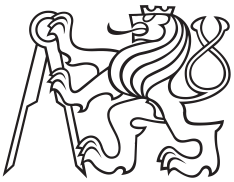
Bachelor Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Computer Science

# Rotation speed estimation from videos

**Denis Gorbunov**

**Supervisor: prof. Ing. Jiří Matas, Ph.D.**
**Field of study: Software**
**May 2024**

# BACHELOR‘S THESIS ASSIGNMENT

## I. Personal and study details

Student's name: **Gorbunov Denis** Personal ID number: **507220**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Computer Science**

Study program: **Open Informatics**

Specialisation: **Software**

## II. Bachelor’s thesis details

Bachelor’s thesis title in English:

**Rotation speed estimation from videos**

Bachelor’s thesis title in Czech:

**Odhad rychlosti otáčení z videí**

Guidelines:

1. Review the literature on rotation speed estimation from videos.
2. Review applications on android and iOS that address related problems (acquisition of past moving objects, rotation estimation)
3. Develop a library for setting android cameras to a state suitable for rotation estimation.
4. Develop a method for detection of a rotating object and for estimation of its angular velocity.
5. Collect a dataset of suitable videos, and obtain the ground truth, e.g. by the method of [4]
6. Evaluate performance of the proposed method.

Bibliography / sources:

1. Computer vision methods for mobile imaging and 3D reconstruction, J Mustaniemi,University of Oulu
2. DeFMO: Deblurring and Shape Recovery of Fast Moving Objects, D Rozumnyi, MR Oswald, V Ferrari, J Matas, M Pollefeys, IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2021)
3. Computer Vision: A Modern Approach 2nd Edition, by David Forsyth (Author), Jean Ponce, 2011
4. Measuring the Speed of Periodic Events using an Event Camera, J. Kolar at al., CVWW 2024

Name and workplace of bachelor’s thesis supervisor:

**prof. Ing. Jiří Matas, Ph.D. Visual Recognition Group, FEE**

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **15.02.2024** Deadline for bachelor thesis submission: **24.05.2024**

Assignment valid until: **21.09.2025**

| prof. Ing. Jiří Matas, Ph.D.<br>Supervisor’s signature | Head of department’s signature | prof. Mgr. Petr Páta, Ph.D.<br>Dean’s signature |
|---|---|---|

## III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

| . | |
|---|---|
| Date of assignment receipt | Student’s signature |

# Acknowledgements

I would like to extend my deepest gratitude to several individuals and groups who have supported me throughout the journey of completing this thesis.

First and foremost, I am profoundly grateful to my family for their unwavering support, encouragement, and love. Their belief in me has been a constant source of motivation.

I would also like to express my sincere thanks to my supervisor, prof. Ing. Jiří Matas, Ph.D., for his invaluable guidance, patience, and insight throughout the research process. His expertise and feedback have been crucial in shaping the direction and quality of this work.

Furthermore, I am thankful to all the members of the Visual Recognition Group for providing a collaborative and stimulating environment. Their input and camaraderie have significantly contributed to the successful completion of this thesis.

Thank you all for your support and encouragement.


# Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses. I also declare that I used the AI tool (ChatGPT) to assist with translating specific sentences from my native language to English and to help rephrase some of these sentences in a style more suitable for academic work.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských prací. Také prohlašuji, že jsem použil AI nástroj (ChatGPT) k pomoci s překladem konkrétních vět z mého rodného jazyka do angličtiny a k přeformulování některých z těchto vět stylem, který je vhodnější pro akademickou práci.

V Praze dne 22. května 2024

.....
Denis Gorbunov

# Abstract

This thesis proposes a novel method for estimating an object's rotational speed from video footage captured by standard smartphone cameras. At a time of widespread smartphone use, our method provides a robust, cost-effective, and easily deployable solution to monitor rotational motion in diverse conditions.

We developed an algorithm that extracts motion information using optical flow and motion tracking followed by robust rotation center estimation. To validate the effectiveness of our approach, we also designed and implemented an Android application that allows users to record videos with different settings and configurations. We conducted extensive experiments using real-world video data captured in a wide range of environments and conditions.

Experimentally, we have obtained that our estimates are within a 2% error margin compared to ground-truth values obtained using the results of measurements with a laser tachometer. This approach demonstrates reliability when certain conditions, for example, carefully adjusted camera settings, sufficient amount of light, and not complete symmetry of the recorded object, are met.



**Keywords:** Visual rotation speed estimation, Optical flow, RAFT

**Supervisor:** prof. Ing. Jiří Matas, Ph.D.



# Abstrakt

V práci navrhujeme novou metodu pro odhad rychlosti rotace objektu z videozáznamů pořízených standardní kamerou chytrého telefonů. V době rozšířeného používání chytrých telefonů poskytuje metoda robustní, levné a snadno nasaditelné řešení pro sledování rotačního pohybu za různých podmínek.

Vyvinuli jsme algoritmus, který extrahuje informace o pohybu pomocí optického toku a sledování pohybu s následným robustním odhadem středu rotace. Pro ověření účinnosti našeho přístupu jsme také navrhli a implementovali aplikaci pro Android, která uživatelům umožňuje nahrávat videa s různými nastaveními a konfiguracemi. Provedli jsme rozsáhlé experimenty s použitím reálných obrazových dat zachycených v široké škále prostředí a podmínek.

Experimentálně jsme zjistili, že naše odhady jsou v mezích 2% chyb v porovnání s reálnými hodnotami získanými pomocí výsledků měření laserovým tachometrem. Tento přístup demonstruje spolehlivost, když jsou splněny určité podmínky, například pečlivě nastavené nastavení kamery, dostatečné množství světla a ne úplná symetrie snímaného objektu.



**Klíčová slova:** Vizuální odhad rychlosti otáčení, Optický tok, RAFT

**Překlad názvu:** Odhad rychlosti otáčení z videí

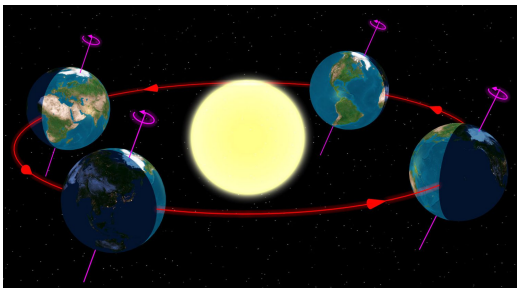
# Contents

# Chapter 1

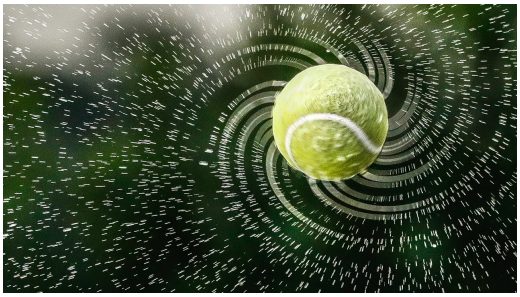
# Introduction

## 1.1 Motivation

Rotation is ubiquitous, found not only in natural phenomena such as the Earth's rotation 1.1a but also in the machinery of various industries 1.1c and even in the athletic pursuits of sports 1.1b. Understanding and precisely measuring rotation are paramount across these domains to ensure efficiency, safety, and performance.



**(a)** The rotation of the Earth [1].



**(b)** The rotation of the tennis ball [2].

**(c)** The rotation of the wankel engine [3].

**Figure 1.1:** Examples of object rotation.

It is logical to quantify rotation, as it provides valuable insights into the dynamics and behavior of rotating systems. Measurement of rotation allows us to monitor speed, detect

irregularities, optimize performance, and even predict potential failures. In sports like baseball, tennis, and golf, the rotation of balls plays a crucial role in determining their trajectory, spin, and ultimately, their accuracy and effectiveness. By precisely measuring the rotation of a baseball pitcher's curveball, a tennis player's topspin serve, or a golfer's backspin on a chip shot, athletes and coaches can gain critical insights into their techniques. Therefore, accurate rotational speed measurement methods are important in many areas of life.

Traditional non-visual methods for measuring rotation often rely on specialized equipment such as encoders, tachometers, or gyroscopes. We provide examples of some of these tools in Fig. 1.2a, 1.2b, 1.2c, 1.2d. Although effective in many scenarios, these methods may have limitations in terms of accuracy, reliability, or adaptability to diverse environments. In addition, they might require complex setups, connecting to other devices, and calibration procedures, making them difficult to use.

Visual techniques for measuring rotation offer a compelling alternative, leveraging advances in imaging technology and computer vision algorithms. Visual methods can provide real-time, non-contact measurement capabilities, offering advantages such as simplicity, versatility, and potentially higher spatial and temporal resolutions. In addition, visual measurements can complement or even surpass the capabilities of traditional sensors, particularly in dynamic or complex environments.

Visual rotation measurement techniques contain various approaches, including optical flow analysis, feature tracking, and motion estimation algorithms. These methods utilize image sequences captured by cameras to track the motion of rotating objects or patterns. By analyzing changes in image features or pixel displacements over time, these techniques can accurately estimate rotational speed and trajectory without the need for direct physical contact or specialized instrumentation.

Rotating objects are among the fastest-moving entities we encounter. Their rapid translation poses challenges for conventional shutter speeds and measurement techniques, leading to phenomena such as motion blur and distortion. Advanced image processing algorithms [8] [9] can be employed to mitigate these phenomena, allowing for a more accurate measurement of rotational speed and trajectory. However, with well-configured cameras, processing such high-speed motion becomes significantly more manageable. That is why it is very important to have an application in which we can adjust the camera settings according to the need in a certain situation.

## 1.2 Problem formulation

Rotation, at its core, refers to the circular movement of an object around a center or an axis. There are several types of rotation:

- **Pure rotation:** This occurs when all points of an object move in circular paths around a fixed axis, maintaining a constant distance from the axis. A classic example is the spinning of a wheel.

- **Orbital rotation:** In this type, an object moves around a point outside of itself, such as the Earth orbiting the Sun. Here, the axis of rotation is external to the object.
- **Combined rotation and translation:** This involves both rotational and translational motion. A common example is a rolling ball that rotates around its own axis while also moving forward along a surface.

The key parameters involved in describing rotational motion include the angle of rotation ($\alpha$), the axis of rotation, and the angular velocity ($\omega$).

Additionally, rotation can be classified into 2D and 3D rotations:

- **2D rotation:** This occurs in a two-dimensional plane, where the object rotates around a point. The rotation is described by a single angle, $\alpha$. An example is a spinning disc on a flat surface.
- **3D rotation:** This involves rotation in three-dimensional space and requires an axis of rotation and an angle of rotation. The rotation can be described using Euler angles, quaternions, or rotation matrices. An example is the rotation of the Earth around its axis.

Smartphones have become ubiquitous in modern society, with their built-in cameras serving as versatile tools to capture various types of motion. Among the applications of smartphone cameras, there lies the potential for accurately measuring rotational speed. Whether it is analyzing sports performances, assessing mechanical systems, or simply tracking the motion of everyday objects, the ability to measure rotational speed using smartphone cameras offers numerous opportunities for innovation and practical use.

However, accurate measurement of rotation speed using smartphone cameras is fraught with a number of difficulties. The main one is that the existing methods involve the use of videos that are shot on cameras with great capabilities, that is, they imply receiving video in good quality. This is a problem because smartphones use cameras with fairly limited capabilities. This, in turn, leads to problems such as image blurring when objects move quickly or limited frame rates. Because of this, existing algorithms make a large error in calculations and become less effective.

The main purpose of this work is to develop a new method for measuring the rotation speed of objects using smartphone cameras and to confirm the proposed method experimentally in various environmental conditions and use cases. In this paper, we focus on measuring the rotation speed for such a rotation which is Pure and 2D at the same time. In addition, this work aims to develop an application to set android cameras to a state suitable for rotation estimation.

To determine the effectiveness of the algorithm, we calculate the relative error it made compared to the ground truth value using the following formula:

$$\text{Relative Error} = \left| \frac{M - GT}{GT} \right| \tag{1.1}$$

where $M$ is a the measured value of the rotational speed of the captured object and $GT$ is a given ground truth value.

## 1.3 Structure of the thesis

This thesis is structured to provide a comprehensive investigation into the measurement of rotational speed using smartphone cameras.

The **Introduction** chapter sets the stage by explaining the importance of the research problem and defining its objectives. This gives an idea of the motivation of the research, emphasizing its relevance in various fields, and also talks about articles already written on this topic, as well as about existing applications for measuring the rotational speed of the objects. In addition, it describes the general structure of the thesis, which gives the reader an idea of the following chapters.

The **Camera Control for Android Phone** chapter delves into the technical aspects of camera control on the Android platform. It discusses the selection of programming language and APIs, offering a detailed account of the development process for the camera application. This includes insights into platform compatibility, language syntax, and integration of application programming interfaces.

The **Proposed method** chapter describes in detail the method of measuring the rotation speed of an object recorded on video using a phone camera. Every aspect of the proposed method is described in detail, from explaining the input data type to calculating the rotation speed.

The **Experiments** chapter details the experimental setup and methodology used to validate the proposed method. Provides a comprehensive analysis of the results obtained from various experimental scenarios, including different shooting angles and parameter settings. Through rigorous experimentation and analysis, the effectiveness and robustness of the proposed method are thoroughly evaluated.

Finally, the **Conclusion** chapter synthesizes the key findings of the research and elucidates their implications. It offers reflections on the strengths and limitations of the proposed method and provides recommendations for future research directions.

## 1.4 Related work

In this section, we discuss existing approaches for estimating the rotational speed of objects. This part of the article shows how extensive this area is.

The first subsection covers non-camera methods that use mechanical, optical, or magnetic devices. It also describes the structure of these devices and their advantages and disadvantages.

The second subsection mentions existing methods based on event camera.

Subsequently, we discuss existing low-cost camera methods and what they offer us.

In the last subsection, we discuss existing applications for measuring the rotational speed of the objects.

### 1.4.1 Non-camera Rotation Speed Measuring Devices

There are many different commercially available devices that offer different ways of measuring rotational speed. All such devices can be divided into contact, which require direct contact with measured object (*e.g.* contact tachometer, wheel encoders, contact proximity sensors) and contactless, which do not require any contact (*e.g.* stroboscope, laser tachometer, optical encoder, magnetic encoder, acoustic sensors, vibration analysis sensor).

A contact tachometer typically consists of a sensor, such as a wheel or probe, that comes into contact with the surface of the rotating object. As the object rotates, the sensor detects the movement and converts it into speed measurements. This method is effective in measuring the speed of objects with a suitable physical surface for contact, but may cause wear or damage to delicate or high-speed rotating surfaces. Furthermore, direct physical connection introduces inaccuracies due to external factors such as surface texture, irregularities, or slippage between the sensor and the rotating object.

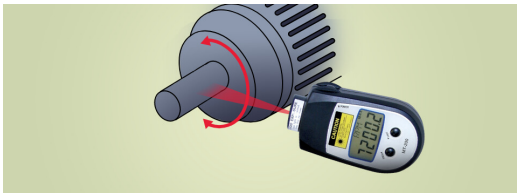
The wheel encoder is made up of two parts: a Hall Effect sensor measuring the intensity of a magnetic field and a ring magnet (similar to a metal washer) connected to the motor shaft, as shown in Figure 1.2c. When the motor turns the wheel, it also spins the ring magnet. The Hall effect sensor located near the ring measures changes in the magnetic field as it revolves.

A laser tachometer directs a laser beam toward an object that spins and detects the frequency change of the reflected beam. This shift corresponds to the object's rotational speed, making it possible to measure speed without touching it.

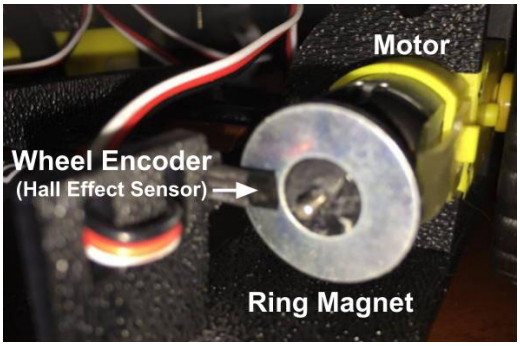
Optical encoders are typically made up of a disk with alternating transparent and opaque segments, sensors (such photodiodes or phototransistors), and a light source (commonly an LED) as shown in Figure 1.2d. As the disc spins, the sensors detect variations in light intensity induced by the passing segments. This pattern is converted into electrical impulses, which provide information on the spinning speed and direction. Because optical encoders might be light sensitive, they need to be properly shielded or filtered to stop outside light sources from interfering with their operation.

The way a stroboscope works is by periodically emitting short flashes of light. These pulses are connected to the rotation of rotating objects, giving the feeling of delayed or frozen motion. Adjust the frequency of the light pulses to match the rotational speed of the object.

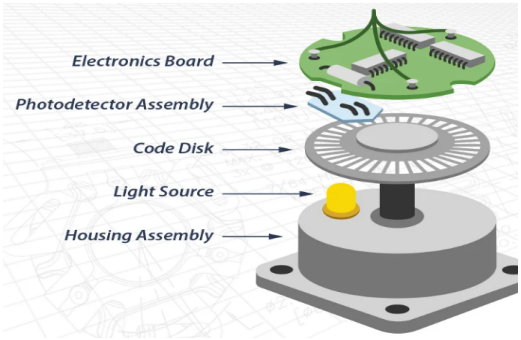
**(a)** Example of a contact tachometer in operation [4].



**(b)** Example of a non-contact tachometer in operation [5].




**(c)** Example of a wheel encoder in operation [10].




**(d)** Example of an optical encoder in operation [7].

**Figure 1.2:** Examples of two types of encoders and two type of tachometers.

### 1.4.2 Event-based Rotation Speed Measurement Methods

Hylton [11] introduces one of the first sets of experiments with an event camera and rapidly moving objects. These experiments show some strengths and weaknesses of the proposed method. In this work he is trying to measure the rotational speed of the cutter unit using an event camera. He compares the results obtained with the results obtained by more reliable mechanical tools, including accelerometer and acoustic. But the algorithm's architecture lacks the complexity needed to handle the noisy, nonstructural event stream and produce precise measurements of high-speed spinning.

EV-Tach method [12] is one of the best methods for estimation of rotational speed by event camera. This algorithm achieves 0.3 ‰ as the Relative Mean Absolute Error (RMAE). An initial centroids selection method based on heat maps is proposed for EVTach to enhance the robustness of clustering. It starts with K-means clustering to extract multiple rotating targets. Once the angle of rotation has been determined, the rotational speed can be calculated utilizing a coarse-to-fine ICP-based event stream registration method. The results of this algorithm are comparable to the results of the work of the laser tachometer in fixed deployment.

J. Kolar [13] uses his method to estimate Periodic Phenomena Properties. In his approach he first collects event camera output along the time axis, computes a 2D correlation between the aggregated data and a chosen template in a chosen region of interest, and then outputs the average time interval between correlation response peaks.

Event-based rotation speed measurement methods have their advantages, such as high temporal resolution or enabling precise tracking of rapid rotational motion. Nevertheless,

these methods may have a problem in situations when the rotating object has a solid and uniform structure without any landmarks or markers.

### 1.4.3 Camera-based Rotation Speed Measurement Methods

In comparison with conventional techniques for rotational speed measurement, the imaging based method has the advantages of remote measurement, simple installation, high robustness and wide applicability. The imaging device does not require physical access to the object to be measured and can be set up at a remote location even hundreds of meters away (when a zoom lens is used).

Wang et al. [14] offer the first method for assessing rotational speed using a low-cost camera device. This approach requires a simple marker on the rotor. For speed estimation, this method uses structural similarity and two-dimensional correlation algorithms. Experimental results suggest that the system is capable of providing the maximum relative error of $\pm 1\%$ with normalized standard deviation less than 0.8% over a speed range of 0 to 700 RPM.

An alternative approach [15] is the approach based on the previous one [14]. This paper focuses mainly on quantifying the effects of various key factors of the imaging system on the accuracy and reliability of the rotational speed measurement and thus identifying the optimal design parameters of the system. Also, they try Visual information fidelity algorithm for assessing of the rotational speed. Experimental results suggest that the system is capable of providing constant rotational speed measurement with a maximum relative error of $\pm 0.6\%$ and a repeatability of less than 0.6% over a speed range of 100 to 900 RPM. Under varying speed conditions, the proposed system can achieve valid measurement with a relative error within $\pm 1\%$ over the speed range of 300 to 900 RPM.

Toby Verwimp et al. [16] investigate the viability of using a smartphone as a cost-effective tool for rotational speed extraction. The proposed method exploits the geometrical image deformations induced by a smartphone's rolling shutter camera, which reads pixel lines sequentially with a delay equal to the rolling shutter period. A method is proposed to measure the speed from the deformation of a zebra pattern as a consequence of the sequential readout of a rolling shutter camera. Normalized RootMean-Square Errors (NRMSE) of about 3% (or less) are reached for the smartphone aligned with the shaft with peaks in the percentage error of about 5% to 8%. Nevertheless, even for a misalignment of 60°, the NRMSE is only 4% with variations in the percentage error to about 10%.

Hui Zhao et al. [17] propose a novel extraction and filter algorithm BSCZT-KF to obtain the accurate roll angular rate of a high spinning projectile, which in the future can be directly used to calculate the rotation speed of objects. They convert the roll angular rate measuring issue into an issue involving frequency estimate. Subsequently, the enhanced CZT technique, known as BSCZT, is utilized to provide a precise narrowband signal frequency estimation. The BSCZT-KF technique is provided to significantly improve the frequency estimation accuracy and the real-time performance when combined with the peak detection approach. The test findings indicate that the average roll angular rate inaccuracy is approximately 0.095% of the maximum roll angular rate.

### 1.4.4 Existing applications for measuring the rotational speed of the objects

There are also a number of applications for iOS and android that can be used to measure the speed of some rotating objects. However, it should be noted right away that all these applications do not use cameras, and therefore the rotation speed in these applications is not measured visually.

**RPM Meter - Strobe Light (Android)**. This app is used for measuring the rotational speed of different machines. This application initially necessitates marking the edge of the machine for which the rotational speed is sought. It's imperative to ensure the machine is placed in a suitable environment with adequate darkness. Following this, the app should be launched, and an estimated RPM value entered into the input field. Then, the flashlight should be directed towards the rotating machine. The RPM value can be adjusted using the slider bar until the marked point on the machine appears stationary. Finally, the displayed RPM value from the input field corresponds to the rotational speed of the machine.

**Stope tachometer (IOS)**. Similar to the RPM Meter - Strobe Light for Android, this app uses the device's flashlight to create a stroboscopic effect and measure the rotational speed of objects.

**RPM Meter: For Rig Compressor (Android)**. To utilize the RPM Meter: For Rig Compressor, one starts by positioning the device near the rig compressor's rotating components. The app relies on sound measurements to determine the compressor's rotational speed. When ready, the user activates the app and ensures that the device's microphone is enabled to capture sound accurately. Upon launching the app, it begins recording the sound emitted by the rig compressor during operation. This sound contains distinct frequency components that correspond to the compressor's rotational speed. The app then analyzes the recorded audio data, identifying specific frequency peaks or patterns indicative of the compressor's speed. Using established algorithms, it calculates the rotational speed based on these frequency components. The calculated RPM value is then prominently displayed within the app's interface, providing real-time feedback on the compressor's rotational speed. Users can monitor this value continuously to ensure optimal compressor performance.

**RPM Speed and Wow (Android)**. When users launch the "RPM Speed and Wow" app, it immediately accesses the phone's internal sensors, including the accelerometer, gyroscope, and magnetometer. These sensors provide crucial data regarding the phone's orientation and movement. Continuously reading data from these sensors, the app tracks changes in the phone's orientation across three-dimensional space. The accelerometer captures linear acceleration along the x, y, and z axes, while the gyroscope measures angular velocity around these axes. Additionally, the magnetometer contributes information about the phone's orientation relative to the Earth's magnetic field. Utilizing this sensor data, the "RPM Speed and Wow" app computes the phone's orientation and its rotation speed in real-time.

**Turntable Speed**. This application is similar to the RPM Speed and Wow application on Android. It has the same measurement principle based on calculating the change in the physical position of the phone.

## 1.5 Contributions

The contributions of this thesis are as follows:

- An application for Android phones, for recording videos with settings suitable for the certain situation. The presence of such an application allowed us to use a large number of all the features of the phone if necessary.
- A rotational speed estimation method. Unlike the methods that were proposed earlier, this method is effective, robust and accurate even for videos that were shot on a simple phone camera.
- This thesis lays the foundation for further research into the capabilities of a phone camera in the context of rotating objects, in order to further improve the cost-effectiveness and portability of the imaging method for rotational speed measurement. In particular, this is facilitated by the video dataset DGBP of rotating objects that we have collected and that is described in Appendix A.

# Chapter 2

# Camera control for Android phone

As mentioned above in Section 1.1, it is very important to have a well-tuned camera to record rotating objects, as this helps to avoid various phenomena.

Most of the phone applications that work with its camera may have a number of limitations. For example, in none of them can we set up video recording so that every 10 frames the shutter speed is reduced by 10 milliseconds. The presence of such limitations may prevent us from using all possible camera settings to select those in which our algorithm will show the best efficiency.

In this regard, we have made our own video recording application. Having a custom application allows for the addition of specific camera modes or settings required at different stages of algorithm development or experimental procedures.

In this chapter, we describe the main properties of this application, explain how to use it, and also give examples of frames from videos recorded with different parameter settings.

## 2.1 Platform

The choice of the platform for which the application will be written was made in favor of Android for the following reasons:

- Android continues to reign in the global market with a 69.88% market share in the most recent quarter of 2024, while iOS has 29.39% of the mobile operating system users [18].
- Android phone was more accessible for development.

## 2.2 Programming language

There were two main options: **Java *and* Kotlin** for choosing the language for implementing the application on the Android system. After studying various materials, it became known that Kotlin is Google's preferred language for Android development [19]. With its many benefits over Java, Kotlin is a popular choice for software projects such as Android development. The following are some of the main benefits of Kotlin over Java:

- **Null Safety:** Null pointer exceptions be avoided at build time by using Kotlin's type system. So, there is a decreased chance of run-time problems.
- **Interoperability with Java**
- **Conciseness:** Java requires more boilerplate code than Kotlin does. Because of its many capabilities, developers may express the same functionality in fewer lines of code, which increases readability and productivity.

## 2.3 API used

Mastering hardware from scratch for subsequent manipulations with it is quite difficult and does not make much sense for the purposes of this project because there are already written and described APIs which Kotlin supports. There was a choice between three possible APIs for connecting to the device camera and its subsequent use and configuration:

- **Camera intents API** - to perform basic camera actions like capturing a photo or video using the device's default camera application. [20]
- **CameraX** - most recommended option. It is a Jetpack library that supports the vast majority of Android devices and provides a consistent high-level API designed around common use cases. CameraX resolves device compatibility issues for you so that you don't have to add device-specific code to your application. CameraX is built on top of the Camera2 API package. [21]
- **Camera2 API** - the best option if the low-level camera control is needed. It supports complex use cases. Camera2 is a good option, but the API is more complex that CameraX and requires to manage device-specific configurations. [22]

For subsequent work with the application, we chose to use the **Camera2 API** because this API gives full control over the phone's camera and directly allows us to adjust and set parameters for this camera, such as ISO, shutter speed, frame rate, focus.

## 2.4 Camera application

The application weighs 37.52 Mb. It can be installed on every Android-based gadget that has an Android version higher than or at least 8.1. The application requires simple rights, so it can save captured videos to the gallery.

It uses a rear camera. In case if the device has more than one rear camera it chooses the Main camera - this is often referred to simply as the "main" or "primary" camera. It is typically the camera with the highest resolution and is designed for general-purpose photography.

The captured videos are saved to the *DCIM* directory on the phone. If this directory does not exist, it will be created. In order for each recorded video to have a unique name, it was decided to name them as follows: **"VIDEO_yyyyMMdd_HHmmss.mp4"**.

The imaging device offers a range of video formats with different resolutions. In general, images with a higher resolution provide more accurate rotational speed measurement. However, such images result in longer computational time and hence slower system response due to the increasing volume of data to process. In order to strike a balance between the measurement accuracy and the response time of the system, the video is saved in *MP4* format with *'1080×1920'* resolution.

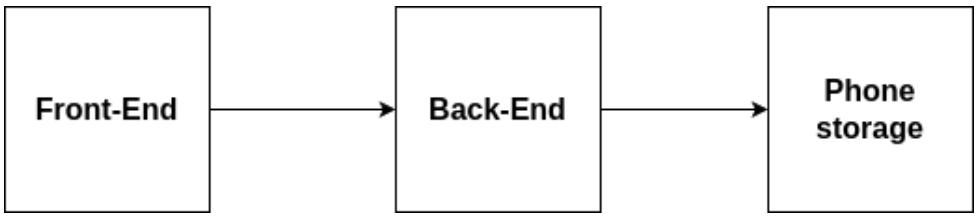



**Figure 2.1:** Component diagram.

From the point of view of the software, the application consists of three parts (Fig. 2.1): the front-end, the back-end, and the phone storage.

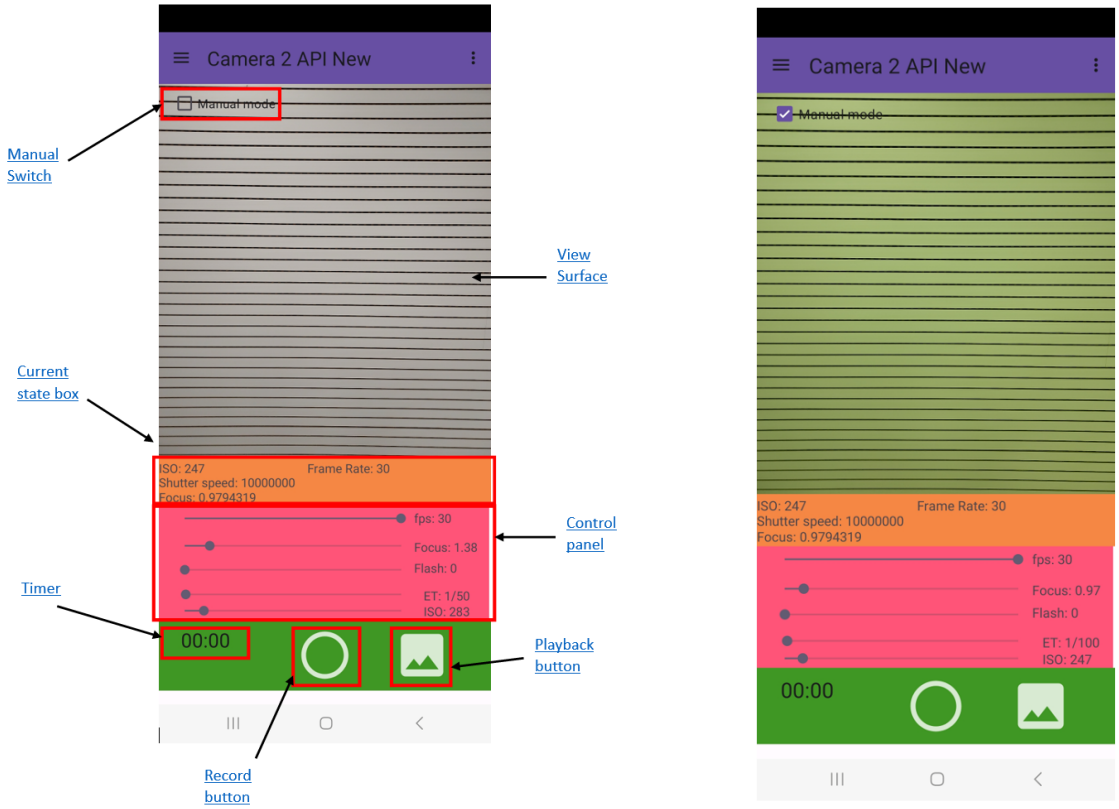
On the front-end, the user switches between automatic and manual mode, sets the shooting parameters, and starts or finishes it. The front-end also displays the camera parameters that the camera has in automatic mode.

On the backend, the phone is connected to the camera and its subsequent configuration, depending on the parameters entered by the user.

After the end of recording, the captured video is saved in the phone's storage.

### 2.4.1 UI and application manual

In this part, we describe how our application works. In particular, we explain what each button or field means.




**(a)** Application display in auto mode

**(b)** Application display in manual mode

**Figure 2.2:** Examples of screens when in different mods.

**Record button:** The record session starts/finishes by clicking on this button depends on the state of the process. (Figure 2.2a)

**Timer:** Shows the duration of the captured video. It starts after the video session has started and stops after the video session has finished. (Figure 2.2a)

**View Surface:** Here the camera preview is rendered. (Figure 2.2a)

**Playback button:** After taking the video, we can immediately watch it in the application using this button. (Figure 2.2a)

**Manual switch:** The application has two different modes: auto mode and manual mode. In auto mode, all camera settings are controlled by the camera itself. Change them depending on the outside conditions. In manual mode, we can adjust the camera settings provided by ourselves depending on our preferences. Changing mode is available during both the preview and the video session. As soon as we switch to manual mode, all settings are fixed to the values they had right before switching. The list of parameters that we can adjust is ISO, shutter speed, focus, and frame rate. (Figure 2.2a)

**Current state box:** Here we can see all parameters that we can adjust and their current values (Shutter speed given in nanoseconds). (Figure 2.2a)

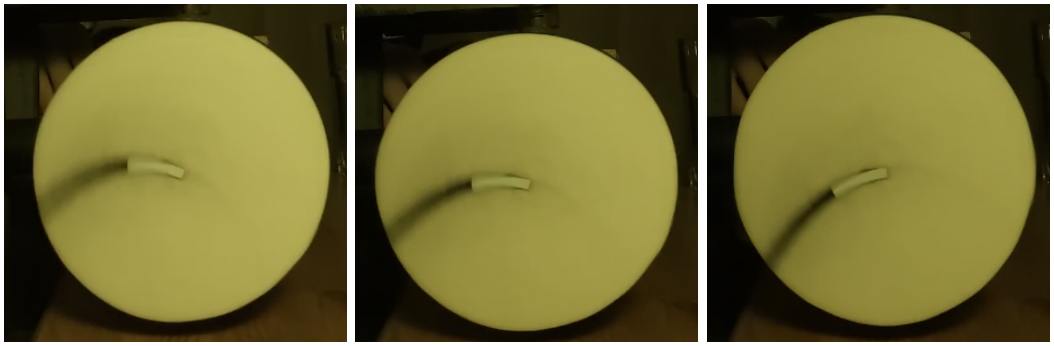
**Control panel:** This is the box with four seek bars to adjust such parameters as frame rate, focus, shutter speed given in seconds and ISO. The range of values that these parameters can take are adjusted depending on the device. The boundaries of these intervals correspond to the minimum and maximum possible values on the current device. We can also change all these values in automatic mode, but it will not be visible because the camera will instantly change them. After switching to manual mode, the values in these panels will automatically set to the corresponding last available values in automatic mode, as can be seen in Figure 2.2b.

The color of the surface was changed (Figure 2.2b) because the default value was automatically set for the white balance setting. Since white balance has no direct impact on the strength or intensity of the light situation, it was determined not to offer the option of modifying this parameter in the application. This is because changing it has no bearing on the rotational speed estimate algorithm's ultimate conclusion.

## 2.5 Examples of videos recorded on the application

In Figure 2.3, we can see frames from 6 videos of a rotating disk with a straight strip with a speed of 1300 RPM recorded on a camera with different shutter speed settings.

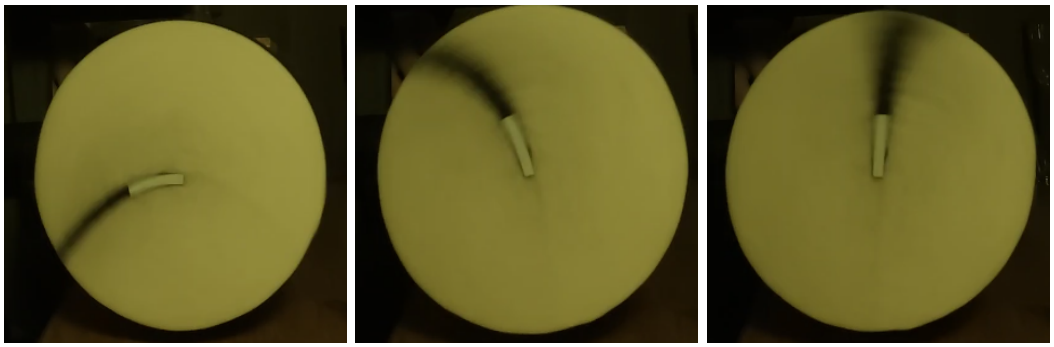
As we can see, the strip seems curved, which indicates the presence of rolling shutter effect. We also see that motion blur effect is present on every video. However, as we can see, with a decrease in the value of shutter speed, the rolling shutter effect becomes less noticeable.



**(a)** ISO: 500, SS: 1/50 ns, fps: 30

**(b)** ISO: 500, SS: 1/60 ns, fps: 30

**(c)** ISO: 500, SS: 1/70 ns, fps: 30



**(d)** ISO: 500, SS: 1/80 ns, fps: 30

**(e)** ISO: 500, SS: 1/90 ns, fps: 30

**(f)** ISO: 500, SS: 1/100 ns, fps: 30

**Figure 2.3:** An example of frames captured on video for a disk with a straight strip rotating at a speed of 1300 RPM for a camera with different values of shutter speed (SS). In these examples, we can also notice the presence of motion blur and rolling shutter effects.

# Chapter 3

# Proposed method

In this chapter, we present a novel method for determining the rotation speed of objects recorded on video from a simple phone camera. This method can be divided into several parts (Figure 3.1).

First, we select the points whose movement we will track on $F$ input video frames (such points we call the selected points), after which we actually track their movement using the optical flow method. Next, we fit ellipses into these trajectories using the least-squares method and select reliable ones. Finally, we estimate the rotation angle of the selected points, which correspond to the selected ellipses, around the common center of rotation, and finally determine the rotation speed of the object.

In this chapter, we will describe each of these steps in more detail.

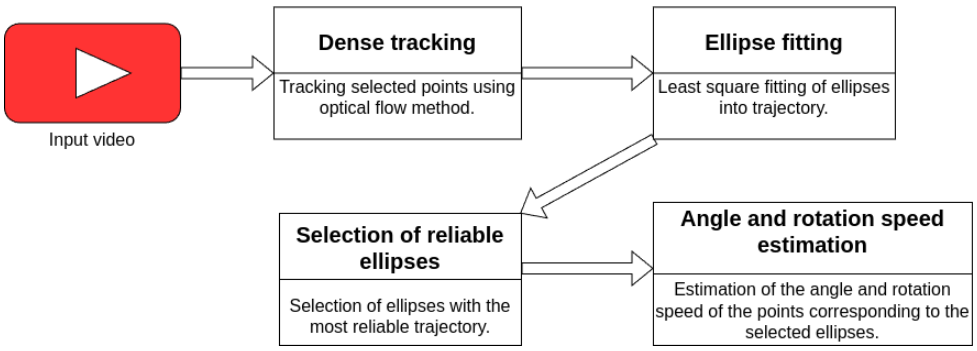



**Figure 3.1:** Process diagram of the method

## 3.1 Input

The input for the algorithm is a video shot on a phone application described in Chapter 2. In addition, this algorithm should get information about the FPS value $f$ with which this video was shot. Since this information is contained in the input video, it does not need to be specified manually.

The algorithm also assumes that the input video was shot subject to the restrictions described in Section 4.10.

## 3.2 Dense tracking

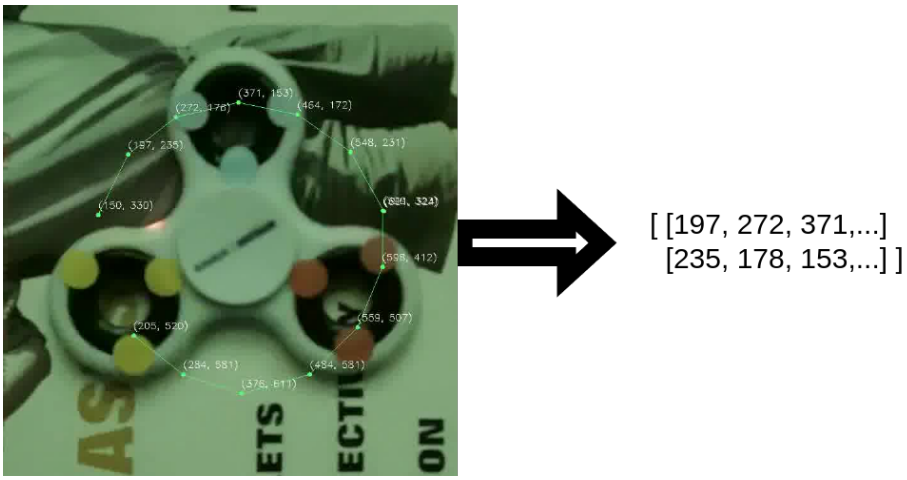



**Figure 3.2:** Visualization of the output data of the RAFT algorithm (shown on the right side of the picture) when tracking the movement of a point for 12 frames. Points with coordinates (on the left side of the picture) indicate the position of the considering point in the previous frames.

The tracking process consists of two parts: choosing the points whose movement we will observe (selected points) and the tracking process itself.

The maximum number of selected points $N$ cannot exceed the size of the video. In other words, we can track the movements of a maximum of each pixel in the video. However, in this case, it will lead to a long wait for the algorithm's response. Therefore, we select the points so that we place them at a distance $\gamma$ from each other both along the $x$ axis and along the $y$ axis. The number of points $N$ obtained is inversely proportional to the $\gamma$ distance. The way in which we choose this parameter $\gamma$ is explained in Section 4.3.

To track the selected points, we use the RAFT (Recurrent All-Pairs Field Transforms) algorithm, which is based on the principles of optical flow estimation. We use the official implementation of RAFT [23] with weights provided by the authors. Both the occlusion and the uncertainty CNNs operate on the same inputs as the RAFT flow-regression CNN. This algorithm was chosen because RAFT is a recent state-of-the-art optical flow algorithm that achieves high accuracy and robustness on various benchmarks. It can be considered as an advanced variant of optical flow methods, leveraging deep learning techniques to improve accuracy and robustness.

Fundamentally, RAFT works by repeatedly improving a preliminary estimate of the optical flow field. It begins by taking both spatially and temporal information from the input frames and extracting useful characteristics from them. These features are then used to construct a 'cost volume', which represents the similarity between different spatial locations and time steps in the frames.

As an output of the RAFT algorithm, is an array of size $N \times F_{\text{cons}} \times 2$, where $N$ is

the number of selected points and $F_{cons}$ is the number of frames in the video that we consider and 2 stands for $x$ and $y$ coordinate. The parameter $F_{cons}$ also influences the performance of the algorithm. This parameter was set according to the value established on the experiments described in Section 4.4. Each *n-th* array contains the coordinates of a given *n-th* selected point in the $F_{cons}$ frames.

For greater clarity, we demonstrate the work of the RAFT algorithm using the example of one point, whose movement we track for 12 frames (Figure 3.2). The left part of the picture shows the trajectory of this point over 12 frames, which was calculated using the RAFT algorithm. Points with coordinates indicate the position of the considering point in the previous frames. We represent this depicted trajectory using the 2D array shown on the right side of the picture. Where the first array is responsible for changing the $x$ coordinate of the point between 12 frames, and the second array is responsible for changing the $y$ coordinate of the point.

We also demonstrate (Fig. 3.4) how the RAFT algorithm works for all $N$ selected points when tracking their movement between two frames. In Figure 3.4a, we can see the position of all the selected points in the first frame of the video. In Figure 3.4b, we can see the position of the same selected points, but already in the second frame. The vector shows the direction of the change in the position of the point relative to its position in the previous frame.

Our goal is to find such points that their movement describes the movement of the given object. The main problem here is that the video can have a lot of noise.

By noise, we mean those points that do not move during the video. Thus, we call the noisy points of the first category. The behavior of these points corresponds to their expected behavior, but they do not carry any information about the rotation of the object.

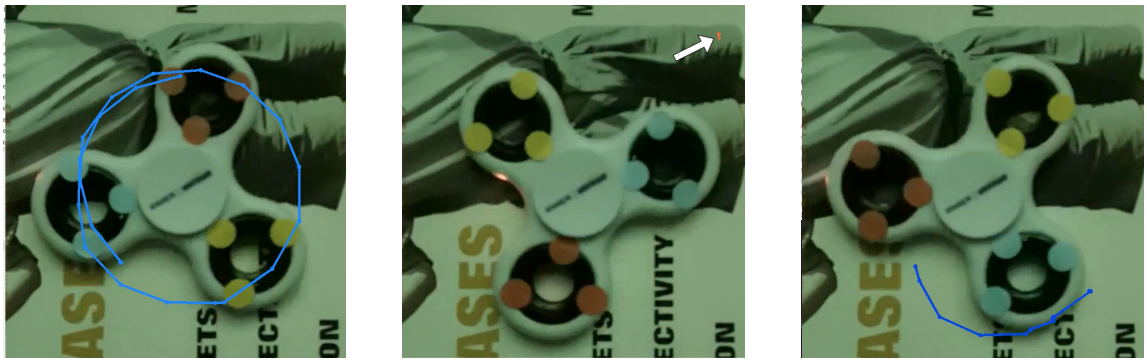
Also, by noise, we mean those selected points their movement occurs along a trajectory that fundamentally does not correspond to the trajectory of rotation of the object. We call such points noisy points of the second category.

In this case, we should find such points and remove them from consideration. The selected points that should move, and which at the same time move along a trajectory that approximately coincides with the axis of rotation of the object, we call "Rotational points" or points that are attached to an object.

Summarizing all the above, we can divide all the selected points into three categories:

1. **Rotational points**. These are the selected points whose trajectory approximately coincides with the trajectory of the object's rotation. An example of the trajectory of a point corresponding to this category is shown in the Figure 3.3a.
2. **Noisy points of the first category**. These are the selected points whose trajectory is static throughout the frames under consideration. An example of the trajectory of a point corresponding to this category is shown in the Figure 3.3b.

3. **Noisy points of the second category**. These are the selected points that move during the video, but their trajectory does not match the trajectory of the rotating object. An example of the trajectory of a point corresponding to this category is shown in the Figure 3.3c.



**(a)** An example of a rotational point trajectory for 30 frames.

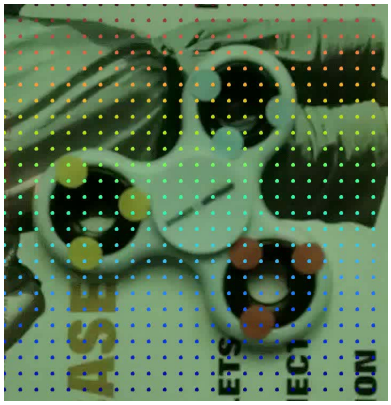
**(b)** An example of a trajectory of the noisy points of the first category for 30 frames.

**(c)** An example of a trajectory of the noisy points of the second category for 30 frames.

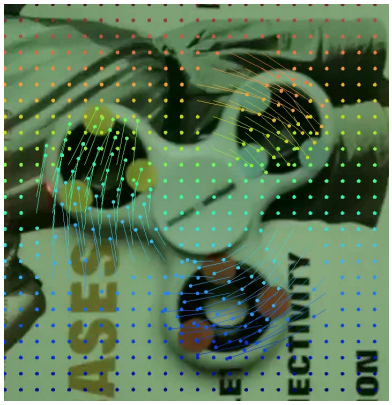
**Figure 3.3:** Examples of point trajectories for each category of a selected points, described above, during 30 frames.

For example, in Figure 3.4b, we see that there are a large number of points on it that do not seem to have a vector. This means that their position has remained almost unchanged compared to their position in the previous frame (the noisy points of the first category), which means that they do not carry any information about the movement of this object.

Note that the color of the dots does not reflect any of their features. The dots have different colors for the possibility of their further identification in the following frames.



**(a)** Location of the selected points tracked by the RAFT algorithm on the first frame. The points are located at a $\gamma$ distance from each other. The parameter $\gamma$ was found in Section 4.3.



**(b)** Location of the selected points in the second frame, which were found by the RAFT algorithm. The vector shows the direction of the change in the position of the point relative to its position in the previous frame. If it seems that a point has no vector, it means that it has practically not changed its position compared to the previous frame.

**Figure 3.4:** The location of the selected points in the first frame (Fig. 3.4a) and the position of the same points in the second frame (Fig. 3.4b), which was estimated using the RAFT algorithm. The color identifies each point.

## 3.3 Ellipse fitting

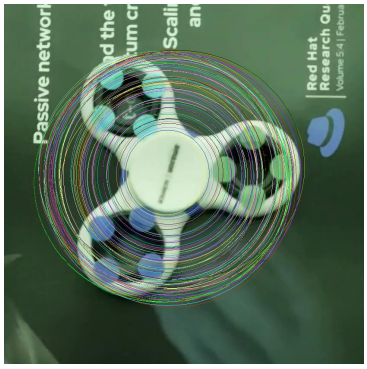
**(a)** Fitting circles for the trajectories of the rotation points for a video shot at approximately 45 degrees to the axis of rotation of the object.

**(b)** Fitting ellipses for the trajectories of the rotation points for a video shot at approximately 45 degrees to the axis of rotation of the object.

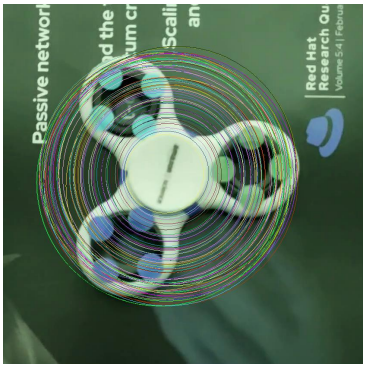
**Figure 3.5:** Examples of fitting circles and an ellipses for the trajectories of rotational points for a video shot approximately at an angle of 45 degrees to the axis of rotation of the object.

As mentioned above, for each selected point we have an array of its positions in each frame. In other words, we can call each such array of coordinates the trajectory of a given point because it gives us an idea of the approximate trajectory of the rotating object.

When we look at the individual points on a rotating object, we notice a few things: first, they all physically move in circles; second, these circles have the same center for each point;



**(a)** Fitting circles for the trajectories of rotational points for a video shot at an angle of 0 degrees to the axis of rotation of the object.



**(b)** Fitting ellipses for the trajectories of rotational points for a video shot at an angle of 0 degrees to the axis of rotation of the object.

**Figure 3.6:** Examples of fitting circles and ellipses for the trajectories of rotational points for a video shot at an angle of 0 degrees to the axis of rotation of the object.

and third, the size of the radii of these circles is limited by the size of the object.

Based on this, we can conclude that the rotational points should also move in a circle. However, this statement is true only for situations where a video of a rotating object was taken at an angle of 0 degrees to its axis of rotation. If the video is shot at an angle to a rotating object, the camera axis is not aligned with the rotation axis, resulting in elliptical object trajectories on the 2D image plane. The higher the angle of deviation from the axis of rotation when shooting, the less the trajectory of the points looks like a circle.

As we can see, two situations can occur. Rotational points move in circles or rotational points move in ellipses. Since we also have arrays with data on the position of each of these points in each frame, we can use the least squares method to fit an ellipse or circumference through these trajectory.

The problem that may arise in this situation is that we need to know when to fit an ellipse and when a circle. This problem can be solved in such a way that we will always fit either only circles or only ellipses.

In the event that we fit a circle in all cases, then we may encounter a problem in situations where the video of a rotating object is recorded at an angle to the axis of rotation of the object. As we can see in Figure 3.5a, in this case, the circles describe the real trajectory of the rotating object very poorly. After conducting a number of experiments, we found out that this leads to a very large relative error of the algorithm.

On the other hand, if we only fit ellipses for all videos, this will not have a negative impact on the results of the algorithm. This is due to the fact that in cases where the video is recorded at an angle to the axis of rotation of the object, ellipses describe the trajectory of the object very well (Fig. 3.5b). However, this method is also effective when a rotating object is recorded at an angle of 0 degrees to the axis of rotation of the object. As we can see in Figure 3.6a, where we fit circles, and in Figure 3.6b, where we fit ellipses, there is

almost no difference if we fit ellipses or circles. This is due to the fact that a circle is a special case of an ellipse.

Based on all of the above, it makes sense to use ellipses for fitting on all videos.

After applying this method to the trajectory of each from $N$ points we receive parameters of the ellipse for each such trajectory. These parameters are a tuple of the center of the ellipse, a tuple of its semi-major and semi-minor axes lengths, and an angle of its orientation $\theta$. So we now have one unique ellipse for each of the trajectory of each of the $N$ points that describes its movement in $F_{cons}$ frames.

## 3.4 Selection of reliable ellipses

As mentioned earlier, we can divide all the selected points into three categories: rotational points, noisy points of the first category and noisy points of the second category. The goal is to identify and select only the rotation points among all the selected points.

Since we know the trajectory of all the selected points, and also have an ellipses, the parameters of which we also know, fitted into each of these trajectories, then these ellipses can also be divided into the same 3 categories. Therefore, we can search and select not rotation points, but ellipses that correspond to their trajectories, since ellipses have a larger number of pairs of meters along which selection can take place. We show an example of ellipses that were fit to the trajectories of the all selected points in Figure 3.8a.

The process of selecting reliable ellipses takes place in two stages. In the first stage, we select the ellipses, among all the ellipses that were fitted into the trajectories corresponding to the selected point, by their center. We describe this process in more detail in Section 3.4.1. We demonstrate the results of this process in Figure 3.8b.

In the second part, we select ellipses from the three ellipses that remained after the first stage, relative to their main and secondary axes. We explain this part in more detail in Section 3.4.2. The result of the selection at this stage is shown in Figrue 3.8c.

### 3.4.1 Selection of reliable ellipses by center coordinates

We decided to select ellipses by their centers, after conducting several experiments in which we visualized the centers of the fitted ellipses in Section 3.3. The result of one of these experiments, which was conducted on the same video that was used in Figure 3.4, can be seen in Figure 3.7.

The centers shown in this figure can be divided into up to two clusters. The first cluster includes centers that are concentrated in a fairly small part. In the second category, we can include those centers whose location resembles a grid. Note that the location of the centers of the ellipses from the second cluster is very similar to the location of the selected points in the first frame (Fig. 3.4a).

As we can see in Figure 3.4a, the selected points, which are located approximately at the

same places as the centers of the ellipses from the second cluster, are not attached to the ellipse. That is, it is assumed that the trajectories of these points are static. Therefore, it can be concluded that the ellipses, the centers of which fell into the second cluster, fit the trajectories of noisy points of the first category.

Simply put, we have found a feature that will help us distinguish the ellipses corresponding to the trajectory of noisy points of the first category from all other ellipses.

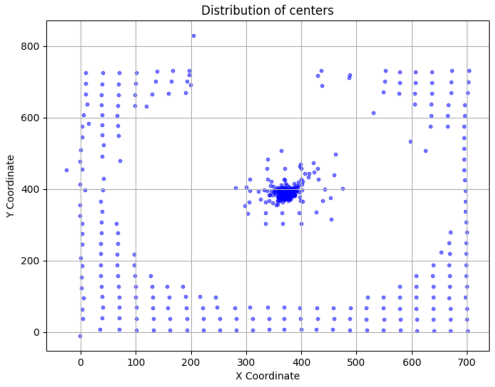



**Figure 3.7:** An example of the distribution of ellipse centers (blut ponints) on the frames surface, after fitting the ellipses to the trajectories of the selected points, described in the Section 3.4.

In order to select only those ellipses whose centers belong to the first cluster, we use the following sequence of actions. First, we find the medians of these centers separately on each of the $X$ and $Y$ axes. The result we denote as $(\mathbf{c}_{med_x}, \mathbf{c}_{med_y})$. After that, we select or do not select each ellipse separately according to the following criteria. An ellipse is considered selected if the coordinate of its center along the $X$ axis is no further than $\varepsilon_x$ from $\mathbf{c}_{med_x}$ and at the same time if the coordinate of its center along the $Y$ axis is no further than $\varepsilon_y$ from $\mathbf{c}_{med_y}$. The parameters $\varepsilon_x$ and $\varepsilon_y$ are hyperparameters that have been found experimentally. The process of finding these parameters is described in more detail in Section 4.5. We use finding the medians, not the average, because the median is less sensitive to outliers.

To demonstrate the results of this part of the ellipse selection process, we applied it to the ellipses shown in Figure 3.8a. The result of the selection by this method for the provided ellipses is shown in Figure 3.8b. In the next part, we will make a selection only among the ellipses selected in this part.

### 3.4.2 Selection of reliable ellipses by semi-major and semi-minor axes lengths

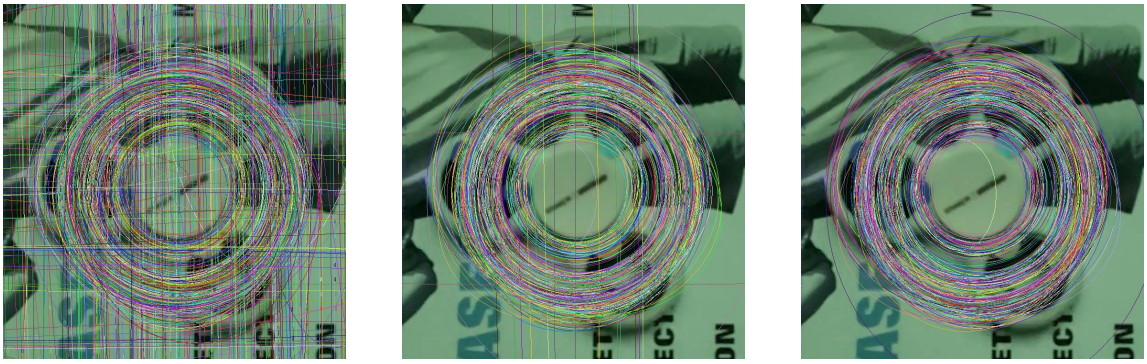
As we can see in Figure 3.8b, after selecting the ellipses by the coordinates of their centers, we still have those ellipses that do not even approximately resemble the relative trajectory of the object's rotation (they look like straight lines, but in fact they are very elongated ellipses). We can see that the peculiarity of such ellipses is that they are very elongated relative to the others. Their elongation is determined by the values of its major and minor axes. However, we cannot select ellipses for these parameters separately.

Therefore, in this part, we will carry out the selection by calculating the ratio of the semi-major axis of the semi-minor axis of the same ellipse for all ellipses and then find the median of all these ratios $r_{\text{med}}$. After that, we select or do not select each ellipse separately according to the following criteria. An ellipse is considered selected if the ration of its major

axis to the minor axis is no further than $\Delta$ from the median $r_{\text{med}}$ we found.

The selection process in this part goes the same as in the selection by center coordinates. First, we find the ratio of the semi-major and semi-minor axes legnths for each ellipse. After that, we find the medians of these ratios among all the ellipses selected in the previous part. The parameter $\Delta$ is a hyperparameter that has been found experimentally. The process of finding this parameter is described in more detail in Section 4.6. We use finding the medians, not the average, because the median is less sensitive to outliers.

To demonstrate the results of this part of the ellipse selection process, we applied it to the ellipses shown in Figure 3.8b. The result of the selection by this method for the provided ellipses is shown in Figure 3.8c. After that, the entire selection process is considered complete. We will consider the selected ellipses as ellipses that were fitted into the trajectories of the rotation points. All subsequent calculations will be performed only on these ellipses and above the points whose trajectory corresponds to these ellipses. These points, as mentioned earlier, we call the rotational points. We will denote the number of these points, and, accordingly, the ellipses, as $N_{rt}$.



**(a)** Fitted ellipses for all selected points using the least squares method for ellipses.

**(b)** The result of selecting reliable ellipses from the ellipses shown in Figure 3.8a by their centers.

**(c)** The result of selecting reliable ellipses from the ellipses shown in Figure 3.8b based on the relations of their main and side axes.

**Figure 3.8:** Visualization of the results of the selection process of reliable ellipses after each of the stages of this selection in the order (a), (b), (c).

## 3.5 Transformation of the coordinate system

Since in the usual case we have points that move along ellipses, this may complicate our further calculation of the angle by which they move in one frame. It would be much better for us if we had points that rotate in circles. However, as mentioned earlier, these points rotate in ellipses, only because the video is not shot at an angle of 0 degrees to the axis of rotation of the object. That is, we can transform the coordinate system into one where the trajectory of the points is transformed from elliptical to circular. This will allow us to simplify the calculations of the rotation angle of the points in each frame.

Each such ellipse, which shows an approximate trajectory of the $i$-th point, can be represented as the following equation:

$$\begin{bmatrix} x - c_{x_i} & y - c_{y_i} \end{bmatrix} \begin{bmatrix} \frac{1}{a_i^2} & 0 \\ 0 & \frac{1}{b_i^2} \end{bmatrix} \begin{bmatrix} x - c_{x_i} \\ y - c_{y_i} \end{bmatrix} = 1 \tag{3.1}$$

where $(\mathbf{c}_x, \mathbf{c}_y)$ represents the center of the $i$-th ellipse, and $a_i$ and $b_i$ represent the semi-major and semi-minor axes lengths respectively for the $i$-th ellipse.

It is worth noting that to have the same transformation matrix for two ellipses, the following conditions must be met:

- Same Center: The ellipses must have the same center. This ensures that the translation component of the transformation matrix is the same for both ellipses.
- Same Orientation Angle: The ellipses must have the same orientation angle (angle of rotation). This ensures that the rotation component of the transformation matrix is the same for both ellipses.
- Same Ratio of Lengths: If the ellipses have different semi-major and semi-minor axes lengths, their ratio must be the same. This ensures that the scaling component of the transformation matrix is the same for both ellipses.

If all of these conditions are met, then the transformation matrices to transform the ellipses into circles will be the same.

Here we are faced with the problem that theoretically, all these three conditions must be met for all selected ellipses. However, this is practically not the case due to the imperfect accuracy of the RAFT algorithm. Although the selected ellipses have different parameters, they can be considered approximately the same. This gives us the opportunity to find one ellipse that is somewhere in between all the selected ellipses.

To find and define such an ellipse, we have to define its center, semi-major, and semi-minor axes lengths, and the angle of its orientation $\theta$. We do it by finding the mean value for each of these parameters separately through all $N_{rt}$ remaining ellipses. We denote the center of this ellipse as $C = (c_x, c_y)$.

Once the ellipse is defined by its parameters we can find the transformation matrix by performing eigendecomposition on the matrix $\begin{bmatrix} \frac{1}{a^2} & 0 \\ 0 & \frac{1}{b^2} \end{bmatrix}$. Eigendecomposition gives us the eigenvalues $\lambda_1$ and $\lambda_2$, and the corresponding eigenvectors $\mathbf{v}_1$ and $\mathbf{v}_2$. This can be represented as follows:

$$A = \begin{bmatrix} \frac{1}{a^2} & 0 \\ 0 & \frac{1}{b^2} \end{bmatrix} \tag{3.2}$$

$$\mathbf{A}\mathbf{v}_i = \lambda_i \mathbf{v}_i \quad \text{for } i = 1, 2 \tag{3.3}$$

After that the transformation matrix $T$ is constructed using the eigenvectors as its columns:

$$T = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 \end{bmatrix} \tag{3.4}$$

Since the transformation should result in a circle, the lengths of the semi-axes of the resulting ellipse (after transformation) should be equal. Therefore, normalize the eigenvalues such that $\lambda_1 = \lambda_2 = 1$. For these purposes, we can construct the scaling matrix $S$ to normalize the eigenvalues:

$$S = \begin{bmatrix} \frac{1}{\lambda_1} & 0 \\ 0 & \frac{1}{\lambda_2} \end{bmatrix} \tag{3.5}$$

The transformation matrix $T$ is then updated as $T = T \cdot S$.

Having a transformation matrix, we can transform each $j$-th point of the trajectory for each $i$-th rotational point in order to have a better idea of their movement in the new coordinate system by applying the transformation matrix $T$:

$$\begin{bmatrix} \mathbf{x}'_{i,j} \\ \mathbf{y}'_{i,j} \end{bmatrix} = T^{-1} \begin{bmatrix} \mathbf{x}_{i,j} \\ \mathbf{y}_{i,j} \end{bmatrix} \tag{3.6}$$

where $(\mathbf{x}_{i,j}, \mathbf{y}_{i,j})$ represents the original coordinates for the $j$-th point along the trajectory associated with the $i$-th rotational point, and $(\mathbf{x}'_{i,j}, \mathbf{y}'_{i,j})$ represents the transformed coordinates for the same point.

## 3.6 Angle estimation

Let $\alpha_{i,j}$ be the angle by which the position of the *i-th* rotational point between the $j$ -th and $(j+1)$ -th frame has changed relative to the common center $C$, where $C = (c_x, c_y)$ is a center defined in Section 3.5.

Since we know the position of each point in each frame, we can calculate this angle $\alpha_{i,j}$ using the following formula:

$$\alpha_{i,j} = \arccos\left( \frac{\begin{bmatrix} x_{i,j+1} - x_{i,j} \\ y_{i,j+1} - y_{i,j} \end{bmatrix} \cdot \begin{bmatrix} x_{i,j} - c_x \\ y_{i,j} - c_y \end{bmatrix}}{\sqrt{(x_{i,j+1} - x_{i,j})^2 + (y_{i,j+1} - y_{i,j})^2} \cdot \sqrt{(x_{i,j} - c_x)^2 + (y_{i,j} - c_y)^2}} \right) \tag{3.7}$$

where $(\mathbf{x}_{i,j}, \mathbf{y}_{i,j})$ are the coordinates of the $i$-th point in the $j$-th frame, $(\mathbf{x}_{i,j+1}, \mathbf{y}_{i,j+1})$ are the coordinates of the $i$-th point in the $(j+1)$ -th frame, and $(\mathbf{c}_x, \mathbf{c}_y)$ are the coordinates of the common center.

Therefore, as a result, for each of $N_{rt}$ points, we have an array of size $F_{\text{cons}} - 1$, where $F_{cons}$ is the number of frames that are considered. We show an example of such an angle calculation for one of the experimental video in Figure 3.9.

Theoretically, we should get the same angle value $\alpha$ for all points and for all frames. However, this is practically not the case. As we can see in this figure, we have a spread of the angle value in a certain range. This is due to the inaccuracy of the RAFT algorithm. This inaccuracy lies in the fact that the algorithm sometimes poorly determines the movement of a point, which leads to the fact that it periodically remains almost in place or does not move perfectly around the circle.

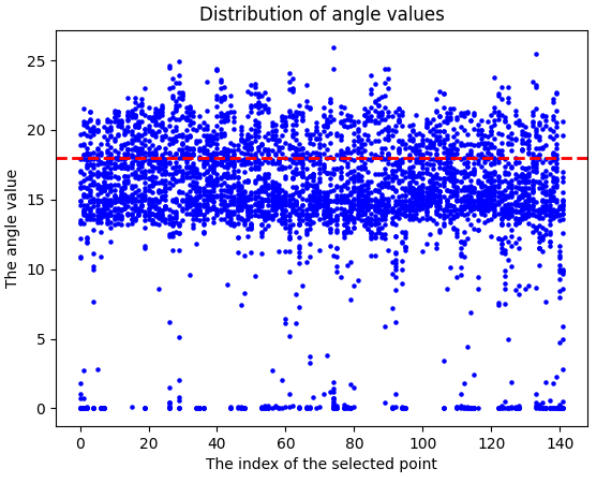



**Figure 3.9:** An example of the distribution of angles $N_{rt}$ for all points for one experimental video. The blue dots indicate the angle value. The x-axis means the index of the point for which the angles were calculated. The red dotted line means ground truth value.

Since it is impossible to get a single angle value $\alpha$ for all points in all frames, the task is to estimate what this value could be based on the data we have. As we can see from the results of one of the experiments (Fig. 3.9), the values of all angles lie close to the real value. However, we can also notice the presence of noisy values, which can have a bad effect on the estimation of the real value of the angle $\alpha$.

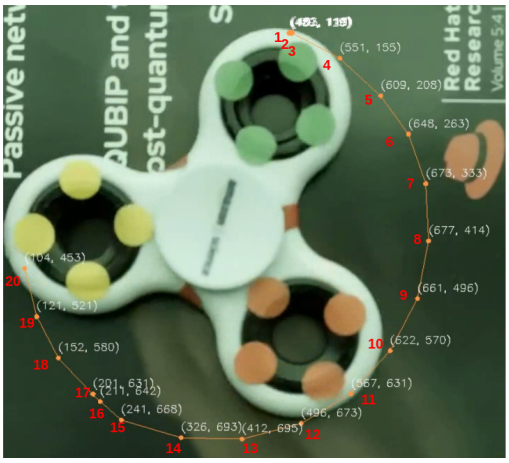



**Figure 3.10:** An example of the trajectory of one point over 20 frames, where the position of the point in each of the 20 frames is indicated by coordinates.

This noise can also be divided into two categories. An example of the noise of the first category can be seen in Figure 3.10. This figure shows the trajectory of one point over 20 frames, where the position of the point in each of the 20 frames is indicated by coordinates. As we can see, the position of the point changed almost slightly between frames 1-2, 2-3 and 16-17. This means that the angle of rotation during these frames will be very small compared to the angle of rotation between the other frames. Such angles, which are very small or very large in comparison with a large number of angles for a single point, we will call noise of the first category.

However, it may also happen that the point will move at such an insignificant angle during all the frames under consideration. We will call such points noisy points of the second category. By an insignificant angle, we mean an angle that is very small in comparison with the average angle of rotation of most other points. We call the average angle value for this point the noisy angle of the second category.

We do not put restrictions on the upper value of the angle because, as we see in Figure 3.9, there are very few values of angles that would be very large compared to the ground truth value. After that, it was noticed that this is also done for the rest of the experimental videos.

Therefore, before starting to estimate the angle value based on the available data, we will make a selection of reliable angles to eliminate the noise.

### 3.6.1 Selection of reliable angles

First, we get rid of noisy angles of the first category inside arrays of these angles for each of the $i$-th points separately. The selection process for angles that are not noise of the first category for the $i$-th point is as follows. We find the median among all the angles that correspond to the $i$-th point. We denote this median as $\alpha_{\text{med}_i}$. After that, the angle is considered the noise of the first category if its value is less than $\Psi$. The parameter $\Psi$ is set based on the results of the experiments described in Section 4.8. Otherwise, the angle is considered selected. We denote the selected angles as reliable.

After that, we find the average value of the angle for each of the $i$-th points among all the selected angles corresponding to this point. As a result, we have $N_{rt}$ angles.

Finally, to determine the value of the angle $\alpha$, we find the medians of these $N_{rt}$ angles.

## 3.7 Rotational speed calculation

Since we know the angle $\alpha$ of rotation of the object in one frame and we know the number of frames per second $f$ that the video was shot with, we can calculate the rotational speed of the object in RPM using the formula:

$$\omega = \frac{\alpha \cdot f \cdot 60}{360} \tag{3.8}$$

# Chapter 4

# Experiments and Results

In this chapter, we conduct a series of experiments to determine the performance and reliability of the algorithm described in the previous chapter. The experiments are conducted under different conditions and with different objects. There are also several experiments in which we look for the best value for all possible hyperparameters. In the end, we present a number of limitations under which this algorithm achieves its best efficiency. All the videos that were experimented on were taken from our dataset. We describe this dataset more in Appendix A. For more convenience, we also sorted these videos because of the experiments they were used in and also put them on GitHub [43].

Also, before describing all the experiments, we would like to note that the videos that were shot at an angle to the axis of rotation of the object were made so that the perspective effects were noticeable. That is, so that the trajectory of the points in the video differs from the circle.

And since, without loss of generality, the method does not exploit in any way the knowledge of that angle, it was calculated only approximately for all videos.

## 4.1 Experimental setup

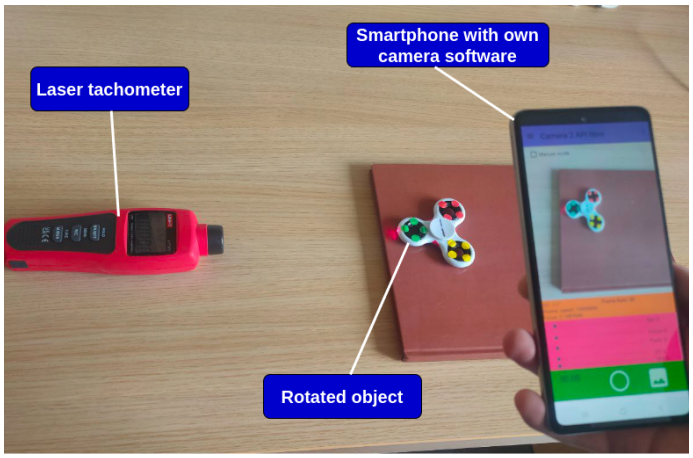



**Figure 4.1:** The integrated rotational speed measurement system.


**(a)** The experimental object (spinner).

**(b)** The experimental object (card).

**(c)** The experimental object (pack of handkerchiefs).

**Figure 4.2:** Experimental objects.

In Figure 4.1 we represent our rotational speed measurement system. This system was used to conduct all subsequent experiments as well as to obtain ground-truth data. This setup consists of three elements: a laser tachometer, one of the rotated objects (Fig. 4.2), and a smartphone with camera software described in Chapter 2.

We use the Uni-Trend UT372 **laser tachometer** [24] to capture ground truth (GT) rotation speed data. The range of the tachometer is 10 to 99 999 RPM with a relative error of ±0.04%. Since the tool with which we measure these values cannot measure a speed less than 10 RPM, the algorithm will also be limited to this value from below. We chose the lowest available measurement output rate of 0.5 seconds and captured the measurements via a USB cable. However, this measurement method has its drawbacks. The tachometer works well only in low light, whereas our algorithm performs better in brighter conditions. This situation requires finding a balance in lighting conditions to ensure optimal system performance.

We used three different rotating objects to carry out the experiments: a spinner, which is shown in Figure 4.2a, a simple card (Figure 4.2b), and a pack of handkerchiefs (Figure 4.2c).

A spinner was chosen as an experimental rotating object because it is quite easy to handle and also because it is common in the world and it is very interesting to be able to measure its speed. In addition, it has several markers of different colors on it. This allows us to obtain an object that is not perfectly symmetric in term of structure, which is one of the conditions for the effective operation of this algorithm. In addition, one of its most important qualities is that it can rotate for quite a long time.

The card is very suitable for the description of a structure that is not perfectly symmetrical in terms of structure. In this case, other problems arise. The experiments required the object to rotate for as long as possible, but achieving this is challenging because of significant friction that caused the card to stop quickly. Consequently, we attach it to the spinner to extend its rotation time.

The pack of handkerchiefs, in turn, was chosen for the same reasons as the card and also has the same disadvantages.

As a video **shooting device**, we used a Samsung M52 B smartphone with our camera software pre-installed. The device has a main camera with the following characteristics: 64 MP, f/1.8, 26mm (wide), 1/1.97", 0.7um, PDAF. The phone has 13 Android version.

As mentioned above, the device uses our own camera software, which is described in more detail in Chapter 2. The application allows to change settings such as ISO, shutter speed, focus, and frame rate. These parameters can be set both before the start of shooting and already in the process.

It is also worth noting that since we are creating an algorithm that any ordinary person will be able to use in the future, for this reason our setup does not presuppose the presence of a device that will hold the phone motionless. At the same time, the algorithm will not work effectively in case of severe instability. Therefore, we are creating an algorithm that would not be sensitive to a small shake, such as a human hand has.

## 4.2 The efficiency range of the algorithm

We conducted a number of experiments for this algorithm to determine how effective it is and at what intervals. To carry out these experiments, we used the setup described in Section 4.1.

We divided the experiments into two parts. In the first part, rotating objects were recorded at an angle of 0° to the axis of rotation of the object. In the second part of the experiments, rotating objects were recorded at an angle of 45° to the axis of rotation of the object.

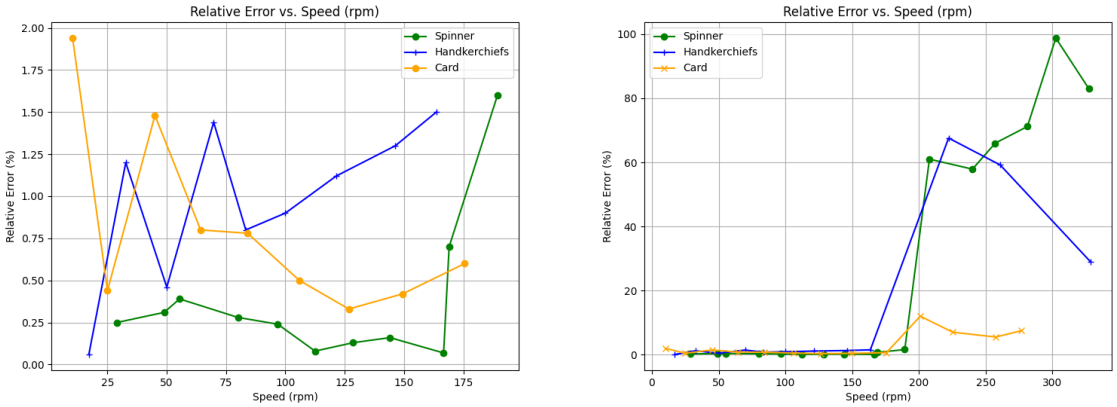
In each part, experiments were conducted on 3 different objects described in Section 4.1. A series of videos for one object within one part of the experiments was recorded with the same external conditions, as well as with the same camera settings. The only parameter that differed was the speed of the object. This allowed us to observe how the algorithm error behaves depending on the speed of the object.

To avoid a blurring effect, we used the maximum possible value for the shutter speed on our device, as this could affect the effectivness of the algorithm. We also tried to find the best balance of light using ISO setting, as it greatly affects the operation of the method.

It is also worth noting that we used the highest frame rate value on this device. We have found that the RAFT algorithm works better when we have more data on the movement of points. Increasing the frame rate value allows us to increase the number of data (frames) received over the same period of time.

All the videos on which experiments were conducted in this section can be found on GitHub [25].

### 4.2.1 Experiments for videos recorded at a 0° angle to the axis of rotation of the object.




**(a)** A graph of the dependence of the relative error of the algorithm on the object rotation speed for the range of 10 to 190 revolutions per minute.

**(b)** A graph of the dependence of the relative error of the algorithm on the object rotation speed for the range of 10 to 327 revolutions per minute.

**Figure 4.3:** Graphs of the dependence of the relative error of the algorithm on the object rotation speed in revolutions per minute for two ranges and three experimental objects that were recorded at an angle of 0° to the axis of rotation of the object. The data on which these graphs are based are in Table 4.1 for the spinner, in Table 4.2 for the card and in Table 4.3 for a pack of hangkerchiefs.

To carry out this part of the experiments, we recorded one series of videos for each rotating object: 16 videos were recorded for the spinner, 13 videos for the card, and 12 videos for the handkerchiefs pack. The recording process can be seen in Figure 4.4a. The results of the experiments for the spinner are shown in Table 4.1, for the card the results are shown in Table 4.2, and for the pack of handkerchiefs the results are shown in Table 4.3. To make these results clearer, we have visualized them in Figure 4.3. All the videos on which experiments were conducted in this subsection can be found on GitHub [26].

In Figure 4.3b, we can see that the algorithm is quite efficient in the rotation speed range of 10 to 190 rotations per minute. The relative error in this interval does not exceed 2%.

Therefore, we have made a separate graph, shown in Figure 4.3a to consider in more detail the operation of the algorithm in the rotation speed range of 10 to 190 turns per minute. As we can see, at this interval, it is difficult to say about any relative error of the algorithm from the rotation speed of the object.

However, when the rotation speed exceeds 190 rotations per minute, the algorithm begins to act rather ineffectively (Figure 4.3b). After making a number of observations, we found that the problem lies in the optical flow method we use in the algorithm. The problem is that the algorithm begins to determine the movement of a point with a very large error. The reasons for this behavior remain unknown to us at the moment.

| Param. / Video name | ISO | Shutter speed (ns) | FPS | Actual speed (RPM) | Calculated speed (RPM) | **Relative error (%)** |
|---|---|---|---|---|---|---|
| video0_26 | 1687 | 66666 | 30 | 11.19 ± 0.01 | 11.25 | **0.53 ± 0.002** |
| video0_27 | 1687 | 66666 | 30 | 29.00 ± 0.01 | 28.93 | **0.25 ± 0.003** |
| video0_28 | 1687 | 66666 | 30 | 48.87 ± 0.02 | 48.70 | **0.31 ± 0.020** |
| video0_29 | 1687 | 66666 | 30 | 79.55 ± 0.03 | 79.77 | **0.28 ± 0.012** |
| video0_30 | 1687 | 66666 | 30 | 96.00 ± 0.04 | 96.24 | **0.24 ± 0.010** |
| video0_31 | 1687 | 66666 | 30 | 112.31 ± 0.05 | 112.40 | **0.08 ± 0.008** |
| video0_32 | 1687 | 66666 | 30 | 128.23 ± 0.05 | 128.40 | **0.13 ± 0.005** |
| video0_33 | 1687 | 66666 | 30 | 143.20 ± 0.06 | 143.43 | **0.16 ± 0.003** |
| video0_34 | 1687 | 66666 | 30 | 166.80 ± 0.07 | 167.40 | **0.70 ± 0.003** |
| video0_35 | 1687 | 66666 | 30 | 189.33 ± 0.07 | 186.30 | **1.60 ± 0.005** |
| video0_36 | 1687 | 66666 | 30 | 220.87 ± 0.08 | 84.20 | **61.0 ± 0.004** |
| video0_37 | 1687 | 66666 | 30 | 240.10 ± 0.09 | 100.90 | **57.9 ± 0.002** |
| video0_38 | 1687 | 66666 | 30 | 257.14 ± 1.00 | 87.18 | **66.0 ± 0.003** |
| video0_39 | 1687 | 66666 | 30 | 281.25 ± 1.01 | 81.00 | **71.2 ± 0.003** |
| video0_40 | 1687 | 66666 | 30 | 302.52 ± 1.02 | 3.38 | **98.8 ± 0.005** |
| video0_41 | 1687 | 66666 | 30 | 327.27 ± 1.03 | 55.00 | **83.0 ± 0.005** |

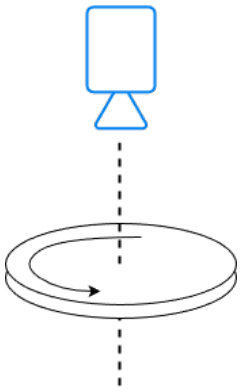
**Table 4.1:** Algorithm results for a **spinner** recorded at an angle of 0° to the axis of rotation of the object and video parameters. The videos that correspond to the data from this table can be found in the GitHub repository [29].

| Param. / Video name | ISO | Shutter speed (ns) | FPS | Actual speed (RPM) | Calculated speed (RPM) | **Relative error (%)** |
|---|---|---|---|---|---|---|
| video0_1 | 2024 | 66666 | 30 | 10.30 ± 0.01 | 10.10 | **1.94 ± 0.001** |
| video0_2 | 2024 | 66666 | 30 | 25.00 ± 0.01 | 25.11 | **0.44 ± 0.005** |
| video0_3 | 2024 | 66666 | 30 | 45.00 ± 0.02 | 45.67 | **1.48 ± 0.010** |
| video0_4 | 2024 | 66666 | 30 | 64.28 ± 0.02 | 64.79 | **0.80 ± 0.004** |
| video0_5 | 2024 | 66666 | 30 | 84.10 ± 0.03 | 84.76 | **0.78 ± 0.012** |
| video0_6 | 2024 | 66666 | 30 | 105.88 ± 0.04 | 105.35 | **0.50 ± 0.010** |
| video0_7 | 2024 | 66666 | 30 | 126.70 ± 0.05 | 126.34 | **0.33 ± 0.007** |
| video0_8 | 2024 | 66666 | 30 | 149.38 ± 0.06 | 148.75 | **0.42 ± 0.005** |
| video0_9 | 2024 | 66666 | 30 | 175.60 ± 0.07 | 176.66 | **0.60 ± 0.003** |
| video0_10 | 2024 | 66666 | 30 | 201.00 ± 0.08 | 225.12 | **12.0 ± 0.003** |
| video0_11 | 2024 | 66666 | 30 | 225.40 ± 0.09 | 241.24 | **7.00 ± 0.004** |
| video0_12 | 2024 | 66666 | 30 | 257.60 ± 1.00 | 271.78 | **5.5 ± 0.002** |
| video0_13 | 2024 | 66666 | 30 | 276.84 ± 1.01 | 297.58 | **7.49 ± 0.002** |

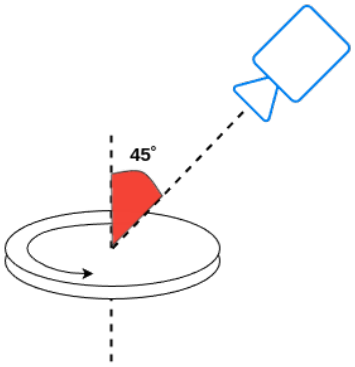
**Table 4.2:** Algorithm results for a **card** recorded at an angle of 0° to the axis of rotation of the object. The videos that correspond to the data from this table can be found in the GitHub repository [27].

| Param. / Video name | ISO | Shutter speed (ns) | FPS | Actual speed (RPM) | Calculated speed (RPM) | **Relative error (%)** |
|---|---|---|---|---|---|---|
| video0_14 | 3300 | 66666 | 30 | 17.14 ± 0.01 | 17.15 | **0.06 ± 0.002** |
| video0_15 | 3300 | 66666 | 30 | 32.70 ± 0.01 | 33.10 | **1.2 ± 0.003** |
| video0_16 | 3300 | 66666 | 30 | 50.00 ± 0.02 | 49.77 | **0.46 ± 0.001** |
| video0_17 | 3300 | 66666 | 30 | 69.76 ± 0.03 | 68.75 | **1.44 ± 0.021** |
| video0_18 | 3300 | 66666 | 30 | 83.30 ± 0.03 | 82.63 | **0.8 ± 0.001** |
| video0_19 | 3300 | 66666 | 30 | 100.00 ± 0.04 | 100.90 | **0.9 ± 0.002** |
| video0_20 | 3300 | 66666 | 30 | 121.40 ± 0.05 | 122.77 | **1.12 ± 0.001** |
| video0_21 | 3300 | 66666 | 30 | 146.34 ± 0.06 | 144.34 | **1.3 ± 0.007** |
| video0_22 | 3300 | 66666 | 30 | 163.63 ± 0.06 | 166.14 | **1.5 ± 0.001** |
| video0_23 | 3300 | 66666 | 30 | 222.22 ± 0.09 | 71.90 | **67.6 ± 0.002** |
| video0_24 | 3300 | 66666 | 30 | 260.86 ± 1.00 | 106.22 | **59.2 ± 0.002** |
| video0_25 | 3300 | 66666 | 30 | 328.47 ± 1.30 | 233.19 | **29 ± 0.001** |

**Table 4.3:** Algorithm results for a **pack of handkerchiefs** recorded at an angle of 0° to the axis of rotation of the object. The videos that correspond to the data from this table can be found in the GitHub repository [28].



**(a)** The process of recording a rotating object at an angle of 0° to the axis of rotation of the object. The disk represents a rotating object.

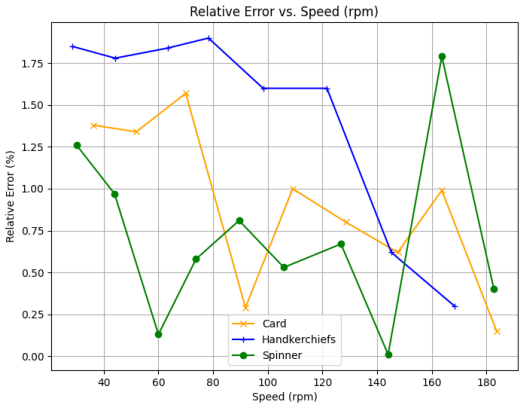



**(b)** The process of recording a rotating object at an angle of 45° to the axis of rotation of the object. The disk represents a rotating object.

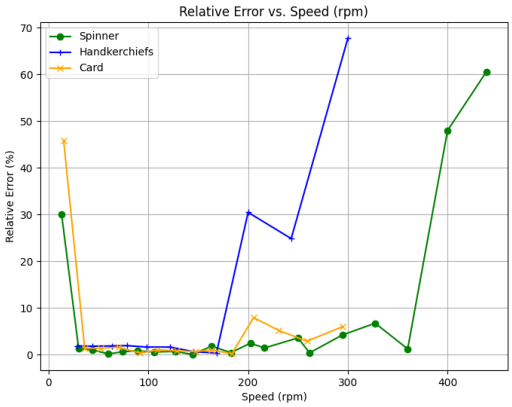
**Figure 4.4:**

### 4.2.2 Experiments for videos recorded at a 45° angle to the axis of rotation of the object.

To carry out this part of the experiments, we recorded one video series for each rotating object: 20 videos were recorded for the spinner, 14 videos for the card and 11 videos for the pack of handkerchiefs. The recording process can be seen in Figure 4.4b. The results of the experiments for the spinner are shown in Table 4.4, for the card the results are shown




**(a)** A graph of the dependence of the relative error of the algorithm on the object rotation speed for the range of 20 to 190 revolutions per minute.




**(b)** A graph of the dependence of the relative error of the algorithm on the object rotation speed for the range of 0 to 440 revolutions per minute.

**Figure 4.5:** Graphs of the dependence of the relative error of the algorithm on the object rotation speed in revolutions per minute for two ranges and three experimental objects that were recorded at an angle of 45° to the axis of rotation of the object. The data on which these graphs are based are in Table 4.4 for the spinner, in Table 4.13 for the card and in Table 4.6 for a pack of hangkerchiefs.

in Table 4.13, and for the pack of handkerchiefs the results are shown in Table 4.6. To make these results clearer, we have visualized them in Figure 4.5. All the videos on which experiments were conducted in this subsection can be found on GitHub [30].

In Figure 4.5b, we see that, as with recording a video at an angle of 0° to the axis of rotation of the object, the algorithms are effective only at speeds less than 190 RPM. However, unlike him, here we see that the algorithm is also ineffective when the rotation speed of the object is less than 20 revolutions per minute.

Therefore, we have made a separate graph, shown in Figure 4.5a to consider in more detail the operation of the algorithm in the rotation speed range of 20 to 190 turns per minute. As we can see, at this interval, it is difficult to say about any relative error of the algorithm from the rotation speed of the object. However, as in the previous part of the experiments, the relative error in this interval does not exceed 2%.

We can see in Figure 4.5b that, as in the previous part of the experiments, at a speed exceeding 190 revolutions per minute, the algorithm begins to produce results with a fairly large relative error. The reason for this also is that the algorithm begins to determine the movement of a point with a very large error.

However, unlike recording rotating objects at an angle of 0 degrees to their axis of rotation, here we are faced with the fact that the algorithm is also ineffective at an object rotation speed of less than 20 revolutions per minute (Figure 4.5b). During a number of observations, we found out that this is due to the insufficiently small number of frames $F$

| Param. / Video name | ISO | Shutter speed (ns) | FPS | Actual speed (RPM) | Calculated speed (RPM) | **Relative error (%)** |
|---|---|---|---|---|---|---|
| video0_67 | 3300 | 66666 | 30 | 13.00 ± 0.01 | 9.10 | **30 ± 0.002** |
| video0_68 | 3300 | 66666 | 30 | 30.10 ± 0.01 | 30.48 | **1.26 ± 0.01** |
| video0_69 | 3300 | 66666 | 30 | 43.95 ± 0.02 | 43.52 | **0.97 ± 0.007** |
| video0_70 | 3300 | 66666 | 30 | 60.00 ± 0.02 | 60.08 | **0.13 ± 0.002** |
| video0_71 | 3300 | 66666 | 30 | 73.77 ± 0.03 | 74.20 | **0.58 ± 0.008** |
| video0_72 | 3330 | 66666 | 30 | 89.50 ± 0.03 | 88.77 | **0.81 ± 0.007** |
| video0_73 | 3330 | 66666 | 30 | 105.88 ± 0.04 | 106.45 | **0.53 ± 0.007** |
| video0_74 | 3330 | 66666 | 30 | 126.76 ± 0.05 | 125.90 | **0.67 ± 0.007** |
| video0_75 | 3330 | 66666 | 30 | 144.00 ± 0.05 | 143.98 | **0.01 ± 0.007** |
| video0_76 | 3330 | 66666 | 30 | 162.63 ± 0.06 | 160.70 | **1.79 ± 0.007** |
| video0_77 | 3330 | 66666 | 30 | 182.70 ± 0.07 | 183.43 | **0.4 ± 0.008** |
| video0_78 | 3330 | 66666 | 30 | 202.24 ± 0.08 | 207.14 | **2.42 ± 0.011** |
| video0_79 | 3330 | 66666 | 30 | 216.39 ± 0.08 | 219.42 | **1.4 ± 0.009** |
| video0_80 | 3330 | 66666 | 30 | 250.00 ± 1.00 | 258.90 | **3.56 ± 0.004** |
| video0_81 | 3330 | 66666 | 30 | 261.62 ± 1.00 | 262.57 | **0.36 ± 0.007** |
| video0_82 | 3330 | 66666 | 30 | 294.68 ± 1.02 | 282.16 | **4.2 ± 0.002** |
| video0_83 | 3330 | 66666 | 30 | 327.42 ± 1.03 | 305.32 | **6.7 ± 0.008** |
| video0_84 | 3330 | 66666 | 30 | 360.30 ± 1.04 | 355.60 | **1.2 ± 0.008** |
| video0_85 | 3330 | 66666 | 30 | 400.89 ± 1.06 | 205.01 | **48.0 ± 0.001** |
| video0_86 | 3330 | 66666 | 30 | 439.36 ± 1.07 | 173.14 | **60.6 ± 0.002** |

**Table 4.4:** Algorithm results for a **spinner** recorded at an angle of 45° to the axis of rotation of the object. The videos that correspond to the data from this table can be found in the GitHub repository [33].

(described in Section 4.4) that we are considering. We could increase this number of frames, which would significantly improve the results in this experiment for this interval; however, this would lead to a deterioration in the results in a much larger number of cases.

## 4.3 Tuning the $\gamma$ parameter

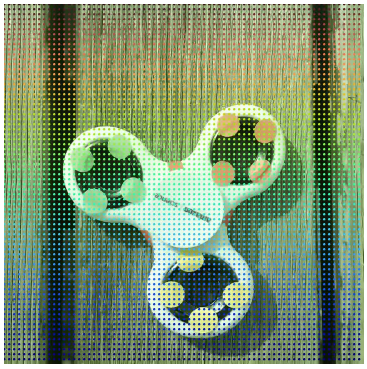
One of the important parameters of the algorithm is the number of selected points $N$. This is the number of points whose movement is tracked by the RAFT algorithm. This number depends on the size of the video, but it also depends on the parameter $\gamma$. This parameter defines the distance between the selected points. We visualize the results of setting different values of $\gamma$ in Figure 4.6 on two examples.

| Param. / Video name | ISO | Shutter speed (ns) | FPS | Actual speed (RPM) | Calculated speed (RPM) | **Relative error (%)** |
|---|---|---|---|---|---|---|
| video0_42 | 2024 | 66666 | 30 | 15.00 ± 0.01 | 41.89 | **45.9 ± 0.001** |
| video0_43 | 2024 | 66666 | 30 | 36.10 ± 0.01 | 36.60 | **1.38 ± 0.003** |
| video0_44 | 2024 | 66666 | 30 | 51.90 ± 0.02 | 52.60 | **1.34 ± 0.007** |
| video0_45 | 2024 | 66666 | 30 | 70.00 ± 0.03 | 71.10 | **1.57 ± 0.002** |
| video0_46 | 2024 | 66666 | 30 | 91.83 ± 0.03 | 92.10 | **0.29 ± 0.008** |
| video0_47 | 2024 | 66666 | 30 | 109.10 ± 0.04 | 110.20 | **1.00 ± 0.007** |
| video0_48 | 2024 | 66666 | 30 | 128.57 ± 0.05 | 129.60 | **0.80 ± 0.006** |
| video0_49 | 2024 | 66666 | 30 | 147.50 ± 0.06 | 148.46 | **0.62 ± 0.008** |
| video0_50 | 2024 | 66666 | 30 | 163.63 ± 0.06 | 162.00 | **0.99 ± 0.007** |
| video0_51 | 2024 | 66666 | 30 | 183.67 ± 0.07 | 183.95 | **0.15 ± 0.004** |
| 1video0_52 | 2024 | 66666 | 30 | 205.50 ± 0.08 | 221.80 | **7.90 ± 0.008** |
| 1video0_53 | 2024 | 66666 | 30 | 230.77 ± 0.09 | 242.70 | **5.16 ± 0.005** |
| video0_54 | 2024 | 66666 | 30 | 259.00 ± 1.00 | 266.38 | **2.84 ± 0.008** |
| video0_55 | 2024 | 66666 | 30 | 295.00 ± 1.02 | 277.31 | **5.99 ± 0.003** |

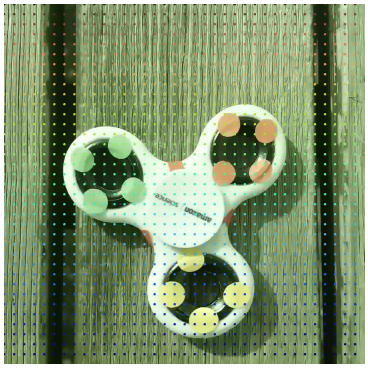
**Table 4.5:** Algorithm results for a **card** recorded at an angle of 45° to the axis of rotation of the object. The videos that correspond to the data from this table can be found in the GitHub repository [31].

| Param. / Video name | ISO | Shutter speed (ns) | FPS | Actual speed (RPM) | Calculated speed (RPM) | **Relative error (%)** |
|---|---|---|---|---|---|---|
| video0_56 | 3300 | 66666 | 30 | 28.57 ± 0.01 | 29.10 | **1.85 ± 0.001** |
| video0_57 | 3300 | 66666 | 30 | 44.20 ± 0.02 | 44.99 | **1.78 ± 0.001** |
| video0_58 | 3300 | 66666 | 30 | 63.35 ± 0.02 | 64.52 | **1.84 ± 0.004** |
| video0_59 | 3300 | 66666 | 30 | 78.30 ± 0.03 | 79.80 | **1.9 ± 0.002** |
| video0_60 | 3300 | 66666 | 30 | 98.29 ± 0.03 | 99.86 | **1.6 ± 0.004** |
| video0_61 | 3300 | 66666 | 30 | 121.62 ± 0.04 | 119.66 | **1.6 ± 0.005** |
| video0_62 | 3300 | 66666 | 30 | 145.16 ± 0.05 | 144.25 | **0.62 ± 0.006** |
| video0_63 | 3300 | 66666 | 30 | 168.22 ± 0.06 | 167.65 | **0.3 ± 0.003** |
| video0_64 | 3300 | 66666 | 30 | 200.00 ± 0.07 | 260.90 | **30.45 ± 0.007** |
| video0_65 | 3300 | 66666 | 30 | 243.20 ± 0.08 | 303.63 | **24.84 ± 0.004** |
| video0_66 | 3300 | 66666 | 30 | 300.10 ± 0.09 | 96.62 | **67.8 ± 0.002** |

**Table 4.6:** Algorithm results for a **pack of handkerchiefs** recorded at an angle of 45° to the axis of rotation of the object. The videos that correspond to the data from this table can be found in the GitHub repository [32].

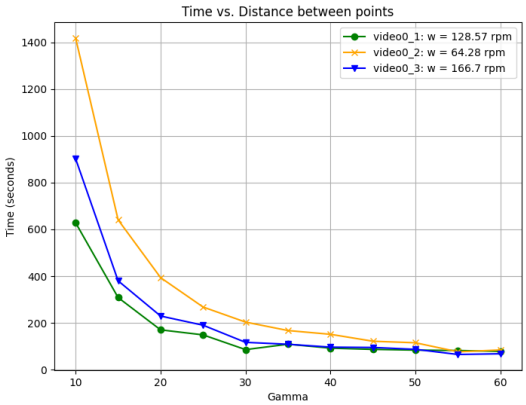


**(a)** Distribution of the 1296 points by the RAFT algorithm in the first frame for $\gamma = 15$

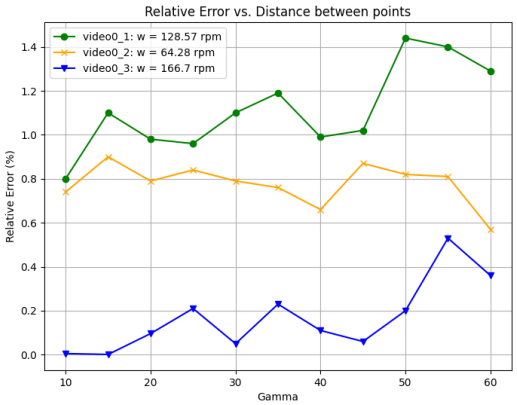


**(b)** Distribution of the 5184 points by the RAFT algorithm in the first frame for $\gamma = 30$

**Figure 4.6:** Distribution of the points for $\gamma = 15$ and $\gamma = 30$ on the first frame of the same video.




**(a)** The graph shows the dependence of the algorithm execution time on the set distance $\gamma$ between query points executed for three different speeds of a rotating object




**(b)** The graph shows the dependence of the relative error of the algorithm on the set distance $\gamma$ between query points executed for three different speeds of a rotating object.

**Figure 4.7:** The effect of the $\gamma$ parameter on the algorithm execution time and on its relative error. Information about the videos on which these experiments were conducted is shown in Table 4.7.

Since we know the size of the video $H \times W$, where $H$ is the height of the video and $W$ is the width of the video, we can easily calculate the number of points for the given $\gamma$ using this formula:

$$N = \left\lfloor \frac{H}{\gamma} \right\rfloor \times \left\lfloor \frac{W}{\gamma} \right\rfloor \tag{4.1}$$

As we see, the number of points depends of three variables: height and width of the video and distance between these selected points. However, we cannot modify the height and width of the video to find the most suitable number of points. Instead, we can easily

| Param. / Video name | ISO | Shutter speed (ns) | FPS | Angle (°) | Actual speed (RPM) | Calculated speed (RPM) | **Relative error (%)** |
|---|---|---|---|---|---|---|---|
| video0_87 | 3300 | 66666 | 30 | 0 | 128.57 ± 0.05 | 128.30 | **0.2 ± 0.002** |
| video0_4 | 2024 | 66666 | 30 | 0 | 64.28 ± 0.03 | 64.79 | **0.79 ± 0.001** |
| video0_89 | 3300 | 66666 | 30 | 0 | 166.70 ± 0.07 | 166.87 | **0.1 ± 0.003** |

**Table 4.7:** Information about the video parameters and the result of executing the algorithm on them with the optimal $\gamma$ parameter set. The videos that correspond to the data from this table can be found in the GitHub repository [34].

tune the $\gamma$. As follows from the formula 4.1 the number of points is inversely proportional to the distance between them in the video.

We conducted 11 experiments on 3 different videos to find the best parameter $\gamma$. In these experiments, we tried various values of this parameter and looked at two important components: the speed of the algorithm execution (Fig. 4.7a) and what error the algorithm will make when using this value of this parameter (Fig. 4.7b). The objects rotated in these videos at different rotational speeds. The results of the experiments are shown in Figure 4.7. In general, these results show that images with a higher number of points provide a more accurate rotational speed measurement. However, such images result in longer computational time and hence slower system response due to the increasing volume of data to process. In order to strike a balance between the accuracy of the measurement and the response time of the system, $\gamma = 30$ was chosen as the distance between the selected points. All the videos on which experiments were conducted in this section can be found on GitHub [35]. Information about these videos is also provided in Table 4.7.

## 4.4 Tuning the number of considering frames $F_{cons}$

As mentioned above, our input data is a recorded video. We suggest that the resulting video contains a rotating object from its very beginning. Since the videos can be different in length, it becomes necessary to determine the optimal number of the considering frames $F_{\text{cons}}$ between which the change in the position of the selected points will be tracked.

During the experiments, it was noticed that in some cases the first 2-3 frames contain interference. Therefore, it was decided to start measuring from the third frame so that the camera was already precisely configured for shooting.

To determine the optimal number of frames $F_{\text{cons}}$ that will be used by the algorithm, we carried out a series of experiments to examine how the number of the considering frames $F_{\text{cons}}$ affects the accuracy of the algorithm. The results of these experiments are shown in Figure 4.8.

As we can see from the graph in Figure 4.8, when considering a small number of frames (in the range of 10 to 15 frames), the relative error of the algorithm is quite large. This is because with fewer frames we have less data on the trajectories of the selected points. This

leads to a greater inaccuracy of the least-squares method.

At the same time, we see that the relative error of the algorithm increases when the number of frames is greater than 40. This is because the further the recording goes, the more likely it is that the camera will change its position, even slightly. This actually leads to an error.

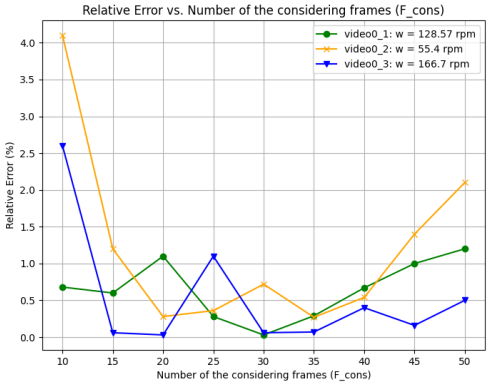



**Figure 4.8:** Graph of the dependence of the relative error of the algorithm on the number $F_{\text{cons}}$ of frames considered for three different speeds of a rotating object. Information about the videos on which these experiments were conducted is shown in Table 4.7.

Based on this, we should choose F from the range from 15 to 35. However, we should pay attention to the following detail. Since the videos for this experiment were recorded with frame rate value $f = 30$, we note that with an increase in the number of $f$, we could consider a larger number of frames, and this would not lead to large error, which is related to changing the camera position. This suggests that the number $F_{\text{cons}}$ depends directly on the frame rate value. All the videos on which experiments were conducted in this section can be found on GitHub [35]. Information about these videos is also provided in Table 4.8.

Based on all of the above, we set the value of $F_{\text{cons}}$ to $f$ (frame rate).

| Param. / Video name | ISO | Shutter speed (ns) | FPS | Angle (°) | Actual speed (RPM) | Calculated speed (RPM) | **Relative error (%)** |
|---|---|---|---|---|---|---|---|
| video0_87 | 3300 | 66666 | 30 | 0 | 128.57 ± 0.05 | 128.30 | **0.2 ± 0.002** |
| video0_88 | 1528 | 66666 | 30 | 0 | 55.40 ± 0.02 | 55.73 | **0.59 ± 0.001** |
| video0_89 | 3300 | 66666 | 30 | 0 | 166.70 ± 0.07 | 166.87 | **0.1 ± 0.003** |

**Table 4.8:** Information about the video parameters and the result of executing the algorithm on them with the optimal$F_{\text{cons}}$ parameter set. The videos that correspond to the data from this table can be found in the GitHub repository [35].

## 4.5 Tuning the $\varepsilon_x$ and $\varepsilon_y$ parameters for selecting ellipses based on the coordinates of their centers.

The ellipse selection process is one of the main parts of the algorithm. Thanks to this process, we get rid of a lot of noise, which can have a negative impact on the final result. One of its parts is the selection of ellipses by the coordinates of their centers, which is described in more detail in Section 3.4.

There we first find the medians of ellipses centers separately on each of the $X$ and $Y$ axes.

The result we denote as $(\mathbf{c}_{med_x}, \mathbf{c}_{med_y})$. An ellipse is considered selected if the coordinate of its center along the $X$ axis is no further than $\varepsilon_x$ from $\mathbf{c}_{med_x}$ and at the same time if the coordinate of its center along the $Y$ axis is no further than $\varepsilon_y$ from $\mathbf{c}_{med_y}$.

These parameters epsilon and epsilon can be written as the following formulas:

$$\varepsilon_x = \epsilon \cdot c_{\mathrm{med}_x}, \tag{4.2}$$

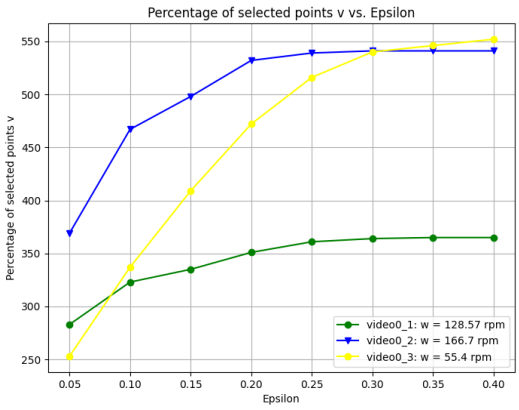
$$\varepsilon_y = \epsilon \cdot c_{\mathrm{med}_y}, \tag{4.3}$$

where$\epsilon$ is a hyperparamer that actually defines the boundary in this case.

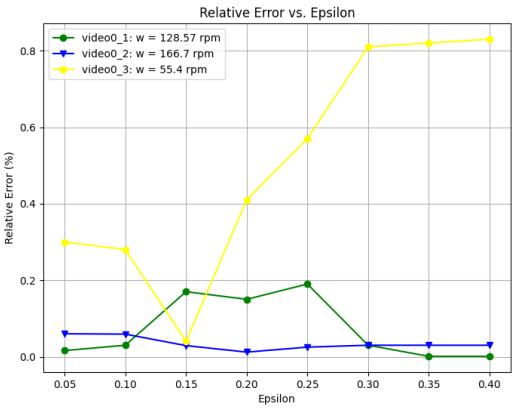
To determine the best value for this parameter, we conducted a series of experiments on several different videos, for which the objects had different rotation speeds. The results of these experiments can be seen in Figure 4.9. All the videos on which experiments were conducted in this section can be found on GitHub [36]. Information about these videos is also provided in Table 4.9.

As we can see in Figure 4.9a, it makes no sense to set the parameter value to a value greater than 0.3, since the percentage of selected ellipses remains almost the same over a long interval. This may indicate that all ellipses with approximately the same centers have already been selected and that only noise remains.

We set the value of the epsilon parameter at 0.13, because, based on Figure 4.9b, the lowest relative error of the algorithm is achieved for this value on average.




**(a)** A graph of the dependence of the percentage of the selected points on the set $\epsilon$ value for 3 videos.




**(b)** A graph of the dependence of the relative error of the algorithm on the set $\epsilon$ value for 3 videos.

**Figure 4.9:** Graphs of the dependence of the percentage of the selected points (Fig. 4.9a) and the relative error of the algorithm (Fig. 4.9b) on the set value of $\epsilon$ for 3 videos. Information about the videos on which these experiments were conducted is shown in Table 4.9.

| Param. / Video name | ISO | Shutter speed (ns) | FPS | Angle (°) | Actual speed (RPM) | Calculated speed (RPM) | **Relative error (%)** |
|---|---|---|---|---|---|---|---|
| video0_87 | 3300 | 66666 | 30 | 0 | 128.57 ± 0.05 | 128.30 | **0.2 ± 0.002** |
| video0_89 | 3300 | 66666 | 30 | 0 | 166.70 ± 0.07 | 166.87 | **0.1 ± 0.003** |
| video0_88 | 1528 | 66666 | 30 | 0 | 55.40 ± 0.02 | 55.73 | **0.59 ± 0.001** |

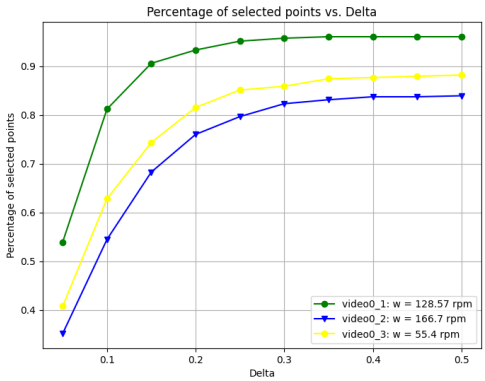
**Table 4.9:** Information about the video parameters and the result of executing the algorithm on them with the optimal $\epsilon$ parameter set. The videos that correspond to the data from this table can be found in the GitHub repository [36].

## 4.6 Tuning the $\Delta$ parameter for selecting ellipses by the ration of its semi-major and semi-minor axes lengths

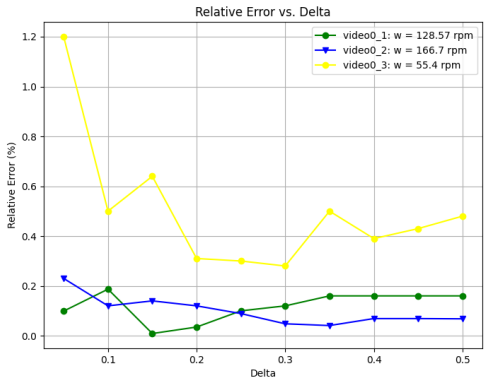
In the second part of the selection of ellipses, we first calculate the ratio of the semi-major and semi-minor axes for each ellipse and then find the median $r_{\text{med}}$ of all these ratios. In this part, an ellipse is considered to be selected if the ratio of its semi-major and semi-minor axes differs from the $r_{\text{med}}$ by no more than a $\Delta$. We express this parameter using the following formula:

$$\Delta = \delta \cdot r_{\text{med}}, \tag{4.4}$$

where $\delta$ is a hyperparamer which actually defines the boundary in this case.




**(a)** A graph of the dependence of the percentage of the selected points on the set $\delta$ value for 3 videos.




**(b)** A graph of the dependence of the relative error of the algorithm on the set $\delta$ value for 3 videos.

**Figure 4.10:** Graphs of the dependence of the percentage of the selected points (Fig. 4.10a) and the relative error of the algorithm (Fig. 4.10b) on the set value of $\delta$ for 3 videos. Information about the videos on which these experiments were conducted is shown in Table 4.9.

To determine the best value for this parameter, we conducted a series of experiments on several different videos, for which the objects had different rotation speeds. The results of these experiments can be seen in Figure 4.10. All the videos on which experiments were

conducted in this section can be found on GitHub [37]. Information about these videos is also provided in Table 4.10.

As we can see in Figure 4.10a, it makes no sense to set the parameter value to a value greater than 0.3, since the percentage of selected ellipses remains almost the same over a long interval. This may indicate that all ellipses with approximately the same centers have already been selected and that only noise remains.

We set the value of the epsilon parameter at 0.3, because, based on Figure 4.10b, the lowest relative error of the algorithm is achieved for this value on average.

| Param. / Video name | ISO | Shutter speed (ns) | FPS | Angle (°) | Actual speed (RPM) | Calculated speed (RPM) | **Relative error (%)** |
|---|---|---|---|---|---|---|---|
| video0_87 | 3300 | 66666 | 30 | 0 | 128.57 ± 0.05 | 128.30 | **0.2 ± 0.002** |
| video0_89 | 3300 | 66666 | 30 | 0 | 166.70 ± 0.07 | 166.87 | **0.1 ± 0.003** |
| video0_88 | 1528 | 66666 | 30 | 0 | 55.40 ± 0.02 | 55.73 | **0.59 ± 0.001** |

**Table 4.10:** Information about the video parameters and the result of executing the algorithm on them with the optimal $\delta$ parameter set. The videos that correspond to the data from this table can be found in the GitHub repository [37].

## 4.7 Filtering by ellipse angle $\theta$

This angle shows how far the ellipse is deflected from the horizontal axis. We get the value of this angle in radians. In cases where the shooting of a rotating object takes place at an angle close to 0 degrees to the axis of rotation of the object, then we are dealing with ellipses which are almost circles. This leads to the fact that this angle becomes very difficult to determine, and subsequently to select the ellipses suitable for us.

After making a number of observations, we found out that the selection of ellipses for this parameter makes practically no changes to the final result of the algorithm.

For these reasons, filtering by this parameter is not performed.

## 4.8 Tuning the $\Psi$ parameter for angle selection

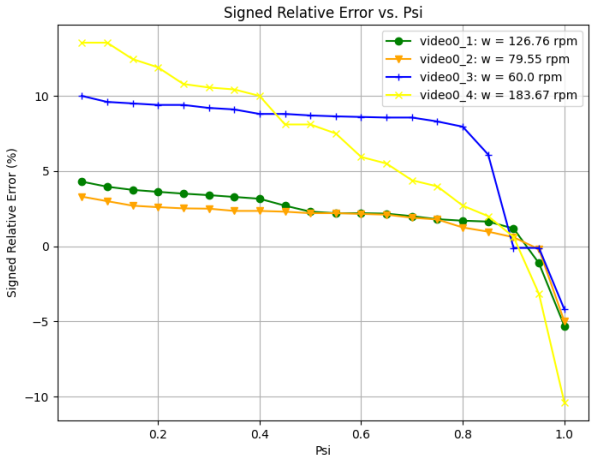



**Figure 4.11:** A graph of the dependence of the sign relative error of the algorithm, taking into account the sign of this error, on the set $\psi$ parameter for 4 different videos. The objects in these videos rotate at different rotational speeds. Information about the videos on which these experiments were conducted is shown in Table 4.11.

In Section 3.6.1, we described the process of selecting angles to further guess the angle of rotation of an object in a frame using selected angles. Recall that the condition for the angle to be considered selected was that its value should not be greater than $\Psi$. We can write this parameter using the following formula:

$$\Psi = \psi \times \alpha_{\text{med}_i} \tag{4.5}$$

where $\alpha_{\text{med}_i}$ is the median value for all angles corresponding to the $i$-th point and $\psi$ is a hyperparameter that we have to set.

To find the best value for this parameter, we conducted a series of experiments in which we tested the relative error with its sign that the algorithm made when setting a certain value of the $\psi$ parameter. All other hyperparameters were the same for each experiment on each of the videos. We demonstrate the results of these experiments in Figure 4.11. Based on the results of these experiments, the parameter $\psi$ has a value of 0.925. All the videos on which experiments were conducted in this section can be found on GitHub [38]. Information about these videos is also provided in Table 4.11.

| Param. / Video name | ISO | Shutter speed (ns) | FPS | Angle (°) | Actual speed (RPM) | Calculated speed (RPM) | **Relative error (%)** |
|---|---|---|---|---|---|---|---|
| video0_7 | 2024 | 66666 | 30 | 0 | 126.70 ± 0.05 | 126.34 | **0.33 ± 0.007** |
| video0_29 | 1687 | 66666 | 30 | 0 | 79.55 ± 0.03 | 79.77 | **0.28 ± 0.012** |
| video0_70 | 3300 | 66666 | 30 | 45 | 60.00 ± 0.02 | 60.08 | **0.13 ± 0.002** |
| video0_51 | 1687 | 66666 | 30 | 45 | 183.67 ± 0.07 | 183.95 | **0.15 ± 0.004** |

**Table 4.11:** Information about the video parameters and the result of executing the algorithm on them with the optimal $\psi$ parameter set. The videos that correspond to the data from this table can be found in the GitHub repository [38].

## 4.9 Effect of shooting angle

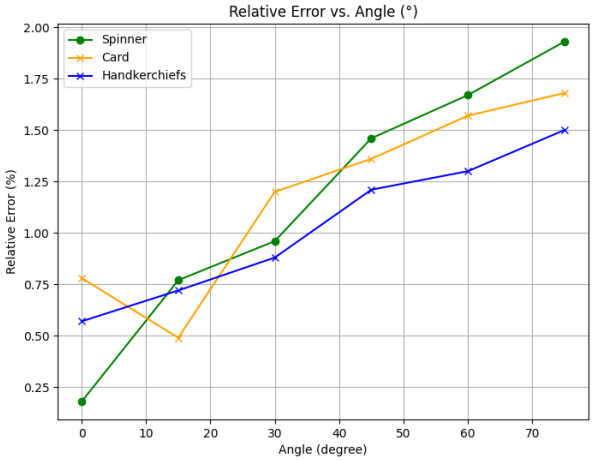



**Figure 4.12:** The graph shows the dependence of the relative error of the algorithm on the angle to the axis of rotation, under which the video was recorded, for 4 different videos. The data on which these graphs are based are in Table 4.12 for the spinner, in Table 4.13 for the card and in Table 4.6 for a pack of hangkerchiefs.

Since the algorithm assumes the possibility of recording objects on video at different angles to their axis of rotation, we conducted an experiment to determine how the recording angle affects the relative error of the algorithm. We show an example of what the frames from the experiment video look like for one object in Figure 4.13.

For the greater purity of the experiments, we tried to make sure that the only parameter that would distinguish each video was angle. However, it turned out to be quite difficult to achieve this for one of the parameters. This parameter was the rotation speed of the object. Therefore, in these experiments, the rotation speeds of the recorded objects can be considered the same only approximately. The camera parameters were set according to the same conditions described in Section 4.2.

The results of this experiment can be seen in Table 4.12 for the spinner, in Table 4.13 for the card and in Table 4.6 for the pack of handkerchiefs. We also visualize them in Figure 4.12. As we can observe, with an increase in the angle of deviation of the object from the axis of rotation during its video recording, the relative error of the algorithm tends to increase approximately linearly. However, even with a sufficiently large angle of view, it does not exceed 2%, which once again demonstrates the reliability of our algorithm. All the videos on which experiments were conducted in this section can be found on GitHub [39].

The increase in the algorithm's inaccuracy with the increase in the angle of inclination towards the axis of rotation of the object can be explained by the fact that as this angle increases, the surface area occupied by the object in the video decreases. Consequently, we have less data about the trajectory of its points. This is the reason for the increased inaccuracy of the algorithm.

**(a)** Video recording angle 0° **(b)** Video recording angle 15° **(c)** Video recording angle 30°

**(d)** Video recording angle 45° **(e)** Video recording angle 60° **(f)** Video recording angle 75°

**Figure 4.13:** An example of frames from a video used in an experiment on the effect of the recording angle on the relative error of the algorithm, for one of the objects.

## 4.10 Limitations

Despite its rather high efficiency, this algorithm has certain limitations. In other words, the efficiency of the algorithm is achieved only if the conditions described below are met:

| Param. / Video name | ISO | Shutter speed (ns) | FPS | Angle (°) | Actual speed (RPM) | Calculated speed (RPM) | **Relative error (%)** |
|---|---|---|---|---|---|---|---|
| video0_90 | 1561 | 66666 | 30 | 0 | 108.4 ± 0.30 | 108.20 | **0.18 ± 0.008** |
| video0_91 | 1561 | 66666 | 30 | 15 | 106.9 ± 0.20 | 106.00 | **0.77 ± 0.007** |
| video0_92 | 1561 | 66666 | 30 | 30 | 107.6 ± 0.30 | 106.57 | **0.96 ± 0.003** |
| video0_93 | 1561 | 66666 | 30 | 45 | 108.0 ± 0.43 | 106.42 | **1.46 ± 0.056** |
| video0_94 | 1561 | 66666 | 30 | 60 | 106.5 ± 0.42 | 104.70 | **1.67 ± 0.001** |
| video0_95 | 1561 | 66666 | 30 | 75 | 107.8 ± 0.43 | 105.70 | **1.93 ± 0.020** |

**Table 4.12:** Experimental data on the dependence of the relative error of the algorithm on the angle to the axis of rotation of the object under which it was recorded. Experiments are conducted for the **spinner**. The videos that correspond to the data from this table can be found in the GitHub repository [42].

| Param. / Video name | ISO | Shutter speed (ns) | FPS | Angle (°) | Actual speed (RPM) | Calculated speed (RPM) | **Relative error (%)** |
|---|---|---|---|---|---|---|---|
| video0_96 | 3300 | 66666 | 30 | 0 | 84.1 ± 0.03 | 84.76 | **0.78 ± 0.003** |
| video0_97 | 3300 | 66666 | 30 | 15 | 85.7 ± 0.03 | 85.28 | **0.49 ± 0.004** |
| video0_98 | 3300 | 66666 | 30 | 30 | 91.1 ± 0.03 | 92.20 | **1.2 ± 0.004** |
| video0_99 | 3300 | 66666 | 30 | 45 | 90.7 ± 0.03 | 89.46 | **1.36 ± 0.004** |
| video0_100 | 3300 | 66666 | 30 | 60 | 89.9 ± 0.03 | 91.32 | **1.57 ± 0.004** |
| video0_101 | 3300 | 66666 | 30 | 75 | 88.8 ± 0.03 | 90.30 | **1.68 ± 0.003** |

**Table 4.13:** Experimental data on the dependence of the relative error of the algorithm on the angle to the axis of rotation of the object under which it was recorded. Experiments are conducted for the **card**. The videos that correspond to the data from this table can be found in the GitHub repository [40].

| Param. / Video name | ISO | Shutter speed (ns) | FPS | Angle (°) | Actual speed (RPM) | Calculated speed (RPM) | **Relative error (%)** |
|---|---|---|---|---|---|---|---|
| video0_102 | 3300 | 66666 | 30 | 0 | 107.20 ± 0.03 | 107.82 | **0.57 ± 0.003** |
| video0_103 | 3300 | 66666 | 30 | 15 | 107.00 ± 0.03 | 106.30 | **0.72 ± 0.004** |
| video0_104 | 3300 | 66666 | 30 | 30 | 98.70 ± 0.03 | 97.83 | **0.88 ± 0.004** |
| video0_105 | 3300 | 66666 | 30 | 45 | 97.70 ± 0.03 | 96.56 | **1.21 ± 0.004** |
| video0_106 | 3300 | 66666 | 30 | 60 | 91.56 ± 0.03 | 92.76 | **1.30 ± 0.004** |
| video0_107 | 3300 | 66666 | 30 | 75 | 118.50 ± 0.03 | 120.29 | **1.50 ± 0.003** |

**Table 4.14:** Experimental data on the dependence of the relative error of the algorithm on the angle to the axis of rotation of the object under which it was recorded. Experiments are conducted for the **pack of handkerchiefs**. The videos that correspond to the data from this table can be found in the GitHub repository [41].

**(a)** An example of an unsuitable object to use the algorithm (the object has a fairly symmetrical structure).

**(b)** An example of a suitable object to use the algorithm (the object has multicolored stickers on its surface, which makes its structure less symmetrical).

**Figure 4.14:** Examples of suitable and unsuitable objects to use the algorithm.

1. According to the Nyquist sampling criterion, the frame rate of the camera must be at least twice of the rotational frequency. In this case, the maximum measurable speed for a given frame rate ($f$) is ($30\times f$) RPM. This condition imposes restrictions on the theoretically possible maximum rotation speed.
2. The algorithm is quite effective only for the rotation speeds of an object lying in the range of 10 to 190 revolutions per minute when shooting at an angle of 0 degrees to the axis of rotation and in the range from 20 to 190 revolutions per minute when shooting at any other angle not exceeding 75 degrees.
3. The rotating object whose velocity we want to determine is completely in the video.
4. During the first few seconds of shooting, the camera is in a stable position. We also consider a slight shaking of the hands that hold the phone when shooting to be a stable position.
5. The effectiveness of the algorithm also depends on the device on which the rotating object was recorded. However, about any particular phone that is being filmed, the algorithm achieves maximum efficiency at the maximum adjusted frame rate and shutter speed values.
6. The object in the video is sufficiently well illuminated by either an external light source or by setting the ISO value.
7. The structure of a rotating object is not symmetric or monotonous. For example, in Figure 4.14a we can see an object that is not suitable for the algorithm, because its structure is very symmetric. However, if we stick multicolored stickers on its surface (Firgure 4.14b), its structure will cease to be sufficiently symmetric. This object is already suitable for the algorithm.
8. The rotating object is in the focus of the cameras during shooting.

9. The rotating object in the video occupies approximately at least 20% of the surface.
10. The background on which the rotating object is being filmed is in a static position.

# Chapter 5

# Discussion, Conclusion and Future Work

The primary objective of this thesis was to develop a novel method for measuring the rotational speed of objects using a smartphone camera. Through the implementation of a dense tracking algorithm and an ellipse fitting technique, we successfully demonstrated a reliable and accurate approach to rotational speed measurement. Our proposed method leverages the accessibility and convenience of smartphone technology, providing an effective alternative to traditional noncamera-based methods, and also expands the scope of already invented camera-based methods.

The experimental results show that the algorithm works well under different conditions under which the input videos were shot, including the angle of the camera to the axis of rotation of the object, the speed of rotation of the object and its shape. As we can see, if all the conditions described in Section 4.10 are met, the algorithm does not allow an error greater than 2%. This low level of error persists even at significant angles of deviation from the axis of rotation, demonstrating the reliability of the described approach. Moreover, this method has proven its effectiveness in a wide range of rotation speeds. This result is also achieved through a good selection of hyperparameters such as the number of the considering frames $F_{\text{cons}}$, the number of selected points $N$, $\varepsilon_x$, $\varepsilon_y$, $\Delta$, and $\Psi$. It is important to note that our algorithm has also proven itself well in the presence of motion blur, which is a common problem with high-speed turns.

Despite promising results, the method has limitations. The accuracy of the algorithm can be affected by extreme lighting conditions and fast movement because the algorithm is very dependent on the optical RAFT algorithm used in it, which has problems tracking points moving at high speed, as well as in very dim or very bright lighting. Because of this, the algorithm has a limited range of speeds that it can measure with great accuracy. An important problem is also that this optical algorithm also does not cope well with objects that have a symmetrical structure. In addition, the method requires careful calibration of camera settings to optimize performance.

Moreover, one of the disadvantages of the algorithm is its execution time. On average, it takes about 60-100 seconds. This may be due to the fact that the algorithm must fit an ellipse for the trajectory of each point of usually 1000 selected points. To solve this problem, we can try using other methods to determine the trajectory of the points in the future.

Future research should also focus on enhancing the algorithm's robustness to handle more challenging scenarios, such as variable lighting conditions and high-speed rotations.

Exploring the integration of machine learning techniques could further improve the accuracy and adaptability of the method. Furthermore, expanding the application to measure the rotation speed of objects with 3D rotation could open new avenues for research and practical use.

In conclusion, this thesis presents a significant advancement in the field of rotational speed measurement using smartphone cameras. The proposed method offers a practical, cost-effective, and accessible solution, with the potential to impact various domains, from industrial monitoring to sports performance analysis. Continued research and development in this area will undoubtedly lead to further innovations and improvements.

# Bibliography


[1] Image source: "Code: Robotics". Retrieved from https://www.bbc.com/rd/sites/50335ff370b5c262af000004/assets/5b4f60f806d63eb5ca00541e/tennis-1920x1080-1381230.jpg (Accessed: May 2024).

[2] Image source: "Code: Robotics". Retrieved from https://www.thoughtco.com/thmb/l_G85oKpLwGfNOUiigGzfJA4WSI=/1500x0/filters:no_upscale():max_bytes(150000):strip_icc()/North_season-5a5f865dd92b0900361b3dd1.jpg (Accessed: May 2024).

[3] Image source: "The Return of the Rotary Engine.". Retrieved from https://res.cloudinary.com/engineering-com/image/upload/w_640,h_640,c_limit,q_auto,f_auto/ima1_ujhest.jpg (Accessed: May 2024).

[4] Image source: "Code: Robotics". Retrieved from https://megadepot.com/assets_images/depiction/resources/MD/what-is-a-tachometer-and-how-does-it-work/contact-tachometer-img4.jpg (Accessed: April 2024).

[5] Image source: "Encoder Products Company". Retrieved from https://megadepot.com/assets_images/depiction/resources/MD/what-is-a-tachometer-and-how-does-it-work/non-contact-tachometer-img1.jpg (Accessed: April 2024).

[6] Image source: "Code: Robotics". Retrieved from https://docs.idew.org/code-robotics/references/physical-inputs/wheel-encoders (Accessed: April 2024).

[7] Image source: "Encoder Products Company". Retrieved from https://www.encoder.com/hs-fs/hubfs/articles/what-is-an-encoder/rotary-encoder-exploded-rendering_1080x608.jpg?width=1080&name=rotary-encoder-exploded-rendering_1080x608.jpg (Accessed: April 2024).

[8] J. Le and A. Gevins, "Method to reduce blur distortion from EEG's using a realistic head model," in IEEE Transactions on Biomedical Engineering, vol. 40, no. 6, pp. 517-528, June 1993, doi: 10.1109/10.237671. keywords: Electroencephalography;Brain modeling;Scalp;Skull;Magnetic heads;Finite element methods;Magnetic resonance;Magnetic resonance imaging;Magnetic recording;Neurosurgery.

[9] S. R, J. R and S. M. A, "Computer vision approach for motion blur image restoration system," 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT), Delhi, India, 2023, pp. 1-9, doi: 10.1109/ICCCNT56998.2023.10306622. keywords: Photography;Visualization;Computer vision;Laplace equations;Surveillance;Image edge detection;Computer architecture;Deep Learning;Motion Blur;inception ResNetv2;Laplacian variance;deblur;Image Classification.

[10] Image source: "Code: Robotics. Wheel Encoders." (April 2024). Retrieved from `https://docs.idew.org/code-robotics/references/physical-inputs/wheel-encoders`, Online; Accessed: April 2024.

[11] K. W. Hylton, P. Mitchell, B. Van Hoy, and T. P. Karnowski, “Experiments and analysis for measuring mechanical motion with event cameras,” Electronic Imaging, vol. 33, pp. 333–1–333–8, Jan. 2021.

[12] G. Zhao, Y. Shen, N. Chen, P. Hu, L. Liu, and H. Wen, “High Speed Rotation Estimation with Dynamic Vision Sensors,” Sept. 2022, 2209.02205.

[13] J. Kolar, R. Spetlik, J. Matas, "EE3P: Event-based Estimation of Periodic Phenomena Properties", February 2024.

[14] Y. Wang, L. Wang and Y. Yan, "Rotational speed measurement through digital imaging and image processing," 2017 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), Turin, Italy, 2017, pp. 1-6, doi: 10.1109/I2MTC.2017.7969697. keywords: Velocity control;Rotors;Rotation measurement;Imaging;Standards;Image processing;Tachometers;rotational speed;measurement;tachometer;image processing;image similarity evaluation.

[15] T. Wang, Y. Yan, L. Wang and Y. Hu, "Rotational Speed Measurement Through Image Similarity Evaluation and Spectral Analysis," in IEEE Access, vol. 6, pp. 46718-46730, 2018, doi: 10.1109/ACCESS.2018.2866479. keywords: Velocity control;Rotation measurement;Imaging;Rotors;Pollution measurement;Tachometers;Transforms;Image processing;rotational speed measurement;spectral analysis;tachometer.

[16] Toby Verwimp, Alexandre Mauricio, Konstantinos Gryllias, Rotating machinery speed extraction through smartphone video acquisition from a radial viewpoint, Mechanical Systems and Signal Processing, Volume 205, 2023, 110836, ISSN 0888-3270, `https://doi.org/10.1016/j.ymssp.2023.110836`. (`https://www.sciencedirect.com/science/article/pii/S0888327023007446`)

[17] Hui Zhao, Zhong Su, Fuchao Liu, Chao Li, Qing Li, Ning Liu, "Extraction and Filter Algorithm of Roll Angular Rate for High Spinning Projectiles", Mathematical Problems in Engineering, vol. 2019, Article ID 3181727, 15 pages, 2019. `https://doi.org/10.1155/2019/3181727`

[18] Ash Turner. Android vs. Apple Market Share: Leading Mobile Operating Systems (OS) (Feb 2024). Retrieved from `https://www.bankmycell.com/blog/android-vs-apple-market-share/#:~:text=Android%20continues%20to%`

20reign%20in%20the%20global%20market%20with%20a,used%20by%20many%20smartphone%20brands., Online; Accessed: February 2024.

[19] Gabriel Gircenko. Kotlin vs. Java: All-purpose Uses and Android Apps. Retrieved from https://www.toptal.com/kotlin/kotlin-vs-java#:~:text=Kotlin%20is%20Google's%20preferred%20language,existing%20Android%20apps%20using%20Java., Online; Accessed: February 2024.

[20] Android for Developers. (2024-02-08). Camera intents. Retrieved from https://developer.android.com/media/camera/camera-intents, Online; Accessed: February 2024.

[21] Android for Developers. (2024-01-05). Choose a camera library. Retrieved from https://developer.android.com/media/camera/choose-camera-library, Online; Accessed: February 2024.

[22] Android for Developers. (2024-01-05). Camera2 overview. Retrieved from https://developer.android.com/media/camera/camera2, Online; Accessed: February 2024.

[23] Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In European Conference on Computer Vision, pages 402–419. Springer, 2020.

[24] Uni-Trend Technology (China) Co., Ltd., "UT370 Series Tachometers - UNI-T Meters | Test & Measurement Tools and Solutions," Aug. 2022

[25] Denis Gorbunov. Bachelor-thesis. [Online]. Available: https://github.com/LiXinGr/Bachelor-thesis/tree/main/Bachelor-thesis-Experiments/Experimental_videos_Section_4.2 (visited on 21/05/2024).

[26] Denis Gorbunov. Bachelor-thesis. [Online]. Available: https://github.com/LiXinGr/Bachelor-thesis/tree/main/Bachelor-thesis-Experiments/Experimental_videos_Section_4.2/Section_4.2.1 (visited on 21/05/2024).

[27] Denis Gorbunov. Bachelor-thesis. [Online]. Available: https://github.com/LiXinGr/Bachelor-thesis/tree/main/Bachelor-thesis-Experiments/Experimental_videos_Section_4.2/Section_4.2.1/videos_for_card (visited on 21/05/2024).

[28] Denis Gorbunov. Bachelor-thesis. [Online]. Available: https://github.com/LiXinGr/Bachelor-thesis/tree/main/Bachelor-thesis-Experiments/Experimental_videos_Section_4.2/Section_4.2.1/videos_for_pack (visited on 21/05/2024).

[29] Denis Gorbunov. Bachelor-thesis. [Online]. Available: https://github.com/LiXinGr/Bachelor-thesis/tree/main/Bachelor-thesis-Experiments/Experimental_videos_Section_4.2/Section_4.2.1/videos_for_spinner (visited on 21/05/2024).

[30] Denis Gorbunov. Bachelor-thesis. [Online]. Available: https://github.com/LiXinGr/Bachelor-thesis/tree/main/Bachelor-thesis-Experiments/Experimental_videos_Section_4.2/Section_4.2.2 (visited on 21/05/2024).

[31] Denis Gorbunov. Bachelor-thesis. [Online]. Available: https://github.com/LiXinGr/Bachelor-thesis/tree/main/Bachelor-thesis-Experiments/Experimental_videos_Section_4.2/Section_4.2.2/videos_for_card (visited on 21/05/2024).

[32] Denis Gorbunov. Bachelor-thesis. [Online]. Available: https://github.com/LiXinGr/Bachelor-thesis/tree/main/Bachelor-thesis-Experiments/Experimental_videos_Section_4.2/Section_4.2.2/videos_for_pack (visited on 21/05/2024).

[33] Denis Gorbunov. Bachelor-thesis. [Online]. Available: https://github.com/LiXinGr/Bachelor-thesis/tree/main/Bachelor-thesis-Experiments/Experimental_videos_Section_4.2/Section_4.2.2/videos_for_spinner (visited on 21/05/2024).

[34] Denis Gorbunov. Bachelor-thesis. [Online]. Available: https://github.com/LiXinGr/Bachelor-thesis/tree/main/Bachelor-thesis-Experiments/Experimental_videos_Section_4.3 (visited on 21/05/2024).

[35] Denis Gorbunov. Bachelor-thesis. [Online]. Available: https://github.com/LiXinGr/Bachelor-thesis/tree/main/Bachelor-thesis-Experiments/Experimental_videos_Section_4.4 (visited on 21/05/2024).

[36] Denis Gorbunov. Bachelor-thesis. [Online]. Available: https://github.com/LiXinGr/Bachelor-thesis/tree/main/Bachelor-thesis-Experiments/Experimental_videos_Section_4.5 (visited on 21/05/2024).

[37] Denis Gorbunov. Bachelor-thesis. [Online]. Available: https://github.com/LiXinGr/Bachelor-thesis/tree/main/Bachelor-thesis-Experiments/Experimental_videos_Section_4.6 (visited on 21/05/2024).

[38] Denis Gorbunov. Bachelor-thesis. [Online]. Available: https://github.com/LiXinGr/Bachelor-thesis/tree/main/Bachelor-thesis-Experiments/Experimental_videos_Section_4.8 (visited on 21/05/2024).

[39] Denis Gorbunov. Bachelor-thesis. [Online]. Available: https://github.com/LiXinGr/Bachelor-thesis/tree/main/Bachelor-thesis-Experiments/Experimental_videos_Section_4.9 (visited on 21/05/2024).

[40] Denis Gorbunov. Bachelor-thesis. [Online]. Available: https://github.com/LiXinGr/Bachelor-thesis/tree/main/Bachelor-thesis-Experiments/Experimental_videos_Section_4.9/videos_for_card (visited on 21/05/2024).

[41] Denis Gorbunov. Bachelor-thesis. [Online]. Available: https://github.com/LiXinGr/Bachelor-thesis/tree/main/Bachelor-thesis-Experiments/Experimental_videos_Section_4.9/videos_for_pack (visited on 21/05/2024).

[42] Denis Gorbunov. Bachelor-thesis. [Online]. Available: https://github.com/LiXinGr/Bachelor-thesis/tree/main/Bachelor-thesis-Experiments/Experimental_videos_Section_4.9/videos_for_spinner (visited on 21/05/2024).

[43] Denis Gorbunov. Bachelor-thesis. [Online]. Available: https://github.com/LiXinGr/Bachelor-thesis/tree/main/Bachelor-thesis-Experiments (visited on 21/05/2024).

# Appendix A

## Collected dataset of suitable videos

One of the main parts of this work was the collection of videos for the DGBP dataset. The videos for this dataset were shot using the setup described in Section 4.1. Also, each of these videos complies with the restrictions described in Section 4.10. A total of 107 videos were shot. Each video has a duration that ranges from 2 to 5 seconds.

Since the views were recorded under different external conditions, some of the parameters, for example ISO, may have different values. That is because these parameters were selected each time in accordance with the external conditions that were present when shooting the video.

Subsequently, all these videos were used to conduct various kinds of experiment, which are described in Chapter 4.

The parameters of each of the recorded videos are in Table A.1. It also contains some information about the object that was recorded. To shoot each video, we used a camera with a shutter speed value of 66666 nanoseconds, since this is the imaginary value that this parameter can take on the camera of the phone that was used for recording. For the same reason, the fps value was set to 30. It is worth noting that in column "The angle of video shooting to the axis of rotation of the object (°)" in the Table A.1 the angle values have not been calculated accurately and are only approximate.

The dataset is located on our GitHub page, which can be accessed by clicking on the following link:

`https://github.com/LiXinGr/Bachelor-thesis/tree/main/Bachelor-thesis-Dataset`.

| Param. / Video name | ISO | Shutter speed (ns) | FPS | The angle of video shooting to the axis of rotation of the object (°) | Rotation speed of the object (RPM) | Rotated object |
|---|---|---|---|---|---|---|
| **video0_1** | 2024 | 66666 | 30 | 0 | 10.30 ± 0.01 | Card |
| **video0_2** | 2024 | 66666 | 30 | | 25.00 ± 0.01 | |
| **video0_3** | 2024 | 66666 | 30 | | 45.00 ± 0.02 | |
| **video0_4** | 2024 | 66666 | 30 | | 64.28 ± 0.02 | |
| **video0_5** | 2024 | 66666 | 30 | | 84.10 ± 0.03 | |
| **video0_6** | 2024 | 66666 | 30 | | 105.88 ± 0.04 | |
| **video0_7** | 2024 | 66666 | 30 | | 126.70 ± 0.05 | |
| **video0_8** | 2024 | 66666 | 30 | | 149.38 ± 0.06 | |
| **video0_9** | 2024 | 66666 | 30 | | 175.60 ± 0.07 | |
| **video0_10** | 2024 | 66666 | 30 | | 201.00 ± 0.08 | |
| **video0_11** | 2024 | 66666 | 30 | | 225.40 ± 0.09 | |
| **video0_12** | 2024 | 66666 | 30 | | 257.60 ± 1.00 | |
| **video0_13** | 2024 | 66666 | 30 | | 276.84 ± 1.01 | |
| **video0_14** | 3300 | 66666 | 30 | 0 | 17.14 ± 0.01 | Pack of handkerchiefs |
| **video0_15** | 3300 | 66666 | 30 | | 32.70 ± 0.01 | |
| **video0_16** | 3300 | 66666 | 30 | | 50.00 ± 0.02 | |
| **video0_17** | 3300 | 66666 | 30 | | 69.76 ± 0.03 | |
| **video0_18** | 3300 | 66666 | 30 | | 83.30 ± 0.03 | |
| **video0_19** | 3300 | 66666 | 30 | | 100.00 ± 0.04 | |
| **video0_20** | 3300 | 66666 | 30 | | 121.40 ± 0.05 | |
| **video0_21** | 3300 | 66666 | 30 | | 146.34 ± 0.06 | |
| **video0_22** | 3300 | 66666 | 30 | | 163.63 ± 0.06 | |
| **video0_23** | 3300 | 66666 | 30 | | 222.22 ± 0.09 | |
| **video0_24** | 3300 | 66666 | 30 | | 260.86 ± 1.00 | |
| **video0_25** | 3300 | 66666 | 30 | | 328.47 ± 1.30 | |
| **video0_26** | 1687 | 66666 | 30 | 0 | 11.19 ± 0.01 | Spinner |
| **video0_27** | 1687 | 66666 | 30 | | 29.00 ± 0.01 | |
| **video0_28** | 1687 | 66666 | 30 | | 48.87 ± 0.02 | |
| **video0_29** | 1687 | 66666 | 30 | | 79.55 ± 0.03 | |
| **video0_30** | 1687 | 66666 | 30 | | 96.00 ± 0.04 | |
| **video0_31** | 1687 | 66666 | 30 | | 112.31 ± 0.05 | |
| **video0_32** | 1687 | 66666 | 30 | | 128.23 ± 0.05 | |
| **video0_33** | 1687 | 66666 | 30 | | 143.20 ± 0.06 | |
| **video0_34** | 1687 | 66666 | 30 | | 166.80 ± 0.07 | |
| **video0_35** | 1687 | 66666 | 30 | | 189.33 ± 0.07 | |
| **video0_36** | 1687 | 66666 | 30 | | 220.87 ± 0.08 | |
| **video0_37** | 1687 | 66666 | 30 | | 240.10 ± 0.09 | |
| **video0_38** | 1687 | 66666 | 30 | | 257.14 ± 1.00 | |
| **video0_39** | 1687 | 66666 | 30 | | 281.25 ± 1.01 | |
| **video0_40** | 1687 | 66666 | 30 | | 302.52 ± 1.02 | |
| **video0_41** | 1687 | 66666 | 30 | | 327.27 ± 1.03 | |

| | | | | | | |
|---|---|---|---|---|---|---|
| **video0_42** | 2024 | 66666 | 30 | 45 | 15.00 ± 0.01 | Card |
| **video0_43** | 2024 | 66666 | 30 | | 36.10 ± 0.01 | |
| **video0_44** | 2024 | 66666 | 30 | | 51.90 ± 0.02 | |
| **video0_45** | 2024 | 66666 | 30 | | 70.00 ± 0.03 | |
| **video0_46** | 2024 | 66666 | 30 | | 91.83 ± 0.03 | |
| **video0_47** | 2024 | 66666 | 30 | | 109.10 ± 0.04 | |
| **video0_48** | 2024 | 66666 | 30 | | 128.57 ± 0.05 | |
| **video0_49** | 2024 | 66666 | 30 | | 147.50 ± 0.06 | |
| **video0_50** | 2024 | 66666 | 30 | | 163.63 ± 0.06 | |
| **video0_51** | 2024 | 66666 | 30 | | 183.67 ± 0.07 | |
| **video0_52** | 2024 | 66666 | 30 | | 205.50 ± 0.08 | |
| **video0_53** | 2024 | 66666 | 30 | | 230.77 ± 0.09 | |
| **video0_54** | 2024 | 66666 | 30 | | 259.00 ± 1.00 | |
| **video0_55** | 2024 | 66666 | 30 | | 295.00 ± 1.02 | |
| **video0_56** | 3300 | 66666 | 30 | 45 | 28.57 ± 0.01 | Pack of handkerchiefs |
| **video0_57** | 3300 | 66666 | 30 | | 44.20 ± 0.02 | |
| **video0_58** | 3300 | 66666 | 30 | | 63.35 ± 0.02 | |
| **video0_59** | 3300 | 66666 | 30 | | 78.30 ± 0.03 | |
| **video0_60** | 3300 | 66666 | 30 | | 98.29 ± 0.03 | |
| **video0_61** | 3300 | 66666 | 30 | | 121.62 ± 0.04 | |
| **video0_62** | 3300 | 66666 | 30 | | 145.16 ± 0.05 | |
| **video0_63** | 3300 | 66666 | 30 | | 168.22 ± 0.06 | |
| **video0_64** | 3300 | 66666 | 30 | | 200.00 ± 0.07 | |
| **video0_65** | 3300 | 66666 | 30 | | 243.20 ± 0.08 | |
| **video0_66** | 3300 | 66666 | 30 | | 300.10 ± 0.09 | |
| **video0_67** | 3300 | 66666 | 30 | 45 | 13.00 ± 0.01 | Spinner |
| **video0_68** | 3300 | 66666 | 30 | | 30.10 ± 0.01 | |
| **video0_69** | 3300 | 66666 | 30 | | 43.95 ± 0.02 | |
| **video0_70** | 3300 | 66666 | 30 | | 60.00 ± 0.02 | |
| **video0_71** | 3300 | 66666 | 30 | | 73.77 ± 0.03 | |
| **video0_72** | 3330 | 66666 | 30 | | 89.50 ± 0.03 | |
| **video0_73** | 3330 | 66666 | 30 | | 105.88 ± 0.04 | |
| **video0_74** | 3330 | 66666 | 30 | | 126.76 ± 0.05 | |
| **video0_75** | 3330 | 66666 | 30 | | 144.00 ± 0.05 | |
| **video0_76** | 3330 | 66666 | 30 | | 162.63 ± 0.06 | |
| **video0_77** | 3330 | 66666 | 30 | | 182.70 ± 0.07 | |
| **video0_78** | 3330 | 66666 | 30 | | 202.24 ± 0.08 | |
| **video0_79** | 3330 | 66666 | 30 | | 216.39 ± 0.08 | |
| **video0_80** | 3330 | 66666 | 30 | | 250.00 ± 1.00 | |
| **video0_81** | 3330 | 66666 | 30 | | 261.62 ± 1.00 | |
| **video0_82** | 3330 | 66666 | 30 | | 294.68 ± 1.02 | |
| **video0_83** | 3330 | 66666 | 30 | | 327.42 ± 1.03 | |
| **video0_84** | 3330 | 66666 | 30 | | 360.30 ± 1.04 | |
| **video0_85** | 3330 | 66666 | 30 | | 400.89 ± 1.06 | |
| **video0_86** | 3330 | 66666 | 30 | | 439.36 ± 1.07 | |

| | | | | | | |
|---|---|---|---|---|---|---|
| **video0_87** | 3300 | 66666 | 30 | 0 | 128.57 ± 0.05 | Spinner |
| **video0_88** | 1528 | 66666 | 30 | 0 | 55.40 ± 0.02 | Spinner |
| **video0_89** | 3300 | 66666 | 30 | 0 | 166.70 ± 0.07 | Spinner |
| **video0_90** | 1561 | 66666 | 30 | 0 | 108.40 ± 0.30 | Card |
| **video0_91** | 1561 | 66666 | 30 | 15 | 106.90 ± 0.20 | |
| **video0_92** | 1561 | 66666 | 30 | 30 | 107.60 ± 0.30 | |
| **video0_93** | 1561 | 66666 | 30 | 45 | 108.00 ± 0.43 | |
| **video0_94** | 1561 | 66666 | 30 | 60 | 106.50 ± 0.42 | |
| **video0_95** | 1561 | 66666 | 30 | 75 | 107.80 ± 0.43 | |
| **video0_96** | 3300 | 66666 | 30 | 0 | 84.1 ± 0.03 | Pack of handkerchiefs |
| **video0_97** | 3300 | 66666 | 30 | 15 | 85.70 ± 0.03 | |
| **video0_98** | 3300 | 66666 | 30 | 30 | 91.10 ± 0.03 | |
| **video0_99** | 3300 | 66666 | 30 | 45 | 90.70 ± 0.03 | |
| **video0_100** | 3300 | 66666 | 30 | 60 | 89.90 ± 0.03 | |
| **video0_101** | 3300 | 66666 | 30 | 75 | 88.80 ± 0.03 | |
| **video0_102** | 3300 | 66666 | 30 | 0 | 107.20 ± 0.03 | Spinner |
| **video0_103** | 3300 | 66666 | 30 | 15 | 107.00 ± 0.03 | |
| **video0_104** | 3300 | 66666 | 30 | 30 | 98.70 ± 0.03 | |
| **video0_105** | 3300 | 66666 | 30 | 45 | 97.70 ± 0.03 | |
| **video0_106** | 3300 | 66666 | 30 | 60 | 91.56 ± 0.03 | |
| **video0_107** | 3300 | 66666 | 30 | 75 | 118.50 ± 0.03 | |

**Table A.1:** The table describes the parameters of each video from our dataset.

# Appendix B

# Attached files

The folder with the attached files contains:

1. `app-debug.apk` is an Android phone application described in Chapter 2, which can be installed on any Android device that meets the requirements described in Section 2.4.
2. `Camera2APINew` folder is an Adnroid phone application project together with the source codes for this application described in Chapter 2.
3. `MFT` folder is the folder containing the source codes for the algorithm described in Chapter 3. In order to run the algorithm for user's video, the user needs to execute the following command within `MFT` directory:

```
python3 demo.py  --video=<path_to_video_file>
```

All these files are located on our GitHub at the link:

`https://github.com/LiXinGr/Bachelor-thesis/tree/main/Files`

# Appendix C

## Acronyms

**BSCZT** Bilinear Sequence Chirp Z-Transform. 7

**BSCZT-KF** Bilinear Sequence Chirp Z-Transform with Kalman Filtering. 7

**CNN** Convolutional Neural Network. 18

**CZT** Chirp Z-Transform. 7

**FPS** Frames Per Second. 17, 35, 36, 38, 39, 41, 42, 44, 45, 47, 49, 60

**ISO** International Organization for Standardization. 12, 14, 15, 33, 35, 36, 38, 39, 41, 42, 44, 45, 47, 49, 50, 60

**LED** Light Emitting Diode. 5

**MP** Megapixels. 33

**ns** Nanoseconds. 16, 35, 36, 38, 39, 41, 42, 44, 45, 47, 49, 60

**PDAF** Phase Detection Autofocus. 33

**RAFT** Recurrent All-Pairs Field Transforms. 18, 19, 21, 26, 28, 33, 38, 40, 53

**RPM** Revolutions Per Minute. 7, 15, 16, 29, 32, 35–39, 41, 42, 44, 45, 47, 49, 50, 60

**SS** Shutter speed. 16

**um** Micrometers. 33