CHALMERS
UNIVERSITY OF TECHNOLOGY

Simple Kinect-based gesture tracker

Master's Thesis in Complex Adaptive Systems

ANDERS RYNDEL

Department of Applied Mechanics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

Master's thesis 2017:09

Simple Kinect-based gesture tracker

ANDERS RYNDEL

Department of Applied Mechanics
Chalmers University of Technology
Gothenburg, Sweden 2017

Simple Kinect-based gesture tracker
ANDERS RYNDEL

Supervisor:Mattias Wahde , Department of Applied Mechanics
Examiner: Mattias Wahde , Department of Applied Mechanics

Master's Thesis 2017:09
ISSN 1652-8557
Department of Applied Mechanics
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 (0)31 772 1000

Typeset in $\LaTeX$
Gothenburg, Sweden 2017

Simple Kinect-based gesture tracker
ANDERS RYNDEL
Department of Applied Mechanics
Chalmers University of Technology

Abstract

This project has developed an alternative to standard touch-based control system by creating a gesture based system built on the Microsoft Kinect for Xbox 360. This system is required to be reliable, robust, and intuitive to use. The developed system uses a method that models a region around the user and classifies anything that extends from that region as a raised hand. It is a crude solution but proved very reliable when used by an experienced user.
The intended purpose of this system is as a control device for a partner agent to be use with seniors. But the system turned out to have weaknesses that makes it less suitable than other possible solutions. Weaknesses includes that the system can be put into an error mode that needs knowledge of the system to recover from, something the intended semographic can't be expected to have. This weakness could be improved upon with a better system, but it still leaves the more important weakness: that the ergonomical requirements of the primary target demographic, which can't be expected to find constantly raising arms to the level of the head to be a comfortable work environment. It is likely that a parter agent user is better served by a hands-on controller, such as a controller for Playstation or XBox.

Keywords: gesture reconition, XBox, Kinect.

Acknowledgements

I would like to thank my supervisor Mattias Wahde for invaluable guidance during the course of this project. I would also like to give a special thanks to my parents who have supported me through the whole process.

Anders Ryndel, Göteborg 18/5/15

Contents

List of Figures

1 Introduction

1.1 Motivation

The future of elderly care is a problem that society faces in the coming decades as a result of an ageing population. It is therefore of interest to make use of robotics to alleviate pressure on the human resources in the elderly care services facing ever growing work loads. While the current methods of elderly care involving human social contact will not become obsolete and cannot be replaced by robotics, it is still possible and meaningful as compliment. It is for this purpose the partner agent is created. A partner agent is a device that functions as a robotic personal assistant, it can help keep calendar and aid the user to keep socially active by providing the means of communication.
One of the challenges involved in creating a partner agent that is practical and reliable is the problem of control input. Only in very recent years has voice recognition become a practical method of control but it is still very sensitive to circumstances. It is not only required that the user speaks the right language, but it might also not function properly for some dialects or accents. Taking into account the elderly target user base cannot be assumed to have good enough diction makes voice recognition a bad choice for a partner agent control method.
The alternative that is explored in this project, is to create a system that uses the Microsoft Kinect for recognise gesture based control input. The Microsoft Kinect is designed to be a control device for Microsoft's line of consoles, the XBox. The Kinect's designed purpose is therefore very closely related to the function needed for the parter agent, it has all the performance necessary to track a user. The Kinect's strength is that it captures a lot of information in colour and depth images at a reasonable pace. This allow software that uses the Kinect to use it as a high detail controller that can interpret gestures and poses, even facial expressions of users. This detail comes at a price of not being very reliable. In fact, one of the greatest complaint about the latest XBox release, the XBox One, was that the bundled Kinect wouldn't work reliably.

1.2 Objectives

The aim of this project is create a system that uses the Kinect in the simplest form possible fulfils the needs of a control device for a partner agent. This is done to limit the reliability problems to the greatest possible extent and to make the partner agent user friendly for the intended demographics. In order to full the needs of the partner agent, the system needs to be able to accurately register when a user is present in view of the Kinect, when this user raises and lowers his hands and know the position of the hands when raised.

1.3 Limitations

It is conceivable that a it would be useful for a parter agent to make use of the actual shape of the hand. This would allow the user to interact with the partner agent by means of sign language, either standard sign language or in simplified custom form. This is outside the scope of this project due to the scale of this problem.

2
Background

2.1 Kinect

The Kinect is a device created by Rare, a subsidiary of Microsoft, as a controller-less control method for their game console, the XBox. The purpose of the device was to enable games based on motions and gestures on the console. The technology that has made this device possible is the low-cost, relatively robust, and high resolution depth sensor that is integrated into it. This technology was originally developed by PrimeSense.
The depth sensor technology operates based on an IR camera and projector. The projector projects a finely grained grid of dots, created by a diffraction grating. The IR-camera is used to capture the dots on the scene. The distance to each point can be calculated by measuring the distance between the points in a neighbourhood because the angle between the dots are known. The Kinect used in the project is shown in image 2.1 and comes from the *XBox 360*. The camera has an angular field of view of 57° horizontally and 43° vertically and the depth data retrieved from the device has resolution 640×480 pixels at 30 frames per second with a depth resolution of 1 centimetre. The accuracy of the depth sensor is about 1% and works best for smooth matte surfaces.
It is possible for the depth sensor to fail to register objects and particularly edges of objects that are smaller than 1 cm across, or has jagged or glossy surfaces. This is especially true if the surface has a high angle to the camera. In these cases, the sensor simply returns not-a-number. Because the IR-projector and camera are not located in the exact same place, there are usually regions visible to the camera that are not illuminated by the projector and because of this. This results in that the depth information in the areas shaded from the projector is unavailable.

2.2 Application

The properties of the Kinect makes it highly interesting for applications outside console gaming. These applications of the Kinect commonly involves *human-computer interaction* (HCI) as a means for a computer to detect the actions of a human user and can be broadly put into two categories.
The first category is where the user takes a passive part in the interaction. In other words, the user is not directly interacting with the Kinect sensor but is simply monitored by it. Important sub-categories of this are surveillance for security purposes [1, 2, 3] and uses for medical observation[4], diagnosis [5] and rehabilitation [6].

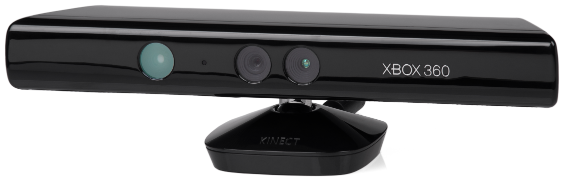

Figure 2.1: The image shows the stand alone Kinect for XBox 360. The apertures on the front from left to right are the IR grid projector, the IR camera, and the RGB camera.

The second category is where the user is an active participant in the interaction. Here, the user is most often using the Kinect for control input for some other device. One significant field is the operation of robots, especially in the form of direct tele-operation of robotic limbs [7, 8, 9, 10, 11].

The other, similar but separate, field is the use of the Kinect as a control method to replace or complement keyboard, mouse, touchscreen and similar. In common for all applications that uses the Kinect for control input is the emphasis on seamless user friendliness. This user friendliness is critical when the intended user is not accustomed to traditional input devices, for example when the users are seniors[12] or children [13]. Furthermore, the seamlessness is important when using information technology to augment an activity that is traditionally a strictly human to human interaction, for example the use of the Kinect to create an immersive experience in a virtual classroom [14]. The uses of the Kinect as a interaction tool for media devices ranges from simple tiled grid navigation[15], to gesture based text input [16] on to fully a functional mouse cursor[17].

Of special interest for this project are methods to identify and track hand palms and finger tips. Identification of the hand is made using either depth imagery combined with colour imagery [18] or depth imagery alone by identifying the shape of the hand[19, 20, 21, 22]. All of the methods provided in the mentioned articles that do the identification based on shapes requires the user to present an open hand with the fingers spread for the identification to work. Two of these methods makes extensive use of functions found in the computer vision library *OpenCV*[20, 18], used for example to find contours and centre of pixel mass of hand surfaces.

2.3 Gesture recognition

Computer vision in general consists of at least three steps, segmentation, feature extraction, and interpretation [23]. This includes the specific field of using a depth image to identify hand gestures [24] in addition to the separate fields of using depth imagery for some other purpose and performing gesture recognition using some other source information.

2.3.1 Segmentation

Segmentation is the part of the gesture recognition process the determines what part of the image data is relevant. This can be made relatively simple with the use of depth imagery to filter out the parts of the image that do not belong to the hand [24]. This information is used to create an outline of the hand. The simplest method of obtaining this outline requires the user to ensure that the only thing in front of the camera in a given depth interval is the hand (as done by [25, 26, 27]).

A further refinement of the outline is to introduce a virtual skeleton overlay onto the outline with joints placed at places matching that of the hand and indeed the entire body. This method reduces the demands on the user since he is not required to ensure that only the hand is in view because the entire body can be tracked. The joints corresponding to the hand are then simply selected as interesting at a later stage of analysis. Skeletal tracking is the default method most commonly used by authors [28, 29, 30, 31].

2.3.2 Feature extraction

Feature extraction is the process of finding and tracking points of interest and to place a representation of the hand in a practical variable space (for example the set describing all angles of the joints in the hand). Methods for feature extraction can be categorized based on whether it captures static or moving gestures. The former is the simpler case chosen for practical purposes where the application does not require more complexity [29, 26]. The latter allows for a much wider range of gestures to be captured and the range of practical applications becomes much wider [32, 33].

Of special interest is to make the variable space invariant such that gestures that appear similar to a human has similar representations in the variable space. In the case of methods used for static gesture detection, this extends only to spatial invariance [34] while methods for moving gestures also include temporal invariance by not only describe the hand state but also trajectories [35, 33, 36, 37, 27].

2.3.3 Gesture interpretation

This third step consists of translating the gesture described by the previously obtained features into useful signals. This is fundamentally a classification problem and there is a plethora of methods to solve such problems. Examples of such methods used are *naive Bayes nearest neighbor* (NBNN)[28, 38, 39], *artificial neural networks* (ANN) [26], and *bag of visual words SVM* [25, 33, 40, 41].

Of these methods, NBNN is perhaps the simplest [42]. The algorithm starts with a set of identifiable classes. Every class C of these classes has a set of corresponding features $\{d_j^C\}$. A proximity function NN^C is defined that maps a feature d onto the closest feature in the class set

$$NN^C(d) = \arg\min_{d_j^C} ||d - d_j^C||. \tag{2.1}$$

The classification is then simply made by finding the smallest mean square error between obtained features and the nearby features in the possible classes

$$\tilde{C} = \arg\min_{C} \sum_{j} ||d_j - NN^C(d_j)||^2 \tag{2.2}$$

An artificial neural network is a classifier that simplistically mimics the function of the biological brain [43]. It is a network made up by nodes called McCulloch-Pitts neurons in a feedforward system that is divided into input, hidden, and output layers. The neurons usually take normalized values between 0 and 1 or -1 and 1. The input value of each neuron in the input layer is taken from the extracted features and the input values of neurons in following layers are calculated solely by the immediately preceding layer by weighting and normalization. Given a layer k with neurons ξ_i^k the the output layer of the k layer and values becomes

$$\xi_i^{k+1} = g(\sum_j \omega_{ij}^k \xi_j^k). \tag{2.3}$$

The normalization function can for example be taken as $g(x) = \tanh(x)$ when the interval -1 to 1 is used. The classification is then made by having each of the neurons in the output layer correspond to one class and selecting the class with the neuron of the highest value. The key to artificial neural networks is that they are created through training. This training is performed by changing the values of the neurons so that the results from using a set of inputs conforms to the intended output. This has the advantage that the person designing the classifier does not need to know what features are important or not, the disadvantage is that a large training set is required where as many as possible of each variation of each class is included.
A linear support vector machine (SVM) classifier is a binary classifier that consists of a hyperplane in a vector space. The classifier is constructed by finding the hyper plane that best separates two classes of points in a vector space. The classification of a of is made simply by determining what side of this plane it is located. The weakness of this method is that it isn't necessarily possible to find a simple hyper plane.
Bag-of-Visual-Words uses a variation of a Bag-of-Words that matches the extracted features to occurrences in a pre-created dictionary of shapes [44]. A "word" in this sense can be a small image fragment and the vector representation of an analysed image counts the number of times of each fragment appears in the image, but the position of the fragments are discarded. Because the resulting vector is sparse, due to the high dimension of available words, a linear SVM can be used to make the classification without hindrance and allows for each class to be attached to a hyperplane in this vector space [45].

3
Method and implementation

3.1 Design goals

The purpose of this project is to create a system for identifying and interpreting hand gestures using the depth data available from the Microsoft Kinect. The system must be able to indicate if either right or left hand is raised or lowered and it should also be able to track either hand when raised, including both at the same time. The system is indented to be used as an input device for a computer and it is therefore important that it should have sufficient update rate and lack of latency to be of responsive and practical in a real-time environment. Furthermore, because the intended primary user demographic consists of seniors, and therefore cannot be assumed to be accustomed with the use of computers, special care must be taken to make all interactions as intuitive as possible.

The possible lack of user experience makes robustness especially important for a solution to be useful. Because of this, considerations must be made to minimize the environmental constrains. Setting up the system for use should not require intricate knowledge of the function of the system but should at most require a simple set of instructions to be carried out and a few conditions to be met with regards to lighting and objects in the room. Robustness not only demands that a system should not be prone to failure but also that it should handle failure gracefully. From this follows that there should be a mechanism for detecting and correcting any errors that may arise and that this mechanism is preferably automatic.

3.1.1 Specification

The system is considered to work nominally if the system is able to register when a user who sits in front of the Kinect raises and lowers hands without giving give false readings by registering other objects as hands. It is necessary that the system has a sufficient update rate in order to feel fluid. This update rate should never go below 20 frames per second, meaning the time between updates should never exceed 50 ms. The latency, the time between the user performing an action and the action is registered should not exceed 200 ms for the system to feel responsive. In order for the system to be considered intuitive to use, a user completely unfamiliar with the system must be able to successfully make use of its nominal function with only a short written or spoken instruction as guide.

3.2 Solution selection

There are a number of methods available to track a user with a Kinect discussed in Sections 2.2 and 2.3. Of these, the most common type of method involves creating a virtual skeleton for segmentation. Such a method was not used to build this system because the reviewed methods required prior knowledge of whether the user is standing or sitting in order to work reliably. It would have been possible to develop these methods such that this would not have been a problem, but since the standing or sitting posture of the user does not actually have an impact for the application, a simpler method was chosen instead.

The application only requires the system to be able to detect raising, lowering and moving of hands. It is therefore feasible to use a method built on refined depth filtering for segmentation. The detection and tracking of the hands are based whether they appear in a certain region of space in front of the user and not the actual pose of the user, something that might have been the case had a skeleton-based segmentation been used. The selected method segments out the desired hand points and skips the rest of the body. This method models the user to be inside a block and classifies anything that extends from that block as a raised hand.

3.3 Solution overview

The system that implements this method is constructed by carefully selecting of the planes defining the surfaces of the user block and the 3D regions in which detection is allowed. The user block is constructed by detecting and tracking a point that corresponds with the user's head. The detection builds on the concept of detecting a maximum point that corresponds to the point that is positioned the largest distance above a plane in that 3D space. The plane used to detect the head is mostly horizontal but with the normal pointing upwards and slightly forward. This results in that the point taken as the head point is located upwards and forwards in the depth image, where the user's head is expected to be.

This method, in its most basic form, makes no considerations of the shape of the objects it is intended to track, something that makes the system vulnerable to error. The system cannot determine if the object it represents is a hand or some other object from a single point alone. It is therefore necessary to use a separate process to verify that the tracked points do in fact represent the intended objects. The head verification determines if an object assumed to be a head is shaped like a head and the hand verification determines if it is possible to trace a path between the hand and head point that does not need to cross over background.

The strength of this arrangement is that the point tracking method is very fast and it is reliable in giving correct readings if recent readings has been correct. It therefore benefits performance to let the point tracker run and let the verification process tag along behind it.

3.4 Components

3.4.1 Coordinate system

Vectors will be used in the following sections to denote the geometry of the solution. These vectors work in units taken directly from the depth data that comes from the Kinect itself. The precision of depth values returned from the Kinect is measured in millimetres, even though the accuracy varies with the object's distance to the camera. The natural unit used for distances along the x- and y-axes are pixels. The depth image of the Kinect has the resolution of 640×480 pixels at a horizontal field of view of 58° and a vertical field of view of 45°. The Kinect is best used in the range $0.8 - 1.5$ m which means that the distance per pixel is about 1.5 mm at 1 m distance from the camera. The internal coordinate system is defined relative to a user facing towards the Kinect such that positive x-axis is in the direction of the user's right, the y-axis in the direction of the users up, and the z-axis in the direction of the user's forward.

3.4.2 Planar separator

The planar separator a component that separates space into two parts. Several of these planes can be combined together to form regions with complex boundaries. It is the basic tool to construct the detection regions where points are allowed to be detected. It is defined internally by a unit normal vector $\vec{\nu}$ and a scalar λ. A vector $\vec{x}$ is determined to be above the plane, relative to the origin, by checking the criterion:

$$\vec{x} \cdot \vec{\nu} > \lambda \tag{3.1}$$

It is appropriate to define the planes in terms of a normal, a reference point, and an offset according to figure 3.1 because these planes are used to to define the detection regions. In order to obtain the internal representation, the normal is taken as-is, only normalized, and the scalar is calculated from the offset k and reference point $\vec{x}_0$ by the following:

$$\lambda = \vec{\nu} \cdot \vec{x}_0 + k \tag{3.2}$$

3.4.3 Local maximum finder

The local maximum finder determines the highest point relative to a plane in a box region of the depth image. Given the normal vector $\vec{\nu}$ of the plane and the points $\{\vec{x}\}$ in a region, the point returned is

$$\vec{x}_{\max} = \underset{\vec{x}}{\operatorname{argmax}}(\vec{x} \cdot \vec{\nu}). \tag{3.3}$$

The box region is defined by a center point, a width, a height, and a depth. The algorithm is an exhaustive search of all points in the rectangle defined by the width, height, and center point. A point is excluded if it is not inside the box as defined by the depth parameter. This arrangement gives two advantages. First, it gives increased performance for a smaller size of the searched region. Second, it makes

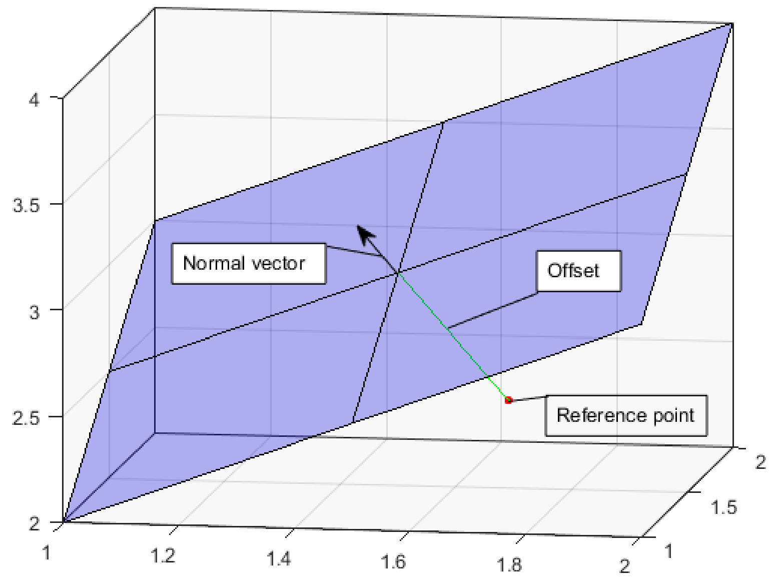

Figure 3.1: The figure shows the arrangement of the reference point, the offset, and the normal vector.

the algorithm find a local maximum around the center point. This assures that the point sticks to its current object instead of skipping around.

3.5 Services

3.5.1 Head point tracker

Central to the method is the ability to detect a point that corresponds to the position of the user's head. This point is used as reference for the definition of the left and right hand regions and is detected using the local maximum finder in two modes. Both modes use the same plane to measure relative height, the plane with a normal vector pointing forward and up relative to the user's orientation and it is in the internal coordinate system defined as $\vec{\nu} = \{0, 2, 1\}$. This normal vector was found to be suitable for this particular purpose because it places the head point at the top of objects in the foreground. The arrangement of the plane relative to the user can be seen in Fig. 3.2. The first mode is used for the case where no previous point is known. This mode limits the search in a box centred in the centre of the depth image and with width and height of half the width and height of the depth image and with infinite depth. Once a point is detected, the system will turn to the second mode intended to track the point. This mode searches in a box centred around the previous point with width and height of 20 pixels and depth of 100mm.
The system will use the last known valid point if there is no valid point in the box for the current update. The system will use the last known valid point until a valid point is found or until one second has passed since the last point was found. The point is considered lost if this time expires and the system will then revert to the

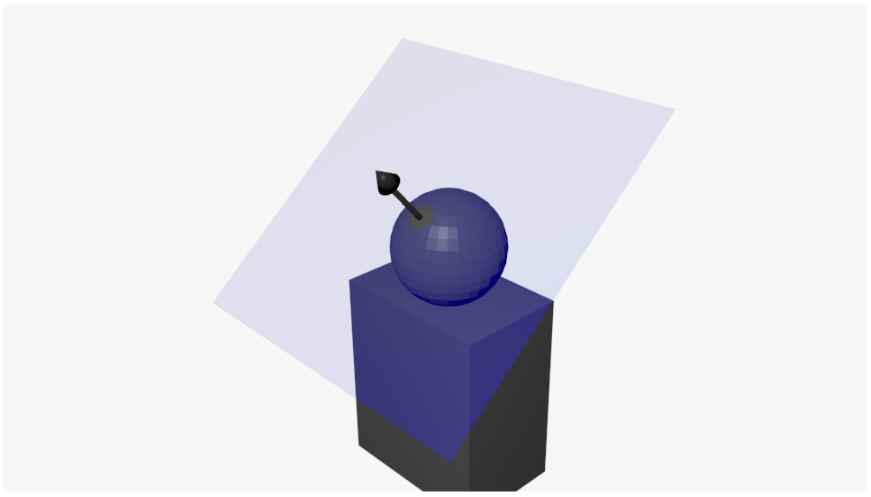

Figure 3.2: The plane used to detect the head and how it intersects with the user.

first mode, attempting to detect the head again.

3.5.2 Head verification

The head verification determines if the head point corresponds to the user's head. This is the system's only method of knowing if the user is in view of the Kinect at all. The head point tracker will reliably detect the users head assuming the user sits or stands upright in front of the Kinect with both arms lowered. The system without the verification will take up a spurious reading for the head point if the user is not in this position and this will result in error in the system. This is a problem if the user for whatever reason decides to leaves and comes back later.

In order to amend this, the verification checks that the part of the image the head point is attached to is indeed shaped like a head. Specifically, the verification algorithm checks how well the profile of the top of the object the head point is attached matches an irregular semi-ellipse.

The algorithm uses a depth threshold of 5cm. If the depth step from one pixel to the next exceeds this threshold, then that pixel is counted as a cut. Pixels like these are only expected to occur at the edge of the users head, thus forming the heads profile. The algorithm starts from the supposed head point and traces upwards until a cut is reached. This point defines the head top point p_t. It then, again from the head point, traces downwards 50 pixels. This point defines the head centre point p_c. Verification fails if a cut occurs before this point can be reached because the investigated object can not be a head. The distance of 50 pixels is taken because it is an approximate distance between the common position of the head point relative to the head and the centre of the head.

Left and right head side points (p_l and p_r) are reached by tracing left and right respectively from the head centre point until cuts are reached. At this stage, the distances p_l, p_r and p_c, p_t are compared to ensure that the supposed head has the right size and proportion before proceeding to the next step. Using the obtained points, two masks, $M_{ij}^{(r)}$ and $M_{ij}^{(l)}$, describing the left and right side of the irregular semi-ellipse are defined as follows:

$$M_{ij}^{(l)} = \begin{cases} 1 & \text{if } \left(\frac{i}{||p_c - p_l||}\right)^2 + \left(\frac{j}{||p_c - p_t||}\right)^2 < 1 \\ -1 & \text{else} \end{cases} \tag{3.4}$$

and

$$M_{ij}^{(r)} = \begin{cases} 1 & \text{if } \left(\frac{i}{||p_c - p_r||}\right)^2 + \left(\frac{j}{||p_c - p_t||}\right)^2 < 1 \\ -1 & \text{else} \end{cases} \tag{3.5}$$

The left mask is defined for when i is less than the x component of p_c and the right mask is defined when i is greater. Together, they form M_{ij} which is defined in the entire box where p_l and p_r forms the lower corners, and p_t forms the upper edge. In the same box, binary depth data D_{ij} is formed by the granular depth data d_{ij} from the the depth image, the depth of the head point d_h, and another threshold $T = 10\text{cm}$

$$D_{ij} = \begin{cases} 1 & \text{if } |d_{ij} - d_h| < T \\ -1 & \text{else} \end{cases} \tag{3.6}$$

In other words, the binary depth data registers positive for points that exists in the same segment as the head point. A normalized verification factor H is formed from these by summing all values of ij inside the box formed by p_l, p_r, and p_t

$$H = \frac{1}{||p_r - p_l|| \cdot ||p_t - p_c||} \sum_{ij} M_{ij} D_{ij} \tag{3.7}$$

This factor will take value between 0 and 1, where 1 indicates that the object the head point is attached to matches the shape of a head very well.

3.5.3 Hand point tracker

There are two separate hand point trackers in the system, one for each hand. The two hand point trackers are mirrored in the forward-vertical plane that includes the reference point provided in by the head point tracker. In other words, both trackers are defined identically except for the x-axis that is inverted. The hand point trackers works similarly to the head tracker in that they operate in two modes, one for full image search when no previous point is known and one for tracking a point by searching a limited area around the last detected point.

The hand detection process uses the same tilted plane as the head tracker to determine the maximum point, the plane with normal $\vec{\nu} = \{0, 2, 1\}$. The same plane is used for the hand detection as for the head tracker because raised hands also appears as objects in the foreground, of which their top point is of interest. The significant difference to the head tracker is the region for valid points is limited. This region is defined by a set of planar separators.

The first separator is intended to exclude points attached to the user's body when the user is a default position with no hands raised. The normal vector of this plane that defines this separator is pointed upward, forward, and to the side of the user, i.e. normal vector for the plane of the tracker for the right hand points to the right. In the internal coordinates this direction is $\vec{\nu} = \{1, 1, 2\}$ for the right tracker and

$\vec{\nu} = \{-1, 1, 2\}$ for the left. The separator plane is off-setted from the head reference point, realistically placed somewhere on the user's forehead, by about 10cm in the direction of the normal vector to properly clear the body. This separator is what essentially defines a raised hand in the system, it admits any object that is sufficiently high up, far to the side or far forward to be considered a raised hand.
Additional planar separators are used to limit the occurrence of spurious objects being registered. This includes a background separator that excludes everything that is behind the user. This separator uses the plane that includes the head reference point and has the normal vector $\vec{\nu} = \{0, 0, 1\}$. Another separator specifically intended to filter out the ceiling is also used. The normal vector of this plane points down and forward, in direction of $\vec{\nu} = \{0, -2, 1\}$, off-setted from the head point by -30cm, placing the plane above the user.
The last separator is the forward-vertical plane with normal vector $\vec{\nu} = \{-1, 0, 0\}$ for the left tracker and $\vec{\nu} = \{-1, 0, 0\}$ for the right intended to separate the left and right hands.

3.5.4 Hand verification

The hand verification is a process where the system checks that a point registered as a hand actually represents the users hand. Points that does represents a hand is deemed to legitimate and the verification fails if the point is not legitimate. When the verification fails and a point is determined to not be connected to the body, an event is raised and the detection system is reset.
The verification is a process that runs outside the usual flow of image updates and lags slightly behind it, skipping some frames when there is not enough time for computation. It is assumed for the sake of performance that a detected point is a legitimate reading until the verification fails after the fact, as long as there exists a legitimate reading before it. A reading will be ignored if no legitimate reading has occurred immediately before it.
The system is the most sensitive to accidentally registering objects that appears above and in front of the user and the primary purpose of the hand verification is to determine if the point registered as a and is connected with the rest of the user The method for accomplishing this is based on calculating a path from the hand point to the head point that does not pass any of the steep edges that contrasts the user towards the background. The algorithm used to create this path is A* and the heuristic used is the distance in three dimension between the current point and the destination but with a penalty added if the current point is too far behind the user.

3.5.5 Hand movement classifier

The hand movement classifier registers the movement of the user's raised hand. The ability to track hand movements rests on the ability to distinguish deliberate movements from non-deliberate movements so that the system only receives commands intended by the user. A hand movement is considered deliberate if the hand moves sufficiently far over long enough time.
The minimum time limitation is necessary in order to avoid avoid registering noise

in the hand point reading. It is for example possible for the hand point to jump between the fingers of a raised hand even though the hand itself is not moving. On the other hand, the minimum time limitation is also problematic because it hinders the system's responsiveness. In order to balance the resilience against noise and the system's responsiveness, a softer limit is used that accepts a slower movement over a longer stretch of time or a very fast movement over a period shorter than a characteristic time.

This is done by using the most recent hand point $\vec{p_r}$, obtained directly from the hand tracker, and a historical centre point $\vec{p_h}$ that represents the position where the hand has recently been. The historical centre point is updated at each iteration by

$$\vec{p_h}^{\text{new}} = a\vec{p_h}^{\text{old}} + (1-a)\vec{p_r} \tag{3.8}$$

$$\tag{3.9}$$

A movement is registered when the distance between $\vec{p_r}$ and $\vec{p_h}$ becomes larger than a distance D. The minimum movement speed and the characteristic time under which only very fast movements will be registered are selected by adjusting the parameters a and D appropriately.

Assume a hand is starting from resting and is travelling along the x-axis at a speed v and the hand is registered by the system at time intervals δ. The positions of the hand points are then

$$p_h = \delta n v. \tag{3.10}$$

Inserted in expression 3.8, this becomes:

$$x^{(n)} = ax^{(n-1)} + (1-a)\delta n v \tag{3.11}$$

The distance $d^{(n)}$ at frame n between the hand point hand the historical centre point is

$$d^{(n)} = x^{(n)} - \delta n v \tag{3.12}$$

The equation for this distance is

$$\begin{aligned} d^{(n)} &= x^{(n)} - \delta n v && (3.13)\\ &= a(x^{(n-1)} - \delta n v) && (3.14)\\ &= a(d^{(n-1)} + \delta v). && (3.15)\\ & && (3.16) \end{aligned}$$

Assuming small δ, this equation is made continuous:

$$\begin{aligned} d^{(n)} &= a(d^{(n-1)} - \delta s) && (3.17)\\ (1-a)d^{(n)} + a(d^{(n)} - d^{(n-1)}) &= a\delta v && (3.18)\\ \frac{1-a}{\delta a}d^{(n)} + \frac{d^{(n)} - d^{(n-1)}}{\delta} &= v && (3.19)\\ \{t = \delta n\} &\rightarrow && (3.20)\\ \frac{1-a}{\delta a}d(t) + d'(t) &= v && (3.21) \end{aligned}$$

The solution for this equation given the original distance 0 is

$$d(t) = \frac{\delta as}{1-a}\left(1 - \exp\left(\frac{a-1}{\delta a}t\right)\right) \tag{3.22}$$

and the criteria for a registered movement becomes

$$D = \frac{\delta as}{1-a}\left(1 - \exp\left(\frac{a-1}{\delta a}t\right)\right). \tag{3.23}$$

This equation is written to be dependant on minimum speed V_{min} and characteristic time t_c

$$V_{\text{min}} = \frac{1-a}{\delta a}D \tag{3.24}$$

$$t_c = \frac{\delta a}{1-a} \tag{3.25}$$

transforming equation 3.23 into

$$V_{\text{min}} = s\left(1 - \exp\left(-\frac{t}{t_c}\right)\right). \tag{3.26}$$

The parameters a and D are functions of V_{min} and t_c are therefore

$$D = V_{\text{min}} t_c \tag{3.27}$$

$$a = \frac{t_c}{\delta + t_c} \tag{3.28}$$

The parameters are selected such that $V_{\text{min}} = 100$ pixels$/s$ and $t_c = 0.5s$.
When a hand movement is registered, the system determines the direction of the movement. The system is able to register movements in the up, down, left, right, in, and out directions. The direction is determined by taking the dominant axis of movement. This gives the user 6 available movement commands at any moment. This number is not limited by the system but rather the user. It is surprisingly difficult for a user to move a hand raised in the air in a specific direction relative to the Kinect because the user tends to do motions in depth even when these are not indented. As a result, the 6 axis directions are about the accuracy the user can manage.

3.6 System architecture

The system is separated into three parts shown in Fig. 3.3The parts consists of the Kinect manager, the Service manager, and the event engine. The Kinect manager is the only object that operates directly with the Kinect. It sets up and shuts down connection on start-up and shutdown. It also runs a thread to capture and temporarily store the depth images from the Kinect. The Service manager is the structure to which all the services that creates the functionality of the system are attached. Lastly, the event engine provides the interface to external software. External software uses the event engine by registering a set of functions with the Service

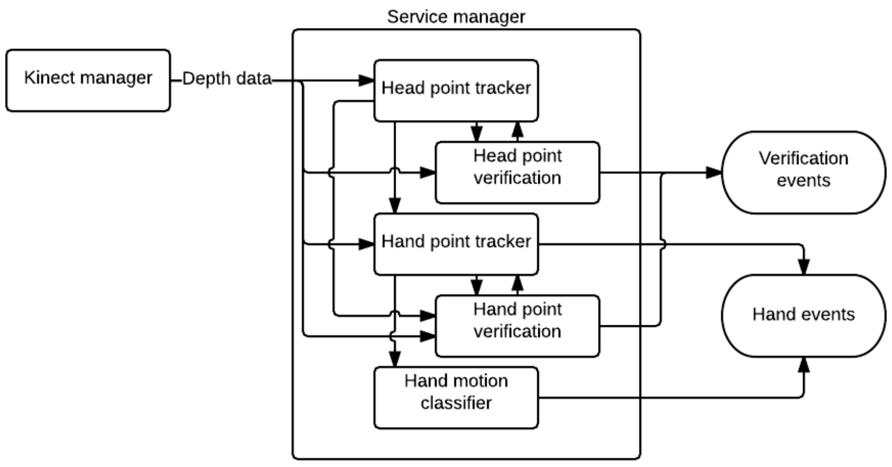

Figure 3.3: Overview of the data flow in the system

manager to be called when any of the verification states change and another set for when the either hand is moved, raised, or lowered.
The system's software is build in a modular fashion. This modularity is created by separate but communicating services described above. Each service is powered by its own thread but the execution of these threads is tied to the rate at which new depth images becomes available. All service threads wait for the next depth image once all required computations for the current image are done. The architectural complexity of this setup is made relatively simple by the fact that no memory resource is written to by more than one thread even though several threads might read that memory. In general, a thread may read from objects attached to other services but may only write to the object attached to the own service.
The services used by the system are displayed in Fig. 3.3 and consists of one head point tracker coupled with a Head point verification, two hand point trackers, each coupled with a Hand point verification, and two Hand motion classifiers. Fig. 3.3 also shows how data are passed between services. Once the Kinect manager signals that a new image is ready, all the services fetch it. The head point tracker provides the head point, as described in section 3.5.1, for other services to read. One of these services is the head point verification service. It uses the head point together with the depth image to determine if the head point is reasonably attached to a head, as described in section 3.5.2. The head verification service announces completed verification to the head tracker and to the event engine.
The second service that makes use of the head point, in addition to the depth data, is the hand point tracker, as described in section 3.5.3. The hand point tracker announces events to the event engine when hands are raised and lowered. The hand point provided by the hand point tracker is then in turn used by the hand verification service together with the head point in order to verify the hand. Similarly to the head point verification, the verification information is then announced to the hand point tracker and the event engine. The last service, which only uses the hand point, is the hand motion classifier and its function is specified in section 3.5.5. This service

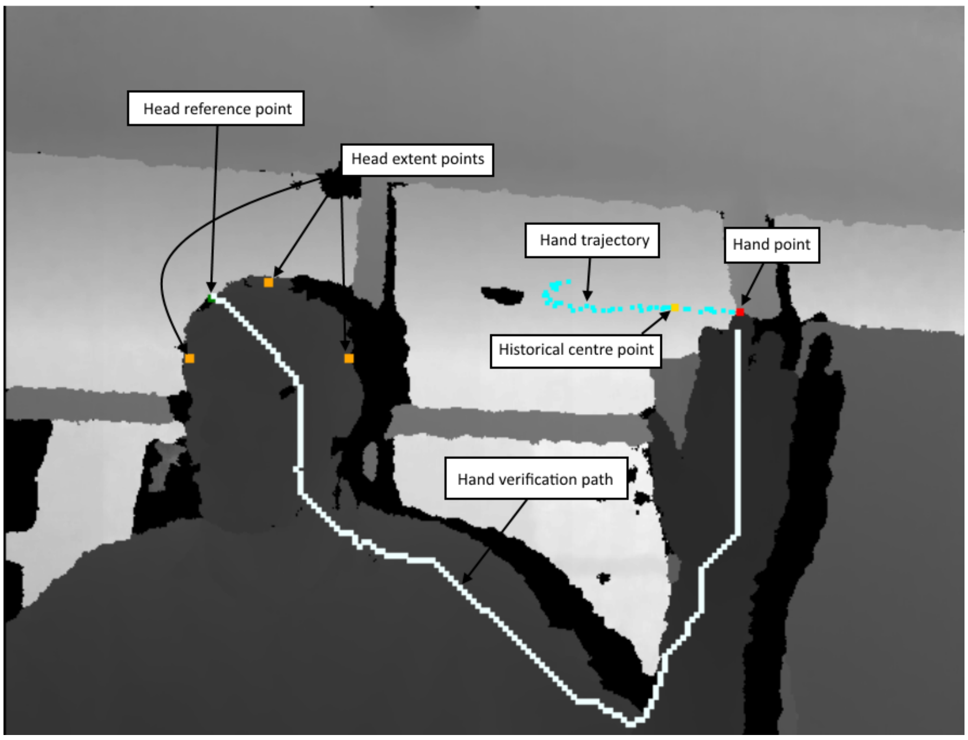

Figure 3.4: The system's debug view with all the visual representations of the components visible and labelled. Note that the labels have been added for clarification; they do not appear in the actual debug view.

announces an event to the event engine when a hand is moved.

3.7 Debug view

The system has a debug view that is shown in Fig. 3.4. The initial purpose of the debug view was to facilitate the development of the system by making it possible to quickly determine how a feature functions. The debug view is constructed from the raw depth image provided by the Kinect with representations of the system's different components drawn upon it. The different components are named in the figure. The included components consists of the head reference point provided by the head tracker, the head extent points which are placed by the head verification, the hand point and the history of hand points that forms the hand trajectory provided by the hand tracker, the hand verification path that is created by the hand verification and finally, the Historical centre point that is used by the hand movement classifier. This debug view is available as a function of the service manager and both colour and depth images are available for use with any possible future application.

4

Experimental setup

4.1 Error analysis

4.1.1 Possible error sources

The system has several possible modes of failure. These modes can be divided into three categories, head misidentification error, hand misidentification error and verification error.

4.1.1.1 Head identification error

Head misidentification error occurs when the point the head tracker takes to represent the head does not correspond to the user's head. This can happen if the user is not in the foreground, either because the user has left the picture entirely or if there is another object in front of the user. The head tracker relies on the head verification to detect this error. This error causes the verification to put the head tracker into search mode until a correct point is found. This means that the system will not function normally until the situation that first caused the error is corrected.

4.1.1.2 Hand identification error

Hand misidentification error is what happens when the system is correctly tracking the user's head but fails to register a hand. This error has two types. In the he true negative, the user attempts to raise a hand but the system fails to detect it, this can happen if if the user has not raised the hand properly and it is still too close to the body. The second error type occurs when the hand tracker detects an object that is not a hand. The most likely cause for this is that an object enters the foreground. This error is handled by the hand verification that will simply cause the system to ignore any object that is not connected to the head point without passing over the background, i.e. it disqualifies any object that is not connected to the user.

4.1.1.3 Verification error

Verification is a set of processes that are designed to catch errors in the other parts of the system and to let it recover from them. The verification processes are separate for the head and hand trackers and they are not immune to errors themselves.
Verification errors comes in two types, the first error type occurs when the verification fails for a valid point. This can happen for the head point verification if the head has an unusual shape, for example if the user is wearing a hat or has a large

hair cut. For the hand point verification, this can happen if the user's entire arm is not inside the Kinect's view and the hand therefore appears separate from the body. The second error type occurs when the verification passes for an invalid point. This can happen for the head point verification if the detected object happens to be shaped like a head even though it is not the user's head.

4.1.2 Effect and severity analysis

The possible errors are ranked by severity as follows:
Head point verification errors are the most severe because they cause the system to stop functioning nominally. The most severe error is a false positive error for the head point verification. With this error, the system is in a state where it is not indicated that it is in error and detection of raised hands will not function. Recovering from this error consists of removing the object the system tracks as a head.
A true negative error for the head verification means that the current head point is taken to be erroneous and the system will search for a new point continuously. The system will not function because it will not be able to register raised hands in this state. A false positive error for hand point verification results in the system registering a raised hand where there is none. This can be very disrupting for the user experience because it issues unintended control input. This error is recovered from by the removing the detected object.
A true negative error for hand point verification causes the system to fail to register a raised hand. This error is less disruptive than the false positive error because instead of something unexpected happening, something expected failed to happen. This error is commonly caused by that the entire arm is not visible by the Kinect and is recovered from by ensuring the user is in centre view of the Kinect.
Errors that do not occur for the verification but for the head and hand trackers are less severe because they will be automatically recovered from if the user returns to default position in centre view of the Kinect with both arms lowered. These errors can, however, still be disrupting. The system will not function during a head misidentification error but it will recognise the error as long as the head point verification works and will be able to advice the user to return to the default position. The system will not be able to detect raised hands during a true negative hand misidentification error. This error is recovered from by the user raising the hand more, separating it further up and to the side away from the body. A false positive hand misidentification error occurs when an object enters the expected hand regions. The error is recovered by removing the object.

4.2 Evaluation protocol for user interaction

It is necessary to test under what circumstances the system enters the different error modes in order to determinate the system's robustness. These test should be conducted by letting a user unfamiliar with the technical details of the system perform a scenario resembling normal operation. The scenario consists of the following steps and fulfilling a number of criteria at each step:

1. Set up the system by plugging in the Kinect, directing the Kinect towards the empty user's seat and start up the software.
    - Ensure that the head point verification fails for as long as there is not a user in the Kinect's view.
2. Let the user take a seat in view of the Kinect.
    - Ensure that the head point is placed on top of the user's head.
    - Ensure that the head point verification succeeds.
    - Ensure that no hand point is registered by the hand tracker.
    - If a hand point is registered by the hand tracker, note if the hand point verification fails or not.
3. Let the user raise one hand.
    - Ensure that the system detects the raised hand.
    - If the system does not detect the raised hand, determine if this is caused by hand point verification error or hand misidentification.
4. Instruct the user move the raised hand in specific directions.
    - Ensure that the system detects the movements in the specified directions.
    - Note all other movements detected by the system
5. Let the user lower the raised hand
    - Ensure that the system registers the lowering of the hand.
6. Repeat step 3. and 4. for the other hand.

Failure to meet any of the above criteria should be noted together with how well it recovers and the cause of the error. Errors can be insignificant, with automatic recovery and without the user noticing; minor, noticed by the user but can be recovered by when the user returns to the default position and major, the user notices and is unable to recover the system without instruction. The system needs to be able to pass the above scenario with at most only minor errors in order to be considered robust.

4.3 Evaluation protocol of the technical operation

The technical operation is evaluated in order to determine how the system fulfils the criterium provided in Section 3.1.1 to ensure that the system is perceived as responsive and fluid. This property is determined by the time elapsed between a user performing an action and the action being registered by the system. According to Miller [**?**]the maximum time a system can take between a user action and response, without disrupting the user with a perceived delay is one second and this value is taken as the upper allowed time limit for the system's response time. This definition excludes the time to perform an action, e.g. raise a hand, because the tested property is not how fast the system can be operated by a user but how the system is responsive to the user's actions.

This test is practically performed by using a timer that is started when the user presses a key. The user presses the key at the moment the user completes an action is completed. The timer then runs until an action is registered by the system. The time reached by the timer when the action is registered is the response time. The response time of pressing a key on a modern computer is very small, less than 10 ms, and is insignificant to the required precision of the measurement. The actions to be timed are

- Raising of hand
- Movement of hand
- Lowering of hand

Each action is timed at least 10 times and average, maximum and minimum times are noted.

5
Results and discussion

5.1 User interaction evaluation results

The system was tested using the protocol provided in Chapter 4 using four test subjects. The subjects were fellow students. The subjects was instructed to take place in front of the Kinect and to follow the instructions. The test subjects all had very similar experience with the system and the outcome was documented at each step of the evaluation the system's practical performance.

1. The system functioned correctly after set up and did not erroneously detect a head that was not present for any of the test subjects.
2. All test subjects has successfully registered by the system after seating themselves in front of the Kinect after some adjustments made to the head verification to allow for less wide heads.

   The system made no spurious hand reading for any of the subjects while sitting normally with hands down in front of the Kinect. The System did however detect objects in the periphery of the Kinect view as raised hands, but because these objects failed verification, they were not registered.
3. The system suffered some issues with registering raised hands. The test subjects, when instructed to raise their hand with no further specification, tended to keep it too close to the body for the system to register it. This issue was overcome for all subjects by displaying the system's graphical interface and register log combined with the instruction to raise the hand "more" until the system responded. The subjects was able to register a raised their hands at will after learning the necessary extent a hand needed to be raised to be successfully registered.

   One exception to this was a problem with the hand verification that occurred on a few occasions. The problem occurred because of bad positioning of the Kinect sensor. The Kinect was aimed to much upwards and failed to capture the lower part of the test subject which resulted in that the the elbow of the subject's raised arm was out of view of the Kinect. Because the arm had no visible connection to the head but extended from what appeared to be from out of the image, the hand verification failed.
4. The system was not universally able to register the subjects' movements when moving intuitively. In other words, the user's intuitive movements in specific directions was not registered as such by the system. Later, the subjects was able to reliably register movements after the they had some time to learn the correct motions.

5. The system was able to successfully register the lowering of hands for all subjects.
6. Once the system was understood for the right hand, the subjects had no issues in using the left hand in a similar capacity.

5.2 Technical evaluation results

The results of the different tests performed according to the procedure in Section 4.3:

Raising hand The hand raising action was performed 18 times. The longest time recorded between the user completed raising the hand was 1089 ms and the shortest recorded time was 618 ms. The average recorded time was 890 ms with a standard deviation of 114 ms .

Moving hand The hand was moved 12 times and the system registered the movement before it was deemed complete by the user every time. In terms of this test, hand movements are recorded instantaneously upon completion of the action.

Lowering hand The hand lowering action was performed 20 times. The largest time recorded was 615 ms for the user to lower the hand and the shortest was 147 ms. The average recorded time was 445 ms with a standard deviation of 162 ms.

5.3 Discussion

The user interaction tests show that the system is fairly robust but it has a learning curve. It is not reasonable to expect a user with no experience with the system to be able to use it without some introduction. The necessary introduction time for users to become accustomed with the workings of the system was reduced by displaying for the user the debug screen showing the the depth image of the user and the user's actions as they are registered.

Part of the learning curve is related to the setting up the system. It is not entirely trivial to set up the Kinect sensor such that the detection system works. If the Kinect is aimed too high, the system might be unable to register the raising of a hand because part of the arm falls outside of the Kinect's view. If the Kinect on the other and is aimed too low, there is the possibility that the system fails to track the head if it falls outside the image.

The combination of these two issues puts into question how well the system fulfils the requirement for intuitiveness. It is apparent that a completely unfamiliar user might struggle to effectively use the system. This seems to be a problem that is not isolated to this system but indeed to the Kinect as a control device for games at large. It is a nice way of expanding the possible range of user control, but it has been consistently unreliable throughout its history.

For the intention of creating a user interface to be primarily used by users with unknown level of physical impairment and experience with technology, any sort of touchless control device, which includes the Kinect, should only be adopted with serious caution. The simple magnitude of movements required for such a system to function should create concern for user ergonomy. A Kinect-based control system keeps the user physically and this may be desired, but it should probably never be used without the option for a physical controller.

6
Conclusion and future work

6.1 Conclusion

This project has sought to create a system that provides a touchless control device for use with a partner agent. This has been achieved with the use of a Kinect and the depth image it produces. The basic problem that the created system is meant to solve is how to reliably track a user. The approach this project has taken to resolve this problem has been to create the system based on only very simple principles. The system is fundamentally built around reducing the complexity of the data provided by the Kinect's depth image into a few manageable points representing the head and possible raised hands. In the process of producing these points, the system uses minimal knowledge of human anatomy, i.e. the system does not have an internal model for the shape of the human body. Instead, hand and head points are simply selected based on the regions they appear in and accepted if they pass verification. This naïvety of the system to avoid tracking a human body is both a strength and a weakness. The strength is that it makes the system very predictable; a user who is familiar with the system will not be surprised by its behaviour because of its simplicity. The downside is that this system might be difficult to use without that familiarity. This makes system only partially fulfils the requirements of the system presented in Section 3.1. It is functional but is not necessarily practical for an inexperienced user. Furthermore, it is questionable whether or not learning to use this system is worth the effort compared to using, for example, keyboard and mouse or a console controller. Control methods such as those are as easy to learn but offer much more precise control and are less physically straining.

6.2 Future work

Because of its simplicity, the developed method for gesture recognition shows limited potential to be further developed. Any sort of improvement would have to include some sort of tracking of the user's body which this method specifically sought to avoid. For the development for a control device for a partner agent, it is suggested that a review of physical handheld devices is made and be tested directly with the intended senior target demographic. Controllers that should be of interest should include standard keyboard and mouse, console controllers for XBox and Playstation, and fighting sticks (specially designed controllers made to mimic the controls on arcade machines).

Bibliography

[1] M. Yang, Z. Lin, W. Tang, L. Zheng, J. Zhou, Human action recognition based on kinect, Journal of Computational Information Systems 10 (12) (2014) 5347–5354.
URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-84904725758&partnerID=40&md5=ea7e0e6ddfb9a08d43a70513b211c131`

[2] C. del Blanco, T. Mantecón, M. Camplani, F. Jaureguizar, L. Salgado, N. García, Foreground segmentation in depth imagery using depth and spatial dynamic models for video surveillance applications, Sensors (Switzerland) 14 (2) (2014) 1961–1987.
URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-84892985659&partnerID=40&md5=650e528817a797a0985ecd487c420f15`

[3] M. Bengalur, Human activity recognition using body pose features and support vector machine, 2013, pp. 1970–1975.
URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-84891940845&partnerID=40&md5=c31c4d8e267091e1b01604439cd1e602`

[4] A. Ben Hadj Mohamed, T. Val, L. Andrieux, A. Kachouri, Assisting people with disabilities through kinect sensors into a smart house, 2013.
URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-84877799663&partnerID=40&md5=baa580249ab9e98d53e27bc164a92ac0`

[5] N. Gonçalves, S. Costa, J. Rodrigues, F. Soares, Detection of stereotyped hand flapping movements in autistic children using the kinect sensor: A case study, 2014, pp. 212–216.
URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-84904985351&partnerID=40&md5=64fbb4a2f5280282d90655a9368ac070`

[6] L. Pedro, G. De Paula Caurin, Kinect evaluation for human body movement analysis, 2012, pp. 1856–1861.
URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-84867422471&partnerID=40&md5=dd80f8e9437bb70678ab1d98cdb08db9`

[7] G. Du, P. Zhang, Markerless human-robot interface for dual robot manipulators using kinect sensor, Robotics and Computer-Integrated Manufacturing 30 (2) (2014) 150–159.
URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-84886533350&partnerID=40&md5=1b2d0705dc11d6ea9d012a5ff30e5b9b`

[8] S. Zolkiewski, D. Pioskowik, Robot control and online programming by human gestures using a kinect motion sensor, Advances in Intelligent Systems and Computing 275 AISC (VOLUME 1) (2014) 593–604.

URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84904599612&partnerID=40&md5=1e1c473cd14a836bf07efa9ace884c49

[9] P. Gil, C. Mateo, F. Torres, 3d visual sensing of the human hand for the remote operation of a robotic hand, International Journal of Advanced Robotic Systems 11 (1).
URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84897565594&partnerID=40&md5=2a2e1a9b53076f774fb3f5d68ef610af

[10] C. Guerrero-Rincon, A. Uribe-Quevedo, H. Leon-Rodriguez, J.-O. Park, Hand-based tracking animatronics interaction, 2013.
URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84893294993&partnerID=40&md5=335c6e6f13772bfe57dfc5033c4b1034

[11] K. Qian, H. Yang, J. Niu, Developing a gesture based remote human-robot interaction system using kinect, International Journal of Smart Home 7 (4) (2013) 203–208.
URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84883235966&partnerID=40&md5=bb8e9f5006af7ea04623c9a89a89d7fd

[12] C.-K. Lim, Application of kinect technology in the design of interactive products for chinese senior citizens, Communications in Computer and Information Science 373 (PART I) (2013) 51–55.
URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84891553469&partnerID=40&md5=fb652c64b47066934d5396b4072a4930

[13] L. Bartoli, C. Corradi, F. Garzotto, M. Valoriani, Exploring motion-based touchless games for autistic children's learning, 2013, pp. 102–111.
URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84880566517&partnerID=40&md5=2411f729aa464c8c81ea7df7a66e1620

[14] B. Hariharan, S. Padmini, U. Gopalakrishnan, Gesture recognition using kinect in a virtual classroom environment, 2014, pp. 118–124.
URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84902589003&partnerID=40&md5=0db22fbf3dab302ae5cf94271628c42a

[15] A. Meneses Viveros, E. Hernández Rubio, Kinect© as interaction device with a tiled display, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 8007 LNCS (PART 4) (2013) 301–311.
URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84880658232&partnerID=40&md5=78188a76454c8be1b6854a0bead332ec

[16] X. Guo, Z.-Y. Liu, Z.-Y. Lu, J.-G. Wu, Big-screen text input system based on gesture recognition, Huadong Ligong Daxue Xuebao/Journal of East China University of Science and Technology 39 (5) (2013) 583–587+640.
URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84890326150&partnerID=40&md5=906a2e2b6ff20ed156dc442d44b5fc3d

[17] B. Juhnke, M. Berron, A. Philip, J. Williams, J. Holub, E. Winer, Comparing the microsoft® kinect™ to a traditional mouse for adjusting the viewed tissue densities of three-dimensional anatomical structures, Vol. 8673, 2013.
URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84878813668&partnerID=40&md5=8fbbb74f8d349fc92bc53ea9bbbc0d9d

[18] H. Tao, Y. Yu, Finger tracking and gesture recognition with kinect, 2012, pp. 214–218.
URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84872355644&partnerID=40&md5=28ef5cd39a5fcceeabd61e7e64e666aa

[19] K. Silanon, N. Suvonvorn, Fingertips tracking based active contour for general hci application, Lecture Notes in Electrical Engineering 285 LNEE (2014) 309–316.
URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84893797125&partnerID=40&md5=b62d9612ed2a0e77737b06fc420b1975

[20] M. Maisto, M. Panella, L. Liparulo, A. Proietti, An accurate algorithm for the identification of fingertips using an rgb-d camera, IEEE Journal on Emerging and Selected Topics in Circuits and Systems 3 (2) (2013) 272–283.
URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84879112921&partnerID=40&md5=d595501597df814122af21a6fb5c5dc0

[21] H. Liang, J. Yuan, D. Thalmann, 3d fingertip and palm tracking in depth image sequences, 2012, pp. 785–788.
URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84871379420&partnerID=40&md5=874c1f3a3362a4129d56196cb37fbfaf

[22] J. Raheja, A. Chaudhary, K. Singal, Tracking of fingertips and centers of palm using kinect, 2011, pp. 248–252.
URL http://www.scopus.com/inward/record.url?eid=2-s2.0-83155190395&partnerID=40&md5=b5edbf7b4531a37ec960795907d972f6

[23] E. Davies, Chapter 1 - vision, the challenge, in: E. Davies (Ed.), Computer and Machine Vision (Fourth Edition), fourth edition Edition, Academic Press, Boston, 2012, pp. 1 – 14.
URL http://www.sciencedirect.com/science/article/pii/B978012386908100001X

[24] J. Suarez, R. Murphy, Hand gesture recognition with depth images: A review, in: RO-MAN, 2012 IEEE, 2012, pp. 411–417.

[25] K. Rodriguez, G. Chavez, Finger spelling recognition from rgb-d information using kernel descriptor, 2013, pp. 1–7.
URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84891538211&partnerID=40&md5=7fd5b8d9c7b59e9648ce662587c0338d

[26] H. Hasan, S. Kareem, Gesture feature extraction for static gesture recognition, Arabian Journal for Science and Engineering 38 (12) (2013) 3349–3366.
URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84887312339&partnerID=40&md5=37a79d1d6e03f2baade29642c66bd1d4

[27] J. Dou, J. Li, Robust human action recognition based on spatio-temporal descriptors and motion temporal templates, Optik 125 (7) (2014) 1891–1896.
URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84892793983&partnerID=40&md5=2114220d110b4393cb327fbce402bbd6

[28] H. Chen, G. Wang, L. He, Accurate and real-time human action recognition based on 3d skeleton, Vol. 9045, 2013.
URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84892615557&partnerID=40&md5=525dc0f382fd51cce43ef9987745d958

[29] Q. Zhu, T. Liu, Gesture recognition and application research of kinect, Advanced Materials Research 926-930 (2014) 1534–1537.
URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-84902297120&partnerID=40&md5=6b94e63d300ce89e84f59fd27a86146f`

[30] F.-A. Huang, C.-Y. Su, T.-T. Chu, Kinect-based mid-air handwritten digit recognition using multiple segments and scaled coding, 2013, pp. 694–697.
URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-84894182108&partnerID=40&md5=df325ba865d04bab980d6d57f807836c`

[31] J. Wang, Y. Xu, D. Wang, W. Guo, Real-time recognition of human actions via multi-pivot skeleton model, Huazhong Keji Daxue Xuebao (Ziran Kexue Ban)/Journal of Huazhong University of Science and Technology (Natural Science Edition) 41 (SUPPL.I) (2013) 144–148.
URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-84890834714&partnerID=40&md5=06c79889d6526374a22b83265cd627d0`

[32] S. Nasri, A. Behrad, F. Razzazi, Spatio-temporal 3d surface matching for hand gesture recognition using icp algorithm, Signal, Image and Video Processing (2013) 1–16.
URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-84890303793&partnerID=40&md5=96c3bd783d09af2659f10afe650c2cb4`

[33] I. Jargalsaikhan, S. Little, C. Direkoglu, N. O'Connor, Action recognition based on sparse motion trajectories, 2013, pp. 3982–3985.
URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-84897747046&partnerID=40&md5=ed07dae42623433fe50871aa395f82e6`

[34] J. Beh, D. Han, R. Durasiwami, H. Ko, Hidden markov model on a unit hypersphere space for gesture trajectory recognition, Pattern Recognition Letters 36 (1) (2014) 144–153.
URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-84893049119&partnerID=40&md5=98d47d10d88225314f68cc29d5f4772d`

[35] H. Gamal, H. Abdul-Kader, E. Sallam, Hand gesture recognition using fourier descriptors, 2013, pp. 274–279.
URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-84893686436&partnerID=40&md5=57227050e013109a7f3effc7e557629e`

[36] C. Li, B. Su, Y. Liu, H. Wang, J. Wang, Human action recognition using spatio-temoporal descriptor, Vol. 1, 2013, pp. 107–111.
URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-84897789004&partnerID=40&md5=accdf69a15cff4f9cff5688cdc83b5b8`

[37] K. Sgouropoulos, E. Stergiopoulou, N. Papamarkos, A dynamic gesture and posture recognition system, Journal of Intelligent and Robotic Systems: Theory and Applications (2013) 1–14.
URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-84890263424&partnerID=40&md5=a20ca9b56877f2d85fc5ac3c88e84e8b`

[38] S. McCann, D. Lowe, Local naive bayes nearest neighbor for image classification, in: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, 2012, pp. 3650–3656.

[39] X. Yang, Y. Tian, Eigenjoints-based action recognition using naive-bayes-nearest-neighbor, in: Computer Vision and Pattern Recognition Workshops

(CVPRW), 2012 IEEE Computer Society Conference on, 2012, pp. 14–19.

[40] K. Nandhini, N. Raajan, Gesture based life saving approach, International Journal of Applied Engineering Research 8 (17) (2013) 2027–2034.
URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84891643081&partnerID=40&md5=17f23bded296b16dff4aefb2312ac4ce

[41] Q. Zhang, H. Wei, Z. Xu, M. Zhang, H. Duan, L. Lv, Research on hand gesture recognition based on inner-distance shape context and bag of words model, Journal of Information and Computational Science 11 (9) (2014) 2895–2904.
URL http://www.scopus.com/inward/record.url?eid=2-s2.0-84903436850&partnerID=40&md5=2c69d534721f189f78330bc2cdb66ee2

[42] O. Boiman, E. Shechtman, M. Irani, In defense of nearest-neighbor based image classification, in: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, 2008, pp. 1–8.

[43] W. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, The bulletin of mathematical biophysics 5 (4) (1943) 115–133.
URL http://dx.doi.org/10.1007/BF02478259

[44] J. Sivic, A. Zisserman, Efficient visual search of videos cast as text retrieval, Pattern Analysis and Machine Intelligence, IEEE Transactions on 31 (4) (2009) 591–606.

[45] C. Cortes, V. Vapnik, Support-vector networks, Machine Learning 20 (3) (1995) 273–297.
URL http://dx.doi.org/10.1007/BF00994018