

Fusion of Range and Intensity Information for View Invariant Gesture Recognition

M.B. Holte, T.B. Moeslund and P. Fihl
Computer Vision and Media Technology Laboratory
Aalborg University, Denmark
t.bm@cvmt.dk

Abstract

This paper presents a system for view invariant gesture recognition. The approach is based on 3D data from a CSEM SwissRanger SR-2 camera. This camera produces both a depth map as well as an intensity image of a scene. Since the two information types are aligned, we can use the intensity image to define a region of interest for the relevant 3D data. This data fusion improves the quality of the range data and hence results in better recognition. The gesture recognition is based on finding motion primitives in the 3D data. The primitives are represented compactly and view invariant using harmonic shape context. A probabilistic Edit Distance classifier is applied to identify which gesture best describes a string of primitives. The approach is trained on data from one viewpoint and tested on data from a different viewpoint. The recognition rate is 92.9% which is similar to the recognition rate when training and testing on gestures from the same viewpoint, hence the approach is indeed view invariant.

1. Introduction

Automatic analysis of humans and their actions has received increasingly more attention in the last decade [12]. One of the areas of interest is recognition of human gestures for use in for example Human Computer Interaction.

Many different approaches to gesture recognition have been reported [11]. They apply a number of different segmentation, feature extraction, and recognition strategies, but are virtually all based on analyzing 2D data, i.e., images. A consequence of this is that approaches only analyze 2D gestures carried out in the image plane. To overcome this shortcoming the use of 3D data has been introduced through the use of two or more cameras, see for example [2, 15]. We follow this line of work and also apply 3D data. To avoid the difficulties inherent to classical stereo approaches (the correspondence problem, careful camera placement and cali-

bration) we instead apply a 3D range camera – CSEM SwissRanger SR-2. Each pixel in this camera directly provides a depth value (distance to object). Even though the technology in range cameras is still in its early days, e.g., resulting in low resolution data, the great potential of such sensors has already resulted in them being applied in a number of typical computer vision applications like face detection [5], face tracking [4], shape analysis [8, 9], and robot navigation [14].

The camera we apply also provides an amplitude value corresponding to an intensity value for each pixel. This means that at any given time instant both a depth image and an intensity image are present. For some applications these two information types compliment each other and are therefore both used. For example in [8] where the objective is to segment planar surfaces in 3D (range) data, the edges in the intensity image are applied to improve the result. Similar benefits of applying both data types can be seen in [4, 5]. We also apply both data types and will show how this improves our results.

Applying 3D data allows for analysis of 3D gestures. However, we are still faced with the problem that a user has to be fronto-parallel with respect to the camera unless we want to train classifiers for each possible viewpoint, which is obviously not desirable. Instead we aim at a view-invariant approach which is trained at examples from one viewpoint and able to recognize gestures from a very different viewpoint, say $+45^\circ$. Another issue we want to combat is the often used assumption of known start- and end points. That is, often the test data consists of N sequences where each sequence contains one and only one gesture. This obviously makes the problem easier and it favors a trajectory-based approach, where each gesture is represented as a trajectory though some state-space with known start and end point. For real-life scenarios the start and end point is obviously not known. To deal with this issue we follow the notion of recognition through a set of primitives [3, 6, 16]. Concretely, we define a primitive as a time instance with significant 3D motion.

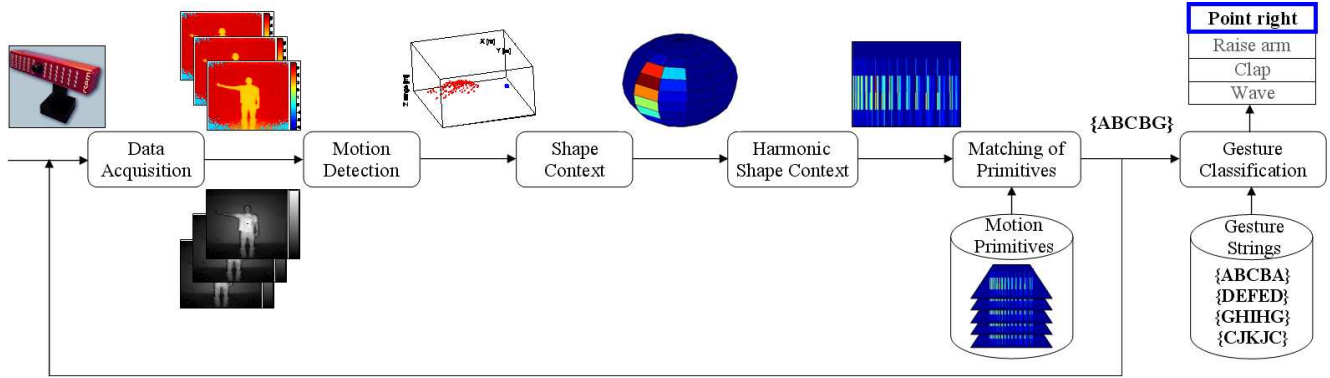


Figure 1. An overview of the range and intensity based gesture recognition system. Note that the feedback loop illustrates that a number of frames are processed before recognition of gestures commences.

So, we represent gestures as an ordered sequence of 3D motion primitives (temporal instances). We focus on arm gestures and therefore only segment the arms (when they move) and hereby suppress the rest of the (irrelevant) body information. Concretely we use 3D double difference images to extract the moving arms and represent this data by their Shape Context. We make the primitives invariant to rotation around the vertical axis by re-representing the Shape Context using Spherical Harmonic basis functions, yielding a Harmonic Shape Context representation. In each frame the primitive, if any, which best explains the observed data is identified. This leads to a discrete recognition problem since a video sequence of range data will be converted into a string containing a sequence of symbols, each representing a primitive. After pruning the string a Probabilistic Edit Distance classifier is applied to identify which gesture best describes the pruned string. Our approach is illustrated in figure 1.

2. Segmentation

2.1. Data acquisition

We capture intensity and 3D data using a CSEM SwissRanger SR-2 range camera (see figure 1) [13]. The camera is based on the Time-Of-Flight principle and emits radio-frequency modulated light in the near-infrared spectrum, which is backscattered by the scene and detected by a CMOS CCD. The resolution is 160×124 pixels with an active range of 7.5 m. The depth accuracy is typically in the order of a few centimeters, depending of the distance range and illumination. Figure 2 (top-left) and figure 2 (bottom-left) show a range and an intensity image of one time instant of a "point right" gesture, respectively.

2.2. 3D motion detection

We detect movements (of the arms) using a 3D version of 2D double differencing [7]. This is done by subtract-

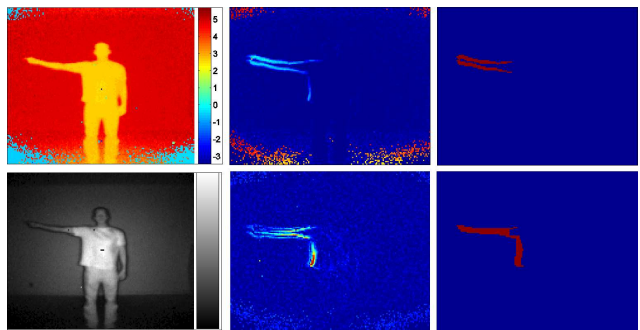


Figure 2. Top-left: A range image, where the pixel values correspond to a distance. Top-middle: A difference range image used for motion detection. Top-right: The resulting motion detected in 2D after hysteresis bandpass filtering and creation of a double difference image. Bottom-left: An intensity image. Bottom-middle: A difference intensity image used for ROI selection. Bottom-right: The resulting ROI in 2D after hysteresis highpass filtering and creation of a double difference image.

ing the depth values pixelwise in two pairs of depth images (see figure 2 (top-middle)), thresholding and finally AND-ing the two binary images. The moving arms (and their shadows) are visible in the binary image, but so is a large amount of noise due to erroneous depth values often produced by the SwissRanger camera. To handle these noise effects, each of the two 3D difference images is filtered with a hysteresis bandpass filter before they are ANDed together (see figure 2 (top-right)). This filter operates in 2D and uses four threshold values T_1, T_2, T_3 and T_4 . The 3D difference values that fall within the motion range $[T_2, T_3]$ are most likely to originate from arm movements. Pixels in the range $[T_1, T_2] \cup [T_3, T_4]$ are also classified as belonging to the arm if and only if they are connected with pixels from $[T_2, T_3]$. This hysteresis principle yields less fragmented motion regions while excluding noisy image regions. Too small motion regions caused by noise or unwanted motion along the body are filtered by a size criterion.

2.3. Falsified distance measurements

After detecting the motion we are left with a point cloud in 3D (see figure 3). However, the points included in the motion cloud are not always perfectly located at the arms.

The camera sensor is controlled as a so-called 1-tap sensor. This means that in order to obtain distance information, four consecutive exposures have to be performed [13]. If the distance a pixel "sees" changes in this time window, the distance calculation is falsified. Fast moving objects in the scene may therefore cause errors in the distance calculations. This error is inversely proportional to the frame rate. In practice, the distance error is best seen in regions of a scene that contain objects with high velocity and distance gradients. In our case a fast moving arm. The pixel could then see the arm for the first two taps and a wall for the last two. Thus, the edges of the moving arm are poorly defined and lead to incorrect distance measurement. Concretely, when visualizing the range data as a 3D point cloud, the points origin from these regions are "stretching" backwards along the edges of the moving arm. An example of a point cloud extracted from one time instant of a "point right" gesture is shown in figure 3 (top).

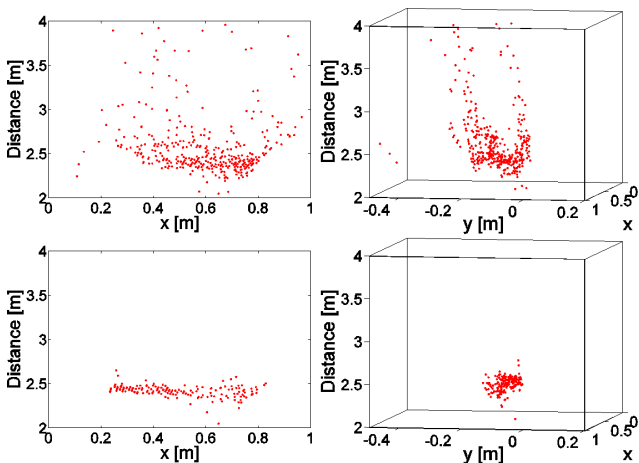


Figure 3. Top: A resulting 3D point cloud (seen from two different view points), after motion detection in the range data as shown in figure 2 (top-row), including falsified error measurements. Bottom: The point cloud after extending the motion detection with an intensity based ROI selection 2 (bottom-row).

2.4. Intensity based ROI selection

To overcome these falsified distance measurements we use the already available intensity information. Specifically, we use the intensity images to select a Region Of Interest (ROI) for motion detection in the range data. This ROI can be used directly like a mask to select motion of interest in the range data, due to the fact that the intensity and range images are aligned.

The idea is, that the ROI estimated in the intensity image will not include these falsified distance measurements. When detecting motion in the range image, the erroneous distance values will cause false motion effects. As a result these false motion pixels will be mapped to 3D and hereby include data points with incorrect distances. In contrast the intensity information do not suffer from this problem. Although it should be noted, that the camera produces intensity images of very low quality, and therefore this type of data is not used for motion detection on its own but solely for ROI selection.

The ROI is estimated by the same technique used to extract motion in the range data, namely image differencing. Again we create a double difference image by subtracting the values pixelwise in two pairs of intensity images (see figure 2 (bottom-middle)), thresholding and ANDing the two binary images. Noise is handled like before by a hysteresis filter. However, in this case we are interested in the pixels including the largest differences. Hence, as opposed to the other bandpass filter, this filter operates as a highpass filter with two thresholds (see figure 2 (bottom-right)). Figure 3 (bottom) shows an example of the 3D point cloud after the intensity based ROI selection has been applied. The figure illustrates how effective the ROI eliminates data points with incorrect distances, and hence how the two data types compliment each other.

3. Motion primitives

3.1. Shape context

The 3D point cloud is represented efficiently using shape context [1]. A shape context is based on a spherical histogram. This histogram is centered in a reference point (center of gravity of the human body) and divided linearly into $S = 12$ azimuthal (east-west) bins and $T = 12$ colatitudinal (north-south) bins, while the radial direction is divided into $U = 5$ bins. The radial division is made in steps of 20 cm. The value of a bin is given by the number of 3D points falling within that particular bin. This results in an n ($S \times T \times U = 12 \times 12 \times 5 = 720$) dimensional feature vector for each frame. Figure 4 gives an example of the shape context descriptor.

3.2. View invariant representation: harmonic shape context

By introducing spherical harmonics we can eliminate one of the two rotational parameters in a shape context descriptor. We eliminate the rotation around the vertical axis, see figure 4, and hereby make our representation invariant to variations in this parameter.

Any given spherical function, i.e. a function $f(\theta, \phi)$ defined on the surface of a sphere parameterized by the colatitudinal and azimuthal variables θ and ϕ , can be decom-

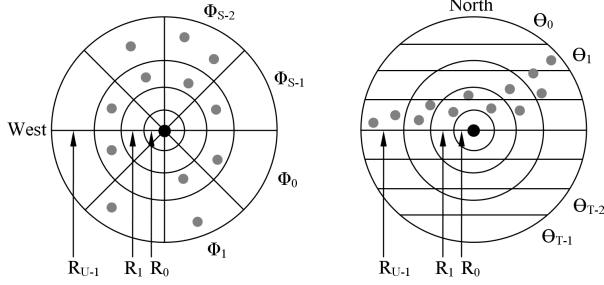


Figure 4. A horizontal and a vertical cross-section of a Shape context descriptor.

posed into a weighted sum of spherical harmonics as given by equation 1.

$$f(\theta, \phi) = \sum_{l=0}^{\infty} \sum_{m=-l}^l A_l^m Y_l^m(\theta, \phi) \quad (1)$$

The term A_l^m are the weighing coefficient of *degree* m and *order* l , while the complex functions $Y_l^m(\cdot)$ are the actual spherical harmonic functions of *degree* m and *order* l . The following states the key advantages of the mathematical transform based on the family of orthogonal basis functions in the form of spherical harmonics. The complex function $Y_l^m(\cdot)$ is given by equation 2.

$$Y_l^m(\theta, \phi) = K_l^m P_l^{|m|}(\cos \theta) e^{jm\phi} \quad (2)$$

The term K_l^m is a normalization constant, while the function $P_l^{|m|}(\cdot)$ is the *associated Legendre Polynomial*. The key feature to note from equation 2 is the encoding of the azimuthal variable ϕ . The azimuthal variable solely inflects the *phase* of the spherical harmonic function and has no effect on the *magnitude*. This effectively means that $\|A_l^m\|$, i.e. the norm of the decomposition coefficients of equation 1 is invariant to parametrization in the variable ϕ .

The actual determination of the spherical harmonic coefficients is based on an inverse summation as given by equation 3, where N is the number of samples ($S \times T$). The normalization constant $4\pi/N$ originates from the fact, that equation 3 is a discretization of a continuous double integral in spherical coordinates, i.e. $4\pi/N$ is the surface area of each sample on the unit sphere.

$$(A_l^m)_{f_u} = \frac{4\pi}{N} \sum_{\phi=0}^{2\pi} \sum_{\theta=0}^{\pi} f_u(\theta, \phi) Y_l^m(\theta, \phi) \quad (3)$$

In a practical application it is not necessary (or possible, as there are infinitely many) to keep all coefficient A_l^m . Contrary, it is assumed the functions f_u (f_u are the spherical functions for $u \in [0; U-1]$) are band-limited why it is only necessary to keep coefficient up to some bandwidth. Concretely we use 136 coefficients (see figure 1).

4. Classification

The classification is divided into two main tasks: recognition of motion primitives by use of the harmonic shape context descriptors, and recognition of the actual gestures using an ordered sequence of primitives (see figure 1).

4.1. Recognition of primitives: correlation

A motion primitive is recognized by matching the current harmonic shape context with a known set, one for each possible primitive. The actual comparison of two harmonic shape contexts is done by the normalized correlation coefficient as given by equation 4. To this end each harmonic shape context is represented as a vector \mathbf{h}_1 and \mathbf{h}_2 of length n containing the (stacked) spherical harmonic coefficients for a specific primitive at time, t :

$$\text{match}(\mathbf{h}_1, \mathbf{h}_2, t) = \quad (4)$$

$$\frac{n \sum \mathbf{h}_1 \mathbf{h}_2 - \sum \mathbf{h}_1 \sum \mathbf{h}_2}{\sqrt{[n \sum (\mathbf{h}_1)^2 - (\sum \mathbf{h}_1)^2] [n \sum (\mathbf{h}_2)^2 - (\sum \mathbf{h}_2)^2]}}$$

The system is trained by generating a representative set of descriptors for each primitive. A reference descriptor is then estimated as the average of all these descriptors for each class (primitive).

The classification of a sequence can be viewed as a trajectory through the feature space where, at each time-step, the closest primitive (in terms of the normalized correlation coefficient) is found. To reduce noise in this process we introduce a minimum coefficient in order for a primitive to be considered in the first place. Furthermore, to reduce the flickering observed when the trajectory passes through a border region between two primitives we introduce a hysteresis threshold. It favors the primitive recognized in the preceding frame over all other primitives by modifying the individual distances. The classifier hereby obtains a "sticky" effect, which handles a large part of the flickering.

After processing a sequence the output will be a string with the same length as the sequence. An example is illustrated in equation 5. Each letter corresponds to a recognized primitive and \emptyset corresponds to time instances where no primitives are detected. The string is pruned by first removing ' \emptyset 's, isolated instances, and then all repeated letters, see equation 6. A weight is generated to reflect the number of repeated letters (this is used below).

$$\text{String} = \{\emptyset, \emptyset, B, B, B, B, B, E, A, A, F, F, F, F, \emptyset, D, D, G, G, G, G, \emptyset\} \quad (5)$$

$$\text{String} = \{B, A, F, D, G\} \quad (6)$$

$$\text{Weights} = \{5, 2, 4, 2, 4\} \quad (7)$$

4.2. Recognition of gestures: probabilistic edit distance

The result of recognizing the primitives is a string of letters referring to the known primitives. During a training phase a string representation of each gesture to be recognized is learned. The task is now to compare each of the learned gestures (strings) with the detected string. Since the learned strings and the detected string (possibly including errors!) will in general not have the same length, the standard pattern recognition methods will not suffice. We therefore apply the Edit Distance method [10], which can handle matching of strings of different lengths.

The edit distance is a well known method for comparing words or text strings, e.g., for spell-checking and plagiarism detection. It operates by measuring the distance between two strings in terms of the number of operations needed in order to transform one to the other. There are three possible operations: *insert* a letter from the other string, *delete* a letter, and *exchange* a letter by one from the other string. Whenever one of these operations is required in order to make the strings more similar, the score or distance is increased by one. The algorithm is illustrated in figure 5 where the strings *motions* and *octane* are compared.

The first step is initialization. The two strings are placed along the sides of the matrix, and increasing numbers are placed along the borders beside the strings. Hereafter the matrix is filled cell by cell by traversing one column at a time. Each cell is given the smallest value of the following four operations:

Insert: The value of the cell above + 1

Delete: The value of the cell to the left + 1

Exchange: The value of the cell up-left + 1

No change: The value of the cell up-left + 0. This is the case when the letters in question in the two strings are the same.

Using these rules the matrix is filled and the value found at the bottom right corner is the edit distance required in order to map one string into the other, i.e., the distance between the two strings. The actual sequence of operations can be found by back-tracing the matrix. Note that often more paths are possible.

When the strings representing the gestures are of different lengths, the method tends to favor the shorter strings. Say we have detected the string $\{B, C, D\}$ and want to classify it as being one of the two gestures: $\#1 = \{J, C, G\}$ and $\#2 = \{A, B, C, D, H\}$. The edit distance from the detected string to the gesture-strings will be two in both cases. However, it seems more likely that the correct interpretation is that the detected string comes from gesture #2 in a situation where the start and end has been corrupted by noise.

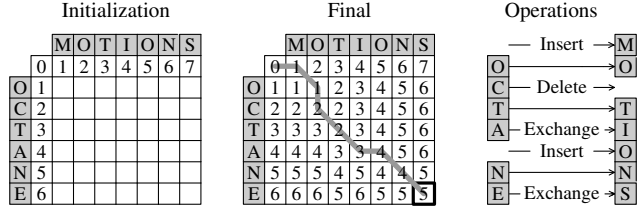


Figure 5. Measuring the distance between two strings using edit distance.

In fact, 2 out of 3 of the primitives have to be changed for gesture #1 whereas only 2 out of 5 have to be changed for gesture #2. We therefore normalize the edit distance by dividing the output by the length of the gesture-string, yielding 0.67 for gesture #1 and 0.2 for gesture #2, i.e., gesture #2 is recognized.

The edit distance is a deterministic method but by changing the cost of each of the three operations with respect to likelihoods it becomes a probabilistic method¹. Concretely we apply the weights described above, see equation 7. These to some extent represent the likelihood of a certain primitive being correct. The higher the weight the more likely a primitive will be. We incorporate the weights into the edit distance method by increasing the score by the weight multiplied by β (a scaling factor) whenever a primitive is *deleted* or *exchanged*. The cost of *inserting* remains 1.

The above principle works for situations where the input sequence only contains one gesture (possibly corrupted by noise). In a real scenario, however, we will have sequences which are potentially much longer than a gesture and which might contain more gestures after each other. The gesture recognition problem is therefore formulated as for each gesture to find the substring in the detected string, which has the minimum probabilistic edit distance. The recognized gesture will then be the one of the substrings with the minimum distance. Denoting the start point and length of the substring, s and l , respectively, we recognize the gesture present in the detected string as:

$$\text{Gesture} = \arg \min_{k,s,l} PED(\Lambda, k, s, l) \quad (8)$$

where k index the different gestures, Λ is the detected string, and $PED(\cdot)$ is the probabilistic edit distance.

5. Test and results

For testing purpose we use a vocabulary consisting of 11 primitives. This is illustrated in figure 6. The criteria for finding the primitives are 1) that they represent characteristic and representative 3D configurations, 2) that their

¹This is related to the Weighted Edit Distance method, which however has fixed weights.

configurations contain a certain amount of motion, and 3) that the primitives are used in the description of as many gestures as possible, i.e., fewer primitives are required. By use of this vocabulary of primitives we describe 4 one- and two-arms gestures: "Point right", "raise arm", "clap" and "wave".

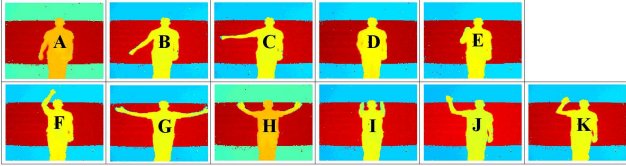


Figure 6. The vocabulary consisting of 11 primitives. The primitives are illustrated by range images of the arm configurations, which are color coded. The color can vary slightly due to error pixels and normalization.

We test the system on data recorded of 10 test subjects, each performing the four gestures 3 times from a 0° and 45° viewpoint with respect to the camera. A total of 240 video sequences have been recorded. Figure 7 shows an example of the visual differences that occur when a gesture is performed from these two viewpoints.

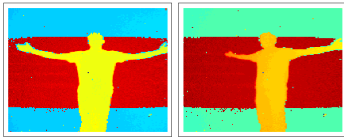


Figure 7. Range data examples of a time instance from a video sequence including a person carrying out a "clap" gesture shown from a 0° and 45° camera viewpoint.

To evaluate the view invariance of the system, the data which is used to train the motion primitives is only from the 0° viewpoint. The overall matching rate is 92.9% (94.9% without false negatives which the system have neglected because of a too high uncertainty. A total of 2.1% of the sequences are false negatives). The error distribution can be seen in the confusion matrix in figure 8. In comparison, when only testing on sequences from 0° we obtain a recognition rate of 95.8%.

	1	2	3	4
1. Point right	93.3	0.0	5.0	1.7
2. Raise arm	0.0	95.0	0.0	5.0
3. Clap	0.0	1.7	93.3	3.3
4. Wave	0.0	1.7	1.7	90.0

Figure 8. Test results (given in percentages) for the 4 gestures recorded from a 0° and 45° viewpoint with respect to the camera.

No significant increase in error can be observed when training and testing on sequences from different viewpoints,

i.e., the approach supports view invariant gesture recognition. The errors observed in both tests are mainly due to personal variations when performing gestures like "point right" and "raise arm". I.e., some tend to raise their arm above the shoulder while pointing while some do not stretch their arm fully when raising their arm. Another example is in the case of a "clap" gesture, where one of the arms might not be visible or segmented properly due to a too extreme viewpoint when the individual performs this gesture. Hence a "clap" gesture might be classified to be more likely another gesture. Furthermore, most of the false negatives originates from badly performed "wave" gestures at 45° . In some of these cases the test person turn more than 45° with respect to the camera. As a result most of the arm is hidden behind the body and therefore nearly invisible, hence only very little and poorly defined motion is present.

5.1. Neglecting intensity information

In comparison the same system without fusion of intensity and range information but solely based on range data yields the results given in figure 9.

	1	2	3	4
1. Point right	81.7	0.0	8.3	10.0
2. Raise arm	8.3	88.3	0.0	3.4
3. Clap	11.7	8.3	78.3	1.7
4. Wave	6.7	1.7	8.3	83.3

Figure 9. Test results when the system only rely on range information (given in percentages) for the 4 gestures recorded from a 0° and 45° viewpoint with respect to the camera.

The overall matching rate is reduced to 82.9%, and when only testing on sequences from 0° we obtain a recognition rate of 84.2%. This clearly states the benefit of using both range and intensity information. The increased error rate is mainly due to the fact, that the points included in the motion cloud are not perfectly located at the arms but stretches backwards as shown in figure 3. These miss-located points causes impact on the primitive descriptors, and hereby lead to errors.

5.2. Unknown start and end time

For each sequence we add "noise" in both the beginning and end of the sequence. By doing so, we introduce the realistic problem of having no clear idea about when a gesture commences and terminates which would be the case in a real situation. To achieve a test scenario that resembles this situation we split the gestures into halves and add one of these half gesture to the beginning and one to the end of each gesture to be processed by the system. The added half gestures are chosen randomly resulting in unknown start and end point of the real gesture.

	1	2	3	4
1. Point right	88.3	1.7	6.7	3.3
2. Raise arm	0.0	90.0	5.0	5.0
3. Clap	6.7	5.0	81.7	5.0
4. Wave	8.3	5.0	6.7	80.0

Figure 10. Test results when the start and end time for each gesture are unknown (given in percentages) for the 4 gestures recorded from a 0° and 45° viewpoint with respect to the camera.

Figure 10 shows the confusion matrix for the test results with unknown start and end time. The overall recognition rate for this test is 85.0% (85.4% without false negatives), and when only testing on sequences from 0° we obtain a recognition rate of 87.5%. The errors are the same as before but with some few additional errors caused by the unknown start and end time of the gestures. Especially, some "clap" and "wave" gestures seem to cause falsified classifications. Additionally, the quantity of false negatives is reduced to 0.4%, and instead replaced with false matches.

6. Conclusion

The contributions of this paper are twofold. Firstly, we show how the 3D data of a range camera can be stabilized by using the intensity image of the range camera to define a robust region of interest for the 3D data. This is valid especially in situations where the object of interest has considerably motion and steep 3D gradients. Secondly, we show how gesture recognition can be made view invariant through the use of 3D data and a spherical harmonic context representation. Furthermore, for the gesture recognition system we also address the problem of not knowing when a gesture commences and terminates. This is done by recognizing the gesture *not* through a trajectory based approach, but by representing a gesture as a sequence of discrete primitives. The presented approach is trained on gestures from 0° viewpoint and tested on gestures seen from both 0° and 45° viewpoints. The overall recognition rate is 92.9% with known start and end time of gestures, and 85.0% when the start and end time are unknown. These results state the robustness and view invariance of the system.

Acknowledgements

This work is partially funded by the MoPrim project (Danish National Research Councils - FTP) and partially by the HERMES project (FP6 IST-027110). The authors would like to thank Professor Thomas Bak, AAU, for providing access to the CSEM SwissRanger SR-2 Camera.

References

[1] S. Belongie, J. Malik, and J. Puzicha. Shape Matching and Object Recognition Using Shape Contexts. *Pattern Analysis*

and Machine Intelligence, 24(4), 2002. 3

[2] I. Cohen and H. Li. Inference of Human Postures by Classification of 3D Human Body Shape. In *Workshop on Analysis and Modeling of Faces and Gestures*, Nice, France, Oct. 2003. 1

[3] J. Gonzalez, J. Varona, F. Roca, and J. Villanueva. aSpaces: Action Spaces for Recognition and Synthesis of Human Actions. In *International Workshop on Articulated Motion and Deformable Objects*, Palma de Mallorca, Spain, November 21-23 2002. 1

[4] M. Haker, M. Bohme, T. Martinetz, and E. Barth. Geometric Invariants for Facial Feature Tracking with 3D TOF Cameras. In *International Symposium on Signals, Circuits and Systems*, Iasi, Romania, 13-14 July 2007. 1

[5] D. Hansen, R. Larsen, and F. Lauze. Improving Face Detection with TOF Cameras. In *International Symposium on Signals, Circuits and Systems*, Iasi, Romania, 13-14 July 2007. 1

[6] O. Jenkins and M. Mataric. Deriving Action and Behavior Primitives from Human Motion Data. In *International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, Sep 2002. 1

[7] Y. Kameda, M. Minoh, and K. Ikeda. Three Dimensional Motion Estimation of a Human Body Using a Difference Image Sequence. In *Asian Conference on Computer Vision*, Singapore, Dec. 1995. 2

[8] O. Khler, E. Rodner, and J. Denzler. On Fusion of Range and Intensity Information Using Graph-Cut for Planar Patch Segmentation. In *Dynamic 3D Imaging Workshop*, Heidelberg, Germany, September 11 2007. 1

[9] R. Larsen and B. Lading. Multiple Geodesic Distance Based Registration of Surfaces Applied to Facial Expression Data. In *International Symposium on Signals, Circuits and Systems*, Iasi, Romania, 13-14 July 2007. 1

[10] V. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Doklady Akademii Nauk SSSR*, 163(4), 1965. 5

[11] S. Mitra and T. Acharya. Gesture Recognition: A Survey. *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews*, 37(3):311–324, 2007. 1

[12] T. Moeslund, A. Hilton, and V. Krüger. A Survey of Advances in Vision-Based Human Motion Capture and Analysis. *Journal of Computer Vision and Image Understanding*, 104(2-3), 2006. 1

[13] T. Oggier, M. Stamm, M. Schweizer, and J. Pedersen. User manual swissranger 2 rev. b. Version 1.02, March 2005. 2, 3

[14] A. Prusak, I. Schiller, O. Melnychuk, R. Koch, and H. Roth. Pose Estimation and Map Building with a PMD-Camera for Robot Navigation. In *Dynamic 3D Imaging Workshop*, Heidelberg, Germany, September 11 2007. 1

[15] D. Weinland, R. Ronfard, and E. Boyer. Free Viewpoint Action Recognition using Motion History Volumes. *Computer Vision and Image Understanding*, 104(2), 2006. 1

[16] A. Yilmaz and M. Shah. Actions Sketch: A Novel Action Representation. In *Computer Vision and Pattern Recognition, vol. 1*, 2005. 1