Application of Kernel Principal Component Analysis for Gesture Recognition

Ryan Zoeller

Abstract—In this paper we examine the application of kernel principal component analysis to trajectory-based gesture recognition. Several gesture normalization methods are compared, and shown to be roughly equivalent in the non-degenerate case. The degenerate case is shown to be solvable using proportional scaling. Two gesture input methods are proposed, and the classification algorithm is shown to be agnostic to the data source, with potential applications in training gesture classification algorithms for use in foreign or unknown environments.

1 Introduction

In many control applications it is desirable to interact with a system without the use of tactile input such as buttons or switches, or without being in direct contact. These situations often arise in environments inherently hostile to traditional human-computer or human-robot interaction methods, such as underwater. The nature of these environments may rule out verbal communication mechanisms, necessitating a visual control scheme.

In these circumstances, gestures can provide a flexible and expressive vocabulary for conveying information. Gesture-based control has several advantages over other control mechanisms: no complex hardware beyond a visual camera is typically needed, a potentially unlimited number of gestures may be learned by a system, and communication through hand gestures is innate to humans. Unfortunately, the accurate classification of these gestures is a nontrivial problem.

A simple machine learning algorithm is proposed using kernel principal component analysis to separate the gestures, at which point a k-nearest neighbors classifier can accurately identify the input. Gestures are recorded as a series of time stamped points – a normalization process ensures that each gesture trajectory is scaled properly and interpolated to contain the same number of points. We consider several normalization methods, and analyze their performance across multiple data sets.

The gesture recognition algorithm is experimentally shown to be agnostic of the data source. Gesture trajectories are input via a mouse, and it is demonstrated that these trajectories are sufficient to correctly classify trajectories input through the use of a visual camera with blob tracking.

The previously ignored degenerate case of one-dimensional gestures is considered, and its tractability analyzed within the context of more normalization algorithms.

2 Related Work

The notion of using gestures to interface with a control system is not novel, especially in the context of human robot interaction. Typically, points from a gesture are sampled temporally (e.g. through a camera) to form a trajectory, and this trajectory is normalized before being compared with known trajectories.

Xu, Dudek, and Sattar propose augmenting an existing robotics control scheme with gesture recognition [3], in the domain of underwater robot control. An iterative closest point approach was proposed, which attempts to map sampled points from a given gesture to points in known gestures. The mapping process uses Euclidean distance to find the nearest point

in the other gesture, with a penalty for deviating temporally beyond a given tolerance. Trajectories are normalized along the axis of the principal eigenvector, as well as temporally to a unit time.

An alternate algorithm for performing gesture recognition is proposed by Ramírez-Giraldo et al., which uses kernel-based method to separate gestures for classification [1]. Kernel principal component analysis (KPCA) is applied to a trajectory sampled using a RGBD sensor, using a linear combination of Gaussian radial basis functions as the kernel. Trajectories are physically normalized to a unit height and width, as well as temporally to a unit time.

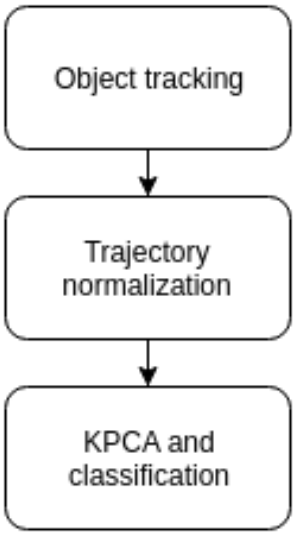


Fig. 1. Gesture recognition process

3 Methodology

3.1 Motivation

This work is largely motivated by shortcomings in existing gesture recognition methods which may cause some gestures to be ill-classified. The gesture recognition algorithms proposed by Xu, Dudek, and Sattar [3] and Ramírez-Giraldo et al. [1] both fail to classify degenerate cases properly: trajectories which are largely linear or otherwise one-dimensional lose their orientation information, which prevents the use of directed lines as gestures. The former reorients gestures such that their principal eigenvectors are aligned in a uniform direction – in the case of a line, all orientation information will be discarded as the eigenvector's direction follows that of the line. In the latter, the trajectory is normalized non-uniformly across the two observed physical dimensions – this effectively diagonalizes all lines which do not align perfectly to an axis.

The motivation for using the kernel principal component analysis approach suggested by Ramírez-Giraldo et al. [1] is its large number of other applications and free accessibility. KPCA's ease of use was demonstrated by Raschka [2] on miscellaneous data, and their work served as a concrete foundation to build the multi-kernel representation for classifying trajectories off of.

3.2 Gesture shape

The choice of gestures for an application is largely context dependent – in some systems, a large vocabulary may be required to accurately express all possible actions or objects. In others, a smaller vocabulary may be used in combination with some sort of gesture composition. In any case, the variation between unique gestures should be sufficiently large such that both the human actor and classification algorithm can reliably reproduce and identify the gesture.

Two sets of gestures were chosen to test the classification algorithm, with the goal of covering a broad range of potential use cases. The first set contains six capital letters chosen from the English alphabet – L, N, O, R, S, and W – which were chosen due to their lack of ambiguity. This set of symbols is a strict superset of that used by Ramírez-Giraldo et al. [1].

The second set of symbols was chosen to reflect the degenerate case in alternate gesture recognition algorithms: eight lines were used, one corresponding to each cardinal direction and the intermediate directions lying halfway in between them. In both implementations discussed in section 2, members of this set would not be properly classified in all cases.

3.3 Agnosticism to gesture source

The proposed gesture classification algorithm operates in several phases, as shown in Fig-

ure 1. Notably, the trajectory normalization and KPCA processes are not inherently aware of the source of the trajectory. As a result, it is possible to have multiple data input sources use the same underlying classification algorithm. Two gesture tracking frontends were created to test this principle: a cursor tracking application and a camera-based blob tracking application.

4 Trajectory normalization approach

4.1 Non-temporal interpolation strategy

With time based interpolation, such as that used by Ramírez-Giraldo et al. [1], temporal pauses or significant speed changes can significantly impact the resulting normalized trajectory. Consider a trajectory drawn perfectly (i.e. that matches training data exactly), except the operator pauses for a period of time at the start of the gesture. Under a time based interpolation method, every resampled point will be significantly shifted from the expected location, which may result in an incorrect classification.

There are several ways to mitigate this effect. When constructing training data, the speed and pauses in a gesture could be intentionally varied, with the goal of anticipating where an operator may pause or change speeds. Unfortunately, this requires some knowledge of the frontend environment – what is the maximum percentage of a gesture that an operator may spend (or appear to spend, due to tracking error) on any section of the gesture? It also increases the variance of the gesture, potentially reducing its separability with respect to other gestures.

Xu, Dudek, and Sattar [3] provide an alternate approach, which removes points that are clustered together and are matching with the same point in the training data. This completely removes the effect of pausing, but is not immediately applicable to a kernel-based approach.

A third option, which is tested, is to interpolate based on distance instead of time. That is, the total length of the trajectory is calculated with Equation 1, and points are sampled uniformly from this physical duration as opposed to the temporal duration. This removes the temporal aspect of the gesture entirely, which may be undesirable in some cases. It also fails in the case of high sampling error – although an operator may not be moving, it is possible that sampling noise will make the distance between identical points non-zero.

$$\text{Traj. Length} = \sum_{i=1}^{n-1} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} \tag{1}$$

4.2 Non-linear interpolation strategy

Linear interpolation is both incredibly fast and incredibly simple. These benefits come at the cost of sampling from a C^0 continuous function. The function produced by linear interpolation effectively can change velocity and acceleration instantaneously, unlike the physical system the trajectory represents.

To more closely model the physical system the trajectory is derived from, a cubic-spline based interpolation approach can be used. Here, samples are taken from a C^2 continuous function, which should more accurately reflect the underlying gesture, at the price of increased computation. The 'not-a-knot' end condition is used when interpolating.

As no elementary closed form exists in general for the length of a parametric function, a length based approach such as the one described in section 4.1 is not considered for non-linear interpolation.

4.3 Uniform scaling strategy

Both implementations discussed in section 2 use non-uniform physical scaling to normalize trajectories. This increases the similarity of gestures which are drawn slightly in the incorrect proportion, which is desirable, in general, for finding a match. Unfortunately, this also has the effect of causing gestures with wildly different proportions but a similar underlying shape to be stretched to the same representation.

For example, consider a long, thin rectangle and a square. Scaled proportionally, other rectangles and squares will match to the appropriate shape. When scaled non-uniformly, the

TABLE 1
Classification of mouse-derived trajectories against one third of mouse-derived samples

Normalization strategy	Average classification failures (maximum 480)	Success rate
Linear (time), non-uniform scaling	0.67	99.9%
Linear (time), uniform scaling	1.33	99.7%
Linear (distance), non-uniform scaling	0.67	99.9%
Linear (distance), uniform scaling	1.33	99.7%
Cubic spline, non-uniform scaling	1.33	99.7%
Cubic spline, uniform scaling	0.67	99.9%

rectangle will be compressed and stretched to a square.

A proportional normalization method is proposed that scales trajectories such that the axis with the largest range is scaled to uniform size. The other axis is scaled by the same factor.

5 Results

5.1 Non-degenerate classification

A total of 720 trajectories across the six gestures (L, N, O, R, S, W) were recorded by six subjects using the cursor tracking front end. The trajectories were normalized using a given trajectory normalization approach, and partitioned into buckets of 240 elements at random. A kernel principal component analysis was performed on one bucket, and the other buckets classified against the first. This process was repeated for each bucket and the number of failures averaged; the entire procedure was then repeated for each trajectory normalization approach. The results are shown in Table 1.

The high classification accuracy can be visually confirmed by examining the kernel similarity matrix of all 720 trajectories, as shown in Figure 2 for the distance-based linear interpolation strategy. It is observed that the diagonal has significantly higher intensity than other areas of the matrix. This indicates that each group of trajectories is very similar to itself, and does not resemble trajectories associated with other gestures.

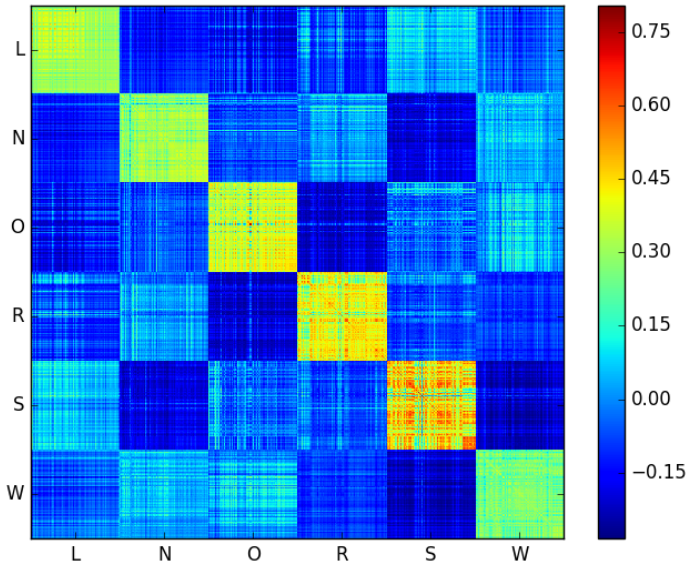


Fig. 2. Kernel similarity matrix under time invariant linear interpolation

TABLE 2
Classification of vision-derived trajectories against one third of mouse-derived samples

Normalization strategy	Average classification failures (maximum 120)	Success rate
Linear (time), non-uniform scaling	2.67	97.8%
Linear (time), uniform scaling	0.87	99.3%
Linear (distance), non-uniform scaling	0.00	100.0%
Linear (distance), uniform scaling	0.08	99.9%
Cubic spline, non-uniform scaling	2.00	98.3%
Cubic spline, uniform scaling	0.33	99.7%

A total of 120 trajectories were recorded using a vision-based blob tracker. The results of classifying this dataset against the 240 trajectory buckets described above can be seen in Table 2.

5.2 Degenerate classification

A total of 320 trajectories across the eight lines following the cardinal and intermediate directions were recorded by two subjects using the cursor tracking front end. The trajectories were

TABLE 3
Classification of mouse-derived degenerate trajectories against one half of mouse-derived samples

Normalization strategy	Average classification failures (maximum 120)	Success rate
Linear (time), non-uniform scaling	46.13	61.6%
Linear (time), uniform scaling	1.00	99.2%
Linear (distance), non-uniform scaling	52.63	56.1%
Linear (distance), uniform scaling	26.00	78.3%
Cubic spline, non-uniform scaling	46.50	61.3%
Cubic spline, uniform scaling	2.25	98.1%

normalized using a given trajectory normalization approach, and partitioned into buckets of 160 elements at random. A kernel principal component analysis was performed on one bucket, and the other bucket classified against the first. This process was repeated for both buckets and the number of failures averaged; the entire procedure was then repeated for each trajectory normalization approach. The results are shown in Table 3.

The kernel similarity matrices for the degenerate case explain why the success rates are so bad in the case of non-uniform scaling. When uniform scaling is applied, as shown in Figure 3, the diagonal remains well structured, as was the case with the non-degenerate gestures. When the gestures are scaled non-uniformly, diagonalization of the trajectories occurs, resulting in the similarity matrix shown in Figure 4. It is immediately clear that many lines on the cardinal directions have become aligned with the intermediate directions.

The first three principal components are enough to see the diagonalization phenomena. Figure 5 shows the the uniformly scaled projection; each gesture is clustered distinctly from all other gestures, and there are eight well defined groups. Figure 6 shows the the non-uniformly scaled projection; there are four primary groups, with significant noise in between the the clusters. Unlike in the uniformly scaled case, there is no easy way to partition this space into eight gestures.

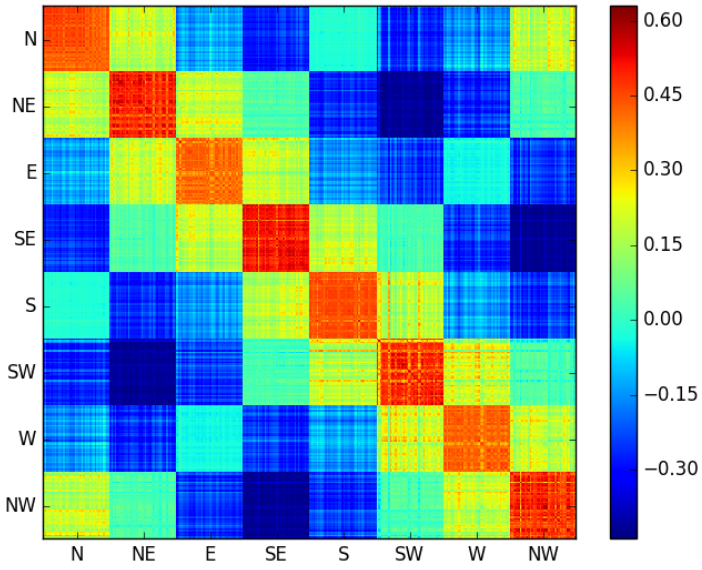


Fig. 3. Kernel similarity matrix under linear interpolation with uniform scaling

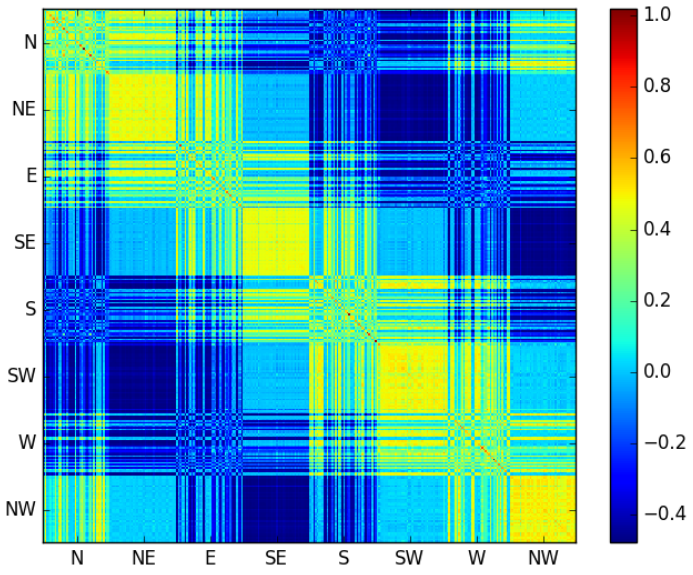


Fig. 4. Kernel similarity matrix under linear interpolation with non-uniform scaling

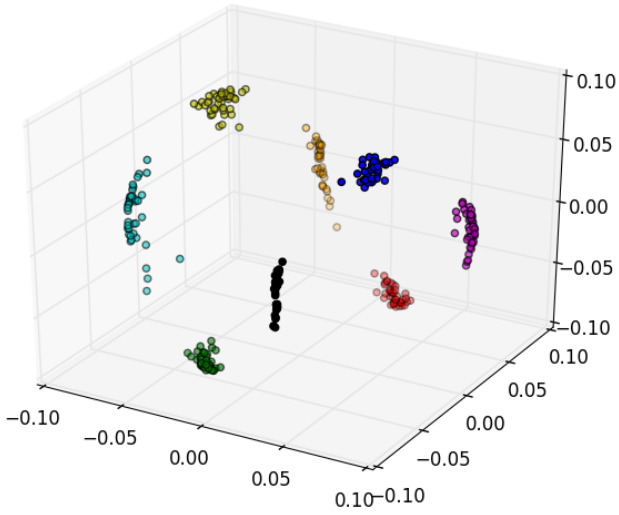


Fig. 5. First three principal components under linear interpolation with uniform scaling

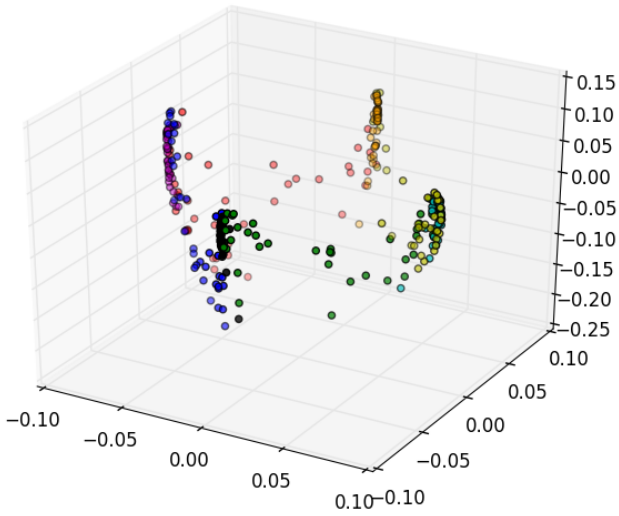


Fig. 6. First three principal components under linear interpolation with non-uniform scaling

6 Conclusion

6.1 Analysis

Using kernel principal component analysis is a viable approach for gesture recognition. However, there is no clear consensus on which trajectory normalization method is best; the normalization methods most accurate for classifying non-degenerate gestures are not the same as the normalization methods most accurate for classifying degenerate gestures.

In the non-degenerate case linear, time invariant interpolation performed extremely well, including when comparing to gestures tracked using an alternate mechanism. The pauses and speeds with which an operator can trace a gesture will vary significantly between media, which makes time a poor indicator to rely on when crossing between applications. However linear, time invariant interpolation performed poorly in the degenerate case, including when uniform scaling was used.

When attempting to classify degenerate trajectories (Table 3), the uniformly scaling time based linear interpolation and cubic spline interpolation methods performed extremely well.

The agnosticism of the underlying KPCA classifier to the trajectory source was largely confirmed. This has potential applications in creating training datasets – it is not necessary to duplicate the production environment in order to train the model. This significantly impacts non-traditional control applications, such as the underwater human-robot interaction work done by Xu, Dudek, and Sattar [3], as it may not be necessary to replicate either the robot or underwater environment to enable new functionality.

6.2 Future work

The iterative closest point method suggested by Xu, Dudek, and Sattar [3] handles the case of ill-defined trajectory bounds, by discarding points on the ends of trajectories that do not match to known models. Further investigation of this approach's relevance to a kernel based approach is desirable.

REFERENCES

[1] D. Ramírez-Giraldo, S. Molina-Giraldo, A. M. Álvarez Meza, G. Daza-Santacoloma, and G. Castellanos-Domínguez. Kernel based hand gesture recognition using kinect sensor. In *2012 XVII Symposium of Image, Signal Processing, and Artificial Vision (STSIVA)*, pages 158–161, Sept 2012.

[2] Sebastian Raschka, PCA Linear, RBF Gaussian, and Locally-Linear Embedding LLE. Kernel tricks and nonlinear dimensionality reduction via rbf kernel pca. 2014.

[3] A. Xu, G. Dudek, and J. Sattar. A natural gesture interface for operating robotic systems. In *2008 IEEE International Conference on Robotics and Automation*, pages 3557–3563, May 2008.

KERNEL BASED HAND GESTURE RECOGNITION USING KINECT SENSOR

Daniela Ramírez-Giraldo*, Santiago Molina-Giraldo*, Andrés M. Álvarez-Meza*
Genaro Daza-Santacoloma* †, and Germán Castellanos-Domínguez*

*Signal Processing and Recognition Group, Universidad Nacional de Colombia sede Manizales, Manizales, Colombia

†Instituto de Epilepsia y Parkinson del Eje Cafetero - Neurocentro, Pereira, Colombia

e-mail: daramirezgi, smolinag, amalvarezme, gdazas, cgcastellanosd {@unal.edu.co}

***Abstract*—Category 4. A machine learning based methodology is proposed to recognize a predefined set of hand gestures using depth images. For such purpose, a RGBD sensor (Microsoft kinect) is employed to track the hand position. Thus, a preprocessing stage is presented to subtract the region of interest from depth images. Moreover, a learning algorithm based on kernel methods is used to discover the relationships among samples, properly describing the studied gestures. Proposed methodology aims to obtain a representation space which allow us to identify the dynamic of hand movements. Attained results show how our approach presents a suitable performance for detecting different hand gestures. As future work, we are interested in recognize more complex human activities, in order to support the development of human-computer interface systems.**

***Keywords*— Depth sensor, human motion, kernel methods.**

I. INTRODUCTION

Interacting with machines and environments is a task of interest in computer vision systems. In fact, being able to detect human activities using computer vision techniques allow us to suitable built human-computer interfaces, which can be useful fields like medicine, sport training, entertainment, controlling process, robotics design, among others [1], [2], [3]. Nonetheless, even when some of the current computer vision systems have provided the ability to realize an interactive human body tracking, the challenge is to develop a low-cost system, reliable in unstructured home settings, and also straightforward to use.

The most common and ancient method of human communication have been gestures. In recent years, the gestures have also employed for interacting with machines or computer-assisted systems, instead of the traditional use of devices such as keyboard, mouse, joystick, etc. The human gesture interaction has several benefits such as free movements, no wired device limitations, free hands to use other important tools. In order to track human full-body pose in real-time, camera-based motion capture systems can be used that typically require a person to wear cumbersome markers or suits [4]. There exist several limitations in the past approaches. Garg [5] uses 3D images in his method to recognize the hand gesture, but this process was complicated and inefficient. The focus should be on efficiency with the accuracy as processing time is a very critical factor in real time applications. Yang [6] analysis the hand contour to select fingertip candidates, then finds peaks in their spatial distribution and checks local variance to locate fingertips. This method was not invariant to the orientation of the hand. Then, the human gestures recognition (particularly hand gestures) is still a challenging task due to the complexity (degrees of freedom) and unpredictability of human movements.

978-1-4673-2761-9/12/ $31.00

Recent advances have developed depth cameras that allow acquiring dense, three-dimensional scans of a scene in real-time, without the need for multi-camera systems. Such depth images are almost independent of lighting conditions and variations in visual appearance, e.g. due to clothing. In every image pixel, these cameras provide a measurement of the distance from the camera sensor to the closest object surface [4].

In this paper, we propose a machine learning based methodology to recognize a predefined set of hand gestures. For such purposes, we use a RGBD sensor (Microsoft Kinect) as the input sensor, and we present a learning algorithm based on kernel methods to discover the relationships among samples to infer the studied gestures. The goal of the proposed methodology is to obtain a representation space which allow us to identify properly the dynamic of the hand movements, which are captured by the Kinect. Attained results show how our approach presents an acceptable performance for detecting different hand gestures.

The remainder of this paper is organized as follows. In section II, proposed methodology for estimating hand position from depth images, and the kernel based framework used for recognizing hand gestures are described. In section III, the experimental conditions and the obtained results are shown. Finally, in sections IV and V, the discussion and conclusion are presented.

II. RECOGNIZING HAND GESTURES

A. Data Acquisition and Preprocessing

Kinect sensor has been widely used in computer vision tasks, due to the several advantages offered by the depth camera included in it [7]. The main advantages of depth sensors over traditional intensity ones are: enhance data representability by introducing a new characteristic (depth information), straightforward 3D reconstruction, capability of work in low light level scenes, and simplify the task of background subtraction. In order to take advantage of the kinect properties, a

preprocessing procedure is proposed to highlight the region of interest (hand) from depth images. In this regard, four different regions are extracted.

The former (gray region) is a dead zone configured by the user, in which the depth points are not taken into account. In the next region (yellow), the kinect sensor searches for the nearest depth point, in our case the hand. Note that, the yellow region does not contain depth data points. Then, in the green region, which is called the region of interest, the kinect establishes a working range, where it is expected to find the hand of the subject. Hence, as result an image containing only the data points that are presented in the green region are obtained. Finally, the last region (red) is also considered as a dead zone, where any object is captured by the sensor. Note that the length of the gray and the green regions can be fixed by the user. The above mentioned depth regions are summarized as in Fig. 1.

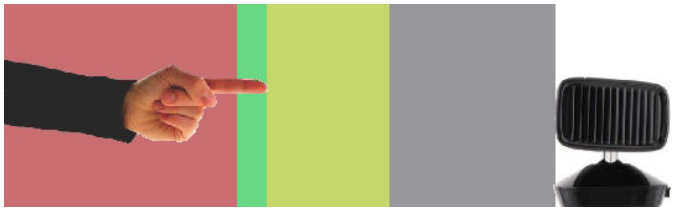


Fig. 1. Kinect sensor depth regions.

Given a depth image matrix $\mathbf{D} \in \Re^{h \times w \times 3}$, all the pixels that belong to the green region are fixed to the $\mathbf{g_r} \in \Re^{1\times 3}$ depth value. Therefore, the binary matrix $\mathbf{B} \in \Re^{h\times w}$ can be computed as in (1)

$$B_{ij} = \begin{cases} 1 & \|\mathbf{d}_{ij} - \mathbf{g}_r\| = 0 \\ 0 & \text{otherwise} \end{cases}, \tag{1}$$

where $\mathbf{d}_{ij} \in \Re^{1\times 3}$ is the depth intensity vector of the (i, j) pixel, with $i = 1, \ldots, h$ and $j = 1, \ldots, w$. Then, in order to identify the temporal dynamics of the hand movement, the centroid (i_c, j_c) of the detected object is estimated as

$$i_c = \frac{1}{N_{g_r}} \sum_{i=1}^{h}\sum_{j=1}^{w} iB_{ij}, \quad j_c = \frac{1}{N_{g_r}} \sum_{i=1}^{h}\sum_{j=1}^{w} jB_{ij}; \tag{2}$$

being N_{g_r} the number of elements in $\mathbf{B}$ that are equal to one. Regarding, let $n > 0$ the number of analyzed frames in a hand gesture, thus, the trajectory matrix $\mathbf{V} \in \Re^{n\times 2}$ is calculated with row vectors $\mathbf{v}_t = [i_c^t, j_c^t]$, being (i_c^t, j_c^t) the centroid of the detected object in frame t, and with $t = 1, \ldots, n$.

Furthermore, a conventional lineal interpolation method is used to properly compare hand gesture trajectories with different sizes. Then, the matrix $\mathbf{S} \in \Re^{T\times 2}$ is obtained from interpolating the columns of $\mathbf{V}$, being $T > 0$ the fixed time trajectory size. Finally, a dynamic range normalization is used over each column of $\mathbf{S}$ to achieve consistency for comparing different trajectories. Therefore, the matrix $\mathbf{X} \in \Re^{T\times 2}$ is estimated as

$$X_{l1} = \frac{2\left(S_{l1} - \bar{s_1}\right)}{\max\left(\mathbf{s}_1\right) - \min\left(\mathbf{s}_1\right)}, \quad S_{l2} = \frac{2\left(S_{l1} - \bar{s_2}\right)}{\max\left(\mathbf{s}_2\right) - \min\left(\mathbf{s}_2\right)} \tag{3}$$

with $l = 1, \ldots, T$, and being $\mathbf{s}_1$ and $\mathbf{s}_2$ the first and second column of $\mathbf{S}$, respectively. Fig. 2 shows the proposed acquisition and preprocessing framework for predicting hand gestures trajectories from depth images.

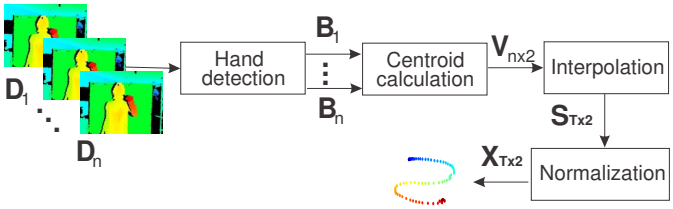


Fig. 2. Data acquisition and preprocessing scheme.

B. Gesture recognition based on Kernel Representation

The use of kernel functions to infer relationships among samples have been widely used for machine learning procedures [8]. Here, we propose to use a kernel representation to unfold the hand gesture trajectories similarities. Recently, some machine learning approaches have shown that using multiple kernels to infer the data similarities instead of just one, can be useful to improve the data interpretability [9]. Given a pair of hand trajectory matrices $\mathbf{X}^p$ and $\mathbf{X}^p$, and assuming Z kernel functions, the multiple kernel representations - MKR based methods aim to infer the combined kernel function $\kappa_\xi\left(\mathbf{X}^p, \mathbf{X}^q\right) = \sum_{z=1}^{Z} \xi_z \kappa_z\left(\mathbf{X}^p, \mathbf{X}^q\right)$, subject to $\xi_z \geq 0$, and $\sum_{i=1}^{Z} \xi_z = 1$ $(\forall \xi_z \in \Re)$. Thereby, the input data is analyzed from different information sources by means of a convex combination of basis kernels.

Using the above described MKR framework, we propose to combine two different kernels, κ_a and κ_o, to estimate the abscissa and ordinate similarities among hand trajectories. Hence, a combined kernel function is computed as

$$\kappa\left(\mathbf{X}^p, \mathbf{X}^q\right) = \xi_a \kappa_a(\mathbf{x}_a^p, \mathbf{x}_a^q) + \xi_o \kappa_o(\mathbf{x}_o^p, \mathbf{x}_o^q), \tag{4}$$

where the vectors $\mathbf{x}_a^p$ and $\mathbf{x}_a^q$ correspond to first column of $\mathbf{X}^p$ and $\mathbf{X}^q$, respectively, and $\mathbf{x}_o^p$ and $\mathbf{x}_o^q$ contain the second ones. Moreover, $p, q = 1, \ldots, N$, being N the number of given trajectories. Thus, the kernel matrix $\mathbf{K} \in \Re^{N\times N}$ can be estimated by (4).

III. Experimental Set-up and Results

To test the performance of the proposed methodology to characterize time-series data, a hand gesture recognition database was recorded using the kinect sensor. We employ the kinect camera which gives a 640×480 image at 30 frames per second, using a depth resolution of $3[mm]$. The database contains 3 different hand gesture symbols performed by 2 subjects. The chosen symbols are the letters O,S and L, and each subject performs each symbol 10 times.

Data is extracted using the libfreenect software provided by OpenKinect [1], and the OpenCV C++ library is used for the image processing operations [2]. The data acquisition is made by using the region scheme explained in section II-A. The gray zone is set to approximately $1[m]$ (suggested distance by Microsoft). The length of the green region is small enough fixed for obtaining more accurate results, approximately $1[cm]$. The centroid of this region is determined by obtaining the mean of the row and column coordinates of the segmented data points by using equation 2. Moreover, to remove outlier data, we used a median filter over the abscissa and ordinate signals (each column of $\mathbf{V}$), with a fixed window of 12 samples. Each signal is scaled and interpolated with $T = 80$ (see section II-A). For each symbol recording, n frames are taken according to each user symbol length. Fig. 3 shows a segmented image using the proposed acquisition and preprocessing framework, and Fig. 4 presents some preprocessed hand gesture trajectories.

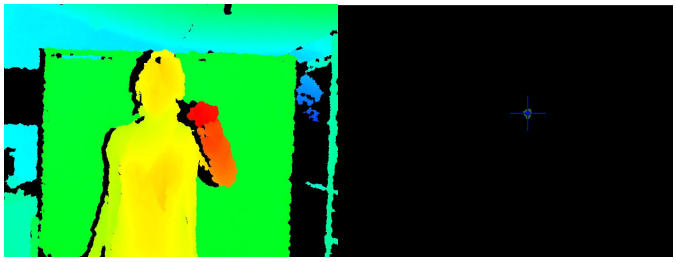


Fig. 3. Hand trajectory prediction example.

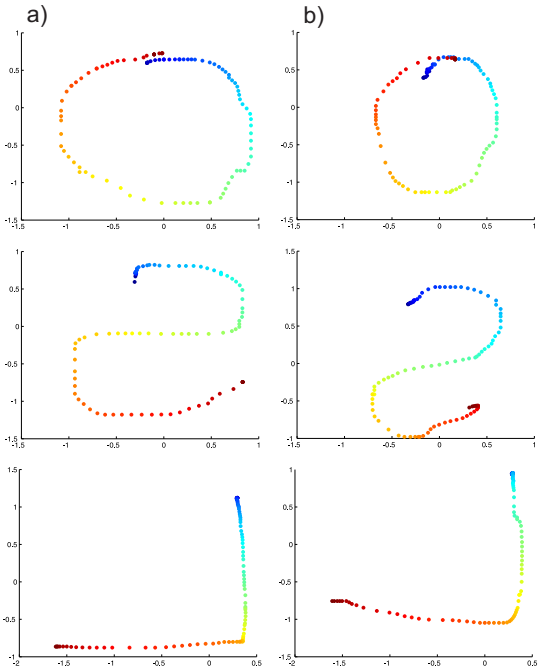


Fig. 4. Some preprocessed hand trajectories. a) Subject one. b) Subject two.

The MKR scheme explained in section II-B is used to represent, as well as possible, the obtained information. A gaussian kernel is used as basis to estimate the relationships among hand trajectories in (4). For concrete testing, the kernel band-width σ is empirically fixed to 3. Besides, ξ_a and ξ_a are set to 0.5 in (4). The resulting kernel matrix $\mathbf{K}$ of the studied dataset can be seen in Fig. 5

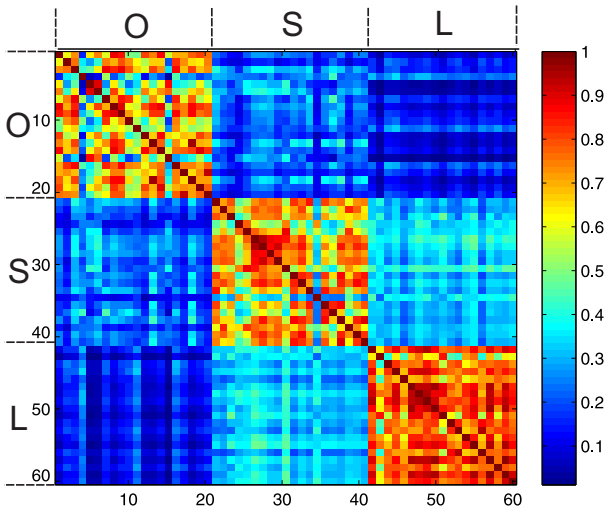


Fig. 5. Gaussian Kernel Matrix.

After that, a Kernel-Principal Component Analysis - KPCA is applied over $\mathbf{K}$ [8], obtaining a low-dimensional feature space $\mathbf{E} \in \Re^{60\times 3}$. Finally, a k-nearest neighbors classifier - knnc is trained over the low-dimensional space. It is important to note, that the system performance is tested using a 10-folds cross validation scheme. In Fig.6 a 3D representation of the studied data is presented.

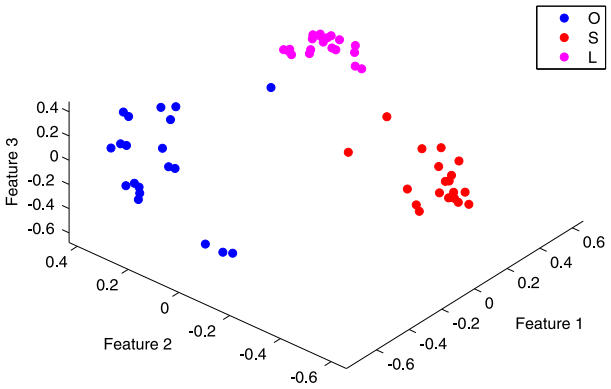


Fig. 6. Low-dimensional KPCA projection - knnc test accuracy = 100[%].

IV. Discussion

According to the preprocessing results show in Fig. 4, it is possible to notice the capability of our approach for characterize the hand trajectory. Due to the region based methodology for inferring hand position, the estimate trajectory is smooth enough for further characterization stages.

On the other hand, the resulting similarity measure obtain by MKR using a gaussian kernel (Fig. 5) confirms that the similarity among signals from the same class is very high, with a mean similarity of 0.69 (orange color). Again, the class that

[1] http://openkinect.org
[2] http://opencv.willowgarage.com/wiki/

exposes the highest intra-similarity corresponds to the symbol L with a mean similarity of 0.78. The classes more similar between them are the S and the L, exposing a mean similarity of 0.32.

The above given measures properties are corroborated by the estimated KPCA low-dimensional projection presented in Fig. 6. It can be notice how the MKR framework facilitates in a major way the classification process. The resulting feature space exhibits an appropriate separation among different classes. It is also noted that the L symbol (third class) shows the highest intra-similarity among all the signals.

V. Conclusions

A machine learning based methodology for recognizing hand gestures using depth images captured by a kinect sensor was proposed. In this sense, a region based acquisition scheme using depth images was employed in order to obtain an accurate segmentation of the region of interest. Moreover, a MKR framework was proposed to combine into a single similarity matrix, the abscissa and ordinate features inferred from the centroid trajectories of hand gestures. Attained results showed that the proposed acquisition methodology obtains very accurate data points, properly identifying the dynamic of the gesture. Furthermore, the proposed MKR framework enhances the separability of the classes, facilitating further classification process. As future work, it should be interesting to include more hand gesture symbols, and also it will be useful to apply a similar MKR approach for skeleton tracking using depth images.

Acknowledgments

This research was carried out under grants provided by a Msc. and a PhD. scholarships, and the project "ANÁLISIS DE MOVIMIENTO EN SISTEMAS DE VISIÓN POR COMPUTADOR UTILIZANDO APRENDIZAJE DE MÁQUINA", funded by Universidad Nacional de Colombia.

References

[1] R. Urtasun and T. Darrell, "Sparse probabilistic regression for activity-independent human pose inference," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[2] T. Jaegglia, E. Koller-Meier, and L. Gool, "Learning generative models for multi-activity body pose estimation," *Int. J. Comput. Vis*, vol. 82, pp. 121–134, 2009.

[3] R. Kehl and L. Gool, "Markerless tracking of complex human motions from multiple views," *Comput. Vis. Image Underst.*, vol. 104, pp. 190–209, 2006.

[4] L. A. Schwarz, A. Mkhitaryan, D. Mateus, and N. Navab, "Human skeleton tracking from depth data using geodesic distances and optical flow," *Image and Vision Computing*, vol. 20, no. 1, pp. 217–226, 2012.

[5] P. Garg, N. Aggarwal, and S. Sofat, "Vision based hand gesture recognition," *World Academy of Science, Engineering and Technology*, vol. 49, pp. 972–977, 2009.

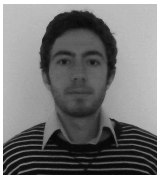
[6] D. Yang, L. Jin, J. Yin *et al.*, "An effective robust fingertip detection method for finger writing character recognition system," in *Proceedings of the Fourth International Conference On Machine Learning And Cybernetics*, 2005, pp. 4191–4196.

[7] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *CVPR*, vol. 2, 2011, p. 7.

[8] B. Scholkopg and A. J. Smola, *Learning with Kernels*. Cambridge, MA, USA: The MIT Press, 2002.

[9] A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet, "SimpleMKL," *Journal of Machine Learning Research*, vol. 9, pp. 2491–2521, 2008.

Daniela Ramírez-Giraldo student of electronic engineering from the Universidad Nacional de Colombia sede Manizales. Her research interests are image and video processing using kinect sensor.



Santiago Molina-Giraldo received his undergraduate degree in electronic engineering from the Universidad Nacional de Colombia sede Manizales in 2012. Currently, he is pursuing a M.Sc at the same university. His research interests are nonlinear dimensionality reduction and kernel methods for motion analysis and signal processing.

Andres Marino Alvarez-Mesa received his undergraduate degree in electronic engineering with honors, and his M.Sc. engineering-industrial automation with honors from the Universidad Nacional de Colombia sede Manizales, in 2009 and 2012. Currently, he is pursuing a Ph.D at the same university. His research interests are nonlinear dimensionality reduction and kernel methods for signal processing.

Genaro Daza-Santacoloma received his undergraduate degree in electronic engineering in 2005, the M.Sc. degree in engineering-industrial automation with honors in 2007, and the Ph.D. degree in engineering-automatics with honors in 2010, from the Universidad Nacional de Colombia sede Manizales. Currently, he is an engineering researcher at the Instituto de Epilepsia y Parkinson del Eje Cafetero - Neurocentro, Pereira - Colombia. His research interests are feature extraction/selection and motion analysis for training pattern recognition systems.

German Castellanos-Dominguez received his undergraduate degree in radiotechnical systems and his Ph.D. in processing devices and systems from the Moscow Technical University of Communications and Informatics, in 1985 and 1990 respectively. Currently, he is a professor in the Department of Electrical, Electronic and Computer Engineering at the Universidad Nacional de Colombia at Manizales. He is Chairman of the GCPDS at the same university. His research interests include information and signal theory, digital signal processing and bioengineering.

A Natural Gesture Interface for Operating Robotic Systems

Anqi Xu, Gregory Dudek and Junaed Sattar

***Abstract*— A gesture-based interaction framework is presented for controlling mobile robots. This natural interaction paradigm has few physical requirements, and thus can be deployed in many restrictive and challenging environments. We present an implementation of this scheme in the control of an underwater robot by an on-site human operator. The operator performs discrete gestures using engineered visual targets, which are interpreted by the robot as parametrized actionable commands. By combining the symbolic alphabets resulting from several visual cues, a large vocabulary of statements can be produced. An Iterative Closest Point algorithm is used to detect these observed motions, by comparing them with an established database of gestures. Finally, we present quantitative data collected from human participants indicating accuracy and performance of our proposed scheme.**

I. Introduction

Gestures are one of the most expressive ways of communicating between people. Whether they are initiated using hands, facial features, or the entire body, the benefits of using gestures in comparison with other media such as speech or writing comes from the vast amount of information that can be associated with a simple shape or motion. In this paper we present an approach for adapting gestures as a communication scheme in the Human-Robot Interaction (HRI) context. More specifically, our work deals with robot control in the underwater domain, where available modes of communication are highly constrained due to the restrictions imposed by the water medium. This paper describes a framework for controlling an amphibious legged robot, by tracing out trajectories using bar-code-like markers.

We are particularly interested in the application where an underwater scuba diver is assisted by a semi-autonomous robotic vehicle. This setup can be thought of as the human-robot counterpart of a broader communication problem. In general, divers converse with each other using hand signals as opposed to speech or writing. This is because the aquatic environment does not allow for simple and reliable acoustic and radio communication, and because the physical and cognitive burdens of writing or using other similar communication media are generally undesirable. On the other hand, visual gestures do not rely on complicated or exotic hardware, do not require strict environmental settings, and can convey a wide range of information with minimal physical and mental effort from the user. Furthermore, by combining spatial gestures with other visual communication modes, a large and expressive vocabulary can be obtained.

Centre for Intelligent Machines, McGill University, 3480 University Street, Montreal, Quebec, Canada H3A 2A7. {anqixu,dudek,junaed}@cim.mcgill.ca

```
for (i = 0; i < 4; i++) {          4 REPEAT
   angle = 90;                        9 0 ANGLE
   duration = 2;                      2 DURATION
   Turn_Left(angle,                   TURN_LEFT
   duration);                         MOVE_FORWARD
   Move_Forward(duration);         END
}                                  EXECUTE
```

Fig. 1. Comparison of C (left) and RoboChat (right) syntax.

While our approach is motivated by underwater robotics, the methods we employ can be used in other human-robot interaction (HRI) contexts as well. Conventional approaches of robot interaction rely on keyboards, joysticks and spoken dialog. These traditional methods can be problematic in many contexts, such as when speech and radio signals cannot be used (i.e. underwater). The approach presented in this paper extends prior work using an interface called RoboChat [5]. Using RoboChat, an underwater diver displays a sequence of symbolic patterns to the robot, and uses the symbol sequence to generate utterances using a specialized language (Fig. 1), which includes both terse imperative actions commands, as well as complex procedural statements. The RoboChat language also features syntactic structures that serve to minimize user input, as well as to increase the flexibility of the language. It is designed to employ any system of fiducial markers to permit robust target detection. The present implementation uses the ARTag marker set [7], although we are transitioning to an alternative deployment based on Fourier Tags [12].

In spite of its utility, RoboChat suffers from three critical weaknesses in its user interface. First of all, because a separate fiducial marker is required for each robot instruction, the number of markers associated with robot commands may be significantly large for a sophisticated robotic system. This requirement can impede the diver's locomotive capabilities, since he must ensure the secure transportation of this large amount of marker cards underwater. Secondly, the mapping between robot instructions and symbolic markers are completely arbitrary, as the diver must first read the labels on each card to locate a particular token. Thirdly, as a consequence of the previous two deficiencies, the diver may require a significant amount of time to locate the desired markers to formulate a syntactically correct script, which may be unacceptable for controlling a robot in real-time.

This paper proposes an interaction paradigm called RoboChat Gestures, which can be used as a supplementary input scheme for RoboChat. It is designed specifically to remedy all three aforementioned weaknesses in the core interface. The main premise is for the diver to formulate discrete motions using a pair of fiducial markers. By interpreting

different motions as robot commands, the diver no longer is required to carry one marker per instruction. The trajectories of RoboChat Gestures are derived from different types of traditional gestures, to take advantage of existing associations and conventions in the form of embedded information. This introduces a natural relationship between trajectories and their meanings, which alleviates the cognitive strain on the user. Additionally, the robot can process the observed gestures and extract features from the motion, such as its shape, orientation, or its size. Each gesture is mapped to a command, while the extracted features are associated with various parameters for that instruction. Because much of the information is now embedded in each trajectory, RoboChat Gestures can express the same amount of information that the previous RoboChat interface could, but in significantly less time, and using only two fiducial markers.

The rest of the paper is organized as follows. Section II presents a brief literature survey. Sections III and IV elaborate on the concept of RoboChat Gestures, and in particular explains the inner workings of the gesture detection process. Implementation results of the proposed scheme is discussed in Section V, both quantitatively and qualitatively. We conclude the paper in Section VI and present possible avenues for future work.

II. Related Work

Our work described in this paper is based on four principal ideas: a navigating underwater robot, the use of robust visual targets, gesture recognition in the abstract, and gestures for robot control.

Sattar et al. looked at using visual communications, and specifically visual servo-control with respect to a human operator, to handle the navigation of an underwater robot [13]. In that work, while the robot follows a diver to maneuver, the diver can only modulate the robot's activities by making hand signals that are interpreted by a human operator on the surface. Visual communication has also been used by several authors to allow communication between robots on land, or between robots and intelligent modules on the sea floor, for example in the work of Vasilescu and Rus [16].

The work of Waldherr, Romero and Thrun [17] exemplifies the explicit communication paradigm in which hand gestures are used to interact with a robot and lead it through an environment. Tsotsos et. al [15] considered a gestural interface for non-expert users, in particular disabled children, based on a combination of stereo vision and keyboard-like input. As an example of implicit communication, Rybski and Voyles [11] developed a system whereby a robot could observe a human performing a task and learn about the environment.

Fiducial marker systems, as mentioned in the previous section, are efficiently and robustly detectable under difficult conditions. Apart from the ARTag toolkit mentioned previously, other fiducial marker systems have been developed for use in a variety of applications. The ARToolkit marker system [10] consists of symbols very similar to the ARTag flavor in that they contain different patterns enclosed within a square black border. Circular markers are also possible in fiducial schemes, as demonstrated by the Photomodeler Coded Targets Module system [1] and the Fourier Tags [12].

Vision-based gesture recognition has long been considered for a variety of tasks, and has proven to be a challenging problem examined for over 20 years with diverse well-established applications [6] [9]. The types of gestural vocabularies range from extremely simple actions, like simple fist versus open hand, to very complex languages, such as the American Sign Language (ASL). ASL allows for the expression of substantial affect and individual variation, making it exceedingly difficult to deal with in its complete form. For example, Tsotsos et al. [3] considered the interpretation of elementary ASL primitives (i.e simple component motions) and achieved 86 to 97 *per cent* recognition rates under controlled conditions.

Gesture-based robot control is an extensively explored topic in HRI. This includes explicit as well as implicit communication frameworks between human operators and robotics systems. Several authors have considered specialized gestural behaviors [8] or strokes on a touch screen to control basic robot navigation. Skubic *et al.* have examined the combination of several types of human interface components, with special emphasis on speech, to express spatial relationships and spatial navigation tasks [14].

III. Methodology

A. Motivation and Setup

RoboChat Gestures is motivated partly by traditional hand signals used by all human scuba divers to communicate with one another. As mentioned in Sec. I, the original RobotChat scheme was developed as an automated input interface to preclude the need for a human interpreter or a remote video link. Usability studies of RoboChat suggests that naive subjects were able to formulate hand signals faster than searching through printed markers. This difference was apparent even when the markers were organized into indexed flip books to enhance rapid deployment. We believe that this discrepancy in performance was due to the intuitive relationships that existed between the hand signals and the commands they represented. These natural relationships served as useful mnemonics, which allowed the diver to quickly generate the input without actively considering each individual step in performing the gesture.

The RoboChat Gestures scheme employs the same technique as hand signals to increase its performance. Each gesture comprises of a sequence of motions performed using two fiducial markers, whose trajectory and shape imply a relevant action known to be associated with this gesture. Because different instructions can now be specified using the same pair of markers, the total number of visual targets required to express the RoboChat vocabulary is reduced considerably, making the system much more portable. This benefit is particularly awarding to scuba divers, who already have to attend to many instruments attached to their dive gear. In general, the expression space for RoboChat Gestures comprises of several dimensions. Different features may be used in the identification process, including the markers' ID,

the shape of the trajectory drawn, its size, its orientation, and the time taken to trace out the gesture. In addition, the gestures provide a way to communicate out-of-band signals, for example to stop the robot in case of an emergency. To optimize the system's usability, numerical values for these non-deterministic features are converted from a continuous representation to a discrete one, for both signal types.

B. Gesture design criteria

The selection of gestures for our system depends highly on the target application. Designing shapes and motions suitable for an aquatic robot comes with a number of restrictions. Firstly, in the water medium, both the diver and the robot are in constant motion, which makes performing and detecting gestures more complex compared to the terrestrial domain. To address this issue, we use two fiducial markers to perform gestures, by using one marker as a reference point or "origin" in the image space, and using the other "free" marker to draw the actual gesture shapes. This approach compensates for the constant motion of the vehicle and the operator, but also reduces the effective field of view of the camera. This problem can also be addressed by increasing the distance between the operator and the camera. With our current implementation with ARTags, successful detection is possible with a separation of up to 2 meters. Also, since the marker detection scheme is impeded by motion blur, we impose on the operator the requirement to pause briefly at the vertices of the gestures. The time span of the pause is usually very small, resulting directly from the robustness of the fiducial detection scheme.

IV. RoboChat Gestures Detection Algorithm

A. Overview

Our gesture recognition system exploits the positions of the visual targets on the image plane over time. Thus the raw input data to the system is a series of points of the form (x, y, t). We use an Iterative Closest Point (ICP) algorithm [2] to determine whether a given point cloud represents a known gesture. Traditional ICP methods match 3-D points independent of their ordering, typically using either a Euclidean or Mahalanobis distance metric. In our case, we augment the ICP distance metric to use the position of the gesture points on the 2-D image plane, as well as the temporal sequence (but not the speed) associated with the gesture. This algorithm attempts to pair up an observation point cloud to different reference clouds, each representing a unique gesture.

The ICP algorithm has two simple steps. First, for each of the points in the observation cloud, we identify the closest point in the reference set. Each point pair returns a distance, which is stored into an error metric vector. Then, we find the optimal method of transforming the observation cloud, to minimize the least square error for the previously obtained vector. Afterwards, we apply the transformation and iterate these two steps until the improvement in the algorithm falls below a certain threshold. When this terminating criterion is reached, we evaluate the final error metric vector, and use it to determine whether the observation accurately resembles the selected reference.

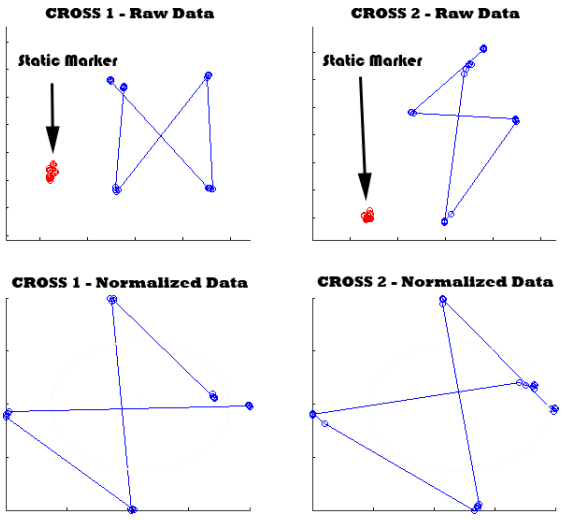


Fig. 2. Raw and pre-processed data for two RoboChat Gestures clouds.

B. Pre-processing

To be able to properly compare point clouds, we need to ensure that the data are on a similar scale. First, we identify the position of the static marker as the origin, by looking for the point sequence with the smallest covariance in the 2-D positional space. We generate the data cloud by centering the other marker about this (time-dependent) origin. To detect rotated shapes, we first obtain the principal eigenvector for each cloud, and rotate the data so that this vector is aligned in every cloud. Additionally, to be able to match gestures with different shapes, we unit-normalize the positional values on the principal eigenvector axis, as well as on its perpendicular axis. This last operation generally does not constrain proportions, which is not an issue if we assume that only non-degenerate 2-D shapes are allowed (*i.e.* no lines). Finally, we unit-normalize the time axis as well, to allow for gestures at different speeds to be compared. We perform these three steps to ensure that similar shapes are already somewhat aligned with each other prior to the detection phase, as shown in Fig. 2. Additionally, it minimizes the number of iterations required by the ICP algorithm, and also minimizes the chance for the optimization part of the algorithm to be trapped by a local minimum.

In order to increase detection rates, we compare the observation cloud against different variants of each reference cloud. We generate these variants by rotating the data by 180' in the positional plane, by inverting points about the principal eigenvector axis, by inverting the time axis, and by permutations of these three transformations. These transformations allow for detection of mirrored shapes, and also cancels out the sign of the eigenvector, which may be different even for similar clouds.

C. Point-to-point matching step

We first obtain the distance vectors between an observation point and all data in the reference cloud. We then compute

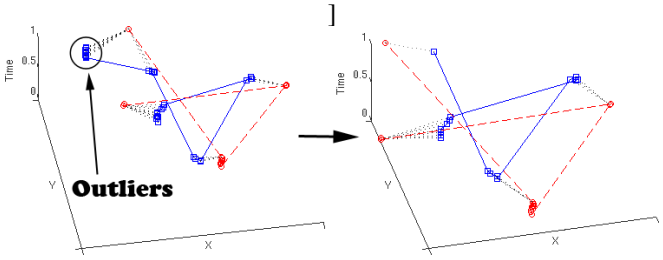


Fig. 3. Effect of trimming the point cloud.

the magnitude array using the Euclidean distance formula. Next, we identify point pairs whose temporal components surpass a certain absolute value, and penalize their corresponding error magnitude by manually adding to it a fixed value. This way, when searching for the minimum magnitude, we select the closest point pair from those with tolerable temporal distances, if such pairs are available. After pairing up each point in the observation cloud with one in the reference, we assemble all the distances into the error metric vector.

Since markers can be detected when the user is bringing them into their starting positions, and also when they are being removed after a gesture has been completed, this can introduce "terminal" outliers. For this reason, we provide the option to trim the observation cloud following the pairing process. If the first few observation points all match to a single reference point, we discard all but the last point. The same operation is also performed on the last few observation points as well. We then stretch the temporal values for the resulting cloud to match the range of the initial set. As shown in Fig. 3, this process can eliminate outliers at both ends of the data.

D. Cloud optimization step

In the subsequent step, we use the error metric vector to solve for an optimal transformation that minimizes the squared distance of this new error metric vector. We introduce two different types of transformations: in the first variant, the algorithm minimizes the point cloud by allowing it to rotate about the positional plane, and to translate in all 3 dimensions. The second variant also allows for 3-dimensional translation, but it employs proportional scaling in the positional plane instead of rotation. These two variants are either linear or can be linearized, and thus both have closed-form solutions to their optimization rules.

The solution for the rotational variant is not exact, because we approximate the cosine and sine of the angle of rotation by 1 and the angle, respectively. As a precaution, we always verify the fidelity of this approximation to ensure that the solution is still qualitatively consistent.

We will now outline the derivation for this variant's solution. Given each point p in the observation (with N total number of points) and each point q in the reference, we attempt to minimize the error magnitude E by computing the rotational matrix R with angle θ and the translational vector T.

$$E = \Sigma_{\forall p}(Rp + T - q)^2 \cdot [1; 1; 1]$$

The rotation matrix R is approximated as follows:

$$R = \begin{vmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{vmatrix} \simeq \begin{vmatrix} 1 & -\theta & 0 \\ \theta & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

After expanding E, taking its derivatives with respect to θ, and equating to zero,

$$\Sigma(p_x^2 + p_y^2)\theta - \Sigma(p_y)T_x + \Sigma(p_x)T_y = \Sigma(p_x q_y - p_y q_x)$$

Similarly, taking derivatives of E with respect to T_x, T_y and T_t, and equating to zero as before, we have:

$$\begin{aligned} -\Sigma(p_y)\theta + NT_x &= \Sigma(q_x) - \Sigma(p_x) \\ \Sigma(p_x)\theta + NT_y &= \Sigma(q_y) - \Sigma(p_y) \\ NT_t &= \Sigma(q_t) - \Sigma(p_t) \end{aligned}$$

Solving the above equations for the unknowns, we have:

$$\begin{aligned} \theta &= \frac{[N\Sigma(p_x q_y) - N\Sigma(p_y q_x)] - \Sigma(p_x)\Sigma(q_y) + \Sigma(p_y)\Sigma(q_x)}{[N\Sigma(p_x^2) + N\Sigma(p_y^2) - \Sigma(p_x)^2 - \Sigma(p_y)^2]} \\ T_x &= \frac{\Sigma(p_y)\theta + \Sigma(q_x) - \Sigma(p_x)}{N} \\ T_y &= \frac{-\Sigma(p_x)\theta + \Sigma(q_y) - \Sigma(p_y)}{N} \\ T_t &= \frac{\Sigma(q_t) - \Sigma(p_t)}{N} \end{aligned}$$

The scaling variant, on the other hand, produces a linear optimization rule and thus returns an exact solution. We provide a similar outline to obtain a, the scale factor, and T, the translational vector, by minimizing E:

$$E = \Sigma_{\forall p}([a; a; 1]p + T - q)^2 \cdot [1; 1; 1]$$

We solve a system of equations, similar to the one above, for a and T:

$$\begin{aligned} a &= \frac{[N\Sigma(p_x q_x) + N\Sigma(p_y q_y) - \Sigma(p_x)\Sigma(q_x) - \Sigma(p_y)\Sigma(q_y)]}{[N\Sigma(p_x^2) + N\Sigma(p_y^2) - \Sigma(p_x)^2 - \Sigma(p_y^2)]} \\ T_x &= \frac{-\Sigma(p_x)a + \Sigma(q_x)}{N} \\ T_y &= \frac{-\Sigma(p_y)a + \Sigma(q_y)}{N} \\ T_t &= \frac{\Sigma(q_t) - \Sigma(p_t)}{N} \end{aligned}$$

E. Algorithmic flow

After pre-processing the observation cloud, we first set the optimization type to translation and rotation. Since the data has just been scaled in the pre-processing stage, naturally this variant produces a better result than the scaling version. The algorithm iterates until the difference in overall normalized error magnitude between two successive iterations falls below a threshold. At this stage, we trim the edges of the observation cloud and perform a translational and scaling optimization on the data. If this recovery attempt results in an improved match, we switch back to the rotational variant and begin the loop anew. Otherwise, we terminate the process and return the final error metric vector.

The result obtained by comparing an observation to a reference may not be identical to that obtained by comparing

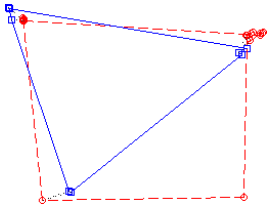


Fig. 4. Triangle gesture compared against the Square gesture, demonstrating the need for inverse matching.

the reference to the observation. In order to account for this asymmetry, we compute an inverse error metric vector by matching the reference to the final observation cloud. We define the average error magnitude as the arithmetic mean of the two magnitudes. The last step in the algorithm can be justified by the following example: assume the observation consists of a right-angle triangle and the reference represents a square, as seen in Fig. 4. The forward ICP loop will yield a very small error vector. However, the same cannot be said for the reverse ICP loop, since the forth vertex on the square will have no homologue in the observation cloud, and thus will increase the overall error magnitude.

As mentioned previously, we compare the observation with each reference cloud several times, once for each transformed variant of the data. At the end, we select the reference variant with the smallest error magnitude, and then select the best-matching reference shape using the same criterion. If the resulting error magnitude performs better than a certain acceptance threshold, we output the appropriate gesture to which the observation cloud corresponds to.

F. Choice of Reference Data

For each gesture, we systematically pick out a reference cloud from a set of training data. The selection mechanism is achieved by evaluating each cloud against the rest of the data using our algorithm and picking the one with the smallest average error. We have experimented with two other types of references as well. In the first of these, we average the data by first selecting a cloud with an average number of points in the training set. We then locate for each point in this cloud the closest points on the other clouds and finally average these point matches. However, because no two gestures are produced at the same rhythm, the temporal component completely distorts the positional values. As result, the averaged cloud generally no longer manifests the original shape. In the second approach, where we attempt to smooth the trajectory of the reference clouds manually, produces poor results. Since the observation data is not smoothed (to ensure real time performance), matching smoothed reference trajectories with the raw observations results in significantly poorer matching.

G. Experimental Validation

To rapidly prototype our system, we have implemented the algorithm using MATLAB. Currently, the detection speed is approximately 0.5 second, with the database containing 5 different reference shapes, each with 6 transformation variants. This result is not ideal, but it does satisfy our goals

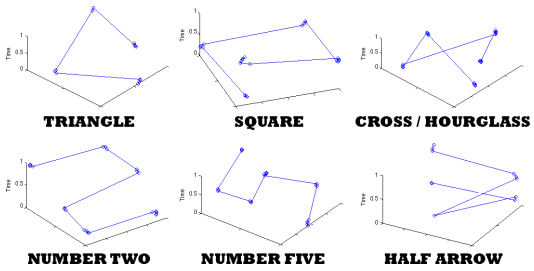


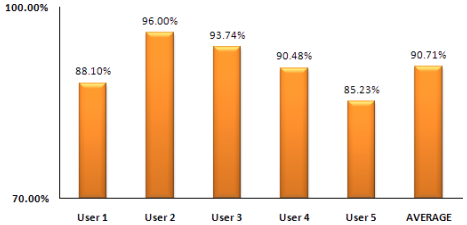
Fig. 5. Set of RoboChat Gestures used in our assessment.

for this prototype. Currently, we begin capturing gesture motions when two fiducial markers are detected by the robot's camera. Similarly, we stop the data capture and send the observation cloud to the ICP algorithm when the robot sees less than two markers for longer than a pre-determined timeout.

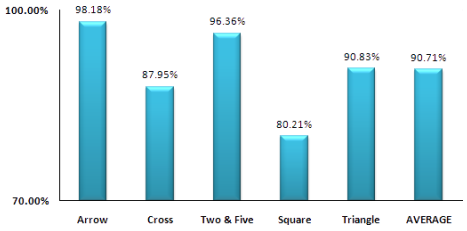
V. Experimental Results

A. Parameter Influence

Despite being algorithmically simple, the ICP code contains a number of parameters, which all can be fine-tuned to increase the performance of the overall system. The most important parameter is arguably the maximum allowed temporal distance, which is required to prevent nearby point pairs with distant temporal values to be associated. However, we have found that this threshold is very user-dependent, most likely due to the fact that every subject has a different sense of rhythm when performing the gestures. The importance of this value also depends on the roster of recognizable gestures. For example, we allowed in our experiments both the square and the hourglass shape. The temporal parameter can always be set to distinguish these two trajectories apart, but the numerical value of this parameter is different for each user.



(a) Per user performance data(all gestures/user).



(b) Per gesture performance data(all users/gesture).

Fig. 6. RoboChat Gestures best-match performance data.

We use two more values to determine the termination criteria for the overall ICP data flow – the minimum improvement in error magnitudes between consecutive iterations, and a maximum number of iterations allowed. These two

numbers do influence the correctness of the outcome (i.e. how accurately a gesture match can be made), but they mostly impact the speed of the algorithm.

Finally, the gesture acceptance threshold represents the largest overall error magnitude for which an observation cloud is deemed to match a reference shape. This value depends on the quality of the selected reference cloud, on the trajectory of the observation, and also on the user tracing the gestures. A badly chosen reference cloud might yield relatively large error magnitudes, and ultimately cause some observations to be falsely matched. Additionally, if the observation is not traced similarly to the chosen reference trajectories, it may result in a false positive or no detection at all. Finally, each user has a different way of drawing gestures, and thus the tolerances in the similarity of the shapes are necessarily different as well.

B. Data Gathering Setup

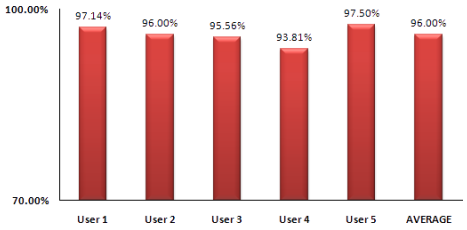
The RoboChat Gestures system was assessed using data sets provided by five volunteers. Each subject was given the instruction to draw the following shapes: triangle, square, hourglass (cross), half-arrow, and finally the segmented versions of the numbers 2 & 5 (Fig. 5). The participants were instructed to actively pause at each vertex. The subjects were shown the output of the camera used in the sessions, to let them know when their markers were out of the camera's field of view. However, most users commented that they did not look at this view, but rather at the visual feedback given each time a pair of fiducial markers were detected by the system. The latter form of feedback is more realistic in practice, because it can be implemented on the actual robotic platform as a simple visual or audible feedback.

C. Performance Assessment

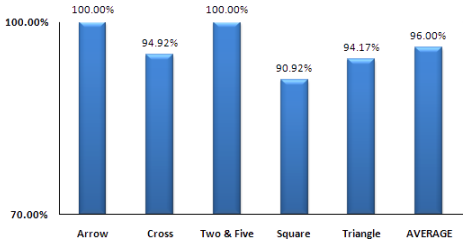
We have collected over 200 point clouds from the five participants. Our ICP algorithm yields an average success rate of 90% for matching the correct shape, as seen in Fig. 6(a) and 6(b). However, this rate increased to 96% if we account for observations which correctly match the runner-up reference shape, as seen in Fig. 7(a) and 7(b). Since these runner-up matches have a very small difference in their error magnitudes between the first and second match, we believe that we can increase the overall performance of the system by applying a Hidden Markov Model on the suggested semantic meanings of the gestures after detection.

As mentioned previously, each user has a different tolerance when tracing out trajectories. This difference is clearly reflected in the maximum error magnitudes for correct gesture matches, as seen in Fig. 8. However, we can also observe from Fig. 7(a) that the algorithm is still sufficiently robust to yield very close results across all users when the runner-up gestures are considered.

Additionally, there is a distinct gap between the average correct match error magnitude of 0.0040 and the average (incorrect match) runner-up magnitude of 0.01689. This result is very promising, since this implies that the gesture



(a) Per user performance data(all gestures/user).



(b) Per gesture performance data(all users/gesture).

Fig. 7. RoboChat gestures second-best-match performance data.

acceptance threshold has quite a bit of flexibility in term of its value, at least for gestures drawn by these five subjects.

Although it might seem that our system fails 10 *per cent* of the time when considering solely the best match, in reality the presented performance assumes that the gesture acceptance threshold is large, and thus does not discard any gestures at all. By tightening this threshold, the algorithm begins to reject matches with high error magnitudes, which are most likely to be the incorrect ones. Fig. 9(a) clearly indicates that the algorithm can successfully detect all gestures with an error magnitude below 0.01. At this value, Fig. 9(b) shows that only 10 *per cent* of the observations were classified as not being gestures. These are the true performances of our ICP algorithm, which is much more promising than our previous results seem to suggest.

The plot in Fig. 9(b) show an exponential increase in the rejection rate as the threshold tightens, and also indicates a threshold beyond which all gestures are rejected. This gives us a useful reference to set the gesture acceptance threshold value.

D. Robot Implementation

The prototype version of the RoboChat Gestures system has been tested on-board the Aqua underwater swimming robot [4] in a closed-water trial. While the marker detection and gesture extraction process took place on-board, we perform gesture detection using Matlab running off the robot, by using a fiber optic tether connecting the robot to a surface operator. While the under water setting did not

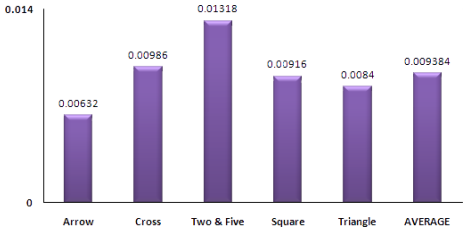
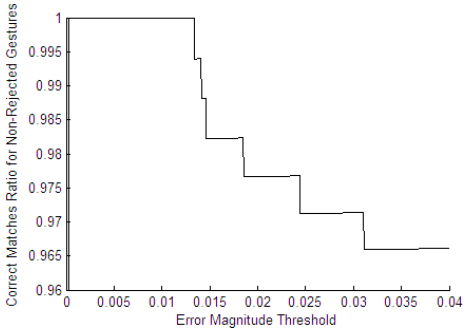
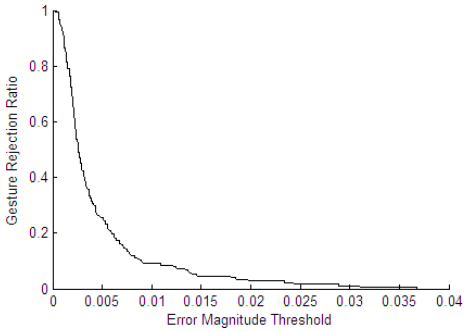


Fig. 8. Largest correct match data per user.



(a) Non-rejected gesture detection accuracy.



(b) Ratio of rejected gestures.

Fig. 9. Performance plotted against the gesture acceptance threshold.

permit gathering and analysis of detailed quantitative data, the gesture detection scheme performed robustly, with little additional cognitive burden imposed on the diver. These qualitative results show great promise in the system, and further experiments and enhancements are currently in the works.

VI. Conclusion

We present a vision-based interaction framework for operating robots in restrictive environments. We found that a gestural input mechanism alone was too error prone for our needs (incorrect interpretations could have high risk), but gestures combined with fiducial targets provided an attractive combination of ease-of-use, expressive power and robustness even underwater.

One very important feature for the detection algorithm to have is the ability to cluster the data clouds and extract vertices from them. If successfully executed, this step would significantly reduce the number of points in each cloud, and hence would drastically improve the speed of the system. Furthermore, the temporal information would be reduced into an ordered index for these vertex clouds, and thus would be possible to pair up gestures performed at significantly different rhythms, which our current algorithm is incapable of achieving. To aid in the robust interpretation of complex gestures, a probabilistic dialog model might be appropriate. In addition, we are interested in conceptual and practical feedback mechanisms to allow more robust interaction between the human and the robot.

In the near future, RoboChat Gestures will be integrated into the core RoboChat framework, to take advantage of the expressiveness of gestures, while maintaining the flexibility of the RoboChat language. Translating the code from MATLAB to C++ is also in the works, to maximize the gesture detection speed. In essence, we intend to implement a natural, robust yet infinitely expressive input interface for our underwater robot and for other similar machines.

References

[1] PhotoModeler Coded Targets Module by EOS Systems. Fore more information: http://www.photomodeler.com.

[2] Paul J. Besl and Neil D. Mckay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, February 1992.

[3] K.G. Derpanis, R.P. Wildes, and J.K. Tsotsos. Hand gesture recognition within a linguistics-based framework. In *European Conference on Computer Vision (ECCV)*, pages 282–296, 2004.

[4] Gregory Dudek, Michael Jenkin, Chris Prahacs, Andrew Hogue, Junaed Sattar, Philippe Giguère, Andrew German, Hui Liu, Shane Saunderson, Arlene Ripsman, Saul Simhon, Luiz Abril Torres-Mendez, Evangelos Milios, Pifu Zhang, and Ioannis Rekleitis. A visually guided swimming robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Edmonton, Alberta, Canada, August 2005.

[5] Gregory Dudek, Junaed Sattar, and Anqi Xu. A visual language for robot control and programming: A human-interface study. In *Proceedings of the International Conference on Robotics and Automation ICRA*, Rome, Italy, April 2007.

[6] R. Erenshteyn and P. Laskov R. Foulds L. Messing G. Stern. Recognition approach to gesture language understanding. In *13th International Conference on Pattern Recognition*, volume 3, pages 431–435, August 1996.

[7] Mark Fiala. Artag, a fiducial marker system using digital techniques. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 590–596, Washington, DC, USA, 2005. IEEE Computer Society.

[8] D. Kortenkamp, E. Huber, and P. Bonasso. Recognizing and interpreting gestures on a mobile robot. In *13th National Conference on Artifical Intelligence*, 1996.

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

[10] I. Poupyrev, H. Kato, and M. Billinghurst. *ARToolkit User Manual Version 2.33*. Human Interface Technology Lab, University of Washington, Seattle, Washington, 2000.

[11] Paul E. Rybski and Richard M. Voyles. Interactive task training of a mobile robot through human gesture recognition. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 664–669, 1999.

[12] Junaed Sattar, Eric Bourque, Philippe Giguere, and Gregory Dudek. Fourier tags: Smoothly degradable fiducial markers for use in human-robot interaction. *Computer and Robot Vision*, 0:165–174, 2007.

[13] Junaed Sattar, Philippe Giguere, Gregory Dudek, and Chris Prahacs. A visual servoing system for an aquatic swimming robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1483–1488, Edmonton, Alberta, Canada, August 2005.

[14] M. Skubic, D. Perzanowski, S. Blisard, A. Schultz, W. Adams, M. Bugajska, and D. Brock. Spatial language for human-robot dialogs. *IEEE Transactions on Systems, Man and Cybernetics, Part C*, 34(2):154–167, May 2004.

[15] J. K. Tsotsos, G. Vergheseand S. Dickinson, M. Jenkin, A. Jepson, E. Milios, F. Nuflo, S. Stevenson, M. Black adn D. Metaxas, S. Culhane, Y. Ye, , and R. Mannn. PLAYBOT: A visually-guided robot for physically disabled children. *Image Vision Computing*, 16(4):275–292, April 1998.

[16] Iuliu Vasilescu, Paulina Varshavskaya, Keith Kotay, and Daniela Rus. Autonomous Modular Optical Underwater Robot (AMOUR): Design, prototype and feasibility study. In *International Conference on Robotics and Automation*, Barcelona, Spain, 2005.

[17] S. Waldherr, S. Thrun, and R. Romero. A gesture-based interface for human-robot interaction. *Autonomous Robots*, 9(2):151–173, 2000.