# Implementing Computer Vision Algorithms for Autonomous UAV Applications

Sean R. Stockholm
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA 56267
stock424@morris.umn.edu

## ABSTRACT

The tracking of moving objects over a large area can be time consuming and expensive. An increasingly popular solution to this is the use of unmanned aerial vehicles (UAVs) with cameras which are capable of navigation and tracking. This paper is about new uses for drones, and methods for their control and navigation. UAVs used are often small quadcopters [1]. These UAVs can fly easily through areas humans could not easily navigate, using computer vision to gather data about these areas. There has been a rapid increase in the number of applications for drones equipped with vision systems. The applications surveyed are the tracking of ships and locating of wild fawns. This has impacts on many different areas from wildlife to drug trafficking. This rapid increase has also lead to the increased implementation of algorithms which are useful in image processing; one we will survey is a signal filtering algorithm called the Kalman filter.

## Keywords

Drones,Computer Vision

## 1. INTRODUCTION

The tracking of moving objects over a large area can be time consuming and expensive. An increasingly popular solution to this is the use of unmanned aerial vehicles (UAVs) with cameras which are capable of navigation and tracking. UAVs used are often small quadcopters. These UAVs can fly through areas humans could not easily navigate. The papers surveyed all use UAVs equipped with thermal cameras and GPS systems to track objects based on their thermal characteristics.

Israel [2] uses drones to detect deer fawn, which often sleep in pastures. Fawns have little scent in the beginning of their life, so they cannot be found with dogs. Additionally their instinct is to not move at all, unless their mother says so, so loud noises will not cause them to relocate. Since they are difficult to detect they are often accidentally killed during the mowing of pastures. While it is unfortunate to kill a fawn, dealing with the corpse quickly becomes an issue. It is far easier to relocate a fawn before mowing it. Additionally a rotting fawn ruins the nearby grass and makes it unusable as hay. The researcher made an app based on the Google Maps API, in which a farmer can draw a path through their field that the drone will follow searching for sleeping fawns, and notifying the farmer of their location. The farmer can then manually relocate them.

Liera et al. [3] use a low cost fixed-wing UAV with a forward facing thermal camera to classify and track objects on the surface of the ocean. There is great interest in this because of its potential uses in border control, namely smuggling of drugs. In section 4 we describe methods of identifying boats. Next we explain how they handle the tracking of boats using a signal filtering algorithm called the Kalman filter.

Ward et al. [6] describes the use of drones in monitoring wildlife and feral animals. It uses methods similar to those employed by Leira et al.[3] and Israel[2]. Ward also incorporates GPS tracking of the operator of the drone, so that the drone can follow the operator, but this aspect of the application is outside the scope of this paper. Ward also implemented the Kalman filter for tracking, but does not go into any detail on his implementation.

## 2. TECHNICAL OVERVIEW

We will start by describing the hardware used by Israel[2], walking through a system diagram and discussing the technologies aboard his drone. Because of similarities between drones used, we will only describe the drone system used by this research paper. Israel captures and processes images used for fawn detection. We will make use of a visible light camera to illustrate process of detecting objects. Leira et al., Israel and Ward have very similar system architectures and utilize similar computer vision techniques.

Ward and Leira et al. go a step farther implementing signal filtering with the Kalman filter as was applied by both Israel and Ward. Making use make use of a simple one-dimensional example to illustrate the main ideas that make the Kalman Filter, we will explore the application of the Kalman Filter by Leira et al.

## 3. HARDWARE SYSTEMS

[1]Quadcopters, octocopters, etc. are similar to helicopters, but have 4, 8 or whatever the prefix implies small propellers, instead of one large propeller.

*UMM CSci Senior Seminar Conference, March 2016* Morris, MN.

The drone deployed by Israel was an octocopter[2], as shown in figure 2. Images are taken with a downwards facing thermal camera, and image processing was done on a small single board computer, similar to a Raspberry Pi. Motor control is handled by a dedicated controller, which contained an on-board GPS. The drone can be flown manually, or along a path provided to it. Information about the drones heading and location are also sent to on board computer.

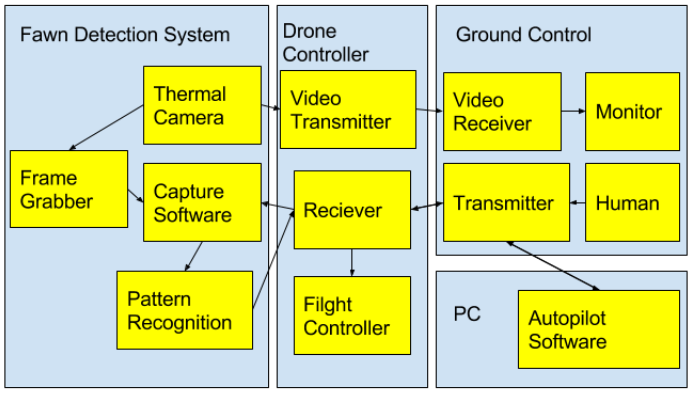


**Figure 1: system architecture diagram**

Figure 1 depicts the system architecture diagram for the drone used by Israel. They use the data from the thermal camera that is sent both to the ground, and to the computer vision system on board. We will discuss the computer vision in section 4. The video feed sent to the ground control is viewable on a monitor.

Being able to view the thermal camera footage from the ground allows the pilot to manually search for fawns. This also gives them the ability to process the thermal camera footage on a more powerful computer, as they did in earlier research. The reason they changed to processing the video data on the drone has to do with the cost effectiveness of using a smaller computer, and the convenience of not needing to bring a larger computer to the field to search for fawn.

For safety reasons, the drone cannot be operated completely autonomous, so a transmitter, receiver, and pilot were still necessary for this research.

## 4. VISION

All research described in this paper performs image processing on an images from a thermal camera. For our examples we replicate this with a visible light camera. This allows us to apply computer vision techniques similar to those used in the sources. The camera is connected to a Raspberry Pi doing the image processing. The Raspberry Pi is similar in computational power, and power consumption to those used by the sources. Installed on it is an open source computer vision package called OpenCV[4]. OpenCV was also used by Liera et al.[3] for their vision processing.

This section walks you through how vision processing is done on a Raspberry Pi on a drone. We will be utilizing a computer vision setup I created which has several similarities to the ones used in my references. For demonstration, we are using a visible spectrum camera instead. We will treat image's greenness as though it is heat. Israel[2] used

**Figure 2: drone deployed by Israel[2]**

unnamed software to do pattern recognition on the images, searching for fawn based on size.

Grass can obscure the presence of the fawn, so flying higher lets the drone more easily peer down between the blades of grass. Because of this, the combination of downwards facing thermal camera and drone is highly potent. Unfortunately the higher the drone flies, the less pixels the fawn occupies in the image captured. As result, the size requirements for a fawn scale with the height of the drone, meaning that they are searching for a smaller thing the higher they fly. For a full discussion of the trade offs see [2]. Scaling with height is also a problem for Leira et al. It is dealt with in a similar way.

For our model, we use a white golf ball as a stand in for a fawn, since both have high contrast with their surroundings.

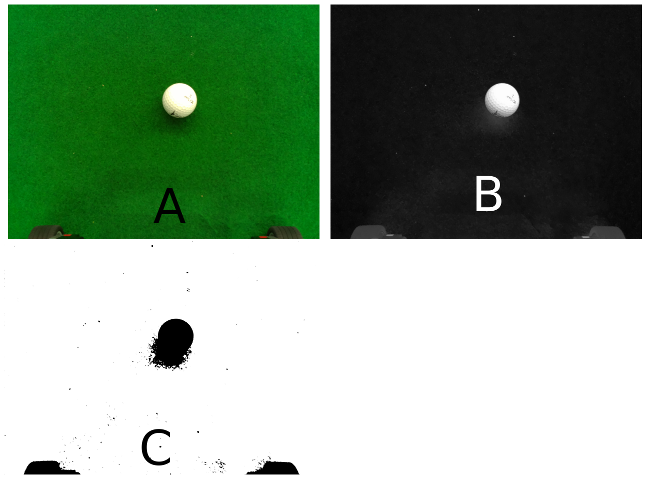


**Figure 3: original image(A) green channel(B) image to have blobs extracted from(C)**

Figure 3(A) is the golf ball against a green background, while figure 3(C) is the green channel of Figure 3(A). The threshold operation turns all pixels below a certain intensity white, and those above that intensity black. Figure 3(C) is the image of the golf ball with the threshold operation applied.

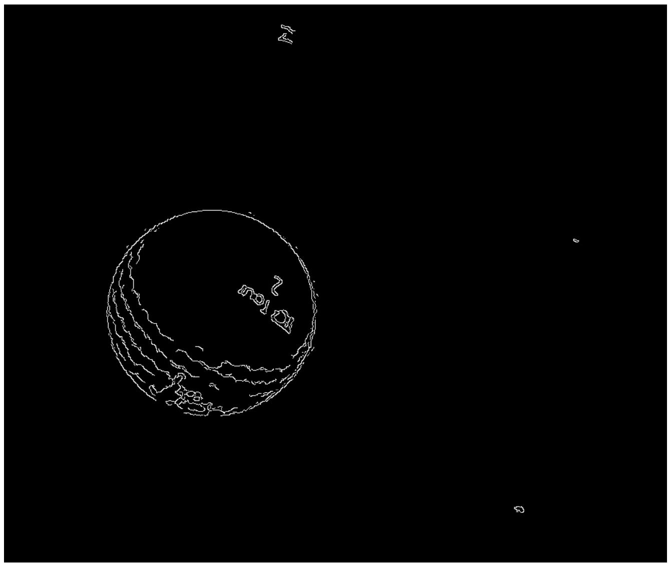


**Figure 4: Edge detection applied to the golf ball.**

Once the threshold operation has been applied to the image, blobs can be searched for. A blob is a data structure representing a cluster of pixels forming a feature, or unique shape which allows the feature to be easily distinguished from the rest of the image. In figure 3(C) three blobs would be detected. Blobs also contain their size, so we can check that against what we expect the size to be. In most cases by the design of our system, this will be enough to know we have detected a golf ball.

Liera et al.[3] uses their thermal image in a different way. Rather than looking for things based on their temperature, they scan for edges of temperature. It is likely that the reason for this is that we do not necessarily know the temperature of a boat or the surrounding ocean, but it is reasonable to assume the ocean does not have large changes in temperature across a relatively small region.

Edge scanning outputs similarly to the threshold operation applied by Israel.

Defined by Pratt[5]"Local discontinuities in image luminance from one level to another are called luminance edges." Since this is the only type of edge we will be using, we will refer to them as edges. Leira et al. does not go into detail about the exact edge finding algorithm they used. Edges are generally found by applying a blur to the image, and analyzing gradients in the blurred image. Once the gradients are analyzed, Edges are curves which are perpendicular to the gradient of the blurred image. Pratt[5] goes into further detail about how edges are detected. Figure 4 depicts the golf ball from earlier with edge scanning applied to it.

Leira et al.[3] goes a step farther classifying the blobs found as human, small boat, big boat, false positive, or noise. The classification is done using a combination of the average temperature of the blob, and the size. When taking the average temperature, they use the location of the blob to identify a bounding box around it, which includes only that blob. They then take the average of the temperature measurements within the bounding box. At this step, they do not consider blobs which are not entirely in the frame. For example, in figure 3(C) blobs detected in the lower right and left corners would have been removed from consideration since they are touching the edge of the frame.

## 5. SIGNAL FILTERING

Scanning, and classifying blobs takes place in a single frame. Tracking objects, by necessity takes place in multiple frames. This creates the problem of, how do we know if something we have identified in a previous frame is actually the same thing as in the current frame? On a short time scale the answer to this is through the use of signal filtering.

### 5.1 Overview

Signal filtering in general is used to remove unwanted components. When tracking an object from a UAV, the thermal camera is not in an ideal environment. It is subject to the vibrations of the aircraft, as well as the intentional motions of the aircraft that are necessary to maintain attitude. Signal filtering attempts to correct for these known, and unknown changes to the image. This is better illustrated in figure 5. Signal filtering could be used to confirm that the octocat in the current frame is indeed the same octocat as was detected in the previous frame.

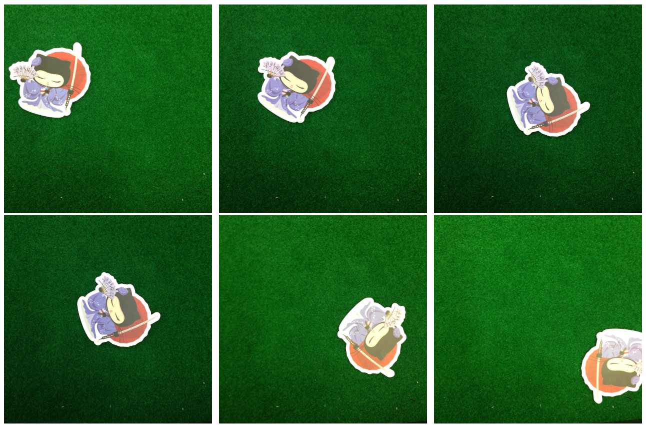


**Figure 5: Time step images of an adorable octocat moving.**

### 5.2 Algorithm

The signal filter we will explore is called the Kalman Filter. The Kalman filter was originally(1960s) met with a high degree of skepticism, forcing Rudolf Kalman to publish it in a mechanical engineering journal.[8] The Kalman Filter is now commonly used in tracking for interactive computer graphics and UAVs such as guided missiles. It was even used aboard the space shuttle! It is used in these applications because it is very good at correcting inaccurate measurements in real time. As for missiles, it helps a missile stay pointed at another missile it may be trying to destroy.

When we talk about the state of a system we are talking about its properties at the time it is being examined. The discreet Kalman filter is a version of the Kalman filter where the state is estimated at discrete points in time.[7] From now on, we will be referring to the discrete Kalman filter as the Kalman filter.

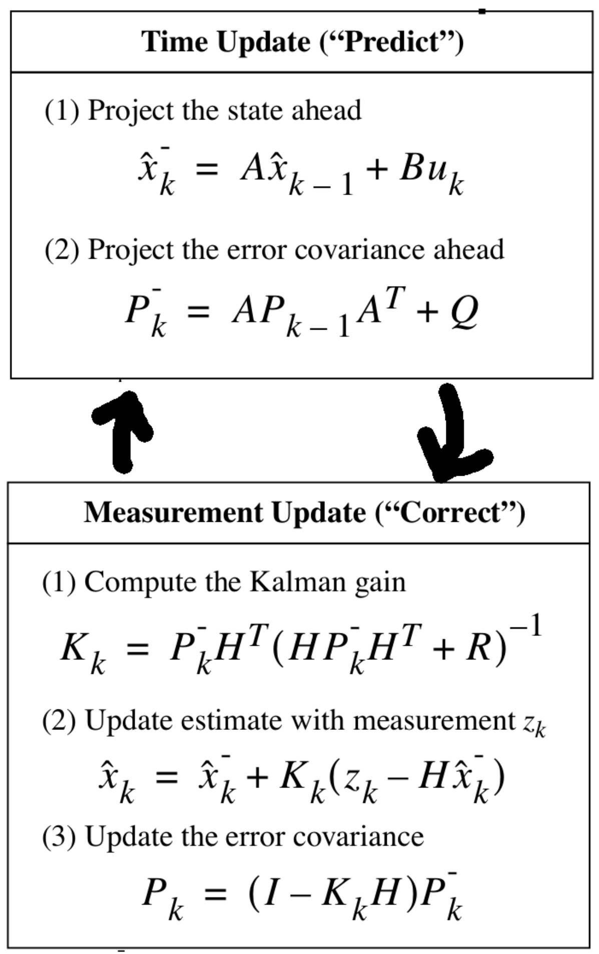
### 5.3 Explanation

The Kalman filter uses a form of feedback control. It predicts what the state should be, using measurements to correct the state.

The basic procedure is:

- Time Update (Predict)
- Measurement Update (Correct)

During the time update, the next measurement is predicted using the previous measurement. Also, the predicted error in the predicted measurement is updated. Next, during the Measurement update, the actual measurement updates the expected measurement, and the actual error updates the expected error. This is depicted with the actual equations in Figure 6.



Time Update ("Predict")
(1) Project the state ahead
$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$
(2) Project the error covariance ahead
$P_k^- = AP_{k-1}A^T + Q$
Measurement Update ("Correct")
(1) Compute the Kalman gain
$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$
(2) Update estimate with measurement $z_k$
$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$
(3) Update the error covariance
$P_k = (I - K_k H)P_k^-$

**Figure 6: Complete picture of operations performed by the Kalman filter from [7]**

Due to the way the Kalman Filter is used, it has good time efficiency. In many cases it works similarly to finding a linear regression line from a live feed of possibly infinite data. Welsh and Bishop[7] provide a more comprehensive explanation of the Kalman filter.

## 5.4 Example

Because of the complexity of higher dimensional Kalman Filters, the implementation of it by Leira et al. is explained in the context of a popular new app called BPM. BPM[1](short for beats per minute) is an app for iPhone in which you tap a button, and the app tells you how many times you are tapping per minute. I worked in a limited capacity with the developers of this app to implement a Kalman filter so that the measurements of how quickly you are tapping could be more accurate. The implementation used in BPM is largely based upon those described by Welch and Bishop [7], so it has many similarities to Figure 6.

```
var X: Double = 1
var P: Double = 0.1
var Q: Double = 0.0001
var A: Double = 1
var R: Double = 0.00001

func predict() {
    X = A * X
    P = A * P + Q
}

func correct(measurement: Double) -> Double {
    predict()
    let K = P * (1 / (P + R))
    X = X + K * (measurement - X)
    P = (1 - K) * P
    return X
}
```

The variable X represents the state current state of the system. In BPM, X is the number being returned. During each iteration, X is first modified by the `predict` function to the expected next value. The time since last tapped is passed into the function `correct`, and used to update X. In Leira et al.[3] X represents the location and speed of the object being tracked.

The variable P represents our confidence in the accuracy of the values of X. We chose the initial P, but subsequent iterations modify P based upon the consistency of the values of X. At the start of an iteration, the `predict` function uses constant values to modify the expected error in the next measurement. Next, P is updated taking into account the difference between the measurement and prediction.

The local variable K in the `correct` function is refereed to as the Kalman gain. The Kalman gain determines how much the Kalman filter can change the value of the measurement. If we set K to 0, the `correct` function would always return only the prediction. This is apparent in the `correct` function in the pseudo code; when K = 0, `X = X + 0 * (measurement - X)`.

The variable Q represents a constant error. If we are certain about some amount of inaccuracy in the measurements, we can account for that here. For example, if while using the app, you are certain that users cannot tap within 10 BPM of the rate they are attempting, it may be appropriate to set Q to a higher value. Q cannot be set to 0, because if it was the predict step would be doing almost nothing, and the Kalman filter would cease to function.

The variable A relates the state of the system in the previous step to the next step. In our case, A is 1 because we expect that users of BPM are attempting to find a constant beat. Leira et al use A to link the speed of the object being tracked with its predicted location. This can be done in higher dimensional Kalman Filters because in higher dimensions, A is a matrix. In higher dimensions, `X = A * X` in the predict function lets us have interdependence between the variables being filtered, when relating the state of the system to the previous state.

The variable R impacts the speed the system responds to change. So suppose you are tapping at 50 BPM, and you change tempo to 100 BPM. A high R value would allow the system to more quickly adapt to the higher beat rate. In this case a low R is desired because of how BPM is typically used.

The Kalman filter also has variables which correct for intentional changes to the system, which are not present in BPM. The modification to include intentional changes to the system be to the `predict()` function, we would add to X the intentional change. In Liera et al. this variable is used to correct for the motion of the aircraft.

It should be noted that carefully choosing each of these variables can lead to very robust filtering, although it can still work with less careful choices. This is because the Kalman filter updates its own predicted accuracy, making it an incredibly robust algorithm. This was apparent in the results produced by Leira et al. One of their major comments was that their vision system was upscaling the framerate, resulting in large amounts of sequential duplicate measurements being passed to the Kalman filter. This strongly effected performance in the field, but despite the handicap their results were still positive.

## 6. RESULTS

Liera et al. successfully detected 99.6% of the objects they were attempting to detect. Classification was also successful, correctly classifying 93.3% of of those detected. Also they succeeded in tracking 54 of the 64 times it initialized the tracking; only 5 of the initializations not actually tracking an object. [3]

Israel tested their system im May and June of 2011 with 15 field campaigns, covering 70.77 hectare at different times of day, and different weather conditions. It was successful when weather and lighting conditions were good. It produced limited results in sub optimal lighting. Altitudes when searching for fawn were between 30 m and 50 m. At the upper bound, fawns were sometimes missed even in the best conditions. They were forced to fly at the lower bound when conditions were not good. Vegetation heights were very low during this time, due to dry weather, so few fawns were killed that year. During the campaigns they were able to identify 14 fawns, 3 adult deer, five rabbits, one fox, and some smaller animals.[2]

## 7. CONCLUSION

In conclusion the these algorithms and techniques enable new and interesting uses for drones. Their lowering costs make them viable to be used in many new applications. They can save the lives of fawns, and keep grass from rotting due to fawn. Also their uses in the tracking of boats are great because they are cheap and can cover a wide area, accurately identifying and tracking many different sizes and shapes of boats. Computer vision aboard drones can save people a great deal of time on tasks covering a large area. Additionally signal filtering with the Kalman filter is highly effective at filtering the errors and noise from images taken by drones.[7] It is especially useful because it can take into account the drone's motions as well, correcting for errors in the data at many levels. It's robustness makes it ideal for use on a fast moving, vibrating platform like a drone.

## 8. REFERENCES

[1] Brian Mitchell, Z. L. BPM. https://github.com/bman4789/bpm. accessed 2016-04-24.

[2] Israel, M. A UAV-based roe deer fawn detection system. *International Archives of Photogrammetry and Remote Sensing 38* (2011), 1–5.

[3] Leira, F. S., Johansen, T. A., and Fossen, T. I. Automatic detection, classification and tracking of objects in the ocean surface from uavs using a thermal camera. In *Aerospace Conference, 2015 IEEE* (2015), IEEE, pp. 1–10.

[4] OpenCV. http://opencv.org/. accessed 2016-04-5.

[5] Pratt, W. K. Digital image processing, new-york. *NY: John Wiley and Sons* (1991).

[6] Ward, S., Hensler, J., Alsalam, B., and Gonzalez, L. F. Autonomous UAVs wildlife detection using thermal imaging, predictive navigation and computer vision.

[7] Welch, G., and Bishop, G. An introduction to the kalman filter. university of north carolina, department of computer science. Tech. rep., TR 95-041, 1995.

[8] Wikipedia. The kalman filter — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Rudolf_E._K%C3%A1lm%C3%A1n. Online; accessed 2016-04-5.