学校代码	
分 类 号	
密　　级	
学　　号	2018280049

题目 Research of Approach Strategy for Fast Drone Autonomous Landing

作者 Julien Mellet

学 科、 专 业 导航、制导与控制

指 导 教 师 王靖宇

申请学位日期 2020 年 5 月

西 北 工 业 大 学

硕 士 学 位 论 文

(学位研究生)

题目 : Research of Approach Strategy for Fast Drone Autonomous Landing

作　　者 : Julien Mellet

学科、专业 : 导航、制导与控制

指 导 教 师 : 王靖宇

2020 年 5 月

Research of Approach Strategy for Fast Drone Autonomous Landing

By
Julien Mellet

Under the Supervision of Professor
Wang Jingyu

A Dissertation Submitted to
Northwestern Polytechnical University

In Partial Fulfillment of the Requirement
for the Degree of
Master of Navigation, Guidance and Control

Xi'an P. R. China
May 2020

Abstract

There is an increasing demand for fast drone deliveries for both consumer's necessities and medical emergencies. All aircraft are subject to landing difficulties because of near-ground aerodynamic effects and complications with moving from air to land. In addition to improving delivery, there have been many developments within space research aiming to reduce global cost. Reusable rockets are regarded as the future of space travel, however, numerous accidents have been observed during landing, making these rockets an unreliable resource for experimentation. The standard approach for both drone delivery and reusable rockets requires speed reduction before landing, as explained by the lack of precision of the embedded sensors combined with the fragility of the frame. The landing shock thus needs to be attenuated to deform irremediably the frame structure. Therefore there is a need for improvement of landing approach for more reliability. The main research contents of this thesis cover different areas for the development of a newly designed electric thrust vectored rocket that aims to incorporate a fast landing approach using this new benchmark system.

Firstly, this research proposes an improved perception with a convolutional neural network architecture for landing pad localization. This is an efficient and robust way to estimate target localization that provides a larger field of view than using the traditional ArUco marker. The lack of a dataset for visual pose estimation in the landing approach forced the research to develop its own. The use of a realistic 3D motor engine made it possible to manage all the parameters of the generated images. An appropriate landing pad shape is also proposed to permits a large range of target vision, even seeing only part of the landing area.

Secondly, a robust control algorithm has been designed for visual servoing. A multi-scale control is proposed to increase drone landing accuracy using a fractal targeted shape for the landing pad. The use of optimal pose estimation, even with a low measurement rate, has increased the drone landing accuracy of the ArUco marker. Using the neural network in a simulated environment, the processing rate has been multiplied by three. The reduction of the landing time was considerably faster than the traditional landing approach, which was a result of increasing the speed at landing.

Finally, this thesis proposes an innovative landing leg kinematic to help at a high-speed landing. The landing collision is smoother with a real reduction of the bounce effect, absorbing more than ten times the kinetic energy of a standard landing leg. Two types of landing gears have been proposed: a standard mechanism that confirmed the interest of absorption of the landing velocity, and an improved compliant mechanism. The second one shows better effects at landing

with a lighter weight and more internal frictions. The use of adapted gears thus provides greater protection for the drone against landing impact.

Key words : Compliant Mechanism, Deep Learning, Convolutional Neural Network, Kalman Filter, Multi-Scale

Table of Contents

Acronyms

CNN - Convolution Neural Network ;
DNN - Deep Neural Network ;
DOF - Degree Of Freedom ;
EDF - Electric Ducted Fan ;
ENU - East-North-Up ;
FC - Flight Controller ;
FLU - Front-Left-Up ;
FPS - Frame Per Second ;
GCS - Ground Control Station ;
GNC - Guidance, Navigation and Control ;
GPS - Global Positioning System ;
IBVS - Image-Base Visual Servoing ;
IMU - Inertial Measurement Unit ;
KF - Kalman Filter ;
NN - Neural Network ;
PBVS - Position-Based Visual Servoing ;
PID - Proportional–Integral–Derivative ;
PLA - PolyLactic Acid ;
PnP - Perspective-n-Points ;
PP - PolyPropylene ;
RC - Radio Control ;
RGB - Red Green Blue ;
RNN - Regression Neural Network ;
ROS - Robot Operating System ;
SITL - Software In The Loop ;
UAV - Unmanned Aerial Vehicle.

List of Tables

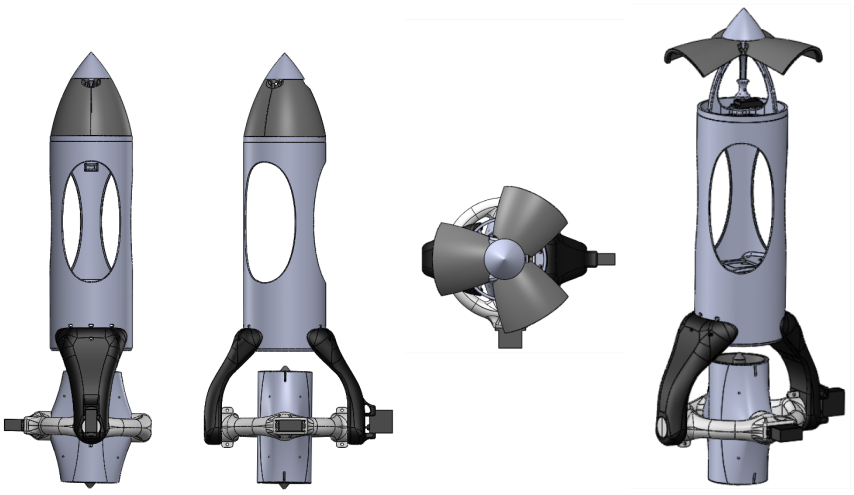
List of Figures

1 Introduction

1.1 Background and Motivation

There is an increasing desire to go space, however, space travel is still extremely costly and there are many uncertainties that remain present. The development of reusable rockets is necessary because it drastically decreases the price of rocket production and ultimately space travel. However, current simulations of rocket algorithms do not always suit reality as there are always unexpected variations that occur between the simulation and the real-world. The electric thrust vectored rocket has the advantage to overcome these variations and be more reliable than a simulation. The rocket also has the advantage to be used in the real world through Guidance, Navigation and Control (GNC) algorithm, and a unique frame that can provide a vectored thrust with intensity control. Although the electrical rocket (e-rocket) has numerous promising qualities in addition to being sustainable, there are still some existing issues that remain, which include the lack of combustion that forces the rocket to stay within the atmosphere.

Here is shown the electric thrust vectored rocket visible on the figure below. The body part of the e-rocket is developed in parallel to the thesis in another project. Most of the components used in the e-rocket come from drone components, in other words, it is controlled like a drone and can easily be substituted for the development. The motors are Electric Ducted Fan (EDF) mounted in series on a 2-axis gimbal, and the entire structure is PLA 3D printed.



(a) Profile view (b) Face view (c) Top view with air brakes opened (d) Three quarter view with air brakes opened

Fig. 1-1 General assembly views of the e-rocket

The e-rocket is cheap and has two main applications: help rocket labs develop algorithms for rockets; and other industrial applications that include emergency deliveries or fast deliveries. The e-rocket can provide the five stages of flying systems listed [1]: takeoff, climb, cruise, descent, and landing. During the descent phase, air brakes open like in Fig. 1-1d Instead of using the e-rocket, this paper uses a standard quadcopter Unmanned Aerial Vehicle (UAV) and focuses on the autonomous landing stage at the end of the descent. The interest points are in the light-weight embedded hardware for system localization, GNC algorithm, and specific landing gears.

As the e-rocket frame was not fully functional at the time when the paper was written, the landing approach will use a standard quadcopter that perfectly replaces the system. The proposed project is to create a benchmark that can embed a GNC algorithm. It does not have combustion problems because it has drone components. The thrust is vectored with two motors in series, mounted on a gimbal. This is done to achieve a conventional rocket design, but switching the traditional combustion motor into a more stable electric motorization. The e-rocket flies at faster vertical speed than standard quadcopters, which justifies the need for further improvement.

Firstly, the landing area needs to be understood faster and more accurately than a standard landing drone. Motion capture technology like *Vicon* proposes an expensive alternative that is not applicable outside of the laboratory. GPS for long ranges and cameras near to the ground also helps obtain better localization. This study provides a monocular camera with a landing pad recognition. Traditional object recognition needs important computational consumption. The machine learning algorithm can reduce this amount of computer consumption and time. For a first Neural Network (NN) implementation, supervised learning can provide quite impressive results. The recent improvement of 3D software rendering, make virtual worlds accurately resemble the real-world. One can notice the recent use of *Unreal Engine* to instantly create the background of the series *The Mandalorian*. Then virtual data can be created for real application but this target estimation depends on the integrity of the system.

Within the drone community, many tools are open-source to help drone autonomous implementation. The autopilot is a fork of the last stable version of the PX4 software. The work will not recognize a low-level algorithm for stability. It will, in turn, use a higher control level of the system with horizontal and vertical speed control. Landing pad localization is done on a companion computer, and landing pad localization is enhanced using a Kalman Filter (KF) application. The localization is multi-scale controlled to have specific control of the altitude.

Observation of every kind of landers like rockets or quadcopters shows uncontrolled bumps at ground impact. Even with a low vertical speed, Space-X boosters or drone landing [2] have

either crashed or have been close to an incident. A good example of this is the comet landing of Rosetta, where the bounce of the space lander has so strong and almost ended the mission. In this thesis, the drone will land on a prepared hard ground. In contrary to sandy terrain which absorbs kinetic energy, a solid landing pad returns all of the energy. The landing gear is composed of many pieces, but this complexity can be resolved using one unique piece. Research on compliant landing gears can keep the necessary kinematic and absorption effect with a specific design.

1.2 Relative Works

Autonomous drone landing is a robotic project combining signal processing, control theory, and mechanical design. The following review shows state of the art about drone autonomous landing and more specifically, landing with visual servoing. This section begins with related papers about autonomous landing UAVs projects, then focuses on image processing for landing pad localization, then follows the guidance of the drone using visual data, and finishes with the design of the landing gear.

1.2.1 Related Landing Project

Drone autonomous landing has been widely studied and exists extensively throughout the scientific literature. Drone landing is constantly challenging because it enrolls various fields that are state of the art. Three major surveys are noticeable. The first [3] lists different kinds of UAVs and highlights the fact that a good Flight Controller (FC) is required. There is no ideal sensor present for landing. Even if vision-based flight provides sufficient results, it might be sensible to fog and mist. The second survey [1] focuses on vision-based landing and categorizes drone systems according to their sized. The research presented in this paper uses on-board vision, in opposition to on-ground vision systems. Requirements are for precision target localization on a low-resolution camera, and a low horizontal and vertical speed to protect the equipment. The third survey, the most recent one [4] has the same challenging problem using a low-resolution camera. [5] reaches to control drone position in a GPS-denied environment with a low-cost drone. The result is inaccurate due to the poor quality of the sensors.

Concerning accuracy, most of the presented projects have a decimeter accuracy at landing. [6] had for instance a $40cm$ error in position for *73 s* to land from a *10 m* height in 2002. Recent study like [7] improve the performance with a *20 cm* error for a *10 s* land from *3 m* altitude. Although the more accurate ones do not specify the duration of the manoeuvre.

These surveys are now outdated and many new drone autonomous landing papers have been published. However, recent studies can answer recent problems with other standards tools and techniques. Recent studies like [8] [9] embed a NN algorithm to try to increase pose estima-

tion rate and accuracy. The request for high-speed pose estimation and control for autonomous drone racing is reviewed in the drone control section.

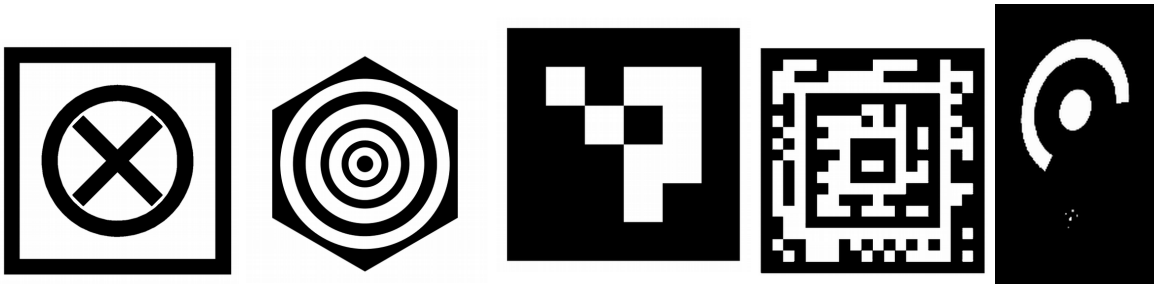
Autonomous drone landing have been done on moving platform [7, 10-13] even with high speed [2]. The approach to land on target having a uniform rectilinear movement is sensibly the same as to land on a static target.

After all, the use of a camera sensor seems to provide cheap and lightweight UAV pose estimation. The following section reviews the use of image processing for pose estimation.

1.2.2 Image Processing

Visual targeting tries to get an accurate localization of the drone relative to the landing pad. Most of the landing techniques involve a coded target to signal the target location. [10] describes an autonomous precision landing technique utilizing QR-codes in an image-based visual servoing control loop for the descent. In the presented method, the majority of the control calculations occurred in image space to reduce computational hardware requirements. Similarly, [14] describes a vision algorithm for detecting a custom landing target, based upon concentric rings. Moreover, their approach utilized in-depth information extracted from the landing target that guided the multi-rotor to the landing target.

A specific image pattern is needed to locate the target.



(a) X-shaped marker (b) Circles marker (c) Square-based fiducial marker (d) Fractal marker (e) Complex shape

Fig. 1-2 Landing pad shapes

Fiducial markers 1-2c have proved their reliability for aggressive drone landing [2] and have recently been used [15]. A recent proposed improvement is the use of fractal marker 1-2d, which allows a wide range of pose estimations and is robust to partial cover. [16] propose a library, like [17] a multi-scale target detection is possible.

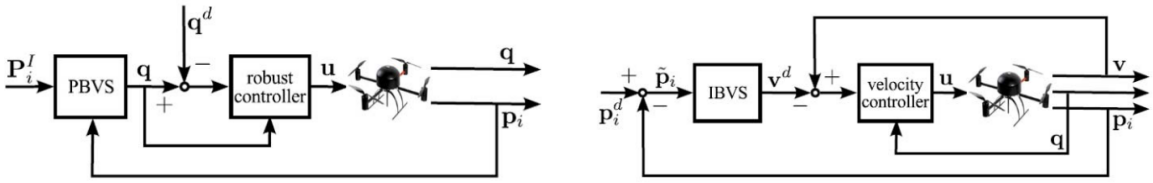
Then more complex shape 1-2e are used for NN algorithms like in [8]. For spaceships in deep space, many algorithms are used for object detection [18-22]. In the present day, a lot of algorithms perform this recognition using deep learning methods. Object detection in images is the traditional form of this problem, but recently there are many different specializations of the classic problem [23, 24]. Recent papers have shown the use of target detection for drone landing that uses these NN algorithms [9, 25, 26]. This type of algorithm [27] can also detect other features.

Dataset for landers exist, but are specific to deep-space exploration [28]. A recent dataset [29] has been published, but it is specific to drone racing. It provides 27 sequences of racing drones piloted in First Person View (FPV) and is inspired by autonomous driving cars. The drone racing dataset contains synchronized IMU, camera, and event camera data recorded in indoor and outdoor environments. This lack of dataset even pushed [30] to create the dataset with a ground vehicle. Otherwise, in a previous study [31], virtual learning was applied to a robot, which was then able to decipher different shapes from one another, proving that virtual learning can be applied in real-world scenarios. Target images need to be combined with their respective 6 DOF localization. Regression in supervised learning, oftentimes only use numerical data. [32] provides a method that mixes numerical data and images in supervised learning for house pricing estimation.

Two methods seem to get similar results. The first one is to use a fiducial marker and standard image processing to estimate drone pose relative to the marker. The second one is to use a specifically trained NN. It should estimate drone localization with a faster sampler rate and thus perform real-time computing. The better the pose of the landing pad area is, the more accurate the drone control should be.

1.2.3 Drone Control

The quality of drone control depends on the quality of the measured data. Robust trajectory landing was firstly developed for deep space exploration [33]. Quadrotor's literature [4] name controlling the motion with visual feedback as visual servoing. An Image-Based Visual Servoing (IBVS) is seen in figure 1-3b to have better results than Position-Based Visual Servoing (PBVS) visible on 1-3a.



(a) The block diagram of the PBVS control

(b) The block diagram of the IBVS control

Fig. 1-3 Difference between PBVS and IBVS block diagram [4]

Fast control for aggressive maneuvers is still a struggle for drone control. [29] have for instance been published to help development of IROS autonomous drone race [34]. And contrary to IBVS, impressive controlled results [35-39] required a precise understanding of the environment using motion capture technology.

Optimal estimation is based on KF [40]. Multi-scale perception has proved its reliability on horizontal high-speed landing [2], and it is often combined with a Proportional–Integral–Derivative (PID) controller [41, 42]. Even waypoint navigation [43] can be controlled with a PID with high accuracy results. Other control including NN [44, 45] or model predictive control [7, 12, 46] wont be studied in this paper.

The use of ROS (Robot Operating System) made an easier implementation of drone control using image-based pose estimation [47]. The combo ROS with the Gazebo simulator is now used to develop a control algorithm on the Linux platform [48, 49]. This last one introduces the use of ArduPilot autopilot firmware, but PX4 autopilot provides the same kind of use as [50, 51]. Moreover, PX4 seems to have proven its reliability on [11, 52]. This autopilot is mainly used with the Pixhawk [53] hardware which is simulated on PX4 with Gazebo.

Thus, using image processing with smart data filtering, and accurate servoing can be done. This allows further to perform aggressive drone control.

1.2.4 Mechanical Leg

Aggressive landing needs to embed an adapted mechanical structure. Space landers are the most studied mechanism that absorbs landing impact like for this lunar lander[54]. Recently [55] proposed a new type of mechanism for space landers. At landing, the theoretical mechanism, absorb impact energy by transforming kinetic energy into elastic energy.

Two notable papers exist regarding drone landing gears. The first one [56] is a theoretical study that provides an active bionic leg. The second one [57] is a bio-inspired landing leg, which facilitates landing on different surface types. When studied on 2cm displacement, the

absorption effect was almost similar to other landing gears. These legs embedded servo-motors for retraction during flight, which increased the dead weight.

Recent studies on compliant mechanisms show different advantages. It is possible to reproduce mechanisms and keep the same properties. [58] for swivel or [57] for spring. For instance, [59] propose pneumatic origamic legs. Another example of a soft landing leg is shown on [60].

Thus, reviews of this section try to find a lightweight landing gear that performs a fair energy absorption. Only [55] performs a tangible energy transformation.

1.3 Main Work and Organisation

The research begins with the basic theory in Chapter 2. All mathematical tools and notations used are presented. Because it is robotic work, different fields of science cross over each-other. Conventions to write equations are also fixed to help the reader.

Chapter 3 follows the perception embedded in the drone. The study provides a lightweight landing pad position estimator, by only using a monocular camera. Two kinds of the algorithm are presented to be compared. Both of them will have a preprocessed image from the camera. The traditional fiducial marker is firstly presented. It allows good localization of marker in space, including rotations related to the camera. Secondly, the smart implementation of NN is used. Well trained NN are known to be fast algorithms with what can be called intuition and fast calculations. This position estimation aims to provide a more precise position estimation at a faster rate. A virtual dataset is generated and allows specific training of the NN with perfect camera localization. Image parameters are also fixed to match the implementation. Here *OpenCV* with *Python* permits image processing.

Chapter 4 presents guidance and control implemented in the drone. During the standard landing, drone descends too slowly and lacks precision. Thus, the drone needs to have a faster vertical velocity, and the position needs to be computed to land more accurately. The modules used are introduced and communication between each-other is explained. Then the thesis presents simulation using Software In The Loop (SITL) and implementing its control. The simulator is *Gazebo*. It is made for the computer vision application and consumes relatively low computing power. A Bayesian optimal estimator provides a filtered system position to land with greater accuracy. To ensure full control of the drone, a scale-adjusted control was created to maintain accuracy before and throughout the descend.

The last part of this thesis presents the mechanical absorption of drone gears in Chapter 5. The purpose of this part is to increase drone speed at landing. The paper will examine the air

to land effect, and the collision absorbed by the structure. It begins with the simulation of the drone bounce at landing. This is a study of the effects of shock absorber for drone landing. This paper will also explore the maximum impact velocity a drone can reach, how kinetic energy is absorbed, and the landing effects associated with a light landing leg. This section use *MATLAB* and *Simulink*. Then the proposed shock absorber kinematic is presented, designed with *Solidworks 2018*. A compliant version of the landing gear tries to have the same kinematic linkage, bounce, and absorption.

2 Basic Theory

Robotics often mix different fields of research. Here image processing, deep learning, control theory, and mechanics will be reviewed. The theoretical background of this different area is presented in this above.

2.1 Image Processing

According to [61] work, vision is the most advanced sense, that why it plays a major role in human perception.

Digital image processing mainly stems from two application areas: improvement of information for human interpretations and image processing for storage, transmission, and representation for the autonomous machine. An image is defined by a two-dimensional function $f(x, y)$. Thus (x, y) are coordinates of the spatial plane and f is the gray level intensity of the picture. It is also called a pixel.

2.1.1 Image Processing Background

Notation for $N \times M$ image will follow this compact matrix form,

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1,N-1) \end{bmatrix} \tag{2-1}$$

Each element of the matrix is a pixel of the digital image. When matrix will be used in the above section, the standard convention will be used,

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & \dots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \dots & a_{1,N-1} \\ \vdots & \vdots & & \vdots \\ a_{M-1,0} & a_{M-1,1} & \dots & a_{M-1,N-1} \end{bmatrix} \tag{2-2}$$

Pixel Neighbors

At coordinate (x, y) a pixel p has 4 horizontal and vertical neighbors called *4-neighbors* and denoted $N_4(p)$,

$$N_4(p) = (x+1, y), (x-1, y), (x, y+1), (x, y-1) \tag{2-3}$$

The four *diagonal neighbors* $N_D(p)$ of p are,

$$N_D(p) = (x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1) \tag{2-4}$$

All the neighbors together are the *8-neighbors* such as,

$$N_8(p) = N_4(p) + N_D(p) \tag{2-5}$$

This can be summarised with the following figure 2-1,

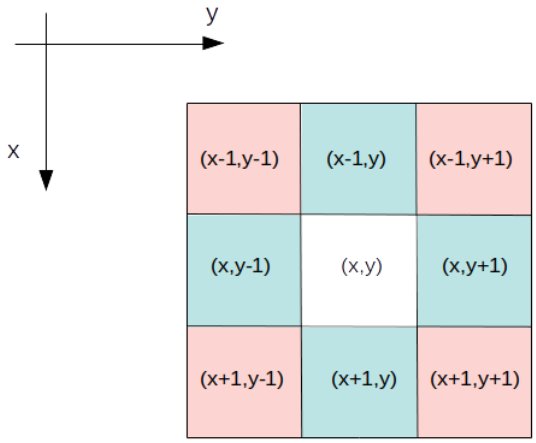


Fig. 2-1 8-neighbors of a pixel ; 4-neighbors in blue ; diagonal neighbors in red

The neighbors can be generalized with more layers of pixels around the target pixel in white in Fig. 2-1.

2.1.2 Image Enhancement

The purpose of enhancement is to process a picture for a specific application and have a result more useful than the original image. It allows us to highlight details, remove noise, and, for a subjective point of view, making an image more appealing. Two techniques are used: spatial domain technique by direct manipulation of the image, and frequency domain technique.

Input image f is though transformed by T to an output g,

$$g(x, y) = T[f(x, y)] \tag{2-6}$$

Basic transformations are then presented for gray level image range [0, L-1] where $L \in]0, \mathbb{N}]$

2.1.3 Basic Transformations

Transformations are used to enhance some image parts and highlight the expected information. The following examples are the most used for image processing.

Negative The negative transformation allows negative pixel values to be passed positively and vice versa. This is defined with the equation 2-7,

$$g(x, y) = L - 1 - f(x, y) \tag{2-7}$$

A graph representation of the negative transformation is shown in the following graph Fig. 2-2,

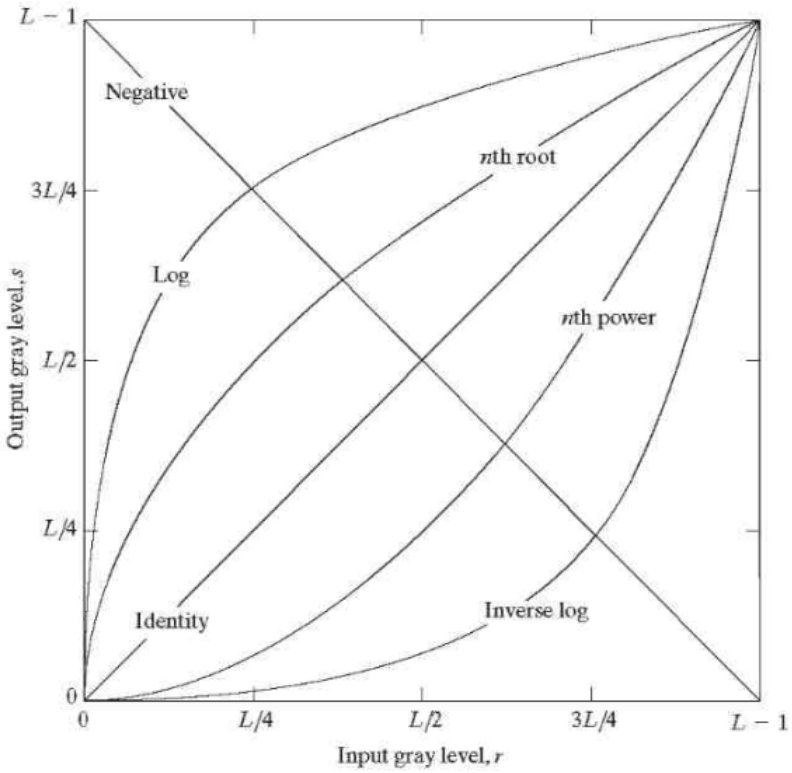


Fig. 2-2 Some basic gray-level function for enhancement

The figure 2-2 also graphically shows how the logarithmic and power-law transformations are taking place. These two equations are respectively written in equation 2-8 and 2-9 below,

Logarithmic

$$g(x, y) = a + \frac{ln(f(x, y) + 1}{b.ln(c)}, (a, b) \in \mathbb{R}^2 \tag{2-8}$$

Power-law

$$g(x, y) = c.f(x, y)^{\gamma}(c, \gamma) \in \mathbb{R}^{+^2} \tag{2-9}$$

A more precise representation of the power-low is given on the following graph Fig. 2-3,

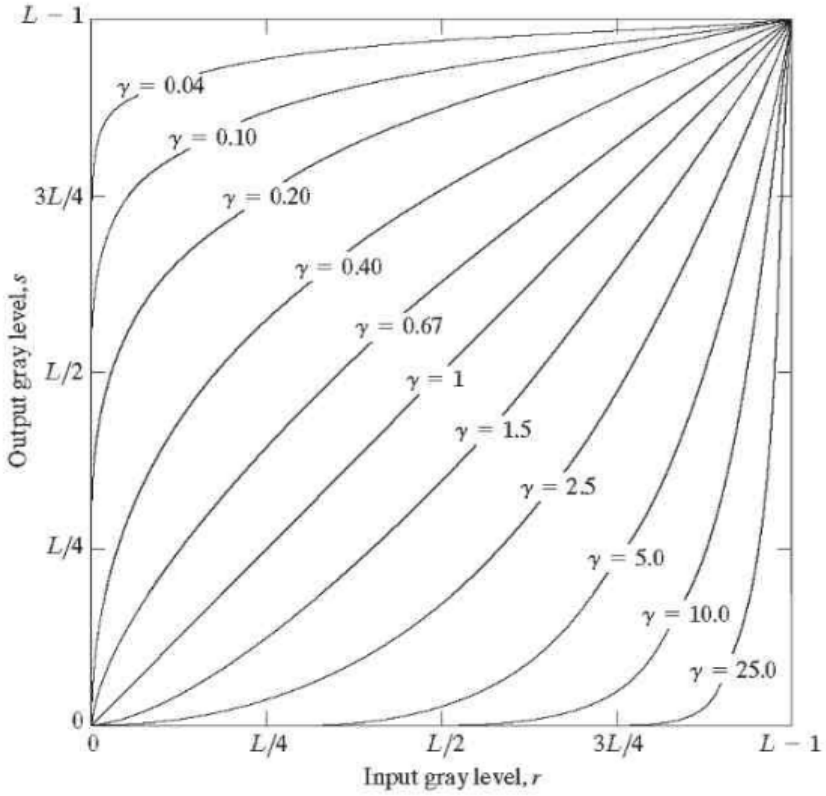


Fig. 2-3 Plots of power-law equation for different γ and $c = 1$

Piecewise Linear Transformation None continuous enhancement is often provided with a piecewise linear function such as the following one in Fig. 2-4,

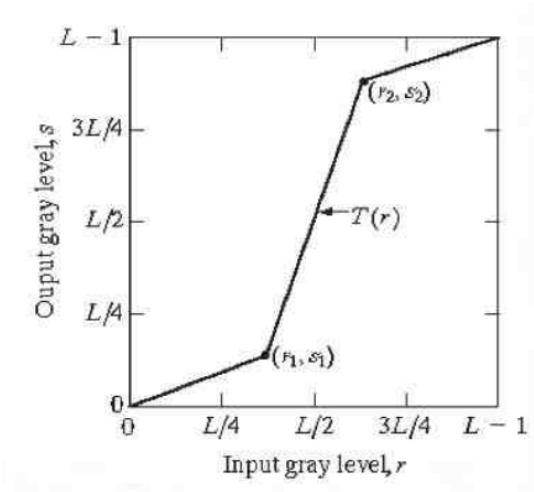


Fig. 2-4 Example of piecewise transformation function

In this example Fig. 2-4, first linear function on $[0, S_1]$ is,

$$g(x,y) = \frac{S_1}{r_1} f(x,y) \tag{2-10}$$

Other examples with intensity-level slicing are then presented in figure 2-5,

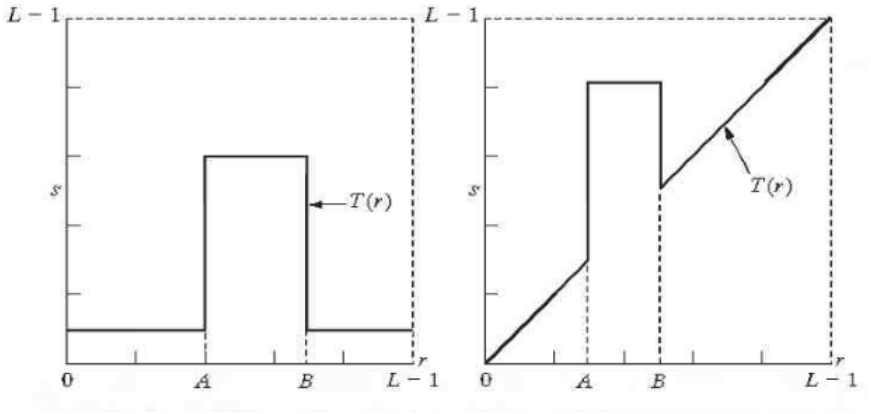


Fig. 2-5 Two examples of intensity-level slicing transformation function

Histogram Processing

Histogram of an image is a discrete function,

$$h(r_k) = n_k \tag{2-11}$$

Where r_k is k^{th} intensity value and n_k is the number of pixels. It is common to normalize the histogram,

$$p(r_k) = \frac{n_k}{n} \tag{2-12}$$

Where n is the total number of pixels.

Better images have spaced histograms because all possible frequency space is used. But it is notable that different images can have the same histogram.

2.1.4 Camera Calibration

This work uses the pinhole camera model. Using optic lenses has an effect to deform image. Camera calibration allows determining the relation between pixels and real-world distances. In this model 3D points are projected into the 2D image plan with perspective transformation such as,

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x_{3D} \\ y_{3D} \\ z_{3D} \\ 1 \end{bmatrix} \tag{2-13}$$

Where (u, v) are coordinates in 2D pixels of the 3D point; (x_{3D}, y_{3D}, z_{3D}) are the 3D coordinates of the point in the world; (c_x, c_y) are coordinates of the image center; f_x, f_y are focal lengths expressed in pixel units.

The matrix composed of c_x, c_y, f_x, f_y is called matrix of *intrinsic parameters*. This matrix is proper for a camera and needs to be calculated only once. The joint rotation and translation matrix are called the matrix of extrinsic parameters. It describes the relative motion between the camera and the point. This section shows how to find intrinsic and extrinsic parameters of the camera, knowing other vectors. But equation 2-13 will be used further reversely, to estimate the 3D position of a marker into its corresponding 2D image. This is called Perspective-n-Point (PnP) and can be resolved using different methods [62].

With $z \neq 0$, one can write 2-13,

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x_{3D} \\ y_{3D} \\ z_{3D} \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \tag{2-14}$$

and

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix} \tag{2-15}$$

Where,

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \end{bmatrix} \tag{2-16}$$

Radial distortion is the one known using a wide-angle camera as a ”fish-eye” effect. All cameras are prompt to this effect with a different range. Radial and tangential distortion factor have been modeled. One can thus re-write 2-15 to let appear coefficients,

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix} \begin{bmatrix} x'' \\ y'' \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix} \tag{2-17}$$

Where,

$$\begin{bmatrix} x'' \\ y'' \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix} \begin{bmatrix} \frac{1+k_1r^2+k_2r^4+k_3r^6}{1+k_4r^2+k_5r^4+k_6r^6} + 2p_1y' \\ \frac{1+k_1r^2+k_2r^4+k_3r^6}{1+k_4r^2+k_5r^4+k_6r^6} + 2p_2x' \end{bmatrix} + \begin{bmatrix} p_2(r^2+2x'^2) \\ p_1(r^2+2y'^2) \end{bmatrix} \tag{2-18}$$

and $r^2 = x'^2 + y'^2$ and 2-16. Thus $k_1, k_2, k_3, k_4, k_5, k_6$ are radial distortion coefficients ; p_1, p_2 are tangential distortion coefficients. Higher order coefficients are not considered in this study.

2.1.5 Convolution operation

This will also be used for the convolution neural network (CNN) explained further. Convolution operator is denoted $*$. It is a mathematical operation between two functions f and g that shows how one is modified by the other. It is defined as the integral of the product of the two functions after one is reversed and shifted.

$$(f * g)(t) = \int_{-\infty}^{+\infty} f(\tau)g(t-\tau)d\tau \tag{2-19}$$

Or by commutativity,

$$(f * g)(t) = \int_{-\infty}^{+\infty} f(t-\tau)g(\tau)d\tau \tag{2-20}$$

t doesn't need to represent the time domain.

2.2 Neural Network

A neural network is a multi-layered decision system. Information is fed into the network, then follows linearly and non-linearly transformations to make a final decision. Decades of research have produced many different techniques, but these have increased these last couple of decades. The better graphics processing unit (GPU) helped in this spread. The term "neural network" was chosen in attempts to find mathematical representations of information processing in biological systems.

The functional form of the neural network provides a specific parameterization of the basis function. Follows here, how to determine parameters of the network by maximum likelihood,

this requires the technique of error backpropagation. Next is the regularization of neural network training and the relation between them.

2.2.1 Feed Forward

In a neural network, feed-forward correspond to go from one layer to another, without backward. We first construct M linear combinations of $x_1, x_2, ..., x_n$ input. The basic neural network model has this form,

$$y_i(x, w) = f(\sum_{i=1}^{D} w_{ij} x_i + w_{j0}) \tag{2-21}$$

Where $j = 1, 2.., M$ is the parameter of corresponding layer; w_{ij} is the weight; w_{j0} is the biases; f activation function fixed during training.

This equation allows to be in simple linear regression problem. Then, two feedforward networks can be presented,

- *Single Layer Perceptron*: This is the simplest feedforward network [63]. It won't be developed here because it is not used for CNN contrary to the next one.

- *Multi Layer Perceptron*: This has one or more hidden layers. It will be developed in the section below.

Multi-perceptron layer

For the multi-perceptron layer, two layers are very popular architecture. In a layer, a biases neuron needs to be added. Taking the previous equation, we can simplify it as follow,

$$\begin{cases} a_j = \sum_{i=1}^{M} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \\ z_j = h(a_j) \end{cases} , j \in [1, M] \tag{2-22}$$

Where j is the output of the first layer; i is input; $^{(1)}$ is the layer number; for multilayer x_i become a function as equation (1).

Graphic representation

The graphic representation of the previous equation 2-22 is this one,

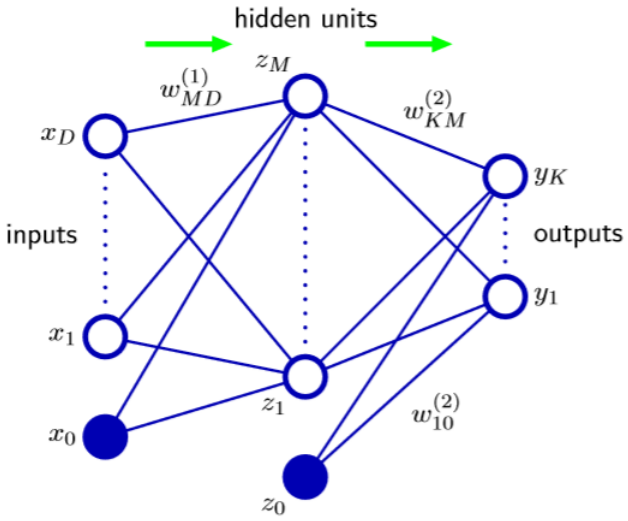


Fig. 2-6 Network diagram for the two layer neural network

Three types of nodes present on the figure 2-6 exist for a feedforward neural network,

- *Input Neuron*: $x = [x_0, x_1, ..., x_D]$ provide information from outside into the network and compose the input layer. There is no computation at this level, information just pass through it.

- *Hidden Neuron*: $z = [z_0, z_1, ..., z_M]$ has no link with outside, this is why it is ”hidden”. Hidden layers are a collection of hidden neurons that links inputs to outputs.

- *Output Neuron*: $y = [y_1, ..., y_k]$ make the output layer and transfer information from the network to the outside.

The weight parameters are represented by links between the nodes, in which the bias parameters are denoted by links coming from additional input and hidden variables x_0 and z_0. Arrows denote the direction of information flow through the network during forwarding propagation.

Sometimes, for some input, a layer can be skipped, but this is not very often used. This case is presented in the following figure 2-7,

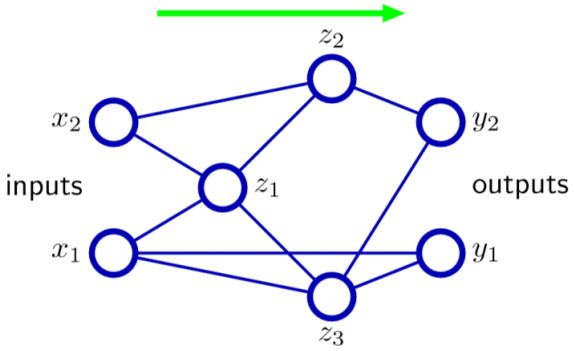


Fig. 2-7 Neural network diagram with hidden neurons

Hidden neurons decrease the complexity of the algorithm and make it more efficient. This can be done with *TensorFlow* Python's library.

2.2.2 Neural Network Training

After linking input vector x to output vector y with a function f. A simple analogy for determining weights w polynomial curve fitting. Here are presented two main ways to minimize the error function.

Curve Fitting

Therefore the error function has to be minimized, knowing target vectors t_n,

$$E(w) = \frac{1}{2}\sum_{i=1}^{N} \|y(x_n, w) - t_n\|^2 \tag{2-23}$$

To optimize linear regression, maximum likelihood function can be used, and it is equivalent to minimize the sum of squares.

If a training set of independent observations is considered, then the error function, which is given by the negative log-likelihood, is then a $cross - entropy$ error function of the form,

$$E(w) = -\sum_{i=1}^{N} t_n ln(y_n) + (1 - t_n) ln(1 - y_n) \tag{2-24}$$

Parameter Optimization

From a geometrical point of view, it is equivalent to search,

$$\nabla E(w) = 0 \tag{2-25}$$

And this is the global minimum of a 3D function. The case is presented on the figure 2-8 below,

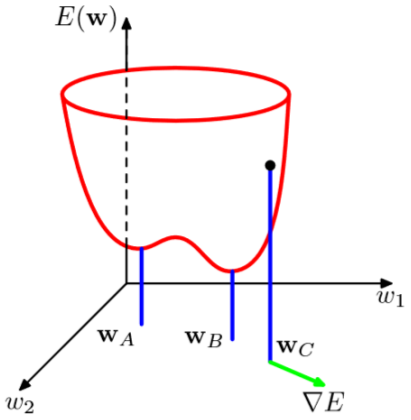


Fig. 2-8 Geometrical view of the error function E(w)

Fig. 2-8 shows w_A as local minimum, while w_B is the global minimum.

Activation Function

An activation function is a node, added the output of a layer or between two layers. Literature also calls it neuron or unit. It allows the output of the neural network to have resulting values between 0 to 1 or -1 to 1. Some of them are more popular such as logistic sigmoid, $tanh$, and $ReLU$. Activation function takes though a number to perform a mathematical operation associated with.

Logistic Sigmoid or Sigmoid uses a real value x input to arrange it into the range [0,1],

$$\sigma(x) = \frac{1}{1 + \exp -x} \tag{2-26}$$

Hyperbolic Tangent or tanh uses a real value x input to arrange it into the range [-1,1],

$$tanh(x) = 2\sigma(2x) - 1 \tag{2-27}$$

Rectified Linear Unit or ReLU uses a real value input and threshold negative values at zero,

$$f(x) = max(0, x) \tag{2-28}$$

Following graphs inf figure 2-9 show each activation function,

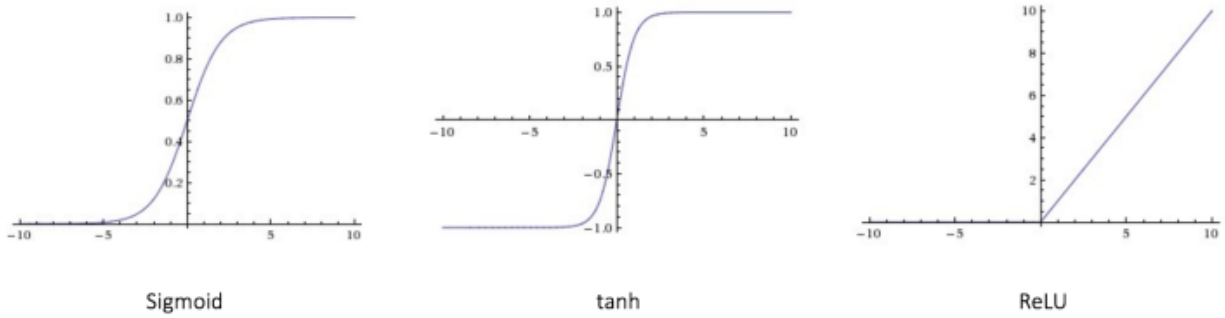


Fig. 2-9 Three different activation functions

Thus, for linear regression we use the sum of square error; for binary classification, we use logistic sigmoid and cross-entropy; for multiclass classification, we use soft-max and cross-entropy; and for CNN ReLU is traditionally more used.

Local Quadratic Approximation

For the optimization problem, we can consider a local quadratic approximation to the error function. Then Hessian matrix is used after to consider the Taylor expansion of E(w) around some point $\hat{w}$ in weight space

$$E(w) = E(\hat{w}) + (w - \hat{w})^T b + \frac{1}{2}(w - \hat{w})^T H(w - \hat{w}) \tag{2-29}$$

Where b is defined to be the gradient of E evaluated at $\hat{w}$; H is the Hessian matrix $\nabla\nabla$ E; $E(\hat{w}) + (w - \hat{w})^T b$; $\frac{1}{2}$ is to eliminate terms when we derive.

It provides better performance but less stability, it sometimes results in oscillations.

2.2.3 Error Backpropagation

Error backpropagation is an efficient technique to evaluate the gradient of an error function $E(w)$ for a feed-forward neural network. It decreases the complexity of a model. The formula provides is the following one,

$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k \tag{2-30}$$

This tells us that the value of δ for a particular hidden unit can be obtained by propagating the δ's backward from units higher up in the network,

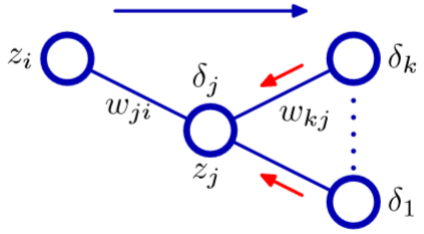


Fig. 2-10 Illustration of backpropagation

Illustration 2-10 of the calculation of δ_j for hidden unit j by backpropagation of the δ's from those units k to which unit j sends connections. The blue arrow denotes the direction of information flow during forwarding propagation, and the red arrows indicate the backward propagation of error information.

Thus backpropagation procedure can be applied as follow,

1. Apply an input vector x_n to the network and forward propagate through the network to find the activation of all the hidden and output units.
2. Evaluate the δ_k for all the output units
3. Backpropagate the δ's to obtain δ_j for each hidden unit in the network.
4. Evaluate the required derivatives.

2.2.4 Regularization in Machine Learning

Principle of regularization

If the number of hidden value (z_i) is too important it will have a problem of over-fitting. Note that M controls the number of parameters of the network, and so we might expect that in a maximum likelihood setting there will be an optimum value M that gives the best performance, corresponding to the optimum balance between under-fitting and over-fitting. An illustration of the phenomenon follows,

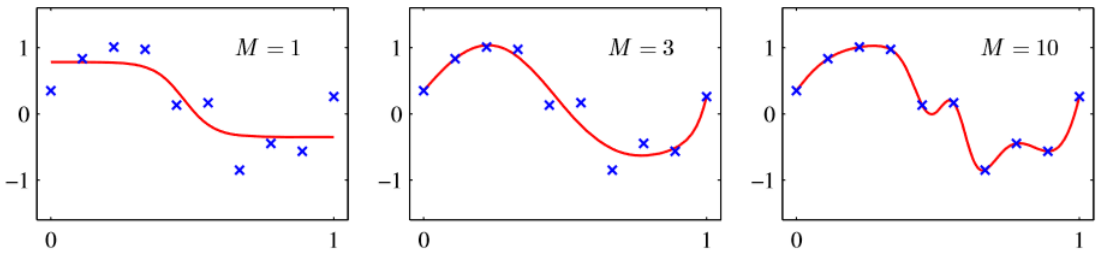


Fig. 2-11 Sinusoidal approximation with regularization term and different hidden units (M=1,3,10)

Figure 2-11 is an example of two-layer networks trained on 10 data points drawn from a noisy sinusoidal. The graphs show the result of fitting networks having $M = 1$, 3, and 10 hidden units, respectively, by minimizing a sum-of-squares error function using a scaled conjugate-gradient algorithm.

Invariances

In many applications of pattern recognition, the prediction has to give the same result. For instance, a building even reversed, is also a building: *rotation invariance*. Also position and size need not to influence result : respectively *translation invariance* and *rotation invariance*. A way to implement it is to feed the neural network algorithm the invariant that we want.

2.2.5 Convolutional Neural Network

Convolutional Neural Network (CNN) is a type of multi-layer neural networks. Like almost every other neural networks they are trained with a version of the back-propagation algorithm. They differ in the architecture and are often resumed as in figure 2-12,

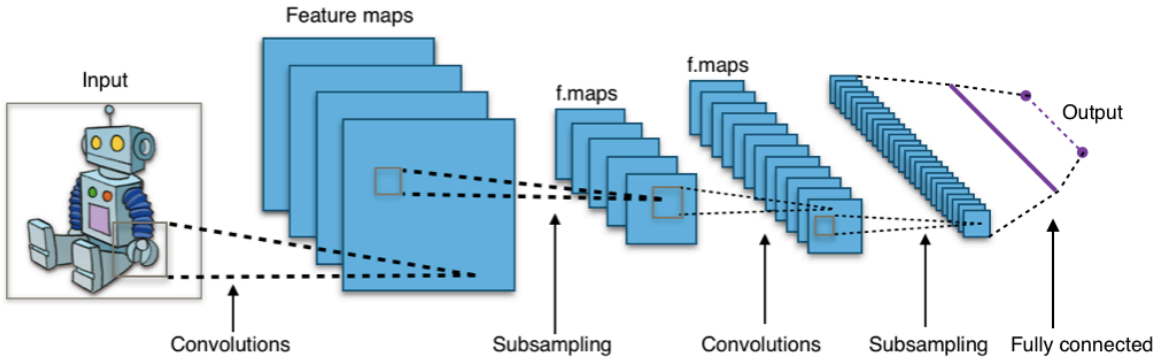


Fig. 2-12 Typical CNN architecture

Convolutional Neural Networks are designed to recognize visual patterns directly from pixel images with minimal pre-processing. They can recognize patterns with extreme variability (such as handwritten characters), and with robustness to distortions and simple geometric transformations.

Convolution Step

Convolution allows extracting features from an input image. It consists to apply the filter as explained in the section above.

2.3 Drone Model

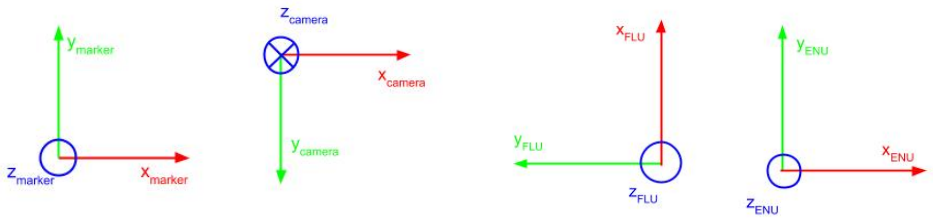
This section presents the tools used to control the drone. It begins with the choice of the appropriate coordinate system. Then the PID control theory is formulated. Finally, the optimal estimator used for the thesis is detailed.

2.3.1 Coordinate Systems

This paper mix different fields that have defined different reference frames. Each system has its frame that is presented succinctly:

- The tag reference frame is attached to the landing pad. It is direct oriented frame with *x-axis* pointing right, *y-axis* pointing front-ward and *z-axis* pointing up-ward. It is presented in figure 2-13a.
- The camera reference frame is centred into camera optical center. It has *z-axis* going out the camera, *x-axis* going right and *y-axis* going down-ward. It is shown in figure 2-13b.
- The drone body frame use FLU (Front-Left-Up) convention, where *x-axis* point front-ward, *y-axis* point the left and *z-axis* point up-ward. The reference frame is presented in figure 2-13c.
- The world reference frame use the ENU (East-North-Up) convention. Here *x-axis* point the East, *y-axis* point the North and *z-axis* point up-ward. The coordinate system is visible in figure 2-13d.

All the coordinate system used in the thesis are shown in the following figure,



(a) Marker reference frame

(b) Camera reference frame

(c) FLU reference frame

(d) ENU reference frame

Fig. 2-13 Convention of the reference frames

One can notice on 2-13 that the Marker coordinate system 2-13a and the world (ENU) coordinate system has the same definition. The thesis will mingle both having the ENU reference frame in the center, with the same orientation, of the Marker reference frame. Then the change

between reference frames is done using rotation matrix and translation vector. Assuming a general rotation where *Euler* angles α, β and γ are respectively rotations around axis z, y and x. Thus one can find,

$$R = R_z(\gamma)R_y(\beta)R_x(\alpha) = \begin{bmatrix} cos(\gamma) & -sin(\gamma) & 0 \\ sin(\gamma) & cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos(\beta) & 0 & sin(\beta) \\ 0 & 1 & 0 \\ -sin(\beta) & 0 & cos(\beta) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & -sin(\alpha) \\ 0 & sin(\alpha) & cos(\alpha) \end{bmatrix} \tag{2-31}$$

$$R = \begin{bmatrix} cos(\gamma)cos(\beta) & cos(\gamma)sin(\beta)sin(\alpha) - sin(\gamma)cos(\alpha) & cos(\gamma)sin(\beta)cos(\alpha) + sin(\gamma)sin(\alpha) \\ sin(\gamma)cos(\beta) & sin(\gamma)sin(\beta)sin(\alpha) + cos(\gamma)cos(\alpha) & sin(\gamma)sin(\beta)cos(\alpha) - cos(\gamma)sin(\alpha) \\ -sin(\beta) & cos(\beta)sin(\alpha) & cos(\beta)cos(\alpha) \end{bmatrix} \tag{2-32}$$

Always assuming the pinhole camera model introduced for camera calibration, previous equations for perspective correction (2-18 and 2-13) can now be reversed for pose estimation. The concept of $homography$ in computer vision is the generic name of transformation between two planes. More detailed explanations can be founded in [64] and [65].

2.3.2 PID Principle

The Proportional–Integral–Derivative (PID) controller is a control loop mechanism with feedback widely used in control systems. It calculates error $e(t)$ between the desired set-point $p^d(t)$ and the measured variable $p(t)$, and applies a proportional K_p, integral K_i and derivative K_d correction. For an output $u(t)$ the formulation is though following equation 2-33,

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \tag{2-33}$$

Where K_p, K_i and K_d are all non-negative coefficients.

In following sections, the time dependency will not be necessarily denoted. For instance, the reader will find u for $u(t)$.

2.3.3 Kalman Filter Principle

A KF is an optimal estimation algorithm. It allows estimating a variable from indirect measurement. In the presence of noisy measurements, parameters of interest such as location, speed, and direction are predicted. Common applications of Kalman filters include guidance and navigation systems, computer vision systems, and signal processing. The first applications

of the KF were for instance used for the Apollo project. Trajectories of the manned spacecraft to the moon and back were though estimated.

KF is used for different reasons:

- Only indirect measurements allow access to variables of interest.
- Various sensors provide the measurement but might be subject to noise.

The KF is a recursive filter that estimates the internal state of a linear dynamic system from a series of noisy measurements. It is used for the discrete system. To estimate the current state, only previous state measurement and current measurements are needed. No history of estimation and observation are required.

State of the filter is thus represented with two variables:

- $\widehat{x}_{k|k}$: estimate of $x(k)$ given measurement at time k
- $P_{k|k}$: error covariance matrix of the state estimate at time k

There are two distinct phases for KF: prediction and update.

Prediction

The predict phase uses the previous timestep to find an estimated state at the current timestep. Prediction state estimation is also known as a priori state. It doesn't include observation of the current state.

A priori state estimate

$$\widehat{x}_{k|k-1} = F_k \widehat{x}_{k-1|k-1} + B_k u_{k-1} \tag{2-34}$$

A priori error covariance

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \tag{2-35}$$

Where $\widehat{x}_{k|k}$ is a posteriori state estimate at time k, given observations at time k; u_k is the command entry; $P_{k|k-1}$ is a posteriori error covariance matrix; F_k is the state transition model from $k-1$ to k; Q_k is the covariance of the process noise; and B_k is the control-input model which link command u to state *x*.

Update

The update phase combines a priori prediction with current measurement to improve state estimate. This estimated combination is a posteriori state estimation.

Innovation

$$\tilde{y}_k = z_k - H_k \hat{x}_{k|k-1} \tag{2-36}$$

Innovation covariance

$$S_k = H_k P_{k|k-1} H_k^T + R_k \tag{2-37}$$

Optimal Kalman gain

$$K_k = P_{k|k-1} H_k^T S_k^{-1} \tag{2-38}$$

A posteriori state estimate

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k \tag{2-39}$$

A posteriori covariance estimate

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \tag{2-40}$$

Measurement post-fit residual

$$\tilde{y}_{k|k} = z_k - H_k \hat{x}_{k|k} \tag{2-41}$$

Where z_k is the observation or measurement at time k; R_k is the covariance of the observation noise; H_k is the observation model that maps true state x_k into observed state z_k; $P_{k|k}$ is a posterior estimate covariance matrix; and I is the identity matrix.

2.4 Mechanical Gear

2.4.1 Hybrid System

For the air to ground transition, the model hybrid dynamic system will be used. It mixes continuous dynamics and discrete events with interaction and changes between each. The drone is assumed as point-mass starting the fall with an initial speed v_0 and an initial height h. It bounces off the ground and dissipates its energy bounce after bounce. As inelastic collision, each impact makes a discrete change to the velocity where the falling equation switches to the damped harmonic oscillator.

Hypothesis

- Ground doesn't absorb energy.
- Magnus effect and buoyancy are neglected.

- Object is an undeformable solid.

To lighten the write, dot notations will be used,

$$a = \frac{dv}{dt} = \frac{d^2y}{dt^2} \Leftrightarrow a = \dot{v} = \ddot{y} \tag{2-42}$$

$$v = \frac{dy}{dt} \Leftrightarrow v = \dot{y} \tag{2-43}$$

Where a is acceleration in $m.s^{-2}$; v is velocity in $m.s^{-1}$; y is position in m; t is time in s.

Fall Object Equation A falling object obeys to projectile motion. According to Newton's second law,

$$\sum F = ma \tag{2-44}$$

Where m is mass of the object in kg.

$$F_G + F_D + F_T = ma = m\dot{v} = m\ddot{y} \tag{2-45}$$

Forces applied to the system are,

- *Gravity*: Force-directed downward and accelerate the object to the ground.

$$F_G = mg \tag{2-46}$$

- *Drag*: Viscous resistance that slows down a moving object. For relatively low speed, without turbulence, Reynolds number is less than 1. In this case, drag is linear and opposed to the movement.

$$F_D = -bv \tag{2-47}$$

  Where b is a constant that depends on fluid properties and object dimensions

- *Thrust*: Motor force is given by the propellers of the drone. Much complex aerodynamic turbulence is generated but not studied. Here is the result of the thrust vector.

$$F_T = N \tag{2-48}$$

  N is the thrust vector in N.

The system has though position x and velocity v as continuous states. The hybrid system aspect comes from impact with the ground.

Collision Equation At object landing, internal deformation makes it bounce. This elasticity is modeled by an oscillator that responds proportionally to the length of deformation. Besides, a frictional force makes decreases amplitude of oscillation with a viscous damping coefficient. It is proportional to velocity. According to Newton's second law, one can write,

$$F = -ky - c\dot{y} = m\ddot{y} \tag{2-49}$$

Where k is elastic coefficient; c is viscous damping coefficient.

This is known as the equation of a damped harmonic oscillator. It is often written as well,

$$\ddot{y} + 2\lambda\dot{y} + \omega_0^2 y = 0 \tag{2-50}$$

Where $\omega_0 = \sqrt{\frac{k}{m}}$ is the undamped angular frequency in $rad.s^{-}1$; λ is damping coefficient in $rad.s^{-}1$ It will be used when the drone will be in contact with the ground.

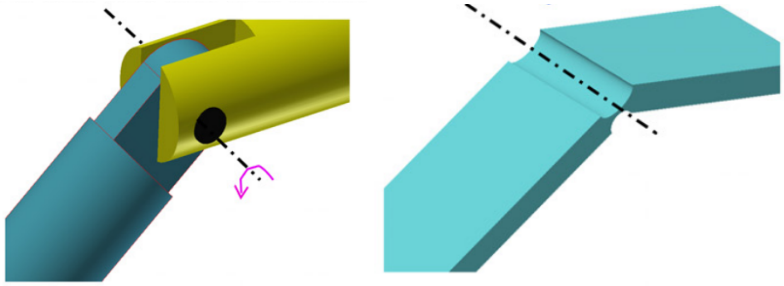
A bouncing object points out the Zeno phenomenon. This problem occurs when an infinite event like switches happens in a finite amount of time. This makes simulations to crash and highlights the fact that the model is inaccurate. In this case, the falling object should bounce a finite amount of time, not infinitely. Each time the object impacts the ground, the loss of energy makes the jump weaker and the object closer.

2.4.2 Mechanical Linkage

A table of mechanical linkage presents all theoretical connections in a mechanism. This also normalizes items to read kinematic draws.

2.4.3 Compliant Kinematic

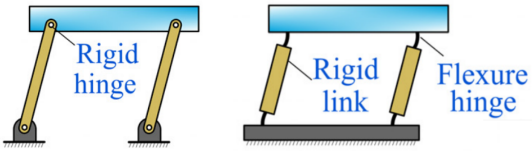
Flexure-based compliant materials are used for precision engineering and robotics. It takes advantage of no friction, no backlash, and minimal assembly requirement. For this study, the third points draw our attention. According to [58], the drawback of the compliant mechanism is the coupling of kinematic-mechanics with serial-parallel configuration, in comparison with a rigid-body mechanism. A first approach is to replace rigid pivot link (Fig. 2-14a) by a compliant hinge (2-14b).



(a) Rigid hinge (b) Flexure hinge

Fig. 2-14 Conceptual comparison between rigid and compliant hinges [58]

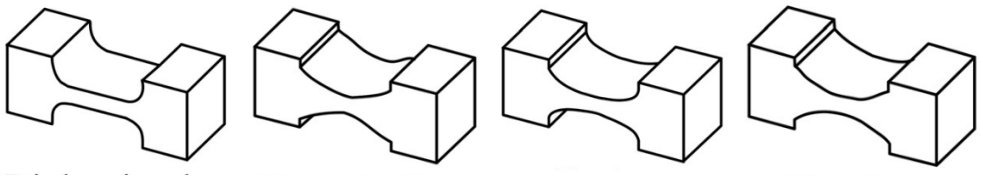
Then serial-parallel mechanism can be presented out of two following class,



(a) Rigid-body mechanism (b) Fully compliant mechanism

Fig. 2-15 Conceptual comparison between rigid and compliant mechanisms [58]

A rigid body mechanism (Fig. 2-15a) is transformed into a compliant mechanism (Fig. 2-15b) with the switch of rigid hinges to flexure hinges. For a compliant mechanism, the hinge is thus done with section reduction of the beam. Different shape exists.



(a) Right-circular (b) Hyperbolic (c) Elliptic (d) Circular

Fig. 2-16 Typical notch flexure hinges [58]

These shapes examples have substantially the same effects on the hinge.

2.4.4 Four-Bar Linkage Theory

Four-bar linkage is the simplest movable closed chain linkage [66]. Connected in a loop, it has four joints of one Degree Of Freedom (DOF). Planar four-bar linkage is an important mechanism for machines. Kinematics and dynamics of these mechanisms are widely studied

for a variety of movement in mechanical engineering like the ones presented in the following figure 2-17,

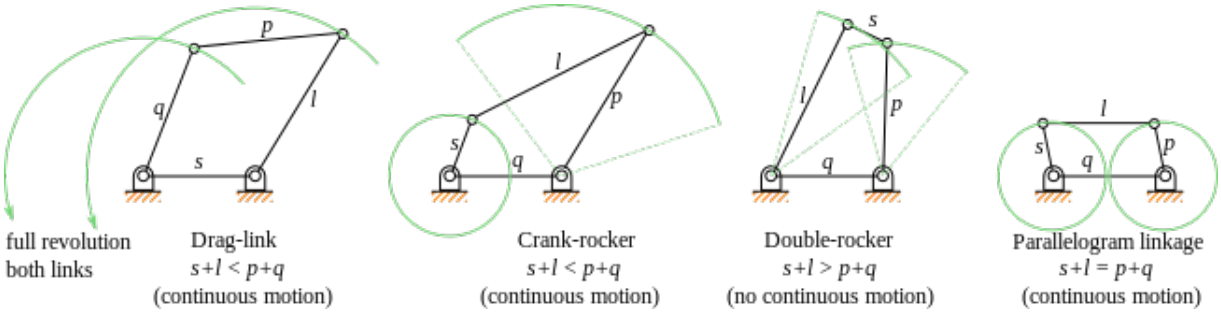


Fig. 2-17 Types of four-bar linkages. *s is the shortest link and l the longer link* [67]

Quadrilateral linkage configuration can though be classified into three types: *convex, concave* and *crossing*. This last one has two links that cross each other. The convex linkage has all its four angles less than $180°$, whereas the concave case has one angle greater than $180°$. The relationship between the lengths of the two diagonals of the quadrilateral can be found [68]. For convex and crossing configuration, the length of one diagonal increases if and only if the other diagonal decreases. While for nonconvex non-crossing linkages, one diagonal increases if and only if the other also increases.

2.4.5 Friction Theory

Friction is the force that resists to the relative motion of an object into a solid surface or a fluid. Friction has been widely studied from antiquity. It converts kinetic energy into heat so friction is a non-conservative force. This thesis takes interest in three types of friction:

- *Dry friction* for two solid surfaces in contact. It is composed of static friction for two non-moving solids and kinetic friction for two moving surfaces.
- *Fluid friction* is the force opposed to an object that moves into a viscous fluid [69].
- *Lubricated friction* mixes the two previous friction. A fluid called $lubricant$ is between two layers of solid surfaces [70].

The dry friction is governed by *Coulomb's Law of Friction* which is an approximate model,

$$F_f \leq \mu F_n \tag{2-51}$$

Where F_f is the friction force that is parallel to the surface and opposed to movement; μ is the coefficient of friction; F_n is the normal force applied on each surface.

The coefficient of friction μ is empirical and can be found on tables. It depends on the material and the asperities of the two surfaces.

The static friction coefficient μ_s is generally higher than the coefficient of kinetic friction. There is a maximum force F_{max} to reach, to begin the slide of one surface onto another,

$$F_{max} = \mu_s F_n \tag{2-52}$$

Any force applied to a static solid inferior to F_max let it at its place. Until F_max is exceed, kinetic friction starts. For a given dry friction, the lubrication reduces friction effects.

The fluid friction, on the other hand, is proportional to the speed. It has been presented as a drag in 2-47.

3 Drone Perception

In this chapter different landing pad areas are presented. It begins with the state of the art standard marker detection and follows with proposed neural network (NN) landing pad position estimator.

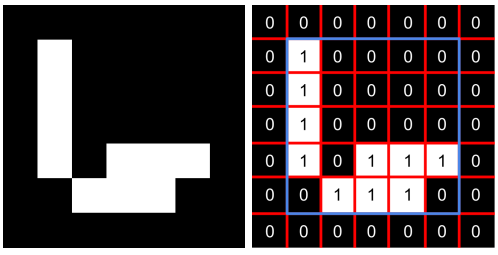
3.1 Standard Fiducial Marker

Presently, a wide number of different fiducial markers exist. The more commonly known markers are ARToolKit, ARTag, AprilTag, InterSense, or ArUco, however, the most common marker is the QR-Code. These markers have many applications for augmented reality due to the ease of usage. All of these markers consist of a black and white contrasted shape and an embedded simple binary code that allows tag recognition so that one tag can be differentiated from another.

3.1.1 Marker Detection

The method of finding a marker in an image is presented in this section and uses the OpenCV ArUco library [71]. First, the RGB (Red Green Blue) captured image is converted into a gray-scale image. An adaptive threshold enhances contours and searches for a continuous border. To detect and extract a square vertex, the ArUco library uses the algorithm presented in [72]. Only the most promising shapes that contain a readable tag are kept for the next stage.

An ArUco tag consists of a squared matrix containing black and white tiles. Each tag has a 5 by 5, circled in blue in 3-1b grid with a coded 10-bit number. The maximum number-coded is $2^{10} = 1024$. For example, the number 7 is coded as follows,



(a) ArUco tag 7 (b) Grids values

Fig. 3-1 ArUco 7 with corresponding grids value

The same principle of detection is applied for a fractal ArUco marker (Fig. 3-2) that has more a simplified shape.

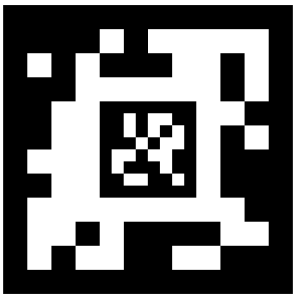


Fig. 3-2 Fractal marker with 0 for the outer square and 1 for the inner square

Each ArUco marker tag is thus identified with a value and a 2D position of its corners. The position of the four corners is then used to estimate the camera position relative to the marker.

3.1.2 Pose Estimation

After the detection of an ArUco marker respective to the imaging unit, its localization can resultantly be measured. Calibration can provide the intrinsic and extrinsic parameters of the camera, which are used in the following equation 2-13. It can be resolved as a Perspective-n-Point (PnP) problem presented in 2.1.4. OpenCV provides *solvePnP()* function, based on [73] that gives a rotation vector and translation vector of a squared marker of the side measurements d in a camera frame 2-13b. The four given coplanar corner coordinates P are defined in the following order,

$$P_0 = [-\frac{d}{2}, \frac{d}{2}, 0] \quad ; \quad P_1 = [\frac{d}{2}, \frac{d}{2}, 0] \quad ; \quad P_2 = [\frac{d}{2}, -\frac{d}{2}, 0] \quad ; \quad P_3 = [-\frac{d}{2}, -\frac{d}{2}, 0] \tag{3-1}$$

The result of the PnP problem is a rotation vector and a translation vector. The rotation vector is transformed into rotation matrix using *Rodrigues* transformation. The measured pose on the camera coordinate system can though be transformed into the world coordinate system. The rotation matrix and the translation vector are augmented in a homogeneous transformation matrix. This $3by4$ matrix that transforms the 3D point from the camera frame to the world position,

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{3-2}$$

Thus, pose estimation with image processing of marker localization can be summarized in this process diagram 3-3,

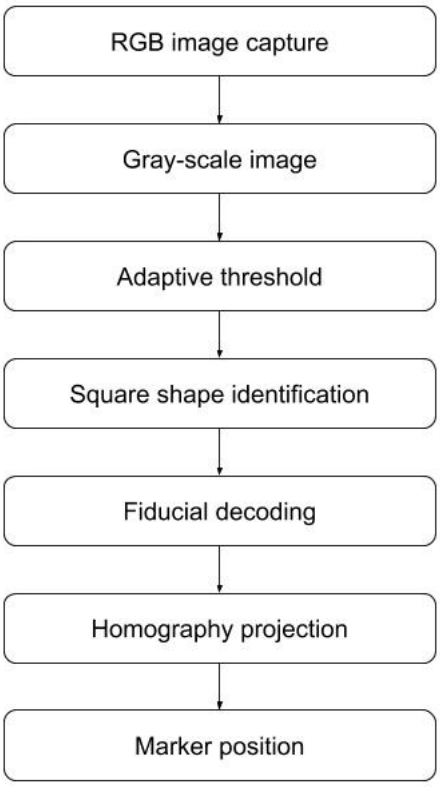


Fig. 3-3 Fiducial marker localization algorithm

After capturing the image, it is converted into a gray-scale image. The adaptive threshold allows working with the image having different lighting conditions in different areas. A threshold is applied to the pixel's neighbors of small areas.

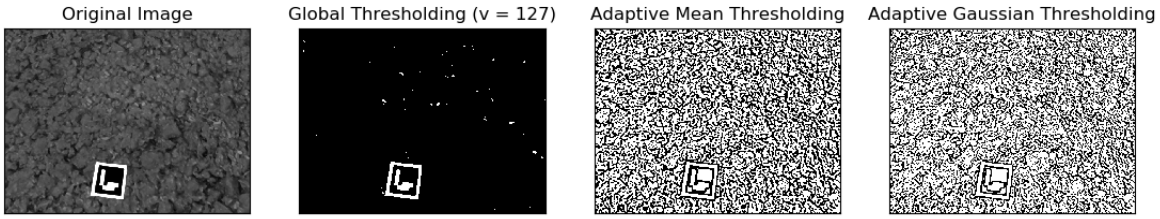
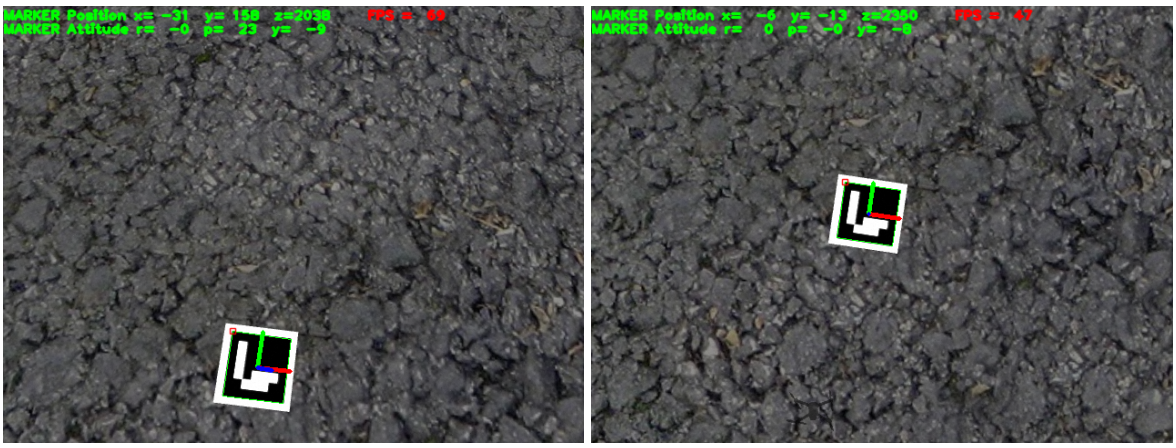


Fig. 3-4 Comparison of global thresholding and adaptive thresholding for a landing pad in the simulation

In the case of simulation, because there is no variation of the illumination, and as seen in figure 3-4 the global threshold could have been sufficient. So even if this takes more time than a simple threshold, it is more robust to noise. Then the square shape detection is done by segmentation and contour detection. The decoding of the fiducial allows here to differentiate a random square from the ArUco marker. The homography projection permit finally to express the 2D measured position into the world coordinate system. This gives though the marker positions of figure 3-5,



(a) ArUco marker processed position in movement (b) ArUco marker processed position in static

Fig. 3-5 Processed ArUco position with visual reference frame drawing

The ArUco pose estimation is relatively fast and accurate, but for a fast approach, the calculation needs to reach real-time. Recent advancements using virtual neural networks for machine vision showed an improvement in data processing velocity and estimation accuracy.

3.2 Deep Learning Localization

During World War II, an American behaviorist B.F. Skinner's attempt to develop a pigeon-controlled guided-missile [74]. the *Project Pigeon* was later renamed *Project Orcon* for *"organic control"*. They were trained to tap on a specific screen with their beak to adjust rocket trajectory. Even if animal intuition for pose estimation was a novel usage method, eccentricity, and impracticality to use real birds explained the end of this research. The recent improvement in computer science with machine learning provides a new point of view surrounding this work using a virtual NN instead of living animals.

Classic image processing uses a lot of computing resources and is struggling to reach real-time. The proposed method will use state of the art Convolutional Neural Network (CNN) to compute regression and extract marker localization.

For supervised learning NN, a data set is needed. This will be used to train the NN to predict drone positions from a picture. An image of the landing pad will be associated with the corresponding homogeneous transformation matrix. The dataset will be composed of localization coordinates with the image of the associated landing pad.

3.2.1 Dataset Generation

Drone localization has a meter accuracy with GPS. Datasets will need to be more accurate to predict a better measurement estimation from the algorithm. It is possible to localize an indoor drone with motion capture technology. Present drawbacks that remain with the millimetric measurement tool are that they are expensive and difficult to use outside of a lab. For these two reasons, a cheap technique is often used. Moreover, the use of a simple camera, robust to crashes, is theoretically sufficient thanks to the amount of data provided. The recent NN techniques give low computer consumption for impressive visual processing. Thus, pose estimation and awareness of the surrounding world can find a solution using cheap sensors and a well trained NN.

A dataset is needed for supervised learning and the thesis proposes to use 3D software. In the case of parameter changes during the study such as pattern, size, and positions, the virtual generation provides more flexibility. A first try to generate dataset by hand is increasingly too time-consuming and not reliable, that's why the proposed method uses an automatic generator.

The following method presents a generation of the dataset needed to train the NN. This combined an image and an associated file with 6 Degree Of Freedom (DOF) localization.

Landing Pad

A first study using ArUco provided a successful result. The dataset will be inspired by these black and white patterns. It is relatively easy to extract the object from the picture with grey level filters and threshold enhancement.

Three points are the minimum number necessary to localize a 3D plane in the space. A redundancy of points with more than three increases precision of estimation. The pattern needs to resemble a triangle. An ”L” shape is the main image of the landing pad. The asymmetry of the shape is created with a catted edge. The center of the area is specified with a circle. The redundancy of points is designed with the smaller symmetric ”L” to ensure reliability.

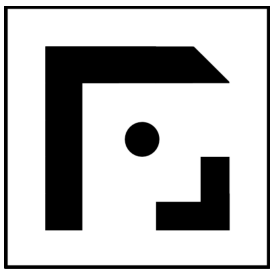


Fig. 3-6 Landing pad pattern

The shape of figure 3-6 is thus a proposed picture for the landing pad. Then the range value of the possible drone 3D position needs to be set.

Positions

The range of values is determined by moving the camera around. When the landing pad just has the view field, an extreme position is there. The range of altitude $z \in [0.3; 50]$. Then,

- for $z = 0.3$, $x \in [-0.6, 0.6]$ and $y \in [-0.55, 0.55]$;
- for $z = 50$, $x \in [-18.4, 18.4]$ and $y \in [-10.4, 10.4]$.

For a given z altitude, a boundary linear equation can be applied as follows,

$$x = m_x z + p_x \tag{3-3}$$

and

$$y = m_y z + p_y \tag{3-4}$$

where

$$m_x = \frac{x_{max} - x_{min}}{z_{max} - z_{min}} \quad ; \quad p_x = x_{min} - m_x z_{min} \tag{3-5}$$

and

$$m_y = \frac{y_{max} - y_{min}}{z_{max} - z_{min}} \quad ; \quad p_y = y_{min} - m_y z_{min} \tag{3-6}$$

from which positive boundaries equations with 10^{-4} accuracy are

$$x = 0.3581z + 0.4926 \tag{3-7}$$

and

$$y = 0.1982z + 0.4905 \tag{3-8}$$

Negative boundary equations are the opposite of the previous ones and permit a range of values from negative to positive ones.

The following figure in blue shows the limit area where the camera position is chosen.

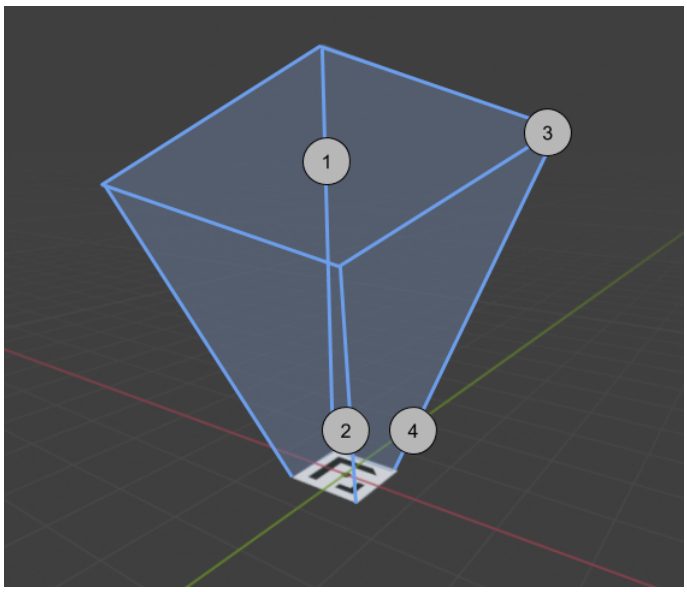
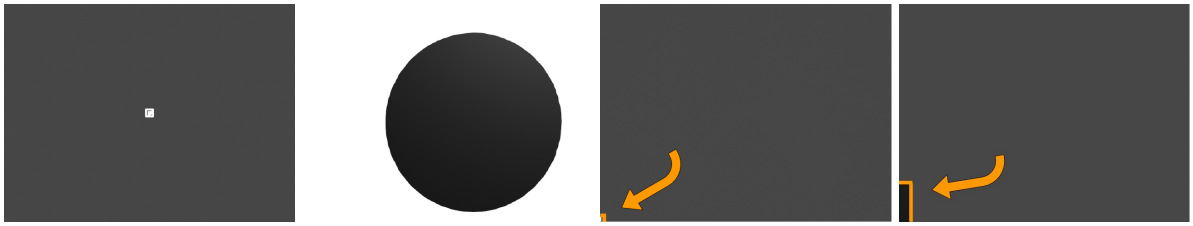


Fig. 3-7 Field of camera position, with camera point of view for interesting areas

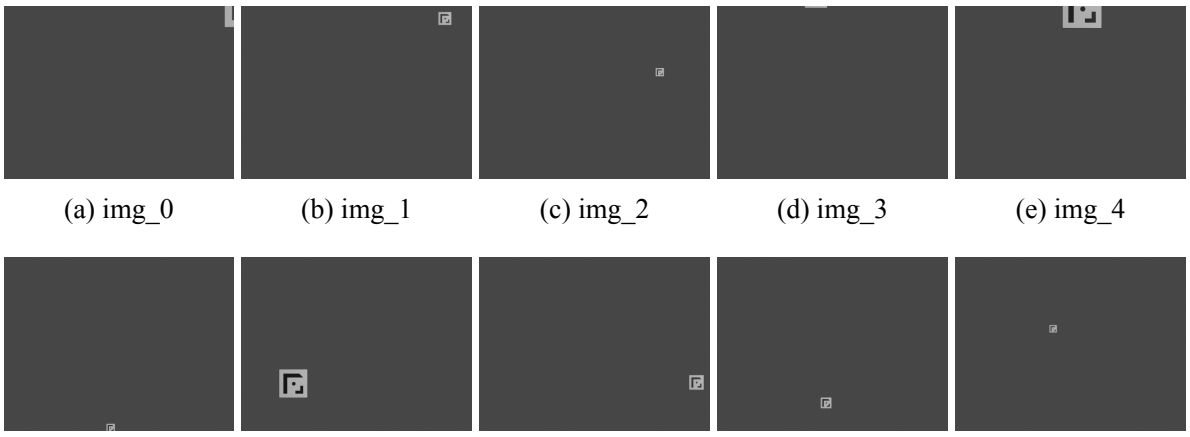
The scale of the blue area compared to the landing pad is of course not respected on the previous figure 3-7. The point of view of the camera on these extreme positions is showed below,



(a) Image (1) (b) Image (2) (c) Image (3) (d) Image (4)

Fig. 3-8 Side images of the dataset

The contours of the landing pad are enhanced in orange at the end of the orange arrow for better visibility. The extreme images like 3-8c, 3-8d should allow the NN to estimate the landing area position when the marker enters the field of view. Then for the entire dataset, the position of the camera is distributed **homogeneously and randomly** in this area. Follows images on figure 3-9 of the positions that the drone could have,



(a) img_0 (b) img_1 (c) img_2 (d) img_3 (e) img_4

(f) img_5 (g) img_6 (h) img_7 (i) img_8 (j) img_9

Fig. 3-9 Images positions randomly generated

To simulate drone pose relative to the landing pad, rotation needs to be added to the drone position.

Rotation

An analog data generator of the position one can be used. It allows the NN to estimate rotation between the marker and the drone. Rotation between the drone and the ground is not essential. Most of the autonomous drone landing algorithms only calculate the center of the landing pad, because it is assumed to be completely flat. The use of the rotation matrix will be explained in the Guidance and Control section.

Some extreme rotation will not be possible during the use of the drone. For instance, the pad will be always visible from the front, that's why the following range of rotations is proposed. The notations are according to camera reference frame 2-13b,

- for the $pitch$: $r_x \in [-\frac{\pi}{4}, \frac{\pi}{4}]$;
- for the $roll$: $r_y \in [-\frac{\pi}{4}, \frac{\pi}{4}]$;
- for the yaw: $r_z \in [-\pi, \pi]$.

To ease the generation of the dataset and to be sure that the landing pad is always on the field of view of the camera, instead of rotating the camera, the marker has the rotation. To be explained on the camera reference frame, from Blender, the vector rotation simply needs to be explained as its opposite.

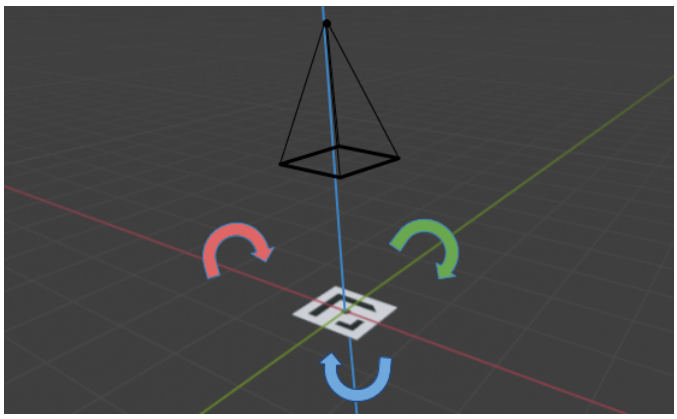
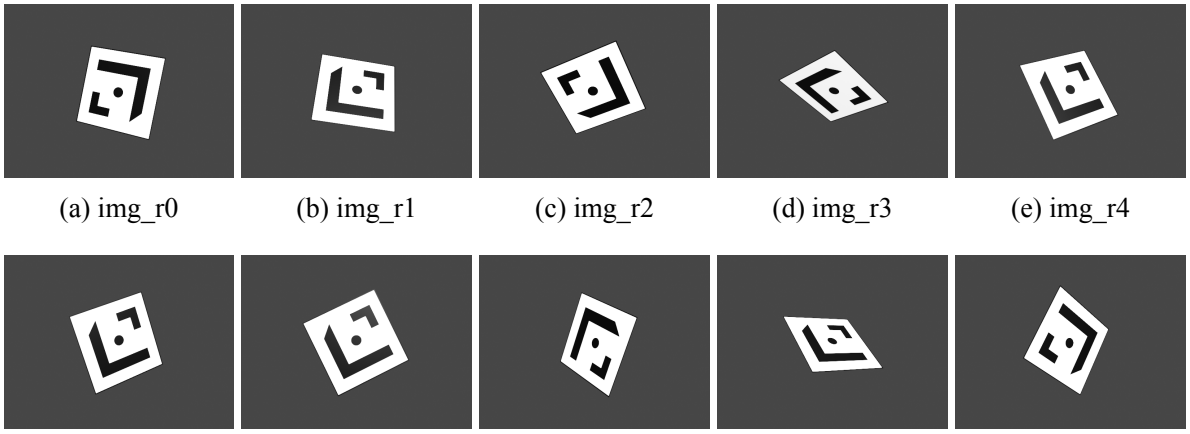


Fig. 3-10 Rotation enabled of the landing pad, with camera looking the area

The camera is represented with the black polygon on 3-10. In red is the rotation around *x-axis*, in $green$ rotation around *y-axis*, and in $blue$ rotation around *z-axis*. Then the rotations generated by the algorithm follows on ten images examples. The camera is fixed at the $(0, 0, 4)$ position and uniform rotations are set,



(a) img_r0 (b) img_r1 (c) img_r2 (d) img_r3 (e) img_r4

(f) img_r5 (g) img_r6 (h) img_r7 (i) img_r8 (j) img_r9

Fig. 3-11 Images rotations randomly generated

After having explained all the views of the drone with positions and rotations, the presentation of the useful data follows.

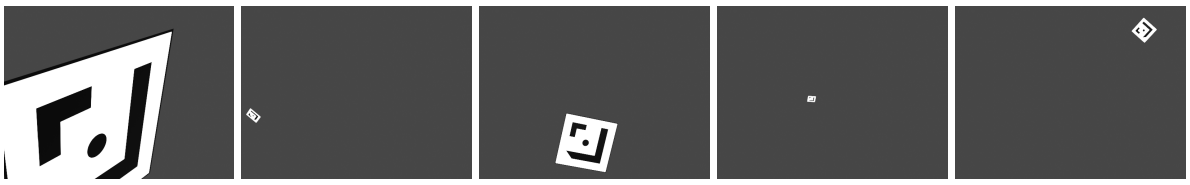
Dataset Shape

After all, the dataset generator can provide position and rotation of the marker, relatively to the camera. For the exercise presented in Chapter 4, the dataset proposed is composed of images and drone positions relative to the marker frame. Using opposite coordinates, the given position is thus the target point of the drone.

For a first try, as shown so far, the background is set completely uniform. This type of image can easily be reached with visual processing. as mentioned before. The pose data are finally saved in a csv file for the proposed implementation. Thus, the drone position is given in the FLU reference frame 2-13c of the drone but projected on the marker reference frame 2-13a. So the dataset csv file has the following information,

- Integer number that associate one image to the drone pose;
- x distance in meter of the marker relative to the drone;
- y distance in meter of the marker relative to the drone;
- Altitude z in meter of the drone relative to the marker;
- Yaw r_z of the drone relative to the marker.

In short, the position is given by the dataset in the horizontal direction that will be directly translated with the control part into the velocity vector. Furthermore, an extract of the images from the data set is shown below,



(a) image_0 (b) image_1 (c) image_2 (d) image_3 (e) image_4

(f) image_5 (g) image_6 (h) image_7 (i) image_8 (j) image_9

Fig. 3-12 Images pose, composed of position and rotation randomly generated

These images are a screenshot from the virtual camera saved in PNG format. The virtual dataset generated with a simple background is then used to train the NN to localize the drone in space.

3.2.2 Neural Network

The use of an NN instead of a standard marker detection bypasses different steps in the localization algorithm.

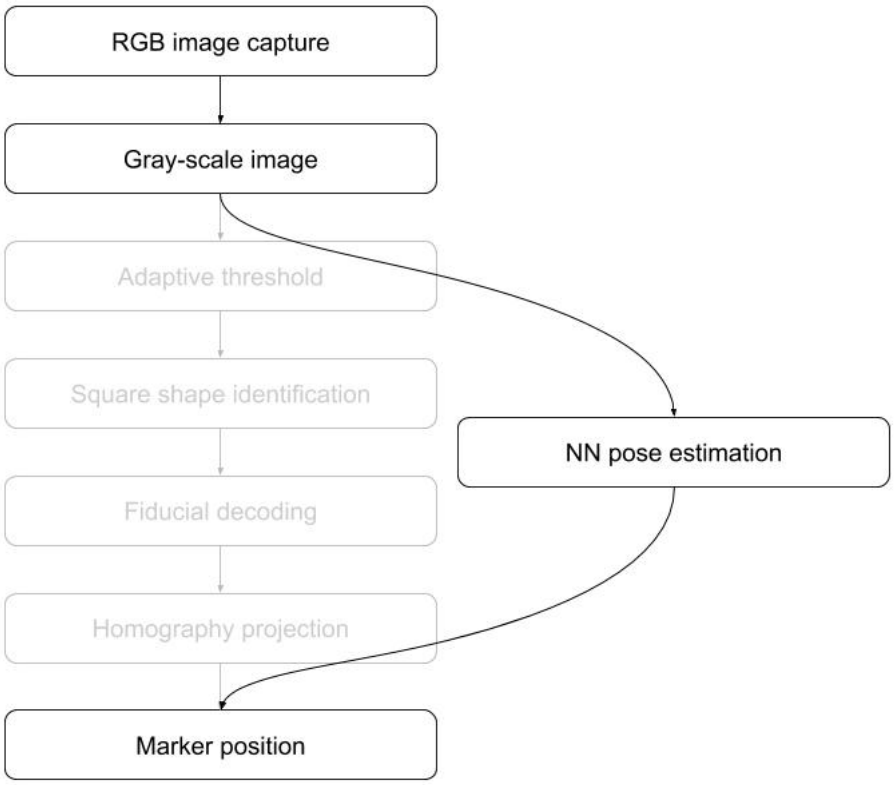


Fig. 3-13 NN algorithm for landing pad detection compared to standard marker detection

The learning approach aims on diagram 3-13 is to reactively estimate the landing pad position using the drone camera facing downward. The pose is later converted into a drone control command that enables the drone to descend with the expected strategy.

Convolutional Neural Network

The Convolutional Neural Network (CNN) architecture has two branches to predict the horizontal position of the drone. Since some tests have presented difficulties when calculating the drone altitude in the NN, this is calculated separately with a second NN containing the same architecture but trained with fewer epochs. The prediction of the linear value instead of the standard classification is called regression. It is done with a linear activation neuron at the end of the NN.

The NN proposed architecture has been found to provide sufficient results. A different number of layers and neurons by layer have been tried. This is mainly inspired by [30].

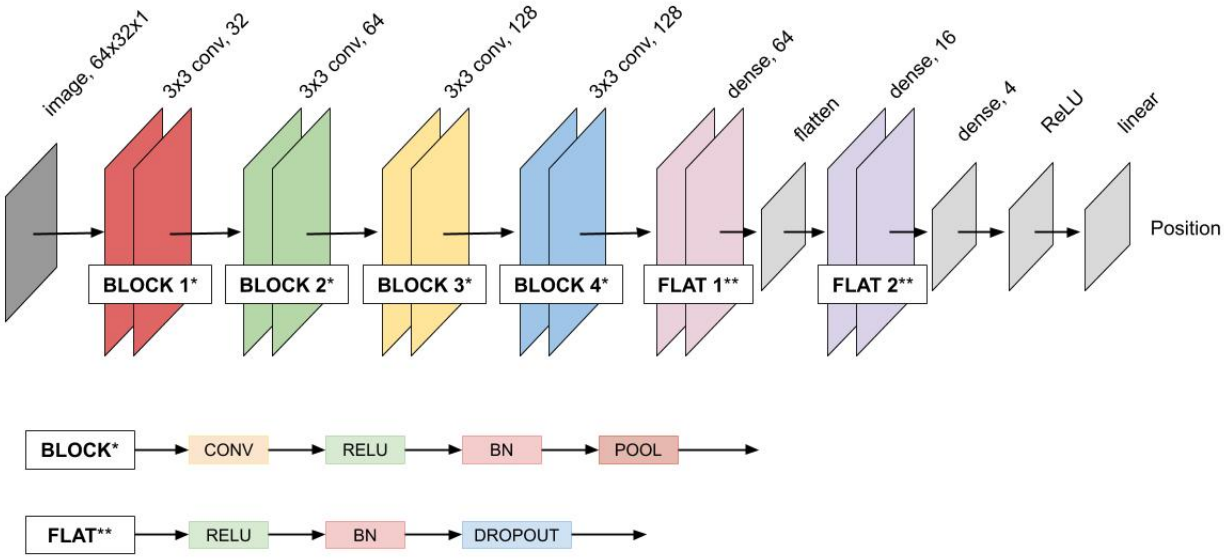


Fig. 3-14 CNN proposed architecture

The same NN architecture proposed in figure 3-14 is used to predict x and y position and runs in parallel. The split is done at the entry of the NN, where a unique image feeds the network. Then the horizontal drone position in the marker coordinate system is given as an output.

Training

Before being used, the NN needs to be trained, thus datas are preprocessed to suit better learning. Datas are first normalized between $[0, 1]$ to improve result of training within the layers. The learning progress of the NN can be observed with the average loss over the training data. The loss is calculated as the *mean absolute percentage error equation* over the data.

$$M = \frac{1}{n}\sum_{t=1}^{n}|\frac{T_t - M_t}{T_t}|100 \tag{3-9}$$

Where n is the number of data, T_t is true value and M_t is the measured value.

The analysis of the training focuses on the horizontal (x, y) position, which will be used for drone control. Without specifically knowing drone attitude the projected position 4-4 on the marker reference frame is estimated by the NN.

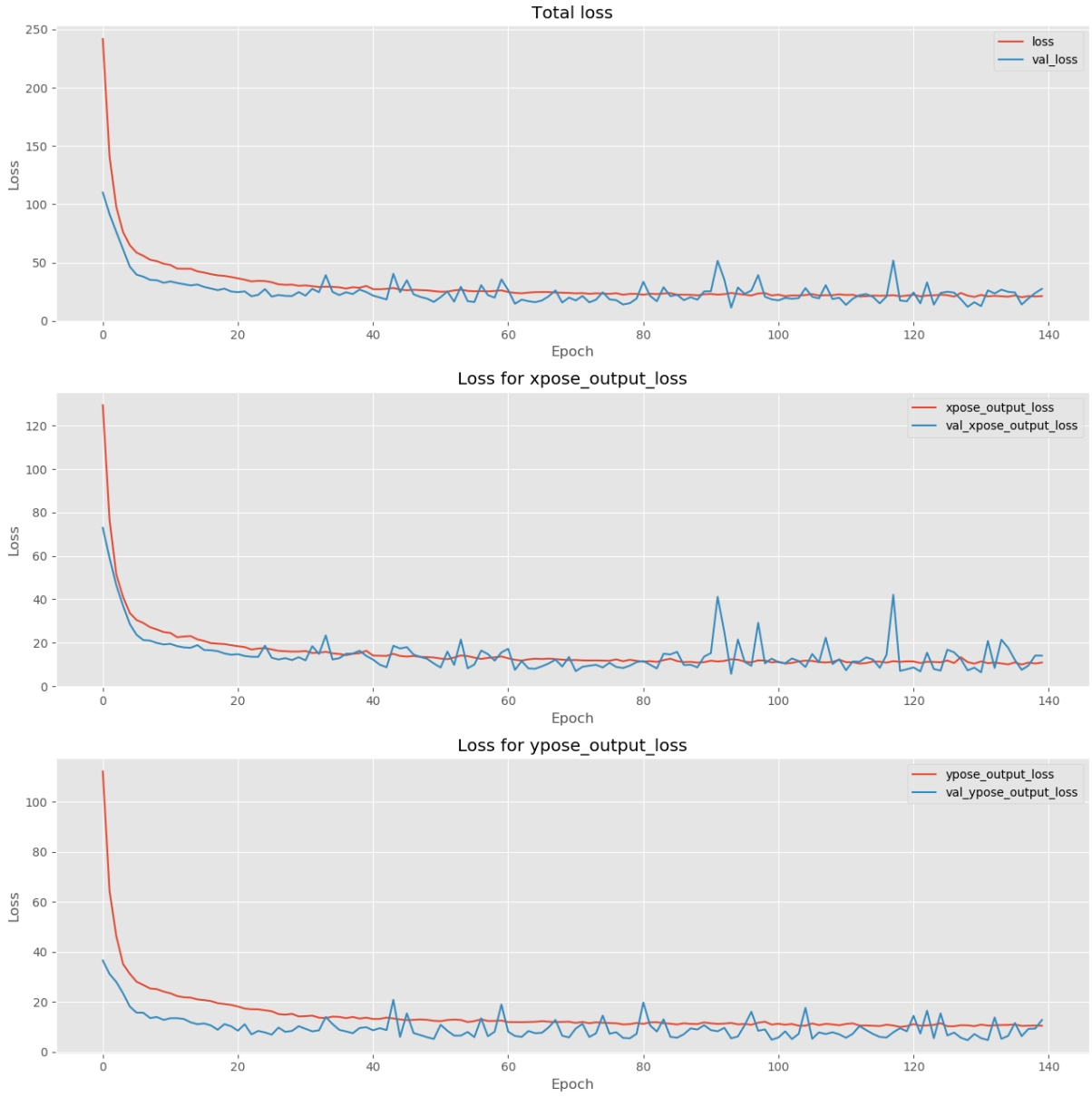


Fig. 3-15 History of the loss of x and y during training

The learning stage has been performed on 140 epochs with a batch size of 16, and with 1980 train images and 100 test images. The number of batches facilities the independent process in parallel with the training of the model. The batch generally approximates the distribution of the input data better than a single input. The larger the batch size, the better the approximation. However, this depends on the memory of the computer. Therefore, the proposed 16 batch size was the largest size supported by the computer used.

The $loss$ and val_loss plotted on figure 3-15 differs where the $loss$ is calculated over all of the training data, whereas val_loss is the same cost function measured on the training dataset using a test dataset. Over-fitting is observable when the val_loss goes over the $loss$. For x po-

sition measurement, learning is at the limit of over-fitting while the y position prediction could still learn could a little even at the end of the training.

The position range is the one presented in the equations 3-7 and 3-8. The learning of the NN is observable with the diminution of the loss between the predicted data and the ones which are used for the training. The loss is calculated using the mean absolute percentage error. An over-fitting has been observed for 190 epochs on x position prediction, that is why 140 training epochs are proposed.

Prediction

Using the gradient-weighted activation mapping also called Gradient-weighted Class Activation Mapping [75] (Grad-CAM), it is possible to observe an image where the CNN is looking. The $TensorFlow$ implementation of this method finds the final convolutional layer of the NN and examine the gradient information. To understand data, a $heatmap$ using $viridis$ standard scale is displayed.

An activated part of the image corresponds to $yellow$ pixel of the activation map, while a non-active part of the image corresponds to a $deepblue$ pixel.

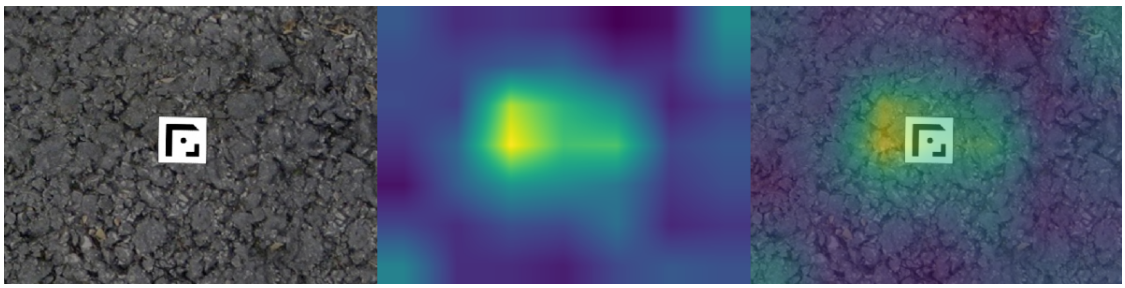


Fig. 3-16 Centered landing pad with Grad-CAM

This activation map visible on figure 3-16 shows that the main information of the landing pad for the proposed NN is the large "L" shape of the marker design. Most of the activated pixels are on the position of the landing pad.

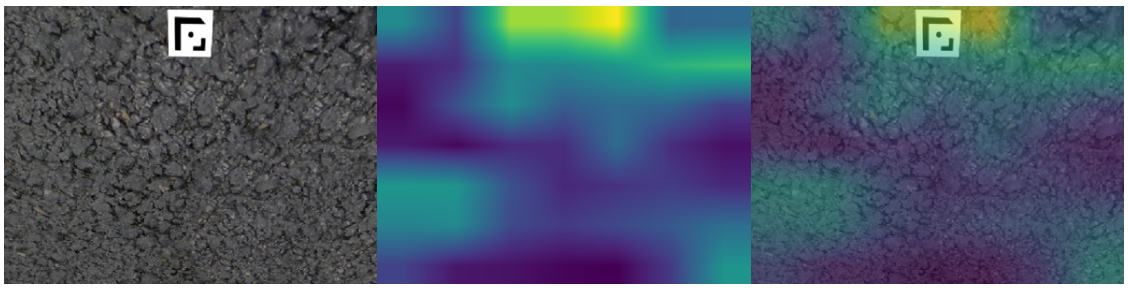


Fig. 3-17 On the edge landing pad with Grad-CAM

Even with a not centered on Fig. 3-17 landing pad position, the activation pixels follow the marker.

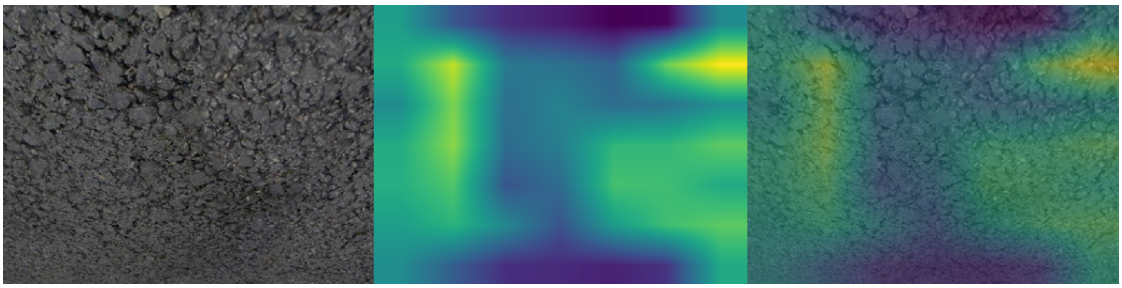


Fig. 3-18 Only background with Grad-CAM

Then the use of the background as an input for prediction shows on figure 3-18 an extended activation area. This kind of image has not been used for the training and the NN activated its neurons especially on the right and left edges of the image, at the expense of the lower and outer borders. The effect of having a simple background for the training and a more complex one for the implementation seems to have a small effect on the landing pad pose estimation.

3.3 Comparison results

The accuracy is estimated by calculating the percentage error measurement between the real value and the measured one. Even if the loss is usually a measurement of data quality, the definition given by the equation 3-9 is the same as the error but is given in the range $[0, 1]$ instead of a percentage,

$$X_{error} = \frac{|experimental\ \ value\ \ -\ \ theoretical\ \ value|}{|theoretical\ \ value|} \tag{3-10}$$

Considering the measurement is following a Gaussian distribution, the reader will pay attention to the accuracy corresponding to the mean and precision corresponding to the standard deviation.

For the experimentation, random marker positions are generated with the previously proposed generator. The marker shape is changed to suit the detection algorithm, but the marker size and the position range is maintained. Before the experiment, the generator provided some images with hidden markers, which would not be recognized with ArUco detection.

Graphs are plotted with the true landing pad position sorted for more visibility. This does not reflect the real random measurement, but makes the result more comprehensive to the reader.

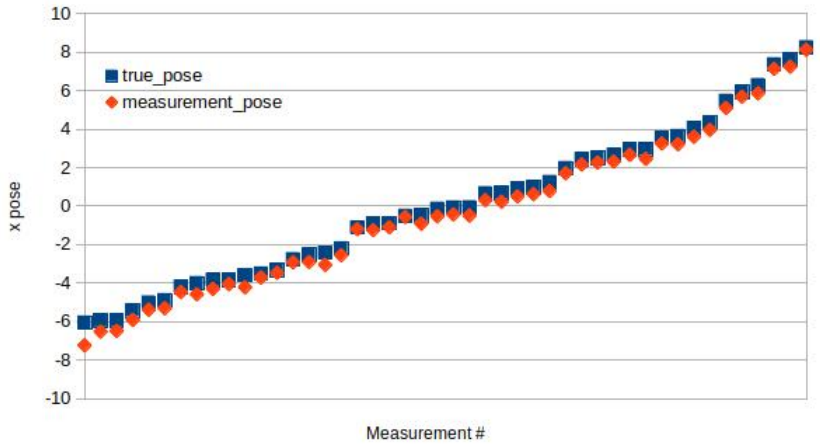


Fig. 3-19 x position ArUco measurement

Position x on camera coordinate system looks on figure 3-19 a little bit underestimate with measured pose under the true marker pose. A small side effect seems observable around $-8m$ where data have a meter accuracy. On the measurement, the true range is $[-6.056, 8.251]$.

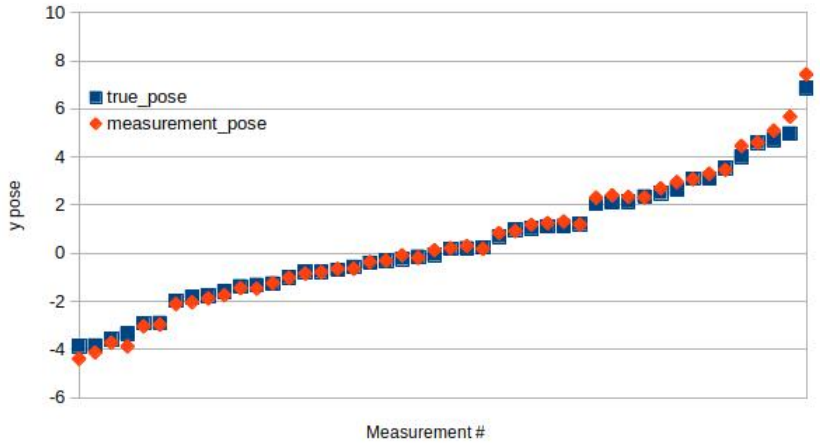


Fig. 3-20 y position ArUco measurement

On y axis the measured positions visible on Fig. 3-20 are more accurate near the pose 0. The side effect shows an overestimation of the real value, where the measured value has a

bigger absolute amplitude than the true pose. The measurement interval is not symmetrical and is $[-3.864, 6.861]$.

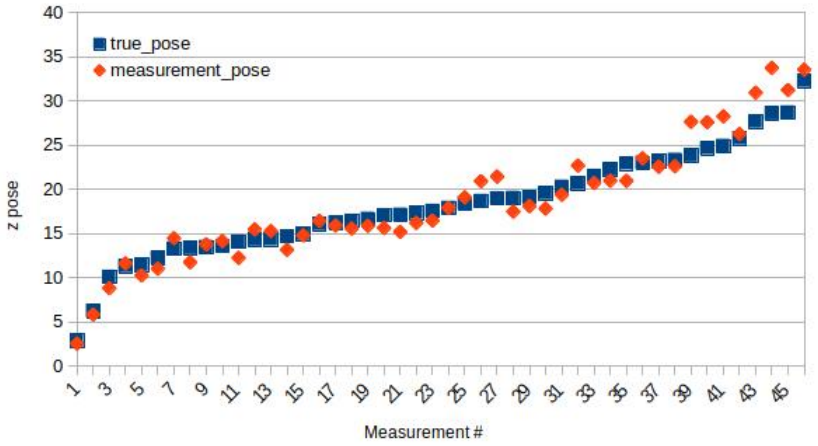


Fig. 3-21 z position ArUco measurement

Then the z position measurement of figure 3-21 is more dispersed on both sides of the measurement. Fewer values under $10m$ altitude has been measured and the range is $[2.896, 32.275]$. The ArUco measurement is more accurate in a horizontal position than on vertical values. Altitude measurement has more noise for higher altitude.

The fractal ArUco marker has scaled fiducial markers inside its shape 1-2d. It is supposed to correct the relatively small range of ArUco measurement and to increase the interval.

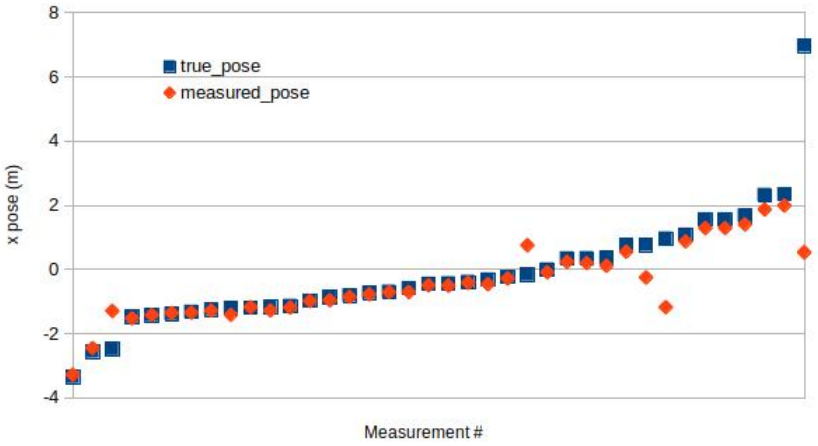


Fig. 3-22 x position fractal measurement

The measurement in Fig. 3-22 looks often accurate and has fewer mistakes on negative measurement than positive measurements. On the right side of the graph, more noisy measurement has been done. The range is $[-3.356, 6.973]$, but can be reduced if the false upper value

is removed from the result.

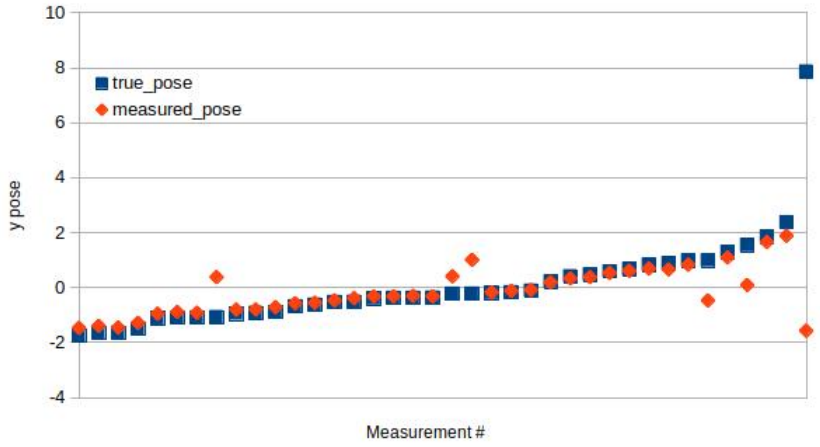


Fig. 3-23 y position fractal measurement

On the y axis of the camera, the same result than on y is observable on the graph of the figure 3-23, where there is more noise for positive measurement. Thus the visible range is $[-1.743, 7.857]$ but the upper false-positive value could be removed.

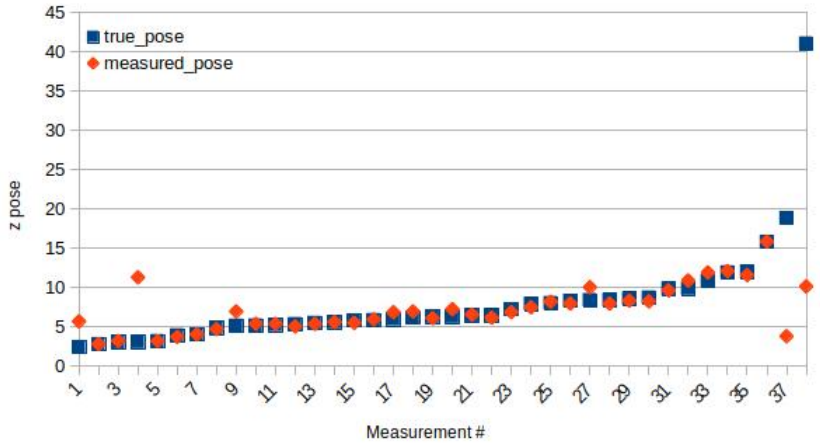


Fig. 3-24 z position fractal measurement

For the vertical measurement observable on the figure 3-24 above, apart from three values on low altitude, the near ground measurement is more accurate than the farthest. The measurement range on z axis is then $[2.424, 41.060]$. With the same problem notice previously, the upper value can be divided per two if the false-positive measurement is removed. On another hand, the proposed NN should recognize marker position in a wider range.

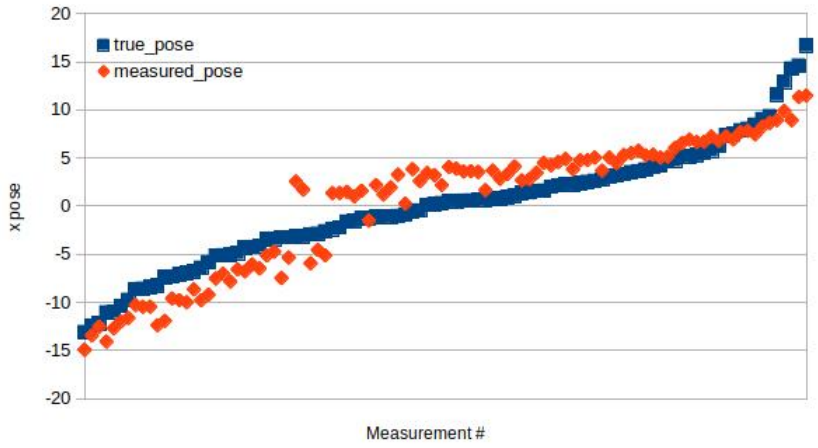


Fig. 3-25 x position NN measurement

On $x-axis$ pose estimation, the measurements near to the center of the frame shows in Fig. 3-25 a lack of accuracy and a jump from the positive measures to the negative measures. Absolute measurement amplitude looks larger than the real marker pose, except for marker position upper to $7m$.

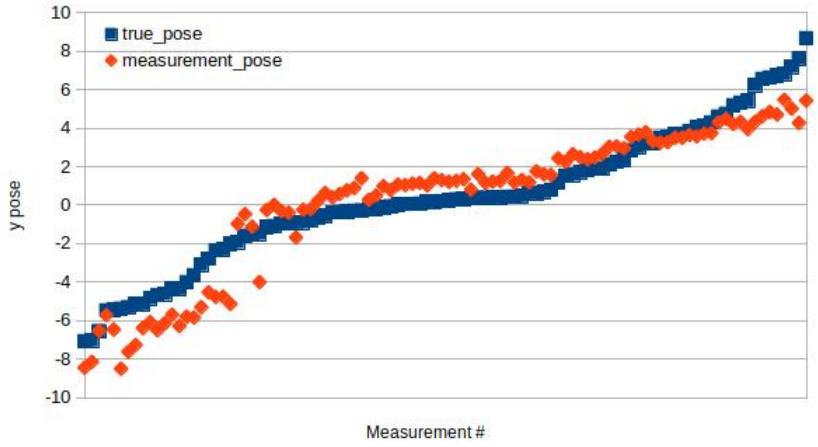


Fig. 3-26 y position NN measurement

Almost the same remarks on $y-axis$ could be done than on the $x-axis$, because global measurement curve has the same shape on the figure 3-26. Negative values are overestimated as positive values until the $4m$ where the position is underestimated.

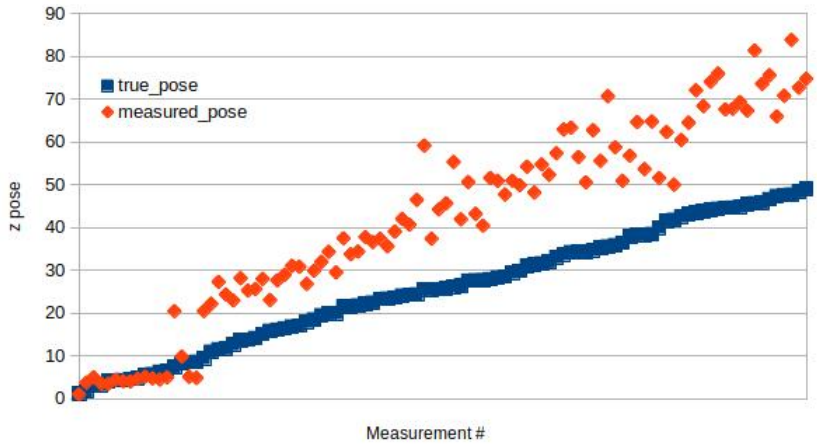


Fig. 3-27 z position NN measurement

Then for vertical measurement, pose estimation looks very accurate on figure 3-27 for near ground altitude. Under $5m$ the estimated position follows almost perfectly the true position. However, there is a bias in the altitude measurement over 10 m where the estimation is 10 meters higher than the true position. Moreover, the global accuracy decreases with an increased height because the measured points are more scattered.

In the different measurements above, the range of the horizontal vision is more important for traditional ArUco detection markers than for fractal markers. Without the false-positive measurement on the $z - axis$, the ArUco marker measurement altitude is $10m$ higher than the fractal marker. But the fractal measurement can estimate the near ground camera position 40 centimeters closer, until 2.4 meters. There are side effects for high altitude measurement of all detectors and for horizontal marker measurement of the NN. The position estimation is less accurate and the result is incorrect.

The table 3-1 follows a comparison of estimated position errors with the three different techniques.

Tab. 3-1 Experimental table of the landing accuracy of three computer vision algorithms

Algorithm	$\mathbf{x}_{error}$ (μ, σ^2)	$\mathbf{y}_{error}$ (μ, σ^2)	$\mathbf{z}_{error}$ (μ, σ^2)	**Number of data**
ArUco	(0.470, 1.225)	(0.174, 0.425)	(0.075, 0.043)	46
Fractal	(0.615, 1.587)	(0.491, 1.012)	(0.200, 0.494)	38
Proposed NN	(0.143, 0.091)	(0.125, 0.112)	(0.627, 0.295)	100

The proposed NN is the most accurate position estimator of horizontal measures. Even though the ArUco estimation on y has the same error, the NN has a smaller standard deviation

and a more precise result. The estimation posed by ArUco is measured almost three times more accurately on the x axis than on the y axis. However, the vertical measurement is almost 10 times more accurate for ArUco marker than for the NN, and almost three times more accurate for ArUco marker than for fractal pose estimation.

In the 200 images used for ArUco and the fractal marker, less than a quarter of the images were usable. This means that the field of view of these traditional monocular pose estimators can be used in a narrower field of view. In this experiment, the global position accuracy provides an overview of the expected results on drone position measurement. Although, this does not provide accuracy surrounding drone landing as this depends on the way it is implemented.

4 Guidance and Control

The all Guidance Navigation Control (GNC) is setup with the following tools:

- Ubuntu 18.04
- Gazebo 9
- ROS melodic
- PX4 Master *(on 02/28/2020)*
- OpenCV 4.2.0
- QGroundControl

These software are presented in the section bellow.

4.1 Drone Simulator

At first, the idea was to use *AirSim*, a powerful simulator for machine vision, which was based on the Unreal Engine game engine. It is well acclaimed for game conception because is allows real-time realistic rendering. Texture in the Unreal Engine so closely mirrors the reality that it now uses as background in movies, more specifically when used by the Industrial Light and Magic company for The Mandalorian movie. The main problem surrounding its usage is the excessive amount of computing power needed, and for this reason, an alternative simulator was chosen. The *Gazebo* simulator was used for this study as it consumes relatively low power and is a widely used simulator for autonomous vision-based drones. Generally, the use of a simulator is a safe and efficient way to confirm system functionality before real-world tests.

The environment is set up with a ground texture resembling a stone-paved floor visible on figure 4-1. Two objects are added to a standard drone simulation: a fixed camera facing done and an ArUco marker. As explained below, the camera emits a ROS topic that can be viewed on the simulator.

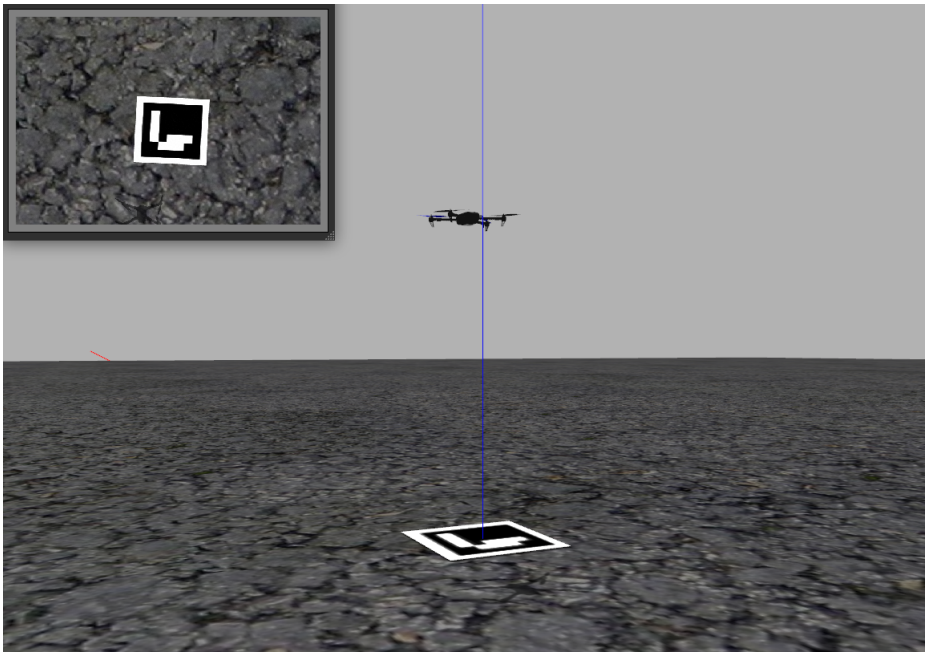


Fig. 4-1 Gazebo setup for simulation with embed camera view

With the presented configuration on Fig. 4-2, at the time when the thesis was being written, the UDP communication video was not possible. The only functional way to receive video was to use ROS (Robot Operating System) wrap and the cv_bridge. ROS is a robotics library that can be used easily with PX4 for offboard control. Its principle is to link $publisher$ topics to $subscriber$. It uses the MAVROS node to communicate with Gazebo Simulator. MAVROS is the MAVLink extendable communication node for ROS with a proxy for Ground Control Station (GCS). MAVLink for Micro Air Vehicle Link is a very lightweight protocol for drone communication. The topic cv_bridge is the specific MAVROS video link and is published by Gazebo. It gives access to a camera model generated on the simulator that transmits the video.

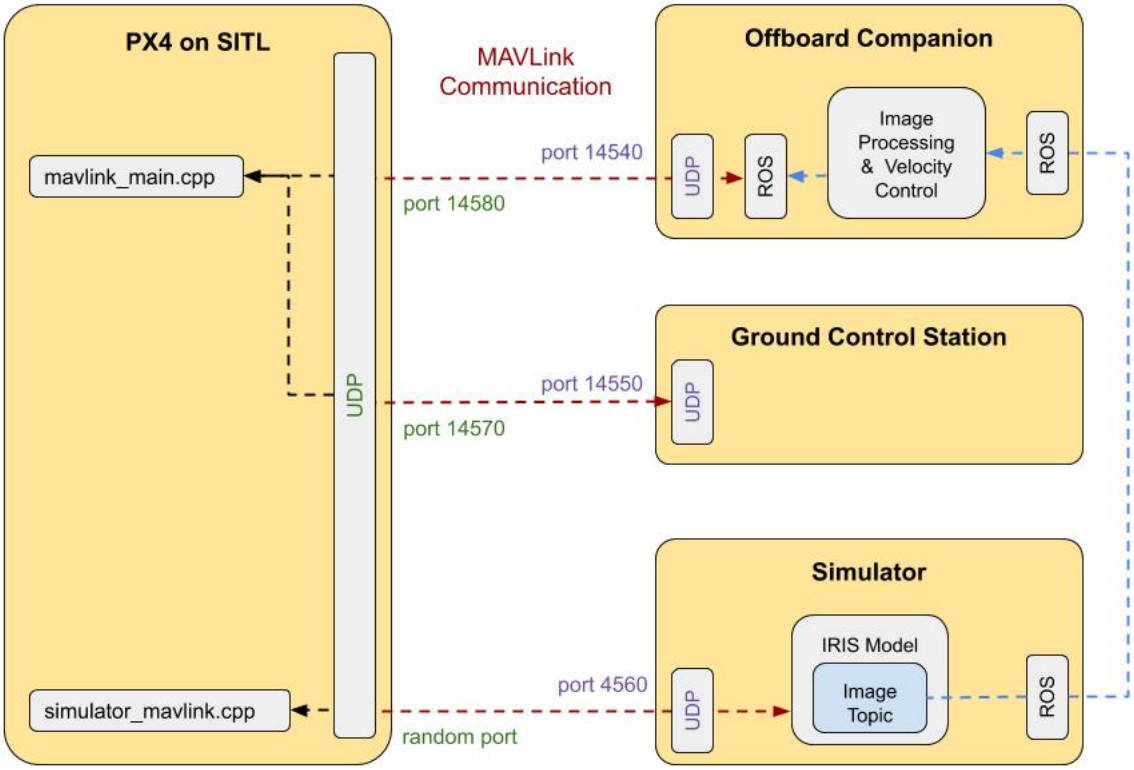


Fig. 4-2 PX4 SITL overview integration

All of the unmentioned settings are provided in the PX4-Gazebo documentation installation.

4.2 Autopilot Presentation

PX4 is the firmware of an open-source autopilot system. It allows an aircraft to be piloted remotely without human intervention. The Pixhawk hardware is the flight control (FC) that embeds the PX4 firmware. The hardware and software are then available to anyone under an open-source BSD license. This license gives anyone the ability to fork the software and use it for industrial purposes.

PX4 running into the Pixhawk is simulated with this integration Fig. 4-2 as SITL. The autopilot behaves exactly as if it were running on the actual Pixhawk FC. Additionally, the command from the GCS can be sent to the drone, and feedback such as position, are visible on the screen. The purpose of having a near-reality simulation is the capacity to transition more easily to real-world applications.

The autopilot allows an offboard mode to control the UAV with an external program. The figure 4-3 below represents an implementation overview of the drone velocity control using image pose estimation,

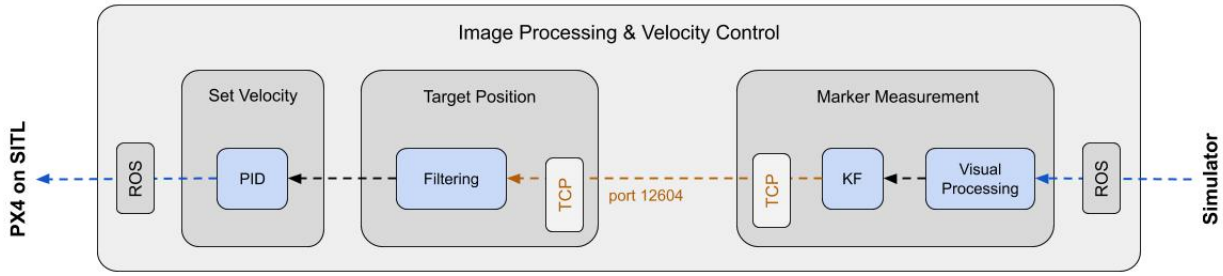


Fig. 4-3 Enlargement of image processing and velocity control integration of Fig. 4-2

This control algorithm provides speed control of the drone and this method of control focuses on the horizontal position of the drone. A pre-stabilization of the drone is done with IMU processing to keep the drone horizontally stable, although it is still prone to drift in the case of wind. Then, the current location is adjusted with the servoing on the fiducial marker position. The next section explains how the measured position is filtered to have an optimal estimation of the drone pose.

4.3 Optimal Estimation

With a camera mounted on a gimbal, its control can make the vision always perpendicular to the ground. The control algorithm of this case is simple because the horizontal pose error is directly proportional to the difference of the image. In this case, the camera is fixed, facing down relative to the drone. With this configuration, some cases send the wrong direction order to the drone. For instance there is a case where the drone has too much angle attitude near to the target. In 4-4, the drone is leaning forward and the marker is head-on using a camera reference frame, but the drone needed to go backward.

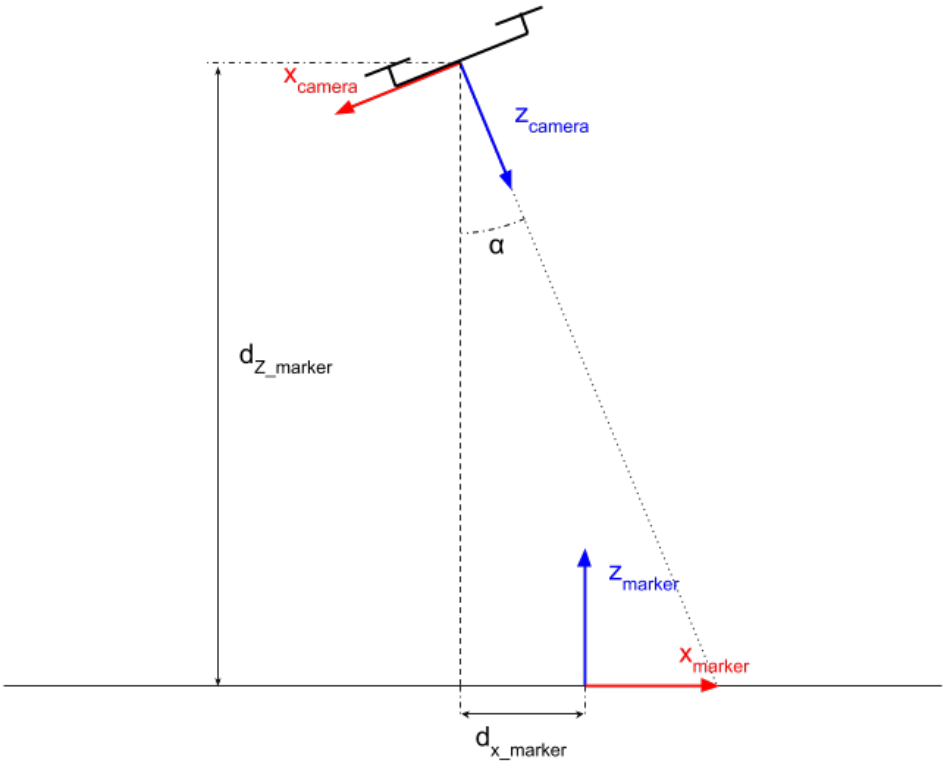


Fig. 4-4 Schematic 2D view of the camera reference frame over the marker

Where α is the pitch of the drone; d_{x_marker} and d_{z_marker} are respectively the horizontal and the vertical distances of the drone respective to the marker. The measurement of these distances is calculated using the equation 3-2. It is relatively noisy due to micro oscillations of the drone and the quality of the image measurement. To smooth data, the signal processing proposed includes KF and a low pass filter. Because the marker is not always detected by the algorithm, it could have a lack of data. The prediction part of the KF can estimate drone position when few positions are measured. Addition of the low pass filter erases micro oscillations and provides drone stabilization.

To land on the target, the thesis approach tries to decorrelate horizontal and vertical control, the KF is applied on two-dimensional positions and linear velocity. These are expressed on the marker system coordinates. The state variable at time k can then be written as,

$$x_{k|k} = \begin{bmatrix} X_{k|k} \\ Y_{k|k} \\ \dot{X}_{k|k} \\ \dot{Y}_{k|k} \end{bmatrix} \tag{4-1}$$

A simple first-order kinematic model is used, thus only position and velocity are considered. Without knowing the control-input model, the state estimation equation 2-34 is,

$$\widehat{x}_{k|k-1} = F_k \widehat{x}_{k-1|k-1} \tag{4-2}$$

and,

$$F_k = \begin{bmatrix} 1 & 0 & \Delta T & 0 \\ 0 & 1 & 0 & \Delta T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4-3}$$

Where ΔT is the sampling period.

2-35 is simplified assuming a negligible noise of the process,

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T \tag{4-4}$$

The estimated noise is assumed to be a Gaussian white noise as a standard approach of the problem, neglecting the little oscillations of the camera measurement that is fixed under the drone. The error covariance matrix with lightweight writing is given by,

$$P_k = \begin{bmatrix} \sigma_X^2 & \sigma_X\sigma_Y & \sigma_X\sigma_{\dot{X}} & \sigma_X\sigma_{\dot{Y}} \\ \sigma_Y\sigma_X & \sigma_Y^2 & \sigma_Y\sigma_{\dot{X}} & \sigma_Y\sigma_{\dot{Y}} \\ \sigma_{\dot{X}}\sigma_X & \sigma_{\dot{X}}\sigma_Y & \sigma_{\dot{X}}^2 & \sigma_{\dot{X}}\sigma_{\dot{Y}} \\ \sigma_{\dot{Y}}\sigma_X & \sigma_{\dot{Y}}\sigma_Y & \sigma_{\dot{Y}}\sigma_{\dot{X}} & \sigma_{\dot{Y}}^2 \end{bmatrix} \tag{4-5}$$

Since variables are independents, and position/velocity measurements are the same regardless of the direction,

$$\sigma_X = \sigma_Y \quad ; \quad \sigma_{\dot{X}} = \sigma_{\dot{Y}} \tag{4-6}$$

and

$$\sigma_X\sigma_Y = 0 \quad ; \quad \sigma_X\sigma_{\dot{X}} = 0 \tag{4-7}$$

So the error estimation matrix can be simplified to,

$$P_k = \begin{bmatrix} \sigma_X^2 & 0 & 0 & 0 \\ 0 & \sigma_X^2 & 0 & 0 \\ 0 & 0 & \sigma_{\dot{X}}^2 & 0 \\ 0 & 0 & 0 & \sigma_{\dot{X}}^2 \end{bmatrix} \tag{4-8}$$

Then, only X and Y positions are measured, so innovation 2-36 has,

$$H_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{4-9}$$

The low pass filters remember the old measurements to attenuate high frequencies,

$$x_{k+1} = (1 - K_L)z_k + K_L x_k \tag{4-10}$$

Where K_L is an empirical factor of the low pass filter which does not add delay; x_k is the filtered state at time k; z_k is the measured state vector at k.

A measurement of filtered values is shown on the figure 4-5 below,

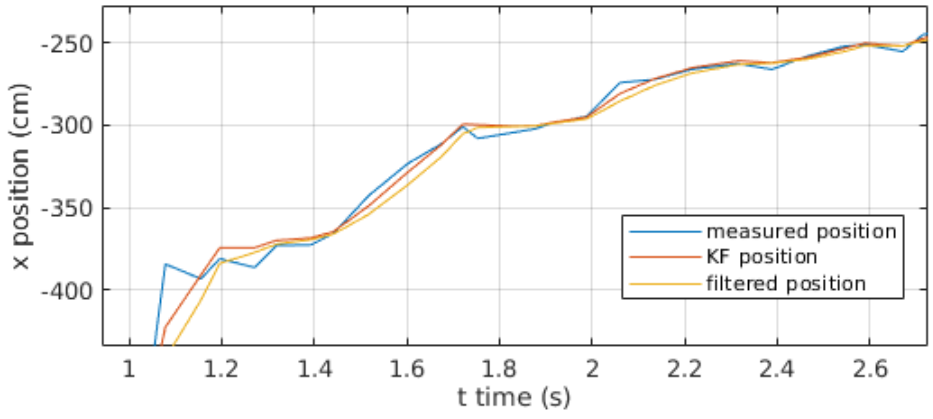


Fig. 4-5 Representation of a pose measurement with KF and low pass filtering stage relative to the time

KF allows to smooth the measurement, but some variations are still notable. The implement of a well-tuned low pass filter reduces these variations without decreasing the instantaneity. The result is a smoother curve that reflects a more continuous drone movement. The comparison result of the three different pose estimators shows that the accuracy of the measure-

ment could depend on the altitude. The closer the drone is to the ground, the measurement is resultantly becoming more accurate. The solution considered in this optimal estimation is to keep the coefficients constant from equation 4-1.

4.4 Drone Guidance

Using the command provided with PX4, the first option was to try and control the drone position to land, which is the standard drone landing approach. It consists of two steps: horizontal and vertical drone movement. This technique uses GPS localization and relatively low vertical descent. The proposed approach is to mix horizontal and vertical movement. It consists to give the vehicle a vertical speed and adjust continuously adjust the horizontal position.

4.4.1 PID Control

To accurately control the drone, the environment measurement needs also to be accurate with a good refresh rate. It consists to estimate the 6 DOF drone pose, including orientation, relative with landing pad measurement. This data processing ensures good quality of the system input. The figure 4-6 below follows the proposed IBVS block diagram.

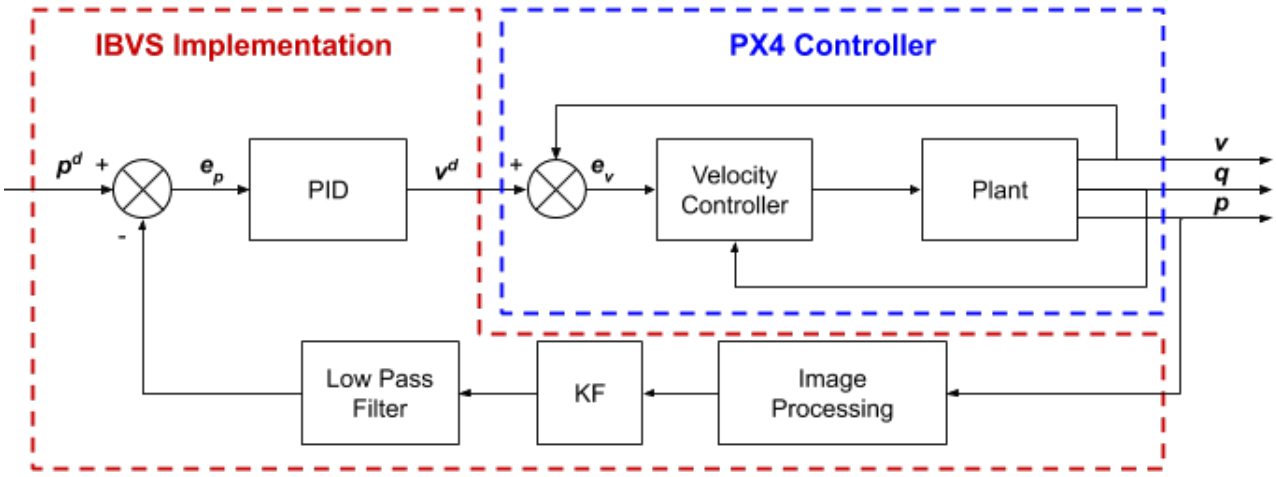


Fig. 4-6 Block diagram of the proposed IBVS implementation. p is the drone position; p^d is desired position; e_p is error on position; v is the drone velocity; v^d is desired velocity; e_v is error on velocity; q is drone attitude.

The part circled in red is the proposed IBVS Implementation, while the blue part is a general overview of the PX4 FC. Here is used a semi-controlled mode. The drone is horizontally stabilized and controlled with linear velocity inputs. For the simulation as for the real implementation, the plant is not mathematically known. A more accurate model is presented in [42], but in this study, the plant can be considered as a linear time-invariant system model with unknown coefficients,

$$\dot{x} = Ax + Bu \tag{4-11}$$

$$y = Cx \tag{4-12}$$

Where x is the state vector; u is a command vector; A, B and C are unknown matrices.

The PID coefficients have been found manually for a sufficient drone servoing. The Ziegler-Nichols [76] method was attempted, however, it did not provide satisfactory results.

4.4.2 Multi-Scale Control

This control directly follows the use of the Fractal marker. When a marker of the scale bellow is visible by the drone, an adapted control is provided. This is a specifically tuned PID. It takes the assumption that the drone is better centered than at the previously seen marker. As a result, PID coefficients have higher values.

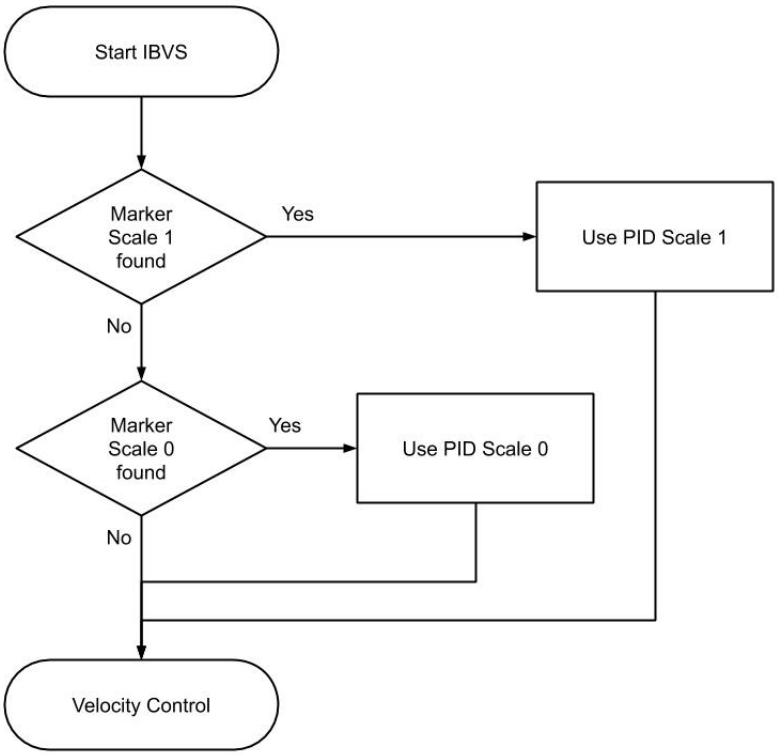


Fig. 4-7 Decision diagram to set the scaled PID

In figure 4-7 above, the algorithm is set with two scales, but the process can be generalized with n scales. The algorithm has a discontinuity at the switch. The effect of the switch will be briefly observed in the experimental section, later in this chapter.

4.5 Landing Simulation

The drone is an IRIS model on the Gazebo simulator. As mentioned prior, the camera is fixed and facing down. The vertical speed is set to the maximum permitted by the FC. With GPS localization it is relatively easy to autonomously bring the UAV to the proposed initial position. The initial altitude is set to *10 m*, which is the fixed height where the transition between GPS localization and visual localization can be done. At this altitude a *0.8*0.8* m^2 marker is well detected and the horizontal (x_{ENU}, y_{ENU}) position is then set as random but visible by the UAV in the respective range $([-4, 4], [-2, 2])$. The following experiments compare different autonomous drone landing scenarios. Each guidance technique has been done *30 times*.

4.5.1 PID Control Simulation

After normalization and synchronization of the measurements, the drone landing trajectory can be plotted. The following trajectory in Fig. 4-8 has been obtained for a fly with the initial world position $(4.9, 1.4, 10)$. The point $(0, 0, 0)$ is aimed by the drone and trajectory is adjusted thanks to its visual measurements.

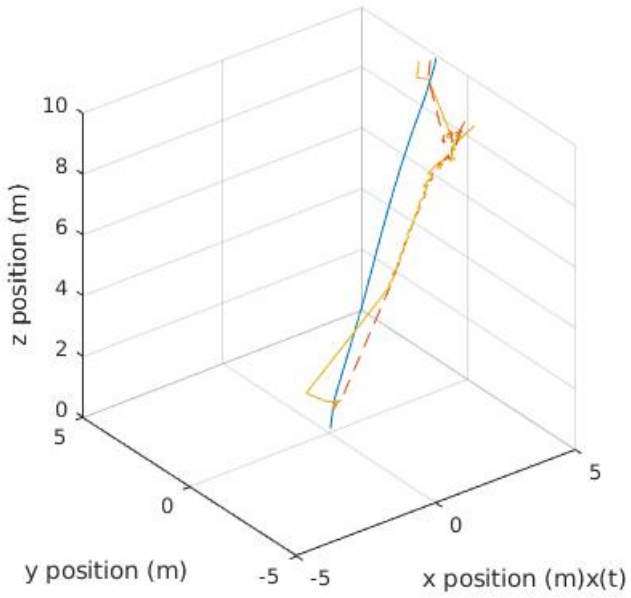


Fig. 4-8 Drone landing trajectory (blue) with measured position (red) and estimated position (yellow) from initial position $(4.9, 1.4, 10)$

The three following landing outcomes (4-9, 4-10, 4-11) are displayed on a $2 * 2m^2$ area graph to make further comparison easier. It begins with the landing position on Fig. 4-9 using an ArUco marker,

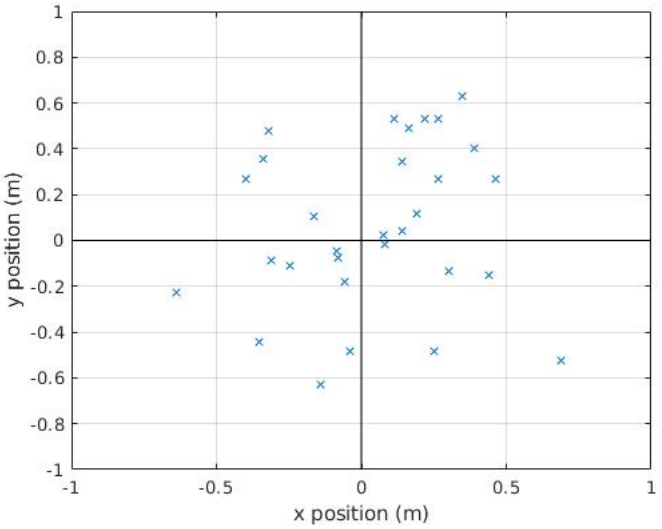


Fig. 4-9 Landing positions of the drone using ArUco marker and PID control

Using the *0.8*0.8* m^2 ArUco marker, the drone lost vision of the target at *1.5 m* altitude. The final positions visible on the graph 4-9 are centered around the origin which is the target point. The landing position using the NN is then plotted on the following figure 4-10,

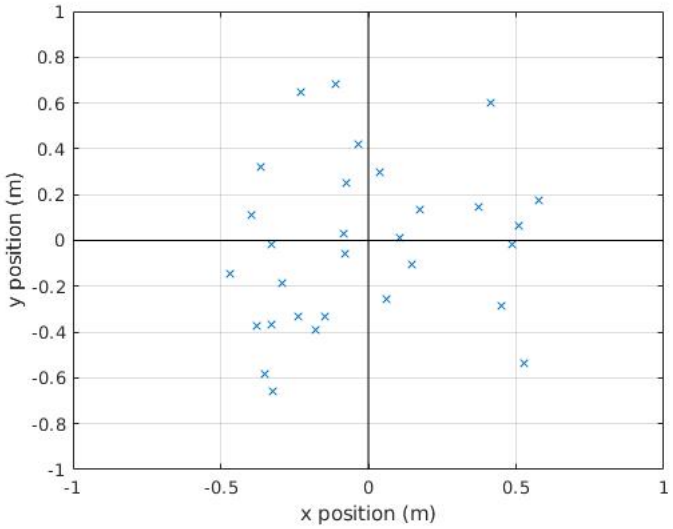


Fig. 4-10 Landing positions of the drone using the proposed NN and PID control

The position plotted on Fig. 4-10 looks similar to the previous one 4-9 using ArUco marker. During the descent, the drone is shown to reach the vertical of the marker faster in comparison to ArUco.

4.5.2 Multi-Scale Control Simulation

Fractal markers are used for this experiment. The size of the larger one is kept to $80 * 80$ m^2 and the control algorithm is adapted according to Fig. 4-7.

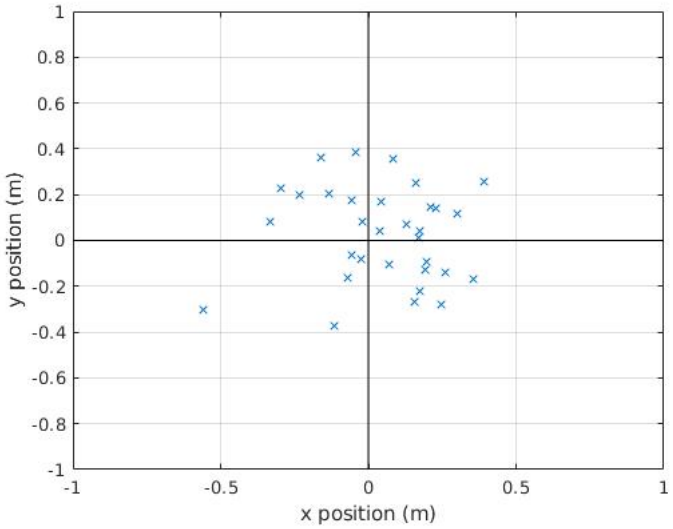


Fig. 4-11 Landing positions of the drone using a fractal marker and a multi-scale control

The plotted positions of figure 4-11 look more grouped than the other two series of measures. No noticeable movement has been detected at switch of PID coefficients of the multi-scaled control.

4.5.3 Comparison Study

The multi-scale control is almost two times more accurate than the standard control on the ArUco marker. The standard deviation shows more precise results.

Tab. 4-1 Comparative table of the landing precision, the duration, as well as the frequency of calculation of three computer vision algorithms

Algorithm	**x** (μ, σ^2) (m)	**y** (μ, σ^2) (m)	**Duration** (μ, σ^2) (s)	**Rate** (Hz)
ArUco	(0.0455, 0.0086)	(0.0597, 0.0166)	(10.62, 0.0155)	7.06
Fractal	(0.0462, 0.0021)	(0.0290, 0.0018)	(10.57, 0.0148)	4.26
Proposed NN	(-0.0179, 0.1050)	(-0.0250, 0.1266)	(10.61, 0.0146)	22.55

Precision

The table 4-1 above compares the accuracy of different control algorithms using different pose estimations. The drone position is considered to follow a normal distribution with mean μ

standard deviation σ^2. Even if the mean position of the proposed NN is closer to zero, the reader will notice that its standard deviation is almost ten times larger than the ArUCo and almost a hundred times larger than the fractal. This highlights the lack of precision of this technique. Moreover the fractal, the multi-scale strategy is almost two times more accurate than standard ArUco with PID on *y-axis*. The duration of the landing procedure and the refresh rate are also compared below.

Speed

The three strategies have the same duration almost *10 s* for *10 m*, which means that in the three cases the UAV has a vertical velocity of *1* $m.s^{-1}$. The standard GPS drone landing is not studied for the landing position. The GPS noise of the Gazebo simulator is too low compared to reality. Previous attempts [52], show that the accuracy does not reflect real a GPS position. However, the landing duration is a suitable characteristic measurement. The landing duration with the Gaussian measurement is $(14.44, 0.40)$. This shows that the GPS landing has an average of $4\ seconds$ longer, and this duration can vary greater than the visual servoing landing.

In the simulator, the maximum vertical speed of the IRIS drone is reached. Even with a fine tune of the PID coefficient, where the drone reaches the vertical target faster, the landing time is the same. Otherwise, the need for a different PID for the different scaled marker shows a non-linearity of the plant.

Rate

The rate is the number of data processed over time. for this study, the same computer has been used. This is a laptop with:

- 8 Go of RAM ;
- Core i5-9300H ;
- GTX1650 as a GPU.

This is a relatively low configuration as a computer, but it can easily be compared as an embed computer. The rate result is relatively inferior to the one that can be found in different studies. Aside from the low configuration of the calculation module, programming codes with Python can also explain the slowness of the process. However, the ratio of the calculation speed of the different algorithms to each other remains a good comparison tool. Thus, the speed of calculation of the strategy using the neuron network is three times faster than the ArUco standard. And even if the fractal is almost twice slower than the ArUco, that did not prevent it from being more precise.

Other Studies

One can notice that the visual measurement obtained in Chapter 3 is not comparable with the one obtained in this section. Even if the size of the landing pad differs between the two studies, the results are counter-intuitive because both image resolutions are the same. When using a larger landing pad, the target is seen closer to the ground than in the previous study. The first explanation is that the random generation of data does not cover all possible data. Therefore, data positions closer to the ground could have been missed. The second explanation is the use of different cameras on different software. The calibration on Blender and Gazebo has resultantly been performed to correct distortion. The final explanation is the difference in luminosity between the two software. Even using an adaptive threshold, fiducial detection is very sensitive to luminosity. All of these combinations explain the variation observed in the data measured in the experiments within Chapter 3 and 4.

In comparison to both of these review papers [6] [7], landing duration has been respectively divided by seven for the same initial altitude. Besides, it is the same value as the one starting at $3m$ to achieve the same accuracy for landing. The reduction of flight time has the drawback of increasing velocity at landing. The next chapter proposes an adapted landing leg to perform a high-speed landing.

5 Mechanical Landing Gears

Drone legs are the only point of contact between the drone and the ground. An ideal landing should keep the drone stabilized on the ground and absorb part of impact at landing. Standard drone landing gears embed a small piece of foam to absorb impact and attempts to stabilize the drone on a flat landing area. Often, the drone lands with a high velocity in comparison to the size of its landing gear. Even if the standard landing approach minimized vertical velocity, an unwanted bounce would still result after ground impact. Rebounds generally have insignificant effects, however, landing on a small area can make the drone bounce and coupled with high velocity can result in damage. Lastly, a failed landing can cause irreparable damage to a drone as well as its payloads such as an expensive camera or an important package holding, or even its internal electronics. Thus, the addition of shock absorber aims to protect the drone from jolts at landing, while also increasing speed and maintaining control at landing.

Some theoretical parameters used for the simulation were introduced in *Basic Theory* section. Figure 5-1 shows a schematic representation of the landing gear and introduces the variables of the simulation.

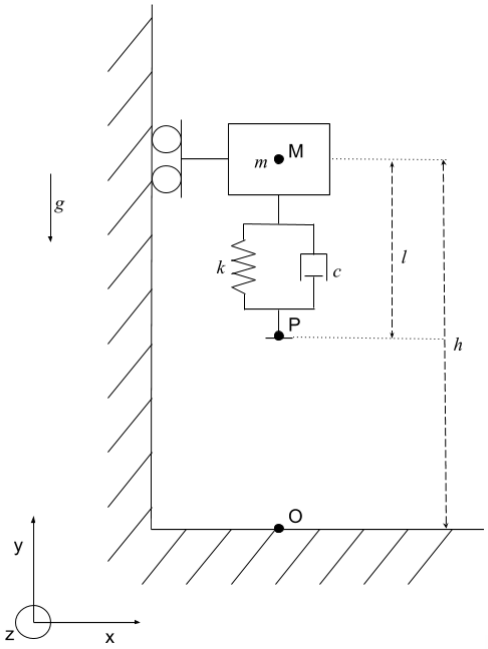


Fig. 5-1 Scheme of landing drone

For a perfect land, without wind or ground effect, the four landing gears of the drone are assumed to touch the ground at the same time. Moreover, the drone attitude is controlled by the autopilot that helps to stabilize it horizontally. The schematization is justified as having

an equivalent damper of the four real ones on a standard drone. Also, this scheme supposes a perfectly vertical trajectory.

5.1 Absorption Simulation

In this section are presented the drone motion equation involved for the landing. Then the dynamic is simulated to finally justify its interest.

5.1.1 Movement Equation

The following reasoning works under two main hypothesis:

- $v_y >> v_x$ which means that only movement over *y-axis* is considered.
- No rotation of the object because drone stabilization in attitude is considered perfect.

Using force equations with the equation of the falling object, ones obtain falling drone equation,

$$-mg - bv + N = ma \tag{5-1}$$

or

$$\ddot{y} = -\frac{b}{m}\dot{y} + \frac{N}{m} - g \tag{5-2}$$

Using collision equations, when the drone hits the ground $y < l$, one can find drone bounce equation,

$$\ddot{y} = -\frac{c}{m}\dot{y} + \frac{k}{m}(l - y) - g \tag{5-3}$$

5.1.2 Landing Kinetic

The landing dynamic is designed using a block scheme of Simulink. It use previous equations 5-2 and 5-3 to model effects of a given initial state. Then, acceleration a, velocity v and position y are plotted.

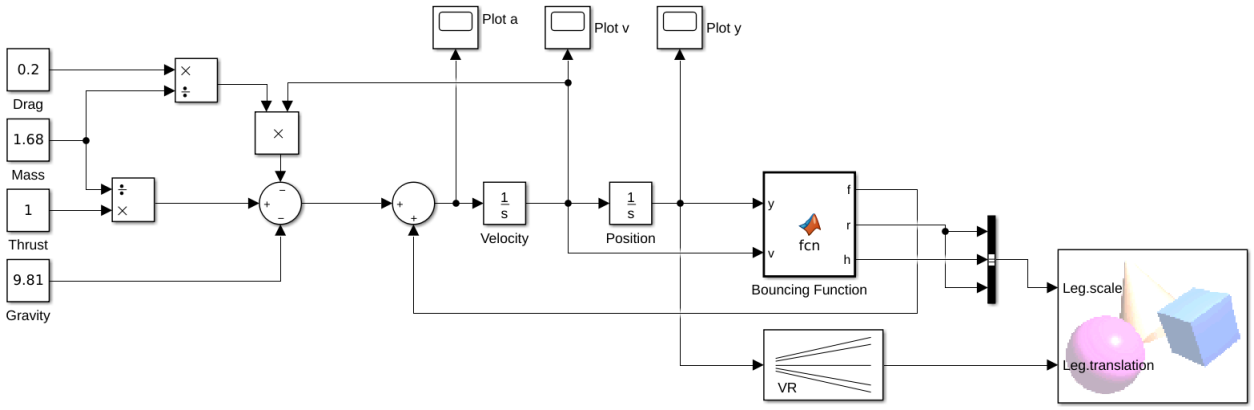


Fig. 5-2 Simulink block scheme of the hybrid system movement equation

The free-fall part of the simulation has been modeled only with block provided by Simulink, while the bounce part has been written as a function.

A quantitative study follows with the landing dynamic observed also made possible by Simulink. Characteristic values of a shock absorber $[k, c]$ are fixed for a series of measurements, and their effect on the movement is observed in the following graphs 5-3, 5-4, and 5-5. Parameters of the simulation are fixed according to appendix A.3.1 and try to get as close to reality, but the elastic and friction coefficient is proposed with a visible and characteristic effect on the collision motion.

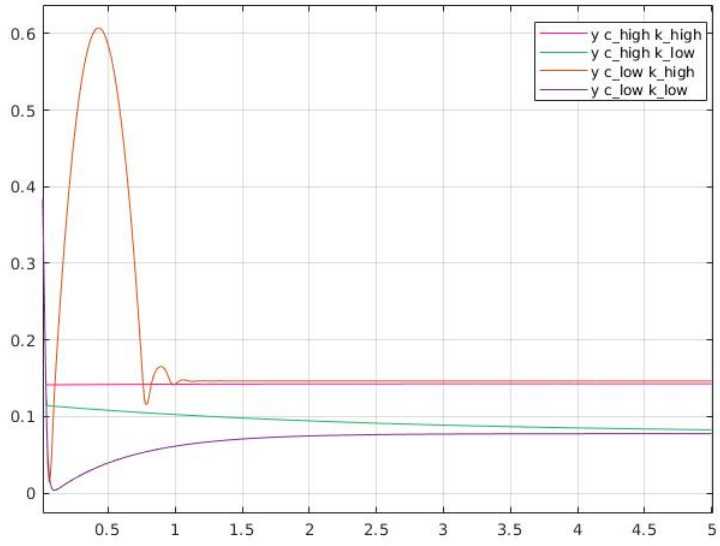


Fig. 5-3 Drone positions at landing for different shock absorber coefficients

Important features can be observed with the measures on the graph 5-3. An impact with a significant stiffness k and a low elastic coefficient of c is the only one bouncing. It reaches $60cm$ at its first jump after landing. One can notice that for a low c, a shock absorber has the maximum oscillation amplitude. This to such an extent that the system almost touches the ground at less than $2cm$. Conversely, high friction coefficient c involves a small oscillation. Explanation on subsection 5.1.3 argues why the final position in the amplitude of the shock absorber is not an issue.

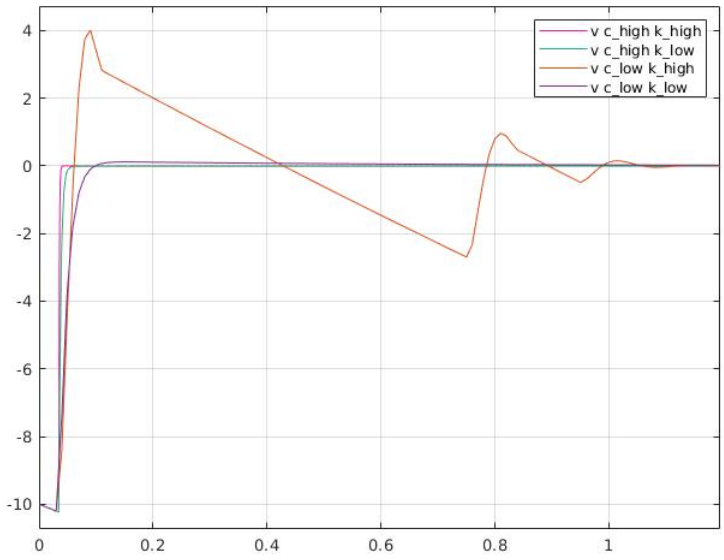


Fig. 5-4 Drone velocity at landing for different shock absorber coefficients

On figure 5-4, velocity at impact is $v_{max} = 10.2\ m.s^{-1}$. In $35cm$ drone speed increased of $0.2\ m.s^{-1}$. The four simulated drones have the same impact speed, and the same kinetic energy.

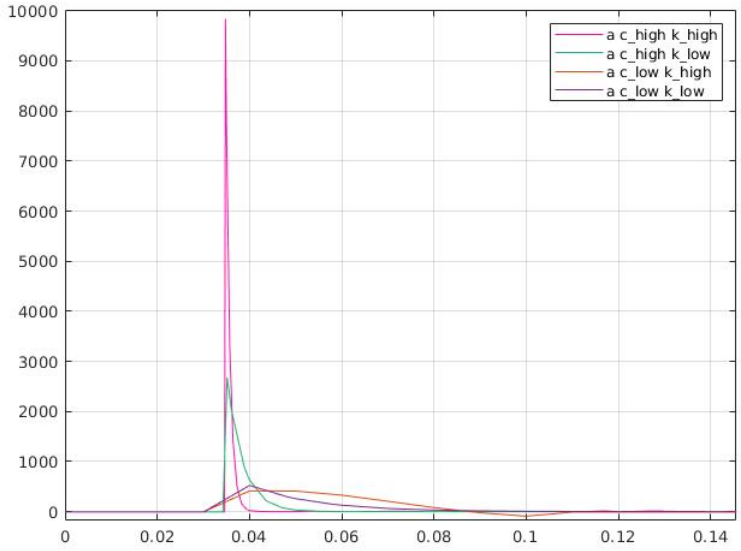


Fig. 5-5 Drone acceleration at landing for different shock absorber coefficients

Maximum acceleration is observable in Fig. 5-5 is reached with a high c value. Combined with a high k, the simulated drone undergoes $1000G$ acceleration, and drops to $273G$ with a low k. A low c shows diffuse deceleration, which does not exceed $50\ G$.

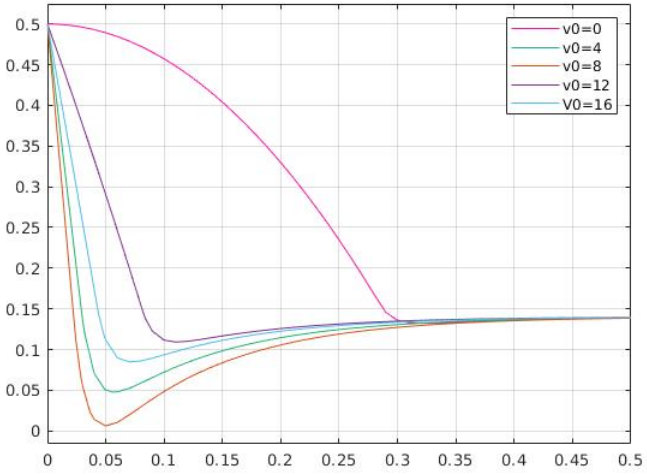


Fig. 5-6 Drone position over time for different initial velocity

Then with fixed coefficients k and c according to A.3.1, it is convenient to find maximum speed at landing. Here on figure 5-6 it is for $v_0 = 16\ m.s^{-1}$ where $v_{max} = 16.1\ m.s^{-1}$. This correspond to the curve approaching as close to the ground.

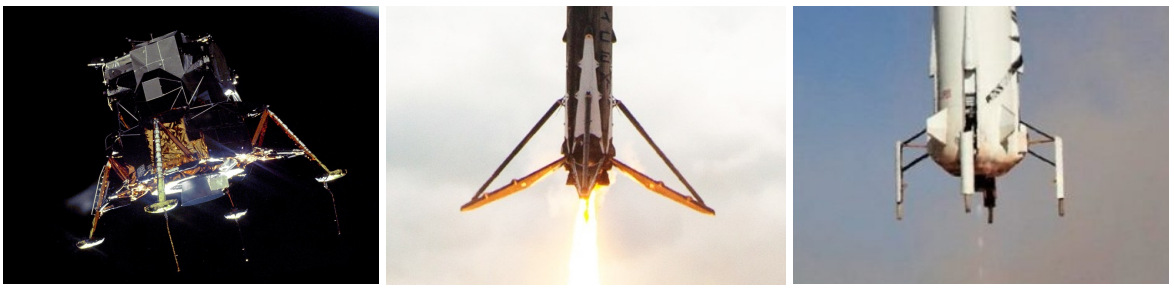
5.1.3 Interest of Shock Absorber

Every material and mechanical structure can be modeled as a shock absorber. But most of these have low absorption effects and will bounce. A good example of materials that have low absorption effects is standard plastic drone landing gears. As a result, they bounce when they have speed at ground collision, and bounce height increases with velocity. Contrary to living beings, machines are less sensitive to important acceleration or deceleration. The only matter for an object deformation at collision is to avoid going into material plasticity.

A standard landing approach focuses on reducing speed to avoid bounce at landing. However, the goal of this thesis is to challenge the standard approach by reaching the maximum speed at landing through using adapted landing gears. This approach can reduce flight time, protect the platform in case of lack of battery, and increase landing control to avoid bounce. Therefore, a shock absorber is required. It will convert kinetic energy into heat with friction. The spring on a standard shock absorber helps the system get back to its initial position. Drone landing, similar to rocket landing, only requires impact absorption. Without any spring effect, absorption systems can be deployed manually between two flights. Though, the only use for drone landing gears is to ensure landing. The capacity to land then take off without outside intervention can be achieved using a relatively soft spring. It is easier to implement for fluid friction and will have a long response time. It doesn't really matter for a drone, because legs will have time to get to their original position during flight time.

5.2 Absorber Gear Kinematic

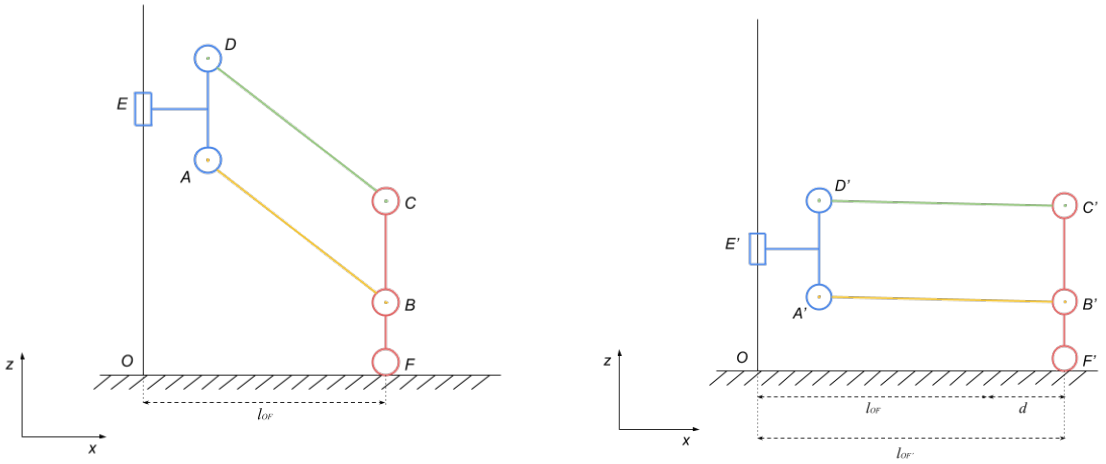
The standard design of landing gears is made for a relatively minimal shock. This implies a small displacement of legs at landing. Two kinematic are noticeable for landers and landing rockets: three hinges and four hinges. Three hinges are the most common ones, and the first lunar landers (5-7a) have already used them. This kinematic is also used by Space X for the reusable rockets (5-7b). These two examples effectively absorb a bit shock at landing, contrary to blue origin (5-7c). Certainly, the kinematics of the landing feet has 4 points of rotation, but the rebounds visible during its landing show that the entire structure absorbs the energy.



(a) Apollo 11 Lunar Module (b) Space X Falcon 9 (c) Blue Origin

Fig. 5-7 Images of landing gears of different landers

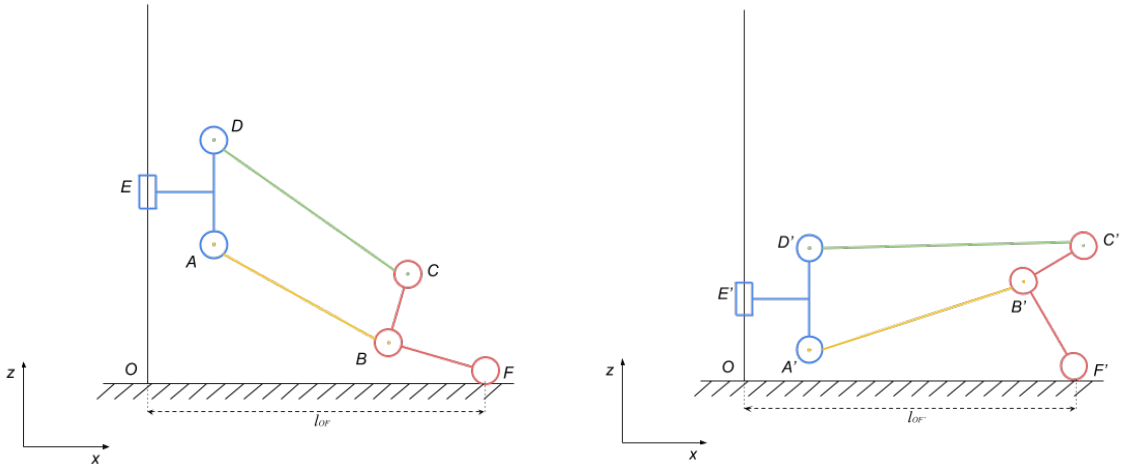
More complex kinematic absorbers exist for downhill mountain bicycles, and some have more than *five swivels*. Therefore, shock absorption is increased tenfold in a restricted displacement. The research is focused on a relative system, so only four swivels will be used. The proposed kinematic tries to minimize the vertical size of the legs. The maximum travel of the landing gear is increased with a size ratio, and displacement of the absorber is amplified at the extremity of the leg. Moreover, the point of contact of the leg has an almost rectilinear displacement which reduces transverse forces and the foot only undergoes compression.



(a) Four swivels parallelogram at initial position (b) Four swivels parallelogram at final position

Fig. 5-8 Four swivels parallelogram landing gear kinematic at landing

A perfect parallelogram mechanism during compression implies a horizontal displacement of the contact point 5-8. Point F moves d distance to F'. This type of displacement has not managed an effect. In the real world, it would be unable to predict how the floor would react. This depends on the type of floor: cement of tarmac, the sand of the desert, or grass of a field. Thus, a vertical landing movement has to minimize horizontal displacement. The proposed kinematic on figure 5-9 tries to reduce d.



(a) Proposed four swivels at initial position

(b) Proposed four swivels at final position

Fig. 5-9 Proposed landing gear kinematic at landing

The length of different arms is mentioned in the appendix. The reader will make attention on the difference between figure 5-9a and 5-9b where the distance OF is equal to OF'. This proposed kinematic thus reduce the observed displacement in Fig .5-8.

5.3 Landing Gear Design

This section presents two landing gear designs: a classical rigid mechanism and a compliant mechanism.

5.3.1 Rigid Landing Gear

For the experiment, a radio-controlled (RC) car shock absorber was used. It has two friction coefficients: dry friction and fluid friction. The first one can be modified with adapted fluids with different viscosity. The second one can be modified with different o-rings around the shaft in translation. Moreover, the spring return force can be changed using different springs. A leverage effect was chosen, as the travel distance of a shock absorber is fixed and relatively short. It is implemented on a rotation bar with the ratio that maximizes both landing gear travel and shock absorber travel. For the dimension of the drone, the shock absorber has a horizontal placement that is more suitable given its size. For rocket implementation, vertical placement of the shock absorber is preferred. To distribute effort symmetrically and avoid bar collision, the shock absorber goes through the mainframe, and this design is observable on 5-10c.

Shock absorbers used for the proposed landing gear come from remote-controlled (RC) cars and are observable on figures 5-11. A set of different springs permits the modification

elastic effect. The friction coefficient could be changed using different viscous fluids. Here, the adopted solution is to use tighter o-rings. More, the friction effect will also depend on the tightness of the hinge. Made from a bolt, it can increase friction effect or not if it is screwed or loosen. The addition of some oil allows reducing static friction of the o-ring.

(a) Face view (b) Side view (c) Three quarter view

Fig. 5-10 General assembly views of the mechanical landing gear

Plastic arms that compose the landing gear in Fig. 5-10 are 3D printed in a basic Polylactic Acid (PLA). Standard printing parameters for this kind of plastic are kept and infill density is 10% with gyroid shape. This infill allows an isotropic reaction of the piece and is suitable for a priori test.

(a) Extended rigid landing gear (b) Compressed rigid landing gear

Fig. 5-11 PLA 3D printed rigid landing leg with RC shock absorber

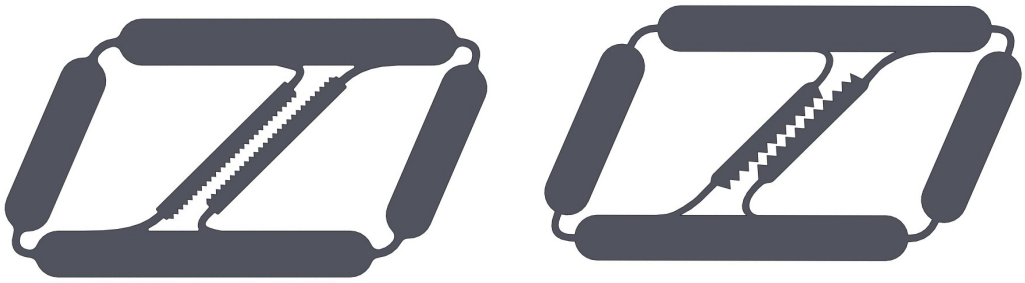
The printed rigid structure 5-11a with shock absorber reaches a weight of $75g$, meaning that the entire drone will increase its payload to $300g$. The leg compressed shows on figure 5-11b a $6cm$ absorption amplitude.

5.3.2 Compliant Landing Gear

Contrarily to a standard rigid structure, the compliant landing gear tries to provide an alternative to the traditional shock absorber. It is possible to print landing gears and mount them faster on the drone.

Absorption Effect

Given a compliant material, a notable spring effect is evident on the hinges. The thickness gives the spring effect: a thin beam has a soft effect while a thicker beam has a more rigid effect. Then, to look like a shock absorber, the friction coefficient needs to be added. Current theories assure that when two diagonal hinges distance grows for convex four-bar mechanisms, the two others get closer. From where the first design has been proposed below on figures 5-12,

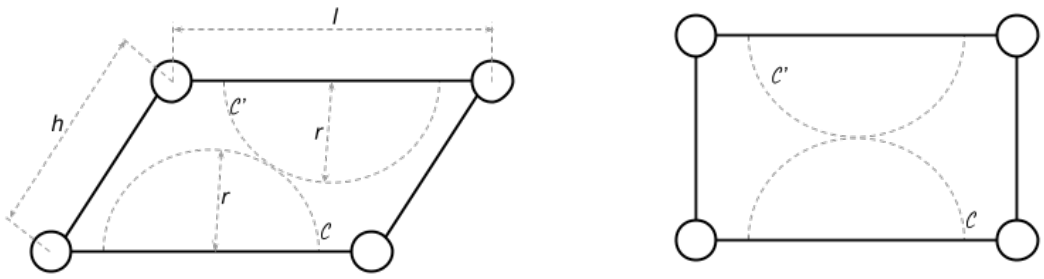


(a) Compliant architecture with friction slats

(b) Improved compliant architecture with friction slats

Fig. 5-12 Four-bar compliant mechanisms with two frictions slats

The second proposed design (Fig. 5-12b) tries to improve the first one (Fig. 5-12a) with a better friction effect and an easier displacement. Unfortunately, the friction effect has been shown too soft on an empirical test. However, a more reliable technique is then proposed. From the basic parallelogram architecture, the rotation of the upper part over the crank is noticeable in Figs. 5-13. It exists at some points that are constantly equidistant during this rotation. These points describe arcs of circles drawn in dotted lines in the following diagrams,

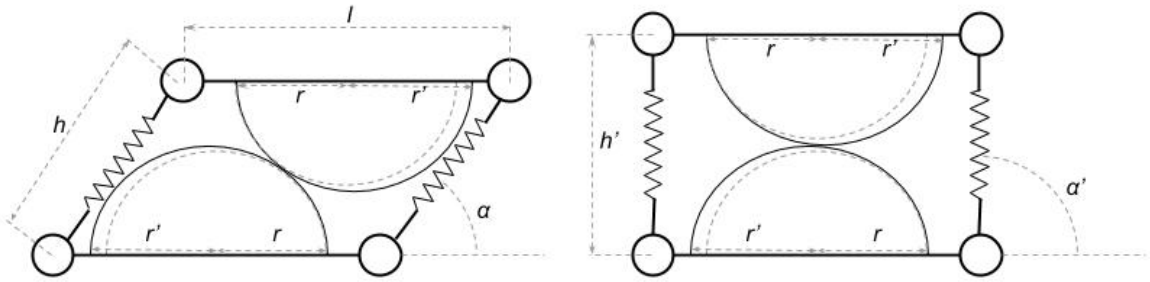


(a) Four-bar mechanism at initial position

(b) Four-bar mechanism after a rotation

Fig. 5-13 Rotation of four-bar parallelogram mechanism

On 5-13 C and C' are always tangent. The proposed idea is to decease this distance by increasing radius curvature along with the rotation.



(a) Four-bar compliant model at initial angle α (b) Four-bar compliant model at angle α'

Fig. 5-14 Rotation of four-bar compliant mechanism model

The proposed scheme of the system is composed of two parallel springs to simulate the normal force to sustain over the two arcs of a circle. On Figs. 5-14, between α to α' elongation of the spring is $\delta h = h - h'$. Over a rotation of $\pi\ rad$ of the system radius curvature goes from r to r'. The radius depends on α and increases linearly in order to write $r(\alpha)$. Assuming the spring return force,

$$F_k = -k\Delta h \tag{5-4}$$

where k is constant elastic coefficient and Δh is elongation of the spring,

$$\Delta h = 2r(\alpha) \tag{5-5}$$

thus combining 2-51, 5-4 and 5-5, for a given material, the absolute friction coefficient of this structure is,

$$F_f \leq 2r(\alpha)\mu k \tag{5-6}$$

Internal mechanism elasticity enhances linearly effort applied to the touching arcs. This explains why the theoretical model is adjusted using springs instead of bars and increases the normal force applied from one surface to another surface. The friction between the two parts during the movement increases.

(a) First compliant design (b) Second compliant design

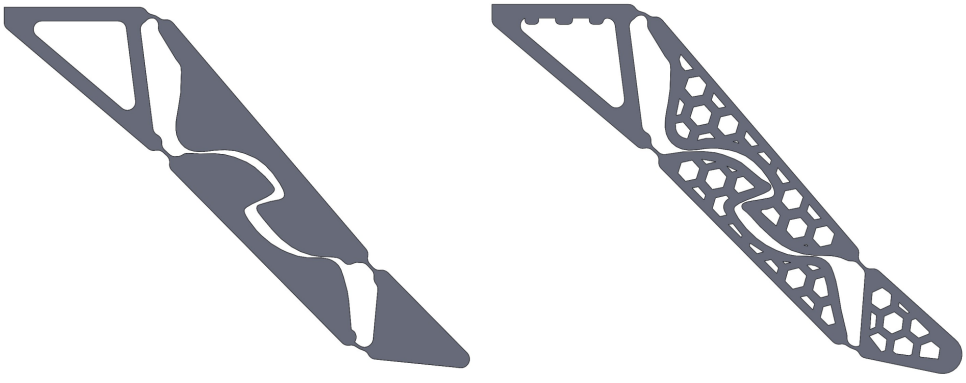
Fig. 5-15 3D printed compliant four-bar absorber

An empirical comparison of the two proposed designs of figure 5-15 shows that the expected effect is more noticeable for the second proposed design. It is also easier to control the friction coefficient. Indeed it is relative to radius curvature of this part.

Compliant Design

This compliant structure is made from one full, 3D printed piece. It is easier to replace it as a unique piece in case of brakes. Without a metallic shock absorber, this mechanism tries to have the same weight as a traditional landing leg, keeping the interesting frictional effects.

The compliant landing gear tries to reuse 5-16a found for the perfect parallelogram and apply it to the proposed kinematic. Hinges of the landing gears are in right-circular shape for the following proposed designs of figure 5-16,



(a) Compliant landing gear shell (b) Compliant landing gear proposed shape

Fig. 5-16 Compliant landing gear design

In 5-16a the two inner curved part has an increasing radius to increase friction during landing absorption. The final design 5-16b compared to the left one has different modifications.

The radius of the frictional part increases drastically at the end of travel. This affects to block the four-bar mechanism motion when the leg is fully compressed. Then to get a light-weight structure keeping rigidity, the landing gear has been drilled with a hexagonal compact shape.

(a) Extended compliant landing gear (b) Compressed compliant landing gear

Fig. 5-17 PP 3D printed compliant landing leg

To have good compliant properties, the plastic used is polypropylene (PP). Even if it is more complicate to 3D print, this plastic has impressive resistance to fatigue and collision. For these reasons, it is widely used by industry for every day containing like saucepot or shampoo container. For the printing, there is no infill density, the rigidity is allowed by the wall of the hexagonal shape. The thickness of the landing gear is chosen as $15mm$ and the thickness of the right-circular hinge is $2mm$. On the real compliant leg, the curvature radius is more important than in theory. This compensates the natural flexion of the material to have the expected frictional effect.

During compression observable on figure 5-17b, the curved parts in contact are subject to friction. Hinge elasticity assures the contact of the frictional parts, and elongation of the radius increases the frictional effect. A slight twist of the upper bar is noticeable in figure 5-17b. The leg compressed has, like the rigid leg, $6cm$ absorption amplitude. The effects of a standard shock absorber are then reproduced on a unique piece.

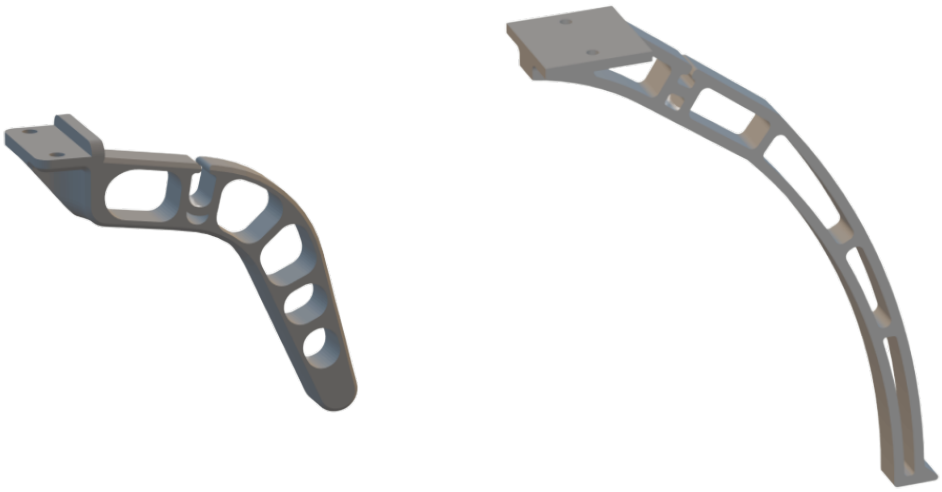
5.4 Shock Absorption Experiment

This experiment tries to confirm the feasibility of shock absorber and effect at ground impact. Different shock absorber will be compared:

- A *little commercial* landing leg that is sometimes used for the common *F450* frame.
- A *big commercial* landing leg that is also often used for *F450* frame.
- The *rigid* landing leg with k_{min} and c_{min}: the softer springs are used and the original o-ring having low friction effect stay on the shock absorber.

- The *rigid* landing leg with k_{min} and c_{max}: difference with the previous one is the use of tight o-rings that are specially PP 3D printed.
- The *rigid* landing leg with k_{max} and c_{min}: it doubles the hardest spring to maximize spring effect and uses the original o-ring.
- The *rigid* landing leg with k_{max} and c_{max}: use both, the doubled hard spring and the tight tailored o-spring.
- The *compliant* landing leg that is a unique 3D printed piece.

The commercial landing gears have been 3D printed in PLA and have the following shape,



(a) Little commercial landing leg

(b) Big commercial landing leg

Fig. 5-18 Images of commercial landing gears used during experiment

5.4.1 Experiment Procedure

A unique landing gear is studied at once, and the weight of the mobile is a quarter of the total drone weight to have the same effect as one landing gear. To keep a vertical fall and avoid any rotation, the system is mounted on a rail. Here, three cables are tightened and three equidistant tubes slide along. These tubes are 3D printed in PLA and attempt to minimize friction during descent. The system has three arms because it has been decided that it is the perfect number to stabilize the descent, without too much friction and to keep the mobile horizontal.

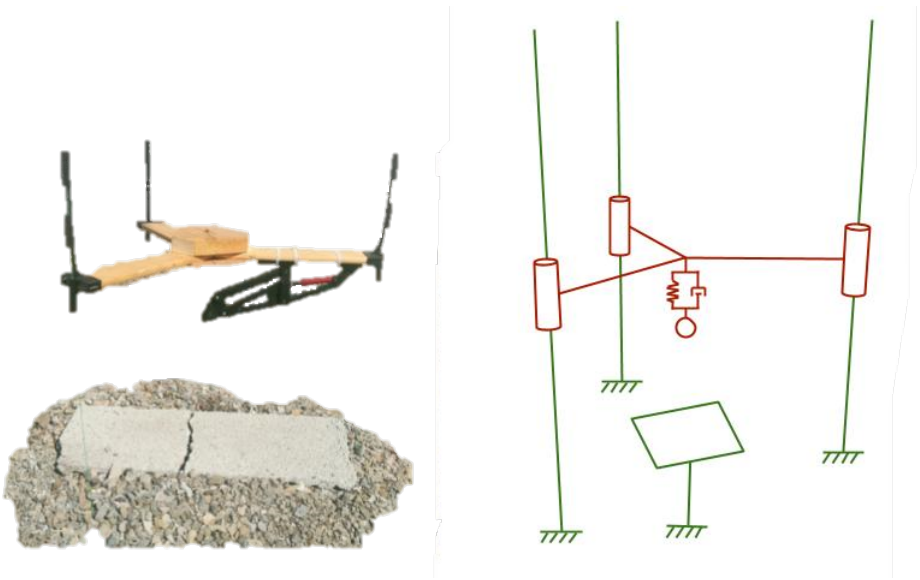


Fig. 5-19 Photo of the experiment benchmark on the left and its schematic kinematic linkage on the right

A standard tarmac, made of cement is really hard. For this reason, the experimental landing area is made from a concrete block. This is the worst situation for the drone because all its kinetic energy will be absorbed by its own structure. Each drop is filmed with a fixed camera. A distance reference is placed on the video to scale measurements, and a visual tracker is placed on the mobile. The video data are then processed using $Tracker\ v5.1.4$, video analysis, and modeling software.

5.4.2 Experiment Results

Following the previous procedure, the results are shown below. In this section, ”*camera frame*”, contrary to the usual meaning given, means the number of the filmed image. It begins with the little commercial landing leg 5-18a,

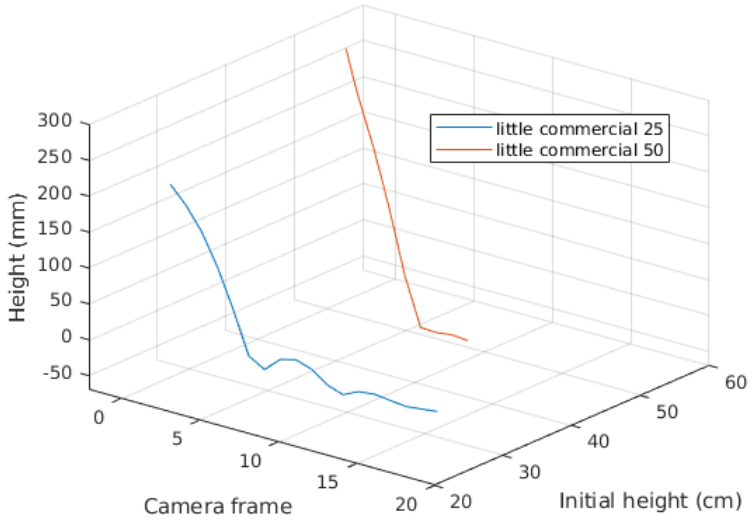


Fig. 5-20 Bouncing trajectory of little commercial landing leg

From the first drop observable on figure 5-20 the system with the little landing leg has bounced. The landing gear brakes after at the second try from only $50cm$ height. Then the big commercial landing leg 5-18b has been tested.

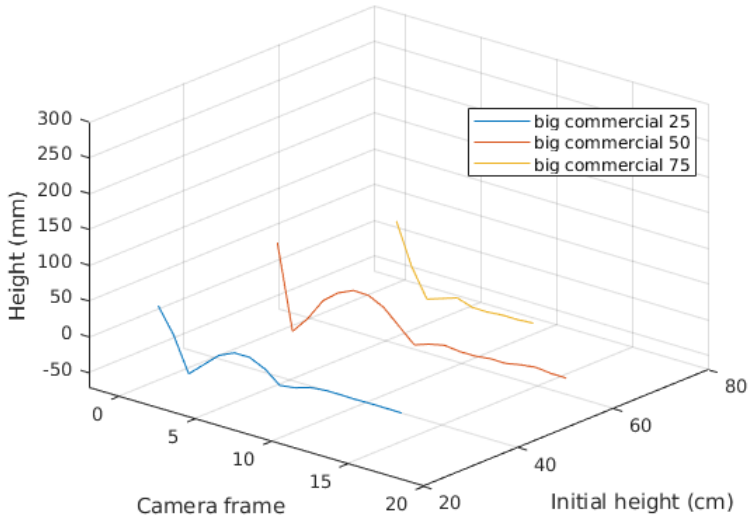


Fig. 5-21 Bouncing trajectory of big commercial landing leg

On the trajectories plotted on figure 5-21, the standard landing leg also had bounced from $25cm$ height. It collapsed at one step further than the previous gears, from $75cm$ height. After experimenting with these two commercial landing gears as a landmark, follows the proposed landing gears. The first is a rigid one with the lowest elastic and frictional coefficients.

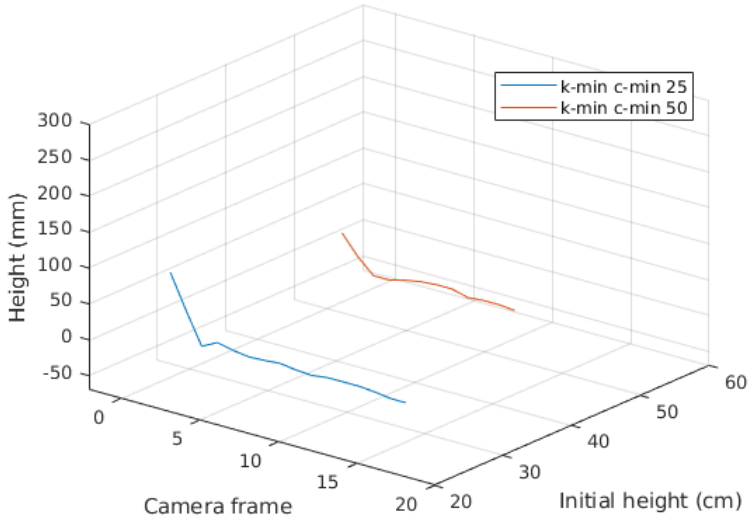


Fig. 5-22 Bouncing trajectory of rigid landing leg with k_{min} and c_{min}

Even if the bounce effect (Fig. 5-22) of this landing gear having a small coefficient k and c has been absorbed from $25cm$ height, the structure cracked at the $50cm$ drop. Keeping a low elasticity k, the friction is maximized for the next measurement. A tighter PP 3D printing o-ring is used and the hinge screws are tightened.

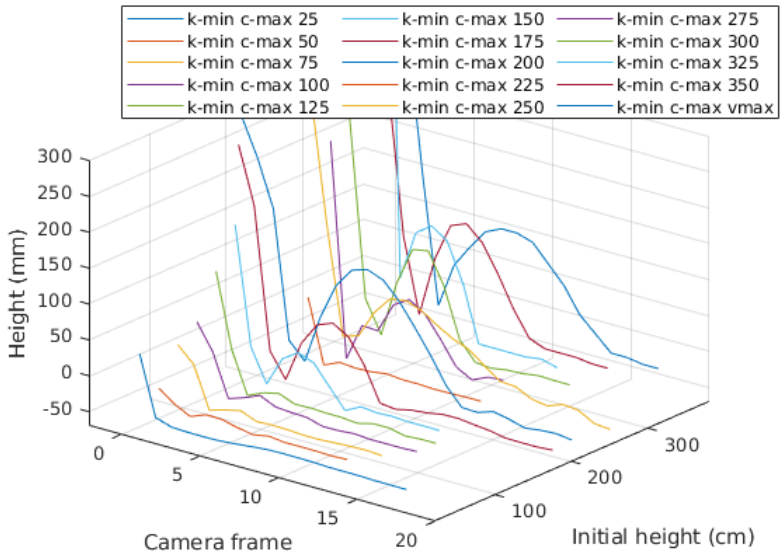


Fig. 5-23 Bouncing trajectory of rigid landing leg with k_{min} and c_{max}

More measurements plotted on figure 5-23 have been done with this configuration of the landing gear. The system did not crack at the maximum $3.5m$ height and has finally been manually projected to the ground. This correspond to the measurement $vmax$ on the graph 5-23. Bounces start at $1m$ with few centimeters amplitude and are mainly due to the frame instead

of the spring of the shock absorber. The trajectory of the system with the hardest spring and minimizing friction follows,

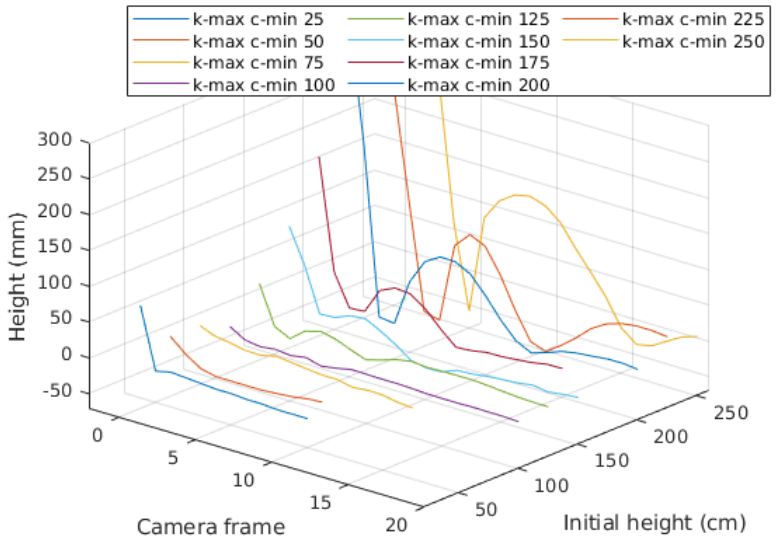


Fig. 5-24 Bouncing trajectory of rigid landing leg with k_{max} and c_{min}

In this configuration, bounce begins at $25cm$ height (Fig. 5-24). Measurement shows a steady increase in the bounce amplitude. The leg collapsed at the manual projection of the system. The final rigid leg configuration used for the experiment follows with the highest friction c and elastic coefficient k of the study.

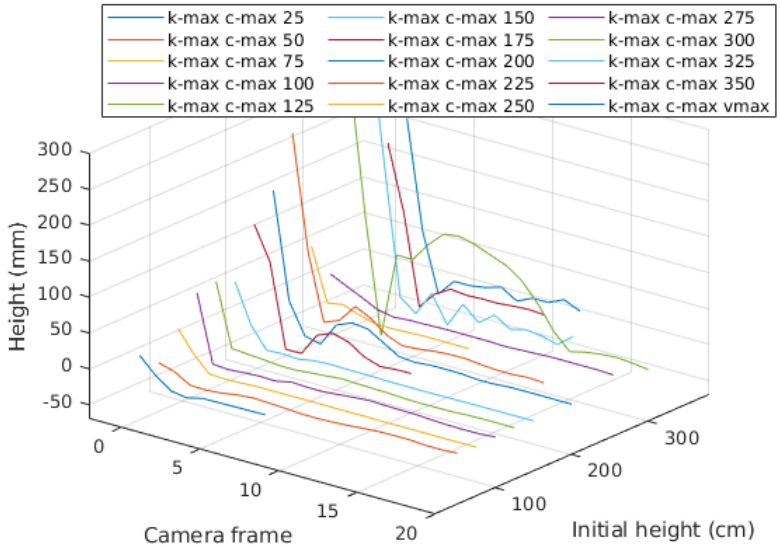


Fig. 5-25 Bouncing trajectory of rigid landing leg with k_{max} and c_{max}

This landing gear has started to bounce at the $75cm$ height at few centimeters (Fig. 5-25). It broke when falling $3.5m$ high. Because the benchmark has friction, some measurement like

the one from $3.5m$ shows a good absorption. This absorption is due to the rails, when the mobile goes up, it has more friction than when it descends. However, this did not prevent the significant rebound from the $2.75m$ height. The bounce experiments terminate with the compliant landing leg,

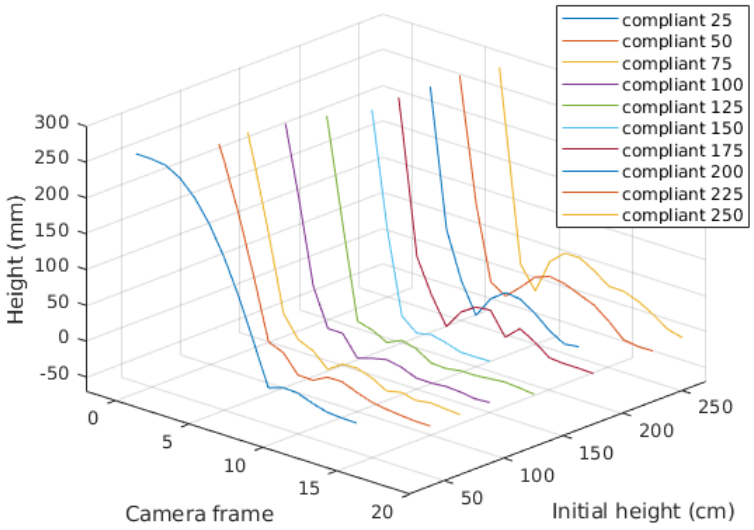


Fig. 5-26 Bouncing trajectory of compliant landing leg

It was noticeable during the experiment that the impact sound of these gears was very soft. It started with a centimeter bounce at the $1m$ drop. The maximum height of dropping has been $2.5m$ and is observable in Fig. 5-26 where the bottom hinge near the fixed frame is torn.

5.4.3 Landing Gear Comparison Study

As a result of the experiment, many measurements have been conducted, the following table 5-1 summarizes important values,

Tab. 5-1 Comparative table of the performances of different landing gears

Landing gear type	**Broken drop height** (cm)	**Max bounce height** (mm)	**Max velocity no break** $(m.s^{-1})$	**Weight** (g)	**Max kinetic energy** (J)	**Max velocity no bounce** $(m.s^{-1})$
Little	50	18	1.478	365	0.399	0
Big	75	34	2.937	369	1.591	0
Rigid k_{min}, c_{min}	50	7	1.175	455	0.314	1.175
Rigid k_{min}, c_{max}	375	119	5.233	455	6.230	2.066
Rigid k_{max}, c_{min}	250	139	4.917	455	5.500	0
Rigid k_{max}, c_{max}	375	115	4.415	455	4.434	0.54
Compliant	250	46	4.350	415	3.926	2.113

Kinetic energy of the table 5-1 is the maximum one measured before the landing leg break.

Commercial landing legs broke before $1m$ of free-fall, as for the landing leg minimizing friction and bound effect. On the other hand, landing gears with at least one coefficient maximized can multiply until almost ten times the height of fall without breaking. The drop exceeded $3.5m$ for a maximized friction coefficient. Bounce was always observable for commercial gears and the compliant gear has the maximum velocity reached without bounce. The maximum amplitude bounce is reached for the rigid leg maximizing k and minimizing friction. Maximum velocity reached before the breakage of the gear and energy kinetic absorbed is ranked similarly to the broken drop height. The weight variation between light commercial landing and the $70g$ rigid has almost no effect on the variation of kinetic energy. The compliant gear has a faster landing velocity, but it broke with a third less kinetic energy than the rigid leg maximizing friction with low elasticity.

During experimentation, all rigid landing gear, including commercial ones, made loud sharp noise, while the PP compliant gear made a little noise.

Some results from the simulation were found in the experiment. A higher bounce of the landing gear with maximum elastic coefficient k and the minimum friction c is evident. Maximizing c and minimizing k supports having the maximum landing speed without bounce, and maximum velocity before landing leg breakage.

5.4.4 Experiment Discussion

As expected with the quantitative study of the simulation, the configuration maximizing the friction coefficient c with a relatively low elastic coefficient k has better results. It creates a drastic increase in speed collision without any bounce, and the bounce notified by this rigid leg

is a result of shock absorption by the body frame.

During experimentation, friction at descent is noticeable for maximum heights. Indeed, in free-fall, the maximum speed is reached.

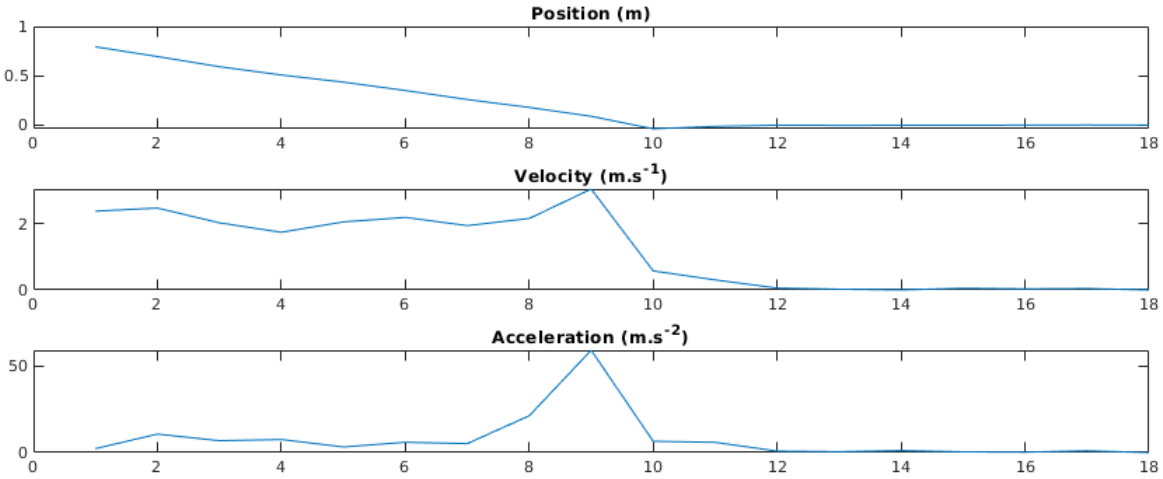


Fig. 5-27 Representation of position, velocity and acceleration of the mobile in free fall from $3.5m$ with k_{max} and c_{max}

The acceleration stop, compensated by the friction, and a maximum speed before the collision is reached. Some bounce was also absorbed by the rail mechanism of the experiment bench. Even with the efforts input to reduce frictions, the altitude of the first bounce should, in fact, have a higher altitude. This effect also explains the differences measured for maximum velocity without breakage of the gear.

Measurements of the falling mobile were done with a camera at 25 FPS. Finer results could have been fined with a higher measurement frequency provided slow motion. This lack of speed sampling frequency is also observable before impact, where only a few points enable the transition in the hybrid system to be seen.

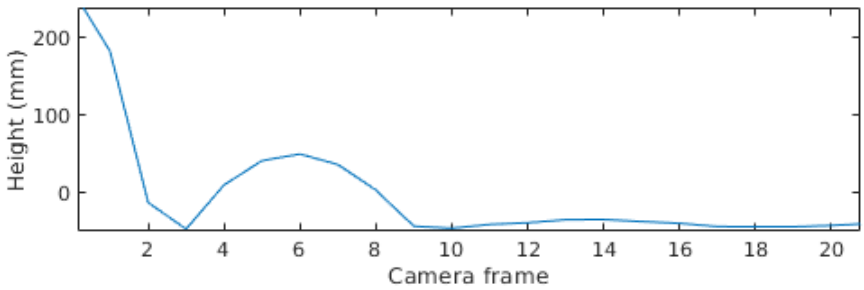


Fig. 5-28 Bounce effect of rigid leg with k_{min} and c_{max} dropped from $175cm$

The transition between free-fall and bounce effect is noticeable on 5-28 is pretty fast. Only

two measurements have been done with the camera. Moreover, measurement has been done for high-velocity collision, so even for small velocity, commercial landing gears starts to bounce.

Some parts of the landing gear have shown fragility that has been fixed. These parts have been reinforced making them larger. Landing gear designs allow the shock absorber to moves on its maximum amplitude. Thus, for high collisions of the frame, the shock absorber is retracted to its minimum amplitude. In this position, effort undergoes by the landing gear is transferred from the ground to the drone structure.

Even if the landing gear weight has almost no effect on the collision, it still affects drone consumption. A dead weight reduce drone fly duration increasing battery consumption. The more the drone will weight to lift, the shorter the amount of time it will have in flight. Therefore, an improvement of the shape must be studied to improve strength distribution at impact and weight must be minimized.

6 Conclusion

6.1 Thesis Summary

This study aimed to present a simple virtual dataset generator using the $Blender$ 3D motor engine. With it a dataset providing an accurate landing pad localization was also generated. This technique permitted after a training step for NN, to have accurate data prediction with low quality images. In the same time, a more traditional localization has been implemented using ArUco and fractal marker. Subsequently, the dataset was performed on a tailored CNN to optimize landing pad localization.

Afterward, visual processing was implemented in a Gazebo simulation. The CNN estimate three time faster the marker position than the standard fiducial pose estimation. The fast landing approach provided accurate results, and the use of the standard marker localization was compared to a multi-scale control. A simple two scale implementation resulted in a more precise outcome in comparison to using a standard marker. The duration of the landing maneuver has been drastically reduced for the same accuracy than standard visual guidance.

Finally, this study analyzed the use of drone landing gears and proved that the landing velocity can thus be safely increased. The landing leg kinematic linkage uses a four-bar theory. For a vertical landing, the contact point also has vertical movement, which is almost completely straight. Transformation of kinetic energy dissipates with friction placed on towards rotating parts. Tight hinges and shock absorbers provide the desired absorption result on a rigid mechanism. While the absorber design has been proposed for a compliant mechanism.

6.2 Discussion and Future Works

Future works will focus on technique improvement and electric rocket implementation. More realistic images can be provided to generate the dataset, which can be done using a more powerful 3D motor engine such as Unreal Engine. The landing pad pattern could be complected using RGB shape and fractal disposition. This kind of complex shape is used on recent motion capture suits. Additionally, the camera lens deformation can be added to the generator and can avoid a step when image processing to increase FPS. A boolean should be added to the dataset, also added some image without the landing pad. This kind of images should help the NN to make the transition between GPS servoing and visual servoing. The estimated position can be done with fusion of visual data with IMU or GPS when it is possible. Resultantly,

the vertical speed can be increased when using other drones. If the vertical speed exceeds the maximum speed that can absorb landing gears, a reduction of speed relative to the height will be implemented. Landing gears should have their weight reduced. Shape of the parts have to be optimized likewise the 3D printing parameters. Real world experiments and adaptations of the work presented on the e-rocket will follow.

Bibliography

[1] W. Kong, D. Zhou, D. Zhang, et al. Vision-based autonomous landing system for unmanned aerial vehicle: A survey[C]//International Conference on Multisensor Fusion and Information Integration for Intelligent Systems (MFI). Beijing, China: IEEE, 2014: 1-8.

[2] A. Borowczyk, A. Nguyen, D. Nguyen, et al. Autonomous Landing of a Multirotor Micro Air Vehicle on a High Velocity Ground Vehicle[J]. IFAC-PapersOnLine, 2016: 2373-2380.

[3] A.Gautam, P.B.Sujit, S. Saripalli. A survey of autonomous landing techniques for UAVs[C]//International Conference on Unmanned Aircraft Systems (ICUAS). Orlando, USA: IEEE, 2014: 1210-1218.

[4] S. Jin, J. Zhang, L. Shen, et al. On-board vision autonomous landing techniques for quadrotor: A survey[C]//35th Chinese Control Conference (CCC). Chengdu, China: IEEE, 2016: 10284-10289.

[5] J. Engel, J. Sturm, D. Cremers. Camera-based navigation of a low-cost quadrocopter[C]//IEEE/RSJ International Conference on Intelligent Robots and Systems. Vilamoura, Portugal: IEEE, 2012: 2815-2821.

[6] S. Saripalli, J. Montgomery, G. Sukhatme. Vision-based autonomous landing of an unmanned aerial vehicle[C]//IEEE International Conference on Robotics and Automation. Washington, USA: IEEE, 2002: 2799-2804.

[7] Y. Feng, C. Zhang, S. B. S. Rawashdeh, et al. Autonomous Landing of a UAV on a Moving Platform Using Model Predictive Control[J]. Drones, 2018: 34.

[8] S. Recker, C. Gribble, M. Butkiewicz. Autonomous Precision Landing for the Joint Tactical Aerial Resupply Vehicle[C]//IEEE Applied Imagery Pattern Recognition Workshop (AIPR). Washington, USA: IEEE, 2018: 1-8.

[9] N. Truong, P. Nguyen, S. Nam, et al. Deep Learning-Based Super-Resolution Reconstruction and Marker Detection for Drone Landing[J]. IEEE Access, 2019, 7: 61639-61655.

[10] D. Lee, T. Ryan, H. Kim. Autonomous landing of a VTOL UAV on a moving platform using image-based visual servoing[C]//IEEE International Conference on Robotics and Automation. Saint-Paul, USA: IEEE, 2012: 971-976.

[11] D. Falanga, A. Zanchettin, A. Simovic, et al. Vision-based autonomous quadrotor landing on a moving platform[C]//IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR). Shanghai, China: IEEE, 2017: 200-207.

[12] J. Macés-Hernández, F. Defaÿ, C. Chauffaut. Autonomous landing of an UAV on a moving platform using model predictive control[C]//11th Asian Control Conference (ASCC). Gold Coast, China: IEEE, 2017: 2298-2303.

[13] Y. Qi, J. Jiang, J. Wu, et al. Autonomous landing solution of low-cost quadrotor on a moving platform[J]. Robotics and Autonomous Systems, 2019, 119: 64-76.

[14] S. Lange, N. Sunderhauf, P. Protzel. A vision based onboard approach for landing and position control of an autonomous multirotor UAV in GPS-denied environments[C]// International Conference on Advanced Robotics. Munich, Germany: IEEE, 2009: 1-6.

[15] Z. Li, Y. Chen, H. Lu, et al. UAV Autonomous Landing Technology Based on AprilTags Vision Positioning Algorithm[C]//Chinese Control Conference (CCC). Guangzhou, China: IEEE, 2019: 8148-8153.

[16] F. Romero-Ramire, R. Muñoz-Salinas, R. Medina-Carnicer. Fractal Markers: A New Approach for Long-Range Marker Pose Estimation Under Occlusion[J]. IEEE Access, 2019, 7: 169908-169919.

[17] Q. C. et al. Efficient Maximum Appearance Search for Large-Scale Object Detection[C]// IEEE Conference on Computer Vision and Pattern Recognition. Portland, USA: IEEE, 2013: 3190-3197.

[18] J. Ming, H. Xian-lin. The Vision Navigation Based On Lunar Surface Control Point Registration[C]//Chinese Control Conference (CCC). Harbin, China: IEEE, 2006: 2236-2239.

[19] Y. Jin, F. He, S. Liu, et al. Small Scale Crater Detection based on Deep Learning with Multi-Temporal Samples of High-Resolution Images[C]//10th International Workshop on the Analysis of Multitemporal Remote Sensing Images (MultiTemp). Shanghai, China: IEEE, 2019: 1-4.

[20] D. DeLatte, S. Crites, N. Guttenberg, et al. Segmentation Convolutional Neural Networks for Automatic Crater Detection on Mars[J]. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 2019, 12(8): 2944-2957.

[21] E. Emami, T. Ahmad, G. Bebis, et al. Crater Detection Using Unsupervised Algorithms and Convolutional Neural Networks[J]. IEEE Transactions on Geoscience and Remote Sensing, 2019, 57(8): 5373-5383.

[22] Y. Yan, D. Qi, C. Li. CVision-based crater and rock detection using a cascade decision forest[J]. IET Computer Vision, 2019, 13(6): 549-555.

[23] A. Krizhevsky, I. Sutskever, G. Hinton. Imagenet classification with deep convolutional neural networks[J]., 2012: 1097-1105.

[24] Y. Jia, E. Shelhamer, J. Donahue, et al. Caffe: Convolutional Architecture for Fast Feature Embedding[Z]. 2014. arXiv: 1408.5093.

[25] J. Akbar, M. Shahzad, M. Malik, et al. Runway Detection and Localization in Aerial Images using Deep Learning[C]//Digital Image Computing: Techniques and Applications (DICTA). Perth, Australia: IEEE, 2019: 1-8.

[26] P. Nguyen, M. Arsalan, J. Koo, et al. LightDenseYOLO: A Fast and Accurate Marker Tracker for Autonomous UAV Landing by Visible Light Camera Sensor on Drone[J]. Sensors, 2018, 18(6): 1703.

[27] D. Safadinho, J. Ramos, R. Ribeiro, et al. UAV Landing Using Computer Vision Techniques for Human Detection[J]. Sensors, 2020, 20(3): 613.

[28] L. Feetham, N. Aouf, O. Dubois-Matra, et al. Image datasets for autonomous planetary landing algorithm development[C]//7th International Conference on Mechanical and Aerospace Engineering (ICMAE). London, UK: IEEE, 2016: 627-637.

[29] J. Delmerico, T. Cieslewski, H. Rebecq, et al. Are We Ready for Autonomous Drone Racing? The UZH-FPV Drone Racing Dataset[C]//International Conference on Robotics and Automation (ICRA). Montreal, Canada: IEEE, 2019: 6713-6719.

[30] A. Loquercio, A. Maqueda, C. del Blanco, et al. Dronet: Learning to fly by driving[J]. IEEE Robotics and Automation Letters, 2018: 1-8.

[31] J. Tobin, R. Fong, A. Ray, et al. Domain randomization for transferring deep neural networks from simulation to the real world[C]//IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Vancouver, Canada: IEEE, 2017: 23-30.

[32] E. Ahmed, M. Moustafa. House price estimation from visual and textual features[J]. ArXiv:1609.08399, 2016: 1-7.

[33] D. A. Lorenz, R. Olds, A. May, et al. Lessons learned from OSIRIS-REx autonomous navigation using natural feature tracking[C]//IEEE Aerospace Conference. Big Sky, USA: IEEE, 2017: 1-12.

[34] H. Moon, Y. Sun, J. Baltes, et al. The IROS 2016 Competitions [Competitions][J]. IEEE Robotics Automation Magazine, 2017, 24(1): 20-29.

[35] B. Morrell, R. Thakker, G. Merewether, et al. Comparison of trajectory optimization algorithms for high-speed quadrotor flight near obstacles[J]. IEEE Robotics and Automation Letters, 2018.

[36] D. Mellinger, N. Michael, V. Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors[J]. International Journal of Robotic Research - IJRR, 2012, 31.

[37] D. Mellinger, V. Kumar. Minimum snap trajectory generation and control for quadrotors[C]//International Conference on Robotics and Automation (ICRA). Shanghai, China: IEEE, 2011.

[38] M. W. Mueller, M. Hehn, R. D'Andrea. A Computationally Efficient Motion Primitive for Quadrocopter Trajectory Generation[J]. IEEE Transactions on Robotics, 2015, 31(6): 1294-1310.

[39] D. Brescianini, M. Hehn, R. D'Andrea. Quadrocopter pole acrobatics[C]//IEEE/RSJ International Conference on Intelligent Robots and Systems. Tokyo, Japan: IEEE, 2013: 3472-3479.

[40] H. Sorenson. Kalman Filtering: Theory and Application[J]. IEEE Press selected reprint series, 1985.

[41] A. Reizenstein. Position and Trajectory Control of a Quadcopter Using PID and LQ Controllers[D]. Linköping University: Electrical Engineering, 2017.

[42] M. Nguyen Duc, T. N. Trong, Y. S. Xuan. The quadrotor MAV system using PID control[C]//IEEE International Conference on Mechatronics and Automation (ICMA). Beijing, China: IEEE, 2015: 506-510.

[43] G. M. Qian, D. Pebrianti, Y. W. Chun, et al. Waypoint navigation of quad-rotor MAV[C]// 7th IEEE International Conference on System Engineering and Technology (ICSET). Beijing, China: IEEE, 2017: 38-42.

[44] E. K. et al. Beauty and the Beast: Optimal Methods Meet Learning for Drone Racing[C]//International Conference on Robotics and Automation (ICRA). Montreal, Canada: IEEE, 2019: 690-696.

[45] I. Siti, M. Mjahed, H. Ayad, et al. New Trajectory Tracking Approach for a Quadcopter Using Genetic Algorithm and Reference Model Methods[J]. Applied Sciences, 2019, 9: 1-21.

[46] D. Falanga, P. Foehn, P. Lu, et al. PAMPC: Perception-Aware Model Predictive Control for Quadrotors[C]//. Mardrid, Spain: IEEE, 2018.

[47] J. Qi, X. Guan, X. Lu. An Autonomous Pose Estimation Method of MAV Based on Monocular Camera and Visual Markers[C]//13th World Congress on Intelligent Control and Automation (WCICA). Changsha, China: IEEE, 2018: 252-257.

[48] C. Sciortino, A. Fagiolini. ROS/Gazebo-Based Simulation of Quadcopter Aircrafts[C]// IEEE 4th International Forum on Research and Technology for Society and Industry (RTSI). Palermo, Italy: IEEE, 2018: 1-6.

[49] R. O. de Santana, L. A. Mozelli, A. A. Neto. Vision-based Autonomous Landing for Micro Aerial Vehicles on Targets Moving in 3D Space[C]//19th International Conference on Advanced Robotics (ICAR). Belo Horizonte, Brazil: IEEE, 2019: 541-546.

[50] J. García, J. M. Molina. Simulation in Real Conditions of Navigation and Obstacle Avoidance with PX4/Gazebo Platform[C]//IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops). Kyoto, Japan: IEEE, 2019: 979-984.

[51] K. Dang Nguyen, T. Nguyen. Vision-Based Software-in-the-Loop-Simulation for Unmanned Aerial Vehicles Using Gazebo and PX4 Open Source[C]//International Conference on System Science and Engineering (ICSSE). Dong Hoi, Vietnam: IEEE, 2019: 429-432.

[52] A. Virtanen. Machine vision based landing zone recognition[D]. Tampere University: Faculty of Information Technology, 2019.

[53] L. Meier, P. Tanskanen, F. Fraundorfer, et al. PIXHAWK: A system for autonomous flight using onboard computer vision[C]//IEEE International Conference on Robotics and Automation. Shanghai, China: IEEE, 2011: 2992-2997.

[54] D. Liang, H. Chai, T. Chen. Landing dynamic analysis for landing leg of lunar lander using Abaqus/Explicit[C]//Proceedings of International Conference on Electronic Mechanical Engineering and Information Technology: vol. 8. Harbin, China: IEEE, 2011: 4364-4367.

[55] S. Seriani. A New Mechanism for Soft Landing in Robotic Space Exploration[J]. MDPI, 2019, 8(4): 103.

[56] J. Ren, J. Wang, X. Liu. Design and Analysis of Terrain-adaptive Bionic Landing Gear System[C]//37th Chinese Control Conference (CCC). Wuhan, China: IEEE, 2018: 5504-5508.

[57] K. Zhang, P. Chermprayong, D. Tzoumanikas, et al. Bioinspired design of a landing system with soft shock absorbers for autonomous aerial robots[J]. Journal of Field Robotics, 2018, 36.

[58] M. Ling, L. Howell, J. Cao, et al. Kinetostatic and Dynamic Modeling of Flexure-Based Compliant Mechanisms: A Survey[J]. Applied Mechanics Reviews, 2020, 72(3).

[59] K. Zhang, Y. Zhu, C. Lou, et al. A Design and Fabrication Approach for Pneumatic Soft Robotic Arms Using 3D Printed Origami Skeletons[C]//2019 2nd IEEE International Conference on Soft Robotics (RoboSoft). Seoul, South Korea: IEEE, 2019: 821-827.

[60] C. Luo, W. Zhao, Z. Du, et al. A Neural Network Based Landing Method for an Unmanned Aerial Vehicle with Soft Landing Gears[J]. Applied Sciences, 2019, 9: 2976.

[61] R. Gonzalez, R. Woods. Digital Image Processing[M]. University of Tennessee, USA: Pearson, 2008.

[62] V. Lepetit, F. Moreno-Noguer, P. Fua. Epnp: An accurate o(n) solution to the pnp problem[J]. International Journal of Computer Vision, 2009, 81(2): 155.

[63] R. Rojas. Weighted Networks –The Perceptron[M]. London, UK: Oxford, 1996: 55-76.

[64] R. Hartley, A. Zisserman. Weighted Networks –The Perceptron[M]. London, UK: Cambridge university press, 2003.

[65] Y. Ma, S. Soatto, J. Kosecka, et al. An Invitation to 3-D Vision: From Images to Geometric Models[M]. 1st ed. New York, USA: SpringerVerlag, 2003.

[66] J. McCarthy, G. Soh. Geometric Design of Linkages[M]. New York, USA: Springer, 2010.

[67] R. Norton. Design of Machinery[M]. New York, USA: McGraw Hill, 2003.

[68] G. Toussaint. Simple Proofs of a Geometric Property of Four-Bar Linkages[J]. The American Mathematical Monthly, 2003, 110: 482-494.

[69] F. Beer, E. Johnston. Vector Mechanics for Engineers[M]. New York, USA: McGraw Hill, 1996: 397.

[70] A. Ruina, R. Pratap. Introduction to Statics and Dynamics[M]. London, UK: Oxford University Press, 2002: 713.

[71] OpenCV, Detection of aruco markers[Z]. How it was published. accessed on 04.06.2020. 2020.

[72] D. Douglas, T. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature[J]., 1973, 10(2): 112-122.

[73] T. Collins, A. Bartoli. Infinitesimal Plane-Based Pose Estimation[J]. International Journal of Computer Vision, 2014: 109.

[74] C. Glines. Top Secret World War II Bat and Bird Bomber Program[J]. Aviation History, 2005, 15(5): 38-44.

[75] R. R. Selvaraju, M. Cogswell, A. Das, et al. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization[C]//IEEE International Conference on Computer Vision (ICCV). Venice, Italy: IEEE, 2017: 618-626.

[76] J.G.Ziegler, N.B.Nichols. Optimum Settings for Automatic Controllers[J]. Journal of Dynamic Systems, Measurement, and Control, 1993, 115(2B): 220-222.

Appendix

A.3 Parameters

A.3.1 Simulink Vertical Fall Simulation

$g = 9.81\ m.s^{-2}$ is gravity acceleration
$v_0 = -10\ m.s^{-2}$ is initial speed, going downward
$y_0 = 0.5\ m$ is initial height
$b = 0.3$ is the drone drag
$m = 1.68\ kg$ is the drone weight
$N = 2\ N$ is the constant drone thrust
$l = 0.15m = m$ is the empty length of the landing leg
k and c are respectively elasticity and damping coefficients,

- k_{high} and c_{high}: $k = 2000$ and $c = 2000$
- k_{low} and c_{high}: $k = 200$ and $c = 500$
- k_{high} and c_{low}: $k = 4000$ and $c = 60$
- k_{low} and c_{low}: $k = 200$ and $c = 120$
- k_{vmax} and c_{vmax}: $k = 1500$ and $c = 170$

A.3.2 Kinematic Proposed Parameters

$AB = 100\ mm$
$BC = 30\ mm$
$CD = 120\ mm$
$AD = 50\ mm$
$BF = 40\ mm$
$CF = 50\ mm$

A.4 Mechanical Linkage

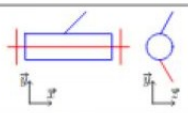
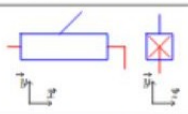
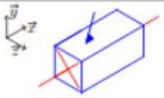
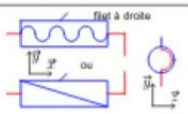
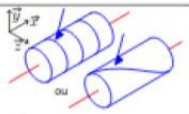
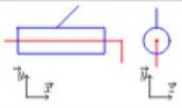
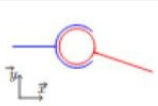
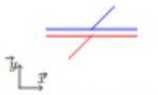
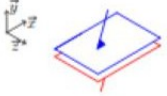
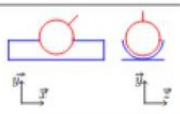
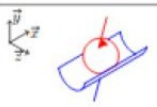
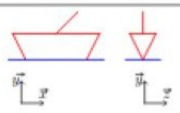
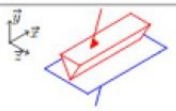
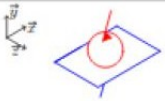
Name	Point of application	DOF	2D representation	3D representation	Kinematic torsor	Torsor of mecha. actions
Built-in	Any point in space	0			$\begin{Bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{Bmatrix}_{\mathcal{R}}$	$\begin{Bmatrix} F_x & M_x \\ F_y & M_y \\ F_z & M_z \end{Bmatrix}_{\mathcal{R}}$
Hinge of axis $(A, \vec{x})$	Any point in space	1			$\begin{Bmatrix} \omega_x & 0 \\ 0 & 0 \\ 0 & 0 \end{Bmatrix}_{\mathcal{R}}$	$\begin{Bmatrix} F_x & 0 \\ F_y & M_y \\ F_z & M_z \end{Bmatrix}_{\mathcal{R}}$
Slide of direction $\vec{x}$	Any point in space	1			$\begin{Bmatrix} 0 & V_x \\ 0 & 0 \\ 0 & 0 \end{Bmatrix}_{\mathcal{R}}$	$\begin{Bmatrix} 0 & M_x \\ F_y & M_y \\ F_z & M_z \end{Bmatrix}_{\mathcal{R}}$
Helical of axis $(A, \vec{x})$	Any point on the axis	1			$\begin{Bmatrix} \omega_x & p.\omega_x \\ 0 & 0 \\ 0 & 0 \end{Bmatrix}_{\mathcal{R}}$	$\begin{Bmatrix} F_x & p.F_x \\ F_y & M_y \\ F_z & M_z \end{Bmatrix}_{\mathcal{R}}$
Sliding hinge of axis $(A, \vec{x})$	Any point on the axis	2			$\begin{Bmatrix} \omega_x & V_x \\ 0 & 0 \\ 0 & 0 \end{Bmatrix}_{\mathcal{R}}$	$\begin{Bmatrix} 0 & 0 \\ F_y & M_y \\ F_z & M_z \end{Bmatrix}_{\mathcal{R}}$
Finger ball joint of center *A* locked on $\vec{x}$	Link center	2			$\begin{Bmatrix} 0 & 0 \\ \omega_y & 0 \\ \omega_z & 0 \end{Bmatrix}_{\mathcal{R}}$	$\begin{Bmatrix} F_x & M_x \\ F_y & 0 \\ F_z & 0 \end{Bmatrix}_{\mathcal{R}}$
Ball joint of center *A*	Link center	3			$\begin{Bmatrix} \omega_x & 0 \\ \omega_y & 0 \\ \omega_z & 0 \end{Bmatrix}_{\mathcal{R}}$	$\begin{Bmatrix} F_x & 0 \\ F_y & 0 \\ F_z & 0 \end{Bmatrix}_{\mathcal{R}}$
Plan support of normal $\vec{y}$	Any point in space	3			$\begin{Bmatrix} 0 & V_x \\ \omega_y & 0 \\ 0 & V_z \end{Bmatrix}_{\mathcal{R}}$	$\begin{Bmatrix} 0 & M_x \\ F_y & 0 \\ 0 & M_z \end{Bmatrix}_{\mathcal{R}}$
Linear annular of axis $(A, \vec{x})$	Link center	4			$\begin{Bmatrix} \omega_x & V_x \\ \omega_y & 0 \\ \omega_z & 0 \end{Bmatrix}_{\mathcal{R}}$	$\begin{Bmatrix} 0 & 0 \\ F_y & 0 \\ F_z & 0 \end{Bmatrix}_{\mathcal{R}}$
Linear rectilinear of axis *(A, x) and normal y*	Any point on the plan $(A, \vec{x}, \vec{y})$	4			$\begin{Bmatrix} \omega_x & V_x \\ \omega_y & 0 \\ 0 & V_z \end{Bmatrix}_{\mathcal{R}}$	$\begin{Bmatrix} 0 & 0 \\ F_y & 0 \\ 0 & M_z \end{Bmatrix}_{\mathcal{R}}$
Punctual in *A* and normal $\vec{y}$	Any point of the normal to contact	5			$\begin{Bmatrix} \omega_x & V_x \\ \omega_y & 0 \\ \omega_z & V_z \end{Bmatrix}_{\mathcal{R}}$	$\begin{Bmatrix} 0 & 0 \\ F_y & 0 \\ 0 & 0 \end{Bmatrix}_{\mathcal{R}}$

Fig. 1-1 Mechanical Normalisation of Linkages

Acknowledgment

I would like to express my sincere gratitude to Prof. Wang Jingyu, School of Astronautics, for giving me the opportunity to work under his supervision. Moreover, I thanks him to providing me continuous guidance, motivation, endless support, assistance, and feedback throughout the entire period of work.

Furthermore I would like to thank the students of my supervisor who were always here to give me support in every stage of my research. I especially thank Mr. Wang Xianyu for his technical support during my research.

I finally thank my family and my friends, especially Mrs. Bastiansz Ashley for the review and the help to improve the reading of my work.

西北工业大学

学位论文知识产权声明书

本人完全了解学校有关保护知识产权的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属于西北工业大学。学校有权保留并向国家有关部门或机构送交论文的复印件和电子版。本人允许论文被查阅和借阅。学校可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。同时本人保证，毕业后结合学位论文研究课题再撰写的文章一律注明作者单位为西北工业大学。

保密论文待解密后适用本声明。

学位论文作者签名：_____　　指导教师签名：_____

年　　月　　日　　年　　月　　日

- -

西北工业大学

学位论文原创性声明

秉承学校严谨的学风和优良的科学道德，本人郑重声明：所呈交的学位论文，是本人在导师的指导下进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容和致谢的地方外，本论文不包含任何其他个人或集体已经公开发表或撰写过的研究成果，不包含本人或其他已申请学位或其他用途使用过的成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式表明。

本人学位论文与资料若有不实，愿意承担一切相关的法律责任。

学位论文作者签名：_____

年　　月　　日