Development of SKUBA Autonomous Underwater Vehicle: Obsidian robot

Kanjanapan Sukvichai
Dept. of Electrical Engineering
Kasetsart University, Bangkok
Thailand 10900
Email: fengkpsc@ku.ac.th

Somphop Limsoonthrakul
Dept. of Computer Engineering
Kasetsart University, Bangkok
Thailand 10900
Email: paeyeng@gmail.com

Tachin Srisombat
Dept. of Computer Engineering
Kasetsart University, Bangkok
Thailand 10900
Email: jidrids@gmail.com

***Abstract*—Obsidian is an Autonomous Underwater Vehicle (AUV) developed by SKUBA team at Kasetsart University. Our team consists of graduated and undergraduate students from several departments. In this paper, we present a design of underwater robot which is carefully produced in order to make sure that the robot can deliver a good performance in the competition. We apply snap lock mechanism to in order to create a waterproof area for the control unit. The frame is designed so that it is flexible and adjustable for equipment mounting. The robot is driven by eight thrusters which is attentively mounted at positions so that it can achieve 6 degrees of freedom motion. Many kinds of sensor are adopted to the robot, e.g. an IMU, a barometer, and a side-scan SONAR. Information from sensor is used as perception for the robot. Along with sensors and actuators, the electrical system is design and developed in order to make sure that the robot can operate reliably. We also implement a software system in order to control the robot. Object detection, SLAM, and planning techniques are integrated in the system so that our AUV can perform the tasks and achieve the goal.**

I. Introduction

Autonomous Underwater Vehicle (AUV) is an interesting topic that challenges all researchers in both robot's hardware design and intelligent software development. SKUBA is the autonomous robot team from Kasetsart University, Thailand. Team is consist of students from Department of electrical, computer and mechanic engineering. SKUBA team has the main objective is to design, develop, research and build a kind of an autonomous robot. SKUBA team had joined several robot competitions such as small-size soccer robot and @Home robot competition in RoboCup competitions [1]. This year, team expand our interested to International RoboSub competition which is held in San Diego, CA.

In this paper, we present a design of Obsidian which is our AUV. Mechanical, electrical, and software designs are described in each section to give an overview of our robot hardware and control system.

II. Robot Design

In this section, the mechanical design and electrical components are explained. Design criteria is defined based on RoboSub competition rules. The robot body is separated into two major parts: hull and frame. Both of the parts are made from aluminum 5083 grade which has a high corrosion resistance property. This criteria is concerned when the robot is submersed into the salt water. The concept design of the robot is shown in Fig 1. Obsidian is our robot name.

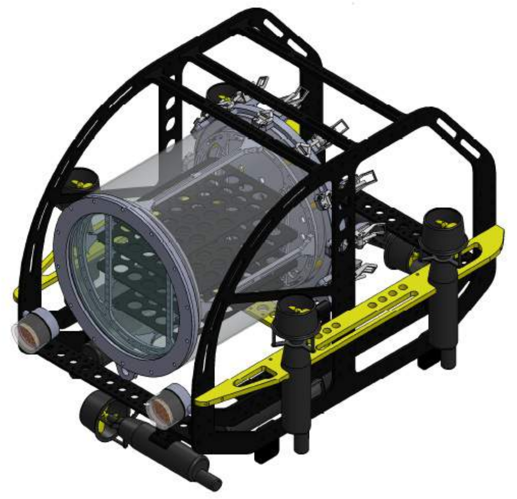

Fig. 1. SolidWorks rendering of Obsidian min robot

A. Robot Frame

Robot frame is made from a 5 mm thick aluminum sheet. The frame has holes in order to reduce robot weight and makes the robot's components, such as sensors and thrusters, removable and relocatable. Two wide angle cameras are attached at the bottom of frame in order to capture the floor images. These images will be used to construct Stereo Vision Simultaneously Localization And Mapping or SV-SLAM which will be used in our navigation system. Two more cameras are attached in side the hull in order to get from view images to do assigned tasks. Sonar and water pressure sensor are also installed at the frame for measuring the depth of the robot. Moreover, the transportation is the big issue for our team, therefore, the frame can be separated and easy to load into the airplane. Robot's frame is shown in Fig 2.

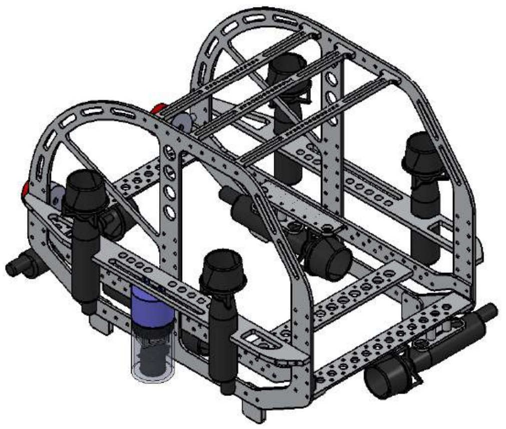

Fig. 2. Robot's frame

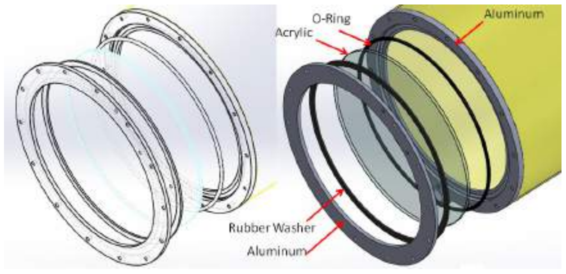

Fig. 3. Front cover structure

B. Robot Hull

The most critical criteria when design an underwater robot is the waterproof system. Robot's hull must be waterproof and passed the IP68 standard. Hull can be separated into three part which are hull tube, electrical rack and back cover. Hull tube is mode from aluminum sheet which is rolled into a cylindrical shape. Front cover of the tube is a stack of waterproof structure which is a stack of aluminum ring, rubber washer, transparent acrylic, o-ring and aluminum base respectively. The cameras will be installed on the back of this stack in order to get the best front view image. Tube's front cover is shown in Fig 3.

Electrical rack is installed inside the hull tube and attached to the back cover. All electrical components such as thruster drivers, power supply unit, batteries and main computer are installed to the rack. The rack consists of 4 layers of aluminum plate and stacked together by side supporters. Back cover is also the critical part of the design because the leakage can happen at this area because the hull tube is a removable part. This back cover must be strong enough to hold the rack and the electrical components. Therefore, aluminum sheet which has 10 mm thickness is selected for the back cover plate. This plate is CNC into specify shape in order to install a waterproof stack and electrical rack.The back cover waterproof system is designed by using the same concept as the front cover. By stacking an o-ring, rubber washer, and aluminum back cover plate respectively, the robot hull is waterproofs and removable. The adjustable snap locks are installed around the tube and back cover in order to make the tight lock between the tube and back cover. Fig 4 shows the complete robot's hull.

The waterproof and snap lock system is shown in Fig 5.

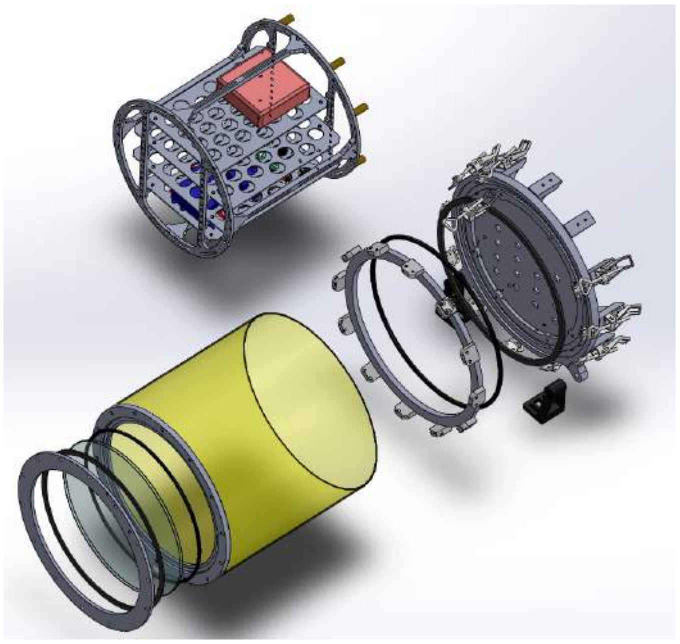

Fig. 4. Robot's hull structure

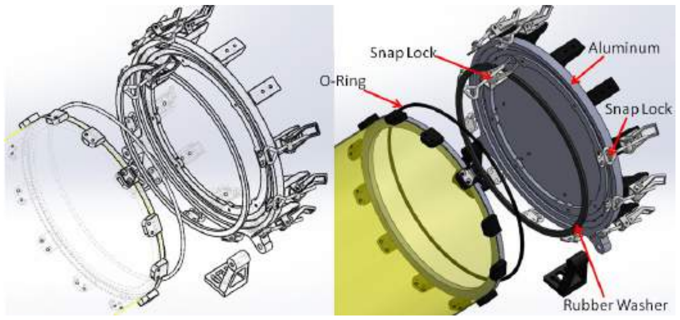

Fig. 5. Back cover structure

After all the hull's parts are manufactured, it is assembled and tested in the pool as shown in Fig 6. All of the seals work properly.

C. Manipulator

Basically, motor is selected as manipulator actuator, but in RoboSub case, pneumatic system is more suitable for an underwater scenario. Due to the fact that the pneumatic system requires less complex waterproof technique, pneumatic manipulator is easier to control the than a pneumatic motor. This year, the gripper is designed instead of a full manipulator due to limitation of the developing time. One pistol pneumatic provides a gripping motion as shown in Fig 7. Gripper is installed at the bottom side of the robot as shown in Fig 8.

Since the Ball-Bullet gun tank is hard to found, therefore, CO_2 tank will be used for our pneumatic system's power source.

D. Thrusters

The 600HF thrusters from CrustCrawler are selected to be the robot's thrusters as shown in Fig 9. the thruster is driven

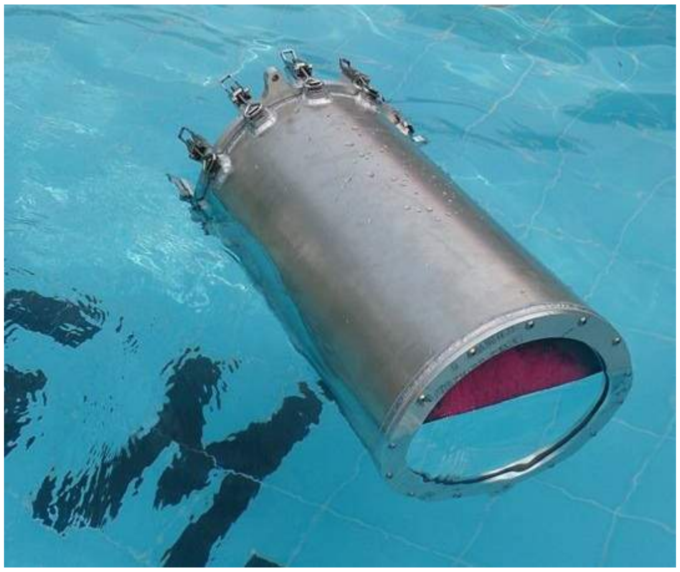
Fig. 6. Waterproof test for robot's hull

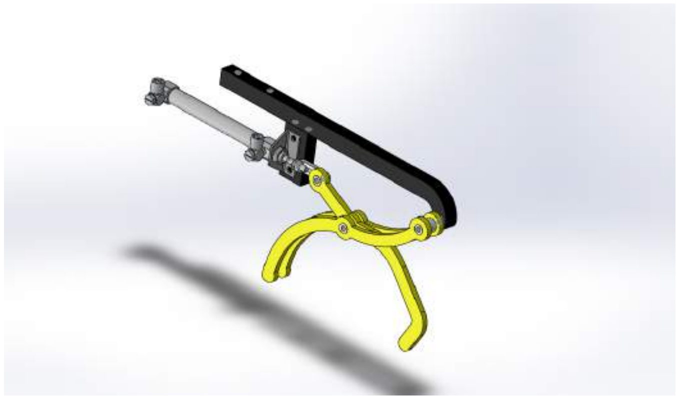
Fig. 7. Robot Gripper

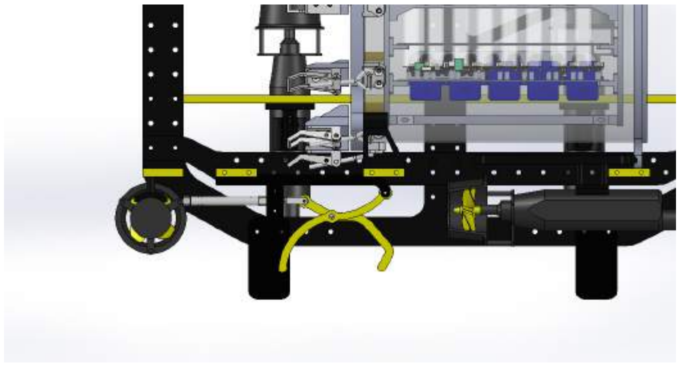
Fig. 8. Mounting Position of the Robot Gripper

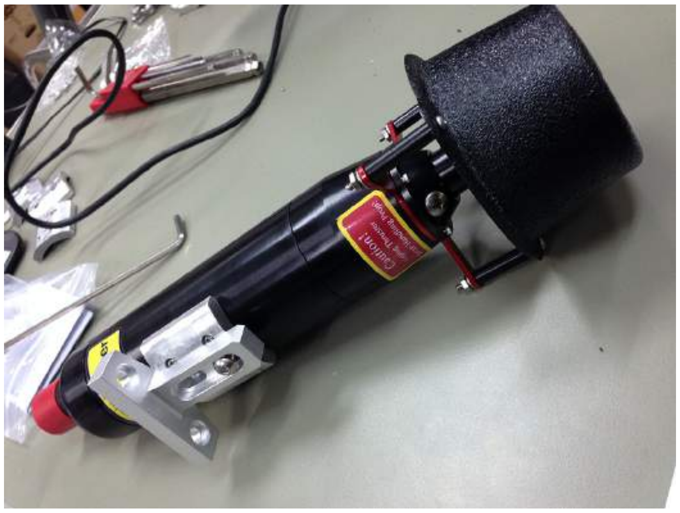
Fig. 9. CrustCrawler 600HF thruster

by powerful 600 Watts brushless DC motor with a 4.28:1 gear ratio which can produce output power up to 140 watts. It can generate maximum thrust of 6.79 kg when operated at 16 Volts and 6 Amps . Seacon ”Wet Mate” male and female connectors are used to connect between thrusters and robot's hull. There are 8 thrusters in the obsidian robot. Four thrusters are used to control robot depth and its stability. Two thrusters are used for forward and backward motion while the last two thrusters are used to move a robot to the left and right.

Robot frame and hull are manufactured and assembled. Thrusters are installed into robot's frame. The total weight is around 43 kg without electrical components. Robot is normally float for safety purpose and required thrusters force in order to drive the robto down into the water. Buoyancy force is calculated in order to find the requirement torque by using equation below. The requirement torque for each thruster is 4.5 kg.

$$thrusterforce = \rho\pi\frac{D^2}{4}Lg - Robotweight \qquad (1)$$

where,

$$\rho = 1029kg/m^3$$

Robot now is tested by adding weight around 10 kg into the frame in order to simulate thrust forces and is dropped into the pool. Fig 10 shows the real robot.

E. Sensors

In this section, robot sensors are explained. High definition cameras are the main sensors and used in many parts in our software such as navigation and object detection modules. Two WideCam F100 wide angle cameras are placed inside a waterproof container with specific length between two cameras. The container is attached to the bottom of robot's frame and placed it face down in order to capture a bottom view which is used in SV-SLAM algorithm and used to obtain orientation of Guide Markers. Distance from the robot to the bottom of the pool is calculated by using stereo vision technique. Moreover, these cameras capture the features of

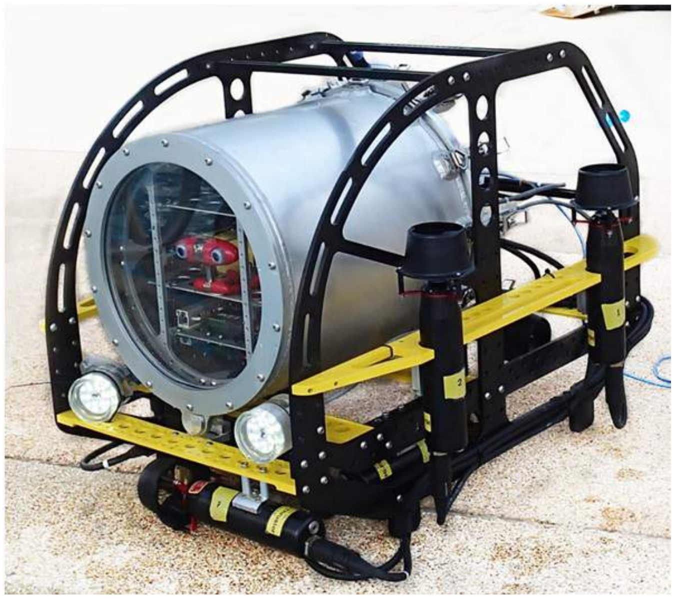
Fig. 10. Obsidian mini

Fig. 11. UM6 Inertia Measurement Unit (IMU)

the floor. Features are used to construct a map. A Logitech C920 cameras are attached to the front of electrical rack behind the transparent acrylic. Front view images are captured and used in order to calculate distance between robot and interested objects. Inertia Measurement Unit (IMU) from CH Robotics is selected for robot motion sensor. This IMU consists of three axes rate gyroscopic sensor, three axes acceleration sensor, and magnetic sensor. Computer unit can communicate to IMU by using RS-232 standard at maximum rate of 112500 baud/sec. Orientation of the robot can be estimated by fusing all measured data using Extended Kalman Filter (EKF). Angle estimates are available as 'Euler Angle' or 'Quaternion Angle' outputs. This sensor is placed at the middle of the robot in order to obtain more precision data. Robot self-stabilizing algorithm uses this estimated angle and rate of change in angle in order to control the robot's stability. UM6 IMU is displayed in Fig 11.

Tritech Micron DST Scanning sonar is installed into the top of robot's frame as shown in Fig 12. It's used for obstacle avoidance and distance estimation. Sonar information is importance when robot operates in turbid water condition. In this bad condition, images from front camera cannot provide accurate distance to interested objects. The Micron scans 360 degrees

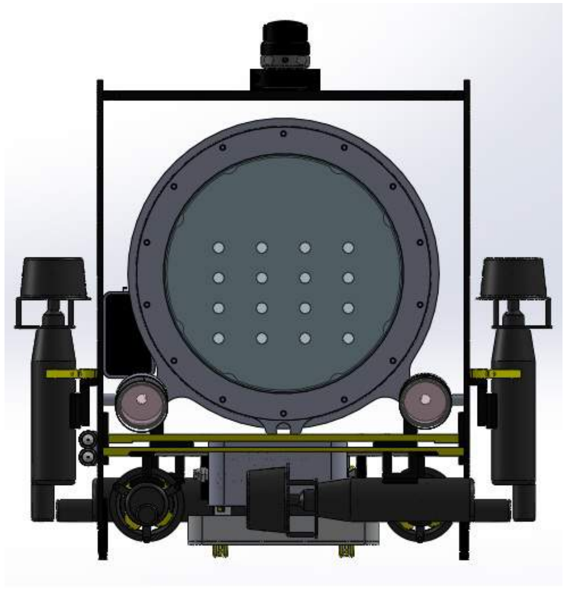
Fig. 12. Micron DST Scaning sonar is installed on top of the robot

Fig. 13. Micron DST Scaning sonar (left) and PA500 Altimeter (right)

with a frequency up to 750kHz. It can scan surrounding environment up to 750 m. PA500 Precision Altimeter sonar is also installed into the robot in order to measure depth of the bottom floor related to the robot. This sensor data will be combined with the stereo vision depth information in order to produce high accurate measurement. Micro DST sonar and PA500 are shown in Fig 13.

F. Main Computer

In order to process images,perform tasks and calculate navigating algorithms, an on-board computer is concerned. It has to have enough calculation power to process images in real-time, and have suitable connection ports to communication with variety of sensors and also consume less energy. Intel based computer is selected for robot's computer since it is flexible and has a environment friendly for both Windows and Linux OS. Next Unit of Computer kit (NUC) is the optimal solution for the robot because it can be customized and has enough connection ports. Dimension of NUC is 116.58 x 112.01 x 34.54 mm including its cover box . NUC in the robot comes with Intel CPU Core i5, 8 GBytes of RAM, 64 GBytes of solid state hard drive, two USB 3.0, one USB 2.0, one mini display port and one mini HDMI. It also has one empty mini PCIe which can be used to add extra card when

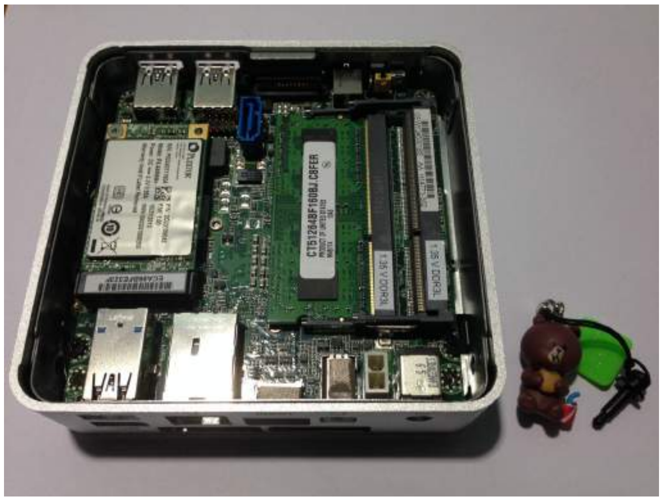

Fig. 14. Next Unit of Computer kit (NUC)

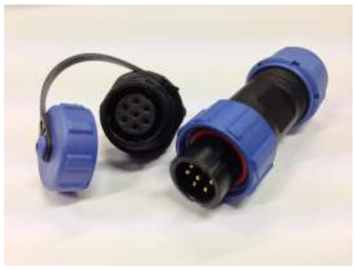

Fig. 15. Waterproof IP68 standard plug

it's required. NUC is shown in Fig 14. Ubuntu is selected as our robot's OS. Robot Operating System (ROS) is installed in order to combine codes from different programing languages easily. The ROS is a set of software libraries and tools that help you build robot applications [2]. From drivers to state-of-the-art algorithms, and with powerful developer tools, ROS is a good solution for robotics project and it's all open source.

G. System Diagram

Robot wiring diagram and waterproof connectors are explained in this topic. IP68 standard waterproof and dust proof plugs are selected as connectors between the hull and outside components such as thrusters and sensors. This connector is made from plastic. It has a thread and o-ring seal to create air tight between female and male connector's head. Spiral rubber band is installed at the end of male head in order to tighten the cable. Fig. 15 shows waterproof plug.

All thrusters will be connected driver boards which are controlled by the embedded controller board. The embedded board will receive commands from NUC by RS485 standard communication. Bottom view cameras are installed inside waterproof chamber and connected directly to NUC via IP68 USB cable. Furthermore, the wireless USB adapter is installed inside the waterproof chamber like the cameras and this adapter is also connected to NUC via IP68 USB cable. Sonar is connected to NUC directly via RS232 standard. Customized waterproof CAT 5 LAN cable is also connected to NUC's LAN port. Inside the hull, the front cameras are connected to NUC via USB port. The IMU is installed into the embedded board

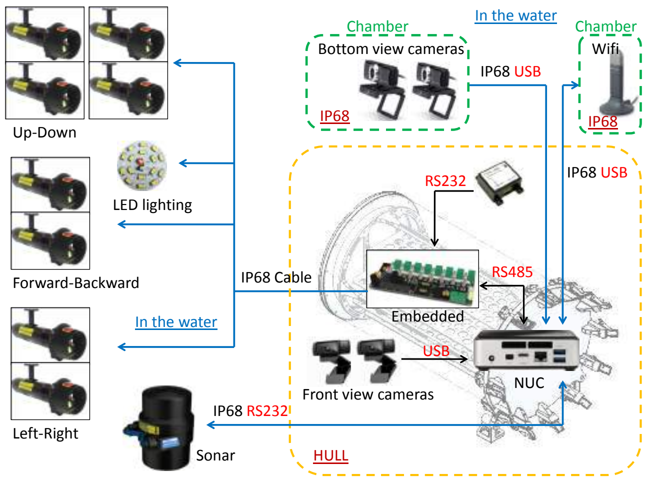

Fig. 16. SKUBA Obsidian robot wiring diagram

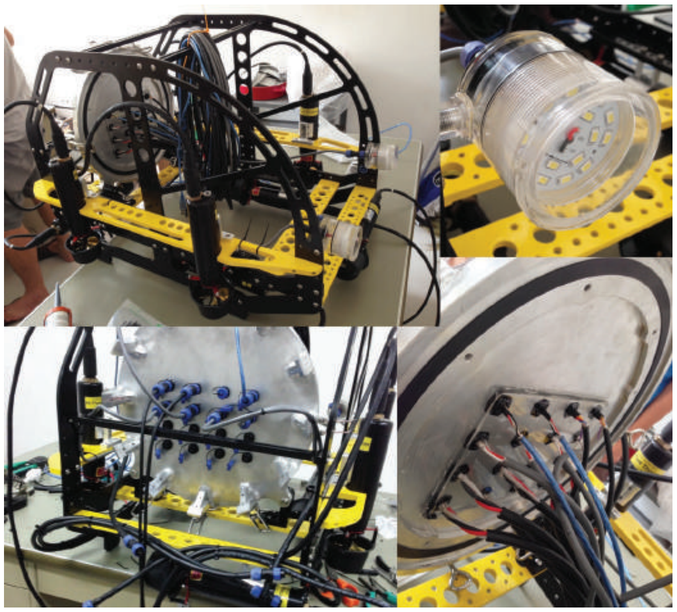

Fig. 17. Robot connectors

and communicated via I2C protocol. All of the wiring diagram is shown in Fig 16. Fig. 17 shows all the cables, connections and the obsidian robot with installed cables following the designed diagram.

After all connectors are installed into the robot, the robot is tested in the acrylic pressured tank which has presure around 1.2 bar which is generated from external air pump as shown in Fig 18.

Robot is submersed in the tank for 6 hours. No leakage is found after the experiment. Then robot is tested and experimented at the pool which has depth of 5 meters as shown in Fig 19.

III. System Design

In this topic, information about robot's software is explained.

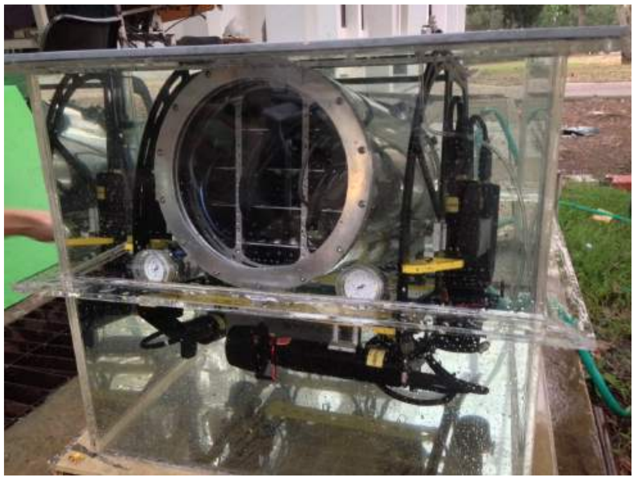
Fig. 18. Robot is tested inside a pressured transparent tank

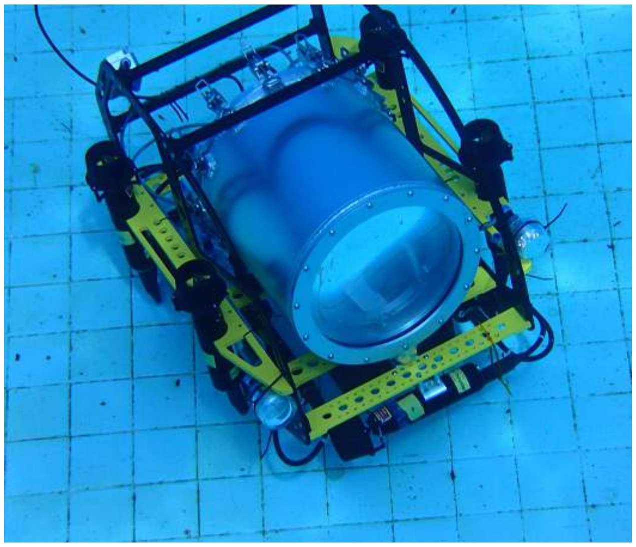
Fig. 19. Robot is tested at 5 meters below the water surface

A. *Vision System*

In order to perform the missions in the competition, Obisidian mini needs to have abilities to detect and identify objects placed in each mission state. Moreover, it also needs to be able to locate itself so that it can navigate itself to the each state and prevent itself from getting lost.

Obsidian mini is equipped with three USB cameras – one for object detection, and two for visual odometry and visual SLAM. The camera for object detection is mounted inside the robot hull and is facing forward to capture the scene in front of it. Other two cameras are mounted outside the hull. They are at the belly of the robot and are facing downward to capture images of underwater ground. The following sections describe the mentioned techniques in detail.

1) Object Detection: Since the shape, size, and color of objects in each task are already given in competition rules, we simply detect and identify those objects in the image captured from front view camera based on such features. The first feature we use to classify pixels of our interest objects out of the captured image is color. Our method uses the HSV color space instead of RGB. The reason is that HSV separates the color information from the image intensity. Considering our situation, the vehicle operates outdoors and the light condition is dynamically changed. In such the situation, RGB tends to fail to represent the true color of the object because all components (red, green, and blue) change according to the light intensity. Unlike RGB, HSV still preserves the color information in Hue component while the Saturation and Value components vary due to the intensity.

We simply use thresholding technique based on both hue and saturation components of HSV to classify interested objects in images. From our experiment, using only hue component is not enough. For extremely high intensity (white) and very low intensity (black) pixels, hue is hard to determined because the difference between maximum and minimum value of color components in RGB tends to zero. Hence, we also use saturation to scope the appropriate range of pixel intensity.

The preliminary experimental result of our color classification method is shown in Fig 20. We detect yellow, red, and green objects in images. The threshold for each object is defined by minimum and maximum values of hue and saturation components. In this experiment, each threshold is tuned manually so that it can cover most pixels of objects.

Since only color detection gives noisy result, we filter out the outliers using shape recognition. The contours that is returned from color detection is filtered by its shape. In case of rectangle shape object, the convex contours which have number of vertices between four and seven are considered to be rectangle. For sphere shape object, the Hough transform algorithm is used to detect circle contours.

We also apply Kalman filter to track the detected shape in order to preserve the smoothness and robustness of the output. The result of object detection and tracking is shown in Fig 21.

B. *Visual Odometry*

During navigation, the vehicle needs to known its movement in order to predict its position and orientation. Hence, in SLAM, we need to construct a motion model of the vehicle so that we can predict the change of the system state. This can be done by using odometer such as wheel encoders or GPS. However, an underwater vehicle is hard to measure such the movement since it moves in water which is unstationary and GPS is unreachable. We propose a vision technique to predict the odometry of an AUV using visual features detected on underwater ground. We assume that the ground is flat and is parallel to the earth surface. The change of features in two image frames is tracked and by transforming image planes to the ground plane, we can track the 3D movement of the vehicle.

1) Optical Flow: The first problem is how to extract features from the image capturing the underwater ground. The good feature in images is the intensity of image pixels. Thus, we would like to find sub-areas which highly change of their intensity. Shi-Tomasi algorithm is the popular methods to detect good features in images. It is based on the Harris detector but the selection criteria are different. The Shi-Tomasi detector considers the threshold on the smallest value of

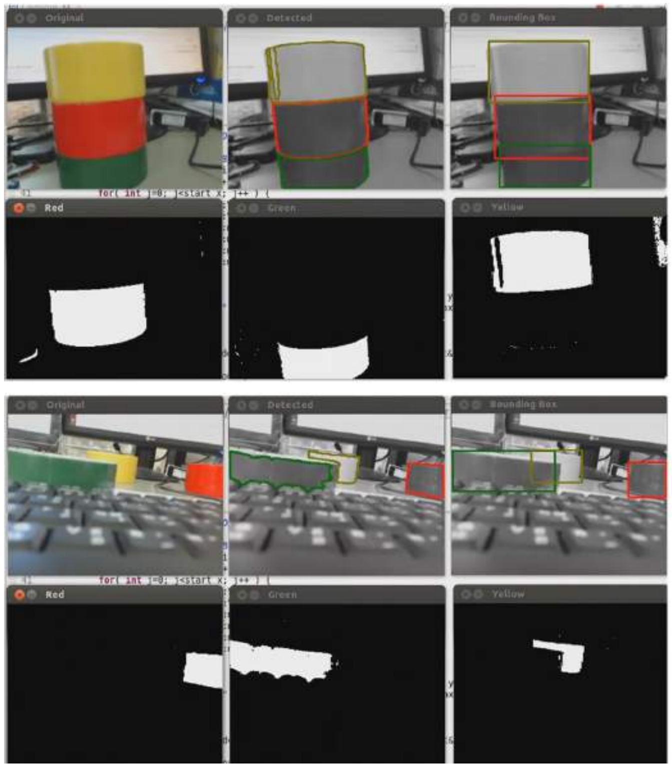

Fig. 20. Three color detection with marked blobs

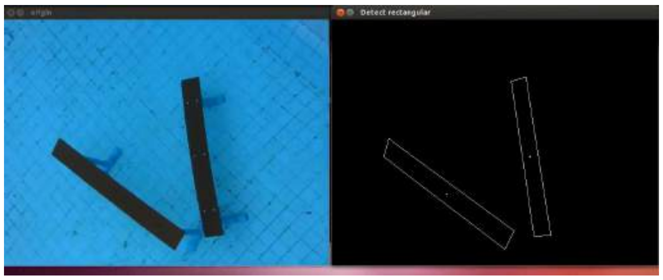

Fig. 21. Rectangular shape detection

two eigenvalues of the gradient matrix so that the feature point with good textureness is guaranteed to be selected. [4] The best feature for tracking have been chosen to satisfy $min(\lambda_1, \lambda_2) > t$ for some threshold t where λ_1 and λ_2 are the eigenvalues of a Harris matrix A.

After finding good feature points in the images, Lucas-and-Kanade [5] method is used to estimated the correspondences of those feature points in two consecutive images so that we can measure the flow of those feature points in the scenes. The method is based on the assumption that the displacement of the feature points between the two images is small. The gradient matrix calculated from the second derivative of the image intensity is used for local search of feature points. The difference of image $F(x)$ and image $G(x)$ is matched by minimising the sum square error function

$$e = \sum \left[F(\vec{x} + \vec{d}) - G(\vec{x})\right]^2 \tag{2}$$

where $\vec{d}$ is the flow vector indicating the displacements of tracked feature. By equating the first derivative of the error function to zero and using the first order Taylor approximation to yield spatial gradient

$$F(\vec{x} + \vec{d}) \approx F(\vec{x}) + \vec{d}\left(\frac{\partial}{\partial \vec{x}} F(\vec{x})^\top\right), \tag{3}$$

it leads to a linear system which can be solved for $\vec{d}$.

2) Image Plane to Ground Plane Transformation: After we get the motion estimated from optical flow, the next problem is how to estimate the actual displacement of the vehicle based on the given displacement of feature points in the image. In this paper, we put an assumption that the underwater ground is flat. Hence, the odometry data can be determined by using invert projection of the flow in the images to the underwater ground.

The camera is mounted at the bottom of the vehicle and points downward. The captured image is a projection of the ground plane (underwater ground) to the image plane. The projection of one plane to another plane can be described by a transformation matrix called Homography. Therfore, we can reverse the observed motion of feature points in images to actual motion on the ground plane using inverse of Homography.

A Homography representing the projection of a point from the ground plane to the image plane is derived from the pinhole camera matrix which is described as

$$\mathtt{P} = \mathtt{K}[\mathtt{R} \mid \vec{t}] \tag{4}$$

where $\mathtt{R}$ is 3×3 rotation matrix, $\vec{t}$ is 3×1 vector representing translation of the world coordinate to image coordinate, and $\mathtt{K}$ is the camera calibration matrix which represents the camera's intrinsic parameters : focal length (f), and the principle point $(p_x, p_y)^T$, as

$$\mathtt{K} = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix}. \tag{5}$$

To find the Homography transforming image plane to ground plane, we need to find the inverse of camera matrix $\mathtt{P}$. However, the camera matrix is not invertible (it is not a square matrix) so a constraint is added into the system and the matrix $\mathtt{P}$ is modified.

The original camera matrix $\mathtt{P}$ describes a transformation of a 3D point in real world to 2D point on image plane. Hence, the information of depth is lost and the inverse transformation is impossible. Nevertheless, if the depth information (h) is known, the camera matrix $\mathtt{P}$ can be modified to Homography $\mathtt{H}$ as follows:

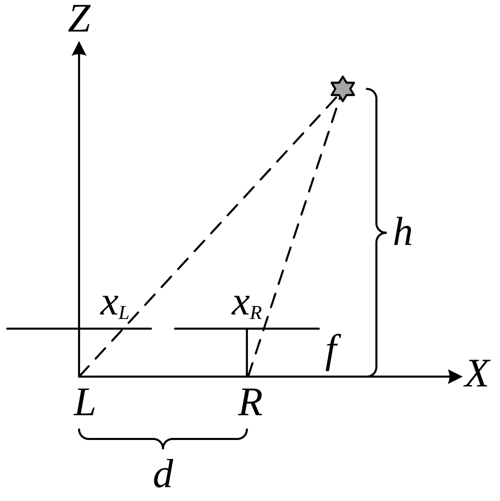

Fig. 22. Geometry for parallel cameras.

$$
\begin{aligned}
\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} &= \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ h \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} p_{11} & p_{12} & p_{13}h + p_{14} \\ p_{21} & p_{22} & p_{23}h + p_{24} \\ p_{31} & p_{32} & p_{33}h + p_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \\
&= \mathtt{H} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}.
\end{aligned}
$$

$\mathtt{H}$ is now a square matrix and we can find the invert projection from image plane to ground plane by finding inverse of $\mathtt{H}$ while the camera intrinsic parameters ($\mathtt{K}$) is determined beforehand, the world coordinate is set at the image center ($\vec{t} = [0\ 0\ 0]^\top$), and the rotation matrix ($\mathtt{R}$) and the distance from the camera to the underwater ground (h) is calculated online.

By assuming that the underwater ground is flat and is always parallel to the world's horizontal plane, the orientation of the camera measured from an inertial measurement unit (IMU) can be used to define $\mathtt{R}$ directly. However, to find h, we use stereo vision technique to estimate the distance between the camera and the ground.

With stereo vision, we can reconstruct the 3D position of a point so we can obtain the relative depth of the object from the camera. To make it simplest, two cameras with the same focal length (f) are installed so that their optical axes are parallel. We need to match the correspondences between two images and then apply with the simple triangulation method to solve for depth information (h). From the geometry of two camera shown in Fig 22., x_L and x_R denote correspondences of the projection of the 3D point on left camera L and right camera R accordingly. Once the relative distance between two cameras (d) is known, the depth can be determined from:

$$h = \frac{d\,f}{x_L - x_R}. \tag{6}$$

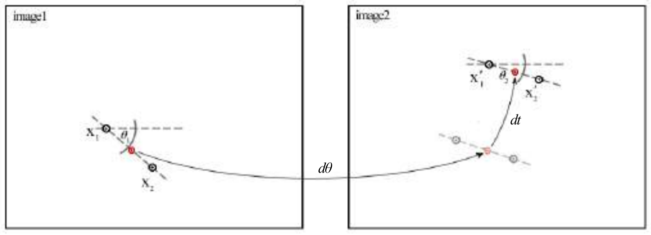

Fig. 23. Finding transformation scheme.

In our work, a feature point detected in pervious section that is closest to the image center is selected to measure h.

C. RANSAC

From the set of correspondences in image pairs, we can transform them to ground plane by using Homography as described in previous section. The next step is to estimate the motion model (translation and rotation) from those correspondences on the ground plane. RANSAC (RANdom SAmple Consensus) is used to predict the model and remove any outliers in the correspondence set.

Before applying RANSAC to the correspondences, some outliers can be eliminated by considering the direction of the motion determined from the acceleration vector from IMU. Thus, any features which do not move along the direction of vehicle movement are considered to be outliers and are removed from the correspondence set.

To find the motion model, two correspondences are selected from coherent images. Then, the angle of the line that pass through those two points are calculated. Moreover, a middle point between those two points are calculated to represent the position of points in each image. From those pre-calculated data (the lines and the middle points), we can find the difference of vehicle's heading ($\Delta\theta$) from the difference of angle of two lines in the first and second images. The translation (Δx and Δy) can be calculated from the difference of position of two middle points in both images. Fig 23. shows the illustration of the described scheme.

Now, we can apply the motion estimation cheme to RANSAC to find the best motion model. The RANSAC algorithm used in this special study is shown in Algorithm 1. The number of loop in RANSAC is dynamically chosen in runtime to ensure that we samples the correspondences without outliers at least one time. The method is based on the probability that the chosen sample is inlier as shown in [6]. After finish the RANSAC, we get a motion estimation (odometry) which can be represented by a Gaussian density with mean $\hat{\mathrm{X}}$ and covariance Σ;

D. Visual SLAM

1) System State: Based on EKF, the state of the vehicle and all landmarks are defined by a probabilistic model. The state is initialized and its uncertainty is assumed to be Gaussian. The state is updated during vehicle motion by using the prediction of the motion and the observation of the landmarks which are also applied with a Gaussian noise assumption.

Algorithm 1 RANSAC

```
N=$\infty$, count=0.
while N<count do
    Randomly choose two correspondences from image1 and image2.
    Estimate the model (translation and rotation) as described.
    Caluculate models' error and the number of inliers and outliers.
    if $outliers < threshold$ then
        Add the model to $model\_list$.
        Find the best model ($\hat{X}$) which give the lowest error.
    end if
    Set $\epsilon = 1 - \mathrm{numberofinlierts/totalpoints}$.
    Set $\mathrm{N} = \log(1-p)/\log(1-(1-\epsilon)^s)$ with $p$=0.9.
    Increment count by 1.
end while
Calculate covariance ($\Sigma$) from $model\_list$.
```

The system state is represented by a state vector ($\vec{x}$) which is composed of the estimates of the vehicle ($\vec{x}_v$) and landmarks ($\vec{y}_i$). The uncertainty is assumed to be a normal distribution with zero means and covariance matrix ($\mathtt{P}$) as follows:

$$\vec{x} = \begin{pmatrix} \vec{x}_v \\ \vec{y}_1 \\ \vec{y}_2 \\ \vdots \end{pmatrix}, \mathtt{P} = \begin{bmatrix} \mathtt{P}_{\vec{x}_v\vec{x}_v} & \mathtt{P}_{\vec{x}_v\vec{y}_1} & \mathtt{P}_{\vec{x}_v\vec{y}_2} & \cdots \\ \mathtt{P}_{\vec{y}_1\vec{x}_v} & \mathtt{P}_{\vec{y}_1\vec{y}_1} & \mathtt{P}_{\vec{y}_1\vec{y}_2} & \cdots \\ \mathtt{P}_{\vec{y}_2\vec{x}_v} & \mathtt{P}_{\vec{y}_2\vec{y}_1} & \mathtt{P}_{\vec{y}_2\vec{y}_2} & \cdots \\ \vdots & \vdots & \vdots & \end{bmatrix} \tag{7}$$

We define two coordinate frames in this system: a fixed world frame W which is assumed to be flat and non-rotating, and vehicle frame V which is set at the center of mass of the vehicle and is rotated respect to the orientation of the vehicle.

The vehicle's state vector ($\vec{x}_v$) consists of a 3D position vector ($\vec{s}^W$), orientation defined by Euler angle vector ($\vec{r}^W$), translational velocity vector ($\vec{v}^W$), and angular velocity vector (ω^V) respect to world frame W and vehicle frame V. The ith landmark's state vector is a 3D vector representing a 2D position of the ith visual landmark on the underwater ground (the third element of the 3D vector is always 0). The representation of system state in the SLAM is:

$$\vec{x}_v = \begin{bmatrix} \vec{s}^W \\ \vec{r}^W \\ \vec{v}^W \\ \omega^V \end{bmatrix}, \vec{y}_i = \begin{bmatrix} y_{1,i} \\ y_{2,i} \\ 0 \end{bmatrix}. \tag{8}$$

2) Visual Landmarks: Our SLAM is based on the features detected on the underwater ground. The visual landmarks should be identical and distinguish from the area around them. We use Shi-Tomasi corner detector to relatively large pixel patches (15×15). As it is shown in [7], large image patches are more unique compared with small patches so they are better to use as long-term landmarks. To store a landmark in the map, we first transform an image patch to be top view image patch by using current vehicle's state to create plane-to-plane transformation matrix from image plane to underwater ground plane. This will resolve perspective distortion which can solve the situation that the same landmark is observed from different angle.

In measurement step of the EKF SLAM, feature matching needs to be performed in order to match the saved landmarks with the current detected features. The regions of interested in the left and right images are generated base on the current state of the vehicle and the uncertainty (covariance matrix) of the saved landmarks. Any detected feature lying within the region of interested will be compared and matched with the image patch of the landmark. For those features which are not in any region of interest will be considered as new landmarks.

As the vehicle navigate during the mission, more and more landmarks will be added into the map and this will increase the computation time. To prevent this problem, we need to cut some landmarks out of the map. There are two criteria that we use to eliminate landmarks. First, the newly introduced landmark will be deleted if there is no more matched features in two consecutive samples. This will guarantee that only good landmarks will be kept and it decreases the number of landmarks introduced to the map. Second, the number of landmarks in the map is limited and if the number is exceeded, the old landmarks in the map will be removed to make a space for new landmarks. We define grids on the map and if the number of landmarks in a grid is more than two, we select one from the grid with highest uncertainty to eliminate. With this method, we can maintain the sparseness of the landmarks.

3) Prediction Model: In our EKF approach, the prediction of the system state is based on IMU and our proposed visual odometry, While IMU gives the rate of change of vehicle's attitude, the visual odometry provides the rate of change of vechile's position. Hence, we can define prediction model of the system at time $k+1$ as:

$$\hat{\vec{x}}_{v(k+1|k)} = \hat{\vec{f}}_v\left(\hat{\vec{x}}_{v(k|k)}, \vec{u}_k\right) \tag{9}$$

$$\hat{\vec{y}}_{i(k+1|k)} = \hat{\vec{y}}_{i(k|k)}, \forall i \tag{10}$$

$$\mathtt{P}_{(k+1|k)} = \frac{\partial \vec{f}}{\partial \vec{x}} \mathtt{P}_{(k|k)} \frac{\partial \vec{f}}{\partial \vec{x}}^{\top} + \mathtt{Q}_k, \tag{11}$$

where $\vec{f}_v$ is a function describing the motion model of the vehicle, and $\vec{u}$ is control inputs. $\mathtt{Q}_k$ is the process noise determined from

$$\mathtt{Q}_k = \frac{\partial \vec{f}}{\partial \vec{x}} \mathtt{U} \frac{\partial \vec{f}}{\partial \vec{x}}^{\top}, \tag{12}$$

where $\mathtt{U}$ is the diagonal covariance matrix of the control input $\vec{u}$.

E. Measurement Model

In this paper, the measurement model is not defined yet. However, in general, the measurement model is defined as

$$\vec{z}_{k+1} = \vec{h}(\hat{\vec{x}}_{(k+1|k)}) + \vec{v}_{k+1}. \tag{13}$$

where $\vec{v}_k$ is measurement noise with mean of $\vec{0}$ and variance defined by covariance matrix $\mathtt{R}$ which is often defined as a diagonal matrix.

F. System Updating

Once we have defined all prediction and measurement models, the Kalman gain $\mathtt{W}$ can be calculated as

$$\mathtt{W}_{k+1} = \mathtt{P}_{(k+1|k)}\mathtt{H}_{k+1}^{\top}\left(\mathtt{H}_{k+1}\mathtt{P}_{(k+1|k)}\mathtt{H}_{k+1}^{\top} + \mathtt{R}_{k+1}\right)^{-1} \quad (14)$$

and the system state and its covariance matrix can be updated as

$$\hat{\vec{x}}_{(k+1|k+1)} = \hat{\vec{x}}_{(k+1|k)} + \mathtt{W}\left(\vec{z}_{k+1} - \vec{h}(\hat{\vec{x}}_{(k+1|k)})\right) \quad (15)$$

$$\mathtt{P}_{(k+1|k+1)} = \left(\mathtt{I} - \mathtt{W}_{k+1}\mathtt{H}_{k+1}\right)\mathtt{P}_{(k+1|k)} \quad (16)$$

where $\mathtt{H}_{k+1} = \left.\frac{\partial\vec{h}}{\partial\vec{x}}\right|_{\hat{\vec{x}}_{(k+1|k)}}$.

IV. Control Software Design and Implementation

In this section we describe the system architecture and the implementation of a basic control system for Obsidian. We are concerned about architectures quality attributes such as extensibility, testability, and modifiability.

A. Software Architecture

The software system is designed base on modular architecture so that the software team can separate each functionality of the robot into independent modules. This gives us abilities to concentrate on individual functionality and makes the software easier to test. We use Robot Operating System (ROS) as a framework and software tools in our implementation. ROS allows us to build software modules (nodes) which can operate with others using publish-subscribe messaging and service. With these communication mechanisms, we can decrease dependencies between modules to a bare minimum.

Fig 24 shows the software architecture of Obsidian's control system. Each module is designed so that it serve only one function and the software logic is encapsulated within itself. Our software team can implement each module in parallel so it can speed up our implementation time.

B. Control Design

Our design of Obsidian's control system follows the concept of sense-plan-act. We use ROS driver nodes and also implement our own nodes to retrieve data from sensors. The actuator module is a simple module which communicate with on-board controller in order to actuate the thrusters. For planning part, there are five modules working together: Object Detection, SLAM, Planning, Robot Pose, and Control.

1) Object Detection: This module uses images from front camera and side-scan SONAR to detect objects in the arena. As describe in III-A1, color detection and shape recognition techniques are done to estimate the position of objects. The pose of objects is recursively evaluated using Kalman filter. We use data from SONAR as measurement.

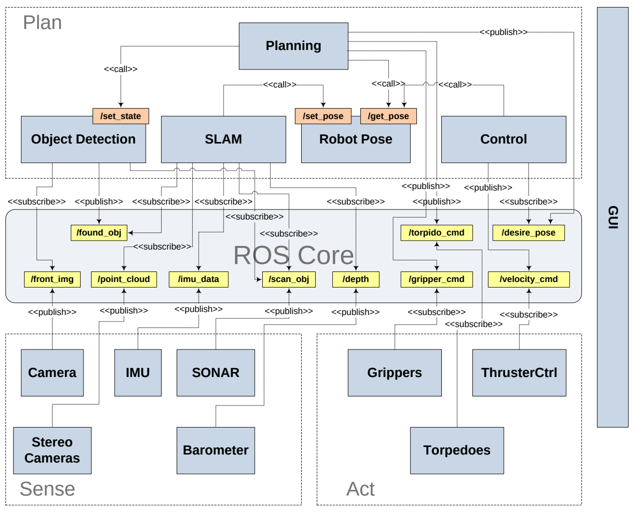

Fig. 24. Obsidian Software Architecture.

2) SLAM: We use bottom-facing stereo cameras to track feature points on the pool bottom in order to generate visual odometry. The landmarks is generated from tracked feature points and also the detected objects from object detection module. We implement our SLAM base on EKF.

3) Planning: The system repeatedly updates the state of competition tasks that have been reached and decided which to pursue next. This module works as the director to synchronize the state for other modules and also command actuators to operate as specified in each task.

4) Robot Pose: This module is responsible for keeping the robot's position and orientation. The data is set by SLAM module and is used by Planning and Control modules.

5) Control: In navigation, the control node transform the desired trajectory from planning into translational and angular velocity commands, and convert to thrusters' speed command. The control method is a simple PD control.

V. Simulation

A robot should be tested in the real environment but it is difficult to build the whole competition site at the university. Simulation is the best way to create the real environment for robot. Navigation and other essential logic modules can be deployed and conduced the experiments in the simulated environment. We expand the UWSIM [3] simulator in order to create an underwater environment. The simulator consists of two major simulation modules; terrain simulation and robot simulation modules. Terrain simulation handles the behavior of designed environment, sensors, cameras, and physic behavior of interested objects that floating on/in the water. Model of the objects and testing environment must be provided to the simulator. Robot simulation handles the physic behavior of the robot such as thrusters force, mass, inertia, buoyancy force and etc. In the simulator, two cameras are added to our robot model which are the front and bottom cameras. By this features, image processing can be simulated. The simulation is shown in Fig 25.

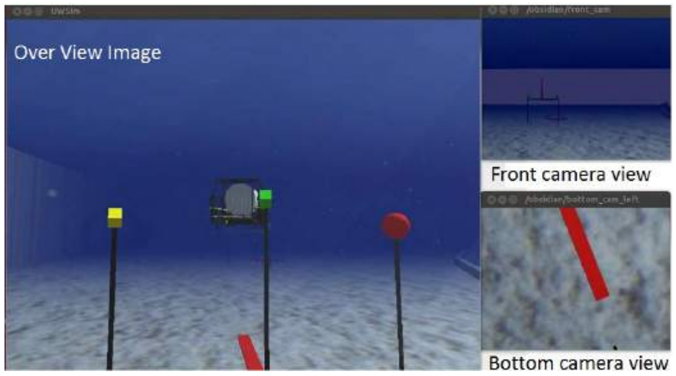

Fig. 25. Simualation of Obsidian Robot

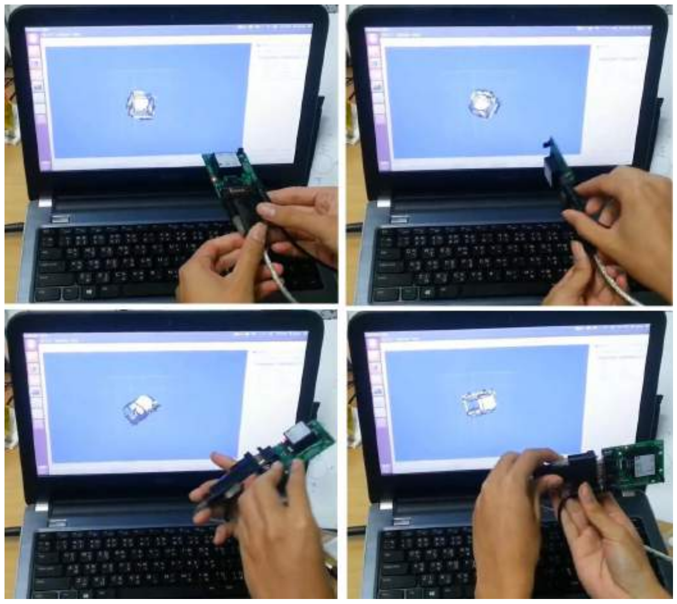

Fig. 26. Monitoring Robot Orientation using IMU

VI. Monitoring System

It is difficult to see robot's behavior when it operates under the water. Monitoring system is developed in order to obtain robot information such as odometry, heading, speed, and depth. Information from sensors are measured by using sensor module in the robot software. IMU data will displayed current robot heading and orientation as shown in Fig 26. Underwater monitoring cameras is also developed and shown in Fig 27 in order the obtain the over view image of the robot and environment in real-time.This monitor system is necessary for debugging the software since it is difficult to program robot intelligent if the programmer cannot see the robot behavior in real-time.

VII. Experiments

In this section, all experiments are explained. The robot body is fully assembled, all thrusters and sensors are installed. The experiments are setup beside the swimming pool which has maximum depth of 5 meters. At setup phase, robot hull is remove and NUC is started. All necessary wireless/wired communication between NUC and monitor notebook is setup and tested as shown in Fig 26. Hull is installed and the obsidian robot is dropped into the water. Thrusters are controlled and all of them work properly. The robot thrusters have enough power to pull the robot down into the water. Left, right, forward, and backward motion can be done by thrusters. Robot motion is tested as shown in Fig 29. Experimental video is recorded by Logitech webcam inside the hull via NUC. The captured images are shown in Fig 30. Video is clear enough to be processed in the image processing module.

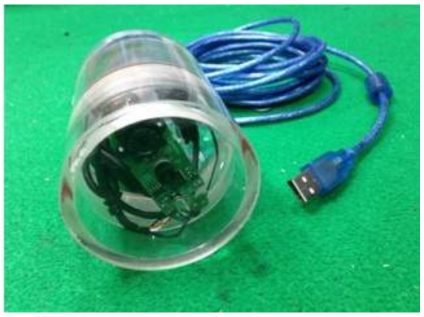

Fig. 27. Under water monitoring Camera

Fig. 28. Robot is setup and tested at the pool

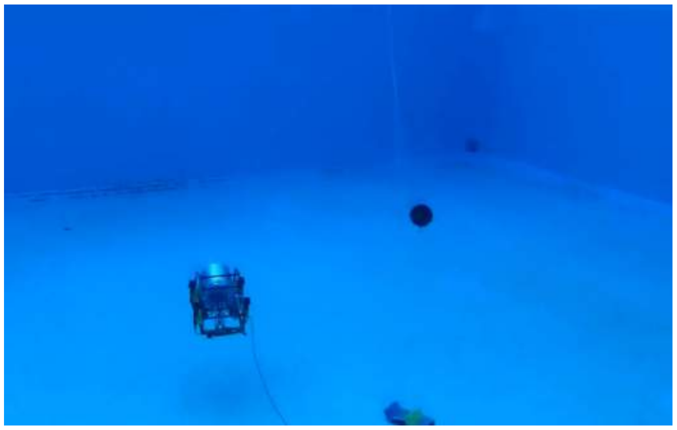

Fig. 29. Robot moves under water

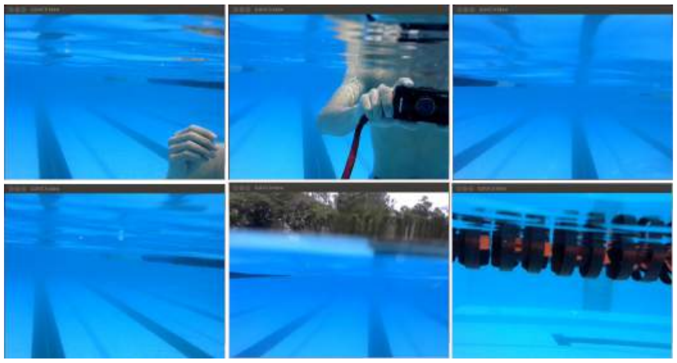

Fig. 30. Captured images from camera inside robot's hull

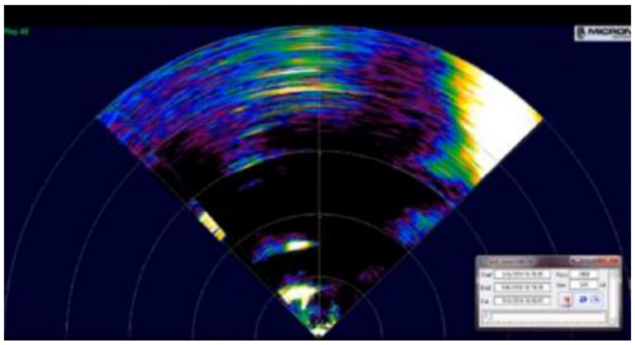

Fig. 31. Sonar Image

Then, the competition environment is setup. The orange rectangular bars are dropped into the pool bottom. These two bar will represent guild line from the current state the next state like in the competition. Image processing algorithm can detect the rectangular shape as show in Fig 32.

Micro sonar from Tritech is test at the pool. First, the sonar information is collected by Tritech software. Sonar is communicated via RS-485 protocol. Data from sonar will be fused with front camera information in order to predict the distance between robot and objects. Data is also used in obstacle avoidance module when front camera cannot capture enough information when it is operated in a bad water condition. Color detection algorithm is applied on the captured video and the result is shown in Fig 32. Red and green kickboards are detected easily when they are close to the surface. Color of the objects are dramatically changed especially red color as shown in Fig 33. This situation will create trouble for the color detection module. The adaptive camera parameters adjustment algorithm is developed in order to overcome this issue.

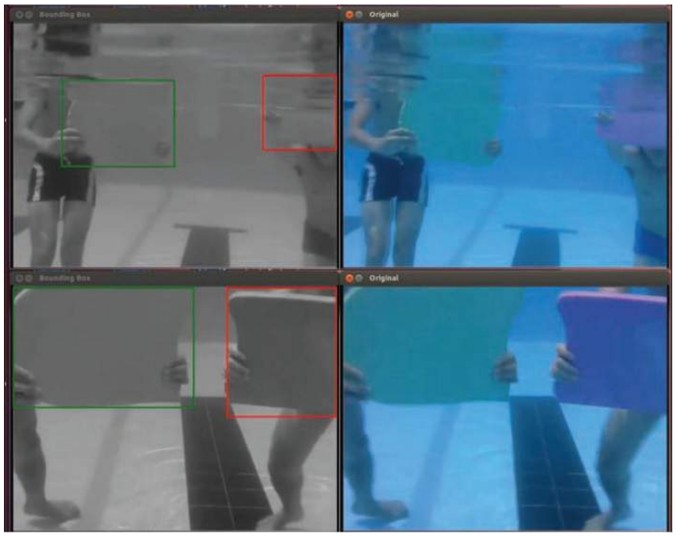

Fig. 32. Green and Red color detection experiment

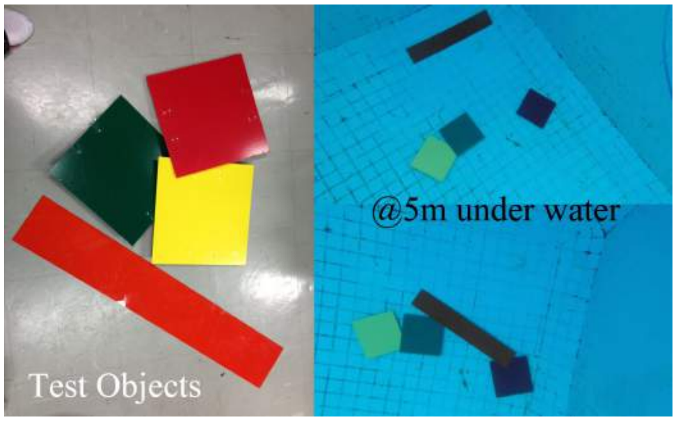

Fig. 33. Comparison of objects at the normal environment and at the pool bottom

VIII. Conclusion

Autonomous Underwater Vehicle, the Obsidian, is designed and developed in order to join the RoboSub 2014 competition. The robot meet the required competition criteria in both size and weight. Robot is tested for waterproof. ROS is deployed into NUC and run on Ubuntu OS. Software is developed and tested in both in simulator and in the real environment.

Acknowledgment

SKUBA team would like to thank PTTEP for technical support and financial sponsorship.

References

[1] http://www.auvsifoundation.org/foundation/competitions/robosub/

[2] http://www.ros.org/

[3] Prats, M.; Perez, J.; Fernandez, J.J.; Sanz, P.J., "An open source tool for simulation and supervision of underwater intervention missions", 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2577-2582, 7-12 Oct. 2012

[4] Shi, J.; and Tomasi, C., "Good features to track", "Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on", 1993, 593–600

[5] Lucas, B.D.; Kanade, T., "An iterative image registration technique with an application to stereo vision", "International joint conference on artificial intelligence", 674–679, 1981

[6] Hartley, R.; Zisserman, A., "Multiple view geometry", 6, 2000, Cambridge university press

[7] Davison, A.J.; Murray, D.W., Pattern Analysis and Machine Intelligence, IEEE Transactions on, Simultaneous localization and map-building using active vision, 2002, 24, 7, 865-880