COOPERATION

Grant Agreement No. 611373

CADDY

Cognitive Autonomous Diving Buddy

FP7-ICT-2013-10

D4.4. Implementation and execution of diver buddy tasks algorithms initiated by interpreted hand gestures

Due date of deliverable: 08/2016

Actual submission date: 09/2016

Start date of project: 01 January 2014 Duration: 36 months

Organization name of lead contractor for this deliverable: IST

Revision: version 1

Dissemination level		
PU	Public	x
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Contents

1 Outline of the deliverable

This deliverable describes the diver buddy tasks algorithms designed and implemented in the scope of the CADDY project. The aim of such tasks is to provide the CADDY system with useful features so that the divers' work is eased by doing tasks like imagery acquiring, picking up objects from the boat (without the need of waiting for decompression, carrying heavy tools, among other aspects).

Over this document, these tasks "diver buddy tasks" that were implemented are going to be described from the theoretical point of view, as well as in-water trials results.

2 Performing a mosaic over an area

Navigation and Control of the robots

One of the most resource-demanding tasks where the CADDY system can undoubtedly improve diving is mosaicking. Nowadays it is required that a human diver goes on a lawn mower maneuver for huge amount of time to acquired quality imagery. They resort to big structures where cameras and light are embedded as seen in Figure 2.1.

Moreover, a navigation system for the diver has to be in place. Usually, before the survey a grid made of ropes is attached to the area of interested so that the diver can follow roughly the maneuver.

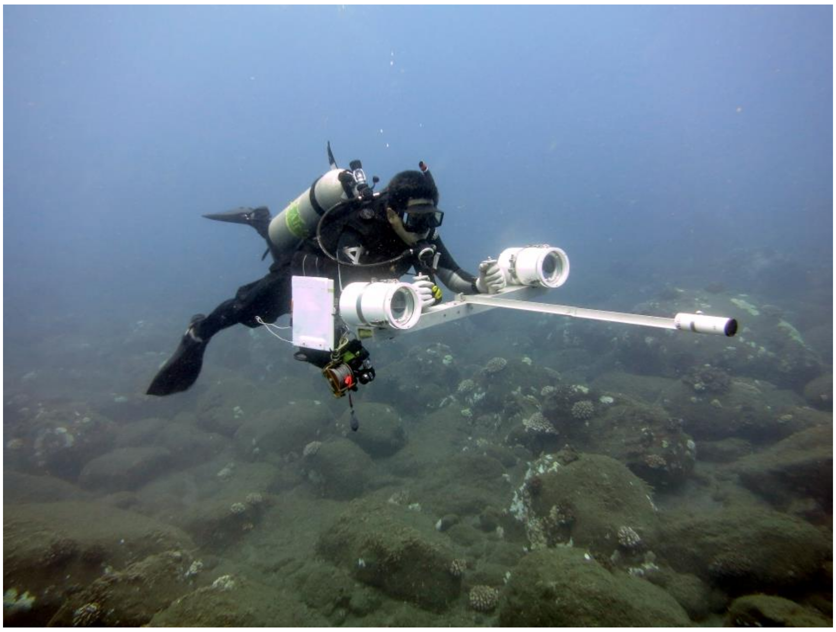


Figure 2.1 – Diver acquiring stereo images (Photo: Andrew Gray/NOAA)

With the CADDY system, both navigation and imagery acquisition are solved, making the same operation faster, with fewer human resources, thus safer.

Regarding implementation, while the surface vehicle is performing the same control strategy as described in Deliverable 4.2, Implementation of cooperative control and formation keeping with diver in the loop, BUDDY is performing a path following maneuver. In [1], one can have a full description of the setup used on this experiment.

The goal of a path-following algorithm is to guide a vehicle to a path at a given velocity profile. No time constraint is set, i.e. it is not designed such that a desired position for the vehicle is issued for each time instant. The current implementation of the path-following algorithm works with paths defined at a predefined depth or altitude. The algorithm works in an outer-loop control design and it is responsible to issue the convenient heading and speed commands to the inner-loops based on the actual position and desired mission. A detailed description of the algorithm can be found in [2]. In short, the algorithm iterates over the following steps:

- computes the closest point on the path, $\boldsymbol{p_c}$, to the estimated position of the vehicle $\boldsymbol{p}$;
- computes the angle β of the vector tangent to the path at $\boldsymbol{p_c}$;
- computes the cross-track error e as the projection of the position of the vehicle relative to the position of the closest point, $\boldsymbol{p} - \boldsymbol{p_c}$, on a direction $\beta + \frac{\pi}{2}$;
- issues a heading command, ψ_d, computed by a Proportional-Integral (PI) controller on e, and a speed command, u_d, to the inner-loops, as given by

$$\psi_d = \beta + \sin^{-1}\left[\sigma\left(-\frac{K_p}{u}e - \frac{K_i}{u}\int_0^t e(\tau)d\tau\right)\right]$$

$$u_d = u(\boldsymbol{p_c})$$

where $\sigma(\cdot)$ is a saturation function that limits the result from -1 to 1, $u(\cdot)$ defines the desired speed as a function of the position along the path and K_x the PI controller gains, Figure 2.2 illustrates the algorithm frames.

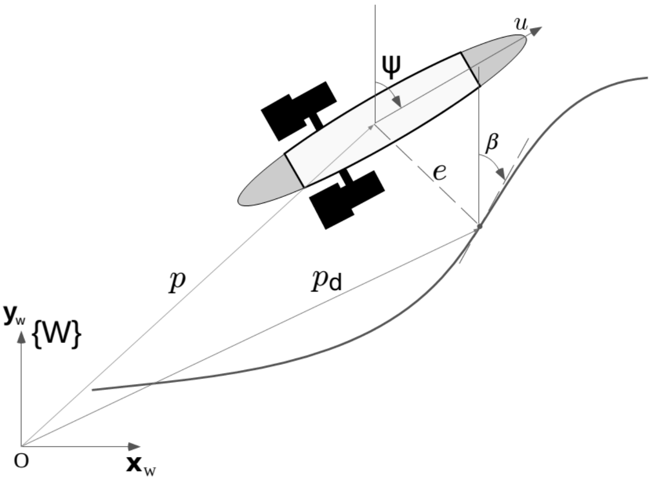


Figure 2.2 – Path following frames showing the body frame and path relations in the navigation frame {W}

Once the path following controller generates surge and yaw set-point, they are passed to the Buddy low-level control architecture. The controller architecture is divided into kinematic position/orientation controllers and dynamic velocity controllers similar to the ones described in [3]. The low-level controllers allow for improved tracking of the yaw reference to ensure path-following stability.

Results

This CADDY functionality was demonstrated in March 2016, in Lisbon, during the 2nd year Review meeting. While preparing for the final demonstration, the path-following performance was tested several times. The lawnmowers varied in length and width and had a predefined 5 m distance between legs. These lawn-mowers, nine in total, were used to give a general performance overview. Cross-track errors for all lawn-mowers and the total distribution are shown in

Figure 2.3, respectively.

The median and mean cross-track errors are both $\approx\ 5\,\mathrm{cm}$. The maximum cross-track error is $\mathrm{L}_\infty\ =\ 2.82\,\mathrm{m}$ and, together with most of the larger cross-track errors, it occurs during the mission start. Lawn-mowers are generated on request over the acoustic transmission and the lawn-mower origin is the current vehicle position. However, due to the asynchronous nature of ROS, the “current” position is taken before the newest position correction provided in the same acoustic message. In fact 50 % of cross-track errors are between $-0.09\,m$ and $0.18\,m$ thus providing a good path-following performance when taking into account sparse position updates.

Shown in Figure 2.4, the example lawn-mower is the final demonstration initiated by the reviewers. The lawn-mower parameters, length and width, were given as the image processing result and relayed over the surface Medusa, through acoustics, to Buddy. The figure shows the underwater and surface vehicle movement during the mission. The given inverse USBL positions are used for visualization purposes since the estimation filter takes surface position, acoustic range and bearing into account separately. The approach allows utilization of precise USBL range measurements even in cases when the bearing measurements are corrupted. On the figure it can be seen in some legs where the inverse USBL calculations diverge from the path.

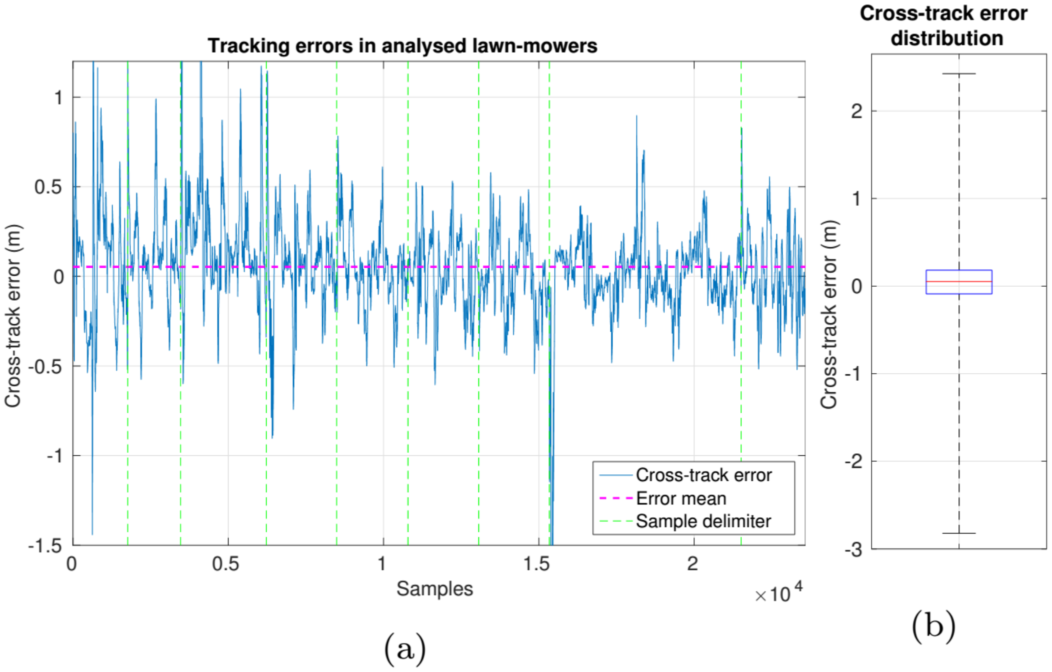


Figure 2.3 – (a) The cross-track error of all lawn-mower experiments and its distribution in (b). Vertical green lines represent the border between lawn-mowers.

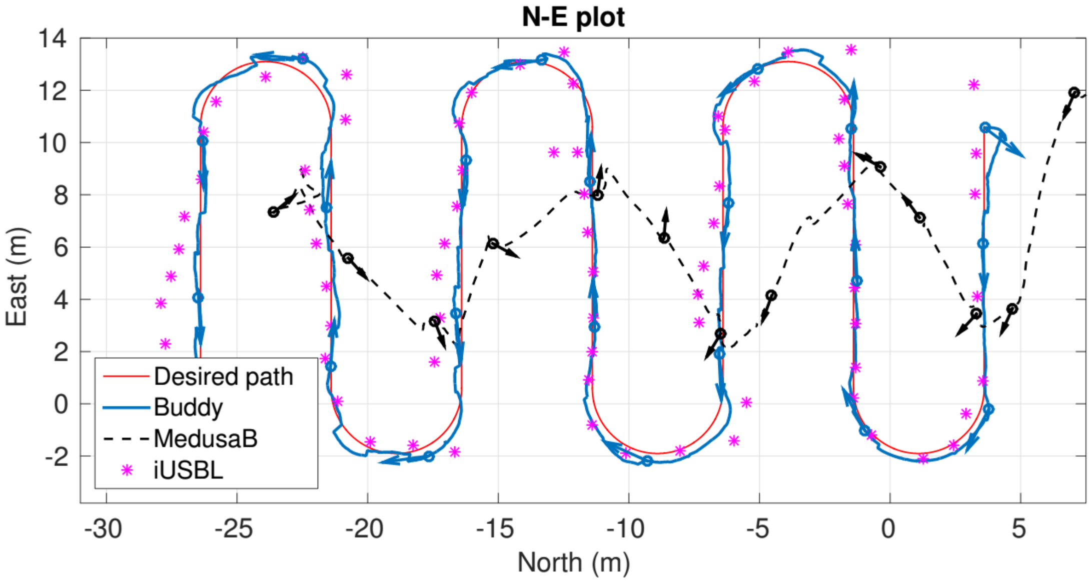


Figure 2.4 - Final demo lawn-mower experiment. The surface vehicle trajectory (dashed-line) is shown for easier visualization of the surface controller behavior

Mapping

Simultaneous Localization and Mapping (SLAM) is the process by which both the trajectory of the robot (localization) and a representation of the environment (mapping) is estimated at the same time [4]. Graph SLAM methods [5] represent the spatial relationships between sensor observations as a graph, also called a pose graph. Vehicle poses are estimated for each sensor observation of interest, forming the vertices of such a pose graph. Edges encode spatial constraints derived from the sensor data recorded at these poses. This may include motion estimates from a registration method or from vehicle odometry (also called navigation), as well as more complex constraints, e.g. feature observation constraints in the spirit of bundle adjustment [6].

Robust graph SLAM methods extend the traditional graph SLAM formalism by explicitly modeling and solving for outliers. Outliers in the spatial constraints may be the result of failed registration, e.g. divergence of ICP, wrong data association, or erroneous sensor data. Generalized Graph SLAM [7] developed at JACOBS is such a robust SLAM method. It explicitly extends the traditional SLAM representation by incorporating multiple possible motion estimates and the presence of a null-hypothesis, the hypothesis that no motion estimate is correct. This is achieved by replacing unimodal normal distributions with mixtures. A novel optimization method, called Generalized Prefilter, is also developed to be able to estimate accurate maps using this new representation. It is introduced as a mid-stage between the SLAM frontend, which is responsible for creating the pose graph data structure by performing the motion estimation, and the backend, which focuses on continuous optimization of the pose graph to estimate an accurate map. This mid-stage performs only discrete optimization based on the several choices of motion estimates in the Generalized Graph SLAM framework. It performs a minimum spanning tree traversal of the multimodal hypergraph using the total number of mixture components per edge as a weight, during which a combinatorial tree is generated with each leaf representing the exact component (including the Null hypothesis choice for hyperedges) chosen in the hypergraph traversal. Each component carries relative pose information used to assign poses to vertices visited in the traversal. This combinatorial tree is searched using beam search, where only the best N leafs, i.e., the ones with the highest joint probability given the computed poses, are kept at each iteration of the hypergraph traversal. When the traversal completes, the best remaining leaf of the combinatorial search tree, and thus the component choices along the corresponding hypergraph traversal, is picked. The computed poses on the path to this leaf are assigned to the vertices in the graph, and the remaining ambiguous edges are disambiguated by picking the component that best explains the estimated poses of the related vertices. Finally, the disambiguated unimodal graph including the computed vertex poses is then returned for subsequent continuous optimization in a standard SLAM backend.

In our article published in the International Journal of Robotics Research in 2016 [4], Generalized Prefilter is compared against a number of other robust SLAM methods.

Generalized Prefilter outperforms these competing methods by several orders of magnitude. This is further substantiated by results obtained in the CADDY project.

The data collected by the CADDY system included high-resolution stereo imagery at approximately 4Hz as well as navigation data of the submerged vehicle which was already corrected by USBL and DVL measurements using a Kalman filter. In addition to a 3D registration method using SURF features localized in 3D through triangulation in the stereo images and subsequent RANSAC, a second motion estimate was produced from the vehicle navigation. While the navigation was quite good in general, it contained many sudden discontinuities from outliers of the USBL system. It thus served as an important but relatively unreliable information source. Therefore, a fusion of both sources of motion information, the 3D registration as well as the navigation, was proposed using the Generalized Graph SLAM framework. This resulted in a pose graph having two motion estimates for sequential edges where the vehicle navigation and 3D registration did not agree up to a small threshold. All information was thus preserved, at the expense of also including failures of either method. Generalized Prefilter was successfully used to estimate an accurate 3D map using this data.

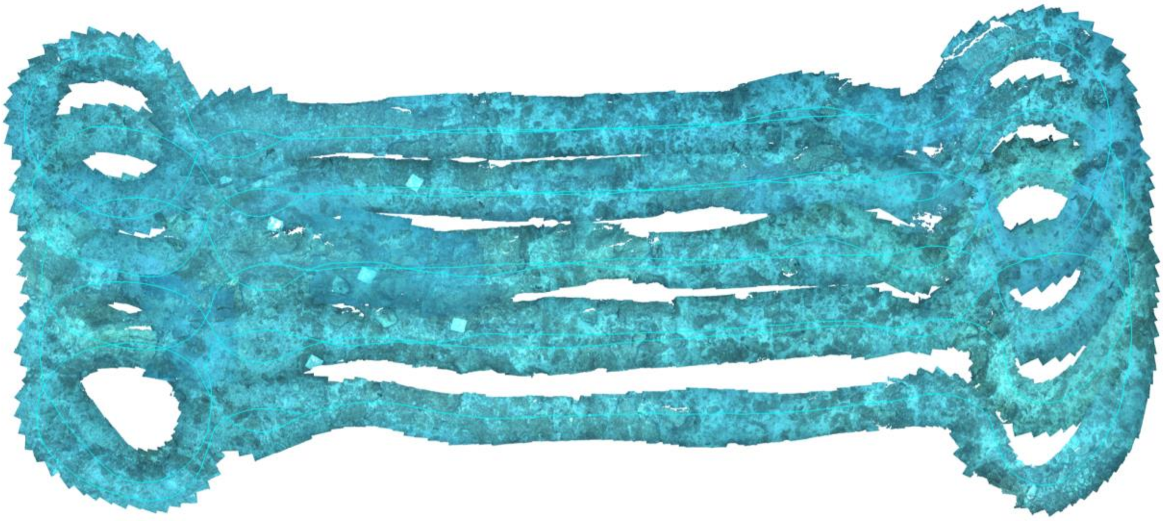


Figure 2.5 – A top-down view of the 3D map produced by Generalized Prefilter. Note the clearly visible and well-aligned square concrete markers (left side)

Other robust SLAM methods, all of them greedy, do not perform as well as Generalized Prefilter.

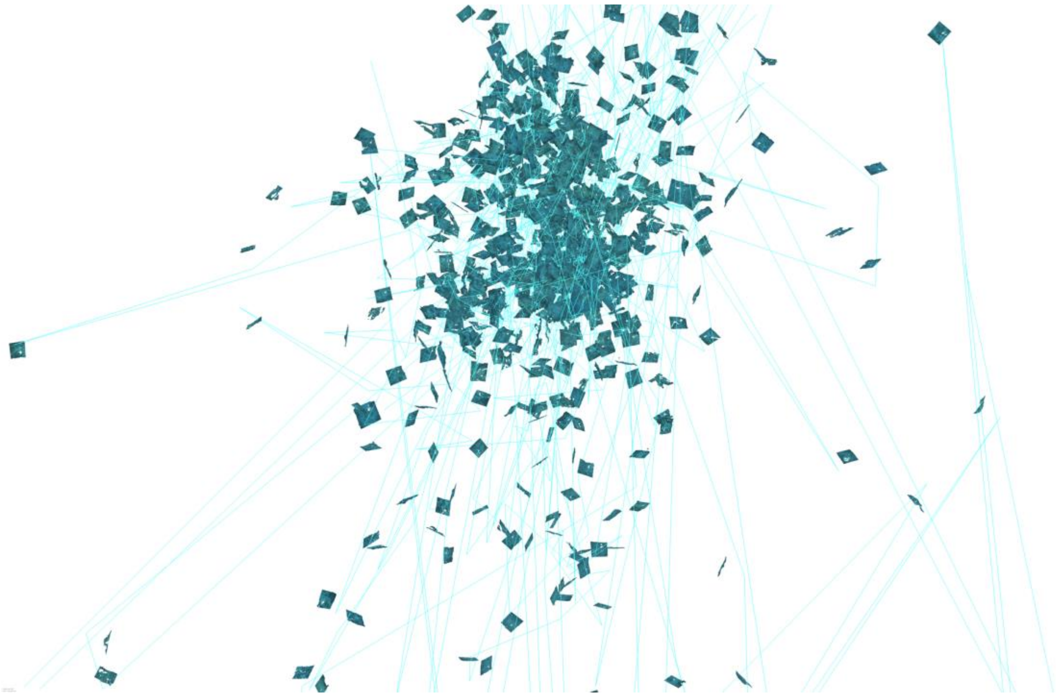


Figure 2.6 – Result of Olson and Agarwal's MaxMixture [8]

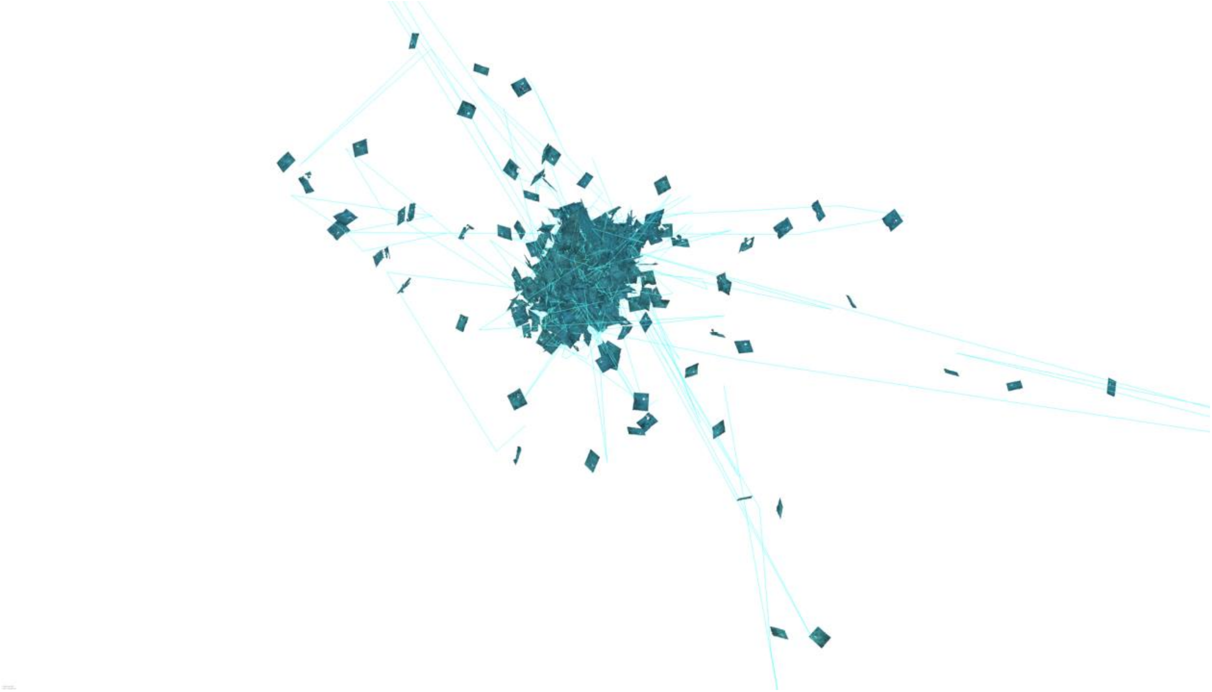


Figure 2.7 – Result of Suenderhauf's Switchable Constraints [9]

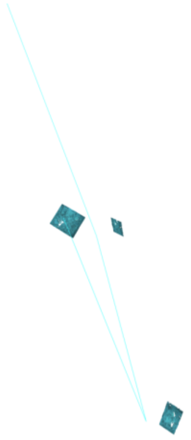


Figure 2.8 – Result of Agarwal's Dynamic Covariance Scaling [10]

Before the experiment, markers were introduced in the area of interest and relative distances were measured. This data was used to quantify the accuracy of the maps produced by the different methods shown above. Generalized Prefilter achieved a root mean square error of only 0.04m. Other methods showed a significantly higher error of up to 600m due to their divergence.

3 Observing the diver

As mentioned in previous deliverables, diver observation/monitoring is carried out by the virtual target algorithm. The algorithm depends on quality of the diver position estimator which fuses measurements from camera, sonar, USBL measurements with raw diver data. Following sensor measurements are available:

1. Stereo camera
   a. provides range and bearing to diver point-cloud – close distance
   b. provides the diver relative orientation to the camera
2. Sonar
   a. provides range and bearing to diver – medium to close distance
3. USBL
   a. provide range and bearing to diver – short to long distance
   b. provide diver heading and depth through acoustic communication

Diver depth is measured from the available depth sensor of the Seatrac modem, offering precision of 0.5 m. Absolute heading of the diver is estimated from diver-net measurements. These measurements arrive once per cycle or approximately every 5 sec. Low update rates do not affect depth since it changes slowly during the dive. However, orientation changes by the diver are much faster and estimation can only be obtained by camera aiding in scenarios where the diver turns in-place. Additionally, changes in orientation during movement can be observed directly by course estimation from range and bearing measurement. Based on these measurements it is assumed that the relative position of the diver to Buddy is known. Previous sections have shown that knowledge of the absolute Buddy position is also available. The Serret-Frenet path {*P*}, the body {*B*} and navigation {*N*} frames were defined in previous deliverables concerning the virtual target tracking algorithm and are not be repeated here. The main terms are the vehicle position in {*P*} denoted as ***d*****,** the diver position in {*N*} denoted as $\boldsymbol{r^n}$ and the desired monitoring position μ^* given as a scalar value of the parameterized path. Note, that μ^* is zero, i.e. in front of the diver, for the monitoring position but can be used to specify arbitrary positions relative to the diver.

Using this data the kinematic controller for the diver observer algorithm can be stated as:

$$\boldsymbol{\nu_1} = \boldsymbol{R_p^b}\left(-\boldsymbol{K_{pd}}\boldsymbol{d} + \boldsymbol{R_n^p}\dot{\boldsymbol{r}}^{\boldsymbol{n}} + \dot{\mu}^*\boldsymbol{t}\right)$$

$$\dot{\boldsymbol{\varpi}}^* = k_{\varpi}(s - \sigma_{\tilde{\boldsymbol{\varpi}}} \tanh k_{\tilde{\boldsymbol{\varpi}}}\tilde{\boldsymbol{\varpi}} + \dot{\mu}^*)$$

where $\dot{\boldsymbol{\varpi}}^*$ and $\tilde{\boldsymbol{\varpi}}$ represent the desired speed of the virtual target along the path and the difference between the virtual target and desired monitoring position. The sigmoid term is added to control the magnitude of the path error thus preventing escape of the virtual target from the vehicle in presence of large errors. Additionally, this term allows faster convergence to the desired point. Detailed description and derivation of the controller can be found in [10].

Previous experiments focused on use of virtual divers and vehicles to emulate diver behavior. In contrast, preliminary results shown here include tests with a human diver during stereo camera tuning experiments in the Croatian Shipbuilding Institute. Results present basic diver tracking using camera measurements only.

Two experiments from the emulated diver are shown in Figures 3.1 and 3.2. First, the diver is static and changes only orientation in 15, 30, 45 and 90° steps. Figure 3.1a shows the absolute path error for radius R=4m. Most range error samples are within +/- 0.1m indicating close tracking of the safety circle around the diver. Transition speed is slow due to limited sway speed.

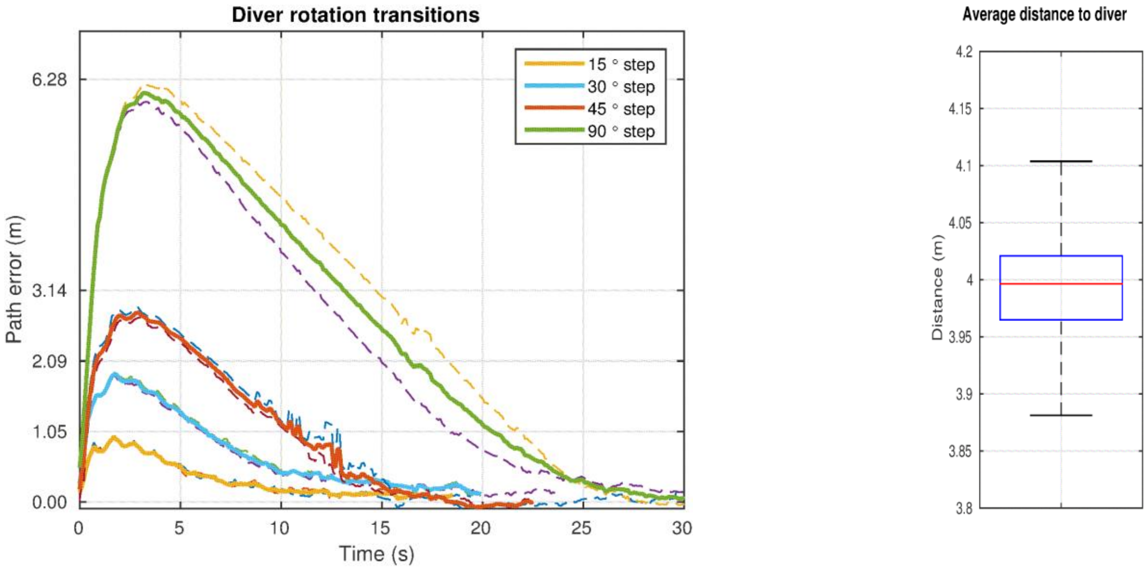


Figure 3.1 – Rotation experiment transitions on left and distance statistics during experiment on right (USV emulated diver)

Second experiment considered a dynamic emulated diver where the vehicle moved along a path and then performed two 90° turns to force the vehicle in an overtaking maneuver. It can be seen that the vehicle converges in front of the diver after overtaking. The tracking errors are inside +/- 0.4m showing a bias. Since pure measurements are used the emulated diver speed is not estimated and no feed-forward is present. This causes the proportional controller to exhibit a static error during tracking.

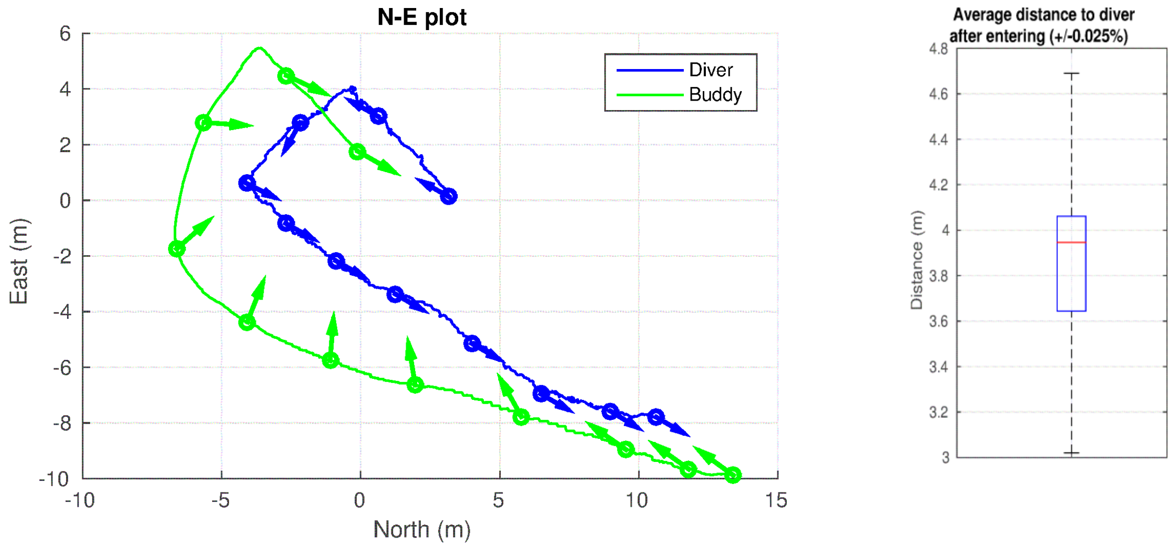


Figure 3.2 – Diver tracking and overtaking experiment (USV emulated diver), distance statistics is shown on right

Previous example considered only the use of multibeam sonar measurements to evaluate basic tracking capabilities. Sonar detection supports blob and highlight tracking and can be easily used with diver emulation. However, issues arise in testing stereo camera detection and recognition as an emulation is not easily achievable. Therefore, preliminary test results for tracking with raw stereo camera measurements utilize a real diver in the loop.

Two rotation experiments were carried out with the diver choosing the pace and magnitude of the rotation. Usually, the diver chose increments between 15-30° and waited for the vehicle to achieve close convergence. The path error was normalized with the safety circle radius to indicate normalized angular errors. Normalized angular errors should allow comparison for missions with different safety distances. Additionally, relative comparison of diver heading and path locations is achieved.

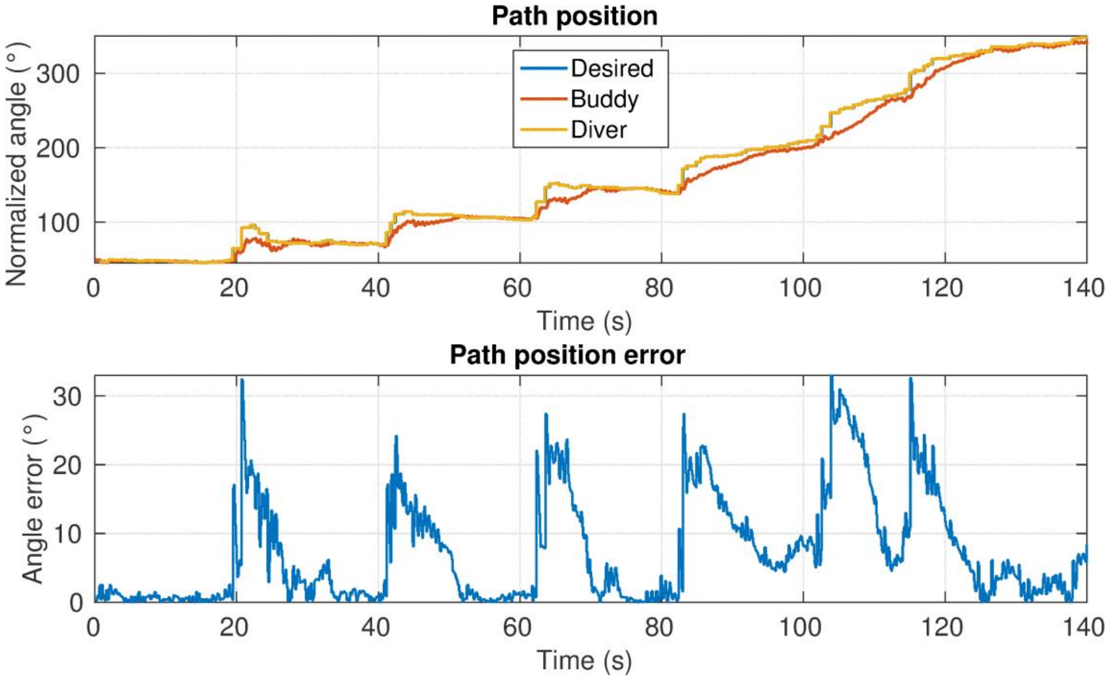


Figure 3.3 – Path position (first rotation experiment)

Figure 3.3 shows the first rotation experiment results. The jumps indicate diver rotation In place. Notice than on average less than 20 seconds are required for convergence. Occasionally, the diver did not wait for convergence as seen in t=100s. Notice that the desired path position always matches the diver heading.

Range and bearing, as estimated by the stereo camera, are shown in Figure 3.4. The range error is < 0.2m during most of the experiment and bearing angle is fluctuation between -/+ 5°. Diver and Buddy movement, for both rotation experiments, are shown in Figure 3.5a and Figure 3.5b. Observe that the diver appears to be moving in a circular fashion. The artifact occurs due to Buddy dead-reckoning as the surface vehicle was not part of the preliminary experiment. The dead-reckoning error is better observable in the case where the diver turns first clockwise and then anti-clockwise around the same point. However, due to the error the paths diverged over time.

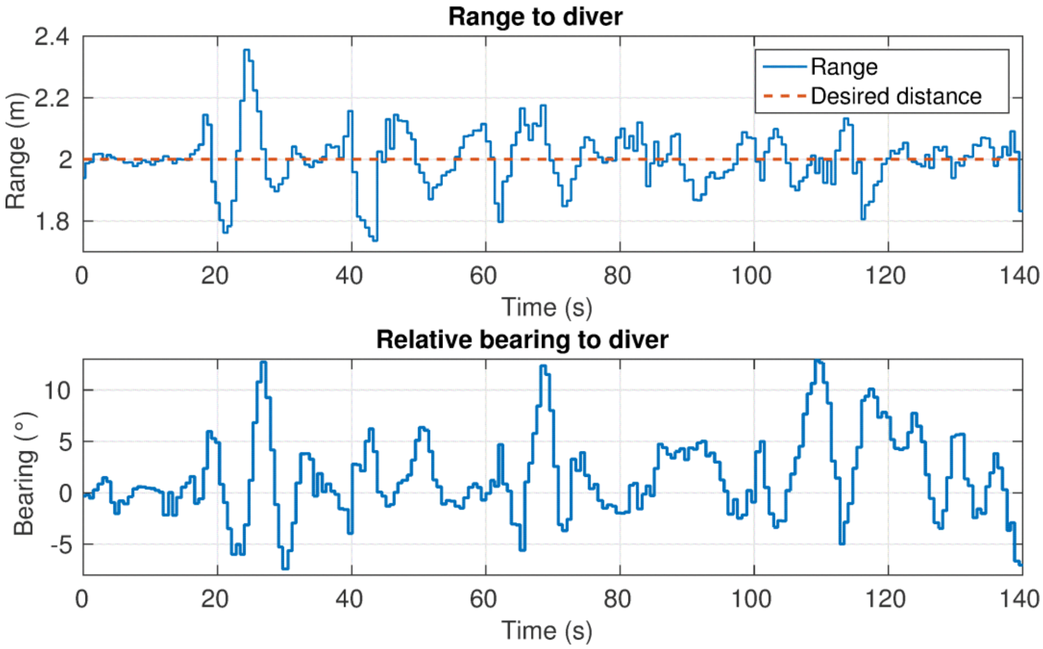


Figure 3.4 – Range and bearing towards diver estimated from stereo camera (first rotation experiment)

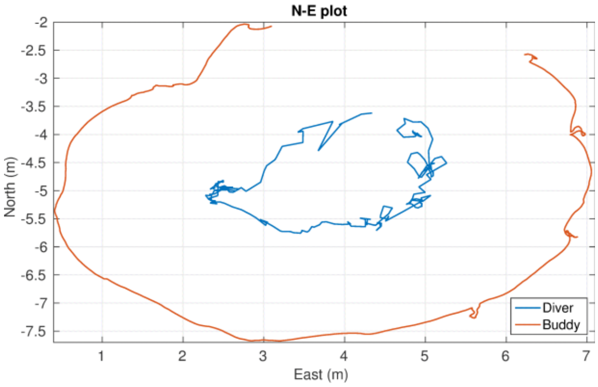


Figure 3.5a – Dead-reckoned position (first rotation experiment)

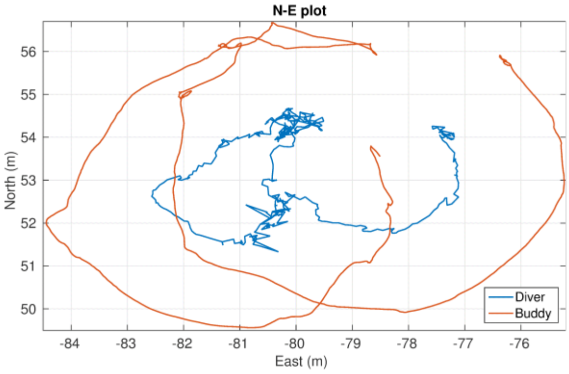


Figure 3.5b – Dead-reckoned position (second rotation experiment)

The second experiment is shown in Figure 3.6 and Figure 3.7. The tracking behavior exhibits similar performance with an exception of the outlier around 40s. Convergence in front of the diver can be observed in the lower plot of Figure 3.6. However, the bearing estimates are increased between +/- 10°. While the effect of this measurement is smaller for the used range R=2 m it contributes to larger oscillations as the safety distance increases. Considering this the measurements from the stereo camera should be ignored above 3, 3.5 m as the bearing estimate will affect also the diver position estimator.

Figure 3.8 and 3.9 show the experiment in which the diver moves toward and away from the vehicle. The vehicle tracks the diver and positions directly in-front. The angular error is small since only linear motion is performed by the diver. When observing range static errors can be seen at 70-90s and 110-160s as the diver moves towards and away from the vehicle. Part of the static error is again attributed to the fact that no feed-forward is calculated and the integral constant is turned off. Additionally, for the safety of the diver the maximum vehicle speed was limited to 0.3m/s in surge thus increasing the error in cases when the diver moved faster than this limit.

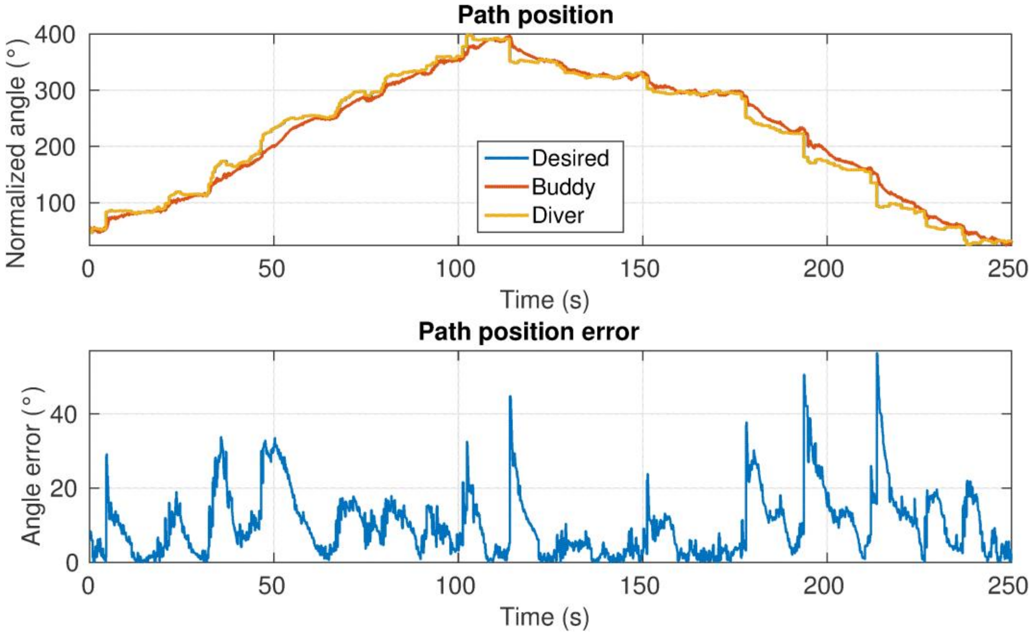


Figure 3.6 – Path position (first rotation experiment)

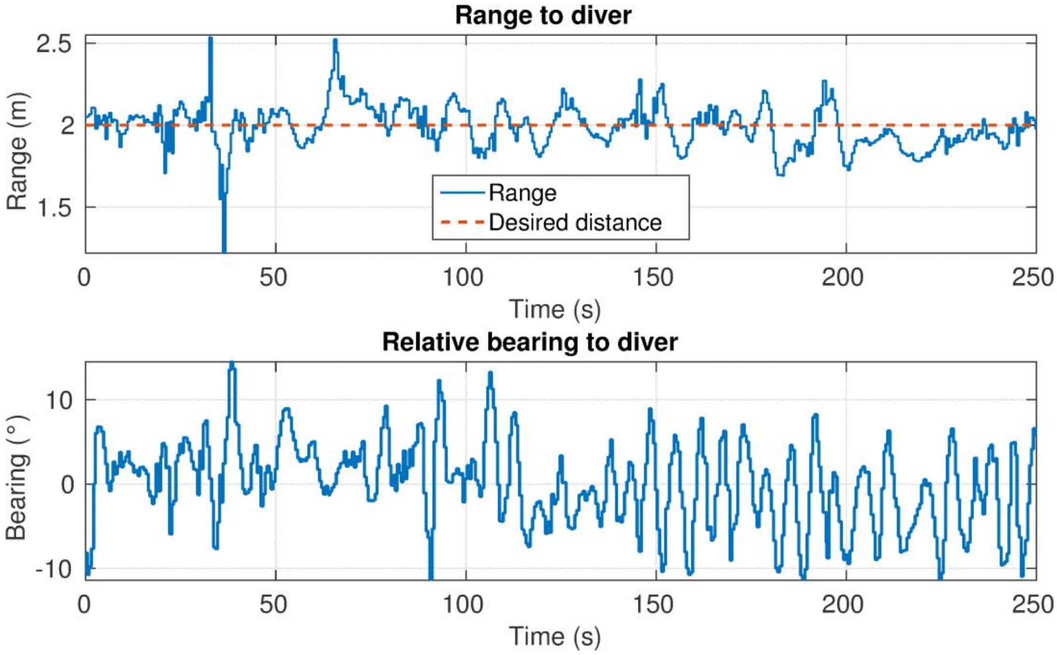


Figure 3.7 – Range and bearing towards diver estimated from stereo camera (second rotation experiment)

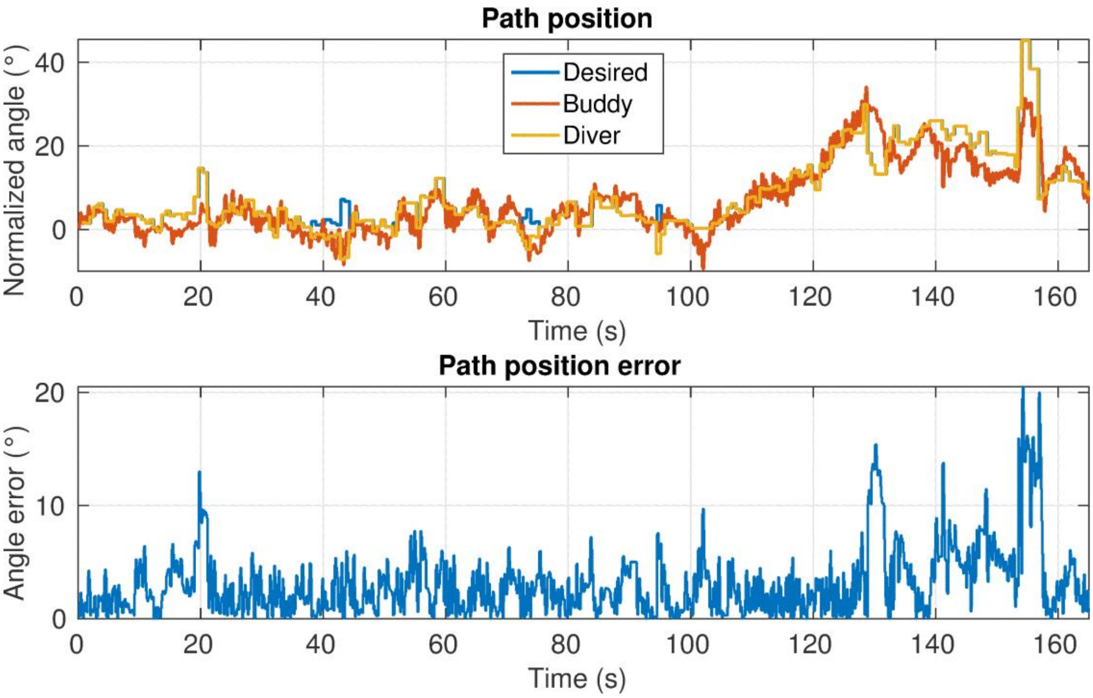


Figure 3.8 – Path position (movement experiment)

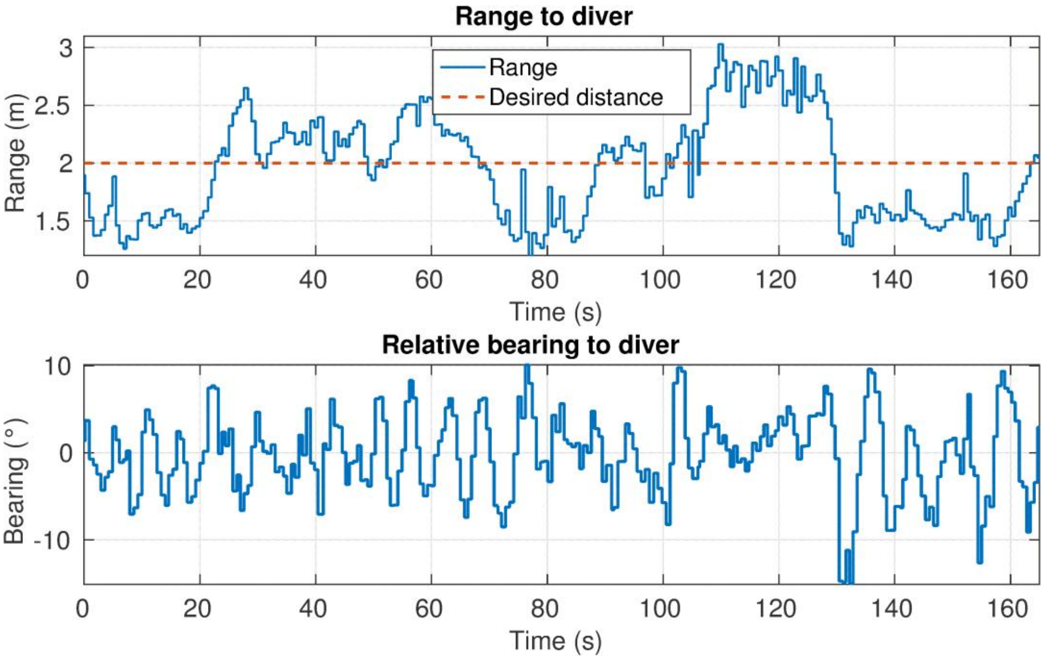


Figure 3.9 – Range and bearing towards diver estimated from stereo camera (movement experiment)

4 Carry-to-boat

Validation trials introduced a Buddy slave action which requires Buddy to carry objects from the diver to the surface station and back, see **Error! Reference source not found.**. The behavior consists of two steps: a) indicating to Buddy that it should allow the diver to approach, b) commanding Buddy to bring the object to the boat.

The first step is commanded by use of the tablet, where the diver issues an *allow_approach* (ID=5) command to Buddy. Once Buddy receives the command it confirms it back over acoustics to the diver and additionally indicates the current mission on the Buddy tablet. The step involves at maximum two controllers - depth and heading. Position control is avoided to allow divers to manipulate Buddy with minimum risk. Depth control is required to avoid surfacing during diver approach. Orientation requires low-usage of horizontal thrusters and can be considered safe and is therefore recommended to be part of the *allow_approach* state.

The state machine accommodates this behavior and switches the usual *guide_me* state in favor of *allow_approach*. Once the diver approaches and hands over the desired tool or found artifact to Buddy he can distance himself from buddy and initialize the *guide_me* behavior via the diver tablet. The second step is then performed by use of gestures to indicate what Buddy should do with the object, e.g. carry it to the boat. Note that in general the second step is optional.

The *carry_to_boat* behavior is triggered by use of gesture commands and consists of three parts. The first part activates surfacing while maintaining position. Once on surface Buddy establishes wireless communication with the surface vehicle and hands over the acoustic interrogation master role. The surface vehicle resumes acoustic localization and communication with the diver. During the second, surface part, Buddy proceeds towards the boat to deliver the object. Successful object retrieval is indicated by the command centre upon removal of the object from Buddy. The third part of the command is activated prompting Buddy to return to the diver in *guide_me* behaviour. Acoustics interrogation on Buddy is resumed upon diving while the surface vehicle switched back to interrogation slave role. Three controllers: heading, depth and horizontal position; are involved during all parts of the *carry_to_boat* behaviour.

All controllers used were already presented in previous deliverables and publications, see [3].

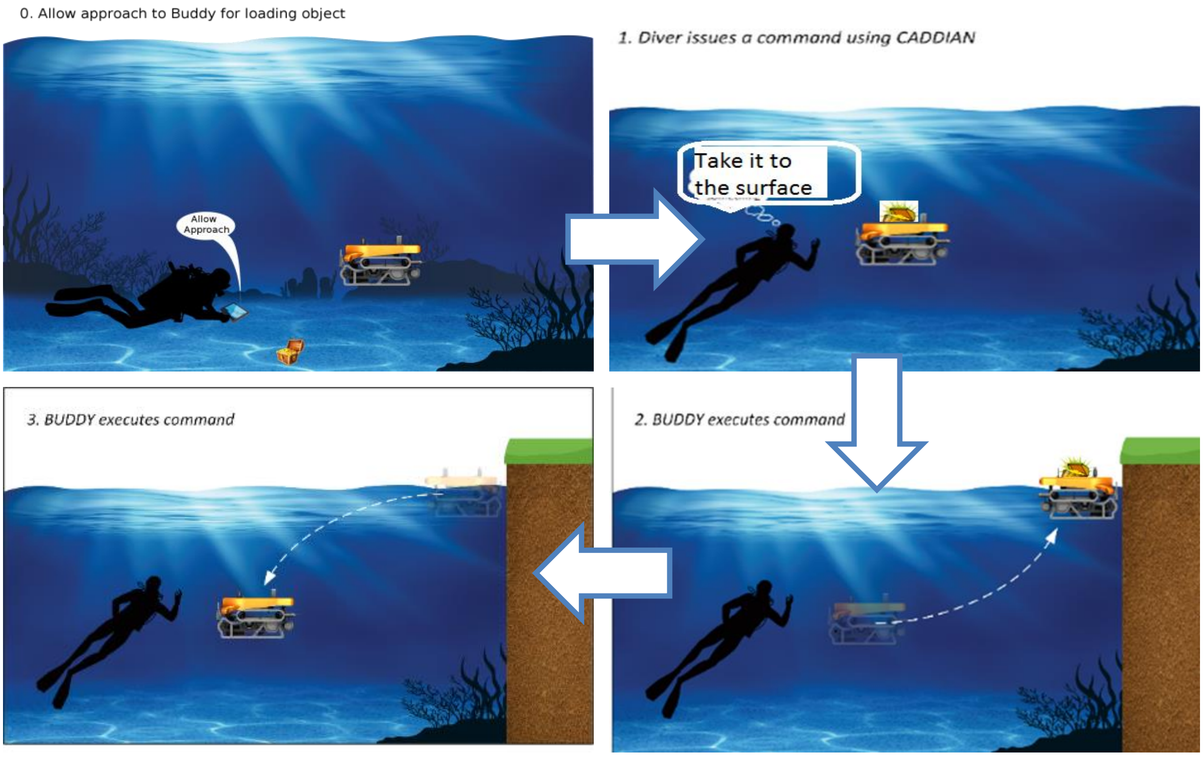

0. Allow approach to Buddy for loading object
Allow Approach
1. Diver issues a command using CADDIAN
Take it to the surface
2. BUDDY executes command
3. BUDDY executes command

Figure 4.1 – The Carry-to-boat concept

5 Robotic tasks activation

The actions executed by the robotic platform, previously described in this deliverable, are triggered by the diver by means of hand gestures that are recognized by the robotic system and translated into suitable commands.

In particular, the mission controller acts as a cognitive planning and supervision system taking as input the decoded gesture sequence executed by the diver, in turn generating proper task activation actions in such a way to trigger the required capabilities needed to provide the requested support. The main goal of the mission control module is to abstract the robotic capabilities at logical level and manage the activation of the real robotic primitives, while at the same time resolving possible conflicts that may arise when selecting the robotic capabilities to be triggered.

The mission controller architecture is implemented my means of a modular Petri net that allows the automatic conflict detection and resolution among the task activations.

The detailed description of the hand gesture interpretation pipeline and mission controller development and employment is reported in the Deliverable D3.3.

In Figure 5.1, for ease of reference, the sequence of gestures that trigger the mission "Carry-to-boat" are described. Further details about CADDIAN and its gestures can be found in Deliverable D3.1.

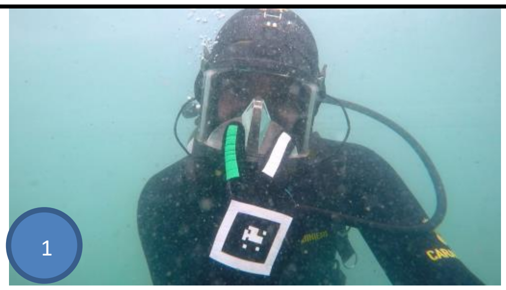
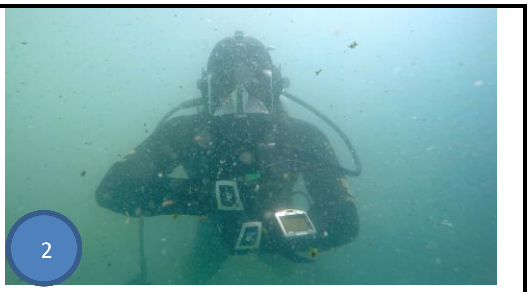
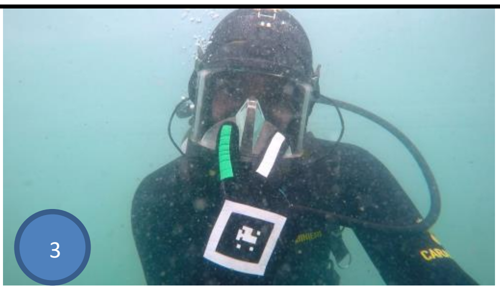
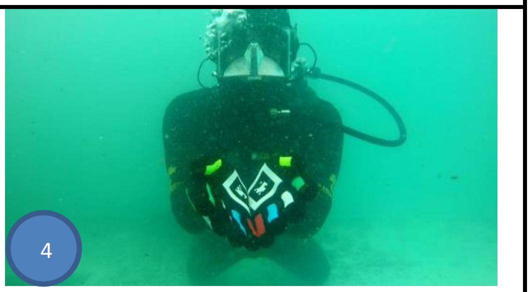
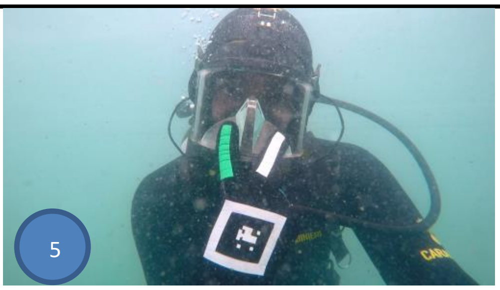
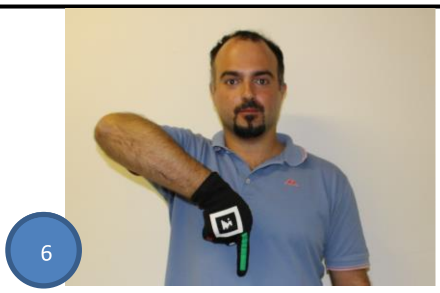
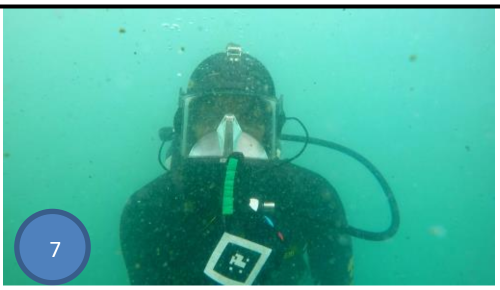
Gestures		
CADDIAN	A	carry
Semantics	start_comm	carry
Gestures		
	A	boat
Semantics	start_comm	boat
Gestures		
CADDIAN	A	here
Semantics	start_comm	come_here
Gestures		
CADDIAN	∀	
Semantics	end_comm	

Figure 5.1 – Gestures, CADDIAN symbols and semantics of mission "Carry-to-boat"

6 Conclusions

This deliverable has described the design and implementation of several buddy tasks from the algorithms to the implementation and results.

By the time of writing this document, the tasks were already tested in water separately (except Carry-to-boat) however not as a system. The whole system was tested in simulation in the Software Integration week in May, Zagreb.

Further testing and integration will be carried out as preparation and as part of the trials of October 2016, Biograd Na Moru, Croacia, where the system will be validated by and properly tuned for professional divers.

7 Literature

[1] D. Nad, M. Ribeiro, H. Silva, J. Ribeiro, P. C. Abreu, N. Miskovic and A. Pascoal, "Cooperative Surface/Underwater Navigation for AUV Path Following Missions," in *10th IFAC Conference on Control Applications in Marine Systems (CAMS)*, Trondheim, Norway, 2016.

[2] P. Maurya, A. Aguiar and A. M. Pascoal, "Marine vehicle path following using inner-outer loop control," in *8th IFAC International Conference on Manoeuvring and Control of Marine Craft*, Guaruja-Brazil, 2009.

[3] D. Nad, N. Miskovic and F. Mandic, "Navigation, guidance and control of an overactuated marine surface vehicle," *Annual Reviews in Control,* pp. Vol 40 - 172--181, 2015.

[4] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Transactions on Robotics and Automation,* vol. 17, no. 3, pp. 229-241, June 2001.

[5] G. Grisetti, R. Kummerle, C. Stachniss and W. Burgard, "A Tutorial on Graph-Based SLAM," *IEEE Intelligent Transportation Systems Magazine,* vol. 2, no. 4, pp. 31-43, 2010.

[6] B. Triggs, P. F. McLauchlan, R. I. Hartley and A. W. Fitzgibbon, "Bundle Adjustment — A Modern Synthesis," in *Vision Algorithms: Theory and Practice, Volume 1883 of the series Lecture Notes in Computer Science*, 2002, pp. 298-372.

[7] M. Pfingsthorn and A. Birk, "Generalized graph SLAM: Solving local and global ambiguities through multimodal and hyperedge constraints," *The International Journal of Robotics Research,* vol. 35, no. 6, pp. 601-630, May 2016.

[8] E. Olson and P. Agarwal, "Inference on networks of mixtures for robust robot mapping," *The International Journal of Robotics Research,* vol. 32, pp. 826-840, June 2013.

[9] N. Sünderhauf and P. Protzel, "Switchable constraints for robust pose graph SLAM," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura, 2012.

[10] P. Agarwal, G. D. Tipaldi, L. Spinello and C. Stachniss, "Robust map optimization using dynamic covariance scaling," in *IEEE International Conference on Robotics and Automation, pp. 62-69*, Karlsruhe, 2013.

[11] Đ. Nađ, F. Mandić and N. Mišković, "Diver Tracking Using Path Stabilization - the Virtual Diver Experimental Results," in *Procedings of CAMS2016*, Trondheim, 2016.