A Closed-Form Solution for Human Finger Positioning

Roel Duits (3479072)
supervisors:
Arjan Egges
A. F. van der Stappen

Figure 1: *Hand poses spelling MIG '15 in American Sign Language using our closed-form solution. This hand is based on a model from the LibHand library.*

Abstract

In this paper we describe a novel technique for solving the inverse kinematics problem for human fingers. We derive a closed-form solution that places the fingertips precisely in the desired location by allowing minor deviations in the rotations of the Proximal Interphalangeal and Distal Interphalangeal joints compared to the fixed ratio that is known to exist between them. The obvious advantage is that with our approach there is no need to iterate until the distance between the fingertips and the desired locations is small enough. We show that this method is reliable and exact while showing minimal differences to the finger poses generated by the original closed-form solution. In our experiments we found the positions of the intermediate joints of the finger to deviate only around 1.5 mm in the worst case from those resulting from a numerical approximation of the original closed-form solution. On average the deviation is less than 0.5 millimeter.

CR Categories: I.2.9 [Computer Graphics]: Three-Dimensional Graphics and Realism—Display Algorithms I.3.7 [Articial Intelligence]: Robotics—Kinematics and dynamics;

Keywords: Inverse Kinematics, Human Finger, Animation

1 Introduction

In our pursuit to create virtual worlds that are as realistic as possible, we strive to recreate all the complex motions that humans use in everyday life. Among these are all the tasks we carry out with our hands, from moving objects to aiding us in conversations. The problem of posing a virtual hand is rather difficult due to the dextrous nature of our hands, containing 27 degrees of freedom (or DOFs).

In order to be able to have a virtual character perform the same range of manipulation tasks as a real human counterpart, we will need a fast and accurate way of working out the joint configurations needed to perform the desired task. To improve our capabilities in simulating these motions we have looked into the flexion and extension of an individual human finger. This will allow us to animate placing a finger on an object when the hand is positioned for a grasp or other manipulation task. Previous methods have relied on known inverse kinematics (IK) solutions that are derived numerically to solve the joint configurations. According to previous research (e.g. [Rijpkema and Girard 1991]), there is a fixed ratio between the angles of the finger joints. Using this ratio, it is possible to write a closed-form expression that uniquely describes the relation between the ratio, the finger bone lengths, the finger joint angles, and the desired distance between the Metacarpophalangeal joint and the finger tip. Unfortunately, the complexity of this closed-form expression prohibits calculating the finger joint angles corresponding to a desired distance.

In this paper, we present a fast analytical method based on this closed-form solution of the human finger. By allowing for some deviation in the ratio of the joint angles, we propose an approximation of the original closed-form expression that does allow for calculating precise finger joint angles, while still ensuring an exact finger tip position. We will show that our solution is both fast and reliable, providing results that are very similar to what would be obtained using a numerical approximation. This will make complex hand animations easier to perform on-the-fly, without needing to perform iterative IK.

Main contribution We propose a novel analytical method for finger posing. We show that our method is faster than iterative IK methods, and creates visually indistinguishable poses by allowing some minor deviations in the joint angles.

2 Related work

Obtaining realistic human finger poses is a challenging problem. Many animation applications use motion capture to obtain realistic animations for the characters. Hand motions are notoriously difficult to capture. A large number of markers is required to capture the subtleties of human hand motions, and one potentially has to deal with many self-occlusions. Wearing a glove with markers may cause a loss in fidelity due to the glove constraining the hand motion. Also, the glove may interfere with the way the soft tissue on the finger tip interplays with establishing a stable grasp. The soft body contacts allow for a large range of frictional forces and torques that can be applied to an object. These forces and torques enable us to grasp many objects with only a few fingers without the risk of dropping it. Some researchers have proposed a model that includes soft contacts. For example, [Ciocarlie et al. 2007] present a soft contact model that provides the normal and frictional forces applied at contact locations. [Barbagli et al. 2004] make a comparison between several contact models.

Some recent work aims to capture motion without using markers and instead relies on video tracking. For example, [Wang et al. 2013] use multiple video streams to estimate a pose that results in a similar image for each viewpoint in the virtual world. Another example of this approach is provided [Ballan et al. 2012], who use a variety of computer vision techniques in addition to learned salient points to enhance the tracking capabilities of their system. Finally, [Erol et al. 2007] provide an overview of various ways to capture hand poses and track their motions.

For interaction in virtual worlds in particular, motions obtained from tracking are not sufficient: one needs to be able to adapt a motion to suit particular objects or grasps. An example of a solution is given by [Pollard and Zordan 2005], who propose a physics-based grasping controller derived from example motion capture data that is able to adapt hand motions to different situations. In addition to adapting motions to suit the task at hand one also needs to account for obstacles in the environment. For example, [Berenson et al. 2007] present a system that plans an appropriate way of grasping an object without colliding with the environment.

A common way to adapt motions is using inverse kinematics to solve the problem of finding the correct joint configurations that will place end-effectors of a virtual character in the correct location. A good overview of these methods is provided in [Buss 2009] and [Aristidou and Lasenby 2009]. While forward kinematics is straightforward and leads to a single possible solution, the inverse problem generally is underspecified. As a result, a pose calculated by an IK solver in many cases is not the perfect solution, because IK solvers do not have any knowledge about the range of human poses that are realistic or comfortable. This can be partially solved by defining additional constraints such as joint range constraints. Alternatively we can provide the solver with higher-level information such as a database of common human poses that can be used as a starting point for finding reasonable IK solutions [Grochow et al. 2004]. Most IK solvers use a numerical approach that only provides an approximation rather than an exact solution, requiring multiple iterations to find a solution or sometimes being unable to find a solution at all even though one should exist.

Ideally, one would tackle the IK problem by using a closed-form solution based on a particular configuration of joints and the constraints that precisely describe a one-to-one mapping between end-effector positions and joint orientations. A closed-form solution is exact and probably also faster than numerical methods. In most cases however, it is impossible to find such a closed-form solution as they often stem from a geometric or algebraic interpretation. Finding a closed-form solution for finger positioning is possible, because of the relatively simple configuration of joints, and existing knowledge about finger bone lengths and joint angle ratio.

In their work [Buryanov and Kotiuk 2010] present a study on the length of finger segments measured across adults from Europe. They provide the mean and standard deviation for each finger bone. Of equal importance is the relationship between the finger joints angles. Many researchers, such as [Rijpkema and Girard 1991], [Lin et al. 2000], [Cobos et al. 2007] and [Adnan et al. 2012], make use of the ratio $\theta_{\text{DIP}} = \frac{2}{3}\theta_{\text{PIP}}$, to be explained later in greater detail. [Kim 2014] presents a method to determine this ratio for any finger by creating a triangle from the three finger bones. However, it is not feasible for humans to put their finger in an exact triangle, due to joint angle constraints, the presence of flesh and skin, and in some cases the finger bone lengths may also prohibit forming such a triangle (for instance, if one finger bone is longer than the two other bones together).

Next to the finger segments lengths and finger joint angle ratio, it is also important to take into consideration what the range of motion of each finger is. Of interest is the research of [Hoyet et al. 2012], which aims to determine an optimal set of markers for tracking a wide range of finger motions. They found that subjects were not able to see any differences between hand and finger motions based on a complete marker set versus motions produced from a reduced marker set using interpolation techniques. This also indicates that minor variations in bending a finger are not perceived differently by subjects.

By exploiting the limited range of finger bone lengths and by allowing some freedom in the finger joint angle ratio of $\theta_{\text{DIP}} = \frac{2}{3}\theta_{\text{PIP}}$ we propose a closed-form solution for finger positioning. To our knowledge, no such closed-form solution has been proposed before.

The remainder of this paper is organized as follows. In Section 3 we will detail the hand model and kinematics that we use for our technique. An in-depth explanation of the final closed-form solution is provided in Section 4, with experiments and results of a comparison to a numerical inverse kinematics solution in Section 5. We conclude the paper and provide some suggestions for future research in Section 6.

3 Kinematics of the human hand

In order to explain our novel approach to the solution of the inverse kinematics problem for human fingers, we will first need to cover the basics of how we model the human hand and the kinematics that we work with. In this chapter we present our hand model in Section 3.1. After that we will take a closer look at an individual finger in Section 3.2. Following this section we will discuss the kinematic equations that we used to derive our inverse kinematics solution in Section 3.3.

3.1 Human hand model

The human hand consists of five fingers, each consisting of three phalanges or finger bones. The only exception is our opposable thumb which has only two bones. Each finger is connected to the wrist via two additional bone structures known as the carpals and metacarpals.

The finger starts with the Proximal Phalanx (**PP**) which is attached to the main body of a hand through the Metacarpophalangeal joint or **MCP** joint. We model this joint with two DOFs to allow for a flexion or extension movement that allows us to open and close our fingers or an abduction or adduction movement that allows us

to spread our fingers apart or bring them back together. The focus of our work will be on the flexion and extension motions of the finger, but the abduction/adduction motion is an important part of positioning as well. It is however easier to solve for. Continuing from the **PP** towards the fingertip we have the Intermediate Phalanx (**IP**) which is connected to the **PP** through the Proximal Interphalangeal joint or **PIP**. This allows for one DOF of flexion and extension movements. Finally we have the Distal Phalanx (**DP**) which is connected to the **IP** through the Distal Interphalangeal joint or **DIP**. This too has only one DOF that allows for flexion or extension. In the next subsection we will detail the dependence that exists between the rotations of the **PIP** and **DIP** joints.

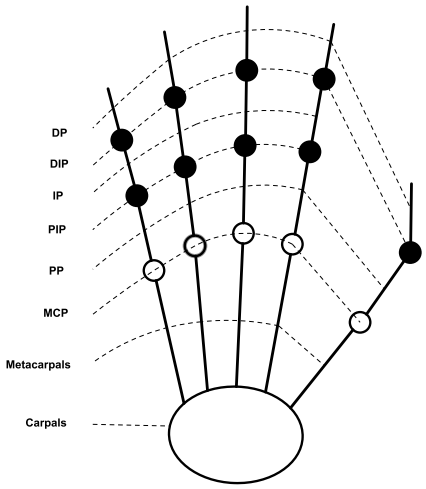

Figure 2: *A structural model of the bones in a human hand.*

The thumb is an exception to this layout as it does not have an IP bone or DIP joint. Additionally the thumb has only one DOF in the MCP joint, but three DOFs in the Carpometacarpal joint that connects the carpal bones to the metacarpal bones. See Figure 2 for an illustration of the skeletal structure described above. This model is similar to that used in [Hoyet et al. 2012], but without the third DOF in the MCP joints that they use to capture a subtle roll in the fingers.

This work focuses on the DP, IP, and PP bones and the DIP, PIP and MCP joints. These terms will be used for the remainder of this work to refer to the elements of the finger skeleton. While the metacarpal bones allow the hand palm to curve and thus alter the final position of the fingers we will not include them in our analysis. We will assume that the hand has been put in place and that each MCP joint is positioned such that each finger only needs to bend correctly to touch an object in a given desired location. After all aligning the flexion/extension direction can be solved easily with a rotational transformation.

The problem we aim to solve is to place the tip of the finger at a desired goal position. Our technique will apply to all fingers, including the thumb if some assumptions are made (see Section 4.3). While the thumb misses a phalangeal bone, the metacarpal provides a similar role to the proximal phalanges of other fingers. Although our technique is able to position the tip of the thumb correctly, it will ignore the increased dexterity that is present in the human thumb with regard to the other fingers.

3.2 Individual finger model

We will now focus on the problem of positioning the tip of a given finger. Throughout the remainder of this paper we will denote by θ_{MCP}, θ_{PIP} and θ_{DIP} the angles of the joints MCP, PIP and DIP respectively. Additionally we will denote by L_{PP}, L_{IP} and L_{DP} the lengths of the PP, IP and DP bones respectively. As noted before, the MCP joint is the only joint that has two DOFs instead of just one like the other joints that are part of the finger. Since the focus of this paper is on the flexion or extension motion of a finger we will treat the MCP joint as a single DOF joint. The sideways motion is easy to compute by aligning the plane in which the fingers bends with the plane through the desired fingertip position and the base of the finger. Figure 3 shows a kinematic diagram of our finger with the single DOF for the MCP joint.

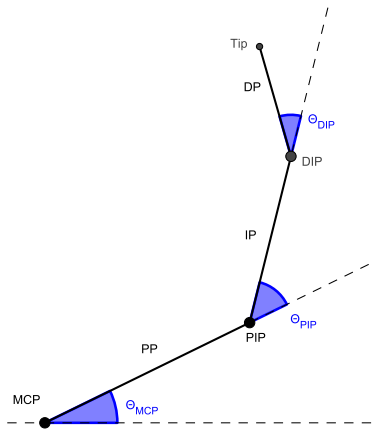

Figure 3: *The kinematics and structure of a single human finger.*

One key characteristic of the human finger is the dependence between the second (PIP) and third (DIP) joint. In [Rijpkema and Girard 1991], [Lin et al. 2000], [Cobos et al. 2007] and [Adnan et al. 2012] the authors all use the following relation:

$$\theta_{\text{DIP}} = \lambda\theta_{\text{PIP}}, \quad \lambda = \frac{2}{3} \tag{1}$$

Kim [Kim 2014] gives a method to determine λ from the lengths of the finger bones but it does not always provide a proper solution. In this work we made use of the measurements made by [Buryanov and Kotiuk 2010], who have gathered data on the bone lengths of human fingers from multiple European adults. With this data it was often impossible to apply the method of [Kim 2014] to find a different λ, as it was not possible to form a triangle using these lengths for the finger bones. As such we will utilize the ratio mentioned in Equation (1). Our approach can still be used if a different λ value is preferred.

Another interesting feature of the human finger is that the distance from the base to the tip of the finger is controlled by the PIP and DIP joints, whereas the MCP joint can be used to let the finger trace an arc of radius equal to that distance around the base of the finger. We will use both of the aforementioned properties of human fingers to find a closed-form solution for the inverse kinematics problem for human fingers in the next section.

3.3 Kinematics of human fingers

With the model described in the previous subsection we can now turn our attention to the kinematic equations. We will provide a

closed-form solution to determine the distance from the base to the tip of the finger, that relies only on θ_{PIP}. After this we can dive into our main contribution in Section 4.

As the flexion and extension of a finger essentially takes place in the plane, we can model the forward kinematics in 2D. This will hold true in a local frame of reference as well in 3D applications. If we let x_{tip} and y_{tip} be the x- and y-coordinates of the fingertip then we can use basic trigonometric identities to obtain the following equations:

$$\begin{aligned} x_{\text{tip}} &= L_{\text{PP}}\cos(\theta_{\text{MCP}}) + L_{\text{IP}}\cos(\theta_{\text{MCP}} + \theta_{\text{PIP}}) \\ &\quad + L_{\text{DP}}\cos(\theta_{\text{MCP}} + \theta_{\text{PIP}} + \theta_{\text{DIP}}) \\ &= L_{\text{PP}}\cos(\theta_{\text{MCP}}) + L_{\text{IP}}\cos(\theta_{\text{MCP}} + \theta_{\text{PIP}}) \\ &\quad + L_{\text{DP}}\cos(\theta_{\text{MCP}} + (1+\lambda)\theta_{\text{PIP}}) \end{aligned} \tag{2}$$

$$\begin{aligned} y_{\text{tip}} &= L_{\text{PP}}\sin(\theta_{\text{MCP}}) + L_{\text{IP}}\sin(\theta_{\text{MCP}} + \theta_{\text{PIP}}) \\ &\quad + L_{\text{DP}}\sin(\theta_{\text{MCP}} + \theta_{\text{PIP}} + \theta_{\text{DIP}}) \\ &= L_{\text{PP}}\sin(\theta_{\text{MCP}}) + L_{\text{IP}}\sin(\theta_{\text{MCP}} + \theta_{\text{PIP}}) \\ &\quad + L_{\text{DP}}\sin(\theta_{\text{MCP}} + (1+\lambda)\theta_{\text{PIP}}) \end{aligned} \tag{3}$$

Note that we have used Equation (1) to eliminate θ_{DIP} from Equations (2) and (3). As mentioned before θ_{PIP} and θ_{DIP} control the distance from the base to the tip of the finger. If we are given a distance d from the base of the finger to the desired location of the tip of the finger, we can use the Pythagorean theorem and trigonometric equalities to obtain:

$$d = \sqrt{x_{\text{tip}}^2 + y_{\text{tip}}^2} \tag{4}$$

$$\begin{aligned} d^2 &= L_{\text{PP}}^2 + L_{\text{IP}}^2 + L_{\text{DP}}^2 \\ &\quad + 2L_{\text{PP}}L_{\text{IP}}\cos(\theta_{\text{PIP}}) \\ &\quad + 2L_{\text{PP}}L_{\text{DP}}\cos((1+\lambda)\theta_{\text{PIP}}) \\ &\quad + 2L_{\text{IP}}L_{\text{DP}}\cos(\lambda\theta_{\text{PIP}}) \end{aligned} \tag{5}$$

See Appendix A for a step by step derivation of Equation (5).

If we could solve Equation (5) for θ_{PIP} we would obtain a closed-form solution for the inverse kinematics problem. It would give us a θ_{PIP} and, by Equation (1), also a θ_{DIP}. To obtain θ_{MCP} given θ_{PIP} and θ_{DIP} we use a triangle consisting of the base of the finger, the tip of the finger after setting θ_{PIP} and θ_{DIP} and finally the desired goal position as the vertices. We know the distance from the base to the tip, which is equal to the distance from the base to the goal position. The distance between the tip and the goal position can be easily calculated. This provides us with the lengths of all the sides of the triangle and allows us to calculate any desired angle. The angle between the sides connected to the base of the finger will tell us how much we need to adjust θ_{MCP} to move the tip onto the goal position.

Unfortunately it turns out to be extremely challenging to solve Equation (5) for θ_{PIP} given a desired value for d. Due to the three different cosines the terms cannot be combined into a single term. At best this would provide us with a closed-form solution that could be solved numerically. However, it turns out that the right hand side of our equality for d^2 behaves very much like a cosine, despite the impossibility to combine the three cosines into one. We propose to approximate the right-hand side of the equation by a single cosine. With this we find a value for θ_{PIP}. We can then compute the required θ_{DIP} that makes the finger tip reach the desired location. Since we relax the dependency constraint between θ_{PIP} and θ_{DIP}, there will be a difference in the locations of the PIP and DIP joints compared to a solution that would strictly adhere to Equation (1). As we will show in Section 5, this difference is minimal.

4 The approximation of the closed-form solution

In this section we will first discuss our approximation of θ_{PIP} and then the computation of θ_{DIP} to position the finger tip at the desired location.

4.1 Approximating θ_{PIP}

As mentioned in Section 3.3, the right-hand side of Equation (5) seems to behave very similar to a cosine function on a part of the domain. Figure 4 shows the distance d from the base to the tip of the finger on a sufficiently large range of values for θ_{PIP}. The continuation of the plot would show that the distance eventually increases again, but this is only possible if we bend the finger beyond the point of what is physically possible, where it would intersect with itself. Realistically we can only bend our finger to the point where

$$\theta_{\text{PIP}} + \theta_{\text{DIP}} = (1+\lambda)\theta_{\text{PIP}} = \pi \tag{6}$$

which corresponds to the DP bending back to be parallel to the PP. Clearly there will be small differences between the fingers as some fingers can bend more easily than others. We will leave these minor differences out of consideration. The part of the plot that is interesting to us lies in the domain ranging from 0 to a maximum value for θ_{PIP}, which, by Equation (6) equals:

$$\theta_{\text{PIP}} = \frac{\pi}{1+\lambda} = \frac{\pi}{\frac{5}{3}} = \frac{3}{5}\pi \tag{7}$$

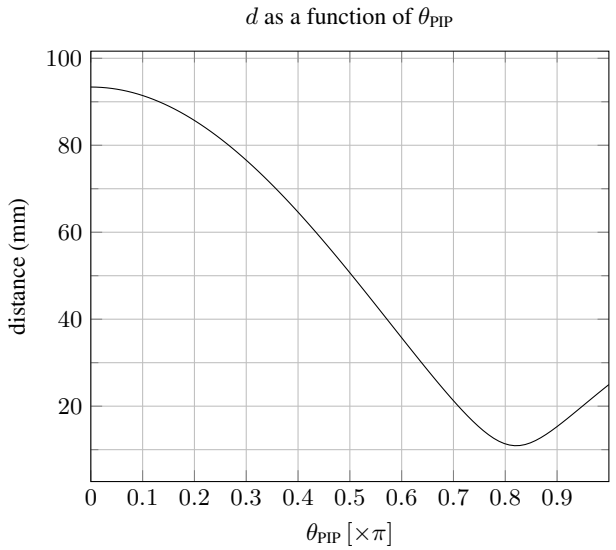

Figure 4: *A plot of the square root of Equation* (5)*. The X-axis is the radial angle for θ_{PIP} and the Y-axis is the distance between the tip and base of the finger. The following lengths in* mm *were used: L_{PP} = 44.63, L_{IP} = 32.33 and L_{DP} = 16.43.*

Within the interval $[0, \frac{3\pi}{5}]$ the plot behaves much like a regular cosine, which is what we will use to approximate Equation (5). To do so we must find both an amplitude and a period so that the cosine lines up as much as possible. We chose to take the total finger length L_T as the amplitude and apply no vertical shift. This makes the cosine start at the correct distance and follow the curve very accurately in the domain of our interest. The period can be found by choosing a second point, which we call the anchor point, on the

original plot. This anchor point is defined by an angle θ_A and the corresponding distance d_A when we insert θ_A into Equation (5). To make our approximation in Equation (10) pass through the anchor point we need to find the following values:

$$L_T = L_{\text{PP}} + L_{\text{IP}} + L_{\text{DP}} \tag{8}$$

$$B = \frac{\arccos(\frac{d_A}{L_T})}{\theta_A} \tag{9}$$

Equation (9) is obtained by solving the general form $d = L_T * \cos(B\theta_{\text{PIP}})$ for B. This then gives us the following approximating equations for d and θ_{PIP} respectively:

$$d = L_T \cos(B\theta_{\text{PIP}}) \tag{10}$$

$$\theta_{\text{PIP}} = \frac{\arccos(\frac{d}{L_T})}{B} \tag{11}$$

With this simpler cosine function from Equation (10) we can approximate Equation (5) accurately. This function can also be solved for θ_{PIP} to obtain Equation (11), allowing us to use it to solve an IK problem. There is however a slight difference between the distances calculated with Equations (5) and (10). This in turn causes an error to appear when we wish to obtain a θ_{PIP} from Equation (11). As such we will need to make a correction to ensure we position the fingertip exactly where we want to.

4.2 Solving exactly by adjusting θ_{DIP}

Our approach will be to allow minor deviations from the ratio between θ_{PIP} and θ_{DIP} as imposed by Equation (1). The freedom to deviate allows us to derive a closed-form expression for θ_{DIP}. As we will show in Section 5, it turns out that the actual deviations are so small that they are practically unnoticeable. Moreover, there is no evidence that the ratio is exactly the same for all fingers and all humans and that it is strictly maintained while going from fully stretched to fully bent.

In order to correct our approximation, we will drop the requirement to satisfy Equation (1) and instead let it be implicitly controlled by manually adjusting θ_{DIP}. After all, the above approximation puts the tip almost at the desired distance but not precisely. If we can find a θ_{DIP} such that the tip is placed at the exact distance then we can remove this error in the distance from the base to the tip completely. The only difference between the original closed-form solution and the solution allowing variations of λ will be in the position of the two finger joints PIP and DIP.

In order to find the exact value for θ_{DIP} that will place the fingertip at the exact distance we desire, we will need to solve a few extra equations. First off we will need the distance between the MCP and DIP joints. This can be calculated using the law of cosines on the triangle between the MCP, PIP and DIP joints. By using L_{PP}, L_{IP} and the angle between them we obtain the distance between the MCP and DIP joints. We will let this distance be m. We can trivially substitute the internal angle with our value for θ_{PIP}. This is done in Equation (12) below.

With m, in addition to L_{PP} and L_{IP}, we can apply the law of cosines again to find the angle α between m and IP. We then turn our attention to the triangle between the MCP and DIP joints and the desired fingertip position to find angle β between m and DP. One last application of the law of cosines with lengths d, m and L_{DP} will give us the exact value for angle β. The angles α, β and θ_{DIP} form a straight angle, allowing us to obtain the exact value for θ_{DIP} that we need to position the fingertip. Figure 5 provides an overview of the additional values needed to complete the positioning. The equations below show all the calculations that will be needed in addition to Equation (11):

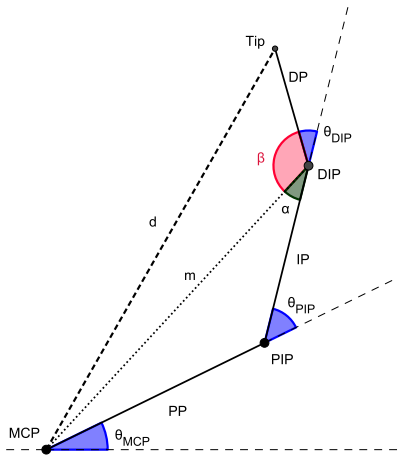

Figure 5: *An overview of the additional lengths and angles used to complete the approximation and place the fingertip exactly at the desired distance from the base.*

$$m = \sqrt{L_{\text{PP}}^2 + L_{\text{IP}}^2 + 2L_{\text{PP}}L_{\text{IP}}\cos(\theta_{\text{PIP}})} \tag{12}$$

$$\alpha = \arccos(\frac{L_{\text{IP}}^2 + m^2 - L_{\text{PP}}^2}{2L_{\text{IP}}m}) \tag{13}$$

$$\beta = \arccos(\frac{L_{\text{DP}}^2 + m^2 - d^2}{2L_{\text{DP}}m}) \tag{14}$$

$$\theta_{\text{DIP}} = \pi - \alpha - \beta \tag{15}$$

As mentioned before the only difference that results from this solution is where the joints PIP and DIP end up. It is also worth pointing out that because of the reliance on arccosines in the calculations for α and β, it is possible our method is unable to find a suitable θ_{DIP}. This will occur only if the value for θ_{PIP}, obtained with Equation (11), places joint DIP at a distance m such that $d > m + L_{\text{DP}}$. This means no triangle is formed between the MCP and DIP joints and the desired fingertip position as the triangle inequality does not hold. Therefore angle β cannot be calculated as in Equation (14). In our experiments we found that this occurs very rarely and only when anchor points with a small angle for θ_A are chosen for Equation (9).

4.3 IK for the thumb

As stated earlier in Section 3.1, our method is not entirely accurate if we want to use it for the thumb. We can however see from Figure 2 that if we include the metacarpal bone of the thumb, we once again end up with three links. Additionally, the MCP joint for the thumb only has one DOF, much like the PIP joint it is missing. However the Carpometacarpal or CMC joint between the carpal and metacarpal bones provides us with 3 DOFs. This will require a different transformation to align the flexion/extension plane with the direction of the desired end position of the fingertip than what would be necessary for the other fingers.

Assuming we can apply a transformation to the CMC joint that aligns the flexion/extension plane of the thumb we could still apply our method. We simply take the Metacarpal or MC, the PP and the DP as the links. The thumb is not able to bend as much as the

other fingers and may require a different anchor point. As noted previously in Subsection 3.1, the thumb also lacks the λ ratio that is present in our other fingers. We can still apply such a relation to the joints if we want however, at a cost of reducing the range of motions we can then animate. In many cases this may not be a problem as the thumb will often bend much like our other fingers to perform a grasp or other such task. If a larger range of motions is desired an alternative method needs to be applied for the thumb specifically.

Taking into account that all our equations depend on a good estimate of θ_{PIP}, it is very important how we define the anchor point in Equations (7) and (9). This will have a big impact on whether the approximation yields a good value for θ_{PIP}. In the following section we will test various anchor points. Based on our simulations, we will suggest an anchor point value that allows us to always successfully place the finger tip at the goal position.

5 Simulations

We wish to determine if our solution works for any choice of finger bone lengths and for any desired reachable goal distance for a character. To do so we implemented a simulation in Python with a simple 2D representation of a human finger that can be bent and extended like a human finger. The application allows us to define the lengths of the PP, IP and DP bones, as well as the anchor point.

5.1 Setup

To investigate the role of the anchor point we relax the right-hand side of Equation (6) by requiring

$$\theta_{\text{PIP}} + \theta_{\text{DIP}} = \theta_U \tag{16}$$

where θ_U is a user-defined angle that represents the total angle that the fingertip is rotated by when starting from a stretched finger pose. This θ_U therefore specifies a distance from which we will derive the anchor point. The angle θ_A that we need for the anchor point is equal to θ_{PIP} in Equation (16). With this angle θ_A we can find d_A with Equation (5) and then B with Equation (9). The anchor point, represented by the the values of θ_A and d_A, defines where the approximation will align perfectly with the closed-form solution. We can vary this anchor point to determine which one provides us with the most similar results when comparing it to the original closed-form solution from Equation (5).

Using this setup we draw two fingers on the screen, as can be seen in Figure 6. One finger will be posed using a gradient-descent iterative method to find the joint configurations. The other finger will then be posed using our proposed closed-form solution so that the fingertip positions match. To verify which anchor point works best we have considered several total angles for θ_U ranging from $\frac{1}{2}\pi$ to $\frac{7}{6}\pi$. We measured the distance between the locations of the fingertips, the distance between the positions of PIP and DIP, and the difference in the ratio λ between θ_{PIP} and θ_{DIP}.

To ensure we cover a large enough range of possible finger configurations, we make use of the data provided in [Buryanov and Kotiuk 2010] for each finger except the thumb. We artificially increase the length of the DP bone to account for the tissue between the distal end of the DP bone and the fingertip. This length is also part of the dataset. For each anchor point, we created a thousand configurations per finger. Each of these configurations was then tested on a thousand different distances. The configurations are created randomly by using a normal sampling based on the length data. The distances are sampled randomly from the range starting at a fully

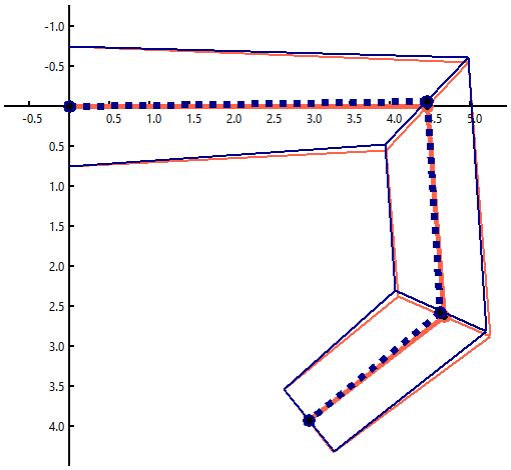

Figure 6: *An example of our simulation program. The red finger is posed using the gradient descent based iterative method. The dark blue finger is posed using our method to align the end points.*

stretched finger and ending at a finger bent so that it satisfies Equation (6). With this we determine the average deviation and corresponding standard deviation, as well as the maximum deviation for the difference in PIP and DIP positioning. We also calculate the average and maximum absolute λ ratio deviation, given by $|\lambda - \lambda_{\text{CF}}|$, where λ_{CF} stands for the ratio obtained by our method. Finally we keep track of how often our approach provides no suitable approximation as explained in Section 4.2.

5.2 Results

The results are presented in Figures 7, 8 and 9. With respect to the positional deviations of the PIP and DIP joints, we see that in most cases the average is less than half a millimeter in difference for both. Only at the high end of our considered θ_U angles do we see the average deviations rise above one millimeter.

We see the maximum deviations can be a bigger issue, going well over ten millimeter at the low end of our considered range. Near the middle we find at most 1.5 mm. Note that these maximum deviations result from the worst case length combinations that throw off our estimate with Equation (11).

We see the absolute difference between λ and λ_{CF} grow from 0.026 to as much as 0.13 as we increase the total angle θ_U. The reason for this is that with higher values for θ_U, the approximation from Equation (11) will mainly return larger angles for θ_{PIP} than Equation (5). With smaller total angles for θ_U the angles provided for θ_{PIP} by Equation (11) will be both larger and smaller than those from Equation (5), resulting in a better estimate before we adjust θ_{DIP}. The required change in θ_{DIP} and resulting λ_{CF} will thus be smaller compared to the ratio in Equation (1).

Keep in mind that for both the average and maximum deviations of the λ_{CF} ratio we took the absolute value of the deviation, to capture the difference regardless of its sign. We see that the difference can be as large as almost 0.4 at the extremes of our considered angles for θ_U. In the middle of our range the maximum deviation is no more than 0.2. Note that these maximum values were taken over all thousand finger length configurations and as such we see what happens with the worst cases of length combinations. Our approximation of θ_{PIP} from Equation (11) becomes a little less accurate, which requires a compensation from θ_{DIP}. As the average deviation shows most of the time the difference will be hardly noticeable with

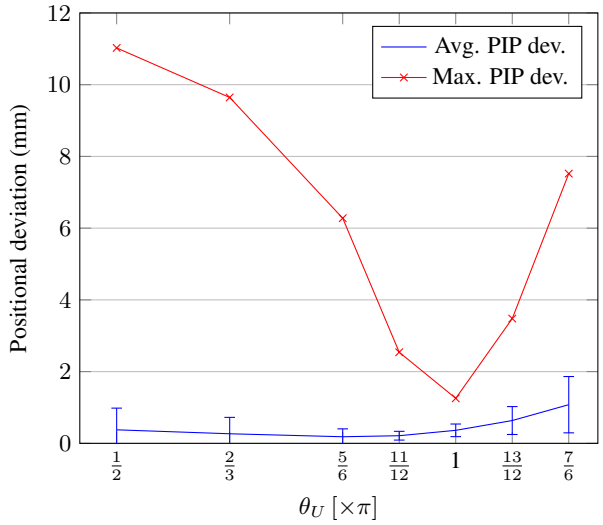

Figure 7: *The average, standard deviation and maximum per anchor point configuration for the PIP positional deviations.*

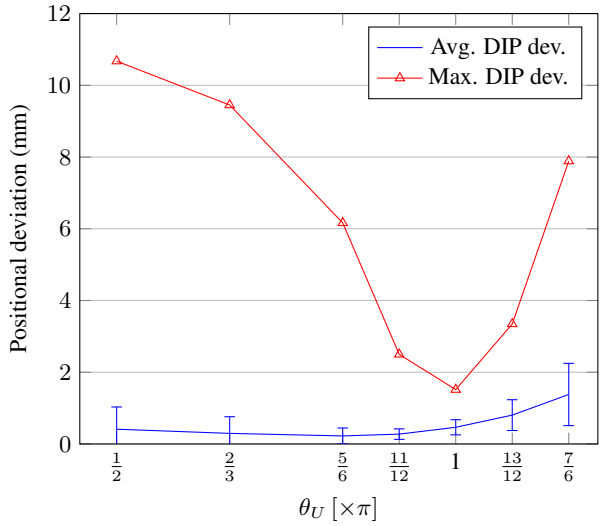

Figure 8: *The average, standard deviation and maximum per anchor point configuration for the DIP positional deviations.*

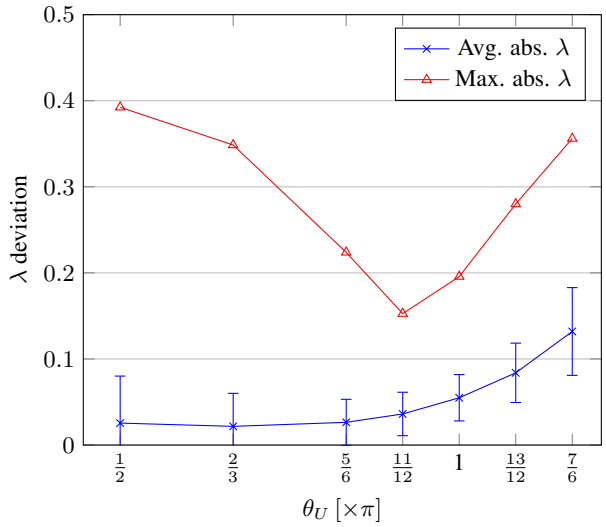

Figure 9: *The average, standard deviation and maximum per anchor point configuration for the absolute λ deviations.*

many only 0.05 away from the desired λ ratio.

From these statistics it becomes clear that the smallest average deviations are found around $\theta_U = \frac{5}{6}\pi$ and the lowest standard deviations from the mean lie around $\theta_U = \frac{11}{12}\pi$. The maximum deviations differ a bit more, with the PIP and DIP positional deviations being the smallest near $\theta_U = \pi$ and the maximum λ deviation around $\theta_U = \frac{11}{12}\pi$.

Our simulations show that for smaller values of θ_U it becomes impossible to position certain instances of the index finger with our method. In particular this occurs for index fingers where the sampling provided a long PP and a short DP according to the data from [Buryanov and Kotiuk 2010]. In these cases the estimate provided by Equation (11) proves to be too far off to still be corrected using θ_{DIP}. We found no such instances when we chose θ_U larger than or equal to $\frac{11}{12}\pi$ and so we suggest taking this as a lower bound when defining the value for θ_U

Based on these observations, we would recommend using a total angle of either $\frac{11}{12}\pi$ or π to base the anchor point around. Taking $\frac{11}{12}\pi$ provides a better average case for both positional deviations as well as the lowest maximum λ ratio deviation. Using π results in the lowest maximum positional deviations for the PIP and DIP joints. The difference in λ is still very small at this total angle as well. While the average deviation and standard deviation rise a little with this choice, it is still only around half a millimeter and thus hardly noticeable. Both of these choices also guarantee that we will always be able to find a solution, which is not guaranteed when using a smaller θ_U.

5.3 Performance

Compared to the gradient descent based method, which uses the closed-form solution from Equation (5), our method finds the answer faster. This is because in a gradient descent, we calculate the derivative on our current estimate and alter our input θ_{PIP} accordingly in iterations. We chose to adjust our current estimate by 75% of the ratio difference / gradient. We start the estimation in the middle of the domain available to θ_{PIP}. This setup would find an answer within 0.0001 mm of our desired distance within 10 to 15 iterations. We made a runtime comparison between the two methods by gener-

ating 100000 random configurations and measured the time it took to find a solution for both approaches. Using our approach a pose was computed in 13.4 microseconds on average ($\sigma = 0.1$) as opposed to 56.0 microseconds ($\sigma = 3.1$) when using the gradient descent method. Thus, our method is at least four times as fast, which is a nice gain considering this potentially applies to each finger of every character being animated. It also varies a lot less between different runs, resulting in a more stable performance, where the gradient descent based method can struggle if it needs more iterations to reach the desired error threshold. It should also be noted that our gradient descent based method may already outperform an even more general IK solver. This is because it uses the λ factor directly, rather than trying to find a solution for the joint angles that also happens to have this property.

5.4 Summary

Below, we provide the pseudocode for calculating the finger joint angles given a desired distance using our closed-form solution. We assume an anchor point value of π in the algorithm.

Require: $(L_{\text{PP}}, L_{\text{IP}}, L_{\text{DP}})$
$L_T \leftarrow L_{\text{PP}} + L_{\text{IP}} + L_{\text{DP}}$ {from (8)}
$\theta_A \leftarrow \frac{3}{5}\pi$ {from (7)}
$d_A \leftarrow d(\theta_A)$ {from (5)}
$B \leftarrow \frac{\arccos(\frac{d_A}{L_T})}{\theta_A}$ {from (9)}
$\theta_{\text{PIP}} \leftarrow \frac{\arccos(\frac{d}{L_T})}{B}$ {from (11)}
$m \leftarrow \sqrt{L_{\text{PP}}^2 + L_{\text{IP}}^2 + 2L_{\text{PP}}L_{\text{IP}}\cos(\theta_{\text{PIP}})}$ {from (12)}
$\alpha \leftarrow \arccos(\frac{L_{\text{IP}}^2+m^2-L_{\text{PP}}^2}{2L_{\text{IP}}m})$ {from (13)}
$\beta = \arccos(\frac{L_{\text{DP}}^2+m^2-d^2}{2L_{\text{DP}}m})$ {from (14)}
$\theta_{\text{DIP}} = \alpha + \beta$ {from (15)}
return $(\theta_{\text{PIP}}, \theta_{\text{DIP}})$

6 Conclusion

In this paper we have presented a novel technique for solving the inverse kinematics problem for human fingers. Our approach is based on an approximation of the closed-form solution derived from forward kinematics. We have adjusted this solution by dropping the constraint of having a fixed ratio between the joint angles. We have shown that the method will always find a solution that places the fingertip at the desired distance. The finger pose can be calculated exactly instead of using numerical methods. Finally our experiments show that the differences between the positions of the joints when using the numerical method or our approach are negligible. We believe that our technique for solving the inverse kinematics problem for human fingers allows developers to include more advanced hand animations in their applications without requiring large amounts of computational power.

As we pointed out before, there are more motions that a human hand is capable of that cannot be solved with just our method alone. The metacarpal bones allow us to curve the palm of our hands, thus changing the plane in which our fingers bend. While our method captures the dependence between the joints of individual fingers, there is also a dependence between fingers in terms of flexion and extension, as well as abduction and adduction motions. This will be a point of future research, with the goal of developing a closed-form solution that allow us to position the fingertips exactly at the desired locations while taking into account all constraints found in real human hands, provided that the wrist has been positioned properly.

Acknowledgments

The hand model used in Figure 1 is based on the the model used in the LibHand application developed by Marin Saric under a Creative Commons Attribution 3.0 Unported License.

References

ADNAN, N. H., WAN, K., SHAHRIMAN, A. B., ZA'BA, S. K., DESA, H., AND AZIZ, M. A. A. 2012. The development of a low cost data glove by using flexible bend sensor for resistive interfaces. *The 2nd International Malaysia-Ireland Joint Symposium on Engineering, Science and Business*.

ARISTIDOU, A., AND LASENBY, J., 2009. Inverse kinematics: a review of existing techniques and introduction of a new fast iterative solver. `http://www.andreasaristidou.com/publications/CUEDF-INFENG,%20TR-632.pdf`.

BALLAN, L., TANEJA, A., GALL, J., VAN GOOL, L., AND POLLEFEYS, M. 2012. Motion capture of hands in action using discriminative salient points. *Proceedings of the 12th European Conference on Computer Vision 2012*, 640–653.

BARBAGLI, F., FRISOLI, A., SALISBURY, K., AND BERGAMASCO, M. 2004. Simulating human fingers: A soft finger proxy model and algorithm. *Proceedings of the 12th International Symposium on Haptic interfaces for Virtual Environment and Teleoperator Systems*, 9–17.

BERENSON, D., DIANKOV, R., NISHIWAKI, K., KAGAMI, S., AND KUFFNER, J. 2007. Grasp planning in complex scenes. *Proceedings of the 7th IEEE-RAS International Conference on Humanoid Robots*, 42–48.

BICCHI, A., AND KUMAR, V. 2000. Robotic grasping and contact: A review. 348–353.

BURYANOV, A., AND KOTIUK, V. 2010. Proportions of hand segments. *International Journal of Morphology 28*, 3, 755–758.

BUSS, S. R., 2009. Introduction to inverse kinematics with jacobian transpose, psuedoinverse and damped least squares methods. `http://www.math.ucsd.edu/~sbuss/ResearchWeb/ikmethods/iksurvey.pdf`.

BUTTERFASS, J., GREBERSTEIN, M., LIU, H., AND HIRZINGER, G. 2001. Dlr-hand ii: next generation of a dextrous robot hand. *Proceedings of IEEE International Conference on Robotics and Automation 1*, 109–114.

CANNY, J. F., AND GOLDBERG, K. Y., 1993. A risc approach to sensing and manipulation.

CIOCARLIE, M., MILLER, A., AND ALLEN, P. Soft finger model with adaptive contact geometry for grasping and manipulation tasks.

CIOCARLIE, M., LACKNER, C., AND ALLEN, P. 2007. Soft finger model with adaptive contact geometry for grasping and manipulation tasks. *WHC '07 Proceedings of the Second Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 219–224.

COBOS, S., FERRE, M., SÁNCHEZ-URÁN, M. A., AND ORTEGO, J. 2007. Constraints for realistic hand manipulation. *Proceedings of the 10th Annual International Workshop on Presence (PRESENCE '07)*, 369–370.

CZYZOWICZ, J., STOJMENOVIC, I., AND URRUTIA, J. 1999. Immobilizing a shape. *International Journal of Computational Geometry and Applications 9*.

DEIMEL, R., AND BROCK, O. 2014. A novel type of compliant, underactuated robotic hand for dexterous grasping. *Proceedings of Robotics: Science and Systems*.

EROL, A., BEBIS, G., NICOLESCU, M., BOYLE, R. D., AND TWOMBLY, X. 2007. Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding 108*, 1-2, 52–73.

FEIX, T., BODO SCHMIEDMAYER, H., ROMERO, J., AND KRAGI, D. 2009. A comprehensive grasp taxonomy. *In robotics, science and systems conference: workshop on Understanding the human hand for advancing robotic manipulation*.

FERRARI, C., AND CANNY, J. 1992. Planning optimal grasps. *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*.

GROCHOW, K., MARTIN, S. L., HERTZMANN, A., AND POPOVIĆ, Z. 2004. Style-based inverse kinematics. *Proceedings of ACM SIGGRAPH 2004 23*, 3, 1–5.

HOYET, L., RYALL, K., MCDONNELL, R., AND O'SULLIVAN, C. 2012. Sleight of hand: Perception of finger motion from reduced marker sets. *I3D '12 Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 79–86.

KIM, B.-H. 2014. Analysis of coordinated motions of humanoid robot fingers using interphalangeal joint coordination. *International Journal of Advanced Robotic Systems 11*, 69.

KRUGER, H., AND VAN DER STAPPEN, A. F. 2011. Partial closure grasps: metrics and computation. *IEEE International Conference on Robotics and Automation (ICRA)*, 5024–5030.

LIN, J., WU, Y., AND HUANG, T. S. 2000. Modeling the constraints of human hand motion. *Proceedings of Workshop on Human Motion 2000*, 121–126.

MARKENSCOFF, X., AND PAPDIMITRIOU, C. H. 1989. Optimum grip of a polygon. *International Journal of Robotics Research 8*, 17–29.

NGUYEN, V. D., 1986. Massachusetts Institute of Technology.

POLLARD, N. S., AND ZORDAN, V. B. 2005. Physically based grasping control from example. *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 311–318.

PONCE, J., SULLIVAN, S., SUDSANG, A., BOISSONNAT, J.-D., AND MERLET, J.-P. 1996. On computing four-finger equilibrium and force-closure grasps of polyhedral objects. *International Journal of Robotics Research 16*, 11–35.

PRATS, M., SANZ, P. J., AND DEL POBIL, A. P. 2007. Task-oriented grasping using hand preshapes and task frames. *IEEE International Conference on Robotics and Automation*, 1794–1799.

REULEAUX, F. 1875. *Kinematics of Machinery*.

RIJPKEMA, H., AND GIRARD, M. 1991. Computer animation of knowledge-based human grasping. *ACM SIGGRAPH '91 Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, 339–348.

RIMON, E., AND BURDICK, J. W. 1998. Mobility of bodies in contact. i. a 2nd-order mobility index for multiple-finger grasps. *IEEE Transactions on Robotics and Automation 14*, 696–708.

SAHBANI, A., EL-KHOURY, S., AND BIDAUD, P. 2012. An overview of 3d object grasp synthesis algorithms. *International Journal of Robotics Research 60*, 326–336.

WANG, Y., MIN, J., ZHANG, J., LIU, Y., XU, F., DAI, Q., AND CHAI, J. 2013. Video-based hand manipulation capture through composite motion control. *ACM Transactions on Graphics (TOG) - SIGGRAPH 2013 Conference Proceedings 32*, 4.

A Expanded equations

In this appendix we will detail how we find Equation (5) from Equations (2), (3) and (4).

$$d^2 = x_{\text{tip}}^2 + y_{\text{tip}}^2 \tag{17}$$

Expanding the individual squares gives us:

$$\begin{aligned} x_{\text{tip}}^2 = {} & L_{\text{PP}}^2 \cos^2(\theta_{\text{MCP}}) + L_{\text{IP}}^2 \cos^2(\theta_{\text{MCP}} + \theta_{\text{PIP}}) \\ & + L_{\text{DP}}^2 \cos^2(\theta_{\text{MCP}} + (1+\lambda)\theta_{\text{PIP}}) \\ & + 2L_{\text{PP}}L_{\text{IP}} \cos(\theta_{\text{MCP}}) \cos(\theta_{\text{MCP}} + \theta_{\text{PIP}}) \\ & + 2L_{\text{PP}}L_{\text{DP}} \cos(\theta_{\text{MCP}}) \cos(\theta_{\text{MCP}} + (1+\lambda)\theta_{\text{PIP}}) \\ & + 2L_{\text{IP}}L_{\text{DP}} \cos(\theta_{\text{MCP}} + \theta_{\text{PIP}}) \cos(\theta_{\text{MCP}} + (1+\lambda)\theta_{\text{PIP}}) \end{aligned} \tag{18}$$

$$\begin{aligned} y_{\text{tip}}^2 = {} & L_{\text{PP}}^2 \sin^2(\theta_{\text{MCP}}) + L_{\text{IP}}^2 \sin^2(\theta_{\text{MCP}} + \theta_{\text{PIP}}) \\ & + L_{\text{DP}}^2 \sin^2(\theta_{\text{MCP}} + (1+\lambda)\theta_{\text{PIP}}) \\ & + 2L_{\text{PP}}L_{\text{IP}} \sin(\theta_{\text{MCP}}) \sin(\theta_{\text{MCP}} + \theta_{\text{PIP}}) \\ & + 2L_{\text{PP}}L_{\text{DP}} \sin(\theta_{\text{MCP}}) \sin(\theta_{\text{MCP}} + (1+\lambda)\theta_{\text{PIP}}) \\ & + 2L_{\text{IP}}L_{\text{DP}} \sin(\theta_{\text{MCP}} + \theta_{\text{PIP}}) \sin(\theta_{\text{MCP}} + (1+\lambda)\theta_{\text{PIP}}) \end{aligned} \tag{19}$$

When we put these back together it gives us:

$$\begin{aligned} x_{\text{tip}}^2 + y_{\text{tip}}^2 = {} & L_{\text{PP}}^2 \Big(\cos^2(\theta_{\text{MCP}}) + \sin^2(\theta_{\text{MCP}}) \Big) \\ & + L_{\text{IP}}^2 \Big(\cos^2(\theta_{\text{MCP}} + \theta_{\text{PIP}}) + \sin^2(\theta_{\text{MCP}} + \theta_{\text{PIP}}) \Big) \\ & + L_{\text{DP}}^2 \Big(\cos^2(\theta_{\text{MCP}} + (1+\lambda)\theta_{\text{PIP}}) + \sin^2(\theta_{\text{MCP}} + (1+\lambda)\theta_{\text{PIP}}) \Big) \\ & + 2L_{\text{PP}}L_{\text{IP}} \Big(\cos(\theta_{\text{MCP}}) \cos(\theta_{\text{MCP}} + \theta_{\text{PIP}}) \\ & + \sin(\theta_{\text{MCP}}) \sin(\theta_{\text{MCP}} + \theta_{\text{PIP}}) \Big) \\ & + 2L_{\text{PP}}L_{\text{DP}} \Big(\cos(\theta_{\text{MCP}}) \cos(\theta_{\text{MCP}} + (1+\lambda)\theta_{\text{PIP}}) \\ & + \sin(\theta_{\text{MCP}}) \sin(\theta_{\text{MCP}} + (1+\lambda)\theta_{\text{PIP}}) \Big) \\ & + 2L_{\text{IP}}L_{\text{DP}} \Big(\cos(\theta_{\text{MCP}} + \theta_{\text{PIP}}) \cos(\theta_{\text{MCP}} + (1+\lambda)\theta_{\text{PIP}}) \\ & + \sin(\theta_{\text{MCP}} + \theta_{\text{PIP}}) \sin(\theta_{\text{MCP}} + (1+\lambda)\theta_{\text{PIP}}) \Big) \end{aligned} \tag{20}$$

During the following simplifications we will ignore the factors containing the link lengths for readability. The first three terms of Equation (20) can be simplified with the following trigonometric identity:

$$\cos^2(x) + \sin^2(x) = 1 \tag{21}$$

The fourth and fifth term can be simplified using the following math rules:

$$\cos(a+b) = cos(a)cos(b) - sin(a)sin(b) \tag{22}$$

$$\sin(a+b) = sin(a)cos(b) + cos(a)sin(b) \tag{23}$$

We apply these to the fourth term to get:

$$\begin{aligned} & \cos(\theta_{\text{MCP}}) \cos(\theta_{\text{MCP}} + \theta_{\text{PIP}}) + \sin(\theta_{\text{MCP}}) \sin(\theta_{\text{MCP}} + \theta_{\text{PIP}}) = \\ & \cos(\theta_{\text{PIP}}) \cos(\theta_{\text{MCP}}) \cos(\theta_{\text{MCP}}) - \cos(\theta_{\text{MCP}}) \sin(\theta_{\text{MCP}}) \sin(\theta_{\text{PIP}}) + \\ & \cos(\theta_{\text{PIP}}) \sin(\theta_{\text{MCP}}) \sin(\theta_{\text{MCP}}) + \cos(\theta_{\text{MCP}}) \sin(\theta_{\text{MCP}}) \sin(\theta_{\text{PIP}}) \end{aligned} \tag{24}$$

We can then combine some terms to get:

$$\begin{aligned} & \cos(\theta_{\text{PIP}})(\cos^2(\theta_{\text{MCP}}) + \sin^2(\theta_{\text{MCP}}) \\ & - \cos(\theta_{\text{MCP}}) \sin(\theta_{\text{MCP}}) \sin(\theta_{\text{PIP}}) \\ & + \cos(\theta_{\text{MCP}}) \sin(\theta_{\text{MCP}}) \sin(\theta_{\text{PIP}}) \end{aligned} \tag{25}$$

The first term again has the aforementioned Identity (21), while the last two terms cancel each other out. This therefore results in $\cos(\theta_{\text{PIP}})$ and $\cos((1+\lambda)\theta_{\text{PIP}})$ for the fourth and fifth term of Equation (20). For the last term of this equation we have the following:

$$\begin{aligned} & \cos(\theta_{\text{MCP}} + \theta_{\text{PIP}}) \cos(\theta_{\text{MCP}} + (1+\lambda)\theta_{\text{PIP}}) + \\ & \sin(\theta_{\text{MCP}} + \theta_{\text{PIP}}) \sin(\theta_{\text{MCP}} + (1+\lambda)\theta_{\text{PIP}}) \end{aligned} \tag{26}$$

This can be expanded into:

$$\begin{aligned} & \big[\big(\cos(\theta_{\text{MCP}}) \cos(\theta_{\text{PIP}}) - \sin(\theta_{\text{MCP}}) \sin(\theta_{\text{PIP}}) \big) \\ & \big(\cos(\theta_{\text{MCP}}) \cos((1+\lambda)\theta_{\text{PIP}}) - \sin(\theta_{\text{MCP}}) \sin((1+\lambda)\theta_{\text{PIP}}) \big) \big] + \\ & \big[\big(\sin(\theta_{\text{MCP}}) \cos(\theta_{\text{PIP}}) + \cos(\theta_{\text{MCP}}) \sin(\theta_{\text{PIP}}) \big) \\ & \big(\sin(\theta_{\text{MCP}}) \cos((1+\lambda)\theta_{\text{PIP}}) + \cos(\theta_{\text{MCP}}) \sin((1+\lambda)\theta_{\text{PIP}}) \big) \big] \end{aligned} \tag{27}$$

And then expanded again by working out the products into:

$$\begin{aligned} & = \cos(\theta_{\text{MCP}}) \cos(\theta_{\text{PIP}}) \cos(\theta_{\text{MCP}}) \cos((1+\lambda)\theta_{\text{PIP}}) \\ & - \cos(\theta_{\text{MCP}}) \cos(\theta_{\text{PIP}}) \sin(\theta_{\text{MCP}}) \sin((1+\lambda)\theta_{\text{PIP}}) \\ & - \sin(\theta_{\text{MCP}}) \sin(\theta_{\text{PIP}}) \cos(\theta_{\text{MCP}}) \cos((1+\lambda)\theta_{\text{PIP}}) \\ & + \sin(\theta_{\text{MCP}}) \sin(\theta_{\text{PIP}}) \sin(\theta_{\text{MCP}}) \sin((1+\lambda)\theta_{\text{PIP}}) \\ & + \sin(\theta_{\text{MCP}}) \cos(\theta_{\text{PIP}}) \sin(\theta_{\text{MCP}}) \cos((1+\lambda)\theta_{\text{PIP}}) \\ & + \sin(\theta_{\text{MCP}}) \cos(\theta_{\text{PIP}}) \cos(\theta_{\text{MCP}}) \sin((1+\lambda)\theta_{\text{PIP}}) \\ & + \cos(\theta_{\text{MCP}}) \sin(\theta_{\text{PIP}}) \sin(\theta_{\text{MCP}}) \cos((1+\lambda)\theta_{\text{PIP}}) \\ & + \cos(\theta_{\text{MCP}}) \sin(\theta_{\text{PIP}}) \cos(\theta_{\text{MCP}}) \sin((1+\lambda)\theta_{\text{PIP}}) \end{aligned} \tag{28}$$

Here the second and sixth as well as the third and seventh terms cancel each other out. The remaining terms can be rewritten as follows:

$$\begin{aligned} & \cos^2(\theta_{\text{MCP}}) \cos(\theta_{\text{PIP}}) \cos((1+\lambda)\theta_{\text{PIP}}) - \\ & \sin^2(\theta_{\text{MCP}}) \sin(\theta_{\text{PIP}}) \sin((1+\lambda)\theta_{\text{PIP}}) + \\ & \sin^2(\theta_{\text{MCP}}) \cos(\theta_{\text{PIP}}) \cos((1+\lambda)\theta_{\text{PIP}}) + \\ & \cos^2(\theta_{\text{MCP}}) \sin(\theta_{\text{PIP}}) \sin((1+\lambda)\theta_{\text{PIP}}) \end{aligned} \tag{29}$$

And then combined into:

$$\begin{aligned} & \big[\cos^2(\theta_{\text{MCP}}) + \sin^2(\theta_{\text{MCP}}) \big] \cos(\theta_{\text{PIP}}) \cos((1+\lambda)\theta_{\text{PIP}}) + \\ & \big[\cos^2(\theta_{\text{MCP}}) + \sin^2(\theta_{\text{MCP}}) \big] \sin(\theta_{\text{PIP}}) \sin((1+\lambda)\theta_{\text{PIP}}) \end{aligned} \tag{30}$$

We simplify this with Identity (21) and expand the $\cos(1+\lambda)\theta_{\text{PIP}}$ and $\sin(1+\lambda)\theta_{\text{PIP}}$ parts of the equation:

$$\begin{aligned} & \cos(\theta_{\text{PIP}}) \big[\cos(\lambda\theta_{\text{PIP}}) \cos(\theta_{\text{PIP}}) - \sin(\lambda\theta_{\text{PIP}}) \sin(\theta_{\text{PIP}}) \big] + \\ & \sin(\theta_{\text{PIP}}) \big[\sin(\lambda\theta_{\text{PIP}}) \cos(\theta_{\text{PIP}}) + \cos(\lambda\theta_{\text{PIP}}) \sin(\theta_{\text{PIP}}) \big] \end{aligned} \tag{31}$$

Working out these products then yields us:

$$\begin{aligned} & \cos^2(\theta_{\text{PIP}}) \cos(\lambda\theta_{\text{PIP}}) - \cos(\theta_{\text{PIP}}) \sin(\theta_{\text{PIP}}) \sin(\lambda\theta_{\text{PIP}}) + \\ & \sin^2(\theta_{\text{PIP}}) \cos(\lambda\theta_{\text{PIP}}) + \cos(\theta_{\text{PIP}}) \sin(\theta_{\text{PIP}}) \sin(\lambda\theta_{\text{PIP}}) \end{aligned} \tag{32}$$

The second and fourth terms again cancel each other out. Furthermore the first and third terms can be combined and simplified using Identity (21), leaving us:

$$\cos(\lambda\theta_{\text{PIP}})(\cos^2(\theta_{\text{PIP}}) + \sin^2(\theta_{\text{PIP}})) = \cos(\lambda\theta_{\text{PIP}}) \tag{33}$$

Finally we can combine all of the above to bring us to the formula presented in the paper:

$$\begin{aligned} d^2 &= L_{\text{PP}}^2 + L_{\text{IP}}^2 + L_{\text{DP}}^2 \\ &+ 2L_{\text{PP}}L_{\text{IP}}\cos(\theta_{\text{PIP}}) \\ &+ 2L_{\text{PP}}L_{\text{DP}}\cos((1+\lambda)\theta_{\text{PIP}}) \\ &+ 2L_{\text{IP}}L_{\text{DP}}\cos(\lambda\theta_{\text{PIP}}) \end{aligned} \tag{34}$$

B Experiment Data

In this appendix we will list the data we obtained during our experiments. We will both provide the numbers in table form for the graphs above as well as split up per finger.

B.1 Index finger graphs

The following graphs show our measurements for the middle finger over a thousand random distances. The index finger is responsible for most of the maximum values across all experiments of all fingers.

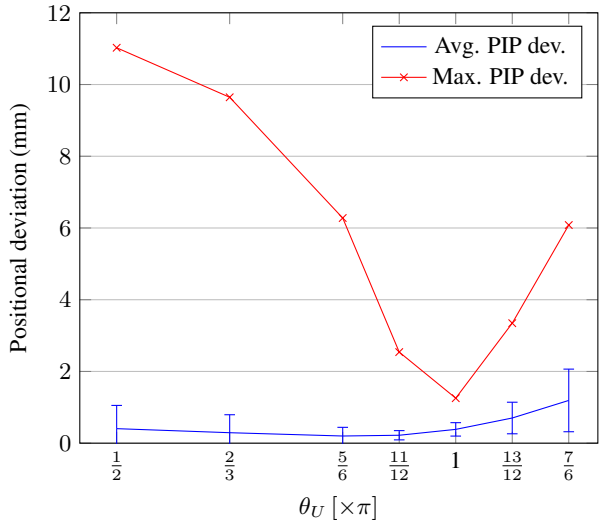

Figure 10: *The average, standard deviation and maximum per anchor point configuration for the PIP positional deviations of the index finger.*

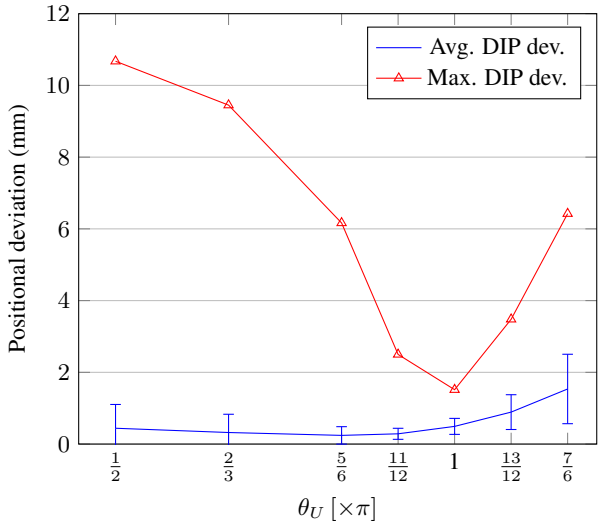

Figure 11: *The average, standard deviation and maximum per anchor point configuration for the DIP positional deviations of the index finger.*

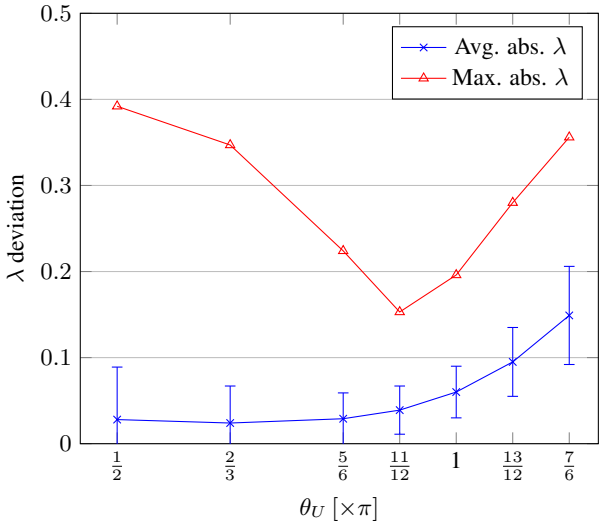

Figure 12: *The average, standard deviation and maximum per anchor point configuration for the absolute λ deviations of the index finger.*

B.2 Middle finger graphs

The following graphs show our measurements for the middle finger over a thousand random distances. The averages are very similar to the other fingers but the maximum values lie a bit lower than those of the index finger at the start. The final anchor point shows an increase in the maximum error however.

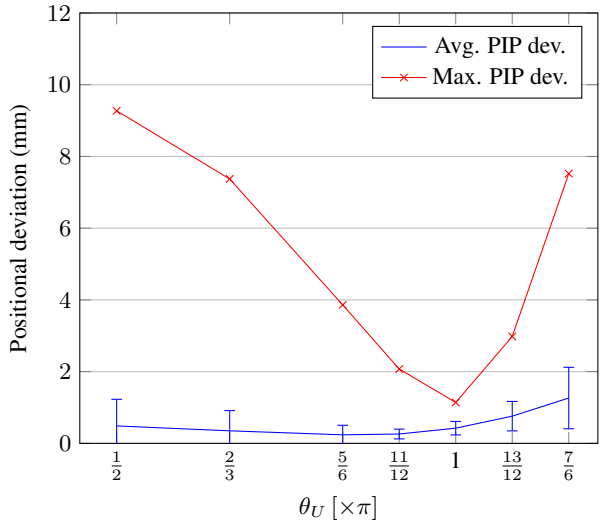

Figure 13: *The average, standard deviation and maximum per anchor point configuration for the PIP positional deviations of the middle finger.*

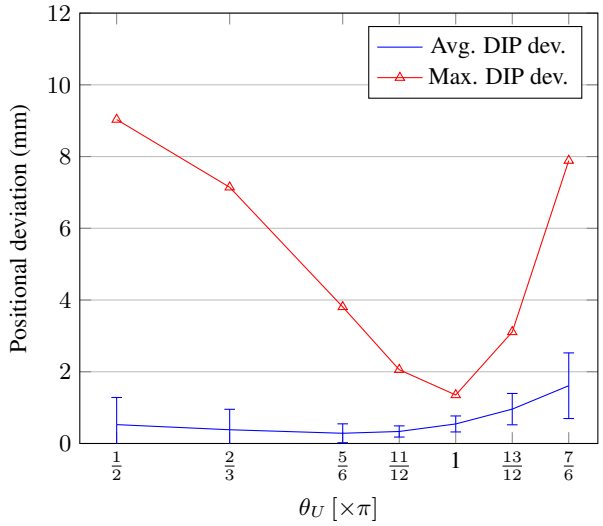

Figure 14: *The average, standard deviation and maximum per anchor point configuration for the DIP positional deviations of the middle finger.*

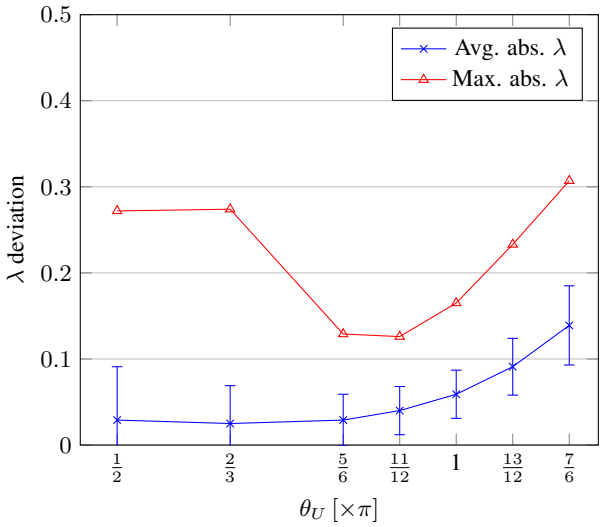

Figure 15: *The average, standard deviation and maximum per anchor point configuration for the absolute λ deviations of the middle finger.*

B.3 Ring finger graphs

The following graphs show our measurements for the ring finger over a thousand random distances. The averages are again similar to the other fingers. The maximum values start similar to the middle finger but end up lower near the high end of the anchor points. The lambda deviation has a peak at $\frac{2}{3}\pi$ but is otherwise similar to the other plots of the lambda deviation.

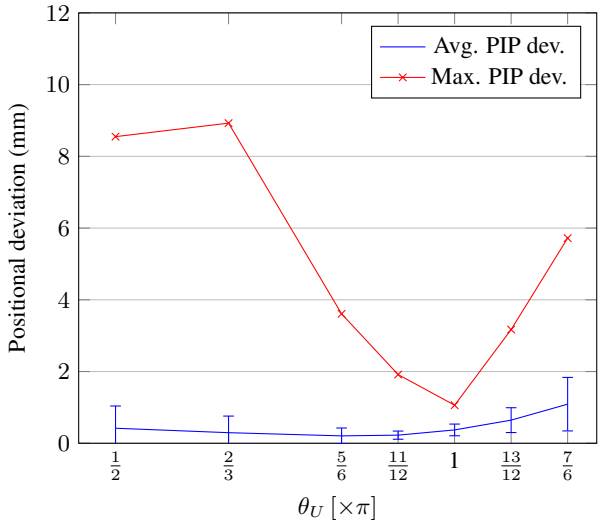

Figure 16: *The average, standard deviation and maximum per anchor point configuration for the PIP positional deviations of the ring finger.*

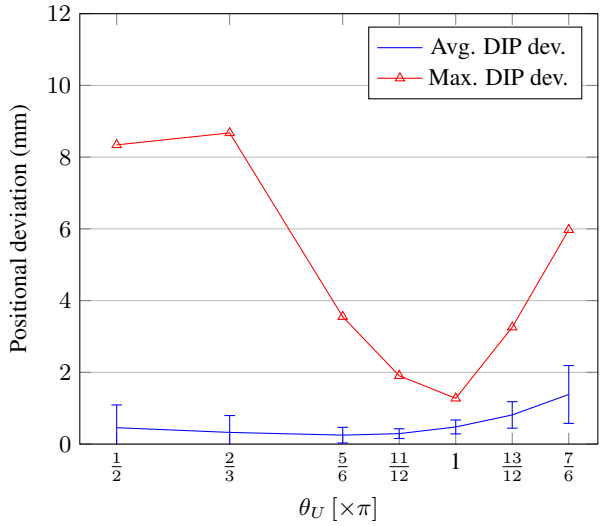

Figure 17: *The average, standard deviation and maximum per anchor point configuration for the DIP positional deviations of the ring finger.*

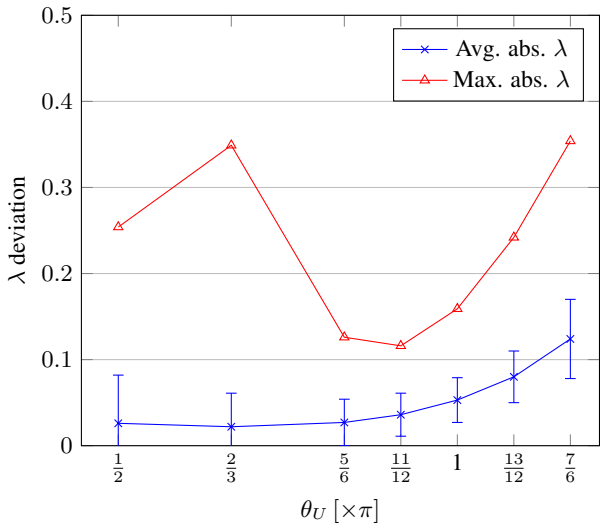

Figure 18: *The average, standard deviation and maximum per anchor point configuration for the absolute λ deviations of the ring finger.*

B.4 Little finger graphs

The following graphs show our measurements for the little finger over a thousand random distances. The averages are much lower when compared to the other fingers but follow a similar curve. The maximum values start much closer to the average but end up rising again near the high end of the anchor points. The lambda deviation is very stable in for the lower anchor points but like the other plots it rises on the higher end.

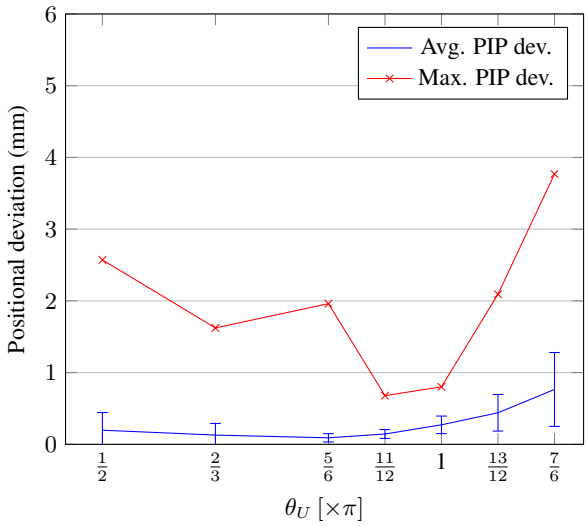

Figure 19: *The average, standard deviation and maximum per anchor point configuration for the PIP positional deviations of the little finger.*

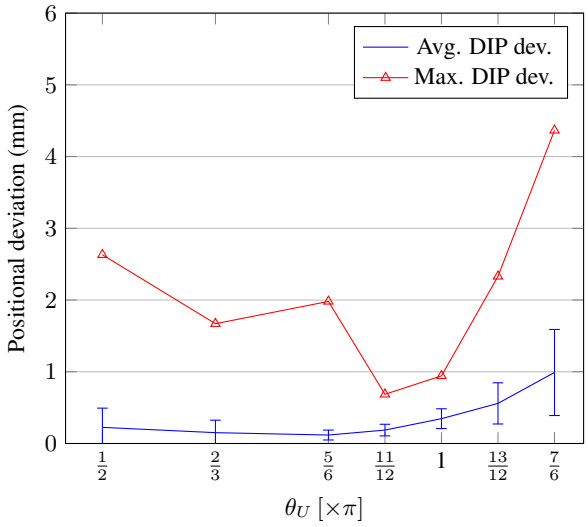

Figure 20: *The average, standard deviation and maximum per anchor point configuration for the DIP positional deviations of the little finger.*

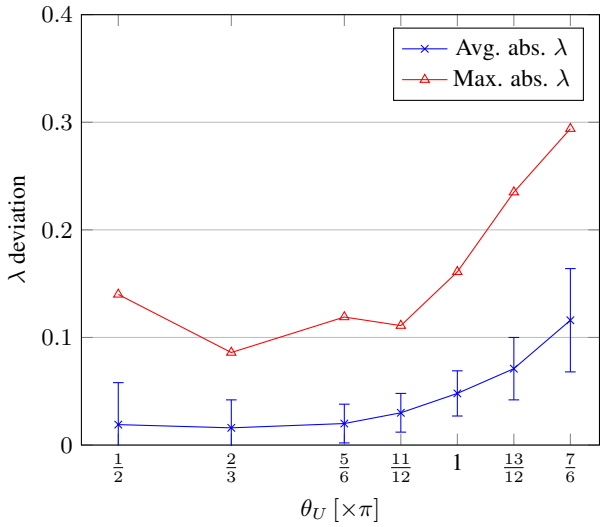

Figure 21: *The average, standard deviation and maximum per anchor point configuration for the absolute λ deviations of the little finger.*

B.5 Measurement data table

Below are several tables containing the averages, standard deviations and maximum values we found during our experiments for reference. First is the data of all fingers combined, followed by data values for individual fingers, starting with the index finger and continuing through to the little finger.

Total angle	PIP error (mm)	DIP error (mm)	abs. λ error
$1/2\pi$	0.377 ± (0.605) [11.025]	0.412 ± (0.620) [10.674]	-0.020 ± (0.030) [0.036]
$2/3\pi$	0.267 ± (0.457) [9.644]	0.295 ± (0.465) [9.450]	-0.005 ± (0.028) [0.064]
$5/6\pi$	0.184 ± (0.220) [6.278]	0.224 ± (0.222) [6.165]	0.018 ± (0.026) [0.122]
$11/12\pi$	0.213 ± (0.122) [2.540]	0.275 ± (0.146) [2.499]	0.034 ± (0.025) [0.153]
π	0.363 ± (0.176) [1.255]	0.465 ± (0.211) [1.516]	0.055 ± (0.027) [0.196]
$13/12\pi$	0.636 ± (0.389) [3.346]	0.805 ± (0.429) [3.479]	0.084 ± (0.035) [0.280]
$7/6\pi$	1.078 ± (0.785) [7.521]	1.380 ± (0.868) [7.888]	0.132 ± (0.051) [0.356]

Table 1: *The means, standard deviations and maximum values for the PIP and DIP positional deviations and absolute deviations of λ across **all** fingers*

Total angle	PIP error (mm)	DIP error (mm)	abs. λ error
$1/2\pi$	0.405 ± (0.648) [11.025]	0.441 ± (0.663) [10.674]	0.028 ± (0.061) [0.392]
$2/3\pi$	0.292 ± (0.501) [9.644]	0.322 ± (0.509) [9.450]	0.024 ± (0.043) [0.347]
$5/6\pi$	0.200 ± (0.242) [6.278]	0.242 ± (0.243) [6.165]	0.029 ± (0.030) [0.224]
$11/12\pi$	0.221 ± (0.129) [2.540]	0.286 ± (0.154) [2.499]	0.039 ± (0.028) [0.153]
π	0.385 ± (0.187) [1.255]	0.493 ± (0.223) [1.516]	0.060 ± (0.030) [0.196]
$13/12\pi$	0.702 ± (0.440) [3.346]	0.891 ± (0.485) [3.479]	0.095 ± (0.040) [0.280]
$7/6\pi$	1.192 ± (0.873) [6.082]	1.536 ± (0.968) [6.424]	0.149 ± (0.057) [0.356]

Table 2: *The means, standard deviations and maximum values for the PIP and DIP positional deviations and absolute deviations of λ of the **index** finger*

Total angle	PIP error (mm)	DIP error (mm)	abs. λ error
$1/2\pi$	0.486 ± (0.743) [9.272]	0.526 ± (0.757) [9.032]	0.029 ± (0.062) [0.272]
$2/3\pi$	0.350 ± (0.563) [7.372]	0.383 ± (0.570) [7.144]	0.025 ± (0.044) [0.274]
$5/6\pi$	0.238 ± (0.267) [3.861]	0.285 ± (0.263) [3.808]	0.029 ± (0.030) [0.129]
$11/12\pi$	0.261 ± (0.136) [2.073]	0.335 ± (0.156) [2.057]	0.040 ± (0.028) [0.126]
π	0.424 ± (0.188) [1.140]	0.545 ± (0.223) [1.352]	0.059 ± (0.028) [0.165]
$13/12\pi$	0.759 ± (0.411) [2.981]	0.958 ± (0.437) [3.110]	0.091 ± (0.033) [0.233]
$7/6\pi$	1.264 ± (0.857) [7.521]	1.611 ± (0.915) [7.888]	0.139 ± (0.046) [0.307]

Table 3: *The means, standard deviations and maximum values for the PIP and DIP positional deviations and absolute deviations of λ of the **middle** finger*

Total angle	PIP error (mm)	DIP error (mm)	abs. λ error
$1/2\pi$	0.419 ± (0.620) [8.549]	0.456 ± (0.634) [8.342]	0.026 ± (0.056) [0.254]
$2/3\pi$	0.295 ± (0.463) [8.926]	0.325 ± (0.470) [8.675]	0.022 ± (0.039) [0.349]
$5/6\pi$	0.206 ± (0.219) [3.608]	0.249 ± (0.218) [3.551]	0.027 ± (0.027) [0.126]
$11/12\pi$	0.226 ± (0.114) [1.916]	0.291 ± (0.136) [1.905]	0.036 ± (0.025) [0.116]
π	0.372 ± (0.163) [1.061]	0.477 ± (0.194) [1.275]	0.053 ± (0.026) [0.159]
$13/12\pi$	0.645 ± (0.346) [3.170]	0.812 ± (0.369) [3.258]	0.080 ± (0.030) [0.242]
$7/6\pi$	1.090 ± (0.746) [5.720]	1.383 ± (0.806) [5.974]	0.124 ± (0.046) [0.354]

Table 4: *The means, standard deviations and maximum values for the PIP and DIP positional deviations and absolute deviations of λ of the **ring** finger*

Total angle	PIP error (mm)	DIP error (mm)	abs. λ error
$1/2\pi$	0.197 ± (0.246) [2.571]	0.225 ± (0.268) [2.631]	0.019 ± (0.039) [0.140]
$2/3\pi$	0.129 ± (0.162) [1.623]	0.151 ± (0.174) [1.670]	0.016 ± (0.026) [0.086]
$5/6\pi$	0.091 ± (0.058) [1.961]	0.118 ± (0.069) [1.981]	0.020 ± (0.018) [0.119]
$11/12\pi$	0.144 ± (0.062) [0.679]	0.187 ± (0.081) [0.686]	0.030 ± (0.018) [0.111]
π	0.272 ± (0.122) [0.801]	0.346 ± (0.138) [0.943]	0.048 ± (0.021) [0.161]
$13/12\pi$	0.440 ± (0.255) [2.093]	0.559 ± (0.287) [2.330]	0.071 ± (0.029) [0.235]
$7/6\pi$	0.765 ± (0.515) [3.769]	0.990 ± (0.600) [4.366]	0.116 ± (0.048) [0.294]

Table 5: *The means, standard deviations and maximum values for the PIP and DIP positional deviations and absolute deviations of λ of the **little** finger*

C Literature Study

Below is the original literature study that preceded this paper.

C.1 Introduction

In this literature study we take a look at the basic principles, recent advances and state of the art techniques related to human grasping. The aim is to obtain an overview of what is currently possible in simulating this difficult task in the field of robotic manipulation as well as in computer animation. Topics of the former include how to form proper grasps, how to measure the quality of such a grasp, how to model the hand and contacts as well as planning and executing the grasps. Topics of the latter will deal with obtaining motion data of human hands performing dexterous tasks to replicate these in a virtual environment.

By studying the existing techniques we can create a framework that is capable of determining grasps for any object in a virtual world and choose an appropriate animation to execute the grasp motion. This framework could then be expanded by adding better animation techniques that do not require a large volume of pre-recorded animations to be matched as well as planning techniques that allow for grasps capable of executing more difficult tasks.

The study is divided into the following sections: Manipulation with subsections for grasping, grasp synthesis, soft finger contacts, hand models, quality analysis, planning and execution; and Animation with subsections for motion capture of hand motions and animating hand motions. Each of these topics are essential in finding and animating realistic grasps for push-grasp tasks.

C.2 Manipulation

Manipulation is a key area in robotics and deals with anything related to physically changing the world around us. This can be as simple as moving an object, which will be our main interest, to something more difficult such as joining objects by welding or gluing. Whilst many of these actions make use of the same basic principles, such as grasping and moving, most of them will need additional techniques to complete the task to be performed. As the goal of this project will involve movement only, this literature study will focus only on manipulation forms that move objects in an environment.

An important distinction that is made with regards to manipulation tasks is whether they are prehensile versus nonprehensile. As the definition of prehensile states "the quality of an appendage or organ that has been adapted for holding or grasping", prehensile manipulation encompasses anything that grasps an object firmly so that we can direct the manipulation as we desire. Examples include picking up an object and moving it elsewhere, rotating something in place with our fingers but also hanging by a tail to move around.

Nonprehensile manipulation contains manipulations tasks that are executed without grasps. Examples include pushing something out of the way with the back of the hand or kicking a ball. Nonprehensile manipulation is done by applying a force to the object to make it move as desired, without necessarily doing anything to bring it back to a rest state ourselves. See Figure **??** for an example. While this form of manipulation is well suited for pushing objects, we will be interested in actually grabbing an object in a realistic fashion to move it. As such we will focus on prehensile manipulation techniques.

Figure 22: *A robotic hand developed by DLR. Image from* `http://www.dlr.de/rmc/rm/en/desktopdefault.aspx/tabid-3975/6161_read-243/`

C.2.1 grasping

Having made our choice for the type of manipulation that we will be using, we will now look into the various elements that make up a good grasp. Within the literature of mechanics and robotics, grasping with a robotic or even human-like hand is a field that has received a lot of attention during the last few decades. Some of the works that are still relevant today date back to more than half a century. The general consensus is that in order to grasp something we need to fully constrain its movement. Humans learn how to do this at a young age and can easily adjust their grasp if needed. In order to allow robots to do the same, we need a way to tell if a grasp will succeed in constraining an object as well as a metric of how well it does so.

There are three main questions in regarding grasping an object. The first is analysis, where given an object and a set of contact locations, we wish to determine if these contacts fully constrain the object according to some definition of closure. In the literature it if often assumed we use frictionless point contacts. This gives us a conservative case as friction can greatly increase the forces we can exert on the object to constrain it. Friction is however not always applicable to the type of analysis used.

C.2.2 Analysis

The analysis can be done by looking at how the placement of the contact points constrains the motions that are possible for the object without moving or intersecting the contact point in any way. With this analysis we can look at first order effects, such as velocities or forces and moments, to constrain the object such that no instantaneous motion is possible. By incorporating second order effects, such as curvature, we obtain a stricter version of grasping that can reduce the fingers needed to grasp an object significantly. Two notions have been widely accepted to determine whether a grasp constrains the object, namely form closure and force closure. The meaning of these conditions are however interchanged between different works by different authors and as such we will provide the definitions that we will follow.

Form closure is credited to the work of Reuleaux in [Reuleaux 1875]. It is defined in [Bicchi and Kumar 2000] as: "Form closure is a condition of complete restraint that can resist any external

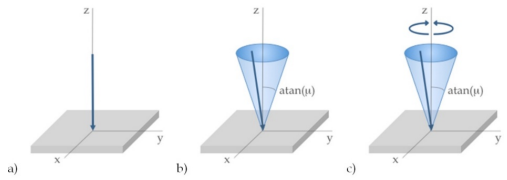

Figure 23: *An example of forces that can be applied by a hard point contact without friction, with friction and finally a soft contact. Image from* `http://www.intechopen.com/books/theoretical-biomechanics/towards-a-realistic-and-self-contained-biomechanical-model-of-the-hand`

wrench applied on the grasped object regardless of the magnitude of contact forces". In other words, the contact points restrain every motion of the object, even infinitesimal ones, due to collision. Form closure provides an analysis from a velocity perspective, limiting all velocities that can be applied to an object.

Further works has been done on form closure more recently where second order effects were taken into account. In [Czyzowicz et al. 1999] a geometric analysis was used for mostly 2D cases to include 2^{nd} order effects in the analysis. In [Rimon and Burdick 1998] the authors used the configuration space, which describes a location and orientation of an object, to determine a 2^{nd} order immobility grasp. While techniques are available for form closure grasps in 2D, the 3D case is not well studied. As such the main focus of our study will be on force closure.

A force closure grasp is defined in [Bicchi and Kumar 2000] as a grasp that is in equilibrium for any arbitrary wrench applied on the object. Equilibrium is defined by the following properties: the sum of the contact and external wrenches adds up to zero and the contact forces and moments are nonnegative and follow a law of friction if applicable. A wrench is a 6-dimensional vector describing a force and a moment along a so called screw axis, which defines a line of action as well as a force and moment working on this line.

The difference with form closure, according to [Bicchi and Kumar 2000], is the notion that this external wrench can be negated by a combination of non-negative contact forces and moments. It can be seen as an analysis from a force perspective where the contacts will push back against the object to maintain the equilibrium. Force closure can also be extended to frictional fingers, as the frictional forces can also be modeled with wrenches. This simply provided a greater range of possible forces per contact. The definition of this condition is often credited to the work of Nguyen in [Nguyen 1986]. In his work he also presented various algorithms to determine independent regions where a contact point could be placed. If a contact was placed in each region, the object would be held in force closure.

An interesting note to make is that a grasp holds an object in first order form closure if and only if it is also held in force closure, when using frictionless fingers, in the planar case. When applying second order effects to the form analysis however, this no longer applies. Even though we have focused here on full closure analysis of grasps, it is also possible to create a partial closure grasp, where one or more directions of velocity or force are left unconstrained. This could be of interest to our research as we want to push an object across a surface. Having the direction of pushing unconstrained could certainly provide useful.

C.2.3 Existence

The second is existence, where given an object and any additional constraints that will apply to the grasp, is there a set of contacts that will fully constrain the object. This question has been used to prove upper or lower bounds on how many contacts are needed to constrain an object. These bounds are dependent on the object at hand and what attributes of it are used.

Examples of different situations include: 2D vs. 3D, the presence of friction and using curvature or other object specific geometry. For some of these more contacts are needed, such as the 3D case, where others reduce the amount needed, such as friction. These bounds are useful to have in order to design robotic hands capable of grasping any object that complies with the attributes used in the design process.

Extensive research over the years has provided us with upper bounds on how many fingers are needed to constrain an object. In most cases these numbers will suffice, but some objects with curvature or special shapes may require additional contacts to fully constrain them. Proofs of these upper bounds have been provided in several papers, such as [Markenscoff and Papdimitriou 1989] for the frictionless case.

In the planar case closure can be obtained using only four fingers. When applying second order effects to form closure analysis this can be reduced to three fingers for most objects. In a 3D environment, seven fingers will suffice to hold an object in force closure with frictionless contacts. Using second order effects for form closure, only four fingers will be necessary to grasp most 3D objects successfully.

C.2.4 Synthesis

Finally there is the synthesis question, where given an object and any constraints, the goal is to find a set of contacts that will result in a fully constraining grasp. With regards to synthesis there are several variants regarding the amount of sets that are found. Some techniques will find a single grasp that holds the object firmly. This is different from the existence question where we are interested in whether it is possible to hold an object. With synthesis we are interested in the actual placement of the contacts given our constraints, so we can use it to perform a grasp.

Other techniques may find multiple sets or regions in which a contact could be placed independently of the others and constrain the object. This could allow for some errors in positioning the contact points. Similarly by increasing the coverage of these regions a grasp should become more stable. Alternatively the grasps generated could be analysed individually to determine which one is best suited to the situation if the approach of the grasp and other obstacles are included.

Lastly there are techniques that deal with finding all possible ways to grasp the object and constrain its movement. This provides us with the widest range of possible grasps. It may however be more difficult or time consuming to construct all possible grasps. In some applications this may not be needed and the previous type of techniques could be sufficient.

In this study we will be mainly interested in the synthesis question, as having access to multiple or all possible grasps of an object will allow us to select one that fits best to the task at hand. In a subsequent section we will describe methods that will generate grasps that firmly hold the object according to the notion of closure. Seeing as we are interested in moving an object by applying a force to it via the contacts, force closure appears like the more intuitive

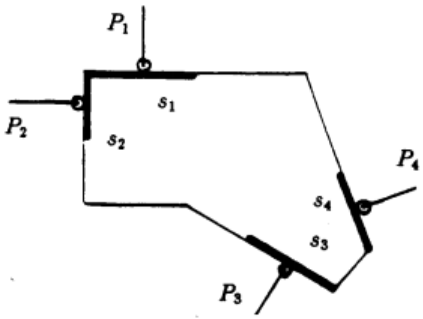

Figure 24: *An example of a generated force closure grasp. Image from [Nguyen 1986]*

condition to achieve. A lot of research has already been done on force closure grasps, as will be discussed in a subsequent section.

C.2.5 Contact properties and grasp quality

Of great importance is the type of contact, such as a point, line or surface, as well as whether friction is involved or not. These properties can greatly increase the difficulty of creating a proper grasp, either because more contacts are needed or the analysis of a grasp becomes more difficult as it becomes less straightforward. Frictionless point contacts can only exert a force or velocity in a single direction. A grasp that works with frictionless point contacts will always work with larger contact surfaces or friction, as is found in most real-life applications. In a subsequent section we will describe how to handle the contact models used in human hand grasping. The effects of the contact model on synthesizing a grasp will be discussed in the synthesis section.

To determine what grasp is best out of all that fulfill some criteria of closure we will need a metric to analyze the quality of the grasp. This metric should quantize a value for the grasp that tells us which is better at performing the task at hand. Examples of a metric include how much external force the grasp can counter or how much force the grasp can execute on the object itself. In a subsequent section we will present some examples of techniques that analyze the quality of a grasp according to such a metric.

C.2.6 Force closure grasp synthesis

There exists a lot of literature on the synthesis of a grasp capable of holding an object in force (or form) closure. As listed before Nguyen in [Nguyen 1986] has provided various algorithms that work in 2D and 3D. For the 3D case he has provided alternatives based on the contacts that are used, namely two soft-finger contacts, three hard-finger contacts and seven frictionless point contacts. In [Ponce et al. 1996] an early technique is presented to generate grasps on 3D objects with four frictional fingers.

In [Sahbani et al. 2012] the authors provide an overview of synthesis algorithms specifically for 3D objects. As the paper dates from 2012 it contains progress of over a decade since the overview provided by Bicchi. They subdivide the techniques as either analytical or empirical. The former uses kinematic and dynamic formulations to determine contact locations that satisfy the task requirement such as force closure. The latter tries to mimic human grasping by selecting an appropriate learned grasp based on the task and object. These techniques can be subdivided in either recreating the motions made by a human teacher by tracking the motion with a data glove or computer vision, or by analyzing the object to be grasped to determine a grasp suitable to its properties. These techniques are intended to find a grasp that holds an object in force closure, rather than to execute a specific task and will not be considered here.

Of the analytical systems discussed, the ones that are task-oriented rather than only force closure are of interest to us, as they will enable us to generate grasps capable of pushing the object in the desired direction. One that stands out is [Prats et al. 2007], where the authors develop a system that takes the task to be performed into account in the early planning stages. Specifically they select their grasp based on the task they need to perform by aligning the direction in which a force needs to be applied with a hand preshape capable of doing so.

They define four preshapes: the power hook, the precision hook, the precision grasp and the cylindrical grasp. The hook grasps use the fingers or fingertips to push on a surface. The precision grasp uses the fingers and thumb to enclose the object. The cylindrical grasp does the same as the precision grasp but also includes a contact point on the palm for further restraining if there is enough space available. Each of these grasps has a task frame which defines the direction in which the grasp can exert the most force.

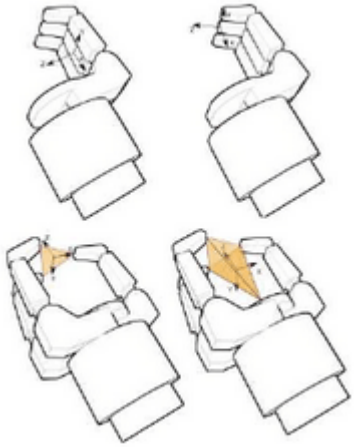

Figure 25: *The preshapes described in [Prats et al. 2007]*

By moving the hand to the object, such that the task frame associated with the preshape approaches the object and aligns with the target frame along the Z axis, they grasp the object by moving closer and closing the fingers until collision. This results in the grasp defined by the preshape and by pushing in the direction of the Z axis in the defined task frame they exert the desired force on the object. See [Prats et al. 2007] for a more in depth explanation of the technique.

By combining this with the taxonomy provided by Feix in [Feix et al. 2009] we can identify grasps types that are useful in pushing objects along a surface. For these grasps we can define a preshape that when closing the fingers will result in a grasp and use these preshapes to plan a task oriented grasp as in [Prats et al. 2007]. This will limit us in the types of grasps we can perform, because we choose a subset of the taxonomy. The taxonomy itself is also not necessarily the complete set of grasps humans are capable of. It is however a representation of the most common grasps we perform on a daily basis and several are already fairly specific to single object. And seeing as how most humans will grab an object based on its looks and weight and previous gained experience, they will likely defer to a set of grasps that are adjusted only slightly if needed.

Having a solid base of grasp motions that need to be recorded or programmed keeps the complexity low whilst improving the quality

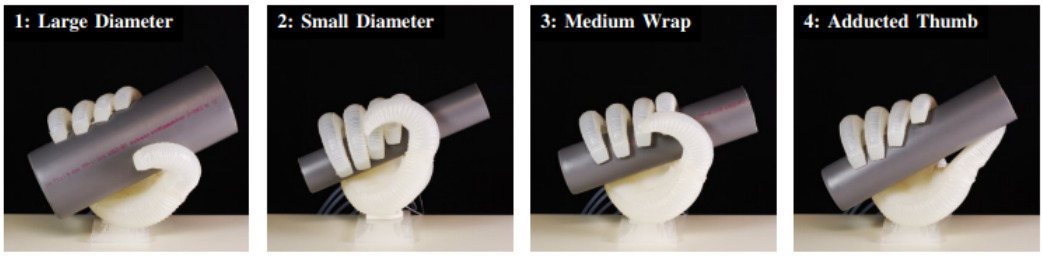

Figure 26: *A few examples of the taxonomy provided in [Feix et al. 2009] performed by a robot hand described in [Deimel and Brock 2014]*

of the animations significantly. While a system capable of creating any grasp that is possible with the human hand would be interesting, the complexity of calculating grasps as well as how 'realistic' the grasp looks becomes so high it is difficult to guarantee an acceptable outcome. By limiting us to good, realistic grasps applied in a smart and predictable way we can control the outcomes much better.

An alternative is presented in [Kruger and van der Stappen 2011] where the authors have opted to generate grasps that create a partial force closure grasp. This will prevent the object to move in any undesired direction, but allows unrestrained movement along a task wrench. Their methods, as well as the metrics provided to analyze the quality of the grasp, are designed around one specific task wrench. This simplifies the problem and allows for a clear measurement, but does not guarantee that the grasp is capable of exerting different forces on the object. Nonetheless their method is capable of defining all sets of features where a contact should be placed such that a contact needs to exert at most a predefined maximum force to exert the wrench on the object. This too would allow us to define grasps in such a way that we can push an object along a table.

Although the above methods provide us a good approach to finding a grasp capable of moving the object, we will need to use a good method to model the soft contacts provided by human fingers. Doing so will enable us to use grasps of the taxonomy in different ways than just enclosing the object in a force closure grasp. We could then make use of the friction to apply a force to the object in the direction we want, which may not always align with the preshape options defined above. This could allow us to select grasps that are perfectly viable but maybe not straightforward for the preshape method. As such we will now look into how to model the fingers.

C.2.7 Soft finger contact models

As mentioned before the contacts that are used in the application or by the robot performing a grasp are of great importance in determining whether a grasp will fully constrain an object. With hard, frictionless point contacts, we are able to exert a force in a single direction on the object. Several of these contacts will be required to ensure we can negate any external wrench on the object.

By adding friction to these contacts, we can also exert tangential forces on the object, provided the force directions lies in the friction cone with the center on the contact point. Most works use Coulomb's law of friction, which is not the most accurate representation of friction, but will suffice in most situations and is easy to compute. The friction cone is then defined by a half angle in regards to the surface normal at the contact point in the 2D case. For a 3D application this friction cone is often approximated with a union of pyramids for ease of computation.

The soft fingers of a human have an added property in that they can apply a torque due to friction around the surface normal at the contact location. This is due to the deformation of finger, which complies with the shape of the objects and creates a surface will 'stick' to the object. By rotating the object, the finger will resist the motion as it is capable of rotating with it, but will only stretch so far. Regular hard contacts are unable to stretch in such a way and will simply slide against each other.

This property is of great importance in grasping an object with only two fingers. By grabbing an object on two opposing sides with frictional hard contacts, we can account for forces in all directions as well as any torque, except for a torque along a surface normal at a contact location. Seeing as such a grasp will likely orient the object such that both contact surface normals align, this will allow the object to rotate along this axis, until gravity puts it in balance or the grasp is broken because the motion of the object exceeds what the grasp can counteract. Having soft finger contacts will allow us to counter this torque directly and reduce the amount of contacts needed greatly.

An important tool in modeling soft finger contacts is the frictional ellipsoid. As frictional forces are limited according to the contact force and a coefficient of friction, the ellipsoid describes the maximum combination of frictional forces and torques the contact is capable of execution. In [Ciocarlie et al. 2007] the authors apply this frictional ellipsoid by calculating the properties of the contact areas of the grasp. They use the relative radii of the contact surfaces to determine an important element of the frictional ellipsoid equation to adjust its shape based on the present contacts. This allows them to model the friction much better and as a result determine if a grasp will provide force closure more accurately.

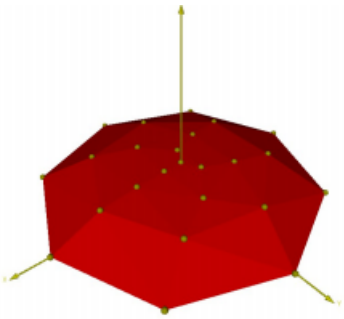

Figure 27: *The fricitonal ellipsoid with the planar axes representing tangential friction forces and the vertical axis representing the frictional torque as described in [Ciocarlie et al. 2007]*

In [Barbagli et al. 2004] a comparison is made between classical Hertzian models describing elastic surfaces as well as more novel

techniques that describe the deformation of a soft contact. This allows them to model the resulting frictional forces that are found in real human fingers more accurately. They provide an overview of how the models compare in different circumstances. Finally they show an implementation of a soft finger contact using point contacts to efficiently simulate the contact forces.

The modeling of the finger contacts is important in determining if a grasp is capable of holding an object in force closure and therefore be capable of manipulating the object as desired. In order to determine if a grasp is possible however, we will need an accurate model of our hand, as will be shown in the next section.

C.2.8 Hand models

In order to synthesize and validate our grasps we need an accurate model of the hand that will perform the grasp. In our research we will focus on the complex human hand, but in the literature a wider variety of hands is used, as robot hands can be adjusted to the task they will need to perform.

Many robot hands that are used in for example assembly lines follow the RISC principle. This stands for Reduced Intricacy in Sensing and Control, from the work of Canny and Goldberg such as [Canny and Goldberg 1993]. The aim was to make use of reliable actions and simple parts to reduce the cost of construction and maintenance of these robots. Other assembly line robots have a selection of tools on their end effector that they can switch between, depending on the task at hand. Tools can include grippers, welding and cutting equipment. For more complex tasks such as service robots or full humanoid bodies that are capable of executing tasks currently performed by humans, a more intricate hand will be required.

A human like hand will have many degrees of freedom as the fingers can stretch and bend and the opposable thumb allows for a great amount of configurations that can enclose an object. The reachable space of each fingertip is limited however as the finger can mostly bend along parallel rotational axis. There is also a limit to the abduction and adduction the fingers are capable of. This is an important fact to remain aware of as not every grasp generated by various synthesis techniques is achievable with a human hand. The contact points may require two hands or a different range of possible motions of the individual fingers to reach all the calculated contact points. As such it is important to either validate the grasp with the hand or design a synthesis technique with the hand structure in mind. The remainder of this section will describe two hands capable of dexterous manipulation.

An example of a widely used hand model for more dexterous manipulation is the DLR Hand 2 as described in [Butterfass et al. 2001]. This robotic hand has three fingers and an opposable thumb to allow it the same type of grasps as a human. The three fingers will suffice in most scenarios as the fourth finger of a human does not add anything special. The hand is engineered such that a lot of computational power is already available in the hand itself, reducing the amount of wiring needed. It is also made of a softer material that can be easily removed for maintenance and simultaneously allows for soft contacts. Due to the human likeness the hand is capable of almost all grasps a human could do. A lot of research is going into creating action sequences capable of execution specific grasps.

A different approach was applied in [Deimel and Brock 2014] to create a compliant, under actuated hand. The hand is made up of soft materials such as silicone rubber and is pneumatically actuated. The hand is subdivided into 6 components that can be inflated which causes the element to elongate in a specific direction that is not reinforced to resist such a motion. This allows the fingers to be

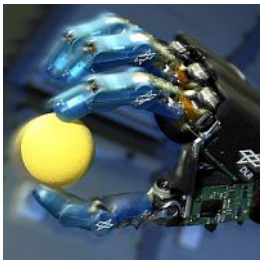

Figure 28: *The DLR Hand II grasping a ball with a two finger pinch grasp. Image from* `http://www.dlr.de/rm-neu/en/desktopdefault.aspx/tabid-3802/6102_read-8923/`

stretched and fold around an object. By inflating the two palm components the thumb can be moved into position. The hand is capable of touching each fingertip with the thumb as well as performing 31 out of the 33 grasps in Feixs taxonomy in [Feix et al. 2009]. Even though the hand has a lower amount of actuation elements it is still capable of performing well. The authors state that the compliance of the hand helps in reducing the complexity of the hand as well as improving its capabilities.

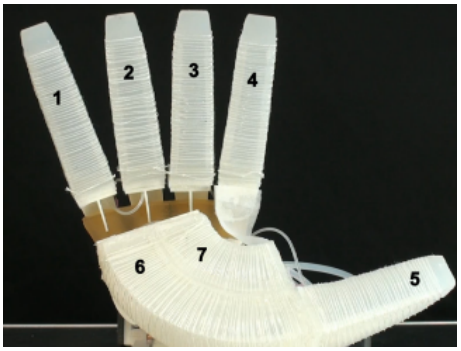

Figure 29: *The hand used in [Deimel and Brock 2014] with the numbers indicating the parts that can be actuated*

C.2.9 Grasp quality analysis

With all the elements that are important to consider in the synthesis of a grasp now covered, we turn our attention to the topic of quality analysis. While it is nice to have a technique capable of generating multiple options for the task we wish the grasp to perform, if we are unable to determine which is best suited for the task we might as well stick with generating a single grasp. As such several works in the literature deal with quantifying the quality of a grasp. In the literature overview provided in [Sahbani et al. 2012] several techniques are listed that relate to grasp quality and task compatibility. These techniques provide a way to determine how well a grasp is capable of executing a task.

One of the more popular metrics is the one developed by Ferrari and Canny in [Ferrari and Canny 1992]. They use a ball fitting technique in the wrench space. They fit the largest ball possible in the wrench set that the contacts can apply on the object. This provides them with the worst case wrench that the grasp can still resist. The greater the force or torque needed to break the grasp the better the grasp is considered to be. This can be analyzed with either the sum or the maximum finger force that is required to undo this worst case wrench.

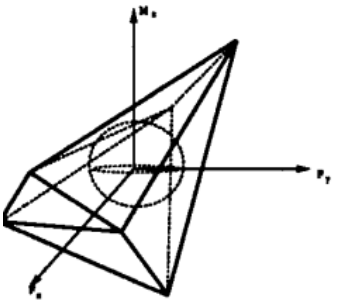

Figure 30: *An example of the analysis from [Ferrari and Canny 1992] of a 2D grasp in 3D wrench space*

In [Kruger and van der Stappen 2011] the authors provided a synthesis technique to create partial force closure grasps, as discussed previously. In their paper they also provide a measuring technique based on that of [Ferrari and Canny 1992] but adjusted for their specific case. Both the maximum and sum metric are adjusted to work for a partial force closure grasp to provide a meaningful insight into the ability of the grasp to perform the task it was designed for.

In [Ciocarlie et al.] the authors extend previous work to incorporate soft finger contacts to the equation. They analyzed whether how the quality of a grasp changes as the finger loses contact with the object by going from a flat contact to a more pinch like contact. The result was that with a greater surface less force was required to perform a grasp at the same contact points. Their technique uses the Finite Element Method to simulate the total frictional forces and torques between the two bodies in contact. Increasing the contact force will increase the contact surface made with the soft finger, which allows for greater frictional forces and torques to be applied. This can be used to increase the quality of a given grasp when using soft contacts.

Finally in [Berenson et al. 2007] the authors argue that many grasps that achieve force closure according to existing techniques and measurements are not necessarily the best grasp available. They provide various examples where synthesized grasps hold the object in a rather awkward way. They also argue that by executing a grasp there are often collisions that cause the object to be in a different configuration than what was initially assumed. This leads to a grasp perhaps not being the most effective as the situation has changed. They developed a system where they included dynamics, to simulate the effects of touching the object as the grasp was being performed, to analyze the final outcome of their grasp. This resulted in grasps that are less likely to move the object or are more robust to it by using power grasps where the fingers fully enclose a part of the object.

By analyzing the quality of the grasps we synthesize, we can pick one that will not only perform the required task of pushing the object across a surface, but also do it better than other grasps. This way we can select a grasp that is not just about capable of executing the task wrench, but one that can do it easily and hopefully more realistically. We could also require that the grasp is easy to perform in real life, rather than just in a virtual world where we have complete freedom, by taking into account collisions and other objects.

C.2.10 Grasp planning and execution

In our research topic we will not include the full body that the hand is attached to. We may also not look at planning the approach of the grasp but rather matching an animation to the grasp that we have chosen. Nevertheless is this an important part in robotics, as just selecting a grasp is not sufficient. The object could be surrounded by other objects that would prevent certain grasps from being performed. Similarly choosing a location for the contacts and closing the fingers until collision is simple enough. Actually getting the hand in that position is another tricky problem.

In [Rijpkema and Girard 1991] a system is provided that uses a knowledge based approach. The system analyses the object and picks a known grasp to pick the object up. They have simulated the entire process including the planning and executing of the task. They state that an approach is most often done with a predictable shape, such as a straight line or smooth curve that stays clear of any objects in the way. This implies that it is probably best to get as close to the object as possible by moving the arm through the air rather than close to any obstacles and approach an object from above if possible.

In [Berenson et al. 2007] the authors have developed a framework that picks a feasible grasp out of a set of generated grasps. Like many other methods they will analyse the object offline and determine a set of force closure grasps on the object that score well with traditional metrics as described above. They then check these grasps online for feasibility, such as grasps that require the contacts to be placed on the bottom for an object that rests on a table. This would be an impossible task for a robot to perform and thus the grasp is excluded from further testing.

This second test looks at the environment as well as the capabilities of the robot to exclude any grasp that is unlikely to succeed. By ranking the generated grasps according to this score, the framework can perform path planning and collision testing for the better candidates first and only moving on to a lesser grasp if these grasps do not perform well enough after planning them out. This allowed them to perform complex tasks with little time wasted on testing grasps at random, which is what would happen if we only synthesized grasps that are force closure without looking at the environment.

C.3 Animation

With the manipulation part of our research area covered, we will now turn our attention to the animation aspects of the system. Although the grasp synthesis will likely fill a larger part in the overall framework, the animations are what will ultimately determine the quality of the system. No matter how good a grasp we can find for an object, if we cannot animate it realistically it seems like wasted effort.

We intend to improve upon the often seen manually created animations where an object just 'sticks' to a hand pose that is usually not adapted to the object being held. For this we will need animations that comply with the grasp we have calculated using the techniques discussed previously. To do this we will look at current techniques to capture realistic hand animations as well as performing such animations based on active control. Before that however we will provide a quick overview of a few important elements of animation in the next section.

C.3.1 Animation basics

Within animation some concepts are key to many if not all systems that are currently developed. These concepts are key frames and the skeleton. While almost all 3D applications with articulated characters or elements will use skeletons, not all systems will use key frames. Some systems instead opt to use so-called dynamics to control the character's motion. Each of these will be discussed shortly below for completeness.

C.3.2 Skeleton

The skeleton of a character is a system that has been implemented to reduce the complexity of creating animations. As computing power has increased over the years so has our ability to create 3D models or polygon meshes of increasing detail. Each character or object in a 3D environment is nowadays represented by many vertices that are grouped into polygons. These polygons, often triangles, are then given a color on their surface according to some texture and lighting to create a realistic virtual model. If we want to animate the individual vertices to make an object move, we would need a lot of data and calculations to make this happen without errors.

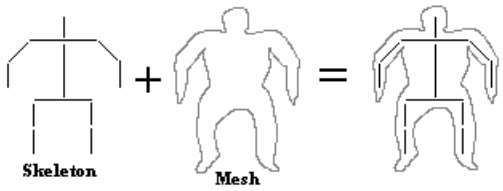

Figure 31: *Basic representation of the skeleton of a 3D model that is used to animate the entire mesh. Image from* `http://www.gamedev.net/tags/ccs/list/`

To reduce the complexity we make use of a skeleton of the model. This skeleton is built from links that are connected through joints. The skeleton's links themselves are rigid, so they do not bend or stretch. The joints can either rotate or translate along an axis, which is called a degree of freedom or DOF. A joint can have multiple DOFs or just a single one. Some joints could even be considered fixed, having zero DOFs. This is useful if a character or object needs a curved or bent part that does not allow motion within itself. By controlling the joint parameters such as rotation or translation magnitudes and directions we can approximate the workings of a human skeleton.

By assigning the individual vertices of the model to various links of the skeleton, we can animate a character by calculating the orientation and position of the skeleton links and applying a transformation to the vertices such that their distance to the skeleton link does not change. This transformation is often part of a hierarchy where a so called root joint is used to determine a 'global' position and orientation. From there we follow the hierarchy to the adjacent joints and apply an additional transformation based on the used DOFs of this joint. This will transform a part of the skeleton local to the parent joint.

By following this all the way through to the final joints, often called end effectors, we can animate a character with a vastly reduced set of values that only describe the DOFs of the joints of the skeleton. This means we can save a predefined animation with only a few values rather than a positional value for each individual vertex of the model for each time step. That saves a lot of storage space for a single animation, though it relies heavily on the transformation calculations to properly animate the character. There are however good techniques to work with multiplying the transformation matrices and reducing the errors that are often part of discrete mathematics.

C.3.3 Key frames

The second important concept in animation is a key frame. A key frame is a still of a particular scene, or more precisely all the information needed to recreate that particular scene whenever necessary. An example would be to take a photograph of someone running. At the instant the photograph is taken, the angles of the various joints in the human body are snapshotted as they are at that particular moment in time. By bending the body accordingly the pose can then be reconstructed.

Within computer animations many motions are stored as a collection of key frames. They detail various poses that are part of the complete motion, usually taken at regular intervals. By interpolating between the key frames as necessary we can obtain a smooth motion that is as close to the original as possible. The type of interpolation and number of key frames are of course important in determining the accuracy or smoothness of the motion. Lack of key frames can make it difficult to properly capture say the trajectory of an arm when throwing a ball. Similarly a linear interpolation could be insufficient when recreating a specific velocity if the key frames are taken at a regular interval. An interpolation type that can adjust itself to a speed or even acceleration would be better to do this, as would including more key frames. The choice will depend on whether memory or computational power is going to be the limiting factor.

By storing the orientation and position of the joints of a skeleton we can recreate any pose necessary for the entire character. For example a running motion would be a cyclic motion where the leg, knee and feet are bent and stretched in a pattern with the torso rotating back and forth along the axis of gravity. By storing a number of key frames of such a cycle and applying them to a skeleton in order we can create an animation of this particular motion. Early on these animations would be created by hand and key framed manually. These days however most applications rely on animations from motion capture, where we track a set of markers on a person with a setup of multiple cameras. The multiple viewpoints allow us to calculate the 3D positions of these markers. With this collection of key framed markers, we apply inverse kinematics. This is a technique where given a set of locations that we want parts of the body to be, we obtain the orientations of the joints that allow us to best match this pose. This is the opposite of forward kinematics, where we apply a set of rotations to the joints and derive the pose of the body from this. With inverse kinematics we can store the orientations as key frames and use later on for animating a character.

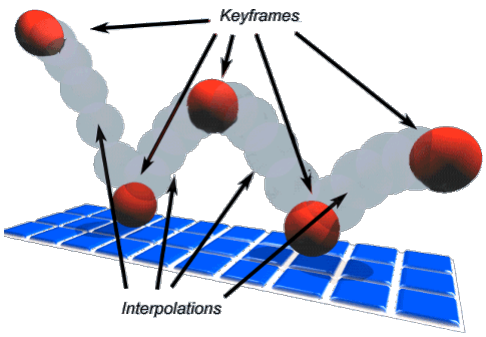

Figure 32: *Example of keyframes of a bouncing ball animation. Image from* `http://www.erimez.com/misc/Softimage/tutorials/si_help/introduction/si_uk_motion_intro.htm`

C.3.4 Dynamics

While the use of dynamics for animation can provide a nice approximation of the way we bend our body at the various joints, it is not what we intend to use in our final framework. We will briefly explain it here as it has become more important in recent years. Rather

than storing the individual angles of the joints at various key frames and interpolate between them to recreate poses, we intend to move the body by assigning velocities to the joints.

For each DOF in the skeleton, we assign a motor that is capable of creating a rotation around the axis of the DOF. By assigning a torque to this motor, we create a rotational velocity that moves the joint in the intended direction. These rotations will allow us to bend and stretch the various parts of the body as we wish, by speeding up or slowing down as necessary. In order to fully animate a character this way we need a controller.

The controller sends the signals to the various motors in the body to cause them to rotate one way or the other. We can then program such a controller to counter the effects of gravity and forces on the body to stay standing upright, to move the legs such that the character walks forwards without falling or grasps an object by closing the fingers until collision. To find such a controller capable of executing one or more tasks is difficult however. Where saving joint angles of a realistic motion is relatively easy, creating a system that can actuate this on the fly requires a constant balancing of the velocities of the motors to ensure the character does not fall. As this is still a topic of active research on how to best control the motion of a character we will not use it in our research. It does allow us to include grasps more easily than we would now, as a controller can be adapted for multiple grasp options, where the key framed animations need to be captured anew.

C.3.5 Motion capture for hand motions

In order to have a system capable of realistic grasp and move motions, we require both a correctly chosen grasp as well as a realistic execution of this grasp. With the former discussed above we now turn our attention to the animations that will be used to execute the grasp. In order to not over complicate the system we will keep the focus to hand motions for now. The full range of character motion will play a role in actually reaching the object in most applications, but we are focusing on improving the grasps themselves. By not using a single grasp pose where we stick the object to in a manually and often erroneous way, we can already improve the quality of the animations by a lot.

In order to create a set of animations capable of executing grasps we will need a way to generate them. Manually crafting them would be too time consuming and would not guarantee that they work in the real world. As such using some form of motion capture would be better, as the grasps can be recorded when the actor actually holds an object at the same time. This allows for a more natural interaction by the actor and should create more realistic animations.

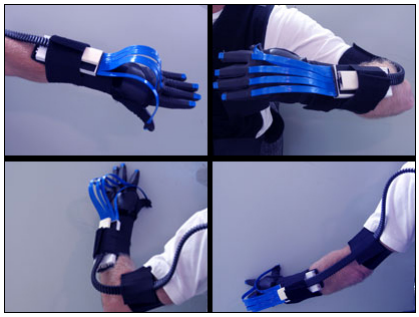

Figure 33: *Example of motion capture done with a dataglove. Image from* `https://www.vrealities.com/products/data-gloves/shapehand`

Care will have to be taken to ensure the grasps are not too dependent on the object that is grasped. By using several objects of different shapes, we can create a greater variety in motions available. These could then be combined or 'blended' together to adjust for different objects as necessary. Keeping in mind that, as shown in [Feix et al. 2009], humans use a specific set of grasps for almost any task, we can safely assume that, by picking the appropriate grasps from this taxonomy, we can animate grasps for a wide variety of objects.

In order to record these animations we will require a technique capable of capturing the full dexterity of the human hand. As there are many joints that can move independently from each other as well as a high amount of self-occlusion, it can be difficult to estimate the pose of a hand correctly. This is why recently there has been a push to creating capture techniques specifically aimed at capturing the motions and interactions of one or two human hands.

Seeing as most motion capture systems that rely on markers are set up to capture a full human body, only a small part of the resolution of the cameras is available for the hands. Having many markers on the hand will then result in being unable to properly assigning a label to each. This makes is hard if not impossible to capture a hand motion. In [Hoyet et al. 2012] Hoyet et al. have looked into how the perception of animations changes as less markers are used to capture the hand motion. In their experiments they recorded a set of motions used often in real life such as during talking, counting or typing. They recorded this with a set of markers on each of the joints. To do so they had to work with a special setup of the camera's that resulted in a more limited space of 1,5m by 1,5m that the actor could move in.

Figure 34: *Some examples of the marker sets used in [Hoyet et al. 2012].*

From this motion capture data they created a standard using twenty markers per hand and forward kinematics to create an animation as close to real as they could. After this they used inverse kinematics on several reduced marker sets to create a similar animation to their standard, to compare them to the standard in terms of how real they looked. They created an experiment setup with movie clips of an animated character performing the animations they had created. By showing the standard clip first followed by any of the reduced sets, they asked a group of participants to choose whether they noticed a difference or not. This resulted in some reduced marker set being hardly distinguishable from the animations that used the full marker set. This could reduce the markers from 20 per hand down to only 8, significantly reducing labeling and occlusion problems.

In [Erol et al. 2007] an overview is provided of various techniques to capture hand poses that rely on computer vision. By removing the need of markers being placed on the hand we hope to obtain more natural motion as the actor will not feel constrained by the equipment. These techniques do face several challenges. As the hand is very dexterous and capable of many poses, self-occlusion becomes an even larger issue as the skin is very similarly colored. Extracting and distinguishing the various parts of the hand is therefore very difficult.

The reliance on video data also means a lot of processing power is needed to analyze the data and the lighting conditions can change the outcome very easily. Some of these are mainly of interest when applying these techniques on mobile devices or arbitrary locations. By recording in a studio as with traditional marker based motion capture some of these issues can be alleviated, although it still remains difficult to segment the hand properly. In [Wang et al. 2013] and [Ballan et al. 2012] two examples are given of how to improve the quality of the recorded motion. In [Wang et al. 2013] they aim to reduce the difference between observed data and the simulated motion by comparing the video feed with a similar graphical representation of the simulated hand. In [Ballan et al. 2012] they create a system capable of locating salient point of the hand more accurately offline to analyze the features in the online video capture data. By combining this with edge detection, optical flow and collisions they improve the quality of the resulting motion.

C.3.6 Animating hand motions

With the data that describes the motion of a virtual body we are set to actually create an animation. With key framed joint angles of a skeleton this is as simple as updating the transformation matrices of each joint at every time step. By applying the transformations whilst following the hierarchy of the joints we can specify a pose for the body both globally on the root joint and locally for each of the other joints.

As many virtual applications allow for multiple motions there has been a desire to switch between these in a smooth manner, rather than suddenly changing to the start of an animation that begins with a completely different pose. This is often accomplished with a motion graph, where transitions are included as edges between nodes that represent the animations.

This can be done by finding common poses where one animation could flow naturally into another. A different approach is to find similar poses and create a small transition animation to connect the two. We could also apply blending to smoothly transition from one animation to the other by interpolating the two different motions provided they are similar enough. This will be more than our system needs to be capable of as we will choose an animation prior to execution. As such we will not rely on changing between two animations.

As mentioned previously we would not make use of dynamics in our framework. There has however already been some research into the use of dynamics with complex hand motion in [Pollard and Zordan 2005]. While their focus is also on animating grasping motions, they show that including some passive components allows the hand to react realistically to objects pushing against it. The active components aim to move the joints to a desired orientation using a PD controller and a Finite State Machine to go from a relaxed pose to a grasp and releasing back to the relaxed pose again.

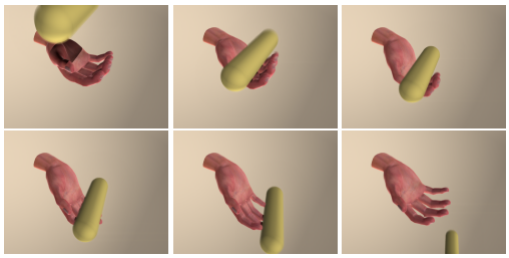

Figure 35: *The hand reacts 'passively' to the objects being dropped on it. Image from [Pollard and Zordan 2005]*

The FSM specifies a gradual transition to the next state by interpolating the desired orientations. Finally by adding in some joint torques that would result from the arm and gravity they obtain their final sum of torques to steer the hand as desired. By using some motion capture data they obtained the parameters for their controller and use this to create new motions and interactions. The system shows promise as they could realistically animate a handshake, which can prove difficult with motion capture due to the close contact and occlusion that this brings.

C.4 Conclusion and discussion

Having reviewed the literature of related subjects we now have a better overview of the existing state of the art techniques that are available to use in our framework. While it is clear that a lot of research exists to make robots able to grasp and manipulate like we do, few of these techniques seem to make it over to a virtual application outside of simulations to test them.

Similarly within computer animation a lot of work has been done to make the motions appear as realistic as possible by moving ever closer to the inner workings of the human body. Indeed by modeling the effects that move us as best as we can we should obtain a more natural motion as it complies with physics. Most of this research has however been focused towards full body motion and in many modern day applications grasps are often nowhere near as polished.

As such we will aim to provide a framework that will allow us to use much better grasping animations by matching them to the task at hand. As there are many tasks possible we will focus on pushing objects across a surface. If we are successful in creating a realistic looking setup for just a hand and a simple task, we could expand by including the full body and more complicated tasks as needed. This should help increase the fidelity of many virtual applications.