12-2015

Aircraft Detection and Tracking Using UAV-Mounted Vision System

Yan Zhang

AIRCRAFT DETECTION AND TRACKING USING UAV-MOUNTED VISION SYSTEM

A Thesis

Submitted to the Faculty

of

Embry-Riddle Aeronautical University

by

Yan Zhang

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

December 2015

Embry-Riddle Aeronautical University

Daytona Beach, Florida

AIRCRAFT DETECTION AND TRACKING USING UAV-MOUNTED VISION SYSTEM

by

Yan Zhang

A Thesis prepared under the direction of the candidate's committee chairman, Dr. Jianhua Liu, Department of Electrical, Computer, Software & Systems Engineering, and has been approved by the members of the thesis committee. It was submitted to the School of Graduate Studies and Research and was accepted in partial fulfillment of the requirements for the degree of Master of Science in Electrical and Computer Engineering.

THESIS COMMITTEE

[illegible] 12/1/2015

Chairman, Dr. Jianhua Liu

[illegible] 12/1/2015

Member, Dr. Richard Prazenica

Ilteris 12/01/2015

Member, Dr. Ilteris Demirkiran

[illegible]

Department Chair, Dr. Timothy A. Wilson
or Graduate Program Coordinator, Dr. Jianhua Liu

1 DEC 2015

Date

[illegible]

Dean of College of Engineering, Dr. Maj Mirmirani

12/2/15

Date

[illegible]

Vice Chancellor for Academics, Dr. Christopher D. Grant

12/2/15

Date

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Zhang, Yan MSECE, Embry-Riddle Aeronautical University, December 2015. Aircraft detection and tracking using UAV-mounted Vision System.

For unmanned aerial vehicles (UAVs) to operate safely in the national airspace where non-collaborating flying objects, such as general aviation (GA) aircraft without automatic dependent surveillance-broadcast (ADS-B), exist, the UAVs' capability of "seeing" these objects is especially important. This "seeing", or sensing, can be implemented via various means, such as Radar or Lidar. Here we consider using cameras mounted on UAVs only, which has the advantage of light weight and low power. For the visual system to work well, it is required that the camera-based sensing capability should be at the level equal to or exceeding that of human pilots.

This thesis deals with two basic issues/challenges of the camera-based sensing of flying objects. The first one is the stabilization of the shaky videos taken on the UAVs due to vibrations at different locations where the cameras are mounted. In the thesis, we consider several algorithms, including Kalman filters and particle filters, for stabilization. We provide detailed theoretical discussions of these filters as well as their implementations. The second one is reliable detection and tracking of aircraft using image processing algorithms. We combine morphological processing and dynamic programming to accomplish good results under different situations. The performance evaluation of different image processing algorithms is accomplished using synthetic and recorded data.

1. Introduction

1.1 Background of the thesis work

Unmanned aerial vehicles (UAVs) have great potential in various military and civil applications. To safely operate UAVs in the national aerospace where non-collaborating flying objects, such as general aviation (GA) aircraft without automatic dependent surveillance-broadcast (ADS-B), exist, the UAV's ability to "see" these objects should be at least at an equivalent level to that of human pilots.

To promote the research of this "seeing" technology, the National Aeronautics and Space Administration (NASA) organized a NASA competition called Unmanned Aircraft Systems (UAS) Airspace Operations Challenge (AOC) (Development Projects INC, 2013). Embry-Riddle Aeronautical University (ERAU) formed a team to participate in this competition, and we started the thesis work to serve the ERAU team.

The UAS-AOC is focused on demonstrations of some of the key technologies that will make integration of UAS into the National Airspace System (NAS) possible. One of the most difficult technical problems is to ensure safe separation with neighboring non-cooperative aircraft that do not broadcast ADS-B messages. Though the competition was cancelled later, the research on this topic continued due to its importance.

Many technologies including radar (Moses, 2013) and computer vision (Rozantsev, 2009) have been researched to improve the sense and avoid ability for UAVs. This thesis focuses on a potentially feasible and affordable way to address the problem of detecting and tracking uncooperative aircraft using cameras mounted on a UAV, a vision system. Such a system provides advantages including light weight and low power consumption compared to active sensors like radar and lidar (Zarandy, Zsedrovits, Nagy, Kiss, & Roska, 2012).

The diagram of a vision-based sense and avoid system for UAVs (Zarandy, Zsedrovits, Nagy, Kiss, & Roska, 2011) is illustrated in Fig. 1.1. The system works as follows. First, the images are captured using the camera block at given time intervals. Then the captured images are passed to an image processing block. After that, the decision on whether an aircraft is detected or not is made. Once an aircraft is detected and tracked, the aircraft's position information as referenced to each image is collected in the Data Acquisition block. The coordinate information about the detected aircraft can be obtained by combining information from onboard Inertial Navigation System (INS), Global Positioning System (GPS), and the local position of detected aircraft relative to the image from the Image Processing block. The other parts of the system are related to collision detection and avoidance control.

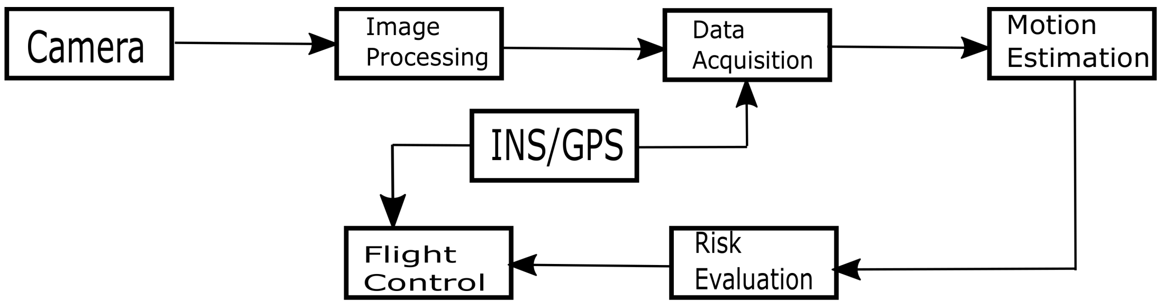


Figure 1.1. Diagram of the vision-based sense and avoid system.

This thesis concentrates on the Image Processing block to acquire reliable detection and tracking of an aircraft. Specifically, we address two issues/challenges. The first is that the image sequence captured from the camera is usually not stable due to the shaky platform of the UAV. To handle this problem, we consider several algorithms, including the Kalman and particle filters, for video stabilization. The second is the detection and tracking of an aircraft. We combine morphological processing and dynamic programming to accomplish good results under different situations. The performance evaluation of different image processing algorithms is accomplished using synthetic and recorded data.

1.2 Literature review

Here we first provide a brief literature review about image stabilization. In (Fergus, Singh, Hertzmann, Roweis, & Freeman, 2006), the in-plane camera rotation is neglected and the camera motion is estimated using a proposed blur kernel estimation algorithm. Then they apply a deconvolution algorithm to correct the blurry image under the assumption that the camera motion is uniformly distributed over the whole image.

Lin et al. (Lin, Hong, & Yang, 2009) present a stabilization system using the modified proportional integrated (PI) controller to remove the shaking from the captured videos while maintaining the panning motion of the camera. The motion compensation vector estimated in the paper is utilized to control the movement of the camera platform through a PI controller. In (Matsushita, Ofek, Ge, Tang, & Shum, 2006), the motion inpainting is implemented to enhance the quality of the stabilized image sequences.

Particle filter (Mohammadi, Fathi, & Soryani, 2011) and Kalman filter (Song, Zhao, Jing, & Zhu, 2012) for video stabilization are particular interesting as those algorithms are applied in a wide range of fields (Zhou, Chelleppa, & Moghaddam, 2004). Also, these algorithms have been developed over years and proven to be reliable (Orlande et al., n.d.).

Now we consider target detection. Zarandy et al. (Zarandy et al., 2011) present a way to detect and calculate the position of known-size aircraft. In the paper, FlightGear, a flight simulator, is used to produce the simulated aerial aircraft images. These images are then transmitted through Simulink to Matlab where the image processing algorithms are applied. The proposed algorithm is completed in two main steps. In the first step, the entire image is handled as a whole and then the region of interest is extracted and processed in the second step. However, this method is only effective in detecting the intruder aircraft in daylight situations when the cloud contrast is medium or small. In complex situations where the contrast of the clouds is high and the image is cluttered with obstacles, the proposed algorithm is not able to detect the aircraft without prior information.

Jaron & Kucharczyk (Jaron & Kucharczyk, 2012) describe two detailed vision system prototypes (a ground tracking and onboard detection and tracking system) to solve the problem of positioning and detection with cameras only. An object identification algorithm was developed and its position was estimated in the ground tracking system. On the onboard system, a FAST (Features from Accelerated Segment Test) feature detection and extraction algorithm is implemented for position estimation and collision detection. This method, however, has not been implemented on hardware and a performance test is needed before being used in real-world applications.

In (Shah, 2009), the author attempts to find the size and location of an obstacle in real world applications by generating a 3D world model from a 2D image received by cameras. The idea behind this is to use one camera mounted on a UAV to detect obstacles by means of feature points. Then the UAV flies around it in a circular path while capturing the feature points at the same time. The advantage of such an idea is that only one camera is needed. However, disadvantages, such as flying around obstacles, make it hardly applicable in applications of sense and avoid.

In (Gaszczak, Breckon, & Han, 2001), the authors present an approach for automatic detection of vehicles based on cascade Haar classifiers with secondary information in thermal images. The presented results show successful detection under varying conditions with minimal false detections. However, the algorithm must be improved to detect aircraft in the real world. The improvements can be as follows. First, the intruding aircraft detection performance at a long distance should be improved. Secondly, since the Haar classifier needs to train hundreds of aircraft images with different angles of view, prior information about the airplane like model and size should be known.

Hajri (Hajri, 2012) provides the preliminary process for target detection and position estimation. The article explores the computer vision detection algorithms. The first algorithm uses edge detection and image smoothing possesses to achieve a high detection rate, yet it exhibits high false alarm rates in highly cluttered image environments at the same time. The other approach is to use morphological filters and color-based detection. This method works effectively with the prior information of the UAV color patch. Hence, it exhibits low detection rates in low lighting condition.

A multi-stage detection method is developed in (Dey, Geyer, Singh, & Digioia, 2009). The approach starts with the morphological filter that looks for high contrast regions in the image that are likely to be aircraft. Next a classifier that has been trained on positive and negative examples has been used. Finally, it tracks the candidates over time to remove false detections. The results of the proposed algorithm demonstrate that it can achieve a high detection rate at a long distance.

Carnie et al. (Carnie, Walker, & Corke, 2005) combine the operation of morphological filter and dynamic programming to detect small aircraft in images with poor signal to noise ratios. The results demonstrate the ability to detect distant objects even in the presence of heavy cloud clutter.

2. Camera calibration

Although camera calibration is not the main focus of this thesis, it is an integral part of the work for vision-based sense and avoid. Hence, we present the work that we have performed on this topic here. In the later chapters, we will assume the camera(s) is calibrated.

2.1 Pinhole model

In this thesis, a pinhole model camera is used for all calculations regarding size, position, and distance estimation, as well as in video stabilization. This simplified model assumes that there exists an opaque wall with only one small hole in the center allowing only one ray of light to pass at a time. The ray then is projected onto the image plane which is at the same distance as the focal length of a camera from the aperture wall, as shown in Fig. 2.1. The advantage of the pinhole model is that the height of the object on the image plane is relative to only one parameter. As it can be seen from Fig. 2.1, the relation between the height of the object and the height of its projection on the image plane is formulated as

$$\frac{f}{h} = \frac{Z}{S},$$

where f is the focal length of the camera, the distance between the image plane and the pinhole, h is the height of the projected object on the image plane with regard to the optical axis, Z is the distance between the object and the pinhole, and S is the height of the object. The intersection of optical axis with the image plane in Fig. 2.1 is called the principal point.

The above model provides an easy way to calculate the distance between the camera and the object, given S and h which can be obtained via image processing. But in reality, the camera is far from the pinhole model. Ideally, the principal point should be placed exactly in the center of the image plane. However, this can never be true because of manufacturing imperfections. Therefore, in order to acquire the precise position of the object, the camera calibration is needed.

2.2 Principle of camera calibration

Image acquisition processes usually comprise one or more digital cameras, and in reality, most cameras are not ideal pinhole models. Thus camera calibration, an essential process for constructing real world models and interacting with real world

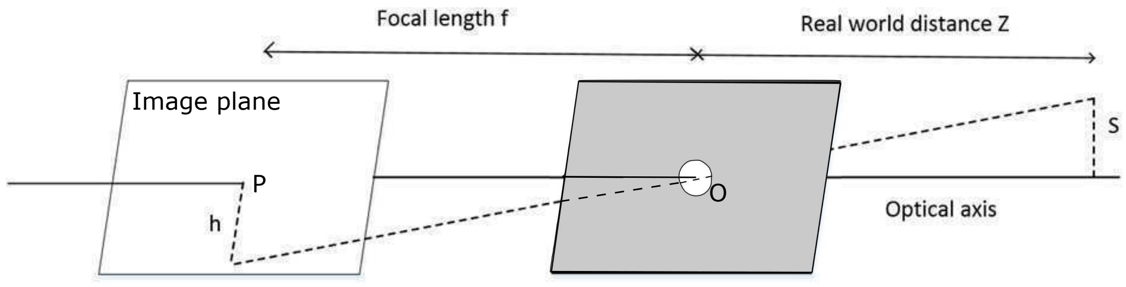


Figure 2.1. Pinhole camera model.

coordinates (Hruska, Lancaster, Harbour, & Cherry, 2005), is needed. Camera calibration is used basically to find a number of internal and external parameters that describe the camera, as detailed below.

An alternative pinhole model, which is more applicable to camera calibration, is given in Fig. 2.2. The reference point for the homogeneous coordinates is O, the same as the pinhole point shown in Fig. 2.1. The image plane is mirrored to the right side of the pinhole point since it does not change the projection point. (x_w, y_w, z_w) denotes the world coordinates while the image coordinates are represented by (x_i, y_i, z_i). W_c is the point that the optical axis intersects with the world plane. The center of the image plane, P, is the principal point. And z_i equals to f since the principal point is the reference point. Fig. 2.2 assumes that the two coordinate vectors are homogeneous. Thus,

$$\frac{x_w}{x_i} = \frac{y_w}{y_i} = \frac{z_w}{z_i},$$

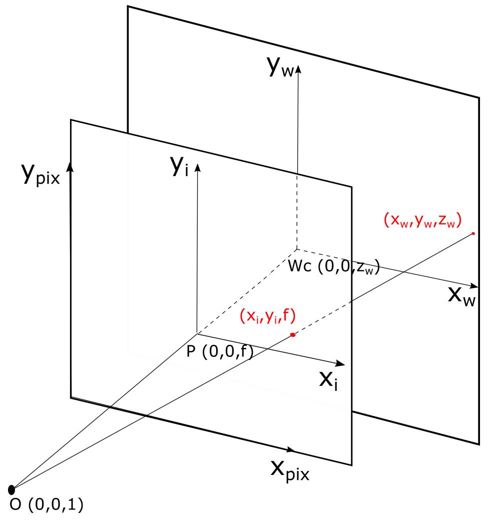


Figure 2.2. Projection coordinates.

which leads to $x_i = f\frac{x_w}{z_w},\ y_i = f\frac{y_w}{z_w}$, where x_i and y_i are both distances from the image center.

To transform from the object lengths to pixels, scaling factors k_x and k_y for x, y directions are defined, respectively. The unit of the scaling factor is distance per pixel. Also, the coordinates for the principal point P on x_{pix} and y_{pix} axes are defined to be (x_0, y_0) in pixels. Therefore, the pixel coordinates for the same projection point are designated as (x_p, y_p), which are formulated as $x_p = x_0 + k_x x_i,\ y_p = y_0 + k_y y_i$. Therefore, (x_p, y_p) is modified as

$$x_p = x_0 + k_x f \frac{x_w}{z_w},$$
$$y_p = y_0 + k_y f \frac{y_w}{z_w}.$$

Moreover, the image plane often is a parallelogram instead of a rectangle. Thus there is a parameter s to correct the skewness.

The transformation from the world coordinates to the a new coordinates can be formulated as

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} fk_x & s & x_0 \\ 0 & fk_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix}, \tag{2.1}$$

which leads to

$$x_p = \frac{u}{w},$$
$$y_p = \frac{v}{w}.$$

The calibration matrix is represented by

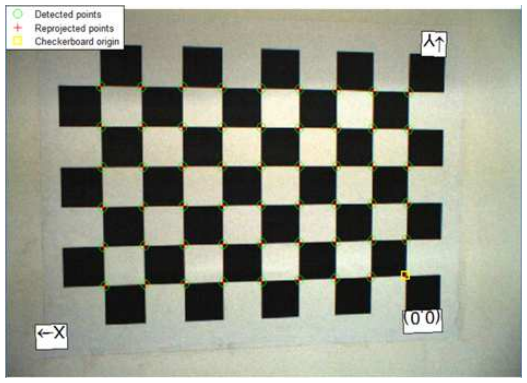
$$M = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.2}$$

where $f_x = fk_x$ and $f_y = fk_y$. f_x and f_y are also referred to as focal length in x and y directions with pixel units. Note the elements in the calibration matrix are intrinsic parameters.

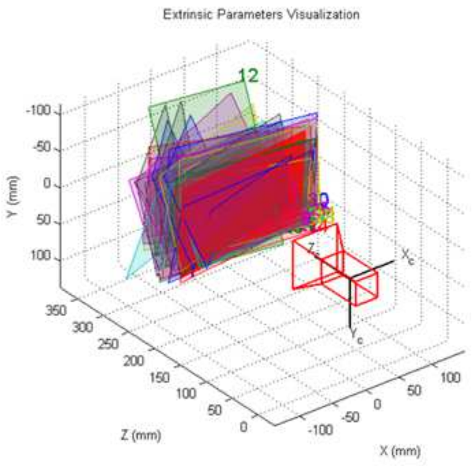
The extrinsic parameters that need to be handled in the calibration process are the lens distortions. The major lens distortions are radial and tangential distortions. Radial distortion is caused by the inappropriate shape of the lens and tangential distortion depends on the accuracy of the aligning lens with the image plane. In this thesis, it is assumed that five distortion coefficients are extracted since higher order distortion lenses are rare (Tsai, 2003). The lens calibration vector is given as

$$d = \begin{bmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{bmatrix}^T.$$

The variables k_1, k_2 are the coefficients of radial distortion, p_1, p_2 are the coefficients of tangential distortion, and k_3 is zero unless a fish-eye is used. So in total, nine parameters are required to be obtained to calibrate the camera.



(a) Picture feature extraction.

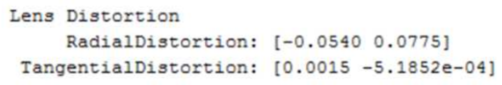


(b) MATLAB simulated viewpoints.

Figure 2.3. Calibration process.

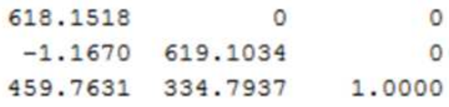
2.3 Camera calibration implementation

Based on what has been introduced above, there are nine parameters (not including k_3) that need to be found through calibration process. Such a complicated process is made easy with the MATLAB calibration toolbox. The toolbox processes many pictures of different point of views from the same object and outputs the camera parameters. The tested object should have a distinctive and repeatable pattern so that it would be easy to locate and track on the image. Therefore, a chessboard is often used in the calibration since it has a repeatable pattern with black and white squares. Furthermore, the more pictures taken from the various views of the object, the more accurate the calibration is. So for the calibration, many pictures, say, 30, are taken from different views of the chessboard and then processed with the camera calibration toolbox in MATLAB.



```
Lens Distortion
     RadialDistortion: [-0.0540 0.0775]
 TangentialDistortion: [0.0015 -5.1852e-04]
```

(a) Lens calibration.



```
 618.1518         0         0
  -1.1670  619.1034         0
 459.7631  334.7937    1.0000
```

(b) Calibration matrix.

Figure 2.4. Calibration results from MATLAB.

Fig. 2.3a demonstrates a chessboard in the feature extraction window in the MATLAB calibration toolbox and Fig. 2.3b shows the simulated process of taking different pictures. Meanwhile, the calibration matrix and the four distortion parameters of the camera are computed. An example of the lens distortion and calibration matrix for a GigE camera (IDS, n.d.) is displayed in Fig. 2.4a and Fig. 2.4b. Note that the matrix in Fig. 2.4b is the transpose of the calibration matrix M in Eq. (2.2). With the calibrated parameters, the real world homogeneous position of a known size object can be easily computed according to Eq. (2.1). For non-homogeneous coordinates, they can be transformed to homogeneous coordinates using translation and rotation matrices.

3. Video stabilization

A stable video is essential to achieve good target detection. The problem of stabilizing video is one of the applications of state estimation. Therefore, before diving into the details of video stabilization, let us discuss the theory and methods of state estimation.

3.1 State estimation theory

State estimation theory is commonly used in dynamic system applications such as signal processing, computer vision, object tracking, etc. The evolution of those dynamic systems is determined by the state of the systems, which are not directly observable, and the inputs to the systems. The observed data, the measurement, is related, to some extent, to the system state and can be leveraged to infer the actual states (Haykin, 2009). Usually, the system is modeled as a discrete system since it is not required to know the system data all the time. In the meanwhile, the measurement at every time interval is available to the observer. The diagram of the system state transition and measurement is given in Fig. 3.1.

In Fig. 3.1, the system is represented by the hidden Markov model (HMM) and the discrete state is denoted as $\boldsymbol{x}_n$, where n is the time step. The Markov model is a random process that features the transition from one state to another. In the Markov model, the current state is only dependent on the system's previous state instead of

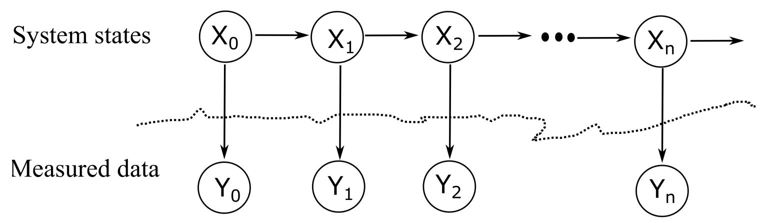


Figure 3.1. State transition.

the entire past state sequence. The available measurement at time step n for the system is denoted as $\boldsymbol{y}_n$.

3.2 Models

Two models are required to describe the state evolution and the measurement of a dynamic system. Under the assumption that the current state $\boldsymbol{x}_n$ evolves only from its previous state $\boldsymbol{x}_{n-1}$, the evolution equation is defined as

$$\boldsymbol{x}_n = f(\boldsymbol{x}_{n-1}) + \boldsymbol{w}_n, \tag{3.1}$$

where f is the system transition function which is also known as the evolution function and $\boldsymbol{w}_n$ is the system dynamic noise. Also, the measurement model is formulated as

$$\boldsymbol{y}_n = g(\boldsymbol{x}_n) + \boldsymbol{v}_n, \tag{3.2}$$

where g is the observation function and $\boldsymbol{v}_n$ represents observation noise.

To estimate the system's state, we need to use the above two models. In (Haykin, 2009), those models are addressed in four different cases.

The first case deals with the linear, Guassian models. In this case, the system evolution and observation functions are both assumed to be linear. The dynamic noise $\boldsymbol{w}_n$ and observed noise $\boldsymbol{v}_n$ are both additive, independent zero-mean Gaussian processes. The Kalman filter, which will be discussed later, is often used to handle this case.

The second case is the same as the first one, except that the system dynamic noise $\boldsymbol{w}_n$ and the observation noise $\boldsymbol{v}_n$ are now assumed to be additive, independent non-Gaussian processes. The tricky part for this case is the complicated non-Gaussian processes, which can be roughly approximated by the summation of several Gaussian processes. Thus a bank of Kalman filters may be used to solve the linear, non-Gaussian state estimation problem.

The third case is the same as the first case, except that the system evolution and observation functions are nonlinear. The nonlinear model issue can be addressed by two different solutions called local approximation and global approximation, respectively. The extended Kalman filter is an example of the local approximation, where the localized estimates are assumed to be linear. For the global approximation, the states and measurements are regarded to be related in a tractable mathematical way so that the approximation boils down to solving a mathematic problem. The particle filter is one of the methods for global approximation.

The fourth case deals with the nonlinear, non-Gaussian models. In this case, the system evolution and observation functions are both nonlinear, and the system dynamic and observation noises are not only non-Gaussian, but also may not be additive. Currently, the problem can be resolved by a particle filter.

3.3 The Baysian filter

The problem of the discussed models can be tackled by a recursive Baysian filter, such as the Kalman filter or a particle filter (Orlande et al., n.d.). The recursive Baysian filter is a process to estimate the probability density function (pdf) using the up-to-date measurements and the mathematical models. The filter is recursive because a new estimation is produced whenever a new measurement is obtained. Instead of processing the whole batch of data, the recursive method makes use of the current measurement, previous system state, and system models to estimate the current state, which is computational efficient in real time (Arulampalam, Maskell, Gordon, & Clapp, 2002). The Baysian filter provides a general framework for sequential state estimation.

As can be seen from Fig. 3.1, at each time step, there will be a hidden updated internal state and a new observable measurement. A filter can be repeatedly applied to solve the posteriori pdf $p(\boldsymbol{x}_n|\boldsymbol{y}_{0:n})$ given all the the observations along with the assumption of the initial pdf $p(\boldsymbol{x}_0|\boldsymbol{y}_0) = p(\boldsymbol{x}_0)$. In general, two steps are involved in the process and they are referred as state prediction and update.

In the prediction step, the priori pdf of the state at time step n, $p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n-1})$, is obtained via the Chapman-Kolmogorov method (Perreault, 2012). Note that the $p(\boldsymbol{x}, \boldsymbol{y})$ is the simplified version of $p(\boldsymbol{x} \bigcap \boldsymbol{y})$ in this thesis.

$$\begin{aligned} p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n-1}) &= \int p(\boldsymbol{x}_n, \boldsymbol{x}_{n-1}|\boldsymbol{y}_{1:n-1}) d\boldsymbol{x}_{n-1} \\ &= \int p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1}, \boldsymbol{y}_{1:n-1}) p(\boldsymbol{x}_{n-1}|\boldsymbol{y}_{1:n-1}) d\boldsymbol{x}_{n-1} \\ &= \int p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1}) p(\boldsymbol{x}_{n-1}|\boldsymbol{y}_{1:n-1}) d\boldsymbol{x}_{n-1}. \end{aligned}$$

Note that we have used the Markovian property of the system in the above equations: if $\boldsymbol{x}_n$ is independent of $\boldsymbol{y}_{1:n-1}$, given $\boldsymbol{x}_{n-1}$, then $p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1}, \boldsymbol{y}_{1:n-1}) = p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})$. The pdf of the state transition $p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})$ can be inferred from Eq. (3.1) and the filtering distribution $p(\boldsymbol{x}_{n-1}|\boldsymbol{y}_{1:n-1})$ is given at time step $n-1$.

In the update step, the new measurement at time step n is used to calculate the posteriori pdf. Based on Bayer's rule, we have

$$p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n}) = \frac{p(\boldsymbol{x}_n, \boldsymbol{y}_{1:n})}{p(\boldsymbol{y}_{1:n})} = \frac{p(\boldsymbol{x}_n, \boldsymbol{y}_{1:n})}{\int p(\boldsymbol{x}_n, \boldsymbol{y}_{1:n}) d\boldsymbol{x}_n},$$

and

$$\begin{aligned} p(\boldsymbol{x}_n, \boldsymbol{y}_{1:n}) &= p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{y}_{1:n-1}) p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n-1}) p(\boldsymbol{y}_{1:n-1}) \\ &= p(\boldsymbol{y}_n|\boldsymbol{x}_n) p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n-1}) p(\boldsymbol{y}_{1:n-1}), \end{aligned}$$

where we have used $p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{y}_{1:n-1}) = p(\boldsymbol{y}_n|\boldsymbol{x}_n)$ because Eq. (3.2) shows that the current measurement is only related to the current state.

Hence,

$$p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n}) = \frac{p(\boldsymbol{y}_n|\boldsymbol{x}_n)p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n-1})}{\int p(\boldsymbol{y}_n|\boldsymbol{x}_n)p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n-1})d\boldsymbol{x}_n}.$$

Thus, the priori density is modified using the current measurement to get the required posteriori density.

The basic procedure of the Baysian filter consists of two stages. Due to the noise variations in the system, it is difficult to solve for the exact posterior probability density function which is also addressed as the optimal Baysian solution. However, the optimal Baysian solution can be achieved by applying restrictions on the system model and noise. This is a situation where the state and measurement systems are assumed to be linear systems with zero mean Gaussian noises. One example from this category is the Kalman filter. However, in most cases where the linear, Gaussian model is not suitable for the system, another approach, such as the paticle filter, that approximates the probability density function, is utilized. In the sequel, the Kalman and particle filters are studied and implemented to compare the applicability and efficiency of the two different methods.

3.4 The Kalman filter

The Kalman filter is a linear recursive algorithm generating least square error solutions (Orlande et al., n.d.). The Kalman filter finds the best current state estimate based on the current measurement, previous state estimate, and mathematical models using the least square optimization method, which produces more accurate results than just one single observed measurement. Whenever a new measurement comes in, the filter updates the new estimate so that the error estimation vector between estimated states and the real states is minimized. The recursive manner and computational efficiency of Kalman filter make it useful in a system where the estimation accuracy and time constraints are highly required. For this reason, the Kalman filter is widely applied in aviation fields such as guidance, navigation, and control.

3.4.1 Derivation of the Kalman filter

The Kalman filter is based on the assumption of linear discrete state-space and Gaussian models, which update the state each time a new observation data is added. The two models for the Kalman filter can be rewritten below. The state-space transition model of Eq. (3.1) is

$$\boldsymbol{x}_{n+1} = \boldsymbol{A}_n \boldsymbol{x}_n + \boldsymbol{w}_n,$$

where $\boldsymbol{A}_n$ is the system state transition matrix at time step n, $\boldsymbol{x}_{n+1}$ and $\boldsymbol{x}_n$ are the system states at time $n+1$ and time n respectively, and $\boldsymbol{w}_n$ is the system dynamic noise at time n and assumed to be independent zero-mean additive Gaussian, as discussed in a previous section. The random noise distribution is $\boldsymbol{w}_n \sim N(\boldsymbol{0}, \boldsymbol{Q}_n)$, where the $\boldsymbol{Q}_n$ is the variance matrix.

The observation model of Eq. (3.2) is

$$\boldsymbol{y}_n = \boldsymbol{H}_n \boldsymbol{x}_n + \boldsymbol{v}_n,$$

where $\boldsymbol{y}_n$ is the actual observed measurement at time n, $\boldsymbol{H}_n$ is the observation matrix at time n, and $\boldsymbol{v}_n$ is the observation noise at time n which is also modeled as independent zero-mean additive Gaussian. Like the system noise, the measurement noise is $\boldsymbol{v}_n \sim N(\boldsymbol{0}, \boldsymbol{R}_n)$, where the $\boldsymbol{R}_n$ is the variance matrix.

Since it is an implementation of the Baysian filter, there are basically two steps involved in the Kalman filter. The first step is the prediction process, which takes the previous estimated states and then outputs the predicted current states based on the given system transition function. This is also called priori estimation process. The second step is to update the prediction, priori estimation, given the current observed measurements to get more accurate estimation. The updated estimation is also called posteriori estimation. In the sequel, we provide an easy to follow derivation of the Kalman filter.

The priori estimation of the state $\boldsymbol{x}_n$ is denoted as $\boldsymbol{x}_n^-$, which is $\boldsymbol{x}_n^- = \boldsymbol{A}_{n-1}\boldsymbol{x}_{n-1}$. The priori estimation $\boldsymbol{x}_n^-$ leads to the estimated measurements, which is formulated as $\hat{\boldsymbol{y}}_n = \boldsymbol{H}_n\boldsymbol{x}_n^-$. The measurement error is defined as the difference between observations and estimated measurements, which is calculated as $\boldsymbol{e}_n = \boldsymbol{y}_n - \boldsymbol{H}_n\boldsymbol{x}_n^-$. With the measurements at time step n, the priori state estimation can be updated to posteriori state estimation which is referred as $\boldsymbol{x}_n^+$. The posteriori estimation is represented by the current priori estimation plus the weighted measurement error. The equation is illustrated below

$$\boldsymbol{x}_n^+ = \boldsymbol{x}_n^- + \boldsymbol{K}_n(\boldsymbol{y}_n - \boldsymbol{H}_n\boldsymbol{x}_n^-),$$

where $\boldsymbol{K}_n$ is a weighted matrix which is also known as the Kalman gain. $\boldsymbol{K}_n$ indicates how much the measurement error changes the estimation (Perreault, 2012).

During the Kalman filter implementation, two estimation errors are computed for the two stages. First, we calculate the priori estimation error, the difference between the actual state and the priori estimated state, which is represented as $\boldsymbol{e}_n^- = \boldsymbol{x}_n - \boldsymbol{x}_n^-$. Then the priori error variance matrix is computed as $\boldsymbol{P}_n^- = E[\boldsymbol{e}_n^- \boldsymbol{e}_n^{-H}]$. Similarly, we calculate the posteriori estimation error $\boldsymbol{e}_n^+ = \boldsymbol{x}_n - \boldsymbol{x}_n^+$, and the posteriori error variance matrix $\boldsymbol{P}_n^+ = E[\boldsymbol{e}_n^+ \boldsymbol{e}_n^{+H}]$. The Kalman filter outputs the estimated state that produces the least square error with the actual state. In this case, $\boldsymbol{K}_n$ should minimize the trace of the posteriori variance matrix $\boldsymbol{P}_n^+$. To do this, let us first formulate $\boldsymbol{e}_n^+$ as

$$\begin{aligned}
\boldsymbol{e}_n^+ &= \boldsymbol{x}_n - \boldsymbol{x}_n^+ \\
&= \boldsymbol{x}_n - (\boldsymbol{x}_n^- + \boldsymbol{K}_n(\boldsymbol{y}_n - \boldsymbol{H}_n \boldsymbol{x}_n^-)) \\
&= \boldsymbol{x}_n - (\boldsymbol{I} - \boldsymbol{K}_n \boldsymbol{H}_n)\boldsymbol{x}_n^- - \boldsymbol{K}_n \boldsymbol{y}_n \\
&= (\boldsymbol{I} - \boldsymbol{K}_n \boldsymbol{H}_n)(\boldsymbol{x}_n - \boldsymbol{x}_n^-) - \boldsymbol{K}_n \boldsymbol{v}_n \\
&= (\boldsymbol{I} - \boldsymbol{K}_n \boldsymbol{H}_n)\boldsymbol{e}_n^- - \boldsymbol{K}_n \boldsymbol{v}_n.
\end{aligned}$$

So we obtain

$$\begin{aligned}
\boldsymbol{P}_n^+ &= E[\boldsymbol{e}_n^+ \boldsymbol{e}_n^H] \\
&= E[((\boldsymbol{I} - \boldsymbol{K}_n \boldsymbol{H}_n)\boldsymbol{e}_n^- - \boldsymbol{K}_n \boldsymbol{v}_n)((\boldsymbol{I} - \boldsymbol{K}_n \boldsymbol{H}_n)\boldsymbol{e}_n^- - \boldsymbol{K}_n \boldsymbol{v}_n)^H] \\
&= E[(\boldsymbol{I} - \boldsymbol{K}_n \boldsymbol{H}_n)\boldsymbol{e}_n^- \boldsymbol{e}_n^{-H} (\boldsymbol{I} - \boldsymbol{K}_n \boldsymbol{H}_n)^H \\
&\qquad + \boldsymbol{K}_n \boldsymbol{v}_n \boldsymbol{v}_n^H \boldsymbol{K}_n^H - (\boldsymbol{I} - \boldsymbol{K}_n \boldsymbol{H}_n)\boldsymbol{e}_n^- \boldsymbol{v}_n^H \boldsymbol{K}_n^H - \boldsymbol{K}_n \boldsymbol{v}_n \boldsymbol{e}_n^{-H} (\boldsymbol{I} - \boldsymbol{K}_n \boldsymbol{H}_n)^H)].
\end{aligned}$$

To simplify the notation, let us make the following definition $\boldsymbol{P}_n^- = E[\boldsymbol{e}_n^- \boldsymbol{e}_n^{-H}]$ and $\boldsymbol{R}_n = E[\boldsymbol{v}_n \boldsymbol{v}_n^H]$, and observe that $E[\boldsymbol{e}_n^- \boldsymbol{v}_n^H] = \boldsymbol{0}$ and $E[\boldsymbol{v}_n \boldsymbol{e}_n^{-H}] = \boldsymbol{0}$. Therefore the posteriori variance matrix $\boldsymbol{P}_n^+$ is expressed as

$$\begin{aligned}
\boldsymbol{P}_n^+ &= (\boldsymbol{I} - \boldsymbol{K}_n \boldsymbol{H}_n)\boldsymbol{P}_n^- (\boldsymbol{I} - \boldsymbol{K}_n \boldsymbol{H}_n)^H + \boldsymbol{K}_n \boldsymbol{R}_n \boldsymbol{K}_n^H \\
&= \boldsymbol{K}_n (\boldsymbol{H}_n \boldsymbol{P}_n^- \boldsymbol{H}_n^H + \boldsymbol{R}_n)\boldsymbol{K}_n^H - \boldsymbol{K}_n \boldsymbol{H}_n \boldsymbol{P}_n^- - \boldsymbol{P}_n^- \boldsymbol{H}_n^H \boldsymbol{K}_n^H + \boldsymbol{P}_n^-.
\end{aligned}$$

To simplify the above equation, let

$$\boldsymbol{A} = \boldsymbol{H}_n \boldsymbol{P}_n^- \boldsymbol{H}_n^H + \boldsymbol{R}_n. \tag{3.3}$$

Thus, by using the quadratic expression, we have

$$\begin{aligned}
\boldsymbol{P}_n^+ &= (\boldsymbol{K}_n - \boldsymbol{P}_n^- \boldsymbol{H}_n^H \boldsymbol{A}^{-1})\boldsymbol{A}(\boldsymbol{K}_n - \boldsymbol{P}_n^- \boldsymbol{H}_n^H \boldsymbol{A}^-)^H \\
&\quad - \boldsymbol{P}_n^- \boldsymbol{H}_n^H \boldsymbol{A}^{-1} \boldsymbol{H}_n \boldsymbol{P}_n^- + \boldsymbol{P}_n^-.
\end{aligned} \tag{3.4}$$

As Eq. (3.4) indicates, to minimize the trace of $\boldsymbol{P}_n^+$, the term that contains $\boldsymbol{K}_n$ should be zero. So we have

$$\begin{aligned}\boldsymbol{K}_n &= \boldsymbol{P}_n^- \boldsymbol{H}_n^H \boldsymbol{A}^{-1} \\ &= \boldsymbol{P}_n^- \boldsymbol{H}_n^H (\boldsymbol{H}_n \boldsymbol{P}_n^- \boldsymbol{H}_n^H + \boldsymbol{R}_n)^{-1},\end{aligned}$$

which leads to

$$\boldsymbol{P}_n^+ = (\boldsymbol{I} - \boldsymbol{K}_n \boldsymbol{H}_n) \boldsymbol{P}_n^-.$$

Note that both $\boldsymbol{K}_n$ and $\boldsymbol{P}_n^+$ are expressed in terms of $\boldsymbol{P}_n^-$ since $\boldsymbol{H}_n$ and $\boldsymbol{R}_n$ are constant at time step n.

Next, let us calculate the priori estimation error variance matrix $\boldsymbol{P}_n^-$. Again, we first consider the error

$$\begin{aligned}\boldsymbol{e}_n^- &= \boldsymbol{x}_n - \boldsymbol{x}_n^- \\ &= \boldsymbol{A}_{n-1}\boldsymbol{x}_{n-1} + \boldsymbol{w}_{n-1} - \boldsymbol{A}_{n-1}\boldsymbol{x}_{n-1}^+ \\ &= \boldsymbol{A}_{n-1}\boldsymbol{e}_{n-1}^+ + \boldsymbol{w}_{n-1}.\end{aligned}$$

Hence,

$$\begin{aligned}\boldsymbol{P}_n^- &= E[\boldsymbol{e}_n^- \boldsymbol{e}_n^{-H}] \\ &= E[(\boldsymbol{A}_{n-1}\boldsymbol{e}_{n-1}^+ + \boldsymbol{w}_{n-1})(\boldsymbol{A}_{n-1}\boldsymbol{e}_{n-1}^+ + \boldsymbol{w}_{n-1})^H] \\ &= \boldsymbol{A}_{n-1}\boldsymbol{P}_{n-1}^+ \boldsymbol{A}_{n-1}^H + \boldsymbol{Q}_{n-1},\end{aligned}$$

where we have used $\boldsymbol{P}_{n-1}^{+} = E[\boldsymbol{e}_{n-1}^{+}\boldsymbol{e}_{n-1}^{+H}]$ and $\boldsymbol{Q}_{n-1} = E[\boldsymbol{w}_{n-1}\boldsymbol{w}_{n-1}^{H}]$ as well as $E[\boldsymbol{e}_{n-1}^{+}\boldsymbol{w}_{n-1}^{H}] = \boldsymbol{0}$ and $E[\boldsymbol{w}_{n-1}\boldsymbol{e}_{n-1}^{+H}] = \boldsymbol{0}$.

3.4.2 Algorithm of the Kalman filter

It is obvious from the above derivation that the Kalman filter can be calculated recursively to each newly acquired data. Assume that at time step n, the followings are given: the system observation matrix $\boldsymbol{H}_n$, system observation noise covariance matrix $\boldsymbol{R}_n$, and the real measurement $\boldsymbol{y}_n$. Also assumed given are: the system transition matrix $\boldsymbol{A}_{n-1}$, the system dynamic noise covariance matrix $\boldsymbol{Q}_{n-1}$, the posteriori estimation error covariance matrix $\boldsymbol{P}_{n-1}^{+}$, and the posteriori estimated state $\boldsymbol{x}_{n-1}^{+}$. Then, the optimal estimated state $\boldsymbol{x}_{n}^{+}$ can be calculated using the following steps.

1. Determine the priori estimation error covariance matrix $\boldsymbol{P}_{n}^{-}$ and predict the priori estimated system state $\boldsymbol{x}_{n}^{-}$:

$$\begin{aligned}\boldsymbol{P}_{n}^{-} &= \boldsymbol{A}_{n-1}\boldsymbol{P}_{n-1}^{+}\boldsymbol{A}_{n-1}^{H} + \boldsymbol{Q}_{n-1},\\ \boldsymbol{x}_{n}^{-} &= \boldsymbol{A}_{n-1}\boldsymbol{x}_{n-1}^{+}.\end{aligned}$$

2. Calculate the Kalman gain $\boldsymbol{K}_n$, update the optimal estimated state $\boldsymbol{x}_n^+$, and determine the posteriori estimation error covariance matrix $\boldsymbol{P}_n^+$:

$$\begin{aligned}
\boldsymbol{K}_n &= \boldsymbol{P}_n^- \boldsymbol{H}_n^H (\boldsymbol{H}_n \boldsymbol{P}_n^- \boldsymbol{H}_n^H + \boldsymbol{R}_n)^{-1}, \\
\boldsymbol{x}_n^+ &= \boldsymbol{x}_n^- + \boldsymbol{K}_n (\boldsymbol{y}_n - \boldsymbol{H}_n \boldsymbol{x}_n^-), \\
\boldsymbol{P}_n^+ &= (\boldsymbol{I} - \boldsymbol{K}_n \boldsymbol{H}_n) \boldsymbol{P}_n^-.
\end{aligned}$$

3.5 Particle filters

As introduced before, an approach to handle nonlinear, non-Gaussian system models is to employ particle filters. Particle filters are the best examples of the Monte Carlo method (Mackay, n.d.), a broad class of algorithms that repeatedly generate random samples to get the results. A particle filter is also known as the CONDENSATION (CONditional DENsity propagATION) algorithm, bootstrap filtering, interacting particle approximations, survival of the fittest, sequential importance sampling, and the sequential Monte Carlo approach (Doucet, Freitas, & Gordon, n.d.). A particle filter is a general and powerful method that can be applied in radar tracking, medical analysis, human machine interaction, image restoration, etc. Compared to the Kalman filter, particle filters do not require tight restrictions on system models. Thus, particle filters are more applicable in most cases.

The basic idea of the particle filter is to generate a set of random weighted samples in order to estimate the posteriori probability density function. The joint posteriori distribution of all the states is denoted as $p(\boldsymbol{x}_{1:n}|\boldsymbol{y}_{1:n})$, where $\boldsymbol{x}_{1:n}$ represents all the system states from the starting time up to current time step n. Correspondingly, all the measurements of the system, denoted as $\boldsymbol{y}_{1:n}$ are available to use. However, the actual sequential system states $\boldsymbol{x}_{1:n}$ are hidden from the observer because of the uncontrollable variables and noise in the system. Hence, it is very challenging to obtain the real posteriori pdf $p(\boldsymbol{x}_{1:n}|\boldsymbol{y}_{1:n})$. The way the particle filter does is to draw samples from the so-called importance density function which is designated as $q(\boldsymbol{x}_{1:n}|\boldsymbol{y}_{1:n})$ instead of the actual posteriori $p(\boldsymbol{x}_{1:n}|\boldsymbol{y}_{1:n})$. Then $p(\boldsymbol{x}_{1:n}|\boldsymbol{y}_{1:n})$ can be approximated by the summation of the weighted samples. The weighted samples are denoted by $\{\boldsymbol{x}^i_{1:n}, w^i_n\}$, where $\{\boldsymbol{x}^i_{1:n}, i = 1, ..., N_s\}$ is a set of sampled points from $q(\boldsymbol{x}_{1:n}|\boldsymbol{y}_{1:n})$. And the matching weight for each sample is illustrated by $\{w^i_n, i = 1, ..., N_s\}$. Usually, the weights are normalized such that $\sum_{i=1}^{N_s} w^i_n = 1$. Since the samples are from the importance density function, the weight for the ith sample can be calculated as

$$w^i_n = \frac{p(\boldsymbol{x}^i_{1:n}|\boldsymbol{y}_{1:n})}{q(\boldsymbol{x}^i_{1:n}|\boldsymbol{y}_{1:n})}. \tag{3.5}$$

Therefore, the posteriori pdf is approximated by

$$p(\boldsymbol{x}_{1:n}|\boldsymbol{y}_{1:n}) \approx \sum_{i=1}^{N_s} w^i_n \delta(\boldsymbol{x}_{1:n} - \boldsymbol{x}^i_{1:n}),$$

where $\boldsymbol{x}^i_{1:n}$ are samples generated from the importance density function $q(\boldsymbol{x}_{1:n}|\boldsymbol{y}_{1:n})$, and $\delta(\cdot)$ is the delta function.

3.5.1 Derivation of the particle filter

In a particle filter, the posteriori density function is approximated by the weighted samples from the importance density function $q(\boldsymbol{x}_{1:n}|\boldsymbol{y}_{1:n})$. We will prove that the particle filter is a recursive suboptimal solution to state estimation. Thus, given the approximation of $p(\boldsymbol{x}_{1:n-1}|\boldsymbol{y}_{1:n-1})$ and the new measurement $\boldsymbol{y}_n$ at time step n, $p(\boldsymbol{x}_{1:n}|\boldsymbol{y}_{1:n})$ can be computed. Every time a new observation is available, new samples are generated from the importance density function. The importance density function is extended as the following according to Bayer's rule

$$q(\boldsymbol{x}_{1:n}|\boldsymbol{y}_{1:n}) = q(\boldsymbol{x}_n|\boldsymbol{x}_{1:n-1}, \boldsymbol{y}_{1:n})q(\boldsymbol{x}_{1:n-1}|\boldsymbol{y}_{1:n-1}, \boldsymbol{y}_n).$$

To simplify the above expression, the importance density function $q(\boldsymbol{x}_{1:n}|\boldsymbol{y}_{1:n})$ can be chosen such that the factorization $q(\boldsymbol{x}_{1:n-1}|\boldsymbol{y}_{1:n-1}, \boldsymbol{y}_n) = q(\boldsymbol{x}_{1:n-1}|\boldsymbol{y}_{1:n-1})$, which means the current measurement has no effect on the system previous states. Then the modified equation is

$$q(\boldsymbol{x}_{1:n}|\boldsymbol{y}_{1:n}) = q(\boldsymbol{x}_n|\boldsymbol{x}_{1:n-1}, \boldsymbol{y}_{1:n})q(\boldsymbol{x}_{1:n-1}|\boldsymbol{y}_{1:n-1}). \tag{3.6}$$

Therefore, it can be inferred that the updated samples $\boldsymbol{x}^i_{1:n}$ are generated by combining the existing samples $\boldsymbol{x}^i_{1:n-1} \sim q(\boldsymbol{x}_{1:n-1}|\boldsymbol{y}_{1:n-1})$ with the samples $\boldsymbol{x}^i_n$ that are drawn from $q(\boldsymbol{x}_n|\boldsymbol{x}_{1:n-1}, \boldsymbol{y}_{1:n})$.

The weights assigned to each particle should also be updated once the new samples are produced, as shown below. First we have

$$p(\boldsymbol{x}_{1:n}|\boldsymbol{y}_{1:n}) = \frac{p(\boldsymbol{x}_{1:n}, \boldsymbol{y}_{1:n})}{p(\boldsymbol{y}_{1:n})},$$

$$p(\boldsymbol{x}_{1:n}, \boldsymbol{y}_{1:n}) = p(\boldsymbol{y}_n|\boldsymbol{x}_{1:n}, \boldsymbol{y}_{1:n-1})p(\boldsymbol{x}_{1:n}|\boldsymbol{y}_{1:n-1})p(\boldsymbol{y}_{1:n-1}),$$

$$p(\boldsymbol{y}_{1:n}) = p(\boldsymbol{y}_n|\boldsymbol{y}_{1:n-1})p(\boldsymbol{y}_{1:n-1}).$$

Thus

$$p(\boldsymbol{x}_{1:n}|\boldsymbol{y}_{1:n}) = \frac{p(\boldsymbol{y}_n|\boldsymbol{x}_{1:n}, \boldsymbol{y}_{1:n-1})p(\boldsymbol{x}_{1:n}|\boldsymbol{y}_{1:n-1})}{p(\boldsymbol{y}_n|\boldsymbol{y}_{1:n-1})}.$$

Applying Bayer's rule

$$p(\boldsymbol{x}_{1:n}|\boldsymbol{y}_{1:n-1}) = p(\boldsymbol{x}_n|\boldsymbol{x}_{1:n-1}, \boldsymbol{y}_{1:n-1})p(\boldsymbol{x}_{1:n-1}|\boldsymbol{y}_{1:n-1}),$$

we have

$$p(\boldsymbol{x}_{1:n}|\boldsymbol{y}_{1:n}) = \frac{p(\boldsymbol{y}_n|\boldsymbol{x}_{1:n}, \boldsymbol{y}_{1:n-1})p(\boldsymbol{x}_n|\boldsymbol{x}_{1:n-1}, \boldsymbol{y}_{1:n-1})}{p(\boldsymbol{y}_n|\boldsymbol{y}_{1:n-1})}p(\boldsymbol{x}_{1:n-1}|\boldsymbol{y}_{1:n-1}).$$

Under the assumption that the system follows a Markovian model, we have

$$p(\boldsymbol{x}_n|\boldsymbol{x}_{1:n-1}, \boldsymbol{y}_{1:n-1}) = p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})$$

. Furthermore, we have

$$p(\boldsymbol{y}_n|\boldsymbol{x}_{1:n}, \boldsymbol{y}_{1:n-1}) = p(\boldsymbol{y}_n|\boldsymbol{x}_n)$$

since the current measurement is merely dependent on the current states. Besides, at time step n, $p(\boldsymbol{y}_n|\boldsymbol{y}_{1:n-1})$ is considered to be constant. Hence we have

$$p(\boldsymbol{x}_{1:n}|\boldsymbol{y}_{1:n}) \propto p(\boldsymbol{y}_n|\boldsymbol{x}_n)p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})p(\boldsymbol{x}_{1:n-1}|\boldsymbol{y}_{1:n-1}), \tag{3.7}$$

where $\propto$ denotes proportional to.

By combining Eq. (3.5), Eq. (3.6), and Eq. (3.7), the weighting update can be computed using the following equation

$$w_n^i \propto w_{n-1}^i \frac{p(\boldsymbol{y}_n|\boldsymbol{x}_n^i)p(\boldsymbol{x}_n^i|\boldsymbol{x}_{n-1}^i)}{q(\boldsymbol{x}_n^i|\boldsymbol{x}_{1:n-1}^i, \boldsymbol{y}_n)}.$$

In practical applications, the estimation is refreshed at every time step. Thus it is realistic to set the importance density in a way that

$$q(\boldsymbol{x}_n^i|\boldsymbol{x}_{1:n-1}^i, \boldsymbol{y}_n) = q(\boldsymbol{x}_n^i|\boldsymbol{x}_{n-1}^i, \boldsymbol{y}_n),$$

which means the density function is now only dependent on the previous value of the system state and the current measurement. The assumption is consistent with the idea of recursive filter since there is no need to store and compute the system past states. Therefore, the weight computing equation is modified as

$$w_n^i \propto w_{n-1}^i \frac{p(\boldsymbol{y}_n|\boldsymbol{x}_n^i)p(\boldsymbol{x}_n^i|\boldsymbol{x}_{n-1}^i)}{q(\boldsymbol{x}_n^i|\boldsymbol{x}_{n-1}^i, \boldsymbol{y}_n)}. \tag{3.8}$$

Since the filtering distribution $p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n})$ is the integration of the posteriori density $p(\boldsymbol{x}_{1:n}|\boldsymbol{y}_{1:n})$, hence $p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n})$ can be approximated as

$$p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n}) \approx \sum_{i=1}^{N_s} w_n^i \delta(\boldsymbol{x}_n - \boldsymbol{x}_n^i).$$

It can be inferred that when N_s approaches infinity, the approximation is equal to posteriori density function $p(\boldsymbol{x}_n|\boldsymbol{y}_{1:n})$ (Haykin, 2009).

3.5.2 Algorithm of particle filters

There are usually two steps implemented in particle filters when a new observation is obtained. The algorithm below is the basic form for a particle filter, which is also referred as sequential importance sampling.

1. Given the $\{\boldsymbol{x}_{n-1}^i, w_{n-1}^i\}$ and the current measurement $\boldsymbol{y}_n$. Draw N_s samples $\{\boldsymbol{x}_n^i, i = 1, ..., N_s\}$ from an importance density function $q(\boldsymbol{x}_n|\boldsymbol{x}_{n-1}^i, \boldsymbol{y}_n)$.

2. Calculate w_n^i for each new sample using Eq. (3.8) and normalize it.

The steps are applied repeatedly to get a new estimation for each time step. However, the disadvantage of the sequential importance sampling is the degeneracy problem, which happens after several iterations of the particle filter, where a few samples have large weights while most of samples are negligible (Arulampalam et al., 2002). The degeneracy problem implies that we will waste the computations in updating the weights of the negligible samples whose contribution to the posteriori pdf is almost zero. One approach to address the degeneracy problem is to add the resampling process after several iterations. So a modified sequence importance sampling algorithm called sequential importance resampling (SIR) is developed (Arulampalam et al., 2002). We will see that the sequential importance sampling algorithm is suited for stabilizing the video in the next chapter, where implementations and testing results are discussed.

4. Implementation of the particle and Kalman filters for video stabilization

In this chapter, we consider the implementation of the particle and Kalman filters for video stabilization. The camera model, the implemented algorithms, and the results are presented and discussed.

4.1 Camera model

Due to the presence of the unexpected movement of the camera, the transformation between consecutive frames is related to the camera motion. The frames in an unstable video suffer from the rotation and the translation of the camera. Assume there is a point in the world coordinates which is denoted by $p(x_w, y_w, z_w)$. At frame k, the projection point of p on the image plane in the camera is assumed to be at (x, y, f), where f is the focal length. Then at the next frame, frame $k+1$, due to the movement of the camera, the projection point of p ends up at (x', y', f). Generally, the distance from the camera to the scene is far enough so that we can ignore the change of the scale factor between the consecutive frames. Also, the rotation angle between the image plane and the z axis is small (J. Yang, Schonfeld, Chen, & Mohamed, 2006). Assume the rotation angle between the image planes of the two frames is counterclockwise and denoted as θ_k. Thus, the 2D affine transformation model is formulated as

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta_k) & -\sin(\theta_k) \\ \sin(\theta_k) & \cos(\theta_k) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} T_{xk} \\ T_{yk} \end{bmatrix},$$

where the T_{xk} and T_{yk} are the translations. The above equation can be rewritten as

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta_k) & -\sin(\theta_k) & T_{xk} \\ \sin(\theta_k) & \cos(\theta_k) & T_{yk} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{4.1}$$

For notational simplicity, the above equation is represented as $\boldsymbol{p} = \boldsymbol{T}_k \boldsymbol{q}$. As Eq. (4.1) indicates, we need to find the θ_k, T_{xk}, and T_{yk} to obtain the transformation matrix $\boldsymbol{T}_k$. The above three unknown variables can be grouped into a vector denoted as $\boldsymbol{x}_k = [\theta_k \; T_{xk} \; T_{yk}]^T$. Hence, for video stabilization, the task is to find $\boldsymbol{x}_k$ between each frame pairs. The first frame in the video is considered to be stable. This means that the transformation matrix for each frame should be referenced to the first frame, which can be achieved with the multiplication of the transformation matrices. The problem of solving for $\boldsymbol{x}_k$ for every frame is considered as a state estimation problem with a nonlinear, non-Gaussian model. Thus, a particle filter is utilized to estimate $\boldsymbol{x}_k$. Note that the nonlinear relationship between $\boldsymbol{x}_k$ and $\boldsymbol{T}_k$ is the main reason to apply the particle filter.

4.2 Implementation of particle filter

As discussed in the previous chapter, a particle filter approximates the posterior probability density using the weighted samples, which is represented as

$$p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k}) = \sum_{i=0}^{N} w_k^i \delta(\boldsymbol{x}_k - \boldsymbol{x}_k^i). \tag{4.2}$$

As Eq. (4.2) suggests, the estimation involves the generation of the samples and the calculation of the corresponding weights. To implement the particle filter, we employ the algorithm proposed in (J. Yang et al., 2006) with slight modifications. Assume that at frame k, the particles are generated from an importance density function which is known as $N_G(\bar{\boldsymbol{x}}_k, \Sigma_k)$, an importance density function with Gaussian distribution with mean $\bar{\boldsymbol{x}}_k$ and the variance Σ_k. Thus, the equation for the particle generations at frame k is defined as

$$\boldsymbol{x}_k^i \sim N_G(\bar{\boldsymbol{x}}_k, \Sigma_k). \tag{4.3}$$

Note that the mean $\bar{\boldsymbol{x}}_k$ is important for the approximations since it provides the baseline estimation. The closer the mean vector to the real state, the more accurate results the particle filter produces.

In general, the mean vector can be obtained via feature detection algorithms (Abdullah, Tahir, & Samad, 2012). The features of an image are usually corners and edges (Manjunath, Shekhar, & Chellappa, n.d.). There are many feature detection techniques that utilizes the edges, corners (Harris & Stephens, 1998), and small blob areas on an image to uniquely characterize the image. Feature detection algorithms have been used in video stabilization, image registration, motion detection, and object recognition (Tong, Kamata, & Ahrary, 2009). The feature detection method employed in this thesis is the Speeded Up Robust Features (SURF) detector (Pinto & Anurenjan, 2011). The SURF detector detects the points on an image that are invariant to scale, rotation, and the change of illumination. Also, the SURF detector is suitable for real time application as it is computational efficient.

So once we have the matched j feature points between two frames from the SURF operation, j equations like Eq. (4.1), exists. Let $\boldsymbol{P} = [\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_j]$ and $\boldsymbol{Q} = [\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_j]$. Then $\boldsymbol{P} = \boldsymbol{T}_k\boldsymbol{Q}$, which leads to

$$\boldsymbol{T}_k = \boldsymbol{P}\boldsymbol{Q}^T(\boldsymbol{Q}\boldsymbol{Q}^T)^{-1}. \tag{4.4}$$

It is obvious that we need at least three matched points to solve for $\boldsymbol{T}_k$ and then for $\boldsymbol{x}_k$. This can be easily achieved with SURF. Hence, we have the mean values $\bar{\boldsymbol{x}}_k = [\bar{\theta}_k \; \bar{T}_{xk} \; \bar{T}_{yk}]^T$.

Now that the mean value $\bar{\boldsymbol{x}}_k$ is available, the samples of the state $\boldsymbol{x}_k$ for frame k can be drawn from the importance density function $N_G(\bar{\boldsymbol{x}}_k, \Sigma_k)$, where the Σ_k is determined independently for different situations. As the samples are generated, the weight for each sample is assigned based on the similarity between the inversely transformed frame using the samples and the reference frame. In our case, there are N proposed particles. So we can apply the N inverse transformations to the current frame using the N samples, and then calculate how similar is the inversely transformed frame to the first frame for each sample. The particle that produces the most similar frame is assigned to a heavier weight. Note that this method works only when the camera takes the video of the same scene.

The processes for measuring the similarity between two images utilize the methods in (J. Yang et al., 2006). The first method is to calculate the Mean Square Error (MSE) M_i^2 between two images. It is obvious that the smaller the MSE, the less difference between the two images. Thus the likelihood of the two images is higher when the MSE is smaller, which can be approximated by the Gaussian distribution below

$$P_{MSE}^i \propto \frac{1}{\sqrt{2\pi}\sigma_M} \exp\{-\frac{M_i^2}{2\sigma_M^2}\}, \tag{4.5}$$

where σ_M is the standard deviation and can be determined by experiments.

The second parameter is the correlation between the two images. The coefficient of correlation P_i indicates the degree that the two images are linearly related (Kaur, Kaur, & Gupta, 2012). The probability of the similarity between two images using correlation coefficient is given by

$$P_{corr}^i \propto \frac{1}{\sqrt{2\pi}\sigma_{corr}} \exp\{-\frac{(P_i - 1)^2}{2\sigma_{corr}^2}\}, \tag{4.6}$$

where the σ_{corr} is the adjustable standard deviation, which is determined by experiments.

After obtaining two weights from Eq. (4.5) and Eq. (4.6), the normalized weight corresponding to each particle at frame k can be calculated as

$$\boldsymbol{w}_k^i = \frac{P_{MSE}^i P_{corr}^i}{\sum_{i=1}^N P_{MSE}^i P_{corr}^i}. \tag{4.7}$$

So far, the samples and the weights are attained, the estimated state vector at frame k is approximated by the discrete summation of the weighted particles. The equation is shown as

$$\hat{\boldsymbol{x}}_k = \sum_{i=1}^N \boldsymbol{w}_k^i \boldsymbol{x}_k^i. \tag{4.8}$$

Therefore, the output from the particle filter provides the estimated vector $\hat{\boldsymbol{x}}_k = [\hat{\theta}_k \ \hat{T}_{xk} \ \hat{T}_{yk}]$ for the global movement between two successive frames.

One more step is to compute the transformation matrix that references to the stable frame. Since the first frame is regarded to be stable, the accumulative transformation matrix can be obtained in terms of the first frame, consider $\boldsymbol{p}_2 = \boldsymbol{T}_1 \cdot \boldsymbol{p}_1$, $\boldsymbol{p}_3 = \boldsymbol{T}_2 \cdot \boldsymbol{p}_2$, $\cdots$, and $\boldsymbol{p}_{k+1} = \boldsymbol{T}_k \cdot \boldsymbol{p}_k$, where p_k denotes frame k and T_k represents the transformation matrix at frame k. Thus, at frame k, the transformation matrix between the first and the current frame is $\boldsymbol{p}_{k+1} = \boldsymbol{H}_k \cdot \boldsymbol{p}_1$, where

$$\boldsymbol{H}_k = \prod_{i=1}^k \boldsymbol{T}_i \tag{4.9}$$

is the accumulative transformation matrix at frame k.

Note that the output from the particle filter gives us the estimation of the global camera motion, the motion with respect to frame. To maintain the intentional movement due to the movement of the UAV and the object motion, extra steps are required, which are explored in the next section.

4.3 Implementation of the Kalman filter

As we only need to get rid of the unwanted movement, the intentional movement on the video should not be removed. Thus, the intentional movement of the airplane should be calculated and used to compensate the global movement (J. Yang et al., 2006). A Kalman filter is utilized to estimate the intentional motion of the camera since the intentional moving camera system can be modeled as a linear system. For the Kalman filter, we need to find the two linear models, for the system state transition model and the observation model. Assume the rotation angle and translations along x and y axises are independent variables. So for the x-axis translation, the state transition model is defined as

$$T_{xk} = T_{x,k-1} + v_{xk},$$

$$v_{xk} = v_{x,k-1} + n_{vx,k-1},$$

where T_{xk} and $T_{x,k-1}$ are the translations along the x-axis at frames k and $k-1$, respectively, v_{xk} and $v_{x,k-1}$ are the moving speed along x-axis at frames k and $k-1$, respectively, and $n_{vx,k-1}$ is the zero mean Gaussian noise and has the distribution $n_{vx,k-1} \sim N(0, \sigma_{vx}^2)$. For the observation model, the equation is simply

$$Z_{xk} = T_{xk} + m_{xk},$$

where Z_{xk} is the measurement at frame k and m_{xk} is a Gaussian noise with zero mean and variance σ^2_{mtx}.

Similarly, translation T_y along the y-axis can be modeled in the same way as T_x. For the rotation angle, the assumption is that there is no intentional angular velocity of the camera. Thus, the state space model in given as

$$\begin{bmatrix} T_{xk} \\ v_{xk} \\ T_{yk} \\ v_{yk} \\ \theta_k \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} T_{x,k-1} \\ v_{x,k-1} \\ T_{y,k-1} \\ v_{y,k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} 0 \\ n_{vx,k-1} \\ 0 \\ n_{vy,k-1} \\ n_{\theta,k-1} \end{bmatrix}, \tag{4.10}$$

where θ_k is the rotation angle at frame k, $n_{vy,k-1}$ and $n_{\theta,k-1}$ are both zero mean Gaussian noises with variance being σ^2_{vy} and σ^2_{θ}, respectively. Accordingly, the observation model is formulated as

$$\begin{bmatrix} Z_{xk} \\ Z_{yk} \\ Z_{\theta k} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} T_{xk} \\ T_{yk} \\ \theta_k \end{bmatrix} + \begin{bmatrix} m_{xk} \\ m_{yk} \\ m_{\theta k} \end{bmatrix}, \tag{4.11}$$

where Z_{xk}, Z_{yk}, and $Z_{\theta k}$ are the measurements of the translations along x-axis, y-axis, and the rotation angle, respectively, m_{xk}, m_{yk}, and $m_{\theta k}$ are the zero mean Gaussian observation noises with variance being σ^2_{mx}, σ^2_{my}, and $\sigma^2_{m\theta}$, respectively.

From the above two models of Eq. (4.10) and Eq. (4.11), the intentional motion vector can be obtained using Kalman filter and is denoted as $\boldsymbol{z}_k = [T_{xk}\ T_{yk}\ \theta_k]^T$. Then the unexpected camera motion is computed as

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} cos(\tilde{\theta}_k) & -sin(\tilde{\theta}_k) & \tilde{T}_{xk} \\ sin(\tilde{\theta}_k) & cos(\tilde{\theta}_k) & \tilde{T}_{yk} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix},$$

where $\tilde{\theta_k} = \hat{\theta_k} - \theta_k$, $\tilde{T_{xk}} = \hat{T_{xk}} - T_{xk}$, and $\tilde{T_{yk}} = \hat{T_{yk}} - T_{yk}$ are the unintentional motion estimation for the rotational angle and translations along both axises. For notation simplicity, the above equation is represented as $\boldsymbol{p} = \tilde{\boldsymbol{T}}_k \boldsymbol{q}$.

Another important issue in video stabilization is to estimate the object motion in the video. In (J. Yang et al., 2006; Song et al., 2012), the object motion is removed before the background motion estimation by detecting the motion speed that is assumed to be faster than the background. However, in our case, we assume that the airplane moves very slowly among successive frames since the target is very far away from the camera. Moreover, the airplane appears to be very small on the image, which has little to zero feature points. The airplane appears to be static compared to the camera motion. Therefore, the motion of the object is not considered in this thesis when doing the background motion estimation.

To summarize the video stabilization algorithms, the detailed operations at each frame k is illustrated as follows.

1. Read the video and load the consecutive frames: frame k and frame $k-1$.

2. Detect, extract, and match feature points of two consecutive frames using SURF.

3. Compute the state vector $\bar{\boldsymbol{x}}_k$ from $\boldsymbol{T}_k$ estimated using Eq. (4.4).

4. Estimate $\hat{\boldsymbol{x}}_k$ using a particle filter with N particles.

   (a) for i = 1:N, generate particles from the Gaussian importance density as shown in Eq. (4.3).

   (b) for i = 1:N, assign the weights to each particle and calculate the normalized weight for every sample, as shown in Eq. (4.7).

   (c) Estimate the state vector using the weighted samples, as illustrated in Eq. (4.8).

5. Calculate the accumulative transformation matrix as stated in Eq. (4.9).

6. Put the accumulative matrix into the Kalman filter to estimate the intentional motion. Then calculate the unexpected transformation matrix $\tilde{\boldsymbol{T}}_k$.

7. Apply inverse transformation using the above $\tilde{\boldsymbol{T}}_k$ to the current frame to form the stabilized video.

Testing results of video stabilization will be shown in the next section.

4.4 Testing results

To accurately test the performance of the video stabilization algorithms, we generate the shaky videos using rotation and translations to a known image so that ground truth values of rotation angle and the translations are known. The parameters for the particle filter are chosen as follows: the number of particles $N = 30$, $\Sigma_k = [0.001\ 10\ 10\ ]$, and $\sigma_{MSE} = \sigma_{corr} = 0.5$. For the Kalman filter, the initial states are all zero. The system noise parameters are $\sigma_\theta = 0.5$ and $\sigma_{vx} = \sigma_{vy} = 5$. The observation noise parameters are $\sigma_{m\theta} = \sigma_{mtx} = \sigma_{mty} = 0.1$. Moreover, the initial error covariance matrix is assumed to be equal to the system noise covariance matrix. These values are controllable and subject to change for different cases.

4.4.1 Testing results for smooth linear motions

The first video comprises a series of the images that are obtained with linear increment in each of the three shaking parameters of the rotation angles, the translations along x-axis, and the translations along y-axis. We use three different schemes on this video for stabilization: Scheme A, SURF-only feature detection, Scheme B, SURF + a particle filter, and Scheme C, SURF + a particle filter + a Kalman filter. Scheme A estimates the transformation matrix using the match points directly, and Scheme B estimates the transformation matrix using SURF first and then a particle filter to obtain more accurate results. Note that the outputs from the first two schemes are about global camera motion as shown in Eq. (4.9). In Scheme C, the Kalman filter is applied to estimate the intentional motion vector following SURF and particle filtering. The testing results are shown as follows.

Example frames of unstable videos, stabilized videos by Schemes A, B, and C are given in Figs. 4.1 to 4.4. Note that for the result in Fig. 4.4, the translation along x-axis is unchanged since it is considered as the intentional move.

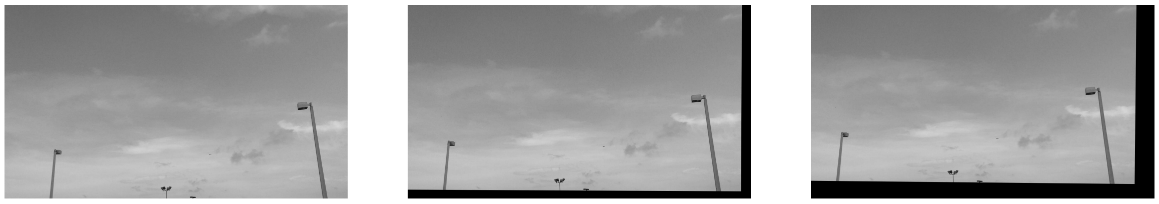

(a) Frame = 1. (b) Frame = 50. (c) Frame = 100.

Figure 4.1. Frames from the unstable video.

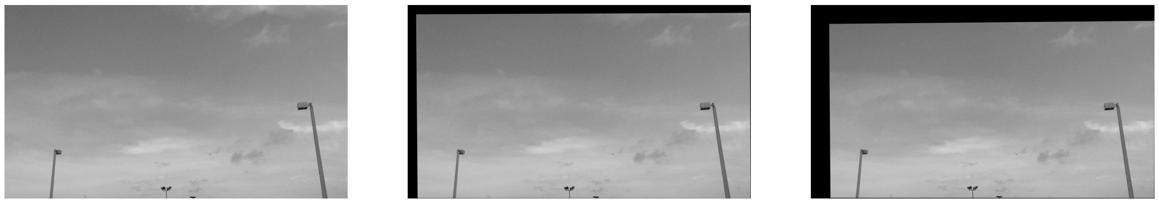

(a) Frame = 1. (b) Frame = 50. (c) Frame = 100.

Figure 4.2. Stabilized frames processed with Scheme A.

(a) Frame = 1. (b) Frame = 50. (c) Frame = 100.

Figure 4.3. Stabilized frames processed with Scheme B.

(a) Frame = 1. (b) Frame = 50. (c) Frame = 100.

Figure 4.4. Stabilized frames processed with Scheme C.

Figs. 4.5, 4.6, and 4.7 show the comparisons of the estimation results for each of the three parameters of camera motion. As can be seen from Fig. 4.7, Scheme B outperforms Scheme A, and since Scheme C considers linear motions as the intentional movement, both the estimation results are close to zero, which means no unexpected motion. However, we can see from Fig. 4.5 and Fig. 4.6 that Scheme A outperforms the other two schemes. This is due to the assumption that no intentional motion along x-axis and y-axis translation.

4.4.2 Testing results for random motions

Another video is produced with rotation angle and translation in y-axis of each frame being random processes to further evaluate the performances of the three

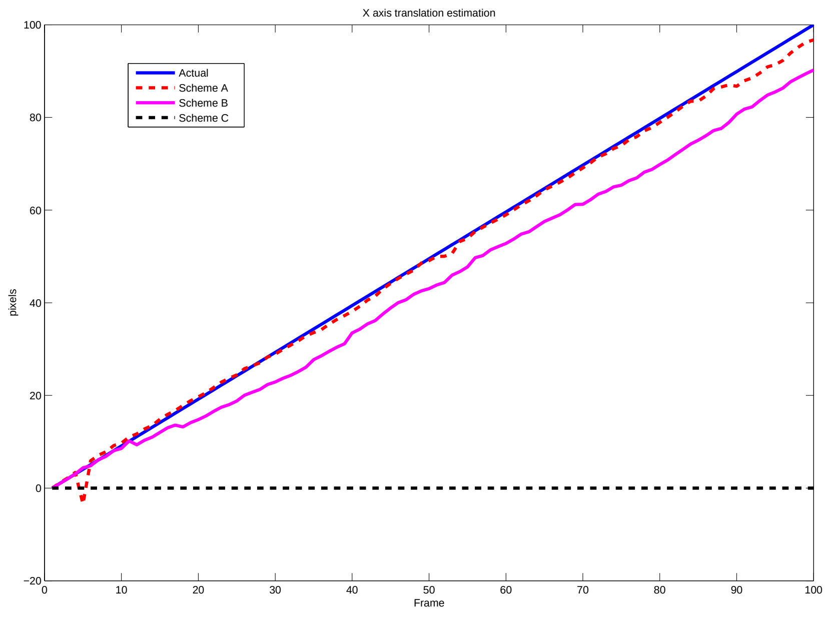


Figure 4.5. Comparison of x-axis translation estimations for linear translation change.

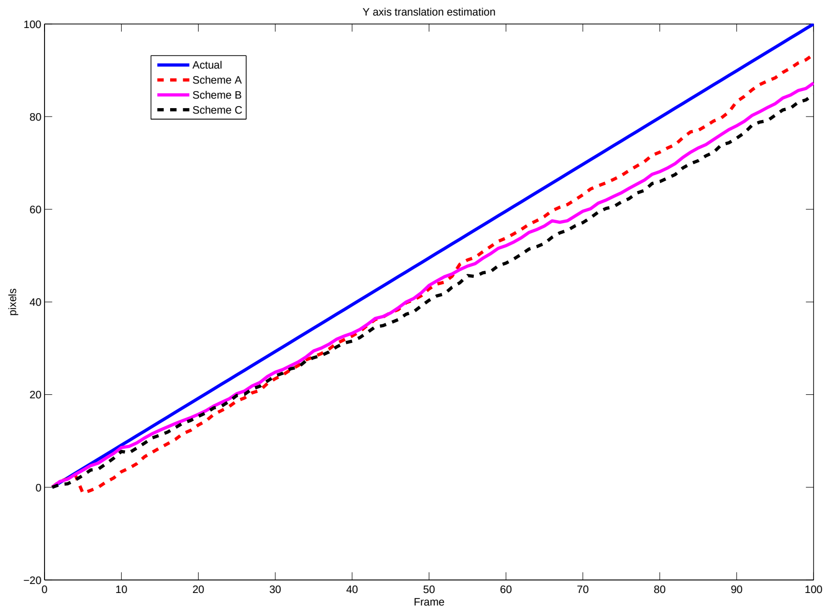


Figure 4.6. Comparison of y-axis translation estimations for linear translation change.

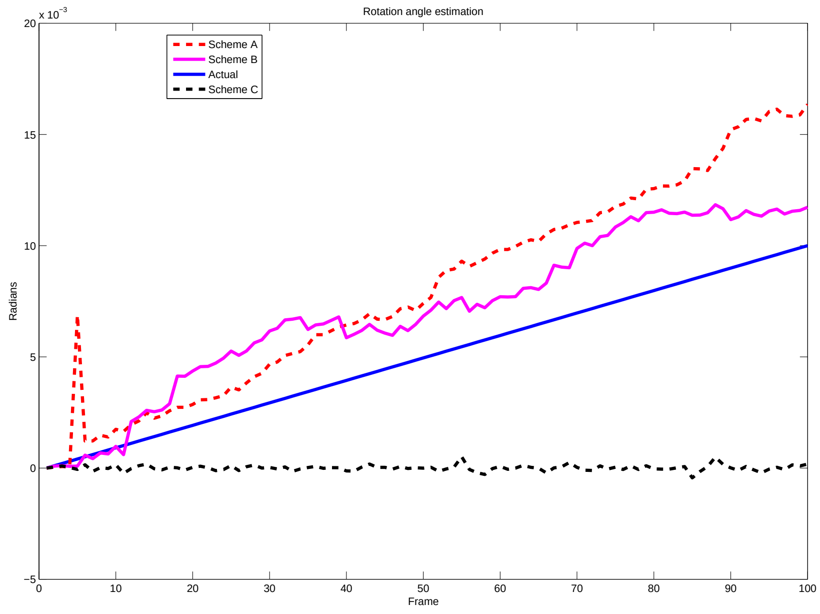
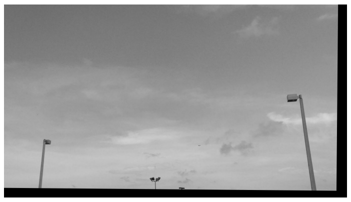


Figure 4.7. Comparison of rotation estimations for linear rotation change.

schemes. The motion along x-axis remains the same linear relationship as that in the first video. The results are shown as follows.

(a) Frame = 1.


(b) Frame = 50.

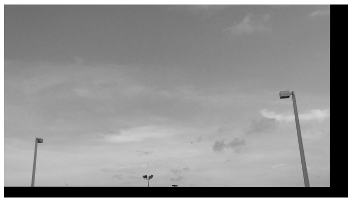
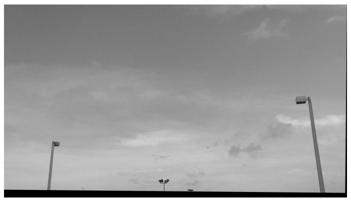

(c) Frame = 100.

Figure 4.8. Frames from unstable video.


(a) Frame = 1.

(b) Frame = 50.

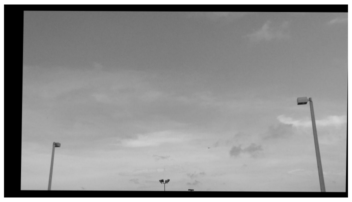
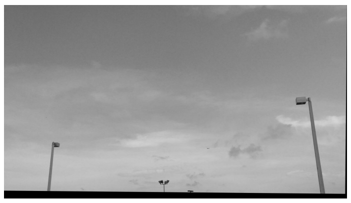

(c) Frame = 100.

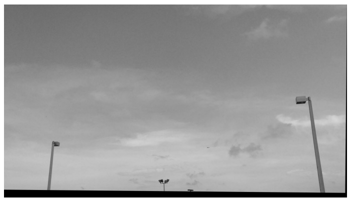
Figure 4.9. Stabilized frames processed with Scheme A.


(a) Frame = 1.

(b) Frame = 50.

(c) Frame = 100.

Figure 4.10. Stabilized frames processed with Scheme B.


(a) Frame = 1.

(b) Frame = 50.

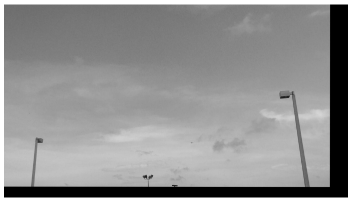

(c) Frame = 100.

Figure 4.11. Stabilized frames processed with Scheme C.

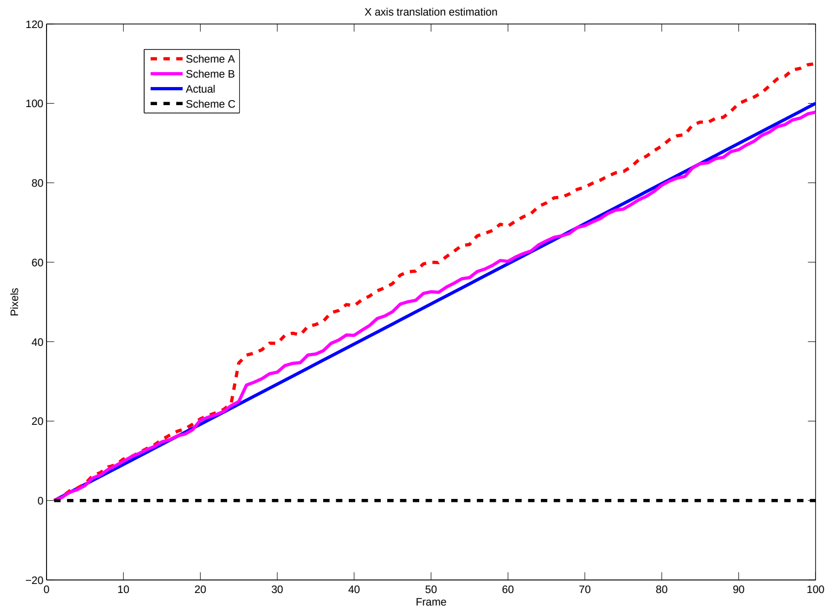


Figure 4.12. Comparison of the x-axis translation estimations for random translation change.

Exemplary frames of unstable videos, stabilized by Schemes A, B, and C are given in Figs. 4.8 to 4.11, and Figs. 4.12, 4.13, and 4.14 show the comparisons of the estimation results for x-axis, y-axis translations, and rotation angle. As seen from Fig. 4.14, Scheme C outperforms both Scheme A and Scheme B, demonstrating the effectiveness of the Kalman filter. Yet, as shown in Fig. 4.13 for y-axis translation estimation, Scheme C performs the worst, still due to the estimation of intentional change. We can see from Fig. 4.12 that Scheme B outperforms Scheme A.

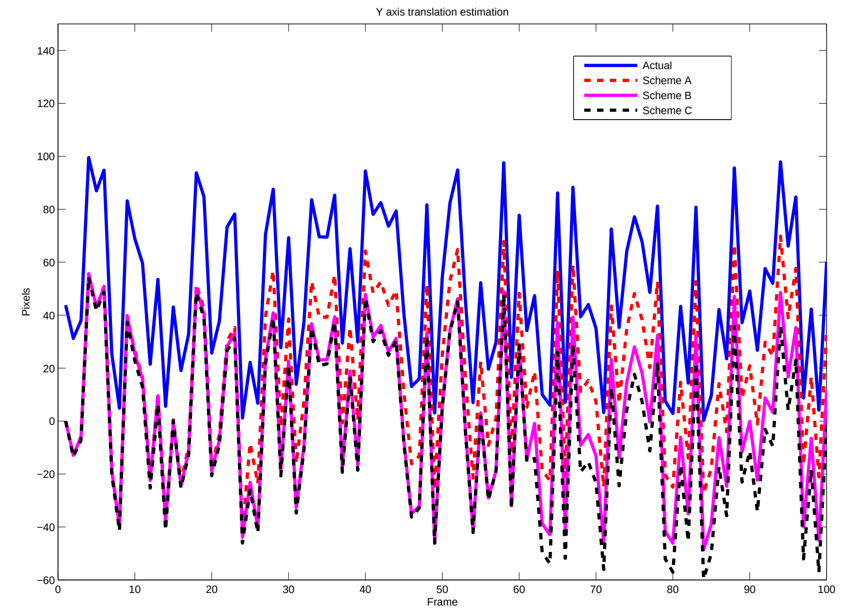


Figure 4.13. Comparison of the y-axis translation estimations for random translation change.

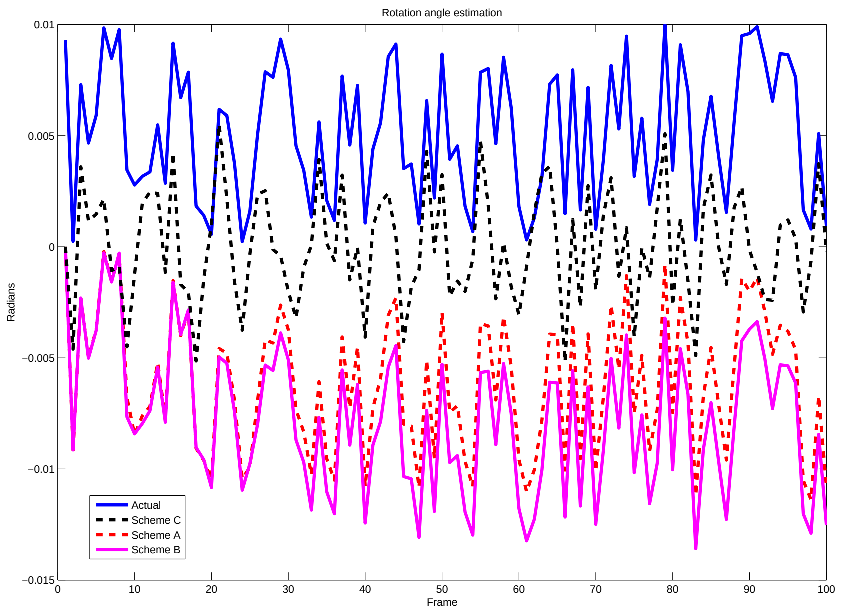


Figure 4.14. Comparisons of the rotation estimations for random rotation change.

5. Object detection algorithms

In this chapter, we discuss object detection algorithms in details. Usually, the detection of an aircraft in an image is hindered by many factors, including pixel noise, heavy clouds, and other obstacles on the ground. Therefore, the development of suitable algorithms are critical for successful aircraft detection among other distractions. Some popular algorithms include edge detection (Bhadauria, Singh, & Kumar, 2013), connected area extraction (Hajri, 2012), morphological filtering (Casasent & Ye, 1997; Sang, Zhang, & Wang, n.d.), local adaptive threshold filtering (Zarandy et al., 2011), and dynamic programming (M. Yang et al., 2002; Barniv, 1985). In the sequel of this thesis, we discuss the development and implementations of several algorithms for object detection.

5.1 Edge detection

Edge detection is one of the fundamental operations in computer vision. Edges are significant local changes of intensity, which usually occur on the boundary between different regions in an image. Therefore, edge detection extracts image features such as corners, lines, and curves on the image. Generally, derivative operations are applied to detect the sudden change of the intensity in an image.

The first-order partial derivatives for $f(x, y)$ are, respectively,

$$f_x = \frac{\partial f}{\partial x} = \lim_{h \to 0} \frac{f(x+h, y) - f(x, y)}{h},$$
$$f_y = \frac{\partial f}{\partial y} = \lim_{h \to 0} \frac{f(x, y+h) - f(x, y)}{h}.$$

By definition, the gradient is a vector with direction and magnitude. For a 2D discrete digital image, the gradient is approximated by finite differences, with $h = 1$, which is denoted as $\nabla f = [f_x \; f_y]^T$:

$$\begin{aligned} f_x &= f(x+1, y) - f(x, y) \\ f_y &= f(x, y+1) - f(x, y) \\ M(\nabla f) &= \sqrt{(f_x)^2 + (f_y)^2} \\ \theta(\nabla f) &= \arctan(f_y/f_x) \end{aligned} \tag{5.1}$$

The operation of edge detection can be considered as the convolution of the image with a mask, a filter. For example, the convolution masks defined in Eq. (5.1) are $[-1 \;\; 1]$ in the x direction and $[-1 \;\; 1]^T$ in the y direction, respectively. Then the edges are determined by finding the local maximum or minimum points, which can be decided by comparing the convoluted results with a threshold.

Another way to acquire the local maximum and minimum points is checking whether the second derivative at the point is zero-crossing. The second derivative of 2D function $f(x, y)$ is obtained as

$$\nabla f^2 = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}, \tag{5.2}$$

which is also known as the Laplacian edge detector. For calculations using Eq. (5.2), one of the popular discrete convolution kernels of the Laplacian edge detector is obtained as

$$M = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

Since the Laplacian operation is sensitive to noise due to the second-order derivatives, the operation is often applied after a Gaussian filter which reduces noise. This is also called the Laplacian-of-Gaussian (LoG) (Maini & Himanshu, 2009).

There are four different edge detectors widely used: Roberts, Prewitt, Sobel, and Canny edge detectors (Shrivakshan & Chandrasekar, 2012). In this thesis, the Sobel edge detector is employed due to its computational efficiency. The Sobel edge detector is an example of applying the first-order derivative to the image to obtain the edges.

The Sobel edge detector uses a pair of 3×3 convolutional kernels which are formulated as follows.

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad M_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Compared to the second derivative, the Sobel operator is less sensitive to unexpected noise.

5.2 Morphological processing

Morphological processing is the collection of non-linear operations related to the shape of morphology of features in an image (Gandhi, Yang, Kasturi, Coraor, & McCandless, 2003). It provides a way for extracting small, point-like targets (Carnie et al., 2005; Yusko, 2007), which is a good application for UAV sense and avoidance. Morphological processing is usually performed before the other image algorithms to preserve the small objects while removing the large cloud clutters on the image. Morphological operations only rely on the relative ordering of the pixel values instead of on their numerical values, so they are widely applied to process binary images. Generally, the operation involves two primary operations which are known as dilation and erosion. The basic element in these two morphological operations is a binary region called a structure element, a small binary matrix whose shape is defined by the pattern of ones and zeros. Unless specified otherwise, the center of the structure element is the origin (Sonka, Hlaval, & Boyle, 2013). During morphology operation, different values (1 or 0) will be assigned to the corresponding area under the structure element as the structure element slides along the image.

The mathematical expression for the dilation and erosion are defined (Sonka et al., 2013) as

$$\begin{aligned} F \oplus SE &= \{z | z = f + se, f \in F, se \in SE\}, \\ F \ominus SE &= \{z | z + se \in F, se \in SE\}, \end{aligned} \tag{5.3}$$

where F is a set with the elements being represented by f, SE denotes the structure element whose members are expressed as se, and $\oplus$ and $\ominus$ stand for the dilation and erosion operations, respectively.

Moreover, the dilation and erosion operations for a 2D gray-scale image f at location (x, y) are defined (Carnie, Walker, & Corke, 2006) by the following

$$\begin{aligned} (f \oplus s)(x, y) &= \max_{(u,v \in s)} \{f(x - u, y - v)\}, \\ (f \ominus s)(x, y) &= \min_{(u,v \in s)} \{f(x - u, y - v)\}, \end{aligned} \tag{5.4}$$

where s is the structure element and (u, v) is a pixel in s.

Additional morphological operations can be achieved by combining the two fundamental operations together. Morphological opening process is an erosion followed by a dilation

$$f \circ s = (f \ominus s) \oplus s,$$

and morphological closing is a dilation followed by an erosion

$$f \bullet s = (f \oplus s) \ominus s.$$

Usually, the opening operation smooths the contours of the object by eliminating thin protrusions and breaking narrow bridges that are too small to accommodate the structure element (Sonka et al., 2013). On the other hand, the closing operation tends to smooth the entire section area by building up the links and filling small holes and gaps.

From the above discussion, we know that the small bright areas are darkened by the opening operation and the small dark areas are brightened by the closing operation. As such, by subtracting the opened image from the original image, the small positive objects are obtained. Similarly, the difference between the original image and the closed image identifies the small negative objects that are darker than the background (Maragos, 1987). Thus, the closed image minus the opened image provides the detections for both the positive and negative objects. Such a process is called the Closing-Minus-Opening (CMO) operation which is formulated as

$$CMO(f, s) = (f \bullet s) - (f \circ s).$$

In (Carnie et al., 2006), a so-called minimum CMO operation is proposed to eliminate the large cloud and ground clutter whose existence causes false detections. The minimum CMO utilizes two 1-D structure elements that are used for vertical and horizontal operations. During the two CMO operations, small objects are preserved if the sizes of the structure elements are bigger than the objects, with large clutter eliminated either in the vertical or horizontal operations. Therefore, most of the large clutters are removed after calculating the minimum values out of the two CMO operations.

5.3 Dynamic programming

While the minimum CMO operation can be used to detect small negative and positive objects with high detection probability, the detection performance is often affected by random pixel noise and poor signal to noise ratio. One optimal solution for moving target detection is to utilize dynamic programming (DP) (Arnold, Shaw, & Pasternack, 1993), a combination of detection and tracking, which returns the target detection and tracking at the same time (Tonissen & Evans, 1995). The DP algorithm has been proven to be efficient in detecting targets with low signal-to-noise ratio and is robust to the camera jitter and random noise (Barniv, 1985). Instead of detecting objects based on a single image, DP makes the decision of the presence of the target after shifting and averaging multiple frames, which is suited for object detection (Gonzalez & Woods, 2008), tracking (Tonissen & Evans, 1995), and even edge detection (Lee, Yan, & Zhuang, 2001).

For aircraft sense and avoid applications, the object movement between two frames is less than 1 pixel, especially at far distances (Hobbs, 1991). Hence, we consider a 2D image, where the target position is represented by (i, j) and the 2D velocity of the

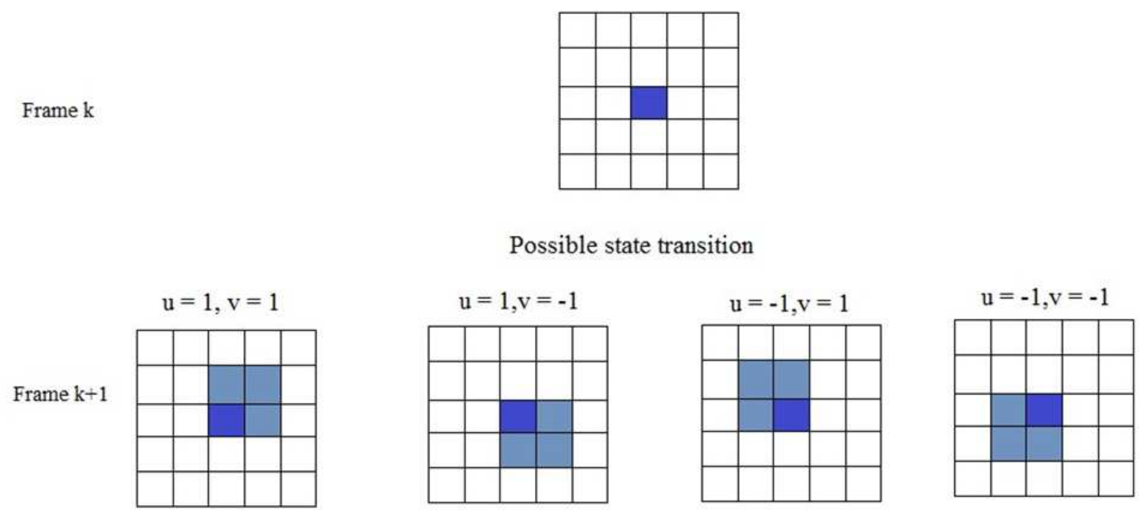


Figure 5.1. Object movement illustration.

target is assumed to be (u, v), with $-1 \leq u, v \leq 1$. The number of target trajectories can be reduced by comparing state transition between consecutive frames.

Assume at frame k, the object is at location (i, j) with the speed (u, v). Then at frame $k+1$, the object can end up at any location centered around (i, j) with the range of 1 pixel, as shown in Fig. 5.1 (Carnie et al., 2006). The dark blue marks the location of the object in frame k and the possible locations in frame $k+1$ are colored in light blue. The nine possible locations are grouped into four cases in terms of the velocity u and v.

The steps for the dynamic programming are detailed in (M. Yang et al., 2002) and are reproduced here for completeness.

Initialization

For frame $k = 0$, $F_{u,v}(i, j, 0) = 0$, $u \in \{-1, 1\}$ $v \in \{-1, 1\}$, where $F_{u,v}(i, j, 0)$ is the (i, j) pixel of frame 0 of the processed image in the (u, v) direction.

Recursion

At frame $k+1$, the value of each pixel of the processed frame in each direction is the summation of the weighted value of the pixel of the input image at frame $k+1$ and the maximum response of four possible transition states in four directions of frame k. The calculation is given as

$$F_{uv}(i,j,k+1) = (1-\alpha)f(i,j,k+1) + \alpha \max_{(x',y') \in Q(i,j,u,v)} F_{uv}(i',y',k), \tag{5.5}$$

where $f(i,j,k+1)$ is the original frame $k+1$, α is the factor that determines how much it should trust the previous frame (also called memory factor) whose values range from 0 to 1, $Q(i,j,u,v)$ represents the pixel values within the window for four different cases as illustrated in Fig. 5.1.

Decision Finally, the pixel value at the (i,j) of the processed image of frame $k+1$ is the maximum value among all the four cases. Thus

$$F_m(i,j,k+1) = \max_{(u,v)} F_{uv}(i,j,k+1). \tag{5.6}$$

The processed image with DP is usually converted into a binary image with a threshold for detection purpose. The target with low signal-to-noise ratio is able to be detected since the dynamic programming raises the signal-to-noise ratio for dim moving target. Note that there is usually clutter besides the target. The large area clutter should be eliminated using the CMO operation before processing with DP.

5.4 Implementation of object detection algorithms

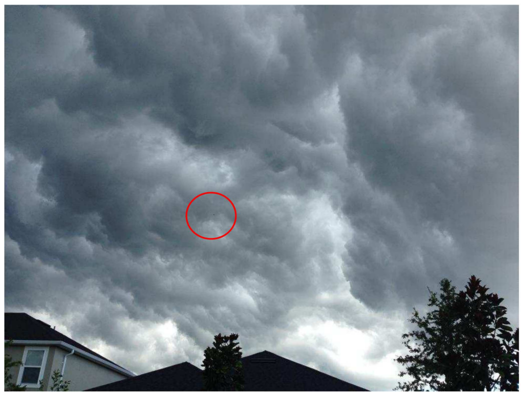
Object detection can be done using various combinations of image processing algorithms discussed above. To test the effectiveness of the different algorithms, we consider three schemes: Scheme 1, the Sobel edge detector, Scheme 2, morphological processing plus the Sobel edge detector, and Scheme 3, morphological processing plus dynamic programming and the Sobel edge detector. The algorithm for Scheme 3 is outlined below.

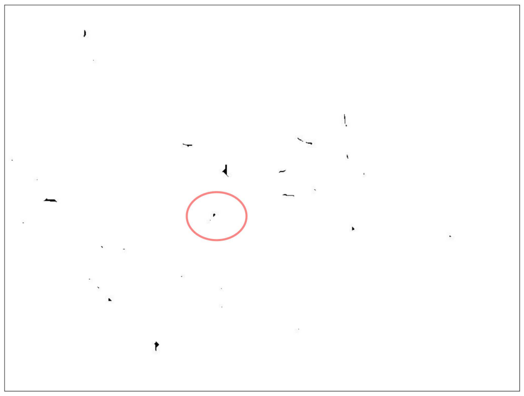
1. Convert the image to grayscale as needed, which is referenced as f.
2. Apply the minimum CMO algorithm to the grayscale image f:
    (a) First process f with CMO using a horizontal structure element to get the horizontal CMO image f_h.
    (b) Then apply CMO with a vertical structure element to f to get vertical CMO image f_v.
    (c) Finally obtain the minimum CMO image f_m by finding the minimum pixel values between f_h and f_v for each pixel.
3. Process f_m with DP as detailed in a previous section to obtain F_m.
4. Detect the edge on F_m using a Sobel detector with a threshold value τ.

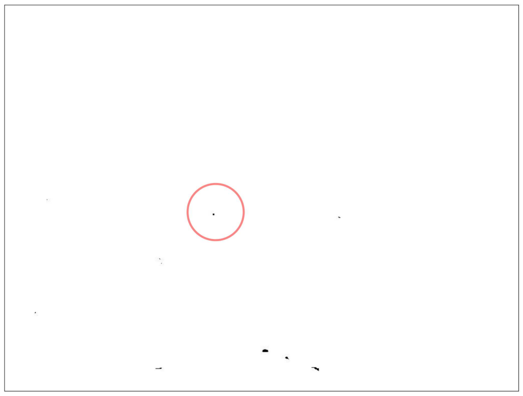
5.5 Results of object detection

In this section, we demonstrate the performance of the three object detection schemes. For the morphological processing, we use a structure element of size 1×10 in the horizontal direction and another of size 10×1 in the vertical direction. The threshold τ is chosen to be 0.3 determined by experiments. For the removal of large clutters, the maximum area size is set to be 300 pixels or 100 pixels depending on the size of the aircraft. This means any connected area whose size is bigger than 300 or 100 pixels will be eliminated. We use four sets of videos to demonstrate the performance of different schemes. In order to show clearly the detected objects in the printed copy, all the binary images are displayed in a way that the background is white and the detected objects are black.

Fig. 5.2a is the first frame from a synthetic video. This video is generated in a way that the background does not change while the object moves between consecutive frames. The size of the target is designed to be 2×2 and the speed of the target is constrained within 1 pixel per frame to be consistent with the assumptions of the dynamic programming. As shown in the figure, there are large dark clouds in the sky and large buildings at the bottom. In addition, the target is very dim and the contrast between the target and the background is not very sharp, making it difficult to detect with the naked eye. Figs. 5.2b to 5.2d demonstrate the object detection results using different schemes. We can see that Scheme 1 works well in terms of object detection, but suffers from too many false detections. We can also see that


(a) The original image.


(b) Result of Scheme 1.


(c) Result of Scheme 2.

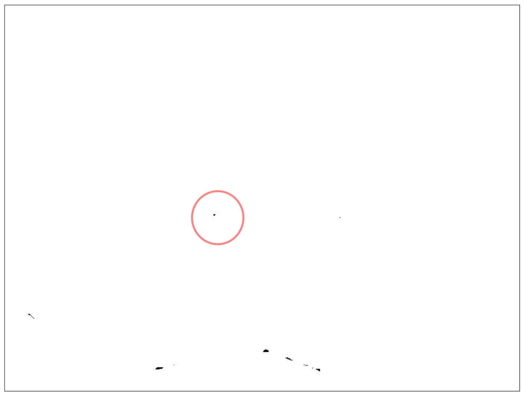
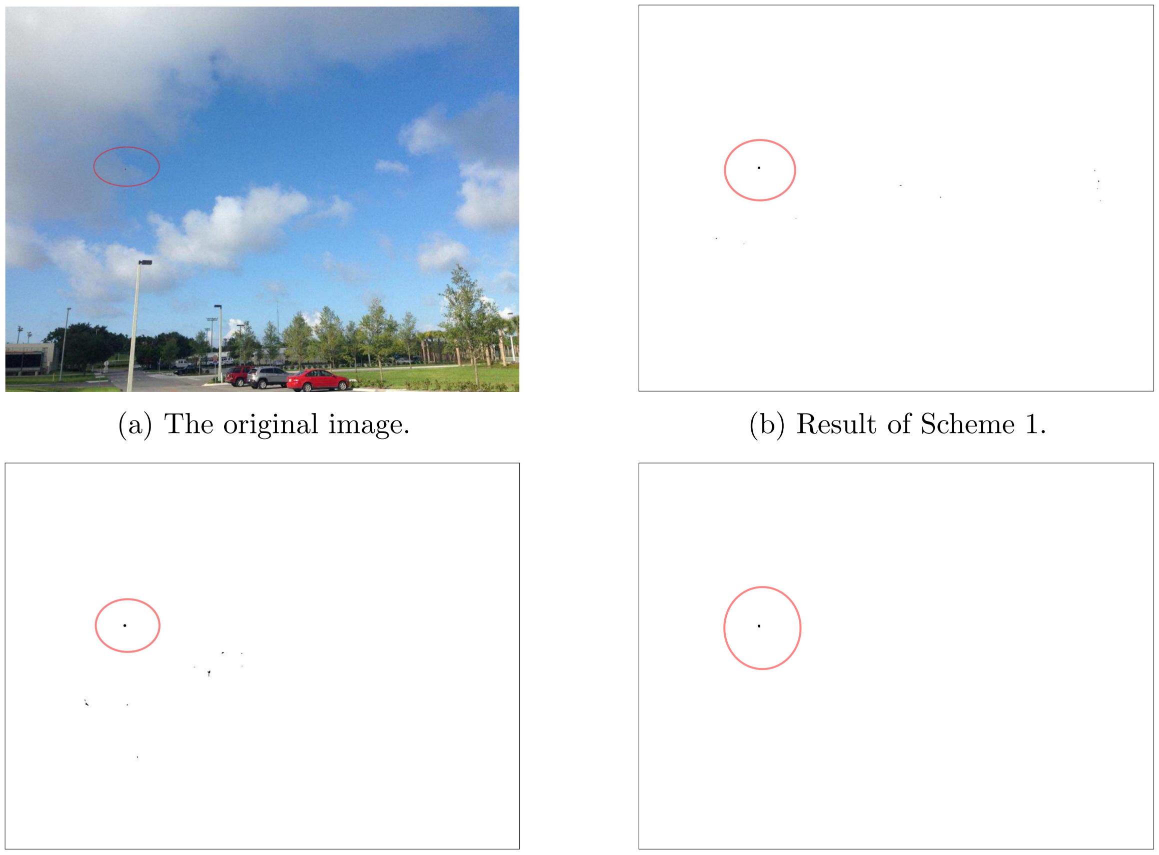

(d) Result of Scheme 3.

Figure 5.2. Object detection results for a synthetic video with dark clouds.

both Scheme 2 and Scheme 3 work very well with a lower number of false detections. Note that the power of DP is not obvious since the video is not very noisy.

The image shown in Fig. 5.3a is a frame from the video that has been added with zero mean Gaussian noise of variance 0.0002. This video is generated in the same way as the one shown in Fig. 5.2a. Figs. 5.3b to 5.3d demonstrate the object detection results using different schemes. We can see that Scheme 1 still works well in terms of object detection and suffers less with false detections due to better cloud conditions.



(a) The original image. (b) Result of Scheme 1.

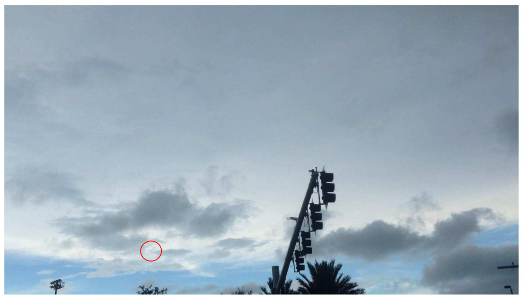
(c) Result of Scheme 2. (d) Result of Scheme 3.

Figure 5.3. Object detection results for a synthetic video with light clouds and added noise.

We can also see that Scheme 3 significantly outperforms Scheme 2 due to the power of DP in the presence of noise.

Fig. 5.4a shows the original image with varying clouds, other clutters, and a relatively small object. The video was recorded on the ground by hand. We can see from Figs. 5.4b to 5.4d that Scheme 3 significantly outperforms the other two schemes in terms of reduced false alarms. This is again due to the effectiveness of DP.

Fig. 5.5a shows the original image recorded on the ground without too many distractions. Although there are some lamp posts in the image, the sky is clear


(a) Original image.

(b) Result of Scheme 1.

(c) Result of Scheme 2.

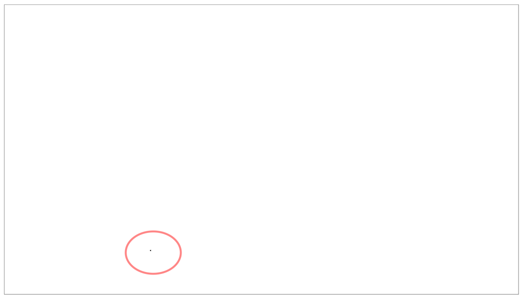

(d) Result of Scheme 3.

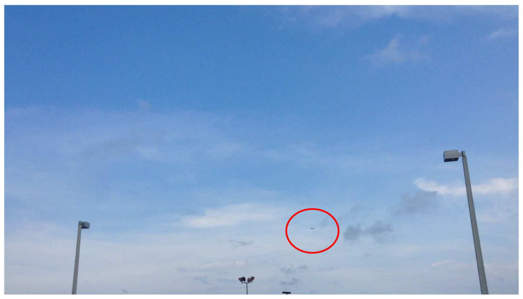
Figure 5.4. Object detection results for a recorded video with varying clouds.

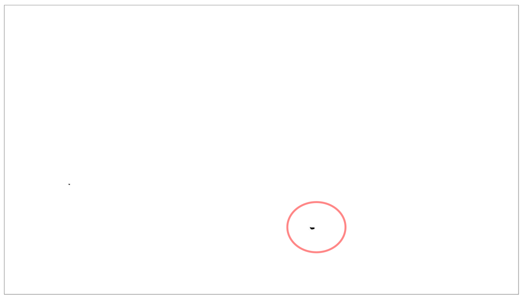
without heavy cloud clutter. Figs. 5.5b to 5.5d demonstrate the object detection results using different schemes. We can see that due to the big difference between the flying object and the background, Scheme 1 performs best.

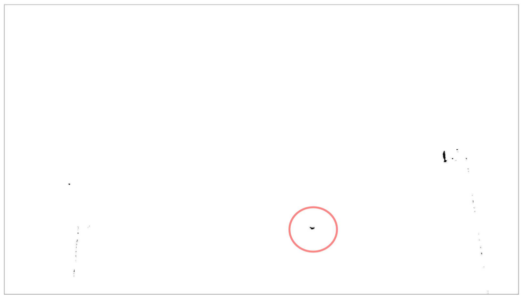
To further show the effectiveness of SNR improvement of DP, we provide results in Fig. 5.6. Though the SNR is not significantly improved, this makes a big difference when SNR is low.

5.6 Remarks about algorithm selection

The decision on which detection shceme to use should be made based on specific situations. We will use Scheme 1 when there is big contrast between the objects and the background, Scheme 2 when there is large area clutter, such as dark clouds, to


(a) Original Image.


(b) Result of Scheme 1.


(c) Result of Scheme 2.

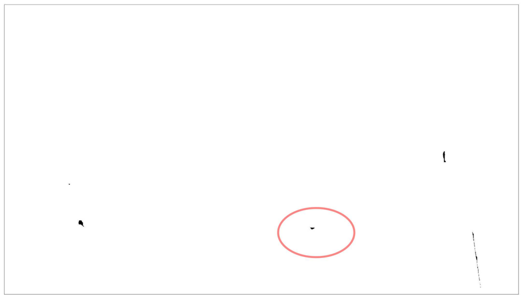

(d) Result of Scheme 3.

Figure 5.5. Object detection results for a video without cloud clutters.

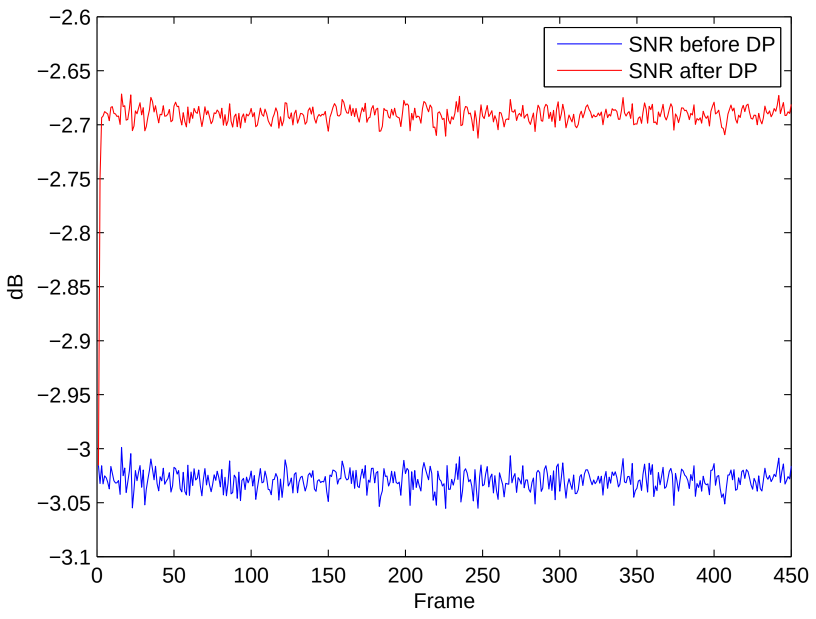


Figure 5.6. The SNR comparison.

be removed first, and Scheme 3 when the image is noisy. However, the DP is not computational efficient since the algorithm searches every pixel on the image in a recursive fashion.

6. Conclusion

This thesis has documented the following work for sense and avoid using cameras mounted on a UAV:

1. Camera calibration. There is no new contribution in this topic. It is included since it is an important part for vision-based sense and avoid in terms of accurately tracking the flying targets.

2. Camera stabilization. There are many different methods for camera stabilization, which is still an active research topic. Here, we choose to address the camera stabilization problem based on Kalman filtering and particle filtering using matched feature points obtained using SURF. We have provided an easy-to-understand derivation of the Kalman filter and summarized the essence of the particle filter. We also implemented both filters, with the particle filter used for global motion estimation and the Kalman filter for intentional motion estimation. Testing results are provided to show the effectiveness of the approach.

3. Object detection. We have focused on the issue of small target detection, which is especially important for vision-based sense and avoid. We have discussed three image processing schemes to address the problem of small target detection: Scheme 1 using a Sobel edge detector, Scheme 2 using a morphological operation

called CMO on gray-level images and then a Sobel edge detector, and Scheme 3 using dynamic programming between the two steps of Scheme 2. We have evaluated the performance of these schemes, and concluded that we can use Scheme 1 when there is big contrast between the objects and the background, Scheme 2 when there are large clutters, such as dark clouds, to be removed first, and Scheme 3 when the image is noisy.

A combination of the above processing algorithms provides a very valuable approach for vision-based sense and avoid for UAV applications.

In the future, the following aspects can be investigated to improve and evaluate the performance of vision-based sense and avoid:

- Research algorithms that are robust to the heavy clutter and environment variations will be explored. One thought is that the aircraft can be detected based on its steady moving speed between the consecutive frames. Therefore, objects with random speed can be classified as noise and outliers (Chen, Dang, Peng, & Bart Jr, 2009; Abe, Zadrozny, & Langford, 2006), which can be removed in a further process.

- Evaluate the performance based on flight simulation involving multiple aircraft and cameras. FlightGear, an open-source flight simulator, is a good candidate for this evaluation.

REFERENCES

Abdullah, L., Tahir, N., & Samad, M. (2012, 7). Video stabilization based on point feature matching technique. *Control and System Graduate Research Colloquium*, 303-307.

Abe, N., Zadrozny, B., & Langford, J. (2006, 8). Outlier detection by active learning. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 504-509.

Arnold, J., Shaw, S., & Pasternack, H. (1993, 1). Efficient target tracking using dynamic programming. *IEEE transactions on Aerospace and Electronic Systems*, *29*(1).

Arulampalam, M., Maskell, S., Gordon, N., & Clapp, T. (2002, 2). A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, *50*(2).

Barniv, Y. (1985, 1). Dynamic programming solution for detecting dim moving targets. *Aerospace and Electronic Systems, IEEE Transactions on.*

Bhadauria, H., Singh, A., & Kumar, A. (2013, 6). Comparison between various edge detectioin methods on satellite image. *International Journal of Emerging Technology and Adavance Engineering*, *3*.

Carnie, R., Walker, R., & Corke, P. (2005). *Computer-vision based collision avoidance for uavs.* Melbourne, Australia.

Carnie, R., Walker, R., & Corke, P. (2006, 5). Image processing algorithms for uav "Sense and Avoid". *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International conference on.*

Casasent, D., & Ye, A. (1997, 1). Detection filters and algorithm fusion for ATR. *IEEE Transactions on Image Processing*, *6*.

Chen, Y., Dang, X., Peng, H., & Bart Jr, H. (2009, 2). Outlier detection with the kernelized spatial depth function. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*.

Development Projects INC. (2013, 9). *Nasa centennial challenge uas-aoc rules* (Tech. Rep.). Retrieved from `http://www.nasa.gov/directorates/spacetech/centennial_challenges/uas`

Dey, D., Geyer, C., Singh, S., & Digioia, M. (2009, 7). *Passive, long-range detection of aircraft: Towards a field deployable sense and avoid system.* Proceedings of Field & Services Robotics.

Doucet, A., Freitas, N., & Gordon, N. (n.d.). *An introduction to sequential monte carlo methods.* Retrieved from `http://www.stats.ox.ac.uk/~doucet/doucet_defreitas_gordon_smcbookintro.pdf`

Fergus, R., Singh, B., Hertzmann, A., Roweis, S., & Freeman, W. (2006). Removing camera shake from a single photograph. *ACM Trans. Graph*, *25*, 787–794.

Gandhi, T., Yang, M., Kasturi, R., Coraor, L., & McCandless, J. (2003, 1). Detection of obstacles in the flight path of an aircraft. *IEEE Transactions on Aerospace and Electronic Systems*, *39*.

Gaszczak, A., Breckon, T., & Han, J. (2001, 1). Real-time people and vehicle detection from uav imagery. *Proceeding of SPIE: Intelligent Robots and Computer Vision XXVIII: Algorithms and Techniques*.

Gonzalez, R., & Woods, R. (2008). *Digital image processing* (3rd ed.). Pearson Education.

Hajri, R. (2012, 6). UAV to UAV target detection and pose estimation. Retrieved from `http://www.dtic.mil/dtic/tr/fulltext/u2/a562740.pdf`

Harris, C., & Stephens, M. (1998). A combined conrner and edge detection. *Proceedings of the Fourth Alvey Vision conference*, 147 - 151.

Haykin, S. (2009). *Neural networks and learning machines* (3rd ed.). Pearson Education.

Hobbs, A. (1991, 4). *Limitations of the see-and-avoid principle* (Tech. Rep.). Retrieved from `https://www.atsb.gov.au/publications/1991/limit_see_avoid.aspx`

Hruska, R., Lancaster, G., Harbour, J., & Cherry, S. (2005, 9). Small UAV-acquired, high-resolution, georeferenced still imagery. *conference: AUVSI Unmanned Systems North America*.

IDS. (n.d.). Retrieved from `https://en.ids-imaging.com/store/produkte/kameras/gige-kameras/show/all.html`

Jaron, P., & Kucharczyk, M. (2012). Vision system prototype for UAV positioning and sparse obstacle detection. Retrieved from `http://www.diva-portal.se/smash/get/diva2:832010/FULLTEXT01.pdf;jsessionid=Iqf9smVDR_7DqAuk08Gd7xKFkKcBIJH3zhqMWZvt.diva2-search7-vm`

Kaur, A., Kaur, L., & Gupta, S. (2012, 12). Image recognition using coefficient of correlation and structure similaity index in uncontrolled environment. *International Journal of Computer Applications*, *29*.

Lee, B., Yan, J., & Zhuang, T. (2001). A dynamic programming based algorithm for optimal edge detection in medical images. *Medical Imaging and Augmented Reality, 2001. Proceedings. International Workshop on*, 193-198.

Lin, C., Hong, C., & Yang, C. (2009, 3). Real-time digital image stabilization system using modified proportional integrated controller. *IEEE Transactions on Circuits and Systems for Video Technology*, *19*.

Mackay, D. (n.d.). *Introduction to monte carlo methods.* Retrieved from `http://www.inference.phy.cam.ac.uk/mackay/erice.pdf`

Maini, R., & Himanshu, A. (2009, 2). Study and comparison of various image edge detection techniques. *International Journal of Image Processing*, *3*.

Manjunath, B., Shekhar, C., & Chellappa, R. (n.d.). *A new approach to image feature detection with applications.* Retrieved from `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1.3625&rep=rep1&type=pdf`

Maragos, P. (1987, 7). Tutorial on advances in morphological image processing and analysis. *Proceeding of SPIE0707. Visual Communications and Image Processing*.

Matsushita, Y., Ofek, E., Ge, W., Tang, X., & Shum, H. (2006, 7). Full-frame video stabilization with motion inpainting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*.

Mohammadi, M., Fathi, M., & Soryani, M. (2011, 6). A new decoder side video stabilization using particle filter. *Systems, Signals and Image Processing (IWSSIP), 2011 18th International article on*, 1-4.

Moses, A. (2013). Radar based collision avoidance for unmanned aircraft systems. *Electronic Thesis and Dissertations*.

Orlande, H., Colaco, M., Dulikravich, G., Vlanna, F., daSilva, W., daFonseca, H., & Fudym, O. (n.d.). *Kalman and particle filters.* Retrieved from `http://www.sft.asso.fr/Local/sft/dir/user-3775/documents/actes/Metti5_School/Lectures&Tutorials-Texts/Text-T10-Orlande.pdf`

Perreault, B. (2012, 4). *Introduction to the Kalman filter and its derivation.* Retrieved from `https://www.academia.edu/1512888/Introduction_to_the_Kalman_Filter_and_its_Derivation`

Pinto, B., & Anurenjan, P. (2011, 2). Video stabilization using speeded up robust features. *Communications and Signal Processing (ICCSP), 2011 International conference on*, 527-531.

Rozantsev, A. (2009, 5). Visual detection and tracking of flying objects in unmanned aerial vehicle. Retrieved from `http://wiki.epfl.ch/edicpublic/documents/Candidacy%20exam/PR13Rozantsev.pdf`

Sang, N., Zhang, T., & Wang, G. (n.d.). Gray scale morphology for small object detection. *Proc. SPIE2759. Signal and Data Processing of Small Targets*, *2759*.

Shah, S. (2009, 8). *Vision based 3D obstacle detection using a single camera for ROBOTS/UAVs.* Retrieved from `https://smartech.gatech.edu/bitstream/handle/1853/29741/shah_syed_i_200908_mast.pdf`

Shrivakshan, G., & Chandrasekar, C. (2012, 9). A comparison of various edge detection techniques used in image processing. *International Journal of Computer Science Issues*, *9*.

Song, C., Zhao, H., Jing, W., & Zhu, H. (2012, 5). Robust video stabilization based on particle filtering with weighted feature points. *IEEE Transactions on Consumer Electronics*, *58*.

Sonka, M., Hlaval, V., & Boyle, R. (2013). *Image processing, analysis, and machine vision* (4th ed.). Cengage Learning.

Tong, C., Kamata, S., & Ahrary, A. (2009, 11). 3D face recognition based on fast feature detection and non-rigid iterative closet point. *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International conference on*, *4*, 509-512.

Tonissen, S., & Evans, R. (1995, 12). Target tracking using dynamic programming algorithm and performance. *Decision and Control, 1995., Proceedings of the 34th IEEE conference on*, *3*, 2741-2746 vol.3.

Tsai, R. (2003, 1). A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal of Robotics and Automation*, 323-344.

Yang, J., Schonfeld, D., Chen, C., & Mohamed, M. (2006, 10). Online video stabilization based on particle filters. *Image Processing, 2006 IEEE International conference on*, 1545-1548.

Yang, M., Gandhi, T., Kasturi, R., Coraor, L., Cmaps, O., & McCandless, J. (2002). Real-time implementation of obstacle detection algorithms on a datacube maxpci architecture. *Real-Time Imaging*.

Yusko, R. (2007, 3). *Platform camera aircraft detection for approach evaluation and training.* Retrieved from `http://www.dtic.mil/dtic/tr/fulltext/u2/a467710.pdf`

Zarandy, A., Zsedrovits, T., Nagy, Z., Kiss, A., & Roska, T. (2011, 5). Collision avoidance for UAV using visual detection. *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, 2173-2176.

Zarandy, A., Zsedrovits, T., Nagy, Z., Kiss, A., & Roska, T. (2012, 8). Visual sense-and-avoid system for UAVs. *Cellular Nanoscale Networks and Their Applications (CNNA), 2012 13th International Workshop on*, 1-5.

Zhou, S., Chelleppa, R., & Moghaddam, B. (2004, 12). Visual tracking and recognition using appearance-adaptive models in particle filters. *IEEE Transactions on Image Processing*.