Feature Extraction and Corner Detection

Harris Operator and λ-

SIFT and Image Matching

Team 07

Faculty of Engineering - Cairo University

Dr. Ahmed Badawy

Eng. Laila Abbas

Collaborators

Name	Section	B.N
Ahmed Emad	1	3
Amir Hesham Khalil	1	10
Omar Mohamed Ahmed	1	46
Hazem Zakaria	1	16
Omar Tarek	1	42

Introduction

In the realm of computer vision and image processing, a multitude of techniques and algorithms have been developed to extract meaningful information from images, enabling applications ranging from object recognition to image stitching. This introduction provides a brief overview of key concepts including the Harris corner detector and Lambda minus operators for corner detection, the Scale-Invariant Feature Transform (SIFT) algorithm for key point detection and descriptor extraction, and image matching techniques utilizing Sum of Squared Differences (SSD) and Normalized Cross-Correlation (NCC).

Harris and Lambda Minus Operators: The Harris corner detector, introduced by Chris Harris and Mike Stephens in 1988, remains a fundamental technique for corner detection in images. It operates by analyzing variations in intensity within a local neighborhood, identifying regions where small shifts in the window result in significant changes in intensity. These regions are indicative of corners or interest points. The algorithm computes a corner response function using derivatives of the image intensity, typically through convolution with Gaussian derivatives. The eigenvalues of this response matrix are then used to determine corner-like structures. Lambda minus operators, an extension of the Harris detector, enhance the detection of corners by considering local image structure along multiple scales, enabling more robust corner localization across different image resolutions.

SIFT Algorithm: The Scale-Invariant Feature Transform (SIFT), proposed by David Lowe in 1999, addresses the challenge of identifying distinctive key points invariant to scale, rotation, and illumination changes. The algorithm proceeds through several stages: scale-space extrema

detection, key point localization, orientation assignment, and descriptor generation. SIFT detects local extrema in a Gaussian scale-space pyramid to identify potential key point locations. Subsequently, key points are refined based on local contrast and stability criteria. Orientation assignment ensures rotation invariance by assigning a dominant orientation to each key point. Finally, descriptors are computed based on gradient magnitudes and orientations in the key point's local neighborhood, resulting in highly distinctive feature vectors robust to various transformations.

Image Matching: Image matching is a fundamental task in computer vision, essential for tasks such as object recognition, image registration, and panoramic stitching. Two common techniques for image matching are Sum of Squared Differences (SSD) and Normalized Cross-Correlation (NCC). SSD measures the discrepancy between two images by computing the sum of squared intensity differences between corresponding pixels. It provides a simple and efficient metric for comparing image similarity but is sensitive to changes in illumination and contrast. NCC, on the other hand, measures the similarity between two images by computing the normalized dot product of their intensity values. By normalizing for variations in illumination and contrast, NCC offers improved robustness to changes in lighting conditions. Both SSD and NCC are widely employed in various applications, providing essential tools for image alignment and similarity measurement.

Experiments and Results

1. Corner Detection

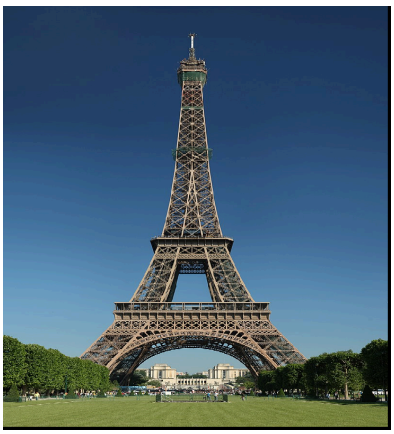
Corner detection is a fundamental task in computer vision and image processing, with applications ranging from object recognition to motion tracking. Harris corner detector and Lambda Minus corner detector are two popular methods used for corner detection.
We are considering two computations for corner detection: Harris and Lambda Minus.

Harris Detector:

The Harris corner detector calculates the Harris corner response function at each pixel, which is a scalar value indicating the likelihood of that pixel being part of a corner. This response function is computed using the eigenvalues of the structure tensor. Pixels with high corner response values are considered as corners. Harris corner detector provides robustness to noise and variation in illumination.

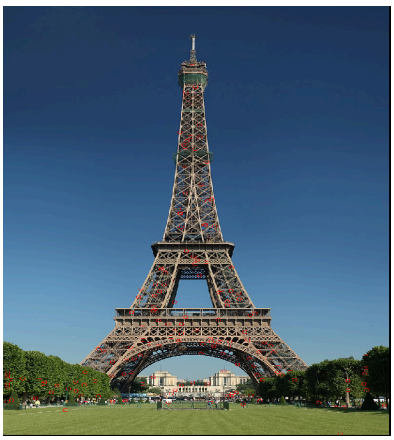
The Lambda Minus Corner Detector:

The Lambda Minus corner detector computes the smallest eigenvalue of the structure tensor at each pixel and uses it as a measure of cornerness. Pixels with a large smallest eigenvalue are considered as corners. Lambda Minus corner detector is computationally efficient compared to the Harris corner detector since it only involves the computation of one eigenvalue. Below are images showing different application results of applying both operators:

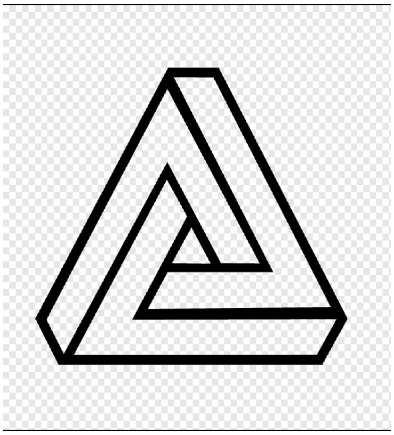


Original image

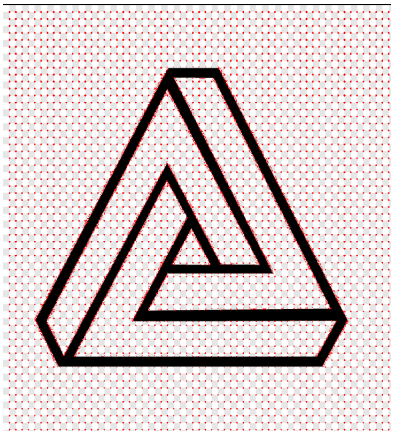
Harris (k=0.4)
Threshold: 0.83
Time: 0.101 s



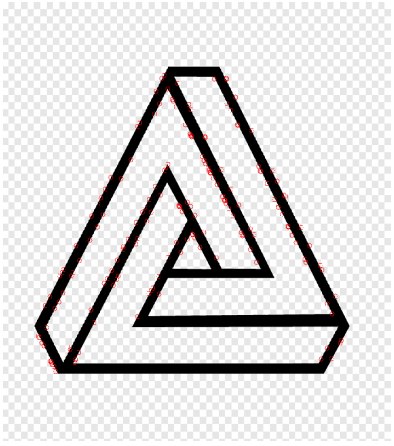
Lambda minus
Threshold: 0.83
Time: 0.676 s



Original image



Harris (k=0.4)
Threshold: 0.58
Time: 0.091 s



Lambda minus
Threshold: 0.58
Time: 0.138 s

2. SIFT Algorithm

The SIFT algorithm typically requires several sequential steps to create the desired functionality which are the key points along with their descriptors. The main essential steps include:

1- Keypoint Detection:

- The implementation follows the standard SIFT approach of detecting keypoints using the Difference of Gaussians (DoG) algorithm.

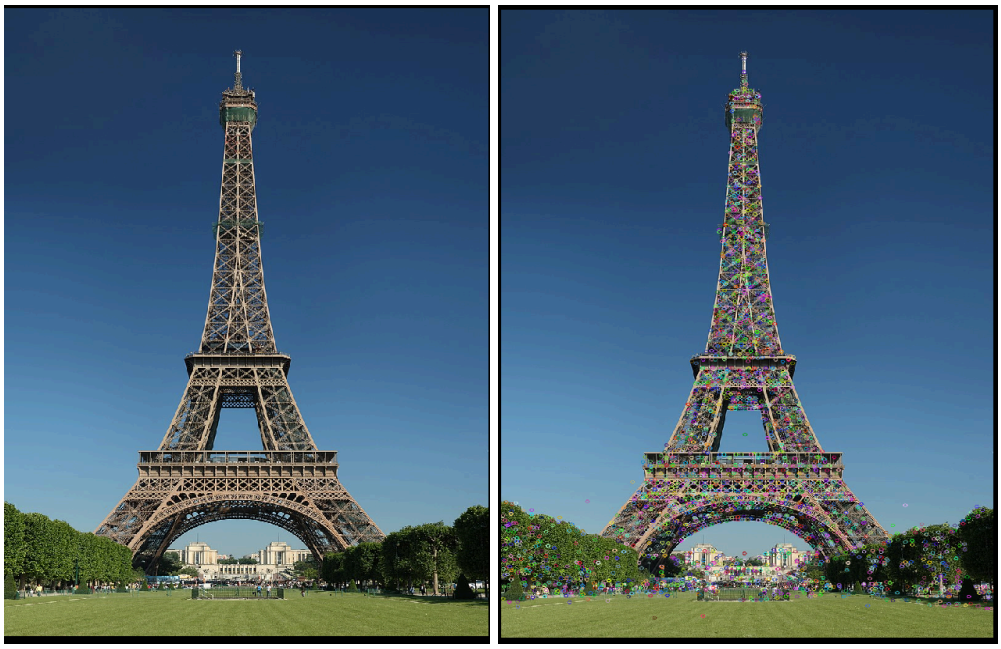
- Key steps include generating Gaussian pyramids, computing DoG images, and identifying local extrema in scale-space.

2- Orientation Assignment:

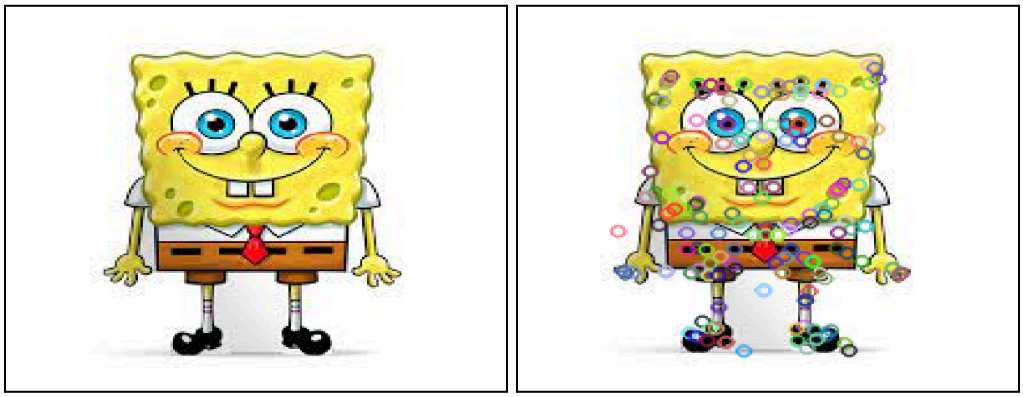
- Keypoints are assigned orientations based on local image gradients.
- Histograms of gradient orientations are computed in a window around each keypoint.
- Dominant orientations are identified as peaks in the histogram.

3- Descriptor Generation:

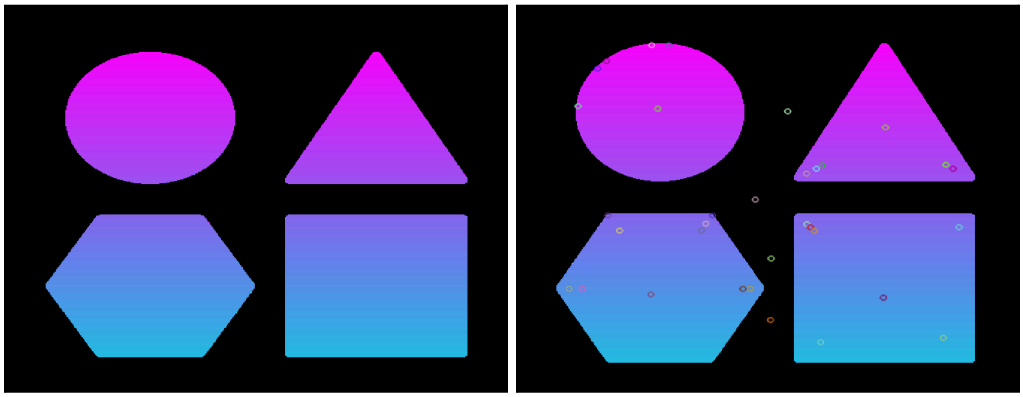
- Descriptors are computed for each keypoint to represent its local image patch.
- A window around each keypoint is divided into subregions, and histograms of gradient orientations are computed for each subregion.
- Histograms are weighted by gradient magnitudes and smoothed.
- Final descriptors are formed by concatenating these histograms.



Before and after extracting features using SIFT algorithm on an image with various details.



Before and after extracting features using SIFT algorithm on an image with moderate details.



Before and after extracting features using SIFT algorithm on an image with low details.

3. Image Matching

In the image matching process utilizing the Scale-Invariant Feature Transform (SIFT) descriptors and employing Sum of Squared Differences (SSD) and Normalized Cross-Correlation (NCC), each key point's descriptor from one image is compared with descriptors from the other image. For SSD, the distance between descriptors is computed using the sum of squared intensity differences:

$$\mathrm{SSD}(d) = \sum_{i=1}^{N} (I_{1,i} - I_{2,i})^2$$

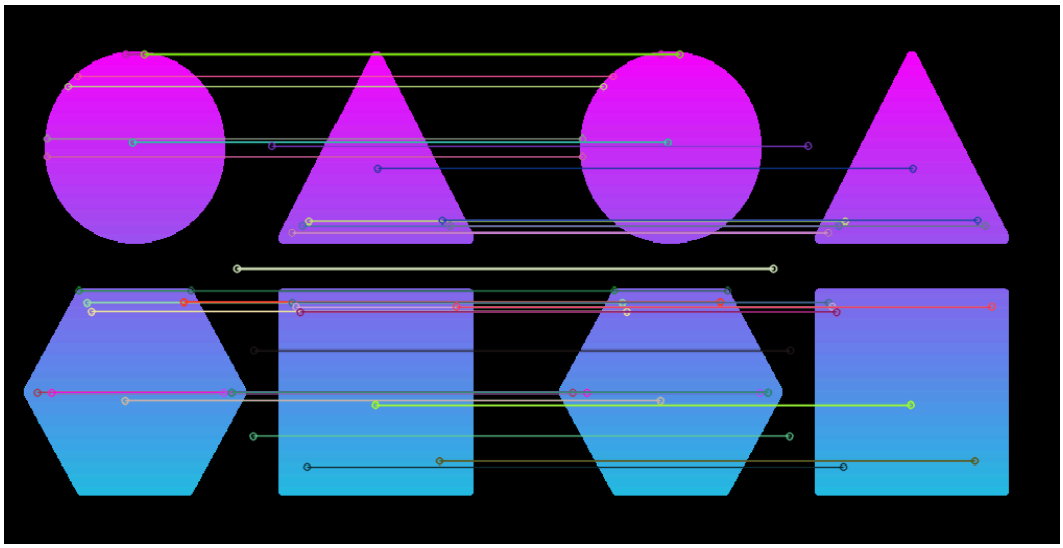
where $I_{1,i}$ and $I_{2,i}$ represent the intensity values of corresponding components in the descriptors of the two images, and N is the dimensionality of the descriptor. The key point pairs with the least SSD distances within the user-defined bounds, set through sliders in the user interface, are considered as matches.

Conversely, NCC calculates the similarity between descriptors by computing the normalized dot product:

$$\mathrm{NCC}(d) = \frac{\sum_{i=1}^{N} I_{1,i} \times I_{2,i}}{\sqrt{\sum_{i=1}^{N} I_{1,i}^2} \times \sqrt{\sum_{i=1}^{N} I_{2,i}^2}}$$

where $I_{1,i}$ and $I_{2,i}$ denote the intensity values of corresponding components in the descriptors of the two images. The key point pairs with the highest NCC scores within the user-defined bounds are identified as matches. By allowing users to adjust lower and upper bounds through sliders, the matching process becomes flexible, accommodating varying degrees of similarity between descriptors, ensuring robust and accurate image alignment.

Matching sponge-bob image with 0 LB and 150 UB for distance between key points' descriptors using SSD.



Matching identical images using NCC.

In the image matching process, Normalized Cross-Correlation (NCC) matches all key points from image 1 to image 2 with a correlation score greater than or equal to **0.75**. This threshold ensures that only highly correlated key point pairs are considered as matches, enhancing the robustness of the matching process. Conversely, Sum of Squared Differences (SSD) takes into account that the distance between descriptors lies strictly between lower and upper bounds set by the user. This approach ensures that only key point pairs with SSD distances within the specified bounds are considered as matches, providing users with control over the matching criteria and enabling fine-tuning of the matching results.

Software Design and Architecture

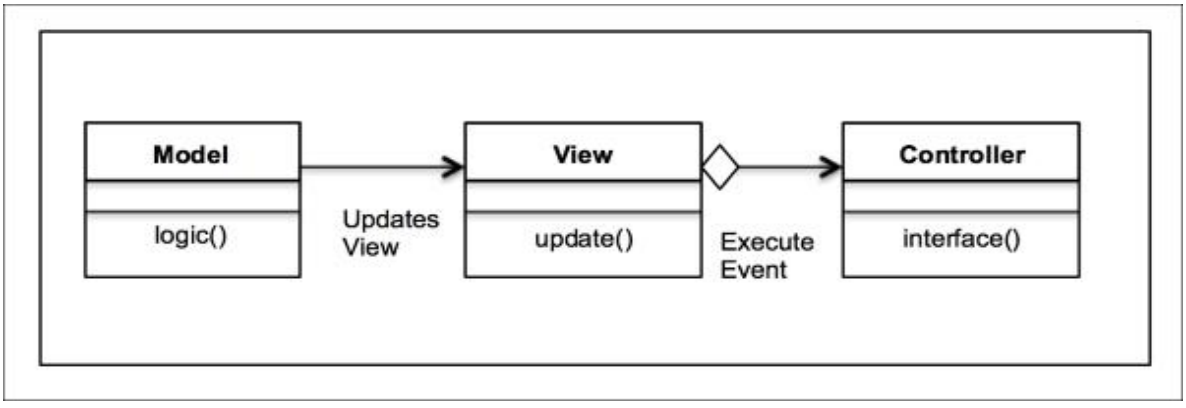
The application is built upon a robust software design architecture that leverages the Python programming language, PyQt framework, Qt Designer for UI design, and follows the Model-View-Controller (MVC) pattern for efficient organization and separation of concerns.

1. Model-View-Controller (MVC) Pattern

● Model: The Model component represents the data and business logic of the application. In our context, it encapsulates image processing algorithms, data structures for image representation, and manipulation functions.
The model in our case is represented in Image processing algorithms, data structures, and functions reside within the Model component, ensuring encapsulation and modularity.

● View: The View component is responsible for presenting the user interface to the user. Qt Designer is utilized to design the graphical user interface (GUI) elements, including buttons, sliders, image displays, and histograms.
The view in our case encompasses the graphical user interface designed using Qt Designer. It includes various widgets for user interaction and image display.

● Controller: The Controller acts as an intermediary between the Model and the View, handling user input and triggering appropriate actions. It binds GUI events to corresponding image processing functions, ensuring seamless interaction between the user and the application.
The controller in our case bridges the gap between the Model and the View, handling user events and orchestrating image processing tasks based on user input.



2. Used Technologies and Tools

● Python: Python serves as the primary programming language for our application, offering simplicity, readability, and extensive libraries for image processing tasks.

● PyQt: PyQt is a set of Python bindings for the Qt application framework. It enables the integration of Qt functionalities within Python applications, providing access to a vast array of GUI components and features for building cross-platform desktop applications.

● Qt Designer: Qt Designer is utilized for designing the graphical user interface of our application. It offers a drag-and-drop interface for creating and arranging UI elements, streamlining the GUI design process. Designed UI files (.ui) created in Qt Designer are seamlessly integrated into the Python codebase using PyQt, allowing for rapid development and modification of the user interface.

3. Goals

● Modularity: The MVC pattern facilitates modular design, enabling independent development and testing of Model, View, and Controller components.

● Separation of Concerns: MVC separates the presentation logic (View) from the application logic (Model), enhancing maintainability and scalability.

● Ease of Development: Python, PyQt, and Qt Designer collectively provide a conducive environment for rapid application development, allowing us, as developers, to focus on functionality rather than low-level details of GUI implementation.