A real-time hand and object tracking system with low cost consumer-grade hardware

Oliver Kalbfleisch
University of Applied Sciences Cologne
Gustav-Heinemann-Ufer 54
50968 Cologne, Germany
oliver.kalbfleisch at smail.th-koeln.de

Abstract— **Professional-grade hand and object tracking systems are usually the result of long years of combining specialized hardware and software components to achieve a tracking solution. This comes at the cost of a high technical system complexity and a corresponding high price-point. The underlying technical concepts of these systems can be boiled-down to rather simple concepts. Therefore this paper will evaluate if it is possible to build a hand and object tracking system on low cost consumer-grade hardware components. The most common approach to spatial tracking, a stereoscopic camera setup, is chosen as the system base. To keep system cost low, a set of Raspberry Pi's with the matching camera is used for image acquisition and processing. These processing units are linked to a Master" unit which takes care of stereoscopic depth calculations, the position value processing and the final rendering of a hand and object model in digital space. Several colored markers are assessed for the prototype application and compared in terms of tracking precision. A relatively new inverse kinematic framework as well as a novel method for input value filtering is also applied and evaluated. The final results show that it is possible to achieve a tracking system solution with the used hardware components that has a suitable tracking update frequency and precision. The resulting prototype is good base for further research in terms of processing optimization.**

I. INTRODUCTION

The standard interface between human and computer has for long years been mouse and keyboard. But with the advance of technology, new interfacing methods were developed in the last few years. Touch technology for interfacing with mobile devices and desktop computers has become a reliable technology and has been integrated into our everyday lives. Advances in capabilities of CPU as well as GPU hardware has built a foundation for the use of advanced augmented- and virtual-reality technology. 3D and stereoscopic rendering can now be accomplished even on mobile devices(with some limitations). For an intensely immersive experience, VR goggles are used to explore digitally created worlds. With this level of immersiveness, a touch device or just a mouse and keyboard setup is rather hindering for the user experience. The logical and more natural way of interfacing with our digital technology would be by utilizing the built-in human control elements that we carry around with us every day.

The human hands offer a form of control element that provides a large number of "*Degrees-of-Freedom*" (DOF's) and furthermore adds the possibility of combining these witch a tactile component. The first more humble attempts of solving this problem already began in the early 1980's, where a lack of computational power and low-res imaging hardware made more complex systems impossible. These systems were mostly aimed at registering simple hand gestures like pointing as an interface method and were not able to reconstruct full hand positions in real-time[1]. Modern day elaborate systems for hand tracking use infrared or stereoscopic cameras to achieve a more performant tracking result by utilizing the calculation capabilities of standard consumer computers. These systems are mostly specialized for the task they are doing and are therefore rather expensive to attain.

This paper assesses, if it is possible to create a real-time hand and object tracking system based on easily accessible consumer grade hardware and open-source software. The system should provide an experience for the user that is as natural as possible in terms of tracking precision and the components should not hinder the motion capability of the hand. Furthermore, the system should aim at being comfortable in terms of design, as a cumbersome system would not be used in everyday life.

II. RELATED WORK

There have been several different approaches on solving the tracking problem of the human hands with their maximum of 27 DoF's each. The systems that try to achieve this have to encounter several difficulties. Self-occlusion plays a great role in the system design, especially when working with only one tracking optical system for reduced complexity. Also these systems need to maintain a certain amount of processing speed for achieving a fluent reproduction of the captured motion. This demands the capability of processing large amounts of data in very short intervals. Most prototype testing for the described systems is done in an controlled environment where the background is known. For a more widespread use, these systems need to be capable of registering the hand on an unrestricted background, which may have a wide range of patterns, color and lighting differences. Human hand motion itself is quite fast, which results in a higher frame rate demand for the tracking cameras.

Early hand tracking systems, dating back to the 1970's [2], [3], relied on glove systems which used different forms of

electronic sensors to measure finger and hand positioning. Newer systems made use of the developments in electronic part size and reduced the mounted sensors sizes [4], [5]. Further readings on sensor based hand tracking systems can be found in [6], [7].

With more processing capabilities optical tracking approaches were also evaluated[8], [9], [10]. These systems take either monocular or stereoscopic images of the hand and process these afterwards to find the hand position. To get a hand tracking from pure visual data, these approaches all use colored marker on gloves. The approaches vary from only colored fingertips [9] of the glove to fully patterned gloves [10] where a known pattern is used for pose estimation. Data retrieved from these markers are then compared against databases of recorded hand position configurations to find a nearest match. The currently most used type of system for tracking hardware is a setup of a combination of RGB and infrared sensors for imaging and depth measurement (RGB-D) [11], [12]. These RGB-D camera systems are capable of suppling image data at up to 90fps. The post processing of this data needs more sophisticated image analysis or neural network processing to figure out feature points of the tracked object from the supplied image data [13], [14]. Such an approach of combining RGB-D Data and the usage of *CNNs* (Convolutional Neural Networks) to estimate hand pose and render a correct digital representation of the hand is shown in [15]. Their approach uses two subsequently applied *CNNs* to localize the hand and regress 3D joint locations. The first *CNN* estimates the 2D position of the hand center in the input. The combined information of the hand position together with the corresponding input depth value is used to generate a normalized cropped image. This image is then fed into a second *CNN* to regress relative 3D hand joint locations in real time. Pose estimation is furthermore refined by using a kinematic pose tracking energy to achieve more accuracy, robustness and temporal stability. They also introduce a new photo-realistic data-set that uses a merged reality approach to capture and synthesize large amounts of annotated data of natural hand interaction in cluttered scenes for CNN training data.

III. PHYSIOLOGICAL STRUCTURE OF THE HUMAN HAND

Lee and Kuni [16] described the human hand as "*an articulated structure with about 30 degrees of freedom [which] changes shape in various ways by its joint movements.*" All of the hand parts are connected to at least one neighboring part via a *joint*. The *joints* affect the position of the connected part. To describe the movement of the hand parts, we can use the rotation angles of the *joints* to correlate to a specific position. To do so, we define a local coordinate system for each of the exiting hand joints. Through these coordinate systems, we achieve a sequence of rotations in the local coordinate systems of the joints. Such a sequence can then be used to describe a specific movement and/or position of a part. Not all of the joints in the human hand have equal *degrees of freedom*. Their functionality can be classified in the amount of DOFs [17], where 1 DOF *joints* can perform a *flexion* or *twist* in one direction. The 2 DOF *joints* can perform *flexion* movement in more than one direction and the 3 DOF *joints* can perform simultaneous *directive* and *twist* movements.

Regarding the human hand, each finger sums up to 4 DOF's and the thumb to 5 DOF's. Also considering 6 DOF's for the rotation and position of the whole hand, the total sum add up to 27 DOF's.

A. CONSTRAINTS IN HAND MOTION

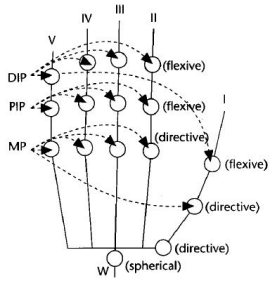

Fig. 1. Name of the human hand joints and their constraint types [16]

A full usage of all the declared DOF's would lead to a large amount of possible combinations. Since the hand is not only made up of bones but also muscles and the skin, we can impose some constraints [18], [19] to the movement of the joints. Ling, Wu and Huang [20] proposed following classification for the constraints. The **Type I** and **Type II** constraints rely on the physiological and mechanical properties of the human hand. **Type III** constraints are results of common and natural movements. Therefore these constraints will be differing form person to person. However, these movements are to a certain degree similar for everyone. Accordingly, a broad grouping can be applied. For the mathematical description of the **Type I** and **Type II** constraints, the following inequalities can be used:

$$0^\circ \leq \Theta_{MP_{flex}} \leq 90^\circ \tag{1}$$

$$0^\circ \leq \Theta_{PIP_{flex}} \leq 110^\circ \tag{2}$$

$$0^\circ \leq \Theta_{DIP_{flex}} \leq 90^\circ \tag{3}$$

$$-15^\circ \leq \Theta_{MP_{abduct/adduct}} \leq 15^\circ \tag{4}$$

For the middle finger, a further specific constraint is applicable as the middle fingers MP joint normally does not abduct and adduct much. Therefore, an approximation is applicable, resulting in the removal of 1 DOF from the model:

$$\Theta_{MP_{abduct/adduct}} = 0^\circ \tag{5}$$

The same behavior can be seen in the combination of hand parts labeled W (the connection point between hand and lower arm). This approximation also eliminates one DOF on the connected thumb:

$$\Theta_{W_{abduct/adduct}} = 0^\circ \tag{6}$$

Since the DIP, PIP and MP joints of our index, middle, ring, and little fingers only have 1 DOF for flexion, we can further assume that their motion is limited to movement in one plane. The **Type II** constraints can be split into *inter-finger* and *intra-finger* constraints. Regarding *intra-finger* constraints between the joints of the same finger, human hand anatomy implies that to bend the DIP joints on the specific finger, the corresponding PIP joints of that finger must also be bent. The approximation for this relation [21] can be described as:

$$\Theta_{DIP} = \frac{2}{3}\Theta_{PIP} \tag{7}$$

Inter-finger constraints can be imposed between joints of adjacent fingers. *Inter-finger* constraints describe that the bending of an MP joint in the index finger forces the MP joint in the middle finger to bend as well. When combining the constraints described in the above equations, the initial number 21 DOF's of the human hand can be reduced to 15. Inequalities for these cases, obtained through empiric studies, can be found in [16].

IV. Kinematics

Kinematic systems contain so-called *kinematic chains*, which consist of a *starting point* or *root*, kinematic elements like *joints*, *links* and an *endpoint*, also called *end-effector*. Applied to the human hand, the whole hand structure represents the kinematic system. This system contains several *kinematic chains*, namely the fingers of the hand with the fingertips being the *end effectors* of each chain. Each of these components has it's own DOF's which can be described mathematically. As hand movement starts, the states of the kinematic chains begin to change. *Joint* angles and *end-effector* positions are modified until the end position is reached. To represent the new position and angle values of our physical hand with a kinematic system, two major paths for achieving a solution can be taken. The *Forward Kinematics*(FK) approach uses the knowledge of the resulting angles and positions after the application of known transformations to the kinematic chain. The resulting new positions of the *joints* and *links* between the *root* and the *end-effector* are used to solve the problem of finding the *end-effector's* position. The advantage of an FK solution is that there is always an unique solution to the problem. In consequence, this approach is commonly used in the field of robotics where the information on the chain elements is easily available. The tracking of the human hand and all of its chain components is rather complicated. Therefore a solution which takes a known position of the *end-effector* and calculates the parameters for the rest of the chain is more desirable. The concept of *Inverse Kinematics*(IK) takes the reversed approach in comparison to the FK principle. Instead of knowing the states of the chain elements and calculating the resulting position of the *end-effector*, it takes the position of the *end-effector* and trys to retrieve the possible states of the other chain elements. In contrary to having a unique solution with the FK approach, the IK approach can end at the point of not finding a suitable solution. Established solution for solving inverse kinematic problems mostly rely on matrix calculations. These methods include solving via *Jacobian Inverse* [22], [23], *Jacobian transpose*, *Jacobian pseudo inverse* [24], [25], *Damped Least Squares* [26], [27] and *Newton Methods*. All of these methods have a rather high computational load and some can suffer from singularities, making a solution not obtainable.

A more recent approach in the kinematics field is the *FABRIK* algorithm proposed by Aristidou and Lasenby [28]. The *FABRIK* algorithm does not depend on these matrix operations as it solves for the position of a point on a line to retrieve the new joint positions. This is done in an forward and also inverse solving approach, iterating these steps until the calculated position converges towards the target position defined by the tracking data.

The chain joints are denoted as $\mathbf{p}_i$ with the distance $\mathbf{d}_i$ being $|\mathbf{p}_{i+1} - \mathbf{p}_i|$. The target point for the end effector is denoted as $\mathbf{t}$. The joint positions are taken from either a previous iteration or from an initial calibration. But before calculations can begin, the algorithm has to check whether the intended target point $\mathbf{t}$ is reachable for the end effector. This is done by measuring the distance between the root of the kinematic chain and the target point $\mathbf{t}$. This value is then compared with the sum of the distances $\mathbf{d}_i$.

$$dist_{t,d_1} < \sum_{k=1}^{i} d_i \tag{8}$$

If the summed distance is greater, then the target $\mathbf{t}$ is within the reach of the system and the calculation can continue, otherwise the calculation has to be aborted and the error has to be manged otherwise. Assuming this requirement to be met, we can now begin with the first calculation. The inverse calculation step is the first step. The calculation is started at the end effector, moving to the root of the chain. Therefore we assume that the new position $\mathbf{p}_n^{'}$ with n=m,...,1 is equal to $\mathbf{t}$.

$$\mathbf{p}_n^{'} = \mathbf{t} \tag{9}$$

From this new point, we can construct a line that goes through $\mathbf{p}_n^{'}$ and $\mathbf{p}_{n-1}$.

$$A = \mathbf{p}_n^{'} B = \mathbf{p}_{n-1}\mathbf{l}_{n-1} = \overline{AB} \tag{10}$$

The resulting position of the new $\mathbf{p}_{n-1}^{'}$ point is located on this line with the distance of $\mathbf{d}_{n-1}$ from $\mathbf{p}_n^{'}$.

$$\mathbf{p}_{n-1}^{'} = \mathbf{p}_n^{'} + \left(\frac{\overline{AB}}{|\overline{AB}|} \cdot \mathbf{d}_{n-1}\right) \tag{11}$$

Consecutively, this is done with the remaining joints until the root joint is reached. This finishes the first halve of the iteration step. With the calculated positions, we now perform a forward calculation, starting from the root until we reach the end effector. Since the root of the system normally does not move from it's initial position, we have to reset the root joint to this value before starting to calculate the new positions of the subsequent joints. Analogous to the procedure in the inverse step, we construct the lines between

the points and determine the new position values of the joints. At this point, we can decide if the result position of the end effector is appropriate in comparison to the value of **t**. A simple threshold value for this case could be the position difference between these two points. Since this algorithm does not rely on the heavy-weight calculation operations, it converges much faster and in less iterations. Furthermore the incorporation of constraints to the calculation does not interfere with this values and is rather easy.

V. SYSTEM CONCEPTION

For the prototype construction, a seated use-case is chosen. The tracking volume that needs to be accomplished for a seated position is limited by our arm length. A tracking volume of about $100x100x100$ cm should be sufficient to track most of the movement. A seated use-case also benefits the camera system, as it needs to be calibrated for a certain depth to attain correct depth measurements. Instead of using one large dimensioned unit which takes care of all calculations, the image processing steps that have to be done on both stereo images anyway, are outsourced onto the Raspberry Pi's. The image processing can be done on the two Pi's in parallel, which cuts down processing time. Furthermore, the image data stays on the device and does not have to be sent to a main unit which would introduce network and data reading latency.
The two **Raspberry Pi 3 Model B** which are used as the controllers for the **Raspberry Pi Camera** are connected to the "Master" PC unit via a network switch and communicate over UDP protocol. The "Master" takes care of the stereoscopic calculations. The 2D positional data from the Pi "slaves" and the calculation results from the stereo image disparity is fed into the hand model running on the "Master". The model solution is then applied to the digital hand model and rendered.

A. CONSTRUCTION DETAILS

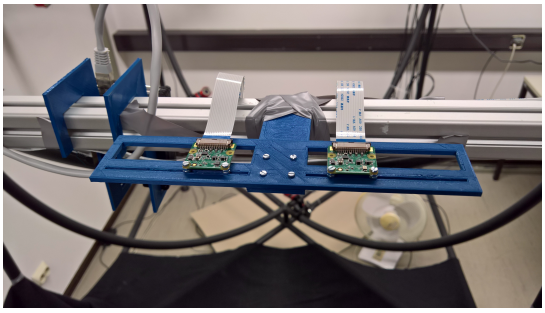

Fig. 2. 3D printed camera holder

A simple holder plate was constructed digitally and printed with a 3D Printer. Cut-outs in the mounting plate for the cameras provide the ability to mount these with two machine screws and move their end position along a fixed axis. This ensures that the cameras are mounted without axis offsets. It furthermore give the option to change the inter-axial distance of the cameras if needed. For the current setup, this distance will be set to 75 mm which corresponds to the inter-axial distance of the human eyes. As tracking features for the position calculation, colored markers for the fingers are chosen. To ensure a natural haptic feedback, glove based solutions are not valuable. A sixth color marker has to be positioned on top of the hand to serve as a global position reference. For the object tracking part, a *HTC Vive Marker* and *Lighthouse* is used.

B. IMAGE ANALYSIS ON THE RASPBERRY

For the image analysis part, *OpenCV3* is chosen. The prototype *C++* implementation running on the Raspberry's for tracking the specified color markers is comprised of an image acquisition stage, followed by image optimization and binary mask filtering. The resulting mask-filtered images are used for color contour finding and position calculation. To reduce image analysis times, a *region of interest (ROI)* is defined for each marker after their successful detection. First frame ROI calculation is based on a minimum *axis-aligned bounding box (AABB)* fitted onto the marker. The offset value from the image origin for the ROI is added to the bounding box values, resulting in a rectangle data-set containing the position of the offset top-left corner $\vec{\mathbf{tl}}$ and the offset **width** and **height** of the ROI rectangle. These values are taken to extract the sub-region of the consecutive at $\vec{\mathbf{tl}}$ with size of **width** and **height**. The $\vec{\mathbf{tl}}$ value is also saved as offset value $\vec{\mathbf{OffPos}}$.
The resulting rectangle has to be fitted onto the size boundaries of the full image frame by clamping the combination of position, width, height and offset to fit into the image boundaries. Should no marker be found in the defined ROI, the frame has to be dropped and the next frame should utilize the whole image data. To compensate for white balance shifting, a non-white, diffuse reflecting background should be used for the tracking space and the camera auto-modes should be turned off. Appropriate values for white-balance and exposure have to be determined at the initialization step of the system. A *Gaussian filter* is applied to the masks which acts as a low-pass filter for the image. After this step, an *erosion* and a *dilation* is applied to the image to further eliminate unwanted noise. On the cleaned binary masks, a search for the marker area is done. Under the assumption that all other parasitic objects have been removed from the image, the marker should be the largest area of positive (white) pixels in the mask frame. This area is taken as the desired tracking marker and the ROI is calculated.

C. MASTER SIDE CALCULATIONS

Figure 3 displays the work-flow on the "Master" side. The UDP data streams are processed on separate threads to retrieve the 2D positions. These are fed into the depth calculation, which measures the disparity between left and right side image to calculate a depth value [29], [30], [31]. The resulting 3D data points are then used to update the hand model and target points in 3D space. The IK algorithm takes the target points and tries to solve for their position. The tracked object data, which is not displayed here also

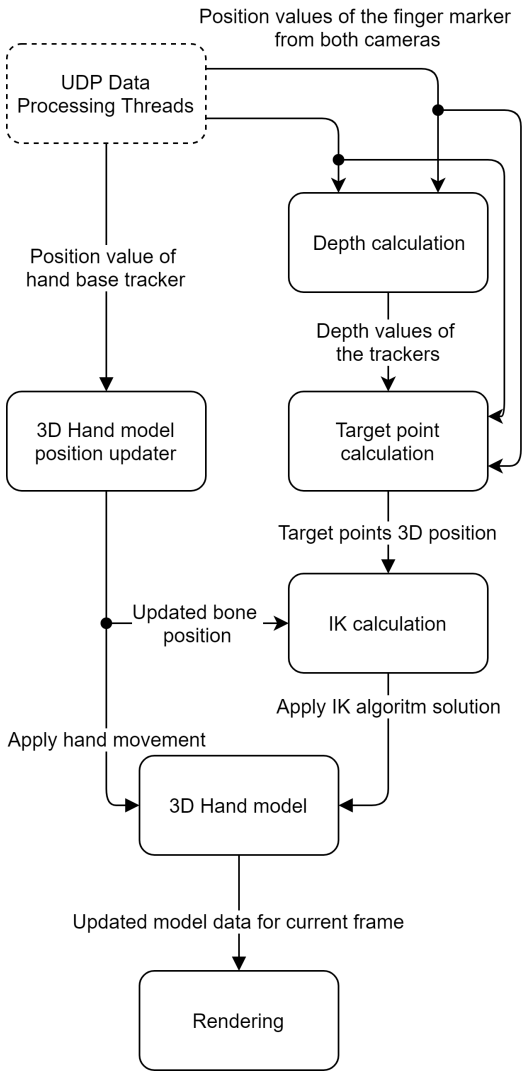

Fig. 3. Work-flow of the ”Master” side calculations

gets processed and the representing digital model position gets updated before the rendering step.

D. IK CALCULATION

For the inverse kinematics calculation, the *FABRIK* algorithm is used. It does not require complex matrix calculations and has a fast convergence rate. For the implementation of the algorithm on the client side, the *Caliko* Java framework [32] is used. For the inverse kinematics calculation to work properly, the updated bone positions and the calculated target points from the finger tracking are needed. For each finger, a kinematic chain is set up with the correct bone length. The length of each bone has to be measured manually for the first prototype. The movement of each bone in the kinematic chains has to be restricted by constraints to restrict the algorithms solutions to physiological possible hand poses.

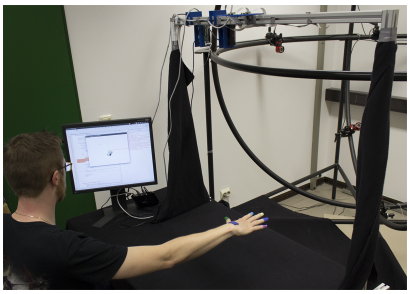

Fig. 4. Final system setup with cameras on construction profile, color markers and visualization.

VI. SYSTEM EVALUATION

A. CAMERA HARDWARE AND CONTROL

The image processing of the camera frames on the raspberry can be a significant bottleneck for the system. The current system setup is limited by the used camera framework at 30fps at a 640x480 resolution. This means that the image processing time, containing frame readout and downstream processing of the images, has to be done in about $1/30s$ or 33 ms to achieve ”real-time” processing. The cameras use a rolling shutter instead of a global shutter. Since these cameras are designed to be of easy use for the normal consumer, they lack means of external synchronization. Attempting the synchronization of the data via software further down the processing pipeline was therefore more reasonable. To synchronize the reading start point of both cameras, the system boots up to the point where all needed components are initialized. At this point, the program waits for a start message from the ”Master” system via UDP. The ”Master” system sends the start message at the end of it’s initialization phase via a UDP broadcast, ensuing that both Raspberry’s get the message at the same time. All automatic modes are turned of in the camera initialization phase and fixed values are loaded from a JSON file. To get the values for the JSON file, a calibration step needed to be implemented. With this step, the user-set calibration values can be written into a new JSON file and are directly loaded into the program after manual calibration is finished. Control over these parameters showed to be crucial for achieving a constant color separation. The threshold values values for the color tracking also needed to be initially calibrated. These values depend on the lighting situation and any change results in the need for recalibration. To simplify this step, a calibration function in the same fashion as the camera calibration was implemented.

VII. IMAGE PROCESSING PERFORMANCE

Initial investigation into code timings showed, that a bottleneck was the image optimization feature which applied an erosion an dilation to remove high-frequency noise in the image. This operation brought the time up to 100 ms processing time, when processing the whole frame area, making the algorithm not usable for ”real-time” application. Removing this feature for whole image scans brought a major speed up in processing time. A test with lower resolution than

640x480 showed that the calculation time can be reduced. This confirmed that the ROI idea would be feasible for further performance optimization. A test at a frame resolution of 320x240 pixels reduced the processing time for all five finger colors to below 10 ms. Performance measurement for stationary marker tracking consistency with the 6 needed color trackers was performed for a time period of 10.000 frames to get qualitative results. The resulting timing values showed to be sufficient to supply a tracking data stream of average 17 ms processing time with a sequential calculation approach.

A. COLOR ACCURACY

To achieve clean separation of the tracker colors and reduce unwanted noise, the defined color threshold ranges were kept as small as possible. Changes in lighting intensity and color temperature can cause the color of the trackers to shift. This results in a fluctuation of the calculated tracker positions. A broader color value range for the tracker colors at the calibration stage showed to be more beneficial for the system. System testing with 5 different colored markers showed, that tracking for colors outside of the color range of the human skin tones (green, blue) is rather unproblematic. The color markers falling into the color space of possible light reflection from skin may cause problems, depending on the lighting situation of the setup. Skin reflections will cause the threshold area of the marker to enlarge and therefore produce wrong positional data. As the final colors, a mixture of blue and green color tones together with red tones in the purple and magenta section were chosen. A set of colored shrink tubing was first evaluated as marker material. It showed to be acceptable as marker material in terms of application onto the finger. The downside of the material showed to be a limited availability of colors and those available had the aforementioned skin color problems. Acrylic paint in comparison is available in many color variations, making it possible to stay outside of the skin tone color ranges for the marker colors. It showed to have the benefits of being easily to apply to the finger. The finger coating is dry in under a minute after application. Adaption for finger size is automatically included in the application process. The system therefore uses acrylic paint as the marker material for the fingers. For the hand top marker, a piece of cellular rubber was used.

B. ROI SIZE

The implemented ROI feature speed up the whole system calculations, once the colors are found. Before this point, the system scanned whole frames to find colors which took more time than the much smaller ROI regions. Empirical evaluations showed, that for the used tracking space and a 640x480 camera resolution, a ROI region size offset of 40 px in x and y directions produces the best results in terms of consistency. Smaller region offsets caused the system to loose tracking for faster hand motion, resulting in system slow-down until the tracking recovered. Larger ROI's produce showed to produce more consistent tracking results, but come at the cost of higher calculation time.

Fig. 5. Finger marked with suitable acrylic colors

C. DEPTH MEASUREMENT ACCURACY

The accuracy of the camera system is limited by the number of pixels in relation to the camera view angles described in [29]. For the system setup of 640 px image width and a horizontal view angle of $62.5°$, the systems depth accuracy $\Delta\varphi$ is equal to $0.0977\frac{1°}{pixel}$. The system error for a $D = 100cm$ would therefore result in about $2cm$ of possible depth error. To measure the real system depth measurement accuracy, the camera rig was aligned horizontally and fixed to a test bench. A large sized colored marker (75 mm x 50 mm) was positioned at altering distances from the camera rig along the it's central axis. Measurements were started at 100 cm distance and were decremented in steps of 5 cm until 20 cm in front of the camera rig. The measured values showed deviations of up to 5% from the correct value. Therefore, a correcting function [33] was applied for the depth calculation. An exponential trend line fitted to the measured results represents the the function needed to fulfill the following equations:

$$D = k * x^z \tag{12}$$

with K being:

$$k = \frac{Bx_0}{2\tan(\frac{\varphi_0}{2} + \phi)} \tag{13}$$

and **x** the disparity in pixels. The ϕ term in the equation above is used as a compensation for possible alignment errors. The calculated values for the system are $k = 4543.3$ and $z = -1.035$. With the utilization of the correction function, the accuracy of the depth measurement can on average be reduced to below 2%.

VIII. DATA FILTERING

Measurements of the base jitter in the data for the finger markers showed that the systems can only supply a certain stability of position tracking. This causes fluctuations in the position results. Jittering of position values also causes the height calculation algorithm to generate incorrect height values. To get more stable results, a filtering of the incoming data-set in several steps with the *1 Euro filter* presented by Casiez et al. [34] needed to be applied. It uses a first order low-pass filter with an adaptive cutoff frequency to

filter noisy signals for high precision and responsiveness. The filter was chosen, because of a relative simple and easy implementation as well as an uncomplicated setup and tuning process. It also produces faster and better results in comparison to the normally used *Kalman filter* [35], *moving average filter* or *low-pass filters and exponential smoothing filters* [36].

IX. Inverse kinematics algorithm

Figure 6 shows debugging representations of the used kinematic structure in the *Caliko* framework.

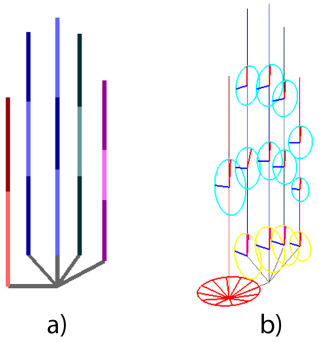

Fig. 6. Hand model representation of the used IK Framework: a) Kinematic chains and base structure b) hand model with visualized movement constraints

It differentiates between base-bone constraints and normal bone constraints. The constraints can be applied with either a global or a local reference axis. The reference axis can the be used for the specific type of constraint. The framework supports hinge type constraints (1 DOF) and rotor constraints (2 DOF). For the rotor constraints only circular curve limitations can be defined. This is a downside of the framework as a parabolic curve description for a rotor-based constraint would better fit the movement capabilities of the fingers.

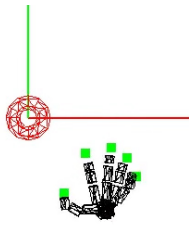

Fig. 7. Digital hand and tracker model used for visualization

X. Grabbing test

To test the tracking limitations of the system, a grabbing test with the tracked *HTC Vive* marker and the tracked hand was done. Before the test was started, it was ensured that the infrared signals from the object tracker do not interfere with the camera images for tracking. The built-in infrared filters of the camera manage to filter out the signals, as no image distortion was visible. The test chore was to pick up the tracked marker based on the position of the marker representation in digital space. A precise initial position calibration for the object tracker is done to ensure matching real and digital positions. The tracking limitations of the system showed when bending the fingers around the marker to pick it up. The bending causes the fingers to be occluded from the cameras and no new position can be tracked. This occlusion problem results from the positioning of the cameras. The top-down view limits the movement areas of the hand where all fingers markers can be tracked. The sixth hand position marker is not affected from this occlusion problem as long as the hand is not turned around. For the system to handle grabbing movements correctly, the cameras would have to be positioned at a different angle than the top down-view or a further pair of cameras recording from a different angle has to be added.

XI. Future work

The system evaluation showed that the principal functionality of the tracking system can be established with the selected component. One major point of improvement that could be applied to the system would be improvements to the used camera control framework. At the current state, a max frame rate of only 30 FPS readout is reachable while the hardware is theoretically able to supply up to 90 FPS readout. Large parts of the processing time on the Raspberry side are accounted for by costly image optimization like blurring or transformations. These operations on whole images are classical candidates to be outsourced onto the GPU of the Raspberry for faster computation. A conversion of the existing *OpenCV* code into the assembler-like code for the raspberry GPU should be a future goal. The calculation results for the depth values were already improved through the correction formula. To get even better results, a measurement session with more measurement points should be done to improve the values furthermore. The color threshold which is done by *OpenCV* could also be optimized to threshold all colors in a single run instead of running an operation for each color. This could also be candidate for GPU outsourcing. In the current state, a lot of values have to edited manually and are only visible on the command line. A graphical user interface should be build to ease the access. The hand data that is used for prototyping is still hard coded and only manually configurable. A calibration procedure for the system as well as a suitable data format for representing and translating these values for the IK algorithm is still to be done.

XII. Conclusion

The constructed prototype showed, that it is possible to construct a basic hand and object tracking system with consumer grade hard and software. The hardware capabilities of the used *Raspberry Pi* showed to be able to supply enough processing power to run live image processing operations. A downside of the used camera hardware is that the used sensor utilizes a rolling instead of a global shutter. This makes the synchronization of the two cameras for the stereoscopic

setup harder. Furthermore, the cameras were not intended to be used in such a scenario, therefore they lack a frame synchronization feature. The only way of achieving a form of synchronization is on the software side. At the selected resolution of 640x480 pixels, the system is able to track 6 color markers at around 50 fps max. The selection of the usable color markers is rather limited. Colors from the orange, yellow and red color spectrum showed to be rather difficult to track. The reflections that can occur when the hand is moved inside the tracking space can fall into these color ranges. This introduces an unwanted error in the marker position calculation. Shrink tubing, although being easy to adapt to finger sizes showed to have the downside of not being available in the needed tracking colors. The selected application of acrylic paint to the fingers as color markers has the benefit of being available in a wide variety of colors. It is also the least restricting type of all used markers in terms of haptics. The used inverse kinematics framework is a good starting point for the display of the digital hand model. It does lack some features like inter-finger and parabolic constraints which would better match the final result to the real world hand position. The grabbing test showed the limitations of the prototype. The system provides a stable object tracking performance even when partially occluding the object tracker with the hand. The hand-tracking can provide a acceptable tracking if all markers are visible. Certain movements of the hand can cause the finger markers to become occluded. Especially when grabbing objects, this can lead to a tracking loss and degrade tracking performance.

REFERENCES

[1] Richard A. Bolt. Put-that-there. In James J. Thomas, editor, *Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, pages 262–270, New York, NY, 1980. ACM.

[2] Thomas A. DeFanti and Daniel J. Sandin. Final report to the national endownment of the arts.

[3] Gary J. Grimes. Digital data entry glove interface device, 1983.

[4] T. Kuroda, Y. Tabata, A. Goto, H. Ikuta, M. Murakami, et al. Consumer price data-glove for sign language recognition. In *Proc. of 5th Intl Conf. Disability, Virtual Reality Assoc. Tech., Oxford, UK*, pages 253–258, 2004.

[5] Jose L. Hernandez-Rebollar, Nicholas Kyriakopoulos, and Robert W. Lindeman. The acceleglove. In Tom Appolloni, editor, *ACM SIGGRAPH 2002 conference abstracts and applications*, page 259, New York, NY, 2002. ACM.

[6] L. Dipietro, A. M. Sabatini, and P. Dario. A survey of glove-based systems and their applications. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(4):461–482, 2008.

[7] D. J. Sturman and D. Zeltzer. A survey of glove-based input. *IEEE Computer Graphics and Applications*, 14(1):30–39, 1994.

[8] Florin Duca, Jonas Fredriksson, and Morten Fjeld. Real-time 3d hand interaction: Single webcam low-cost approach. In *Proceedings of the Workshop at the IEEE Virtual Reality 2007 Conference; Trends and Issues in Tracking for Virtual Environments*, pages 1–5, 2007.

[9] Jonas Fredriksson, Sven Berg Ryen, and Morten Fjeld. Real-time 3d hand-computer interaction. In Konrad Tollmar, editor, *Proceedings of the 5th Nordic conference on Human-computer interaction building bridges*, page 133, New York, NY, 2008. ACM.

[10] Robert Y. Wang and Jovan Popović. Real-time hand-tracking with a color glove. In *ACM transactions on graphics (TOG)*, volume 28, page 63, 2009.

[11] Zhengyou Zhang. Microsoft kinect sensor and its effect. *IEEE Multimedia*, 19(2):4–10, 2012.

[12] Intel Corporation. Inter realsense technology, 2018.

[13] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake, editors. *Real-Time Human Pose Recognition in Parts from Single Depth Images*, 2011.

[14] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In Jesse Hoey, editor, *Proceedings of the British Machine Vision Conference 2011*, pages 101.1–101.11, Durham, 2011. BMVA Press.

[15] Franziska Mueller, Dushyant Mehta, Oleksandr Sotnychenko, Srinath Sridhar, Dan Casas, and Christian Theobalt. Real-time hand tracking under occlusion from an egocentric rgb-d sensor. 2017.

[16] Jintae Lee and T. L. Kunii. Model-based analysis of hand posture. *IEEE Computer Graphics and Applications*, 15(5):77–86, 1995.

[17] James Urey Korein. *A geometric investigation of reach: Zugl.: Pennsylvania Univ., Diss. : 1984*. ACM distinguished dissertations. MIT Press, Cambridge, Mass., 1985.

[18] Norman Badler, Kamran Manoochehri, and Graham Walters. Articulated figure positioning by multiple constraints. *IEEE Computer Graphics and Applications*, 7(6):28–38, 1987.

[19] V. I. Pavlovic, R. Sharma, and T. S. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):677–695, 1997.

[20] John Lin, Ying Wu, and T. S. Huang. Modeling the constraints of human hand motion. In *Proceedings, Workshop on Human Motion*, pages 121–126, Los Alamitos, Calif, 2000. IEEE Computer Society.

[21] Hans Rijpkema and Michael Girard. Computer animation of knowledge-based human grasping. In James J. Thomas, editor, *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 339–348, New York, NY, 1991. ACM.

[22] David E. Orin and William W. Schrader. Efficient computation of the jacobian for robot manipulators. *The International Journal of Robotics Research*, 3(4):66–75, 1984.

[23] W. Wolovich and H. Elliott. A computational technique for inverse kinematics. In *The 23rd IEEE Conference on Decision and Control*, pages 1359–1363. IEEE, 1984.

[24] Gene Golub and William Kahan. Calculating the singular values and pseudo-inverse of a matrix. *ISAM*, 2(2):205–224, 1965.

[25] Wolfgang Dahmen and Arnold Reusken. *Numerik für Ingenieure und Naturwissenschaftler*. Springer-Lehrbuch. Springer-Verlag Berlin Heidelberg, Berlin Heidelberg, zweite, korrigierte auflage edition, 2008.

[26] Charles Wampler. Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1):93–101, 1986.

[27] Yoshihiko Nakamura and Hideo Hanafusa. Inverse kinematic solutions with singularity robustness for robot manipulator control. *Journal of Dynamic Systems, Measurement, and Control*, 108(3):163, 1986.

[28] Andreas Aristidou and Joan Lasenby. Fabrik: A fast, iterative solver for the inverse kinematics problem. *Graphical Models*, 73(5):243–260, 2011.

[29] Jernej Mrovlje and Damir Vrancic. Distance measuring based on stereoscopic pictures. *9th International PhD Workshop on Systems and Control: Young Generation Viewpoint*, 2008.

[30] Yasir Dawood Salman, Ku Ruhana Ku-Mahamud, and Eiji Kamioka, editors. *Distance measurement for self-driving cars using stereo camera in Zulikha*, 2017.

[31] Holger Tauer. *Stereo-3D: Grundlagen, Technik und Bildgestaltung*. Fachverlag Schiele & Schoen, 2010.

[32] Alastair Lansley, Peter Vamplew, Philip Smith, and Cameron Foale. Caliko: An inverse kinematics software library implementation of the fabrik algorithm. *Journal of Open Research Software*, 4(1), 2016.

[33] Manaf A. Mahammed, Amera I. Melhum, and Faris A. Kochery. Object distance measurement by stereo vision. *International Journal of Science and Applied Information Technology (IJSAIT)*, 2, 2013.

[34] Géry Casiez, Nicolas Roussel, and Daniel Vogel. 1 euro filter. In Joseph A. Konstan, Ed H. Chi, and Kristina Höök, editors, *CHI 2012, it's the experience*, page 2527, New York, NY, 2012. ACM.

[35] G. Welch and G. Bishop. An introduction to the kalman filter; siggraph 2001 course 8; los angeles, ca; aug. 12-17, 2001; http. *info. acm. org/pubs/toc/CRnotice. html*, pages 1–80, 2001.

[36] Joseph J. LaViola. Double exponential smoothing: An alternative to kalman filter-based predictive tracking. In *Proceedings of the workshop on Virtual environments 2003*, pages 199–206, 2003.