Towards A Human-Robot Teaming System for Exploration of Environment

Fan Bu* and Yan Wu†
* Hwachong Institution, Singapore
E-mail: fan_bu@163.com
† A*STAR Institute for Infocomm Research, Singapore
E-mail: wuy@i2r.a-star.edu.sg

***Abstract*—The use of robots has been proposed as promising human assistants in accomplishing tasks that are otherwise too tough or dangerous to be executed by human alone. As robots in different forms have their differentiating strengths, a system that uses a combination of these robots can be more robust and resilient in achieving its mission. For example, UAVs can be used to quickly explore an unknown environment while ground robots can be used to navigate through the environment, clearing obstacles and potential dangers for human to enter. In this work, we design a unifying framework to allow human to control and accomplish an exploration mission using heterogeneous robot types. A prototype system is implemented using the Google Tango to control an UAV and a wheeled robot for the human to perform environment mapping. Preliminary experiments have been carried out to study the feasibility of the system.**

I. Introduction

Robots have been used as human assistants in applications across different domains. Many of these applications focus on the use of a single type of robot to accomplish a particular task, e.g. transportation of goods, manipulation, social interaction and so on. As robots in different forms have different strengths and weaknesses, a combination of these robot types together with a human in the loop could open up a greater range of tasks robots could do for humans. For instance, an aerial robot is capable of exploring an unknown terrain within a relatively short duration, however, it is relatively difficult to detect threats hidden in the environment. It is the completely opposite for a ground robot in this particular scenario. Adding a human into the system will also accelerate the exploration process and zoom into the targets of interest as the human can provide expert knowledges which could be otherwise hard to transfer to robot systems.

In this project, we focus on the prototype design of a human-robot teaming system via an interactive interface. This system will allow users to control different types of robots seamlessly on a single application to perform terrain exploration. We implement this system using the Google Tango Tablet as the interactive command interface with two types of robots - a UAV and a mobile robot. This allows the human to dispatch an UAV to perform a fast environment search/mapping task before deploying a ground robot to clear the passage for the human to move in. The system is design to have two independent modules - a motion tracking and control module and an image processing and mapping module. We conduct a number of experiments to test and benchmark the system modules including a human preference on a UAV control mode. These preliminary experimental results confirm that our implementation is positive.

The rest of this paper is organised as follows: We will present some brief related work in Section II before introducing the overall system architecture in Section III followed by the detailed implementation of modules. Experiments with results are described and discussed in Section VII before conclusion for future work.

II. Related Work

Teleoperative control interface for remote robots have been researched and discussed in many bodies of work [1], [2], [3]. However, there is limited work on actual implementation of a system to allow a user to explore an environment without the need of introducing a global view camera [2].

There are several control modes that have already been invented and implemented for the AR Drone 2.0. The most well known modes are from the official app [1] by Parrot Company running on a generic iOS or Android device. The app provides a normal control mode with virtual joysticks as well as a creative mode where the user can tilt his/her device to control the drone. Application has alsobeen designed to work with a game controller such as the PS3s *dron-ps3-controller* [2]. However, to our knowledge, there is no study on the design of a seamless control mode that enable users to control UAVs, such as use of Goolge Tango, a tablet device with motion tracking capability to perform control. The use of Project Tango on robots has also limited literature. The most common research is on autonomous flight, with a Tango device mounted on the drone [4]. This is not possible with AR Drone 2.0 as payload is the primary concern.

To enable UAVs to perform aerial mapping, panoramic stitching is key. However, development work on mapping with the AR Drone is very limited. Most work uses a 3rd party stitching algorithms [5]. However, many of these algorithms are computationally expensive and we have not notice any proposed system architecture allowing toggling for computations to be carried out aboard or externally.

[1] https://play.google.com/store/apps/details?id=com.parrot.freeflight&hl=en
[2] https://www.npmjs.com/package/drone-ps3-controller

III. Overall Architecture

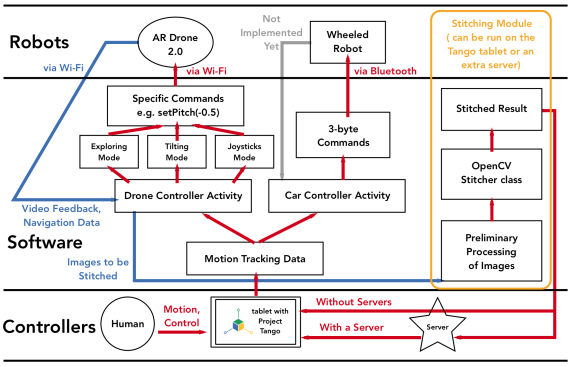


Fig. 1. The overall architecture of the implemented system

Fig. 1 illustrates the overall architecture of the implemented system with two different types of robots for an environment exploration mission. The system is divided into two main modules, the motion tracking and control module (MTC) and the image processing and mapping module (IPM). The MTC module implements specific control drivers for each robot type and integrates the control function into one unifying control interface - in this case, the Google Tango Tablet, for seamless control experience. The captured data from the robot will then be sent to the IPM module for mapping. As having external server may not be an option, the system allows processing both with and without a server.

IV. Drone MTC Module

Using the API [3] of Project Tango, the motion tracking data could be extracted at the frequency of $100Hz$ from the device. This 6-axis raw data from Tango is then treated using a low-pass filter [6] to reduce noise level as well as sudden fluctuations according to (1):

$$y_i = \alpha x_i + (1 - \alpha) y_{i-1} \tag{1}$$

where x_i is the ith-new data and y_i is the ith-smoothed data. α is the smoothing factor that can be determined empirically. A smaller α gives less effect in attenuating high-frequency noises and a larger α creates a delay in recognition of motion. The value of $\alpha = 0.1$ is empirically determined to be the optimal value for the Tango output using data-driven approach.

Typically, a UAV API provides an control interface for gaz, pitch, roll and yaw. For AR Drone 2.0, it also provides several higher level functions, such as keep stationary and blending of motion primitives.

Three control modes have been designed and implemented in this work using the tracked motion data from the Tango Tablet as inputs:

[3]https://github.com/SUPENTA/ardrone-sdk-android

A. Joysticks Mode

Joysticks Mode is the most traditional way to control a drone. We integrated the joysticks mode provided by AR Drone 2.0's official App into our system. Two control screen-based joysticks are shown. One controls the yaw and altitude while the other controls the pitch and roll of the drone. We use a simple linear interpolation method to set the values according to the displacements the fingers are offset from the joystick centres. This mode provides direct control of the drone parameters, however, it is not very intuitive or user-friendly. Training is required for a user to adapt to this control mode.

B. Tilting Mode

We borrow a classic concept in video games for controlling cars and drones to map the motion tracking signal of the Tango Tablet to the AR Drone - the tilting mode. The user simply tilts the Tango Tablet to move the drone towards the direction of desire. Although the official App provides a basic tilting control interface, several optimisations have been done to improve the user experience:

First, a minimum tilt threshold to activate the tilting control is implemented. As human hands are neither accurate nor precise, holding an external device at a particular pose, such as 0 pitch and roll, is an extremely difficult task for any brief moment. This is exacerbated by the noise picked up by the IMU aboard the Tango. The aforementioned low-pass filter is applied to remove the negative effect of "shaky hands" while we introduce a threshold value to set the minimum angle with which a "tilt" motion can be considered valid. A default value of ± 0.2 radian is set using empirical results and can be customised on-the-fly.

Secondly, we set the natural holding pose of the device as the stationary reference pose. When the user holds the tablet naturally, the tablet is not upright. Therefore, from the user experience point of view, an offset to the forward-backward tilt angle should be introduced. In practice, 0.75 radian is a good reference and can be customised on-the-fly.

Thirdly, a customisable toggle button to send controller commands to the drone is implemented. As the user will be controlling the drone most of the time when the application is launched, the function of the button in the official App to indicate user taking control should be reversed. The drone should ideally listen to commands when the user is not pressing the button, removing the drudgery of holding a button on the screen most of the time. Nevertheless, user should be able to customise this functional button.

Fourthly, the original App provides a 2-D joystick for the user to control the yaw and altitude of the drone. This can be reduced to a 1-D joystick as the yaw of the drone can be synchronised with the direction the user is facing. The detailed method will be described in *Exploring Mode*.

Tilting mode provides an easy control interface for the user when the drone is in sight. However, when the drone goes off sight, user can only rely on the video feed to control the drone which makes this mode less desirable.

C. Exploring Mode

One primary contribution of this work is the introduction of the exploring control mode. This mode allows a user to control the drone intuitively and without any prior training or adaptation to the interface. The drone will simply follow the user's own physical motion while holding the Tango Tablet at hand. For example, the drone will turn anticlockwise should the user turns anticlockwise himself. User can use the video stream to decide on where he/she wishes to move to like exploring in a virtual environment.

The velocity in the x (forward) and y (leftward) directions of the user is obtained by differentiating the displacement of the Tango Tablet. A minimum threshold value is set to trigger movement in each direction with a heuristically derived default value of ± 0.25 m/s in linear directions. A scaling factor is then applied to the velocities to set the power controlling the drone motion. The default value in both linear axes is 0.4 and can be customised by the user on-the-fly.

The yaw (rotational) velocity initially followed similar approach as the linear ones. However, significant drifts have been discovered due to the low quality on board sensor. Instead, we use the absolute yaw of the Tango Tablet to directly map to the drone. For example, a yaw power value of 50 will be set if the yaw values on the Tablet and the drone are 120 and 70 respectively.

To control the altitude of the drone, a 1-D joystick is provided on the screen and according to the extent of the users control, the *setGaz(double power)* function will be called with corresponding power.

The system also provides a button to pause syncing the user's motion to the drone so that the user can reset the reference frame. When the button is pressed, the drone is set to be stationary and the user can freely relocate to a new pose.

V. Mobile Robot MTC Module

To ensure seamless control experience for the user, a controller module is built for a wheel-based mobile robot into the same control interface on the Tango Tablet.

A. Hardware Set-up

The wheeled robot consists of an Arduino Uno board with an additional Bluetooth module connected to a robot shield (as shown in Fig. 2). 4 pins of the Arduino chip are connected to the two motors, with every two pins controlling one wheel(forward spinning and backward spinning). The Bluetooth module is also connected to the board I/O to provide wireless data transmission. Video streaming hardware similar to the Drone set-up can also be added to the robot if needed.

B. Communication and Commands

This wheel robot is programmed to run a 3-state finite state machine (FSM) [7]. The states are *STANDBY*, *INCOMMAND* and *RETRIEVINGEXTRA*. When the robot is powered-on, it takes the *STANDBY* state. The state machine will transition into *INCOMMAND* and listen to ASCII character commands upon the Bluetooth serial port receiving a specific byte (in this case, '-128' is used). There are four command characters: 'F' denotes Forward, 'B' denotes Backward, and 'P' denotes Brake and 'S' denotes Changing Speed. After receiving a command character, the FSM will advance to the state of *RETRIEVINGEXTRA* in order to retrieve the third byte that provides extra information for each command. For example, when the command is 'F' or 'B', the value of the third byte should range between $-100 \sim 100$ to set the rate of turning, where a negative value denotes turning left and vice versa. When the command is 'S', the third byte ranges from $-0 \sim 255$ indicating how fast the motor should run. After receiving the third byte, the robot goes back to *STANDBY* state. The control-loop frequency is set at $25Hz$.

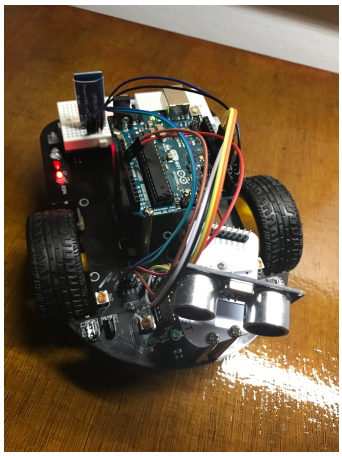


Fig. 2. The prototype wheel-based mobile robot used for development of our intuitive control interface

C. Driving Mode

To optimise the controlling experience, a control mode much like driving a car is implemented on the Tango tablet. There are two buttons provided on the UI interface - 'Forward' and 'Backward' which correspond to the 'F' and 'B' commands respectively. When no button is pressed, a stream of '-128' 'P' '0' is constantly sent to the robot to keep robot stationary. The rate of turning λ is determined by the pitch value of the Tango Tablet providing the value of left and right tilts θ in 2. A threshold on the pitch value for setting the turning rate is set at the default value of ± 0.1 radian. A ceiling for the pitch value is also introduced at ± 0.9 radian. Both are customisable on the fly.

$$\lambda = sign(\theta) int(((|\theta| - 0.1)/(0.9 - 0.1) * 100) \qquad (2)$$

Power for the motors can be adjusted via the Android SeekBar on the UI interface with which the user can adjust the power of the motor.

VI. Image Processing and Mapping Module

One primary task the system could accomplish while performing terrain exploration is mapping. In this work, we demonstrate how vision-based mapping could be done on the

drone while could later be easily replicated onto the ground robots.

A. Setting Up the Parameters

In order to perform visual mapping on the AR Drone 2.0, the frontal camera (720p @ 640×360 resolution) attached to the drone has to be modified to face downwards to provide a clearer aerial view beneath the drone for image stitching. Through the AR Drone API, images captured by thi camera can be streamed at $30fps$. In order to store the images and process the images at the same time, two threads are created to handle the two processes independently while the latter could be handled by an external machine if available.

To enable more accurate image stitching, sufficient number of salient features should be present in any two successive images. Empirically, two successive images should not differ more than 50% in distance in both axes on the AR Drone 2.0. In fact, an optimal distance is approximately 10% of the dimension of the image. Depending on the speed of the drone, images can be collected at different frequencies to ensure successful stitching. The approximate relationship can be described by 3:

$$f = \frac{v_{max}}{\alpha \cdot 2h \cdot \tan(\frac{FOV}{2})} \tag{3}$$

where v_{max} is the $max(v_x, v_y)$ of the drone, h is the altitude of the drone, FOV is the field of view of the camera and α is the percentage of movement of two successive images, 10% will be a good value. The height of the drone can be retrieved from the navigation data of the AR Drone, as it has a ultrasound emitter along with a pressure barometer for measuring its relative altitude.

To simplify implementation on the drone for computational efficiency, based on a suitable data collection frequency, we could set a speed ceiling for the drone. TABLE I shows an example with $h = 1$ m, $\alpha = 10\%$ and $FOV = 64°$:

TABLE I
RELATIONSHIP OF THE DRONES VELOCITY AND THE FREQUENCY OF IMAGE CAPTURING AT RELATIVE ALTITUDE = 1 M

Drone's v_{max}(m/s)	Frequency(Hz)
$0 < v_{max} \leq 0.62$	$0.62/(10\% * 2 * 1 * tan(32°)) = 5$
$0.62 < v_{max} \leq 1.25$	10
$1.25 < v_{max} \leq 1.87$	15
$1.87 < v_{max} \leq 2.50$	20
$2.50 < v_{max} \leq 3.12$	25

As image blurring occurs when either the object in view or the camera is at motion. In order not to further limit the travelling speed of the drone, according to [8], we sharpen the images using a sharpening kernel({-1,-1,-1},{-1,9,-1},{-1,-1,-1}). A comparison before and after the treatment is shown in Figs. 3a and 3b respectively. Fig. 3b can be stitched using OpenCV tools while Fig. 3a fails.

(a) Before applying a sharpening kernel

(b) After applying a sharpening kernel

Fig. 3. Sharpening the images for better image quality

B. Image Stitching Using OpenCV

OpenCV provides a Stitching library which contains a high level API for stitching images. However, the computation cost is potentially high. We devise a simple naive algorithm to benchmark the quality difference against OpenCV as shown in TABLE II

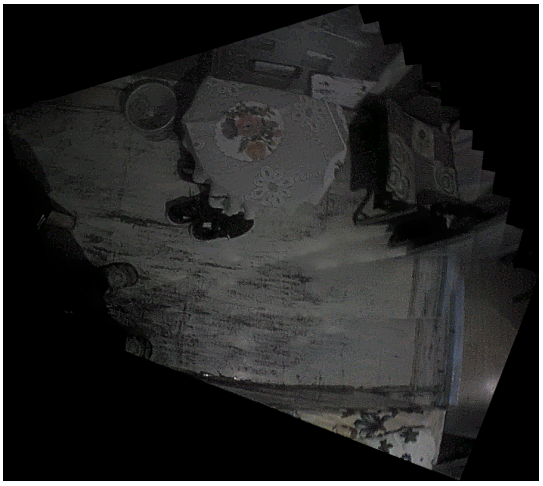
TABLE II
A SIMPLE IMAGE STITCHING ALGORITHM

Step	Description
1	Find features in two sharpened images using a feature detection algorithm. (In this case OpenCVs ORB Feature Detector is used since SIFT algorithm is non-free).
2	Get the descriptors of the features using OpenCV and used brute force to get the 2 best matches for each query and apply a ratio test [9] to optimise the matches.
3	Find the homography transformation of the second image from the matches using RANSAC algorithm provided by OpenCV.
4	Apply the homography transformation to the second image using OpenCV function perspectiveTransform and copy the second image onto the first image.

We compare results in stitching the 13 successively captured images from the AR Drone 2.0 using both methods. The empirical results are shown in Fig. 4

We can see from Fig. 4 that the stitched result is significantly better/with less distortion than that from a naive algorithm. For example, the dustbin remained round in Fig. 4b while being distorted in Fig. 4a. We thus decided to trade-off the

(a) Stitched result using our naive algorithm



(b) Stitched result using OpenCV's stitching module

Fig. 4. Empirical stitching results using a naive method and the OpenCV stitching tool.

computational cost for better quality. The computational cost issue could be solved externally using additional distributed computing resources.

C. Optimisations

In order to expedite the image stitching process, we apply the optimisation technique proposed in [10]. Firstly, a matching mask can be applied so that only successive images are compared in feature matching:

```
Stitcher stitcher = createDefault();
Mat matchingMask(num_imgs,num_imgs,CV_8U,0);
for (int i = 0; i < num_imgs-1; ++i){
    matchingMask.at<uchar>(i,i+1) = 1;
}
stitcher.setMatchingMask(matchingMask.
getUMat(ACCESS_READ));
```

Secondly, the registration resolution and seam estimation resolution of the algorithm can be modified to a smaller value to give a significant reduction in the time used.(using *stitcher.setRegistrationResol(double x)* and *stitcher.setSeamEstimationResol(double x)*).

VII. Experiments & Results

A. Comparing Different Drone Control Modes

Fig. 5. Illustration of a student controlling the drone in an indoor environment.

As mentioned in Section IV, three control modes have been implemented in this work, namely *Exploring Mode*, *Tilting Mode* and *Joysticks Mode*. We conducted an experiment with 30 senior high school students who are technologically inclined. All the participants are introduced to the 3 control modes of the drone and are asked to freely control the drone in an indoor and spacious room (shown in Fig. 5). After experiencing the 3 different control modes, a survey is conducted to determine their preferred control mode which they feel most easy and comfortable with in controlling the drone. The results are tabulated in Table III.

TABLE III
Survey results on the preferred drone control mode

Control Mode	Number of Votes	Percentage of Total Votes
Exploring	19	63.3%
Tilting	9	30%
Joysticks	2	6.67%
Total Votes	**30**	**100%**

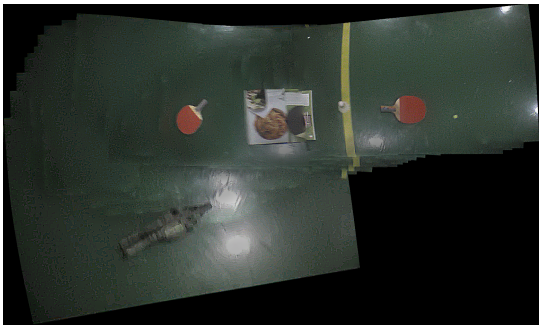
From the results, it is evident that exploring mode is the preferred control mode among the subjects. Joysticks mode are least favourable and only should be used for more precise and accurate controlling. Hence, the new control mode designed in this work can be a better alternative for controlling any given drone.

B. Mapping on the AR Drone

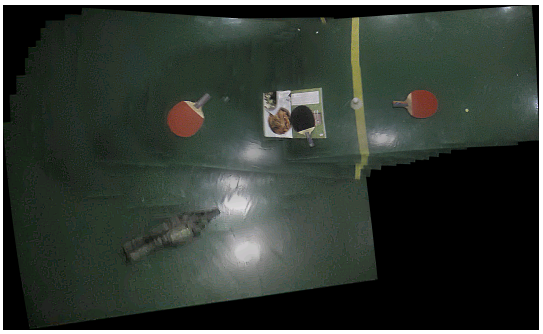
To test the IPM Module implemented in Section VI, an experiment to stitch 32 images collected from AR Drones camera while flying is conducted on both a MacBook Pro (2.6

GHz Intel Core i5) and Project Tango Tablet using OpenCV with the same parameter configuration. There are various objects placed in the environment, thus there can be enough features identified in the images.

It took 5 minutes and 12 seconds for MacBook Pro to complete the task and 16 minutes and 12 seconds for the Tango tablet. The stitched results are shown in Fig. 6:



(a) from MacBook Pro



(b) from the Tango tablet

Fig. 6. Scene stitched results

From visual inspection, the result produced by the Tango Tablet is more natural than that by the MacBook Pro. This could be due to some minor optimisation algorithms used by OpenCV in iOS and Linux. However, processing aboard the Tango Tablet took significantly longer than on the MacBook Pro acting as a remote server. Further investigation is needed to use another Linux server for OpenCV stitching quality verification.

C. Estimating Complexity of OpenCV's Stitching Algorithm

To empirically determine the algorithmic complexity of the OpenCV algorithm, we plot the time taken against the number of images stitched as shown in Fig. 7. A quadratic estimation has been fitted to the 32 data points. The statistics shows strong correlation between the fit and the raw data. Thus, the empirical algorithmic complexity is approximately $O(n^2)$.

VIII. Conclusions

In this paper, we presented our prototype design of a human-robot teaming via an intuitive interaction interface. Two novel control modes have been proposed in this work on a single control unit. We implemented the system using the Google

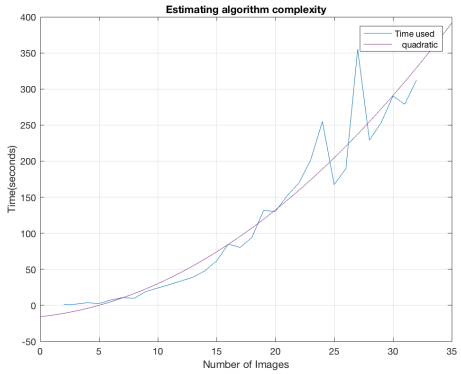


Fig. 7. Graph of time taken to stitch images v.s number of images to stitch.

Tango Tablet as the interactive media to command two types of robots - a UAV and a generic mobile robot with different noise treatments. Three experiments were conducted to understand human preference on UAV control mode, computational efficiency and mapping algorithm performance running on different operating systems.

We plan to introduce stereo vision for the ground robot as well as GPS for both types of robots to provide non-vision-based SLAM services for the robots. This will help to expedite computation for mapping and localisation.

Acknowledgment

This research is supported by NTU-A*STAR-HCI H3 Science Research Programme.

References

[1] G. S. Guthart and J. K. Salisbury, "The intuitive/sup tm/telesurgery system: overview and application," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 1. IEEE, 2000, pp. 618–621.

[2] J. Kato, D. Sakamoto, M. Inami, and T. Igarashi, "Multi-touch interface for controlling multiple mobile robots," in *CHI'09 Extended Abstracts on Human Factors in Computing Systems*. ACM, 2009, pp. 3443–3448.

[3] P. Lapides, E. Sharlin, and M. Costa Sousa, "Three dimensional tangible user interface for controlling a robotic team," in *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*. ACM, 2008, pp. 343–350.

[4] B. Kakillioglu and S. Velipasalar, "Autonomous altitude measurement and landing area detection for indoor uav applications," in *Advanced Video and Signal Based Surveillance (AVSS), 2016 13th IEEE International Conference on*. IEEE, 2016, pp. 166–172.

[5] A. Triaca, "Autonomous flight path mapping with the parrot ar drone," *Thesis, Rhodes University, South Africa*, 2014.

[6] D. Ahn, J.-S. Park, C.-S. Kim, J. Kim, Y. Qian, and T. Itoh, "A design of the low-pass filter using the novel microstrip defected ground structure," *IEEE transactions on microwave theory and techniques*, vol. 49, no. 1, pp. 86–93, 2001.

[7] C. Zeng, N. Saxena, and E. J. McCluskey, "Finite state machine synthesis with concurrent error detection," in *Test Conference, 1999. Proceedings. International*. IEEE, 1999, pp. 672–679.

[8] P. Joshi, "Detecting edges and applying image filters," in *OpenCV with Python By Example*. Packt Publishing Ltd, 2015, ch. 2.

[9] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, pp. 91–110, 2004.

[10] B. Jones, "Fast panorama stitching," 2014. [Online]. Available: https://software.intel.com/en-us/articles/fast-panorama-stitching