UAV System Integration of Real-time Sensing and Flight Task Control for Autonomous Building Inspection Task

Gong-Yi Li[1], Ru-Tai Soong[1], Jyi-Shane Liu[1, 2], Yen-Ting Huang[1]
[1]Department of Computer Science
National Chengchi University
Taipei, Taiwan
[2]Pervasive Artificial Intelligence Research (PAIR) Labs, Taiwan
e-mail: jsliu@cs.nccu.edu.tw

***Abstract*—Most current research on autonomous UAV control are conducted in virtual environment and often focus on specific technical domains, such as communication protocol and visual computing, etc. These research are rarely integrated as one, making it less applicable and effective. However, application platforms solving practical problems require system integration in real-world environments. In this paper, we propose a UAV task-oriented flight control system integrated with real-time sensing based on Robot Operating System. The flight control system also incorporates behavior tree as a decision control mechanism. We apply our system in a building inspection task, whereas the UAV takes-off near the riverbank, flies forward following the road across a bridge, and arrives at a designated building to perform a zig-zag image scan. The system is implemented and field tested on a ready-to-fly quadrotor. Captured images are transmitted through Wi-Fi to a laptop for real-time visual sensing, and flight directions are then calculated and provided to the UAV for navigation. Our experiment shows promising results - the UAV can successfully complete the task within 25 minutes. Some suggestions and solutions are provided for future improvements.**

Keywords—Visual Sensing; Real-Time Computing; UAV; Navigation Control; Behavior Tree; Robot Operating System

I. Introduction

The use of unmanned aerial vehicles (UAV) in civilian applications has increased in recent years. Recent examples such as Amazon's Prime Air Delivery Service aim to deliver packages to customers within 30 minutes using UAV [1]. Classical applications, such as search and rescue operations [2], have also long been considered using UAV as a more effective searching tool. However, most current UAV research focused on individual technical components, such as remote sensing, image post-processing for vegetated areas [3] and landslides [4], object detection using UAV imagery [5][6], and coverage path planning for different goals [7][8]. Furthermore, most UAV research is conducted in simulated environment [9] [10], which often underestimates the potential risk and uncertainty of real-world environment. For UAV to solve problems in real-world applications, it needs rigorous verification in real environments with specific objectives. Hence, our technical goal is to develop a task-oriented UAV-based flight control system with actual implementation in real-world environments.

In this paper, our experiment takes place along the bank of Jingmei river at sub-urban Taipei. Our UAV's task in this scenario is to inspect a cylinder-like building located on the other side of the river. The UAV will take-off on one side of the river, follow the road alongside the river, fly pass over the bridge to the other side of the river, and arrive at the designated target building to perform a task-oriented path image scan, which is a zig-zag pattern scan. UAV-based building inspection [11] has been considered as a safe alternative with a lower risk of inspection personnel. Such application can be widely used in many fields, such as construction area and detonation site, etc.

Previous works only focus on learning simple behaviors to build corresponding trees, which is generated and verified in game or lab environments. Hence, they may not deal with such complex and long-duration flight executions. In real-world tasks with insufficient image data for training, it is required to incorporate traditional vision algorithms and integrate with convolutional neural network models. Our proposed approach aims to design flexible and reusable modules with expert experience, and each module represents a general and basis behavior tree that can be adapted to other specific vision tasks with little modification.

Our system uses behavior trees [15] as a decision control mechanism to manage context changes and flight behaviors. To complete the described inspection task, we developed three visual sensing functions with captured images by UAV's front camera. The first function is for road detection, which allows the UAV to fly following the road. It uses a convolutional neural network to produce image segmentation, which is further processed to produce flight directions for UAV. The second function is for landmark recognition. It is used to identify a specific landmark with a pre-captured image to verify a flight waypoint. The third function is a task-oriented path planning function. It uses an image histogram to compare the current UAV's captured image with a pre-captured image. As a result, the relative position of the UAV to the target can be calculated, hence allowing the UAV to inspect the target in a zig-zag manner.

In most UAV system, GPS is considered as a basis of navigation control. However, the instability of GPS accuracy may often lead to deviation of the flight path, sometimes causing hazards to the UAV itself and the surrounding environments. Instead of relying on GPS as the main navigation input, we use visual sensing as the primary source of contextual information. Instead, GPS is used as supportive information only for waypoint verification. In other words, we use landmark recognition from visual sensing as the

978-1-7281-4666-9/19/$31.00 ©2019 IEEE

primary input and GPS information as the secondary input for waypoint verification. This double-checking has increased the robustness of flight navigation.

In summary, our system adopts three types of visual sensing to provide specific contextual information of the environment and the task target, while using GPS only as supportive waypoint information. For task-oriented control, we use behavior tree to integrate perception with UAV navigation. For safety precautions, our operator uses a ground control station to supervise the whole process and can interrupt the UAV operation with manual commands in case of emergency.

II. Technical Components

A. Robot Operating System as system foundation

The Robot Operating System (ROS) [12] is an open-source robotic platform that provides hardware abstraction, low-level device control, implementation of commonly-used functionality, and message-passing between processes. ROS uses graph architecture. Each process is a node in the computational graph model, whereas a *node* uses *topics* to send and receive *messages* in an asynchronous manner.

Our system consists of five nodes, including the *Roscore* that handles overall ROS environment and graph model, the *ROS Driver* that handles communication with the UAV, the *RosBridge_Suite* that handles message-passing with ground control station interface through HTTP, the visual sensing node that processes captured images into meaningful information, and the flight task control node that hosts behavior tree as well as navigation control and other sensors' input. The last two nodes are self-developed for our system.

B. Behavior Tree as Flight Task Control

Behavior tree is a mathematical model of plan execution used in computer science, robotics and control systems. It describes switching between a finite set of tasks in a hierarchical manner. There are four types of nodes in behavior tree, the sequential and fallback nodes are control flow nodes, labeled as an arrow and a question mark, the condition and action nodes are execution nodes, which are often used as leaf nodes.

Take figure 10 as an example, the *Take Off* node with a question mark below is a fallback node. A fallback node will only execute until a child node returns a value of true. Hence, it will only execute the *Take off* action node when the *h>=xm* condition node returns false. This fallback node illustrates that if the UAV does not meet certain height, it assumes the UAV is on land, and hence take-off action is executed. Sequential node, such as the *Road following* node with an arrow, will execute all nodes from left to right until false is returned. Hence, the UAV will only stop the *road detection and following* action node if it is in state 2 or 3, and according to our diagram, it will only switch to state 2 or 3 if the UAV is closing on a given waypoint.

C. Road Detection using DeepLab in MobileNetv2

Image segmentation is the task of partitioning an image into multiple segments, where each segment is represented by a layer of color. This makes the image easier to be analyzed. DeepLabv3+ [13], leveraged by the trade-off between speed and accuracy, is a deep learning neural network for semantic segmentation, whose goal is to assign semantic labels (e.g., person, dog, cat and so on) to every pixel in the input image. We tested DeepLabV3+ with Xception and MobileNetV2 as backbone in our specified environment, and obtained a score of 0.6 in 4 frames per second as shown in Figure 1.

Xception is a network structure intended for server-side deployment, whereas MobileNetV2 is a faster network structure designed for mobile devices. The network takes a captured image of 856x480 pixels from UAV as input, and outputs semantic segmentation of classes as defined in the Cityscapes dataset for further processing. We then extract only the largest segmentation of road, which is used to calculate the central line of the road to identify where the UAV should direct its course of the flight.

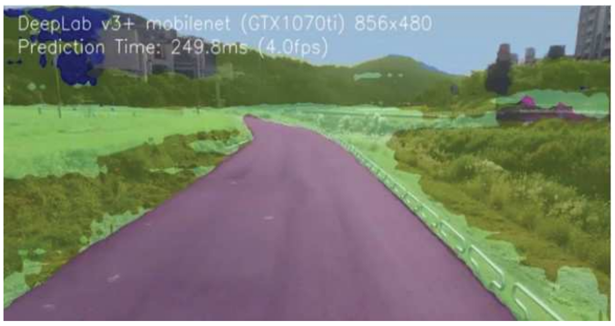

DeepLab v3+ mobilenet (GTX1070ti) 856x480
Prediction Time: 249.8ms (4.0fps)

Figure 1. 4 FPS of DeepLabV3+ in MobileNetV2

D. Landmark recognition using SURF

We prepared an image of a selected landmark, then uses a Speeded Up-Robust Features (SURF) provided by OpenCV to produce keypoints and their descriptors. Due to insufficient image data for training and the feasibility of real-world practice, we address this problem through feature matching instead of the NN-based method. Using the keypoints and descriptors produced as a baseline, we compared it with the currently captured image by UAV's front camera using K-nearest neighbors search and produced a list of feature points. We then filtered the list to remove unsuitable points by calculating its distance difference using David Lowe's ratio test. The final list of keypoints will be used for UAV's position calibration and waypoint verification.

E. Building relative position detection using histogram

An image of the designated target is provided and processed into a histogram with OpenCV. We apply histogram-based approach to extract target features because it is hard for CNN model to extract arbitrary object contour due to poor generalization to unseen data. The captured image of the task target is split into left side and right side, each a 428x480 image. Both images are further sliced into 3x3 blocks. A score is given to each block by comparing its histogram with the histogram of the provided image. Hence, we can compare the left and right images with blocks of scores to acquire the relative position of the UAV to the target. If the blocks of scores that exceed a certain constant are less in the left image compare to the right image, it means the UAV position is on the left edge of the designated target.

F. Navigation control using ROS topic

We use ROS topic *cmd_vel* to instruct UAV's moving direction and orientation. During the road detection phase, the calibration point is used to guide the UAV to fly towards the middle of the road, while the vanishing point is used to adjust the orientation of the UAV to face it. However, due to the asynchronous characteristic of computing speed and UAV flight speed, a timer is set to prevent visual sensing from using older frames as inputs to produce outdated results. In summary, the UAV flight movement during road detection and following is composed of a sequence of three actions: move, orientation adjustment, and waiting.

During the zig-zag scanning phase, the flight task control node takes in a relative position. The histogram function then determines whether the UAV is on the left edge, middle, or right edge of the building. If the UAV is at the edge, it will start to lower its height, then move towards the edge of the other side.

G. Ground control station

The ground control station (Figure 2) is a simple interface built by Unity for monitoring UAV behavior. It uses ROS# provided by Siemens to communicate with ROS through *RosBridge_Suite* to enable message-passing. Other than information display, the GCS also implemented additional functions, such as manual control using a keyboard or mouse, sending commands to UAV to perform a series of actions, or simply halting its current operation.

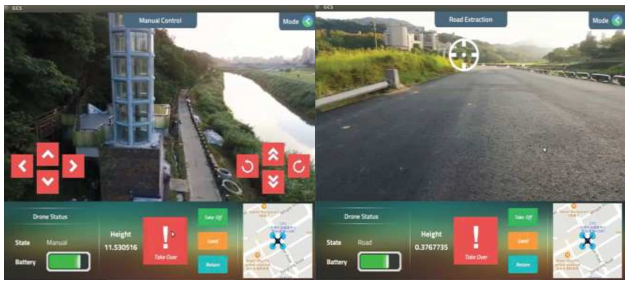


Figure 2. Ground control station screenshots in manual mode (left) and road detection following with crosshair (right)

III. SYSTEM ARCHITECTURE

Our current UAV system has three main modules. The cognition module, performed by the flight task control node, handles decision making and flight control. The perception module, implemented as the visual sensing node, processes sensory inputs into informative outputs for the cognition module. The ground control station, which is part of the interface module, serves as an interface between the UAV and a human operator.

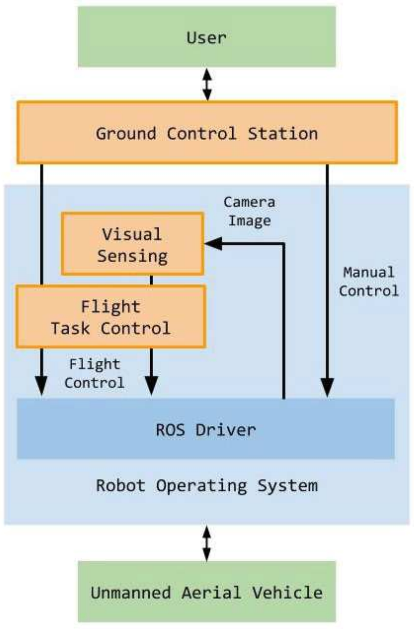


Figure 3. System Architecture

As shown in Figure 3, the task is initiated by the operator through our ground control station. The UAV starts to take-off once the configuration is set and initiates road following mode at the beginning. The visual sensing node then takes in camera images from the relative ROS Driver, processes through the DeepLabV3+ with MobileNetV2 to produce a segmentation mask. This mask is further processed into calibration point and vanishing point.

A. Road detection and following

We apply approximation to the largest contour in the semantic segmentation image making drone less shaking, then produce a central line of the contour, which is smoothed to further reduce jagged. The central line is composed of many points. Hence, we store all the points in an array, arranged in sequence of bottom to top. We first select the 31st point as a starting point, then calculate the previous 30 points and the next 30 points relative to it to produce a relative linear slope equation for both 30 points. We then find the slope difference between these two equations. The slope difference is stored in a list and proceed with the 32nd point, and so on. This operation continues to proceed until the last 31st point.

A constant point is selected as the center of the UAV vision. It is used to compare the list of slope differences. The point with the largest slope difference to the center of UAV vision is selected as the calibration point. The last point in the list is selected as the vanishing point. Both points are then passed to the flight task control node as shown in Figure 4.

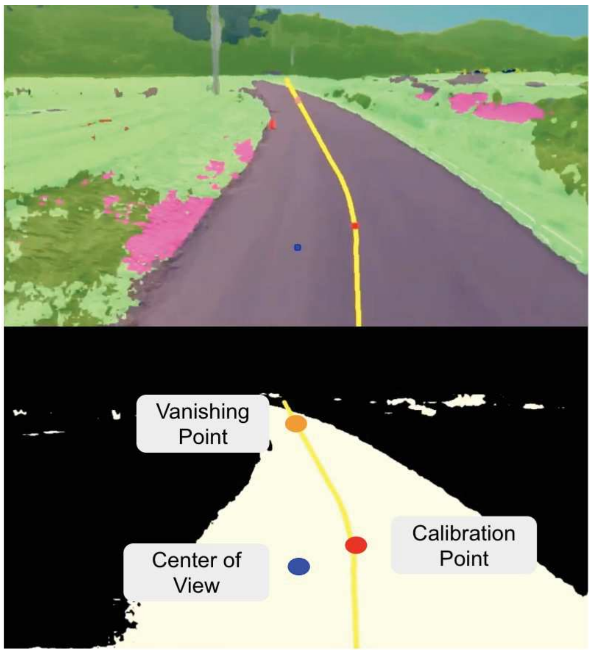


Figure 4. Calibration point and vanishing point calculated from the central line

As mentioned earlier, our flight task control node will use both points to perform road detection and following navigation. Upon approaching a given waypoint, the system will perform waypoint verification and switch its task context accordingly.

B. *Landmark recognition*

Upon reaching a certain parameter of waypoint 2, the UAV will initiate the landmark recognition function using a pre-captured image to compare with the currently captured image. The UAV will adjust its position and close in on the landmark. Upon arriving at waypoint 2, the UAV will turn and face waypoint 3 according to GPS calculation.

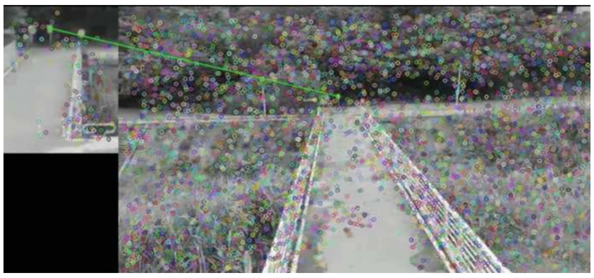


Figure 5. The landmark near waypoint 2 is specified by a pre-captured image to compare with the current image stream

C. *Building inspection*

Once arriving at the designated building, the system will adjust the UAV's orientation to face the building using GPS. The UAV system is then switched to target select and scan node, and starts to rise to a certain height previously specified by the operator. A region of interest (ROI) is selected in real-time by the operator as input. The visual sensing node then takes in the currently captured image and the image selected to produce a building relative position based on histogram for the UAV to maneuver. Upon reaching a constant height, the UAV will remain hover and wait for further instruction.

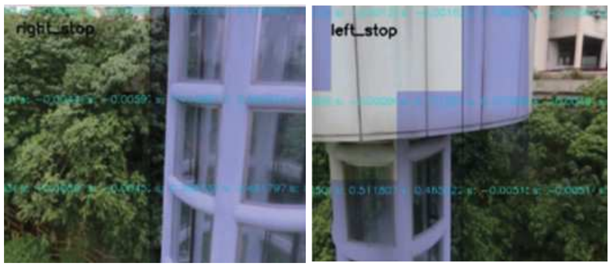


Figure 6. The left image shows the UAV is on the left edge as histogram detects the target on right part, the right image shows the UAV is on the right edge determined by the score left larger than right in purple color

IV. Field Test and Result

We selected a ready-to-fly UAV, the Parrot Bebop 2 developed by Parrot Inc. as our UAV for field-tests. It is a quadrotor equipped with built-in orientation and altitude sensors, fisheye lens front camera, and Wi-Fi module. We use ROS Driver for Parrot Bebop Drone [14] developed by Autonomy Lab as an interface to ROS. A laptop with GTX1070ti is served as the computing device for the system, which communicates with the UAV through Wi-Fi. Hence, our ground control station, ROS, related drivers, libraries, and codes are all computationally processed on the laptop.

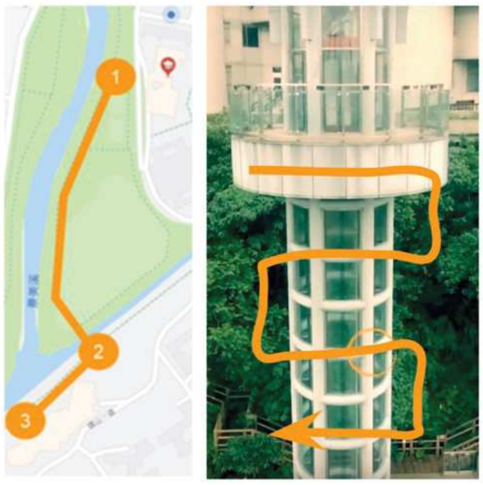


Figure 7. Waypoint and the inspection target with zig-zag pattern

We divided our inspection task into three waypoints. First, the UAV will be placed on waypoint 1 by the operator. It will take-off and fly forward by following the road until it passes a bridge and reaches waypoint 2. The UAV will then adjust its orientation to align with the road and fly towards waypoint 3, where the designed target is located. The target is an elevator (Figure 7 right) connected to a higher ground infrastructure with a height of 30 meters.

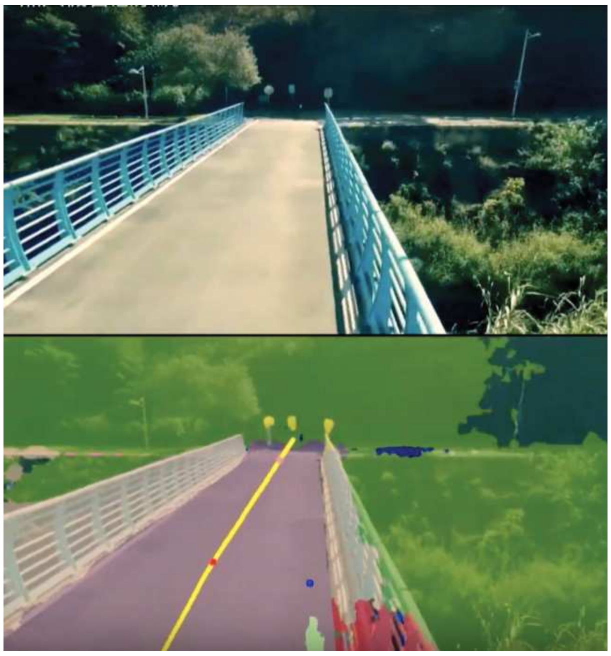


Figure 8. The UAV is passing the bridge before reaching waypoint 2

Upon arrival at waypoint 2, the UAV will turn and face waypoint 3 based on GPS. The path to waypoint 3 is a straightforward paved road along the riverbank. After reaching waypoint 3, the UAV will turn to face the designated target based on GPS again. The UAV rises to a height of 25 meters. The operator then selects an image as input for the histogram. The UAV starts to perform zig-zag scanning according to its relative position to the building. It is set to lower by 2 meters once it reaches the edge of building and moves towards the other side of the target. Once the UAV's flight height is reduced to 5 meters, its task is completed. It will remain hovering, waiting for the operator to provide landing instruction.

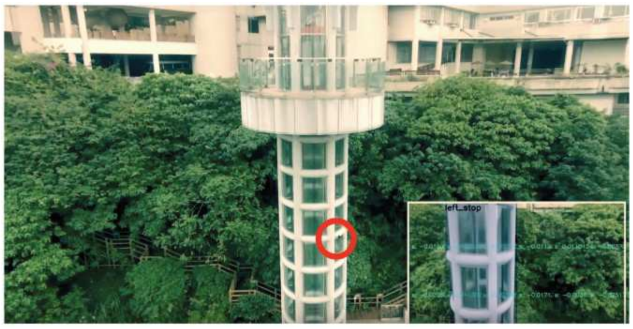


Figure 9. Image taken from another UAV to record the scanning UAV (red circle), the lower right image shows the processed histogram

Over fifty experiments, there were thirty-five successful autonomous flights, eight successful flights with operator support and seven failed flights. A video recording of one of the successful experiments of autonomous task-oriented flight is provided [16]. Wind interference is sometimes significant around the area, however, no failure was caused by the wind. Due to battery capacity, the UAV has a flight time constraint of 25 minutes. But the overall results show our system capable of task completion within the time limitation. Although most of our field tests ran successfully, some issues required the operator to react immediately.

A. *Camera image freeze*

There is a slight chance that the camera image stream may freeze and require a reset to recover. As mentioned earlier, we used a ready-to-fly commercial UAV, and due to its closed system, we were unable to identify the source of the problem because everything else was working normal, except the camera stream always passed the same image to our node. This issue can be solved by comparing each keyframe with the previous one to launch an aerial reset to the system.

B. *Using landmark recognition for GPS problem*

As mentioned, we use landmark recognition with SURF when closing in range of waypoint 2. Due to the GPS unstable signal in that particular area, the bridge area will direct the UAV towards the left side riverbank of the bridge, which is about 10 meters far from waypoint 2. Hence, our alternative method was to identify a landmark so the UAV can reach waypoint 2.

C. *Environment light source problem*

We define path A as waypoint 1 to waypoint 2 and path B as waypoint 2 to waypoint 3. We tested our system at different times and found out that there were significant differences in test results. In the morning, the light source at path A is strong, which resulted in a problem where the contour may not be identified as a road. However, after switching to path B, the UAV can fly smoothly. For evening when the sun is in another direction, the light source in path B became strong, resulting in a reverse behavior as compared to morning. As our experiment took place near the river bank, the strong light may also cause the model to recognize the river as the road, which may result in the UAV flying *towards* the river. To counter this issue, we are developing a model that enhances detection for degraded images.

V. Conclusion

This paper describes the application of a real-time autonomous task-oriented flight control system for building inspection. The image stream is processed through a neural network and is calculated to enable the quadrotor to fly towards the designated target by following the road. While arriving at the designated target, it uses edge-triggered movement patterns to perform coverage scanning. The designed system architecture and the overall task performance have been successfully verified in real-world environments. We also provide the following observations.

A. Behavior tree excels at UAV task management

With the initial success of the building inspection task, we are in the process of designing new behaviors for more complex inspection tasks. Although there are other technical issues in adding multiple sensors and functions, the process of adding more sub-trees does not complicate our system. We conclude that behavior tree is effective in building an autonomous UAV task management system in real-world environment.

B. Path planning for performing 2D/3D inspection

To provide better structure inspection, we are developing a motion module to produce a series of waypoints as flight paths. This will enable the UAV to surround a building to perform inspection based on given information such as 3D model or building height and area, which greatly improves the range of application tasks. Since most buildings have accurate measurements and blueprints, this can be considered a general way to perform building inspection. With the addition of a path planning module, the autonomous task control architecture can also be applied to the inspection of larger areas, such as farms and factories.

ACKNOWLEDGMENT

This work was partially supported by The Ministry of Science and Technology, Taiwan, under GRANT No. MOST108-2634-F-004-001 through Pervasive Artificial Intelligence Research (PAIR) Labs.

REFERENCES

[1] Here's Amazon's New Transforming Prime Air Delivery, The Verge, 2019. [Online]. Available: https://bit.ly/2Wsdnz2

[2] C. Beard, Z.-Q. Chen, V. Kumar, Y. Lee, W. D. Leon-Salas, and P. Rao, "Saveus: Saving victims in earthquakes through unified systems," IJCNDS, vol. 10, no. 4, pp. 402–420, 2013.

[3] Niethammer, U., James, M. R., Rothmund, S., Travelletti, J., & Joswig, M. UAV-based remote sensing of the Super-Sauze landslide: Evaluation and results. Engineering Geology, 128, pp. 2-11, 2012.

[4] Salamí, E., Barrado, C., & Pastor, E. UAV flight experiments applied to the remote sensing of vegetated areas. Remote Sensing, 6(11), 11051-11081, 2014.

[5] Bazi, Y., & Melgani, F. Convolutional SVM networks for object detection in UAV imagery. Ieee transactions on geoscience and remote sensing, 56(6), pp. 3107-3118, 2018.

[6] Torres-Sánchez, J., López-Granados, F., & Peña, J. M. An automatic object-based method for optimal thresholding in UAV images: Application for vegetation detection in herbaceous crops. Computers and Electronics in Agriculture, 114, pp. 43-52, 2015.

[7] Chen, Y. B., Luo, G. C., Mei, Y. S., Yu, J. Q., & Su, X. L. UAV path planning using artificial potential field method updated by optimal control theory. International Journal of Systems Science, 47(6), pp. 1407-1420, 2016.

[8] Di Franco, C., & Buttazzo, G. Energy-aware coverage path planning of UAVs. In 2015 IEEE International Conference on Autonomous Robot Systems and Competitions, pp. 111-117, April 2015.

[9] Páli, E., Mathe, K., Tamas, L., & Bușoniu, L. Railway track following with the AR. Drone using vanishing point detection. In 2014 IEEE International Conference on Automation, Quality and Testing, Robotics, pp. 1-6, May 2014.

[10] Barton, J. D., Cybyk, B., Drewry, D. G., Frey, T. M., Funk, B. K., Hawthorne, R. C., & Keane, J. F. Use of a High-Fidelity UAS Simulation for Design, Testing, Training, and Mission Planning for Operation in Complex Environments. Wiley Encyclopedia of Operations Research and Management Science, 2010.

[11] Al-Kaff, A., Moreno, F. M., San José, L. J., García, F., Martín, D., de la Escalera, A., ... & Garcéa, J. L. M. VBII-UAV: Vision-based infrastructure inspection-UAV. In World Conference on Information Systems and Technologies, pp. 221-231, Springer, Cham, April 2017.

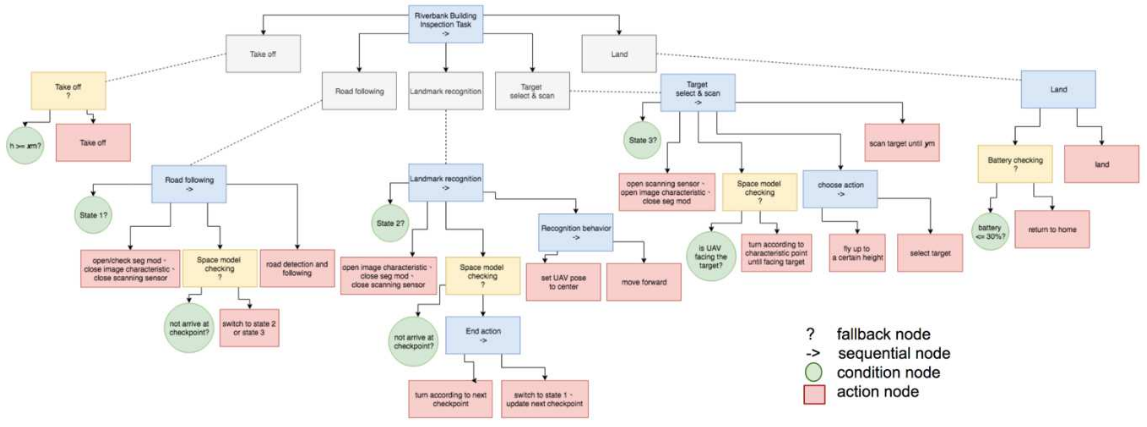
[12] Von. "Ros.org", 2018. [Online]. Available: http://wiki.ros.org/mavros, [Accessed 6/06/2018]

[13] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510-4520, 2018. Available: https://github.com/tensorflow/models/tree/master/research/deeplab

[14] Mani. "bebop_autonomy", 2015. [Online]. Available: https://bebop-autonomy.readthedocs.io/en/latest/, [Accessed 6/06/2018]

[15] Klöckner, A. Behavior trees for UAV mission management. INFORMATIK 2013–Informatik angepasst an Mensch, Organisation und Umwelt, 2013.

[16] UAV System Ingeration of Real-time Sensing and Flight Task Control for Autonomous Building Inspection Task, 2019. https://www.youtube.com/watch?v=vLp0Q5p47a0



Riverbank Building Inspection Task ->
Take off
Road following
Landmark recognition
Target select & scan
Land
Take off ?
h >= xm?
Take off
Road following ->
State 1?
open/check seg mod - close image characteristic - close scanning sensor
Space model checking ?
road detection and following
not arrive at checkpoint?
switch to state 2 or state 3
Landmark recognition ->
State 2?
open image characteristic - close seg mod - close scanning sensor
Space model checking ?
not arrive at checkpoint?
End action ->
turn according to next checkpoint
switch to state 1 - update next checkpoint
Recognition behavior ->
set UAV pose to center
move forward
Target select & scan ->
State 3?
open scanning sensor - open image characteristic - close seg mod
Space model checking ?
is UAV facing the target?
turn according to characteristic point until facing target
choose action ->
fly up to a certain height
select target
scan target until ym
Land
Battery checking ?
battery <= 30%?
return to home
land
? fallback node
-> sequential node
condition node
action node

Figure 10. Building inspection task is composed of 3 main sequential nodes: road following, landmark recognition, target select and scan.