Track Check in Helicopter Main Rotor Blade using Image processing

Submitted by:

Rahul Saini 2020ME10959

Mansi 2020ME21031

B.Tech Mechanical Engineering

Supervised by : Prof. Arpan Gupta

Signature of the supervisor: 27/4/24

Department of Mechanical Engineering

Indian Institute of Technology Delhi

April 2024

Contents

List of Figures

List of Tables

Abbrevations

RTB Rotor Track and Balance

PCR Pitch Control Rods

TAB Trailing Edge Tabs

GUI Graphical User Interface

PDF Portable Document Format

API Application Programming Interface

UI User Interface

UX User Experience

ISO International Organisation for Standardisation

FPS Frames per second

CSRT Channel and Spatial Reliability Tracking

ROI Region of Interest

cv Computer Vision

Units and measurements

mm millimeter

cm centimeter

° degrees

px pixels

hz Hertz

1 Introduction

1.1 Background

Helicopters are widely used in modern aircraft for their ability to hover and perform vertical takeoffs and landings. However, helicopter rides are often uncomfortable due to the significant levels of vibrations they produce. These vibrations can lead to structural issues and even accidents. There are two main types of vibrations in helicopters: vertical and lateral. Vertical vibrations, in particular, are caused by the rotor blades being out of alignment, as illustrated in Figure 1. This type of vibration makes the helicopter bounce up and down during flight. It occurs because one of the blades lifts the helicopter during one part of its rotation but loses lift in the remaining quadrants.

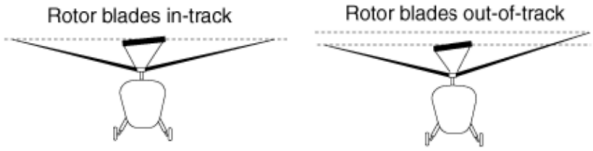


Figure 1: In-track and out of track blades [1]

Blade tracking is a critical process used to manage vertical vibrations. It involves calculating and adjusting the vertical position of each blade's tip based on its position in the air. Typically, one blade is selected as a reference, and the position of the other blades' tips is measured relative to this reference blade. If the difference in blade tip position falls within a certain threshold (usually around 20 mm), the blades are considered to be in track; otherwise, they are deemed out of track or misaligned. This threshold may vary for different helicopter models.

1.2 Problem Definition

Regular maintenance and inspection of helicopter rotor blades are crucial for ensuring safe operations and extending the lifespan of aircraft. Traditional manual methods for track check inspection are time-consuming and prone to human error. In response to this challenge, image processing techniques offer a promising avenue for automating the process and enhancing accuracy.

1.3 Previous progress

In the previous semester, a Python code was developed to analyze videos of rotating helicopters, employing edge and corner detection algorithms to identify blade tip positions and track values. However, the implementation lacked user-friendliness and required manual intervention, limiting its practical utility. Furthermore, the use of standard camera equipment led to some errors in the analysis.

To address these limitations, this project aims to develop a software application that automates the backend Python code, making it more user-friendly and accessible to operators without extensive programming knowledge. Additionally, by utilizing the best available camera technology, the software will be retested on previous videos to address any errors

encountered and undergo real-time testing on helicopters in defense sectors to validate its performance in practical scenarios.

1.4 Project Objectives

1. Develop a user-friendly software application to automate the backend Python code for track check of helicopter main rotor blades.

2. Integrate advanced camera technology to enhance the accuracy and reliability of blade tip position and track value detection.

3. Retest the software on previously analyzed videos to address any errors and improve overall performance.

4. Conduct real-time testing of the software on different models to validate its effectiveness in practical scenarios.

5. Provide a comprehensive documentation and user manual for the software application to facilitate easy adoption and usage by operators.

2 Literature Review

2.1 Rotor Track and Balance (RTB) Techniques (Bechhoefer et al [2])

The study suggests using weights, pitch control rods, and trailing edge tabs for rotor blade adjustments to counter inherent non-uniformities. Ferrer (2001) underscores the linearity of adjustment coefficients, crucial for algorithmic developments in rotor track and balance (RTB). Efforts to minimize blade non-uniformity aim to reduce track split errors, driven by the overarching goal of mitigating vibration in rotor track and balance operations.

2.2 Use of the Hough Transformation To Detect Lines and Curves in Pictures (Duda et al) [3]

Duda's paper introduces Hough transform, converting figure points into a parameter space to identify concurrent lines. Lines are represented parametrically with angle θ and distance ρ, facilitating collinear point detection. Transforming points into sinusoidal curves in parameter plane enhances computational efficiency for detecting collinear points.

Properties of Point-to-Curve Transformation:

Property 1: A point in the picture plane corresponds to a sinusoidal curve in the parameter plane.

Property 2: Designing a system suitable for indoor/outdoor use and in-flight/on-ground operation.

Property 3: Developing a system that requires no modifications to the blades like attaching sensors to blade tips.

Property 4: Creating an affordable system.

2.3 A Novel Shi-Tomasi Corner Detection Algorithm Based on Progressive Probabilistic Hough Transform (Mu et al) [5]

The Moravec algorithm, upon which Shi-Tomasi is built, detects corners by analyzing changes in windowed pixel intensity. However, its lack of rotational invariance limits accuracy when the image is rotated. In contrast, Shi-Tomasi improves corner detection by integrating differential operations and autocorrelation matrices, while also discussing the Hough transform for geometric shape recognition

2.4 Python Tkinter Library Documentation

The research paper delves into the documentation and analysis of Python's Tkinter library for Graphical User Interface (GUI) development, providing valuable insights into its features, advantages, and limitations. Tkinter emerges as a prominent choice for GUI development, owing to its simplicity, cross-platform compatibility, and seamless integration with the Python ecosystem. The research highlights Tkinter's extensive widget library, which includes a variety of built-in components such as buttons, labels, entry fields, and text boxes, facilitating the creation of interactive and user-friendly interfaces. Furthermore, Tkinter's minimal dependencies and lightweight nature make it suitable for rapid prototyping and deployment of GUI applications across different operating systems. However, the research also acknowledges Tkinter's limitations, such as its default appearance not always aligning with the native look and feel of various operating systems, and the absence of some advanced widgets compared to other GUI libraries. Despite these limitations, Tkinter's advantages, including ease of learning, rapid prototyping capabilities, and active community support, position it as a compelling choice for GUI development in Python, catering to a diverse range of development needs and preferences.

2.5 A GUI Based Application for PDF Processing Tools Using Python CustomTkinter [6]

This paper presents a Graphical User Interface (GUI) application for PDF processing tools and file conversion tools. The application provides a user-friendly interface for users to perform various operations on PDF documents, such as splitting, merging, extracting, rotating, and deleting pages. Depending on the user's needs, the user can perform these operations on every page, even pages, odd pages, random specific pages, all pages after some nth page, and between some specific ranges. The application is designed to be user-friendly and simple to use, allowing users to complete tasks quickly and easily. Nowadays, most universities and schools provide their students with all of their course materials online, therefore many students access these resources via PDF files. Since this application operates without an internet connection and is therefore advantageous to all users, Using the PyPDF2 library for PDFs, the Python graphics package Tkinter is used to develop the graphical user interface and UI library CustomTkinter which provides fresh, modern, and completely customizable widgets. Many students upload their files to the internet in order to use these tools, but our offline GUI application provides all of these tools for free. It also includes PDF encryption, decryption, and file conversion tools. These tools are simple to use and can help users save time and effort.9

3 Work Plan and Gant Chart

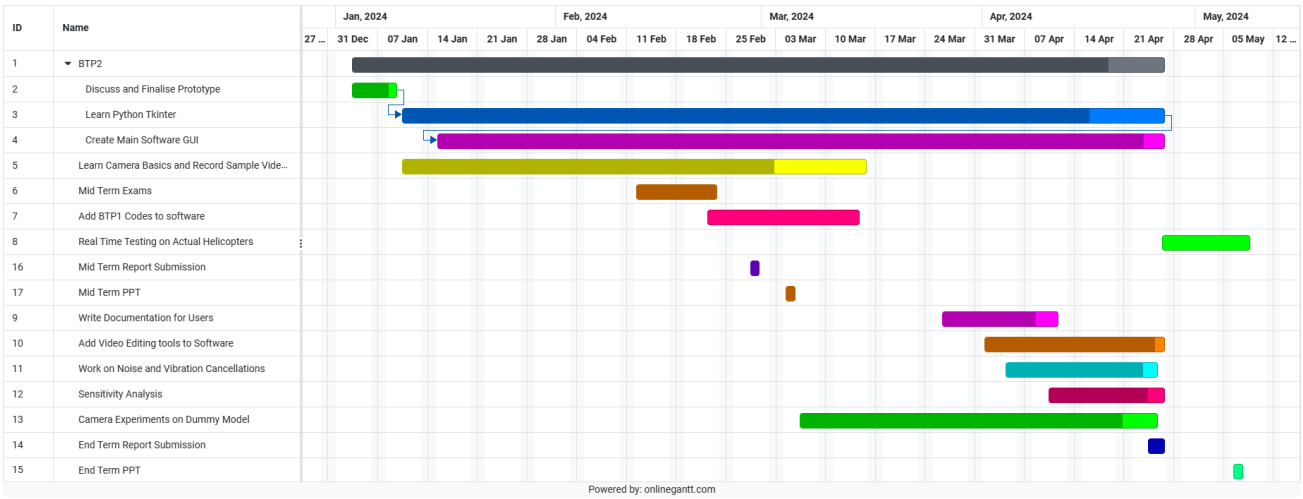


Figure 2: Gantt Chart

4 Work Done

1. Software Prototype: Developed a comprehensive prototype of the software's graphical user interface (GUI) using the **Figma design tool**. The prototype provides a visual representation of how the final product will look and function, serving as a blueprint for further development.

2. Figma Screens: Created multiple screens within Figma to depict various aspects of the software, including the main dashboard, video upload interface, real-time camera module, and inbuilt PDF reader. Each screen is designed with user-friendliness and functionality in mind, ensuring a seamless user experience.

3. Learning Tkinter: Acquired proficiency in **Python's Tkinter library**, essential for creating the graphical user interface (GUI) of the software.

4. Workspace Creation: Established a workspace within the software that enables users to upload videos for analysis, serving as the core functionality of the application.

5. Video Analysis: Implemented the initial framework for video analysis, allowing users to upload videos and initiate automated processing for tasks such as edge detection and corner detection.

6. Video Editing Tools: Developed functionalities to crop and trim videos within the software interface. Users can easily manipulate video content by selecting specific regions for cropping or specifying start and end times for trimming.

7. Run Python Script: Integrated a Python compiler into the software, allowing users to execute Python scripts directly within the application. This feature enables users to automate tasks, perform custom analysis, or implement additional functionalities through Python scripting.

8. Documentary Integration: Integrated an inbuilt PDF reader into the software, enabling users to access and reference relevant documentation directly within the application environment.

9. An extensive amount of research has been done on camera settings to find the best parameter for increasing the accuracy of results being captured.

10. The **Sony RX10 IV camera** was acquired for the purpose of conducting several recordings across different environmental and with varying parameters.

11. Controlling Vibration: Utilized CSRT Tracker method for tracking rotor blade displacement, offering superior accuracy but slower speed and instability when object is lost.

12. Sensitivity Analysis: Conducted on dataset, found high sensitivity to background color with preference for distinct color difference. Also, noted issues with room lighting frequency affecting video analysis.

13. Image Correction Interface: Replaced hard-coding with interactive sliders for image property adjustments, enhancing user-friendliness and adaptability to different videos.

4.1 BladeSense - Our Tkinter Software

BladeSense is a versatile multimedia software designed to simplify video editing and scripting tasks. With a sleek and intuitive interface built using the CustomTkinter library, it offers a range of functionalities such as video trimming, cropping, and script execution. Users can effortlessly upload videos, perform edits like trimming and cropping, and even execute Python scripts within the application. The software provides a seamless experience for both novice and experienced users, with features like customizable appearance modes and UI scaling. Whether you're a content creator, developer, or enthusiast, BladeSense empowers you to unleash your creativity and productivity in video editing and scripting tasks.

4.1.1 Main Workspace

We developed a user interface equipped with various functionalities, including video uploading, real-time video recording, and the ability to save session progress as a historical record. It is created using Tkinter's Frame widget. It serves as the central area where users interact with various features of the software.

4.1.2 Uploading Videos

This feature allows users to upload videos for subsequent processing. Whether it's a homemade clip or a professionally shot video, users can easily import it into the software to begin editing or analyzing. This functionality is implemented using the filedialog module from Tkinter, allowing users to select video files from their system. Once a file is selected, it can be processed further within the application.

```
import tkinter.filedialog

def upload_video():
    file_path = tkinter.filedialog.askopenfilename(filetypes=[("Video
    Files", "*.mp4;*.avi")])
    if file_path:
        # Further processing logic, e.g., displaying the video or storing
    the file path
        print("Video uploaded:", file_path)
```

4.1.3 Cropping Video

With the ability to specify x, y coordinates along with horizontal width and vertical height, users can define a square region of interest (ROI) within the video to be cropped off. This feature is handy for removing unwanted sections or focusing on specific areas of interest.

```
from moviepy.video.io.VideoFileClip import VideoFileClip

def crop_video(video_path, x, y, width, height):
    clip = VideoFileClip(video_path)
    cropped_clip = clip.crop(x1=x, y1=y, x2=x+width, y2=y+height)
    cropped_clip.write_videofile("cropped_video.mp4")  # Save the cropped
    video to a file
    print("Video cropped successfully.")
```

4.1.4 Trimming Video

Users can input start and end times to specify the duration of the desired trimmed video. The software then saves a trimmed version of the original video, containing only the selected portion between the specified start and end times.

```
from moviepy.video.io.VideoFileClip import VideoFileClip

def trim_video(video_path, start_time, end_time):
    clip = VideoFileClip(video_path)
    trimmed_clip = clip.subclip(start_time, end_time)
    trimmed_clip.write_videofile("trimmed_video.mp4")  # Save the trimmed
    video to a file
    print("Video trimmed successfully.")
```

4.1.5 Running Script

This feature allows users to load and execute Python scripts directly within the software. By simply clicking a ”Run” button, users can run custom scripts for various tasks such as data analysis, automation, or any other Python-based operation.

```
def run_script(script_path):
    with open(script_path, 'r') as file:
        script_content = file.read()
    exec(script_content)  # Execute the script
    print("Script executed successfully.")
```

4.1.6 Showing PDF

The software enables users to view PDF documents directly within the application interface. This feature offers convenience and efficiency, allowing users to access documentation, reports, or other important materials without leaving the software environment.

```
import webbrowser

def show_pdf(pdf_path):
    webbrowser.open_new_tab(pdf_path)
```

4.1.7 Adjust Sharpness

Users can adjust the sharpness of videos or images. This feature enhances the clarity and crispness of visuals, making details more defined and pronounced.

```
from PIL import Image, ImageEnhance, ImageFilter

def adjust_sharpness(image_path, factor):
    image = Image.open(image_path)
    enhanced_image = image.filter(ImageFilter.SHARPEN)
    enhanced_image.show()
```

4.1.8 Adjust Brightness

This feature enables users to adjust the brightness of videos or images. By increasing or decreasing brightness levels, users can control the overall illumination of the visual content.

```
from PIL import Image, ImageEnhance, ImageFilter

def adjust_brightness(image_path, factor):
    image = Image.open(image_path)
    enhanced_image = ImageEnhance.Brightness(image).enhance(factor)
    enhanced_image.show()
```

4.1.9 Convert to Grayscale

Users can convert videos or images to grayscale, removing color information and representing the content in shades of gray. This feature offers a monochromatic representation, often used for artistic or analytical purposes.

```
from PIL import Image, ImageEnhance, ImageFilter

def convert_to_grayscale(image_path):
    image = Image.open(image_path)
    grayscale_image = image.convert("L")
    grayscale_image.show()
```

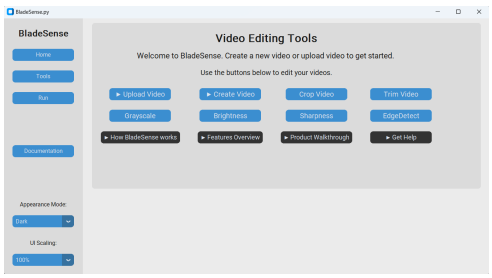
4.1.10 Edge Detection

This feature detects edges within videos, highlighting the boundaries of objects and enhancing their visibility.

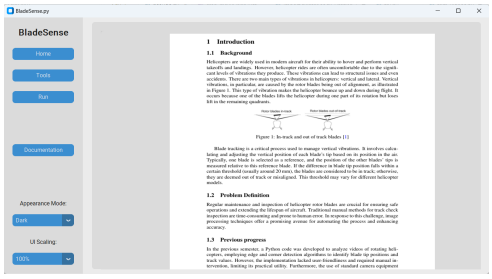
```python
import cv2

def edge_detection(video_path):
        # Define codec and create VideoWriter object for output video
        fourcc = cv2.VideoWriter_fourcc(*"mp4v")
        output_video_path = video_path[:-4] + '_edges.mp4'
        out = cv2.VideoWriter(output_video_path, fourcc, 23.976, (1280,
    720), isColor=False)
```

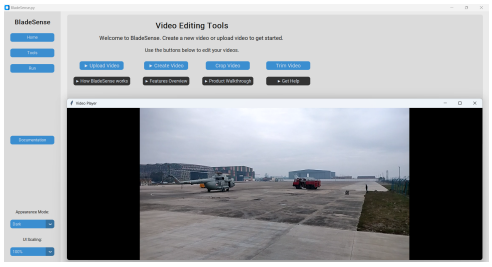
(a) Home Screen



(b) Tools Screen

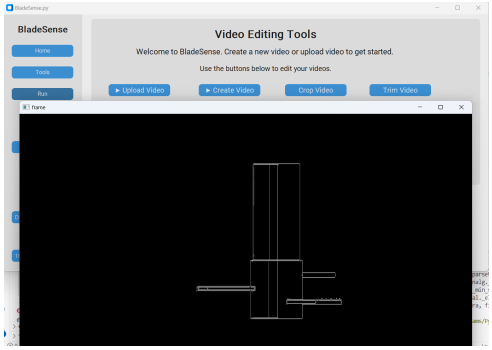


(c) PDF Viewer



(d) Upload Video Screen

(e) Code Execution Screen



(f) EdgeDetect Screen

Figure 3: Screens of GUI using Python Tkinter

4.2 Camera specification

Our research entails recording video footage of rotating helicopter blades and subsequently applying a tailored algorithm to analyze their motion. Our main goal is to track blades precisely without making touch, which means that we have to record the footage at a large distance. We have assessed a number of crucial characteristics that are necessary for this task in our investigation of appropriate cameras, including:

1. Frame rate (FPS): it is the most important parameter for our experimental videos. The term frame-per-second (FPS) describes the number of individual frames or images a camera takes in a single second. It is the industry standard for determining frame rate. More frames per second (FPS) produces smoother, more detailed video footage that gives security personnel important information.

2. ISO: The International Organization for Standardization (ISO) established the ISO Sensitivity standard, which uses a numerical value to indicate sensitivity to light. A higher number denotes more light-capturing capacity and sensitivity.

3. Shutter Speed: The duration of each frame's exposure to light is known as the shutter speed or exposure time. A photographer can get a longer exposure time with a slow shutter speed and a shorter exposure time with a fast shutter speed.

4. Aperture: The hole in the lens called the aperture regulates the amount of light that enters your camera. It is one of the three crucial components of the exposure triangle, along with shutter speed and ISO. The depth of field, or how clear or blurry particular parts are in a picture, is also influenced by your aperture.

5. White Balance: It is a camera setting that establishes the true color of white. This produces a baseline from which all other colors are measured.

(a) (b) (c) (d)

Figure 4: Sony RX10 IV

We require a camera with high frames per second (FPS) and rapid shutter speed to accurately capture the swift rotation of helicopter blades, which rotate at an angular speed of 200 RPM. Moreover, optimal ISO, aperture, and white balance settings are essential for producing high-quality video data, pivotal for the successful application of our algorithm.

4.3 Comparison of Sony RX10 IV with other cameras:

The Sony RX10 IV stands out among its peers in several key aspects. Its ISO range is notably wide, allowing for versatile low-light performance. With an impressive fastest shutter speed and burst rate, it excels in capturing fast-paced action with precision. Additionally, its high-speed video capabilities offer creative flexibility for capturing dynamic scenes. The camera's max aperture further enhances its performance in varying lighting conditions, ensuring sharp and detailed images.

	Sony RX10 IV	Sony RX10 III	Sony RX10 II	Panasonic FZ1000	Panasonic FZ2500
MSRP	$1699	$1499	$1199	$899	$1199
ISO range (native)	100-12800	100-12800	100-12800	125-12800	125-12800
Lens (35mm equivalent)	24-600mm	24-600mm	24-200mm	25-400mm	24-480mm
Max aperture	F2.4-4	F2.4-4	F2.8	F2.8-4	F2.8-4.5
AF system	Phase detect	Contrast detect	Contrast detect	Contrast detect	Contrast detect
AF points	315-point	25-pt	25-pt	49-pt	49-pt
Fastest shutter speed	1/32,000 sec (e-shutter), 1/2000 (mechanical)	1/32,000 sec (e-shutter), 1/2000 (mechanical)	1/32000 sec (e-shutter), 1/2000 (mechanical)	1/16000 sec (e-shutter), 1/4000 (mechanical)	1/16000 sec (e-shutter), 1/4000 (mechanical)
Touchscreen	Yes	No	No	No	Yes
Burst rate	24 fps	14 fps	14 fps	12 fps	12 fps
Video	4K/30p	4K/30p	4K/30p	4K/30p	4K/30p
High-speed video	Up to 960 fps @ 800 x 270	Up to 960 fps @ 800 x 270	Up to 960 fps @ 800 x 270	120 fps @ 1920 x 1080	120 fps @ 1920 x 1080

Figure 5: Comparison between different cameras

4.4 Controlling Vibration:

We are going to record the main rotor blade of the helicopter rotating at a speed of between 200 and 300 revolutions per minute. Vibration from the motor and wind force from the blades are examples of the forces that will be involved. The incorporation of all disturbances was a challenging task, and the failure to take into account all of these disturbances would lead to undesirable outcomes. By measuring the displacement of the track of the blades in relation to some element on the helicopter itself, we were able to discover a novel method for cancelling out all vibrations completely.

4.4.1 Method: [4]

We utilized the "cv2.legacy.TrackerCSRT" method for tracking a distinctive element as a reference. This method leverages the CSRT Tracker, which employs spatial reliability maps to adjust the filter support within the selected region of the frame for tracking. This feature enhances the capability to expand the search area and track non-rectangular objects. The reliability indices indicate the quality of the applied filters per channel and serve as weights for

localization.
Advantages: Compared to previous algorithms, it exhibits superior accuracy and resilience to overlapping objects.
Disadvantages: The speed is notably lower, and the operation becomes unstable when the object is lost.

To be specific we used tracker instead of detection because:
Tracking is faster than detection: Using an object tracker, we specify the bounding box of an object once and track it based on its position, speed, and direction, which is faster than using a pre-trained classifier, which must detect an object at every frame of the video.
Tracking is more stable: Tracking algorithms are more robust to partial occlusion than detection algorithms, which may "lose" the tracked object.
Tracking provides more information: While the detection algorithm cannot track an object's movement path, the tracking algorithm can if we don't care about its class.

4.5 Image correction Interface:

Earlier, we were hard-coding image properties and going through a lot of iterations before we finally arrived at the best possible result. In addition to being less user-friendly, it was time-consuming. The user interface that we have implemented is highly interactive, and it allows us to make adjustments to the image properties by utilizing sliders. As a result, our software became less rigid as a result of the modification of image variables according to the different videos.

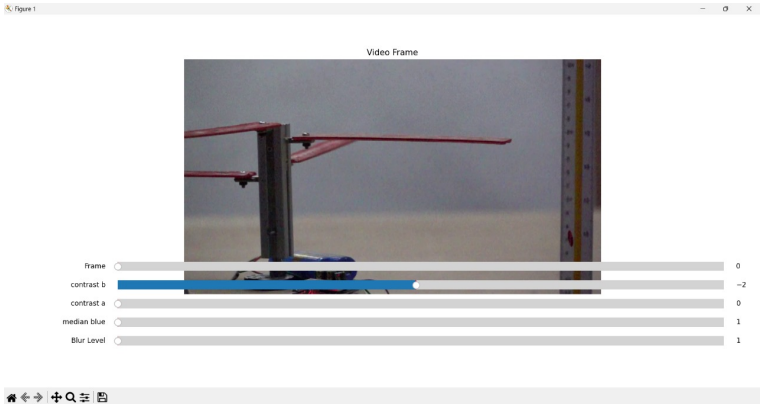


Figure 6: Comparison between different cameras

4.6 Experiments:

4.6.1 Previous experiment results:

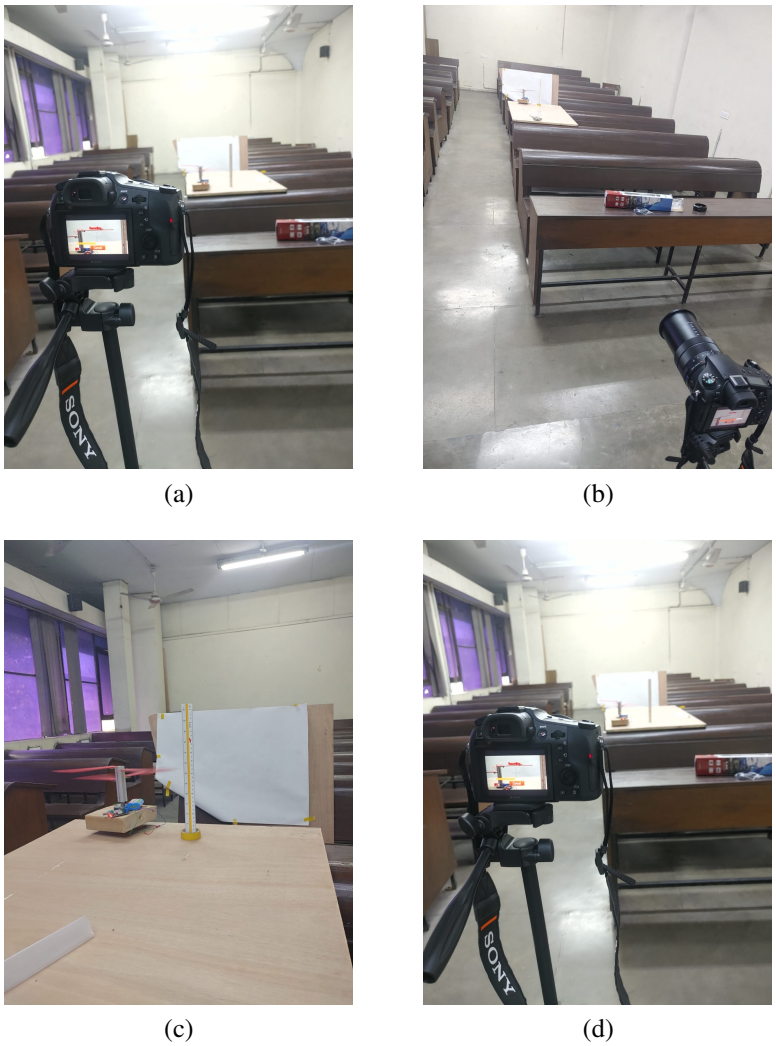
During our earlier work, we reached the results that are presented below. However, in order to achieve better results than these. In spite of the fact that our previous strategy produced some results, we were of the opinion that utilizing tracking algorithms would provide a number of distinct advantages.

Table 1: Errors in Verticle distance of Blades

Experiment (mm)	Maximum Error (%)
solidwork 14mm	1.85 %
solidwork 9mm	1.836 %
dummy model 13mm	22.12 %
dummy model 6mm	21.23 %

4.6.2 Experimental setup:

We recorded our model in closed room with tube light as light source.



(a) (b)

(c) (d)

Figure 7: Experiments performed using Dummy model

4.6.3 Dataset:

The new dataset that we have created includes fifteen video recordings that were captured by a Sony high-speed camera. The video found in this dataset was captured at 250, 500, and 1000 frames per second.

Table 2: Verticle distance of Blades results

Distance from model	Frame rate (fps)	true track value (mm)	measured track value (mm)	error ($\%$)
4m	1000	30	29.30	2.33
-	500	30	25.97	13.43
-	250	30	24.89	17.03
2.5m	1000	40	43.45	8.6
-	500	40	37.64	5.9
-	250	40	36.43	8.92
5m	1000	40	35.56	11.1
-	500	40	42.11	5.27
-	250	40	45.08	12.7

From the above experiments we can conclude that:

1. In comparison to the experimental setting that we used in the past, we have made significant progress in terms of our measurements. The utilization of a camera with a higher frame rate and a higher resolution than the one that was previously used is the cause of this improvement.

2. We can see that we have kept our error to within 15 percent, which is far lower than the 25 percent error that we had in our earlier studies.

3. Taking into consideration the results shown above, we are able to draw the conclusion that the distance between the model and the camera does not significantly affect the results. It solely depends on the resolution of the camera.

4.7 Sensitivity analysis:

Analysis of sensitivity was carried out on the dataset that we had. We took recordings of our model helicopter while it was positioned at varying distances from the camera and against a variety of backgrounds. Videos were recorded in room environment considering tube light.

Our code is highly sensitive towards background of test model. There should be a distinct color difference between model and background because we are detecting position of blade by detecting the color in searching space. If difference is less than algorithm get stuck and may output wrong position of blade. Most appropriate background for our model is white as

blades of red color.

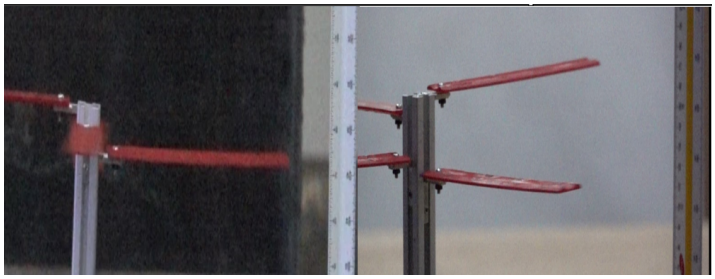


Figure 8: Background Difference

We found out that recording video in room environment is error prone as we are recording at High frame rate and room light frequency is between 30-60 Hz. These frequencies can be visible during video analysis as there occurs light gap at some regular intervals.

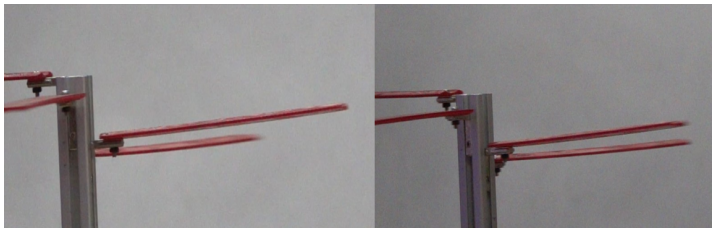


Figure 9: Light Disturbance

Lastly we found out that recording at very frame rate is also not right option. Recording on very high frame rate leaves a very narrow window for other variables but as our projects is based on working on outdoor with minimal specific conditions. High frame rate constrains lighting, vibrations and other parameters.

References

[1] S. Azevedo and T.E. McEwan. Micropower impulse radar. *IEEE Potentials*, 16, 2002.

[2] Eric Bechhoefer, Austin Fang, and Ephrahim Garcia. Rotor track and balance improvements. 2013.

[3] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15, 1972.

[4] Michael. A complete review of the opencv object tracking algorithms. *https://broutonlab.com/blog/opencv-object-tracking/*, 2000.

[5] Zixin Mu and Zifan Li. A novel shi-tomasi corner detection algorithm based on progressive probabilistic hough transform. 2019.

[6] Roshen Tariq Ahmedhamdi, Ayad Al Jubori, and Waleed Ibrahem. Using personal computer for vibration measurements and rotor balancing. *Journal of Engineering/university of Baghdad/issn 1726-4073*, 16:5149–5129, 01 2010.