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++) {
   angle = 90;
   duration = 2;
   Turn_Left(angle,
   duration);
   Move_Forward(duration);
}
```

```
4 REPEAT
   9 0 ANGLE
   2 DURATION
   TURN_LEFT
   MOVE_FORWARD
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.

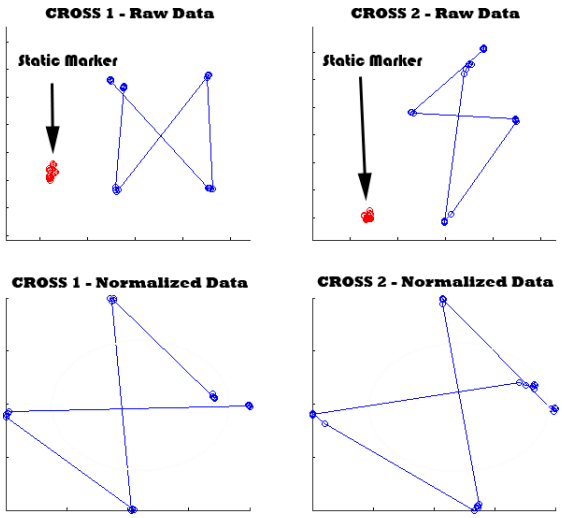


Fig. 2. Raw and pre-processed data for two RoboChat Gestures clouds.

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

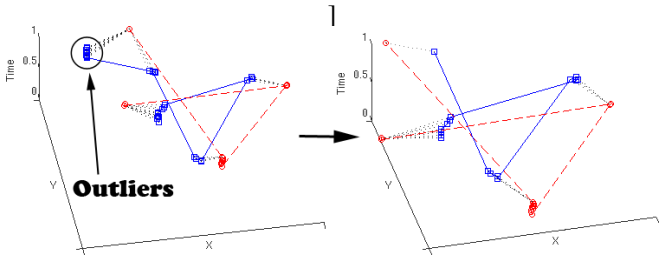


Fig. 3. Effect of trimming the point cloud.

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

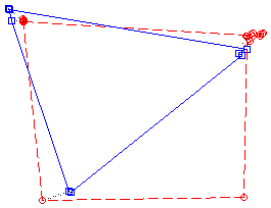


Fig. 4. Triangle gesture compared against the Square gesture, demonstrating the need for inverse matching.

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

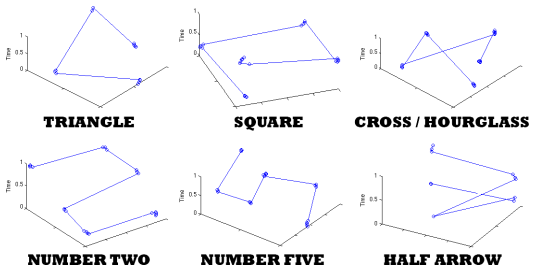


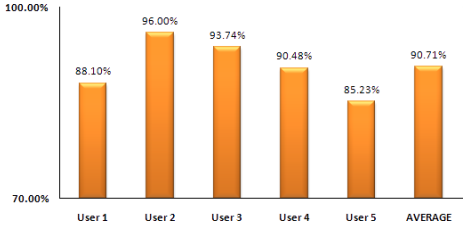
Fig. 5. Set of RoboChat Gestures used in our assessment.

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.

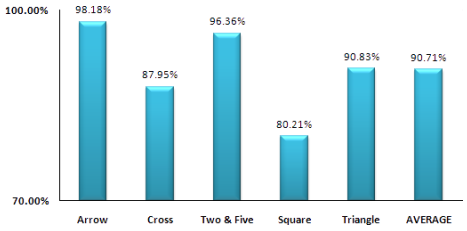
V. Experimental Results

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

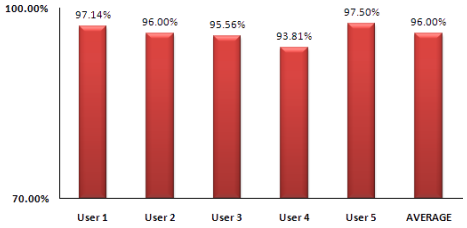
The RoboChat Gestures system was assessed using data sets provided by five volunteers. Each subject was given the instruction to draw the following shapes: triangle, square, hourglass (cross), half-arrow, and finally the segmented versions of the numbers 2 & 5 (Fig. 5). The participants were instructed 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

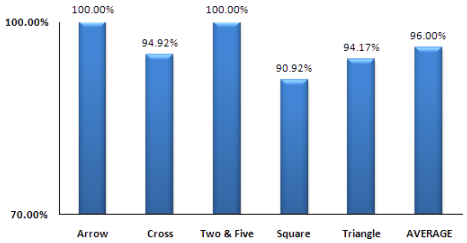
We have collected over 200 point clouds from the five participants. Our ICP algorithm yields an average success rate of 90% for matching the correct shape, as seen in Fig. 6(a) and 6(b). However, this rate increased to 96% if we account for observations which correctly match the runner-up reference shape, 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

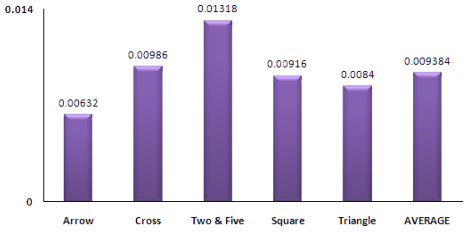
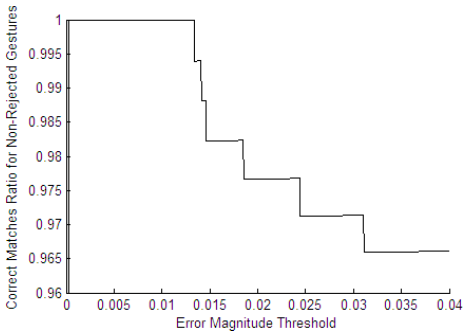
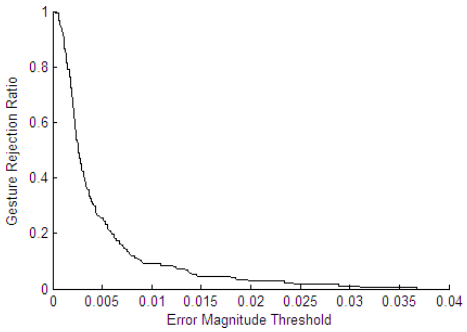


Fig. 8. Largest correct match data per user.



(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.