November 21, 2013

Abstract

Contents

1 Introduction

General information on quad/hexacopters and computer vision.

1.1 Background and Motivation

Introduce the wind turbine case. What are the benefits of doing it the UAV way? What are the main challenges.

1.2 Previous Work

Find out what else has been done on this particular subject. Mention some of the methods that I have utilized. Sources and citation spam.

1.3 Contribution and Scope of this Report

What in particular is achieved. (as opposed to the general benefits mentioned in "background and motivation").

1.4 Organization of this Report

1.5 Nomenclature and Notation

2 Materials and Methods

Change title name?

2.1 Key Features of a Wind Turbine

Describe features that are and can potentially be utilized in a/the program. Including geometry, color/texture, size (including in relation to distance), movement, variety, environment/background.

2.2 Program Overview and Strategy

Introduce the plan/strategy, include UML or similar diagram. Should all modules be introduced at once, or should the program be presented incrementally as each module is described.

2.3 Module Descriptions

Work independently, more could be added to improve the algorithm.

2.3.1 Color Thresholding

How does it work. Why BGR? Range and "bias". Extracting binary image.

2.3.2 Canny Edge Detection

How does it work. Why use it?

2.3.3 Dilation Filter

How does it work. Why, fill holes and cracks.

2.3.4 Finding Contours

How does it work. Why results are useful; getting angles, area, circumference, convexity defects.

2.4 Autonomous Parameter Tuning Using Moments

Introduce tuning problem. How to effectively do it autonomously. What are moments and how do they help.

2.5 Input Samples

Single image, video, bad conditions. Isolate problems. Image: initial algorithm, finding parameters, controlled/fixed environment Ill conditioned: check for robustness, how far before impossible Video: investigate frame to frame variations,

The program is capable of both taking still images as well as video clips or video streams as input, which allows more control and diversity on testing environment. Any resolution is accepted, but in order to limit computation costs the input array will be downscaled past a user-defined limit using the Gaussian pyramid.

To begin with, still images will be used as input in order to investigate the basic functionality of the algorithm. This way it will be clear how the image filtering methods handle specific image conditions. Images will range from very good to very ill-conditioned. An image can typically be classified as good by possessing the following properties: Clear foreground to background distinction, absence of objects of similar colour and shape as the wind turbine, insignificant amounts of noise and homogeneous colours on the wind turbine (i.e. few shadows and no overlapping objects). Images will be tested first, in order to effectively investigate the image filtering methods

2.6 Output Data

Contours; with convexity defects and 'arms'. Angles. Center of turbine.

3 Results

3.1 Image as Input

Images with good contrast to background, and few things of same color as wind turbine.

3.2 Ill Conditioned Image as Input

Not fulfilling previous conditions.

3.3 Video as Input

With purpose of showing how the algorithm varies from frame to fram. Also try bad conditioned video??

3.4 Computation Speed

Both with and without autonomous tuning. Without meaning that parameters are kept from previous frame, thus greatly reducing computation needed.

4 Discussion

4.1 Applicability of Output Data

What benefits are obtained from the outputs? Detection of center of turbine. Orientation from angles, how much is overlap a problem? Can the turbine even be recognized at a bad angle? Are the results reliable enough, how does the program handle good/bad conditions, how about frame to frame consistency?

4.2 Causes of Errors

Judging from results, what are the major causes of errors. Sky color, clouds. Wind turbine overlap. Wind turbine color variation, mostly due to shadows. Other distracting objects.

4.3 Cross-Platform Compatibility

Mainly with focus on computation time. How does the panda-board handle it.

4.4 Future Work

4.4.1 Frame to Frame Memory

Kalman filtering in order to obtain smoother and robust tracking. If not done: keeping parameters from previous fram, perhaps probing lightly in both directions.

4.4.2 Cross-Referencing Different Methods

Use other methods and check agreement for robustness.

5 Conclusion

6 Acknowledgements

7 Literature Cited

8 Appendices

8.1 C++ Code

8.2 UML Diagrams