A Motion Compensation Method with an UR5-Robot in Medical Applications

Nasser Attar, Mirko Schimkat, Konstantin Stepanow and Arvind Ramasamy

***Abstract*—This paper presents a method, where a robot is used to precisely compensate the body movements of a human patient during medical treatments. The method uses visual tracking to determine the position and orientation of the relevant body parts of the patient and afterward, based on the received tracking data, the robot is moving a specific medical device, such that the device attains a commanded spatial relation to the relevant body parts of the patient. In the first section of this paper, it is shown, why such a method is useful and which fields of application exists. Furthermore, a specific medical scenario is presented, which will be used in this paper to demonstrate the motion compensation method. The second section presents the used robot and his kinematic model. The third section introduces the visual tracking system and the calibration and tracking algorithms to estimate the posture of the patient's body. The fourth section considers the constraints and conditions the motion of the robot is subject to and derives appropriate path planning algorithms to fulfill them. The fifth section shows how the actual motion compensation is assembled from the previous contents and the last section summarizes the results.**

I. Introduction

In a number of medical applications it is necessary to place a medical device next to a human patient in such way that the device keeps exactly a predetermined position and orientation relative to the body of the patient. One example, is the placement of a radiation coil in space, next to the head of a patient, such that it can radiates a specific part of the patient's head. Naturally, a human patient is moving his body parts, unconsciously, and therefore the considered device must follow his movements to maintain the commanded relation. Obviously, a robotic operator is suitable for executing this task. A tracking system measures the movements and posture of the patient and it does the same for the medical device. The tracking system sends this data to a PC, which processes the data, to determine which movements the manipulator of the robot must execute to maintain the specified spatial relation between the medical device and the patient. Then the PC sends movement commands to the robot and the robot is reacting with a movement of his manipulator, which carries the medical device. Of course, care must be taken that the robot or the device is not colliding with the patient. To demonstrate the above explanations the scenario shown in Fig. 1 is considered. The robot and the tracking system are shown which have their own coordinate systems attached to them. In this scenario, the only important body part of the patient is his head. The coil, which is carried by the manipulator of the robot, shall keep a predetermined posture relative to the head. To express this posture, the head and the coil also have corresponding coordinate systems attached to them. Then the relative posture between

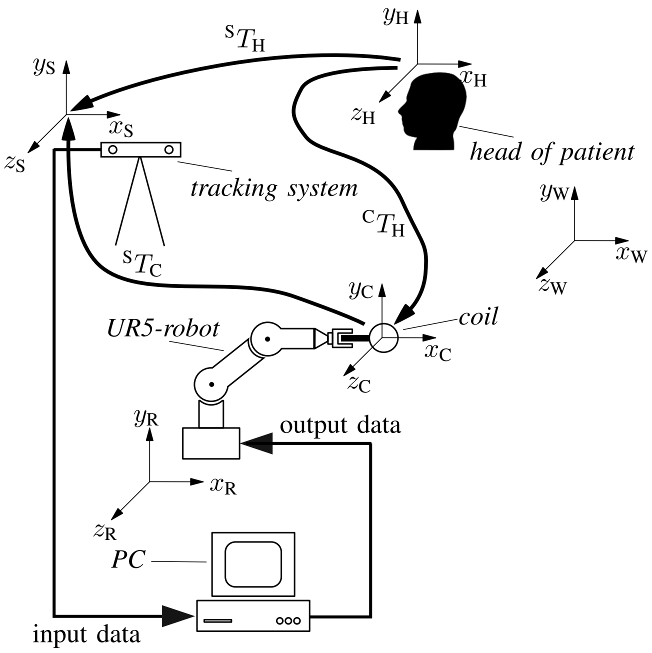


Fig. 1. Medical scenario, where the motion compensation method is used.

them can be expressed with the homogeneous transformation matrix $^\mathrm{H}T_\mathrm{C}$. Analogously, the *desired* posture between head and coil has the corresponding homogeneous transformation matrix $^\mathrm{H}T_\mathrm{C,g}$. The goal of the motion compensation method is to move the robot such that the relation

$$^\mathrm{H}T_\mathrm{C} =^\mathrm{H} T_\mathrm{C,g} \tag{1}$$

always holds. Of course, due to measurement errors of the tracking system and the finite precision of the robot, $^\mathrm{H}T_\mathrm{C}$ can only approximate $^\mathrm{H}T_\mathrm{C,g}$. The coordinate frames of the patient's head, tracking system, robot and coil have in each case a posture relative to the world coordinate frame, denoted by the axes triple $(x_\mathrm{W}, y_\mathrm{W}, z_\mathrm{W})$. The tracking system measures the relative postures of the head and the coil relative to the coordinate system of the tracking system $(x_\mathrm{S}, y_\mathrm{S}, z_\mathrm{S})$, in terms of the homogeneous transformation matrices $^\mathrm{S}T_\mathrm{H}$ and $^\mathrm{S}T_\mathrm{C}$. They serve as input for the PC, which processes them together with the posture of the tracking system and robot in the world coordinate system to output data for the robot. This data processing is done in MATLAB. The robot will then, based on the output data from the PC, move its manipulator, such that Eq. (1) is approximated as good as possible.

II. Tracking

The tracking of objects with the KINECT System is done in five different steps.

1) Taking the Infrared (IR) image and the depth image with the KINECT System
2) Tracking the fiducials in the IR image
3) Convert the position of fiducials, which is given in pixels to mm
4) Identify the fiducials with respect to the reference model of the tracked object
5) Create the transformation matrix

The KINECT System takes two different images. One infrared image and one image which represents the depth. While the values of the pixels in the infrared images are a decimal value between zero and one, the values of the depth images holds the actual depth in mm. In the end, we want to estimate the position of an object. This is done by placing fiducials on the object. This fiducials reflect the infrared light in a strong way, which appears in the infrared images as a bright dot. These dots can be tracked in the images and the position can be calculated. By knowing the distances between the fiducials on the object (stored in a reference model) and comparing these distances with the measured distances between the tracked points, we can determine the actual position of the object in the three dimensional space.

But as in any real system, we have to deal with distortions. There are two big main problems:

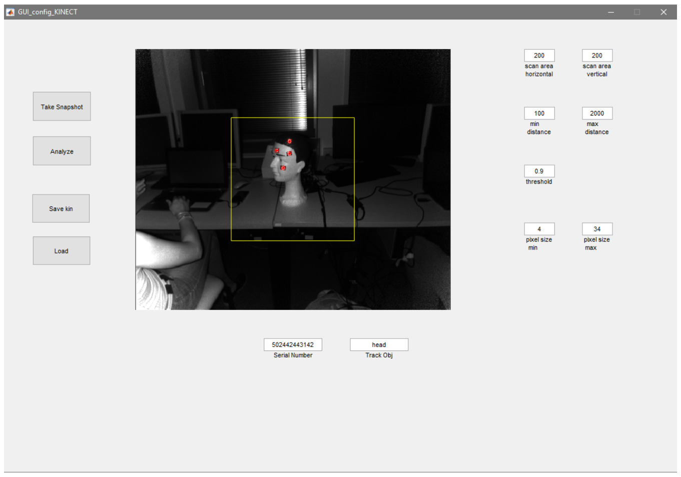
1) Not only the fiducials can appear as bright dots. Also light from the sun or reflections of the infrared light from other objects can appear as bright dots in the image.
2) The distance measurement, which is used for creating the depth image, sometimes has problems to measure the right distance. In this cases, the distance for that point is zero. This is mostly the point, when a material reflect the infrared light in a strong way, which is nearly always true for the fiducials.

To track only the fiducials and not some distortions, some filtering is done. To simplify the filtering, some parameters are set before the measurement starts:

- threshold for converting IR image from grayscale to black and white
- minimal distance between object and camera in mm
- maximal distance between object and camera in mm
- minimal size of the fiducials in pixel
- maximal size of the fiducials in pixel
- scan area, in which the object will appear

These settings can be set via a GUI, which also shows the scan area (in yellow) and the found fiducials. This is illustrated in Fig. 2. With these parameters, the following filtering steps are performed:

1) Converting the IR image from gray scale to black and white. This is done by interpreting every pixel which is below the brightness of the threshold as black and every pixel with a brightness over or equal to the threshold as white.



200 scan area horizontal
200 scan area vertical
100 min distance
2000 max distance
0.9 threshold
4 pixel size min
34 pixel size max
502442443142 Serial Number
head Track Obj

Fig. 2. The GUI with the preview of the KINECT System and the parameters.

2) Every white pixel which is to near to the camera or to wide from the camera is deleted (switched to black). For the actual distance, the depth image is taken.
3) Every white pixel outside of the scan area is turned to black.

These three steps are shown in Fig. 3, where step 1 is the original IR image, step 2 is the image converted to black and white, step 3 is the image, where the pixels which are to near or to wide away are deleted (note, that because of distortions in the depth image, the pixels in the lower left of the image are still there) and in step 4 all pixels outside of the can area are deleted. In the next processing step, the pixels are clustered. The clustering groups all pixel, which are connected to each other to one group (see Fig. 4a). For each cluster, the following values are calculated:

- The area size of the pixels
- The x and y coordinate (in pixels) of the middle of the pixel area
- The bounding box for the pixel area. This describes the smallest possible rectangle, which surrounds the pixel area, without crossing a pixel.

Now, for each cluster, the area size is checked, and if the area is in the specified region, the cluster is identified as a fiducial. As soon as the fiducial is identified, the x and y coordinates are stored in a vector. Because of the fact, that the depth image holds no information for the exact location of the fiducial, depth is calculated by the mean value of the depth of the pixels, which surrounds the fiducial. To simplifies this process, the bounding box is used, to get the pixels from the depth image, which are outside of the fiducial, but close to it (see Fig. 4b). In the last step from measuring the position of the fiducials, the x and y component of the fiducials are converted from pixels to mm with the help of the intrinsic matrix. Until this point, we have on the one side the reference model and on the other side normally four fiducials tracked fiducials, each with their x, y and z components. But we dont know, which tracked fiducials corresponds to which fiducial in the reference model. To calculate the translation and rotation of the tracked object in reference to the camera, we have to identify each tracked

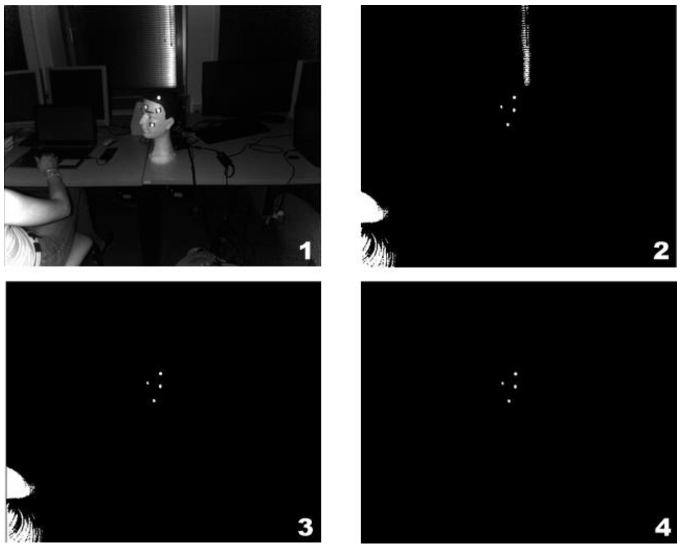
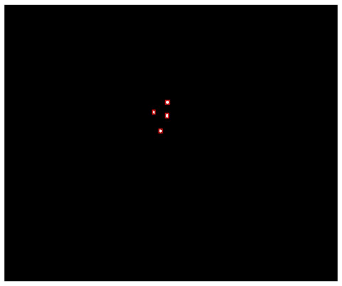
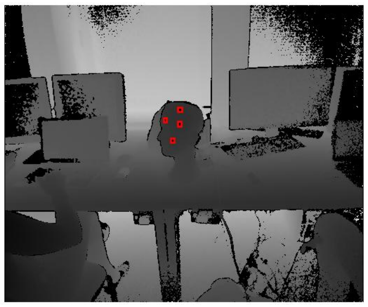


Fig. 3. The four steps of the filtering. 1: original IR image. 2: After converting to black and white. 3: Deleting all pixels which are to wide or to near from/to the camera. 4: Deleting all pixels outside of the scan area.



(a) Pixels in clusters. Red: Boundingbox of each cluster



(b) Depth image with bounding boxes around the fiducials.

Fig. 4. Identification of the bounding boxes of the fiducials.

fiducial with respect to the fiducials in the reference model. This is done by calculating the distances between all points (this is done with the reference model and with the tracked fiducials). To avoid problems with measuring errors, distances in the tracked system, which are to similar are sorted out. To identify all fiducials, at least 4 distances have to remain. By comparing the distances between the tracked fiducials and the fiducials in the reference model, we can create connections

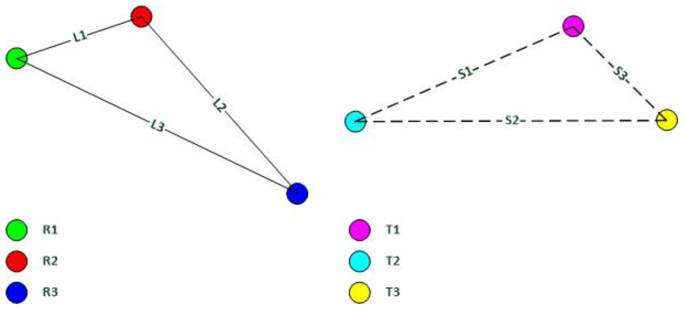


Fig. 5. Reference model (left) and tracked fiducials (right)

between the points and identify the fiducials. This is shown in Fig. 5 and Fig. 6 (but with only three fiducials and therefore three distances). For each distance from the reference system, the matching distance from the tracked system is searched. After that, a solution is searched which holds for each point of the reference system a point of the tracked system in a way, that the matching distances are made from the same points. The algorithm is pictured in Fig. 6. The first point from the reference system (green) is taken and it is stored, which points from the tracked system it can be (cyan and yellow). Then the algorithm searches for the next distance, where the point one (green) appears. Then it checks if the yellow or the cyan point appears in the distance from the tracked system. The point which appears, corresponds to the first point. This is done for all three points from the reference system. The last process is to generate the transformation matrix out of the tracked points. To define a translation and rotation of the object, we first have to define a coordinate system for the object. The coordinate system is based on the points from the reference system, where the first point is used as base. The three unit vectors are created with the formulas below:

$$\vec{e}_x = \frac{\vec{p}_2 - \vec{p}_1}{|\ \vec{p}_2 - \vec{p}_1\ |} \tag{2}$$

$$\vec{s} = \langle \vec{e}_x, \vec{p}_3 - \vec{p}_1 \rangle \vec{e}_x \tag{3}$$

$$\vec{e}_y = \frac{\vec{p}_3 - \vec{p}_1 - \vec{s}}{|\ \vec{p}_3 - \vec{p}_1 - \vec{s}\ |} \tag{4}$$

$$\vec{e}_z = \frac{\vec{e}_x \times \vec{e}_y}{|\ \vec{e}_x \times \vec{e}_y\ |} \tag{5}$$

This simple definition makes the calculation of the transformation matrix for the tracked points really easy. By defining the

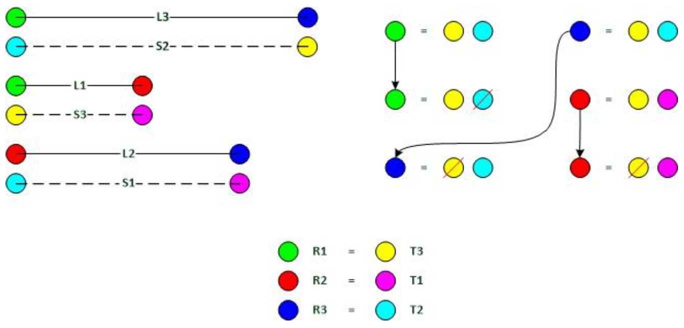


Fig. 6. Algorithm for identifying the points from the tracked system

point $\vec{p}_1$ from the tracked system as base, the same formula can be used for the tracked points as it was used for the reference system.

III. Conclusion

The conclusion goes here.

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.