# Object detection in motion-blurred images
# IACV Project

Angela Remolina
Politecnico di Milano
angelasofia.remolina@mail.polimi.it

Semjon Tukmatsov
Politecnico di Milano
semjon.tukmatsov@mail.polimi.it

## Abstract

*This report presents a comprehensive study on the processing of blurred images, focusing on the novel application of alpha-matting techniques and the extraction of contours of moving objects. The challenge of handling motion blur in images is a significant hurdle in various fields such as surveillance, autonomous navigation, and real-time video analysis.*

*The approach taken introduces a refined alpha-matting method, an advanced technique typically used in image editing for extracting a subject from its background. An adaptation of this method is made to effectively separate moving objects from their blurred backgrounds, thereby enhancing the clarity and definition of the objects.*

*In addition to alpha-matting, it is explored advanced contour extraction techniques to accurately trace the outlines of moving objects in blurred images. Traditional edge detection methods are often inadequate in dealing with the ambiguities presented by motion blur. The method presented utilizes gradient-based techniques to overcome these challenges, offering a more robust and accurate contour extraction process.*

*The report provides an analysis of the performance of these techniques through a series of experiments conducted on various blurred images. The results demonstrate a significant improvement in object detection and contour extraction in motion-blurred scenarios compared to conventional methods.*

## 1. Introduction

Object detection in motion-blurred images presents a significant challenge in the field of computer vision. This challenge stems from the degradation of image quality caused by motion blur, a common occurrence in dynamic environments. Motion blur occurs when the relative motion between a camera and an object (or the environment) is rapid compared to the exposure time of the camera. This results in images where objects are smeared or blurred along the direction of relative motion, reducing the clarity and distinctness of the object features.

The problem of object detection in motion-blurred images is particularly relevant in various real-world applications. For instance, in surveillance systems, where monitoring fast-moving objects, clear object detection is crucial for security and monitoring purposes. Similarly, in the field of sports analytics and wildlife monitoring, where subjects are frequently moving quickly, effectively detecting and tracking objects in motion-blurred images can provide valuable insights and data.

Recognizing the contours of moving objects in motion-blurred images is highly relevant due to its critical role in enhancing the accuracy and reliability of various computer vision applications. This capability is essential in autonomous navigation systems, such as self-driving cars and drones, where accurately identifying and responding to moving objects despite motion blur is crucial for safety and effective operation. Additionally, in fields like surveillance, sports analytics, and robotics, the ability to discern moving object contours in less-than-ideal visual conditions can significantly improve the performance and functionality of systems relying on real-time image analysis. Thus, addressing this challenge not only advances the field of computer vision but also has wide-ranging practical implications in today's technology-driven world.

## 2. Definitions

Before the beginning of the study it is needed to clarify each term that is relevant to the matter of study. Some of these terms are enumerated in the next section.

1. **Motion-blurred images:** In the context of the project, a motion-blurred image, is the picture of a moving object taken with long exposure. In that sense, in the capture it can be identified a transparent "trail" that shows the trajectory of the movement. That can be achieved,

since long exposure allows the camera shutter (component that controls the duration for which light reaches the sensor) to be open during the whole time that the object is moving.

2. **Alpha-matting:** As it is stated by Marco Forte and François Pitié, "Alpha matting refers to the problem of extracting the opacity mask (alpha matte) of an object in an image." [2]. With this process the background can be separated from the foreground, which is useful when it is needed to put objects on top of other backgrounds, or how it is used in this project to obtain the contours of the object (along with other techniques since motion-blurred images are used).

3. **Threshold:** In the context of image processing and thresholding techniques, a threshold is a value used to separate elements of an image into two categories based on their intensity or color. It is a fundamental concept used in various image processing tasks, such as image segmentation, object detection, and noise removal.

4. **Trimap:** A trimap is a three-valued mask that divides an image into three regions: foreground, background, and unknown. The known regions, foreground and background, are represented by white and black pixels respectively, while the unknown region is represented by gray pixels.

5. **Sobel operator:** The Sobel operator is a widely used edge detection algorithm in image processing and computer vision. It performs a 2D spatial gradient measurement on an image to highlight areas of significant intensity change, which typically correspond to edges or boundaries between different objects or regions in the image. Mathematically, the Sobel operator computes the gradient magnitude at each pixel as the square root of the sum of the squares of the horizontal and vertical gradient components. This gradient magnitude represents the strength of the edge at that pixel.

## 3. State of art

Having a clear set of concepts, now relevant topics to the project can be analyzed from past research of other authors related to object contour detection in images, alpha-matting, and others. The following are key works to provide essential context for understanding the project's scope and limitations.

### 3.1. A Closed Form Solution to Natural Image Matting [1]

This groundbreaking contribution remains a foundation in this field. The paper from these authors introduced an innovative approach to image matting, where they derived a closed-form solution for estimating alpha values directly from natural images. By exploiting the assumptions on foreground and background colors, they addressed the problem of alpha estimation. Their method has enabled accurate separation of foreground and background objects without the need for human interaction with user scribbles.

### 3.2. On the Apparent Transparency of a Motion Blurred Object [5]

The work by Professor Caglioti and Giusti builds upon Levin's approach (as mentioned earlier). In their article, they explore techniques for capturing frames that describe an object's motion. Their insights significantly influence the approach taken in this project, enhancing it by eliminating the need for user-provided scribbles.

Moreover, Caglioti and Giusti's research sheds light on the apparent transparency of motion-blurred objects, revealing valuable properties of alpha mattes. These findings not only contribute to the field of image matting but also have practical implications for applications such as blurred image interpretation.

### 3.3. Pymatting: A python library for alpha matting [3]

More recently, Thomas Germer, Tobias Uelwer, Stefan Conrad, and Stefan Harmeling developed **PyMatting**, a Python library specifically focused on alpha matting. Their work, offers practical tools for handling transparency and foreground extraction. PyMatting builds upon the principles of the Closed Form described by Levin [1] and introducing other techniques from other authors. The developed approach for this project uses this library for the alpha estimation with closed form.

## 4. Proposed approach

To obtain the contour of a moving object, an approach is proposed and structured in the following diagram, fig. 1.

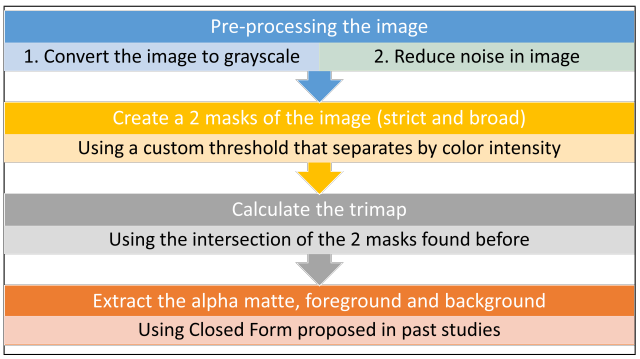


Figure 1. Proposed algorithm to obtain the alpha matte.

Along with this process, a parallel approached is taken into account to obtain better defined outlines of the moving object, this second approach can be seen in fig. 2.

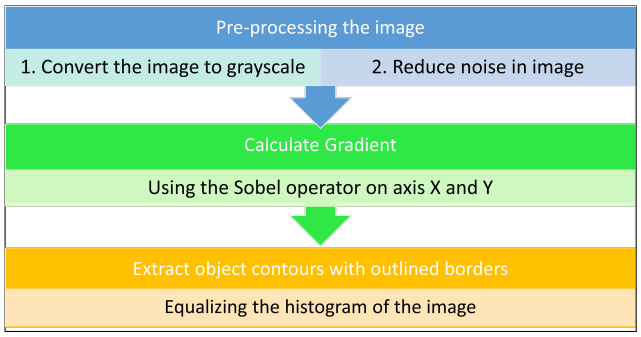


Figure 2. Proposed algorithm to obtain an outline of the object.

## 5. Implementation

Python language programming was used to implement the before mentioned algorithms along with open source tools such as OpenCV (Open Source Computer Vision) [4] and Pymatting libraries [3].

### 5.1. Alpha matting algorithm

To show the step by step implementation the alpha matting algorithm, let's consider this image of a black pencil case rotating approximately 30 degrees to the right, fig. 3.

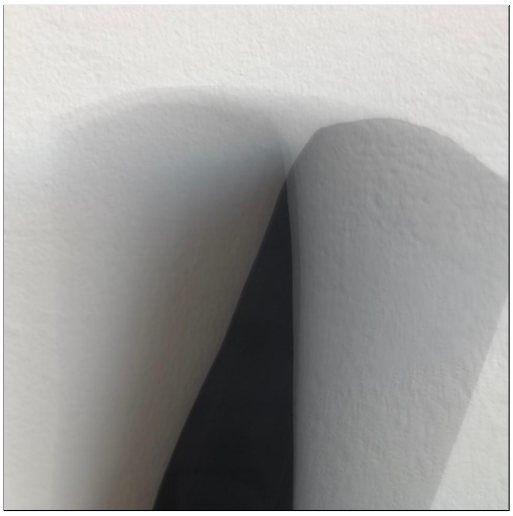

Figure 3. Pencil case, original image

The method *threshold()* from OpenCV is used to obtain 2 different masks for the image. First it is applied a threshold of 200 to the image, each pixel's intensity value (for gray-scale images) is compared against this threshold value. Pixels with intensity higher than 200 are assigned one category (white or foreground), while pixels with values lower than 200 are assigned to the other category (black or background). That is made to obtain the first mask (called broad mask), and then the same process it is repeated but now assigning the threshold value to 50 making it more strict, obtaining the second mask (called strict mask). The following code shows how that is done, and the result can be seen in fig. 4.

```
# Load image in gray-scale
img = cv2.imread("path", cv2.IMREAD_GRAYSCALE)

# Perform and average to reduce noise
img_clean = cv2.medianBlur(img, BLUR_KERNEL_SIZE)

THRESH_VALUE_BROAD = 200
THRESH_VALUE_STRICT = 50
THRESH_MAX_VALUE = 255

# Create broad mask
_, broad_mask = cv2.threshold(img_clean,
    THRESH_VALUE_BROAD,
    THRESH_MAX_VALUE,
    cv2.THRESH_BINARY_INV
)

# Create strict mask
_, strict_mask = cv2.threshold(img_clean,
    THRESH_VALUE_STRICT,
    THRESH_MAX_VALUE,
    cv2.THRESH_BINARY_INV
)
```

Figure 4. Broad mask and strict mask for pencil case image

It is possible to intersect these 2 masks obtained, or, in other words, find all the pixels that are different from one another, and take that as the trimap, overlapping the strict mask on top. That can be achieved like it is shown in the next block code, and the resulting image can be seen in figure 5.

```
# Generate Trimap
trimap = broad_mask - strict_mask

# set white pixels to gray
trimap[trimap == 255] = 153

# overlap the strict mask with the black pixels
trimap[strict_mask == 255] = 255
```

Figure 5. Trimap for pencil case image

To conclude the alpha matting algorithm and extract the alpha matte, background and foreground, the method **estimate_alpha_cf()** from Pymatting library is used. As mentioned before, this methods calculates the alpha values based on the image itself and a provided trimap using the Closed-Form Alpha Matting technique proposed [1]. The python code is presented below and the results can be seen in figure 6 and 7.

```
# alpha matte
alpha = estimate_alpha_cf(img_clean, trimap)

# Foreground and Background
fore, back = estimate_foreground_cf(img_clean,
alpha, return_background=True)
```

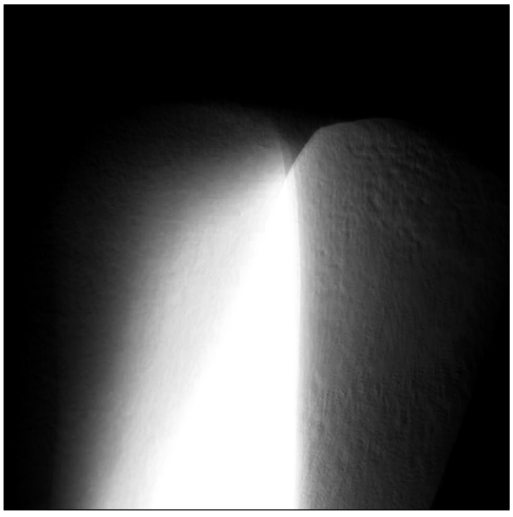

Figure 6. Alpha matte for pencil case image

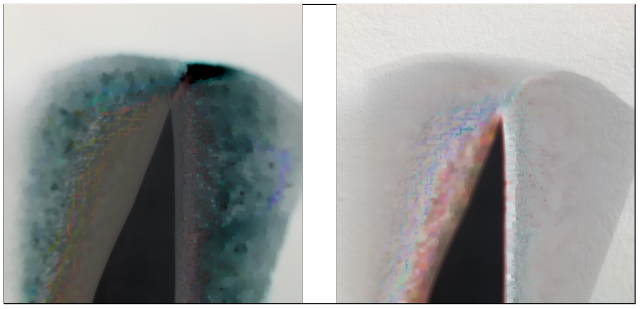

Figure 7. Foreground and background for pencil case image

### 5.2. Gradient-Outline algorithm

To show the implementation of Gradient-Outline algorithm let's consider an experimental image of bouncing ball illustrated on figure 8.

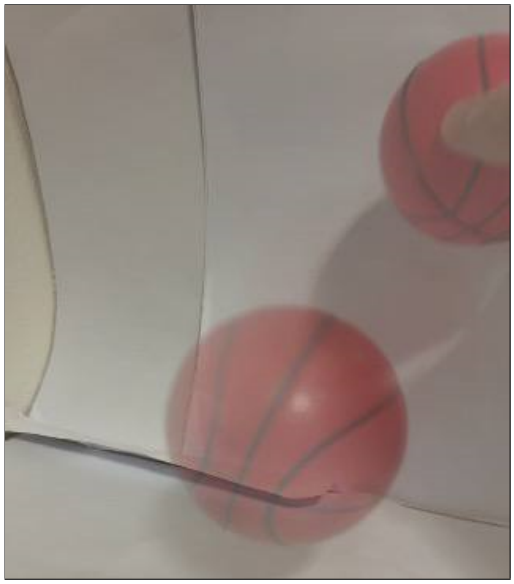

Figure 8. Bouncing ball, original image

Before computing gradient the image needs to be pre-processed. First of all, the image needs to be converted to grayscale for reduction of computational complexity by reducing amount of channels in image, and only intensity of an image is used for computation of gradient. Grayscale conversion emphasizes changes in intensity, making it easier to calculate and interpret gradients. After convertion Gausian-blur can be implemented to reduce noise on the image.
The following code shows how it is done. The result is presented on figure 9.

```
# Image download
image_path = 'bouncing_ball.png'
image = cv2.imread(image_path, cv2.IMREAD_COLOR)

# Transition into shades of gray
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Gauss Blur implementation
blurred = cv2.GaussianBlur(gray, (5, 5), 0)
```

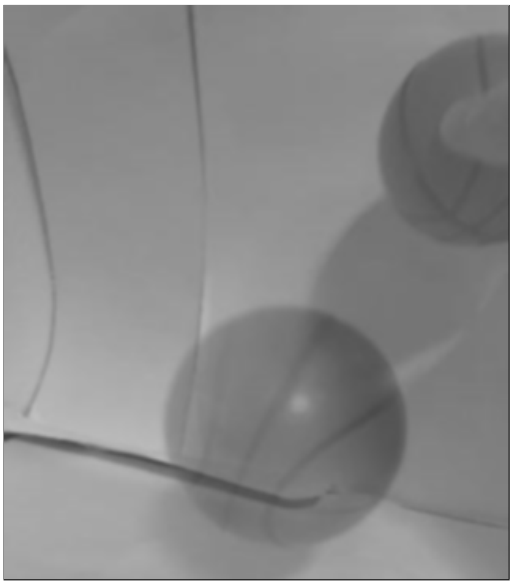

Figure 9. Bouncing ball, pre-processed image

Next step is exactly computing gradient. There are different methods to do this, for example Laplacian Derivatives and Sobel Derivatives. The Sobel Derivatives are used, because it is a joint Gaussian smoothing plus differentiation operation, so it is more resistant to noise, which is still on the picture even after usage of Gaussian blur.
In OpenCV direction of derivatives to be taken can be specified. So it is needed to get the resulting gradient after getting derivatives in two directions. It is calculated using following formula.

$$gradient = \sqrt{{Sobel_x}^2 + {Sobel_y}^2} \tag{1}$$

After computing gradient it is normalized and then converted into negative for better visualization, because as the result Sobel operator returns an image, where contours are white. In OpenCV it can be done using bitwise_not(img).

The implementation of this step in code is presented below, the result is in the figure 10.

```
# Sobel operator implementation for X and Y
sobelx = cv2.Sobel(blurred, cv2.CV_64F,
1, 0, ksize=5)
sobely = cv2.Sobel(blurred, cv2.CV_64F,
0, 1, ksize=5)

# Gradient calculation
gradient = np.sqrt(sobelx**2 + sobely**2)

# Gradient normalization
gradient = np.uint8(gradient / gradient.max() *
*255)

# Invert colors for better visualization
inverted_edges = cv2.bitwise_not(gradient)
```

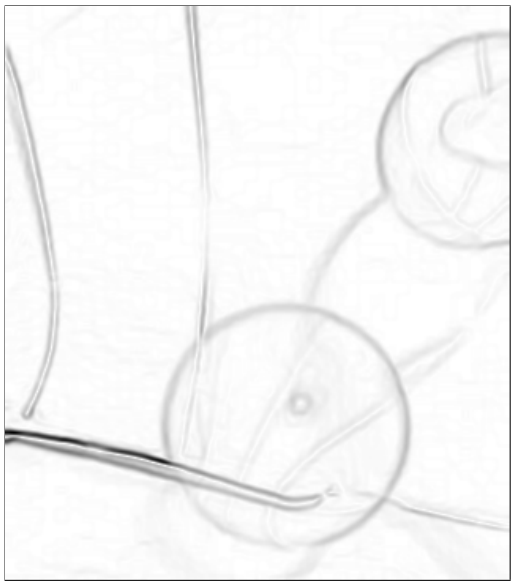

Figure 10. Bouncing ball, gradient

On the figure 10 it can be already seen contours of moving object, but they are not bright and do not stand out too clearly against the background. To improve the contrast in an image, an Histogram Equalization is used. Equalization implies mapping one distribution (the given histogram) to another distribution (a wider and more uniform distribution of intensity values) so the intensity values are spread over the whole range. With OpenCV it can be done using equalizeHist(img). In the following code this function is implemented and results is on the figure 11. As it can be seen contours became more visible, but background artefacts are also more visible. So depending on the purpose this step can be skipped.

```
#increase intensity
equalized = cv2.equalizeHist(inverted_edges)
```

Figure 11. Bouncing ball, equalized gradient

## 6. Experiments

**Datasets** The concept of a *motion-blurred image* was clarified before in the definitions section, to establish explicitly that the project does not analyze blurred image for a shaken camera, or a not focused one. In all the images that were taken for the project, the camera was still, and the object was moving.
Part of images used for experiments were taken from the work by Professor Caglioti and Giusti[5]. Parameters of those images are unknown. Other used pictures (bouncing ball, ball, Rubik's cube and pencil case) were taken by smartphone camera, iPhone to be exact, but using default camera app does not allow to make long exposure photos with specific exposure.
For that reason, apps "Slow Camera" and "ProCam 8" were used for this pictures with exposure time of 2 seconds. All freshly photographed images have resolution of 960 x 1280 pixels (they were cropped in for this paper).

**Experiments setup.** For the alpha extraction algorithm it is needed to set some parameters that may influence the performance of the algorithm according to each image. Those parameters are listed below:
Note: the names mentioned in capital bold letters are the name of the constants in the python code.

1. **THRESH_VALUE_BROAD**: This value represents the threshold for strict mask. It is an integer number from 0 to 255 that states if pixels will be assigned to foreground or background in the mask. This number is higher than the STRICT mask, because it will result with a bigger countour for the object.

2. **THRESH_VALUE_STRICT**: This value represents the threshold for broad mask. It is an integer number from 0 to 255 that states if pixels will be assigned to foreground or background in the mask. This number is lower than the BROAD mask, because it will result with the pixels that might have never changed in the image representing solid bits of the object.

3. **THRESH_BINARY** or **THRESH_BINARY_INV**: This parameter can have those two possibilities, to be inverted or not. *INV* will be used when the background of the image is a lighter in color than the object (e.g. white background, black foreground). And the other configuration is used in the opposite case, when the the background of the image is a darker in color than the object (e.g. black background, white foreground).

**Results and discussion.** Experiments of implementations of two discussed approaches are listed here.

### 6.1. Experiments on alpha-matting

For experiments on alpha-matting let's also take the image of ping pong player from figure 12. Applying the alpha

Figure 12. Ping Pong player, Original image

matting algorithm implemented, it is obtained the results of the experiment on figures 13 and 14.

As you can see alpha-matting was applied well in this case, the "silhouette" of the player matches quite accurately the alpha-matting result, getting good images results of background and foreground.

Figure 13. Ping Pong player, alpha-matting

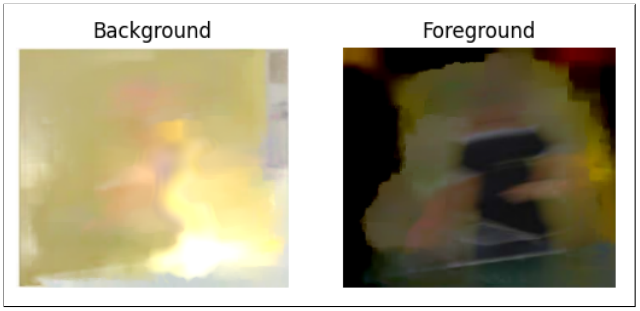


Figure 14. Ping Pong player, background and foreground

## 6.2. Experiments on contour definition

On figures 15-17 you can see results of the gradient-outline algorithm.

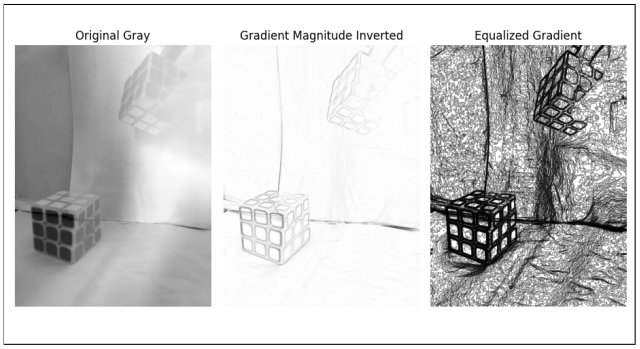


Figure 15. Rubik Cube, gradients

Analysing the results obtained, it can be stated that not for all of the cases equalization is needed. The picture becomes better after equalization in figures 16 and 17. But for figure 15 the image becomes too noisy and it is hard to say if the contours became more visible or not. Basing on the experiments with bad results it is important to say that blurred object should be still visible on the image and it should not blend into the background.

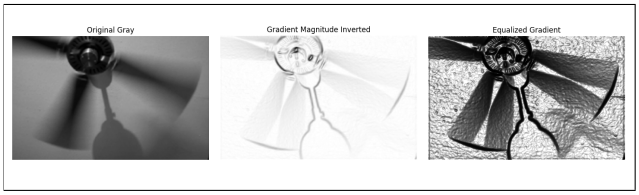


Figure 16. Fan, gradients

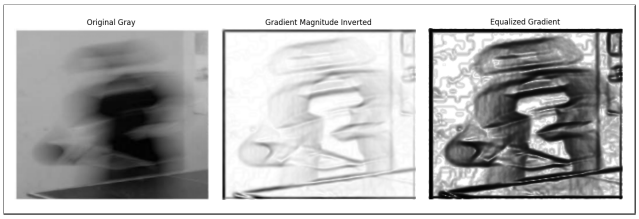


Figure 17. Ping Pong player, gradient

## 6.3. Other experiments

During the process of selecting an approach to address the given task, it is also explored the feasibility of utilizing a thresholding for obtaining the contours of objects. Results of thresholding are illustrated on figures 18-20. It can be observed that in all these experiments, the obtained contours were discontinuous and did not always align with the actual outlines of the objects. Consequently, the decision was made to abandon the use of thresholding for contour delineation and shift towards employing gradient methods.

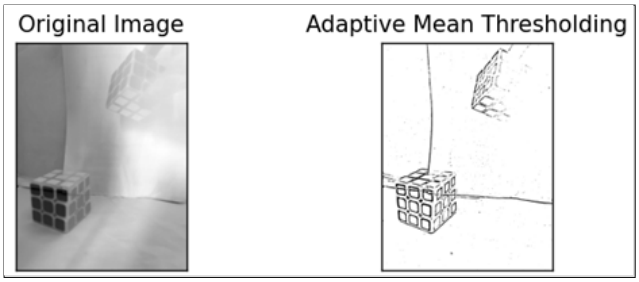


Figure 18. Rubik Cube, Result of thresholding experiment

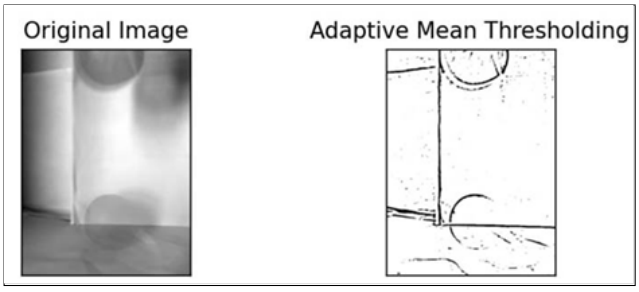


Figure 19. Ball, Result of thresholding experiment

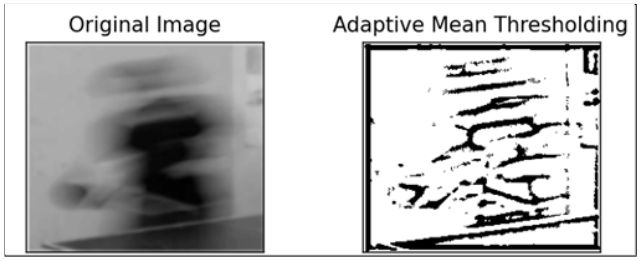

Figure 20. Ping Pong player, Result of thresholding experiment

## 7. Conclusions and future work

In the exploration of object detection within motion-blurred images, two complementary algorithms were composed. The first algorithm, centered around alpha estimation, using the Closed Form method to directly compute alpha values from natural images. Notably, this approach eliminated the need for user-provided scribbles, enhancing accuracy and usability. The second algorithm focused on extracting well-defined object contours, utilizing techniques such as the Sobel operator and equalizing histograms, to obtain a precise contour extraction. Importantly, all experiments were conducted exclusively with motion-blurred images, capturing the graceful trails left by moving objects during extended exposures. The deliberate exclusion of blurred images resulting from camera shake or lack of focus ensures the relevance of the findings to real-world photographic scenarios.

For future researchers that wish to continue this investigation, there is a lot left to do, since this investigation did not cover all possible states of a motion-blurred image. With that being said, open problems and new horizons await, including dynamic scenes with objects in motion, varying speeds, the fluidity of non-rigid objects, and the complexities of low-light conditions in motion-blurred images.

## A. Supplementary Material

The source code for this project can be found at https://github.com/stukmachev/IACV-Project

## References

[1] A. Levin, D. Lischinski, Y. Weiss. A closed form solution to natural image matting, 2007. IEEE transactions on pattern analysis and machine intelligence, 30(2):228–242, 2007. 2, 4

[2] M. Forte and F. Pitié. F, b, alpha matting, 2020. arXiv preprint arXiv:2003.07711. 2

[3] T. Germer, T. Uelwer, S. Conrad, and S. Harmeling. Pymatting: A python library for alpha matting. *Journal of Open Source Software*, 5(54):2481, 2020. 2, 3

[4] OpenCV. OpenCV Documentation, 2024. https://docs.opencv.org/4.x/. 3

[5] V. Caglioti, A. Giusti. On the Apparent Transparency of a Motion Blurred Object, 2010. Int. Journal of Computer Vision n. 86: pp. 243–255. 2, 6