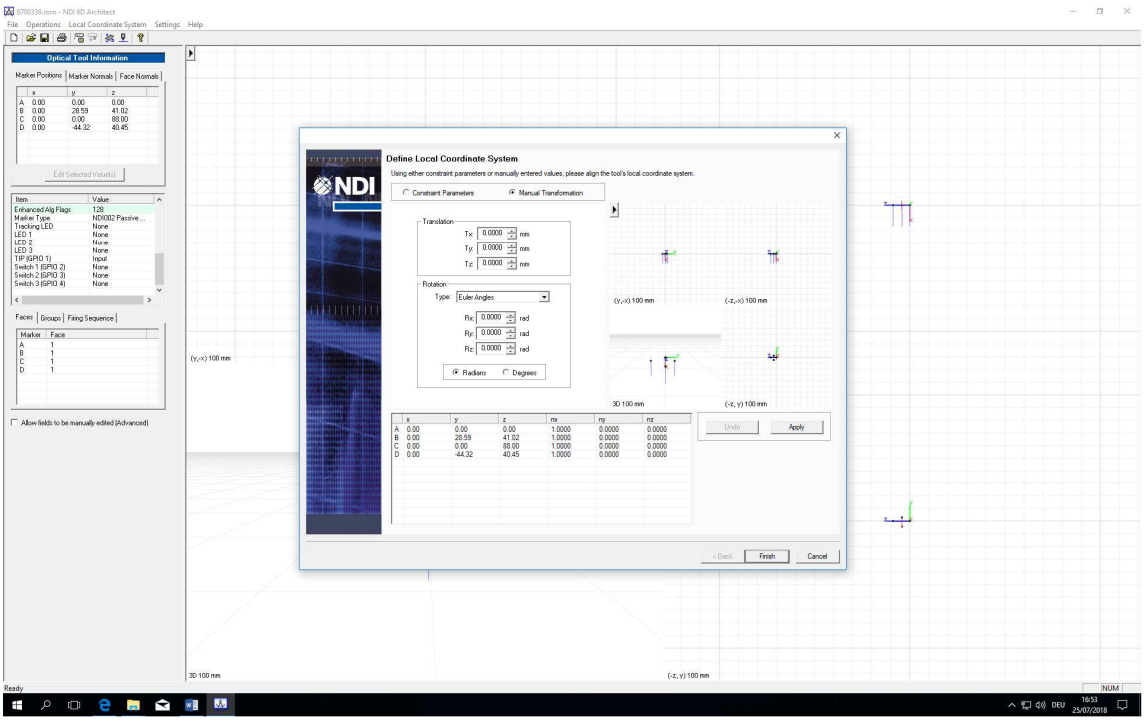

Figure 23. Window dialog for the definition of a local reference system

- Use the Edit transformation to change values and then click **Apply**
- Click **Finish** to apply the selected transformation
- Click on **File → Save As** and save the edited file with an appropriate name (DO NOT OVERWRITE!).

4.5 Tracking tools

The following procedure can be used to track a specific tool with the Hybrid Polaris Spectra system:

- Open the **NDI Track** software 

  The corresponding executable can be found under

  C:\Program Files (x86)\Northern Digital Inc\ToolBox\Track.exe

- The system will beep to establish the connection between SCU and Position Sensor
- A 3D view of the acquisition volume is displayed, showing the calibrated volume and the position of tracked tools, as shown in Figure 24

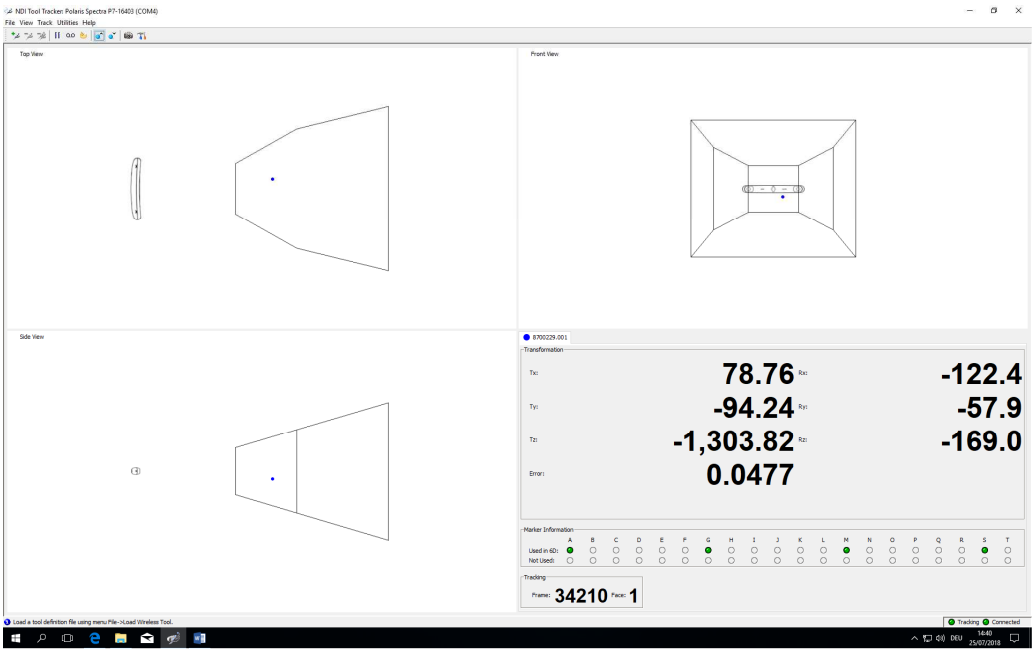

Figure 24. Main window of NDI Track, showing the position of a connected tool

Please note that only tools that have been pre-loaded (and positioned in the field of view) will be visible, which includes connected active tools and wireless tools that have not been unloaded in the previous acquisition session

- Verify that connection is established on COM4 (as shown in the main window title), by checking

  File → Connect to →

  which should highlight COM4 as pre-selected

- To track an active tool, plug the tool into the SCU or strober
- To track a passive or active wireless tool:
  - Click **File→Load Wireless Tool** to load the tool definition files for the tool you want to track
  - In the dialog that appears, browse to the desired tool definition file(s). Hold down **Ctrl** and click to select more than one file
  - Click **Open**
- Once a tool has been plugged in or a tool definition file has been loaded, the Hybrid Polaris Spectra system will automatically attempt to track the tool
- Move the tool throughout the characterized measurement volume, making sure the markers on the tool face the Position sensor. As you move the tool, the symbol representing the tool in the graphical representation will move to reflect the tool´s position.

A tool can be set as a reference, so that all other tools will be tracked with respect to the reference one. To setup a tracked tool as reference, proceed as follows:

- For each tool, there is a tab in the bottom right section of the **NDI Track** utility. Select the tab corresponding to the tool you want to set as reference, as shown in Figure 25

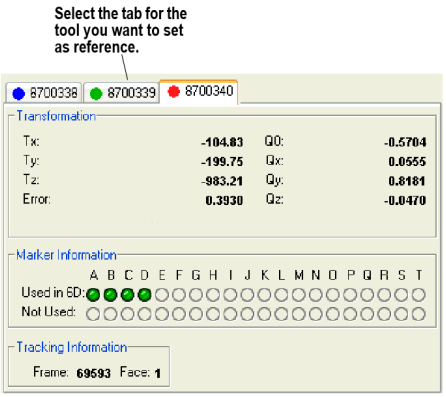

Figure 25. Selection of a reference tool

- Right click on the tool tab, then select **Global Reference**.

The reference tool will appear as a square in the graphical display. The other tools will be displayed inside a square that is the color of the reference tool. The positions and orientations of other tools will now be reported in the local coordinate system of the reference tool.

4.6 Pivoting tools

This section describes how to determine the tool tip offset of a probe or pointer tool by pivoting. Once the tool tip offset is applied, the Hybrid Polaris Spectra will report the position of the tip of the tool, instead of the position of the origin of the tool:

- Setup the system to track tools, as described in the previous paragraph
- Plug in (active tools) or load a tool definition file (passive and active wireless tools) for the probe or pointer tool
- For each tool, there is a tab in the bottom right section of the **NDI Track** utility. Select the tab corresponding to the tool you want to pivot
- Select **Pivot tool**
- Select in the **Pivot tool** dialog window that appears a start delay (e.g. 5 seconds) and a duration of about 20 seconds
- Pivot the tool.

The pivoting procedure should be carried out as follows:

- Place the tool tip in the hole at the center of the pivot base, as shown in Figure 26

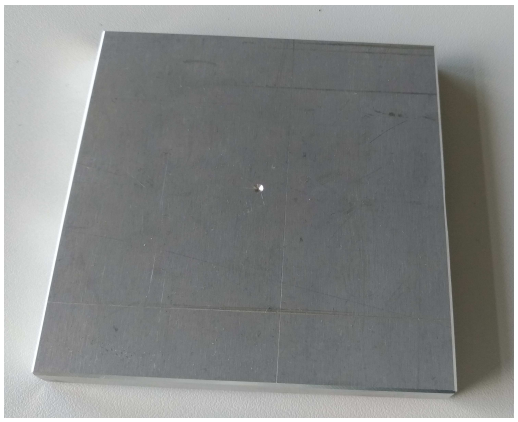

Figure 26. Pivoting base

- Ensure that the tool is within the characterized measurement volume, and will remain within the volume throughout the pivoting procedure
- Click **Start Collection** in the **Pivot tool** dialog
- Pivot the tool in a cone shape, at an angle of 30 to 60 degrees from the vertical (Figure 27)
    - Keep the tool tip stationary, and ensure that there is a line of sight between the markers on the tool and the Position sensor throughout the pivoting procedure
    - Pivot the tool slowly until the specified pivot duration time has elapsed

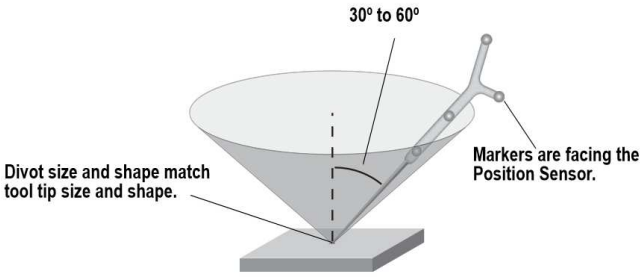

Figure 27. Pivoting technique

- When the pivot is complete, the **Pivot Result** dialog appears (Figure 28). Click **Apply Offset** to report the position of the tip of the tool.

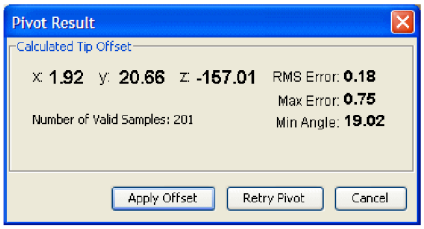

Figure 28. Pivot Result dialog

4.7 Stray passive markers

A stray passive marker is a passive marker that is not part of a rigid body or tool. For example, by placing stray markers on a patient´s chest, the markers may be used to gate/track the patient´s breathing in order to achieve motion compensation in radiation therapy.

As already pointed out in the description of passive tools, stray markers should be handled with care when they are positioned and removed, so that the reflective coating is not damaged.

When you request stray passive marker data from the Hybrid Polaris Spectra, the system will report tool transformations, as well as 3D data (position only, no orientation information) and out-of-volume information for up to 50 markers that are not used in tool transformations. It is then necessary to eliminate phantom markers (i.e. undesired reflections), and verify that the stray markers are within the characterized measurement volume.

It is important to be aware of the potential hazards associated with using the stray passive marker reporting functionality. The hazards are as follows:

- An external IR source, for example and IR transmitter of incandescent light, may be reported as a stray marker
- No marker identification is possible from frame to frame. It is therefore the user´s responsibility to devise a method to keep track of which 3D position belongs to which marker
- There are no built-in checks to determine if the 3D result is a real marker or a phantom marker, generated by other IR sources or makers in view of the Position sensor.
- Partial occlusions of markers cannot be detected or compensated for by the Position sensor. The user may be able to detect the apparent shift if the marker position can be constrained in the application software. For example, the marker position has to be constrained along a vector and its position relative to another marker is supposed to be fixed within some tolerance.

In order for the Position sensor to measure stray passive markers, a tool definition file must be loaded (which implies a passive tool is used), and the associated port handle must be initialized and enabled, even if no tools are being tracked. The Position sensor illuminators emit IR light only when a tool definition file is loaded.

4.8 Intel® RealSense™ cameras

The Intel® RealSense™ D455 series depth camera (Figure 29) uses stereo vision to calculate depth. The stereo vision implementation consists of a left imager, right imager, and an infrared projector. The infrared projector projects non-visible static IR pattern to improve depth accuracy in scenes with low texture. The left and right imagers capture the scene and sends imager data to the depth imaging (vision) processor, which calculates depth values for each pixel in the image by correlating points on the left image to the right image and via shift between a point on the left image and the right image (disparity). The depth pixel values are processed to generate a depth frame. Subsequent depth frames create a depth video stream.

The camera system has two main components, the Vision processor D4 and the Depth module. The Depth module incorporates left and right imagers for stereo vision with the optional IR projector and RGB color sensor. The RGB color sensor data is sent to vision processor D4 via the color Image Signal Processor (ISP) on Host Processor motherboard or D4 Board.

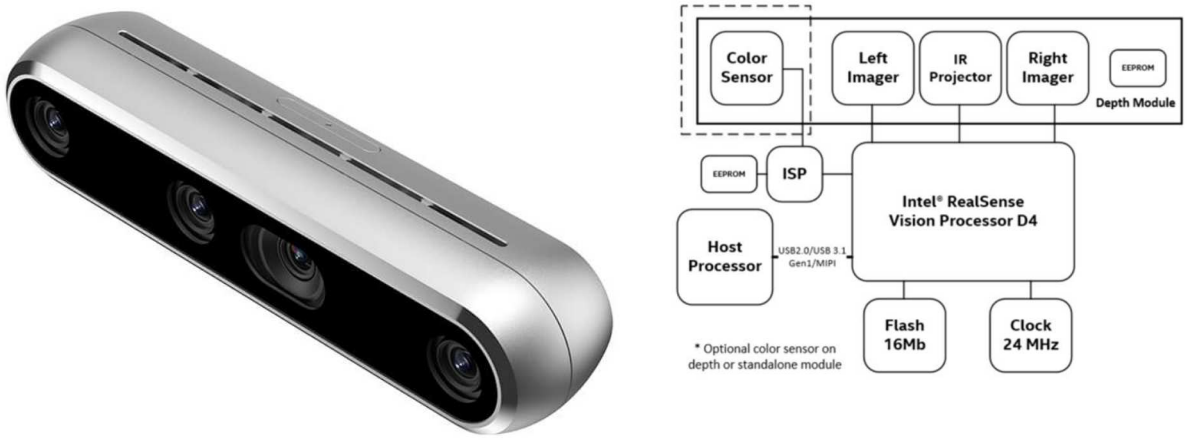

Figure 29. Intel® RealSense™ D455 depth camera (left panel) and camera system block diagram (right panel)

The main features of this specific camera model are as follows:

- Up to 1280x720 active stereo depth resolution
- Up to 1280x800 RGB resolution
- Diagonal Field of View over 90°
- Dual global shutter sensors for up to 90 FPS depth streaming
- RGB global shutter sensor for up to 90 FPS
- Range 0.4m to over 10m (varies with lighting conditions)
- Inertial Measurement Unit (IMU) for 6 degrees of freedom (6DoF) data.

5. Experimental procedure

The following preliminary operations should be performed for the Hybrid Polaris before acquisition:

- Switch on power in the Hybrid Polaris Spectra system, with the switch positioned on the rear part of the SCU (Figure 30)

Figure 30. Rear panel of the SCU, with the power button on the left side of the power cable

- The SCU and the Position sensor will beep in a sequence
- Wait for system to be ready: the two green LEDs on the Position sensor will stop flashing. When they are both on and green the system is ready (Figure 31)

Figure 31. Position sensor, showing the steady green LEDs on (center of the device) for correct system operation

Please report any anomalous behavior before continuing with the next steps, as this might compromise the reliability of your measurements and/or the system itself.

The following preliminary procedures should be followed for the Intel® RealSense™ camera. The camera comes with USB C type connector and a small tripod (Figure 32):

- Connect the camera (USC C connector end) to one of the USB 3.0 port in PC. The camera is powered through same connector.
- Make sure the tripod is tightened up (lateral screw, see Figure 32) for camera stability during measurements.

Figure 32. Intel real sense camera and usb c connector

5.1 Quantification of calibration accuracy

The following procedure should be followed to perform 2 acquisitions within the calibrated volume:

- Start the **NDI Track** application

  The corresponding executable can be found under

  C:\Program Files (x86)\Northern Digital Inc\ToolBox\Track.exe

- Check that at least one passive tool is connected, looking at the tabs on the right bottom part of the main window. **Please double check that the connected tool is a passive one, otherwise the IR LEDs of the Hybrid Polaris Spectra will not flash, which would hinder the acquisition of stray markers**
- Check that **Track → Report Stray Markers** is flagged, you should also see a **Stray Markers** tab on the bottom right of the main window

- Check that tracking in ON: the **Frame** counter on the bottom right is continuously increasing. In case tracking is not active, press the **Play** button  or click on Track → Resume Tracking
- Prepare the passive tool probe so that it can be used as a 2 markers bar. Select two markers at known distance (the larger the better) and remove carefully (**USE GLOVES!!**) the additional two markers that are not needed. Please refer to Figure 33 and Table I for the selection of markers. Keep track of the nominal distance between the 2 markers

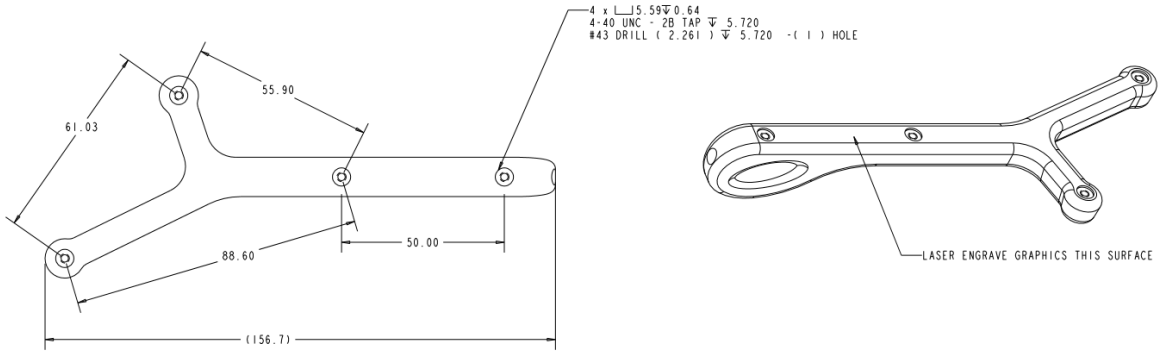

Figure 33. Drawing of the passive tool probe showing the geometric distance between markers

Table I. Marker and tool tip coordinated in the passive tool probe definition file

	X [mm]	Y [mm]	Z [mm]
A	0	0	0
B	0	0	50
C	0	25	100
D	0	-25	135
Tip	-18.305	0	-163.4125

- Prepare to record data by clicking the **Record** button or selecting **File→ Record Data**
- Select a file name (with .tsv extension), and appropriate folder to store your data (the default will be **C:\Polaris_data**)
- Select an appropriate **Start Delay** (10 seconds) and **Record Duration** (30 seconds)
- Stray markers will be displayed in real-time with the following symbol (**+**): please check throughout the acquisition to judge which part on the field of view you are actually covering
- Move the bar in the calibrated volume. Perform at least **2 acquisitions**, one at the center of the field of view and one close to the extreme border. Please consider that in the lab setup, the effective field of view will be limited to the area closer to the Position sensor, as depicted in Figure 34.

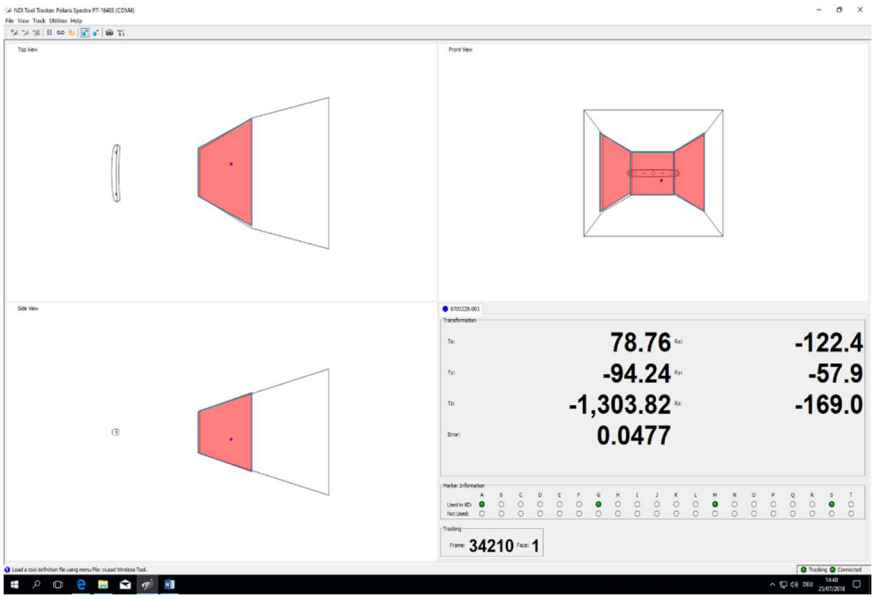

Figure 34. Useful calibrated volume in the laboratory setup (shaded red area)

5.2 Reference phantom acquisition (intel-sense + Polaris)

The purpose of this procedure is to acquire a reference position for the male torso phantom (Figure 35). The following tasks should be carried out for acquisition with the Hybrid Polaris system:

- Start the **NDI Track** application

  The corresponding executable can be found under

  C:\Program Files (x86)\Northern Digital Inc\ToolBox\Track.exe

- Check that the active tool (Figure 21) is connected to the SCU
- Check that one of the passive tools is connected to the software application. If not, follow the procedure described in paragraph 3.5 to connect it (using the tool definition file **8700339.rom** or **8700340.rom**, that can be found in **C:\Optical_tracking_praktikum\tracking_tool**). This is required to activate the tracking options for stray markers
- Position the pivoting stage in the FOV, and check whether pivoting (see paragraph 4.6) is feasible in that position
- Pivot the active tool as described in 4.6, so that the probe tip is reported
- Check that **Track → Report Stray Markers** is flagged in the Hybrid Polaris software, you should also see a **Stray Markers** tab on the bottom right of the main window
- Position the torso phantom (Figure 35) in a reference position, making sure that the Hybrid Polaris covers the surface areas corresponding to the reflective markers (the reflective coating of the marker disks will be detected as stray marker)
- Make a short acquisition (5-10 s) for the stray markers, making sure that at least 4 out of 5 are visible
- Track the reflective marker disks while sweeping the round border with the active tool, which has a round tip. Pay attention to keep the tool tip in contact with the border, at an inclination so that the active tool is correctly localized with the Hybrid Polaris. SUGGESTION: acquire the 5 reflective

marker disks with 5 separate acquisitions, for better handling data in the analysis phase. Make sure the phantom does not move for the whole duration of data acquisition

- Save data following the steps described in paragraph 5.1.

Figure 35. Torso phantom setup, showing the five reflective dots positioned onto the surface for measurement with the optical tracking device

As a second step, acquire a reference with the Intel® RealSense™ camera:

- Open **Matlab R2017a** and set the directory to **C:\Optical_tracking_praktikum\Intelsense**
- Place the Intel® RealSense™ camera facing the torso phantom and later in the Matlab **command window** type **realsense.Single_frame_depth**
- A reference frame (Figure 36) will be acquired and saved in the default folder **C:\Intel_Realsense_data** named as **Reference_frame.mat** (one can change the default folder/filename name to save the reference frame in the source code of the function **C:\Optical_tracking_praktikum\Intelsense\+realsense\Single_frame_depth.m**)

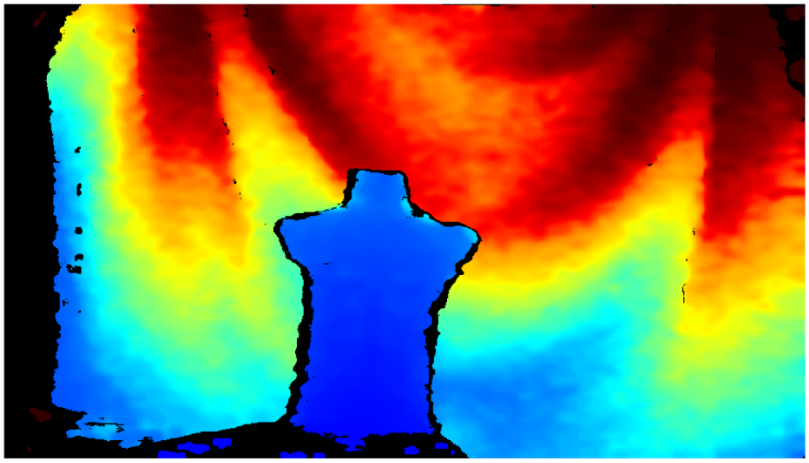

Figure 36. Reference frame of the torso phantom

5.3 Phantom repositioning based on surface information

Move the male torso phantom in a different position with respect to the reference position (paragraph 5.2). Using the interactive visual feedback from the Intel® RealSense™ camera, repositioning manually the phantom in approximately the same position defined as reference, according to the following procedure:

- Make sure the Matlab directory is set to **C:\Optical_tracking_praktikum\Intelsense**
- In the Matlab **command window** type **realsense.Continous_frame_depth_with_reference**
- A pop window will open with an overlay figure of current frame and reference frame. One can reposition the phantom to the reference frame as shown in Figure 37

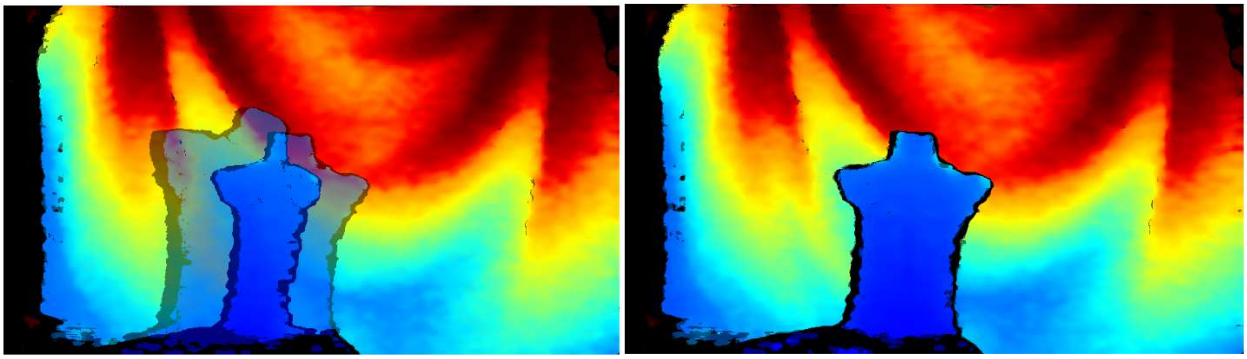

Figure 37. Overlay of torso phantom in the initial position (before alignment, left panel) and after reaching the final setup (after alignment, right panel)

- **Save a screenshot of the overlay** in the initial and final position (same as Figure 37)
- The source code of function **realsense.Continous_frame_depth_with_reference** can be found in **C:\Optical_tracking_praktikum\Intelsense\+realsense\Continous_frame_depth_with_reference.m**
- Make sure to change the respective **folder/file name** of reference frame (if modified in **realsense.Single_frame_depth.m**)

- Note: The default time for continuous frame acquisition is set to 10 minutes approx. but one can change this time in the function **C:\Optical_tracking_praktikum\Intelsense\+realsense\ Continous_frame_depth_with_reference.m**

Once the final position has been reached, save the corresponding data with the Hybrid Polaris according to the same procedure used in paragraph 5.2. This means **you will need to re-measure the position of the 5 reflective marker disks**, i.e. we need the circles of each disk at the initial position (reference) and after repositioning with the Intel® RealSense™ camera.

6. Data analysis

6.1 Quantification of calibration accuracy

Calibration accuracy should be calculated and reported as follows:

- Read the acquisitions using either:
  - the **read_stray_markers.m** script, which will be available in **C:\Optical_tracking_praktikum\scripts** (edit the **max_markers** variable to take into account that 2 markers are expected → **max_markers = 2**)
  - the Python script **read_stray_markers.py**, also available in **C:\Optical_tracking_praktikum\scripts**
- Calculate the standard deviation and 95th percentile of distance between markers (variable **delta**, as reported and plotted by the scripts)
- Compare such results with the nominal distance, as calculated in paragraph 5.1
- Report calibration accuracy in the center of the field of view vs. close to the border

6.2 Quantification of phantom repositioning accuracy

Proceed as follows to localize the phantom in the reference position:

- Read the reference acquisitions using either:
  - the **read_tool.m** script, which will be available in **C:\Optical_tracking_praktikum\scripts**
  - the Python script **read_tool.py**, also available in **C:\Optical_tracking_praktikum\scripts**
- Verify the measurement error, as reported by the optical tracking system in the variable **err**: please check that the maximum error is below 0.15 mm
- In case the maximum error exceeds the 0.15 mm threshold, filter out points whose error is larger: if remaining points are sufficient to characterize the reflective marker disk proceed, otherwise repeat the measurement as described in paragraph 5.2
- Quantify the centroid of the acquired circle trajectory, representing the center of the reflective marker. Please take into account partial acquisitions (i.e. leading to a shifted average position) and the fact that the active marker tool tip has a 3 mm diameter

Repeat the previous steps for the acquisition relative to paragraph 5.3, thus characterizing the phantom position after manual repositioning based on surface information.

Finally, quantify the repositioning accuracy by comparing the two acquisitions, and interpret the obtained results.

7. References

Wagner TH, Meeks SL, Bova FJ, Friedman WA, Willoughby TR, Kupelian PA, Tome W. Optical tracking technology in stereotactic radiation therapy. Med Dosim. 2007 Summer;32(2):111-20

Bert C, Metheany KG, Doppke K, Chen GT. A phantom evaluation of a stereo-vision surface imaging system for radiotherapy patient setup. Med Phys. 2005 Sep;32(9):2753-62

Walter F, Freislederer P, Belka C, Heinz C, Söhn M,Roeder F. Evaluation of daily patient positioning for radiotherapy with a commercial 3D surface-imaging system (Catalyst™). Radiat Oncol. 2016 Nov 24;11(1):154

Förstner W, Wrobel BP, Photogrammetric Computer Vision: Statistics, Geometry, Orientation and Reconstruction, Springer, ISBN 978-3-319-11549-8
Chapters 11, 12

Forsyth DA, Ponce J. Computer Vision: A modern approach, Prentice Hall, ISBN: 978-0136085928
Chapter 05

Zhang, A Flexible New Technique for Camera Calibration, MSR-TR-98-71

Bradski G, Kaehler A, Learning OpenCV, O'Reilly Media, Inc., ISBN: 9780596554040
Chapter 11

Brown C, Decentering Distortion of Lenses", Photometric Engineering 32(3) (1966), 444–462.

Northern Digital Inc., Hybrid Polaris Spectra User Guide, Revision 5, June 2013

Fielding AL, Pandey AK, Jonmohamadi Y, Via R, Weber DC, Lomax AJ, Fattori G. Preliminary study of the Intel RealSenseTM D415 camera for monitoring respiratory like motion of an irregular surface. IEEE Sensors Journal, doi: 10.1109/JSEN.2020.2993264.

https://www.intel.com/content/www/us/en/architecture-and-technology/realsense-overview.html

Appendix

MatLab scripts

read_tool.m

```
close all;
clear all;
serial_number='34801403'; %Polaris serial number
data_file='new_praktikum_01.tsv';

%read file
S=importdata(data_file,'\t');

%...GET TOOL.....................................
n_frame=zeros(size(S.textdata,1)-1,2);
data=NaN*ones(size(S.textdata,1)-1,3);
err=NaN*ones(size(S.textdata,1)-1,1);
n_tools=str2double(S.textdata(2,1));
% skip_cells=1+16*(n_tools-1)+13;
format_header='%s';
for k=1:size(S.textdata,2)-1
    format_header=strcat(format_header,'%s');
end
format_data=format_header;
for k=1:n_tools
    format_data=strcat(format_data,'%s%s%s%s');
end
FID = fopen(data_file);
H = textscan(FID,format_header,1); %read the header
ind_tool=[];
for k=1:length(H)
    if strcmp(H{k},'Tx')==1
        ind_tool=[ind_tool;k];
    end
end
temp=fgetl(FID); %empty line
for i=2:size(S.textdata,1) %potential correct frames (from 2 to end of file)
    temp=str2double(S.textdata(i,3));
    if mod(temp,1)==0 %check if the frame number is an integer value (valid
frames)
        n_frame(i-1,1)=temp;
%         temp=textscan(FID,format_data,1);
        temp=fgetl(FID); %empty line
        temp2=strsplit(temp);
        if strcmp(temp2{9},'OK')==1 %check if tool has been detected
            data(i-1,1)=str2num(temp2{ind_tool(1)}); %13 to 16
            data(i-1,2)=str2num(temp2{ind_tool(1)+1});
            data(i-1,3)=str2num(temp2{ind_tool(1)+2});
            err(i-1,1)=str2num(temp2{ind_tool(1)+3});
        else
```

```
            n_frame(i-1,1)=NaN;
        end
    else
        n_frame(i-1,1)=NaN;
    end
end
fclose(FID);
...END (GET TOOL).....................................

%PLOT
close all;
plot3(data(:,1),data(:,2),data(:,3),'.');
xlabel('X');
ylabel('Y');
zlabel('Z');
axis equal;
ok_err=find(~isnan(err));
if ~isempty(ok_err)
    figure
    hist(err);
end
```

read_stray_markers.m

```
close all;
clear all;
serial_number='34801403'; %Polaris serial number
data_file='phantom3.tsv';
max_markers=5; %maximum number of stray markers

%read file
S=importdata(data_file,'\t');

%...GET STRAY MARKERS.................................
n_frame=zeros(size(S.textdata,1)-1,2);
stray_markers=NaN*ones(size(S.textdata,1)-1,max_markers,3);
n_tools=str2double(S.textdata(2,1));
% skip_cells=1+16*(n_tools-1)+13;

format_header='%s';
% for k=1:(skip_cells+4*max_markers)-1
for k=1:size(S.textdata,2)-1
    format_header=strcat(format_header,'%s');
end
FID = fopen(data_file);
H = textscan(FID,format_header,1); %read the header
temp=fgetl(FID); %empty line
for i=2:size(S.textdata,1) %potential correct frames (from 2 to end of file)
    temp=str2double(S.textdata(i,3));
```

```
    if mod(temp,1)==0 %check if the frame number is an integer value (valid
frames)
        n_frame(i-1,1)=temp;
        temp=fgetl(FID); %get next line (number of entries vary depending on
tool status
        ind_OK=strfind(temp,'OK');
        ind_OK=[ind_OK,size(temp,2)];
        if length(ind_OK)>max_markers %>=
            ind_OK=ind_OK(end-max_markers:end);
            for k=1:(max_markers)
                data_k=temp(ind_OK(k)+2:ind_OK(k+1)-1);
                data_temp=str2num(data_k);
                if length(data_temp)>0
                    stray_markers(i-1,k,1)=data_temp(1); %X
                    stray_markers(i-1,k,2)=data_temp(2); %Y
                    stray_markers(i-1,k,3)=data_temp(3); %Z
                else
                    stray_markers(i-1,k,1)=NaN;
                    stray_markers(i-1,k,2)=NaN;
                    stray_markers(i-1,k,3)=NaN;
                end
            end
        end
    else
        n_frame(i-1,1)=NaN;
    end
end
fclose(FID);
%...END(GET STRAY MARKERS).........................

%PLOT
figure;
x=stray_markers(:,:,1);
y=stray_markers(:,:,2);
z=stray_markers(:,:,3);
for k=1:max_markers
    plot3(x(:,k),y(:,k),z(:,k),'.');
    hold on
end
axis equal
```

Single_frame_depth.m

```
function Single_frame_depth()
    % Make Pipeline object to manage streaming
    pipe = realsense.pipeline();
    % Make Colorizer object to prettify depth output
    colorizer = realsense.colorizer();

    % Start streaming on an arbitrary camera with default settings
    profile = pipe.start();
```

```
    % Get streaming device's name
    dev = profile.get_device();
    name = dev.get_info(realsense.camera_info.name);

    % Get frames. We discard the first couple to allow
    % the camera time to settle
    for i = 1:15
        fs = pipe.wait_for_frames();
    end

    % Stop streaming
    pipe.stop();

    % Select depth frame
    depth = fs.get_depth_frame();
    % Colorize depth frame
    color = colorizer.colorize(depth);

    % Get actual data and convert into a format imshow can use
    % (Color data arrives as [R, G, B, R, G, B, ...] vector)
    data = color.get_data();
    img = permute(reshape(data',[3,color.get_width(),color.get_height()]),[3
2 1]);

    % Display image
    imshow(img);
    title(sprintf("Reference depth frame from %s", name));

    % Path and filename for saving reference frame
    % Change to desired file location/name
    save(['C:\Intel_Realsense_data\Reference_frame'],'img','-v7.3')
end
```

Continous_frame_depth_with_reference.m

```
function Continous_frame_depth_with_reference()

    % Make Pipeline object to manage streaming
     pipe = realsense.pipeline();
    % Make Colorizer object to prettify depth output
    colorizer = realsense.colorizer();

%     Image_reference_frame=uint8(ones(480,848,3));

% % % Loading Reference frame

Image_reference_frame=load('C:\Intel_Realsense_data\Reference_frame.mat');
    % Start streaming on an arbitrary camera with default settings
    profile = pipe.start();
    figure('visible','on');  hold on;
    % Get streaming device's name
```

```
    dev = profile.get_device();
    name = dev.get_info(realsense.camera_info.name);

    % Main loop
%  for i = 1:1000 for ~119sec
%  for i = 1:5000 for ~10mins
%  for i = 1:10000 for ~20mins
    tic
    for i = 1:5000

        % Obtain frames from a streaming device
        fs = pipe.wait_for_frames();

        % Select depth frame
        depth = fs.get_depth_frame();
        %color = fs.get_color_frame();
        color = colorizer.colorize(depth);
        data = color.get_data();
        depth_rgb=fs.get_color_frame();
        color_rgb=depth_rgb;
        data_rgb = color_rgb.get_data();
         % Depth frame
        img =
permute(reshape(data',[3,color.get_width(),color.get_height()]),[3 2 1]);
        % Rgb_frame
        img_rgb =
permute(reshape(data_rgb',[3,color_rgb.get_width(),color_rgb.get_height()]),[
3 2 1]);
           hold off;

        %Skipping inital few frames to allow the camera time to settle
           if i>15
%                 Image_reference_frame=im2bw(rgb2gray(img),0.05);
            % % % % Difference of realtime frame Vs reference frame

imshowpair(img,Image_reference_frame.img,'blend','Scaling','joint')
% %                current depth frame
%                imshow(img);
% %                current RGB frame
%                imshow(img_rgb);
              pause(0.01);
           else

%


           end
        % pcshow(vertices); Toolbox required
    end
toc
     % Stop streaming
    pipe.stop();

end
```

Python scripts

read_stray_markers.py

```
# Import section
import pandas as pd
import matplotlib.pyplot as plt
from numpy import linalg as LA

# Data file
data_file = "calib_check.tsv"

# Read data
marker_data = pd.read_table(data_file, sep='\t', engine='python',
on_bad_lines='skip')

# Filter data
marker_data = marker_data.loc[(marker_data["Markers"] == 2) &
(marker_data["State.2"] == "OK")]

markers = [marker_data[["Tx.1", "Ty.1", "Tz.1"]].to_numpy(),
           marker_data[["Tx.2", "Ty.2", "Tz.2"]].to_numpy()]

# Plot marker trajectories
fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(111, projection='3d')
ax.set_title("Marker trajectories", fontsize=16)
ax.set_xlabel("$x$ [mm]", labelpad=12)
ax.set_ylabel("$y$ [mm]", labelpad=12)
ax.set_zlabel("$z$ [mm]", labelpad=12)
ax.scatter3D(markers[0][:, 0], markers[0][:, 1], markers[0][:, 2])
ax.scatter3D(markers[1][:, 0], markers[1][:, 1], markers[1][:, 2])
plt.show()

# Plot marker distance
fig_dist = plt.figure
delta = markers[1] - markers[0]
dist = LA.norm(delta, axis=1)
plt.plot(dist)
ax_dist = plt.gca()
ax_dist.set_title("Marker distance", fontsize=16)
ax_dist.set_xlabel("Frame number", labelpad=12)
ax_dist.set_ylabel("distance [mm]", labelpad=12)
plt.show()
```

read_tool.py

```
# Import section
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Data file
data_file = "new_praktikum_02.tsv"

# Read data
marker_data = pd.read_table(data_file, sep='\t', engine='python',
on_bad_lines='skip')

# Filter data
markers = marker_data.loc[(marker_data["State"] == "OK")]

tool = markers[["Tx", "Ty", "Tz"]].to_numpy()
err = markers[["Error"]].to_numpy()

# Print maximum error
print("Max measurement error:")
print(np.max(err))

# Plot error histogram
fig_err = plt.figure
plt.hist(err, 20)
ax_err = plt.gca()
ax_err.set_title("Error Histogram", fontsize=16)
ax_err.set_xlabel("Error [pixel]", labelpad=12)
ax_err.set_ylabel("# counts", labelpad=12)
plt.show()

# Plot marker trajectories
fig_traj = plt.figure
ax_traj = plt.axes(projection="3d")
ax_traj.scatter3D(tool[:, 0], tool[:, 1], tool[:, 2])
ax_traj.set_title("Marker trajectories", fontsize=16)
ax_traj.set_xlabel("$X$ [mm]", labelpad=12)
ax_traj.set_ylabel("$Y$ [mm]", labelpad=12)
ax_traj.set_zlabel("$Z$ [mm]", labelpad=12)
plt.show()
```