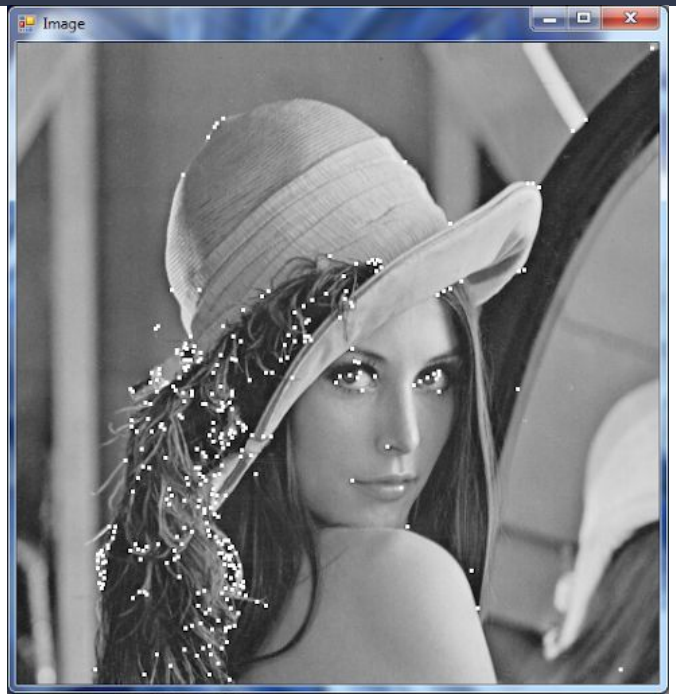
FAST: Features of Accelerated Segment Test

Christian Badoy

Department of Computer Science, Cal Poly Pomona

Introduction

- Computer Vision enables computers as a way to interpret the real world
- One of the important factors of Computer vision is keypoint detection, description, and matching
- These factors combined can be used to give an image high content information such as shape, contrast, or details



Motivation/Challenges

- Computer vision has applications in the industry
    - 3D Image Shaping, Object Detection, Gesture Recognition
- There is ongoing research to increase efficiency, robustness, and repeatability of corner detections
- Not every system can provide too much computational power, and must use its resources with care
- If image size input grows, then running vision algorithms becomes more computationally expensive

Features of Accelerated Segment Test (FAST)

- FAST is an algorithm that utilizes the brightness details of an image to determine if a pixel is a corner or not [1-6]
- Advantages:
  - Locality – A pixel has a well-defined position and high local information content.
  - Robustness to Intensity Variation – Can adjust to different levels of brightness based on its design
  - Performance – Designed for high-speed corner detection. Useful for rapid feature extraction applications
- Disadvantages
  - Sensitive to noise – If there is a significant amount of noise in an image, FAST can produce false key points
  - Not inherently invariant to scale, rotation, or high-speed objects in an image.
  - Lacks subpixel accuracy – FAST can detect corners but not precisely locate corners.

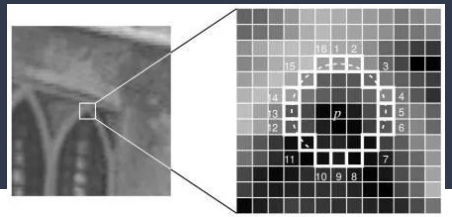
Methodology of FAST

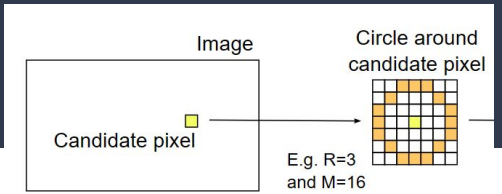
- Overview of FAST Algorithm [1, 6]
  - **Selecting a Pixel of Interest:** Locating all the key points of an image
  - **Defining a Circular Neighborhood:** Creating a circle around a selected pixel for neighbor pixels.
  - **Thresholding and Intensity Comparisons:** Setting a brightness threshold value and determining contiguity test for pixels on the circle
  - **Non-Maximum Suppression:** Eliminating redundant key points that are concentrated in certain areas of an image
  - Steps 1-3 repeated for all pixels

Methodology of FAST

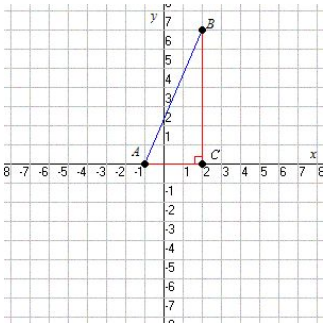
- **Selecting a Pixel of Interest**
  - For a pixel to be considered a keypoint, some user-defined parameters and initializations of FAST are required beforehand
  - Values
    - Circle size of radius *r*
    - Brightness Threshold Value *T*
    - Contiguous Threshold Value *N*
  - Image
    - Converted to Grayscale
  - All pixels on the image must be tested and checked, hence why it can get computationally expensive to checking the images

Methodology of FAST





- **Defining a Circular Neighborhood**
  - In order to give a keypoint pixel high local information content, the nucleus pixel ***P*** will utilize the perimeter of a circle as its neighboring pixels ***P→x***
  - Circles can be implemented either implemented with Bresenham circles or Euclidean distances
  - Euclidean Distance
    - A concept of distance in two-dimensional space using Cartesian coordinates
  - Creating a Circle with Euclidean Distance
    - Limit the traversal with a bounding box, with min, max set as the (pixel coordinate ± radius of the circle)
    - Calculate the Euclidean Distance for all pixels in the bounding box
    - Take the absolute value of (Distance - Radius) and test if the value is less than 1
    - If less than 1, then record coordinate as a neighbor pixel ***P→x***



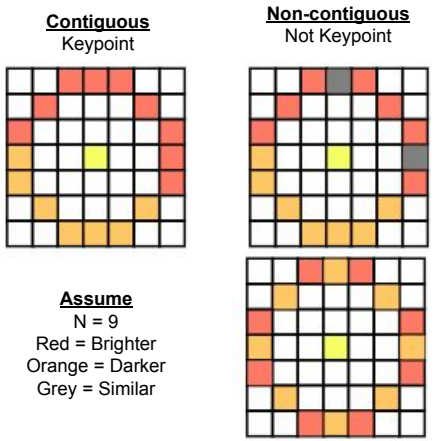
Methodology of FAST

$$S_{p\to x} = \begin{cases} d, & I_{p\to x} \le I_p - t & \text{(darker)} \\ s, & I_p - t < I_{p\to x} < I_p + t & \text{(similar)} \\ b, & I_p + t \le I_{p\to x} & \text{(brighter)} \end{cases}$$

- **Intensity Variation Comparison and Brightness States**
  - A pixel's high local information content originates from the pixel intensity values
  - Image is converted to grayscale to measure a pixel's brightness value
    - Grayscaling yields a numerical value from 0 (black) to 255 (white)
  - A brightness threshold value, defined by the user, is used to control the how much intensity difference is required for a pixel to be a key point
    - A pixel has three pixel states: bright, neutral, darker
    - For a pixel to be considered a brighter or darker state, the difference between the neighbor pixel's and the center's grayscale value must be less than than the threshold
  - The pixel coordinate will store one of the three states

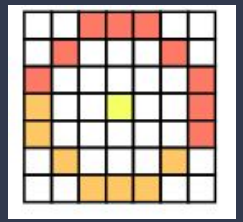
Methodology of FAST

- **Intensity Variation Comparison and Brightness States (cont.)**
  - A part of the FAST algorithm involves a contiguity test
  - The contiguous test requires for a certain number of adjacent neighbor pixels to have the same pixel state of either bright or dark
    - Assume contiguous requirement is N = 9
    - Keypoint
      - 9 or greater adjacent neighbor pixels have same state
    - Non-contiguous/Not a Keypoint
      - 8 Adjacent neighbor pixels of same state
      - A group of pixels of same state interrupted by a different state pixel

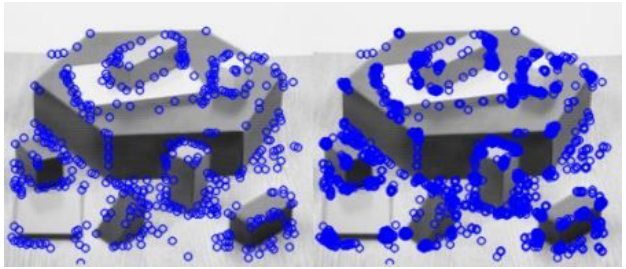


Non-Maximal Suppression

$$V = \max \begin{cases} \sum(\text{pixel values} - p) \text{ if } (\text{value} - p) > t \\ \sum(p - \text{pixel values}) \text{ if } (p - \text{value}) > t \end{cases}$$



- Secondary Test that accompanies FAST
- Eliminates redundant key points within adjacent locations after FAST
- A Methodology
  - For all key point pixel points, compute a score function *V*.
  - *V* = Sum of absolute difference between the pixels in the contiguous arc and the center pixel
  - Consider two adjacent points
  - For two adjacent key point pixels (eg key pixel “B” within key pixel “A’s” circle), compare score values
  - Discard the key pixel with the lower V score.



NMS No NMS

Research Goals

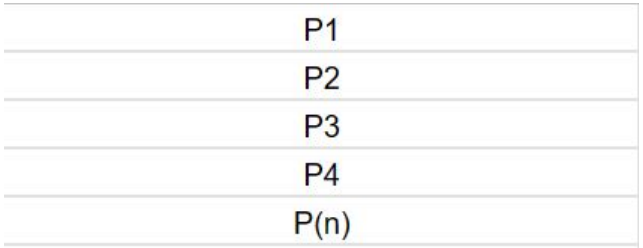
- The goal of this research is to be able to re-implement FAST from scratch while introducing multiprocessing do conduct partial processing for an image
- The purpose of multiprocessing is to allow execution of multiple processes simultaneously on a multi-core processor
- FAST’s task is to compute and record keypoint data of an image
- When implementing multiprocessing into FAST, it is possible to distribute work to each processor by dividing a whole image into smaller, equal chunks
    - Each processor handles finding key points within a chunk of the process in parallel

Multiprocessing

- A system of utilizing more than one processor of a time [7-9]
- Pros
  - Domain Decomposition – Dividing the data into smaller chunks & assign a core/thread for independent execution in parallel
  - Performance – Having each processes work on equally-sized data fine-grains the work distribution. Achieves true parallelism
  - Overcomes Global Interpreter Lock limitations – MP spawns processes, while threading is limited by GIL in executing one Python code at a time
- Cons
  - Complexity – Introduces more complexity into a program, such as managing processes, coordinating communication and synchronization
  - Concurrency issues – Face issues such as race conditions, deadlocks, and shared memory management

Implementing Multiprocessing into FAST

- Methodology
  - In this research, the image is divided into an equivalent chunks (rows) based on number of processors available
  - Each processor will be assigned to compute a chunk of the image to conduct FAST algorithm
    - Shared memory item is the image
    - Shared storage data structure is handled by a Manager
  - All processes will compute FAST in their designated chunk in true parallel
  - Highly improves the processing time significantly

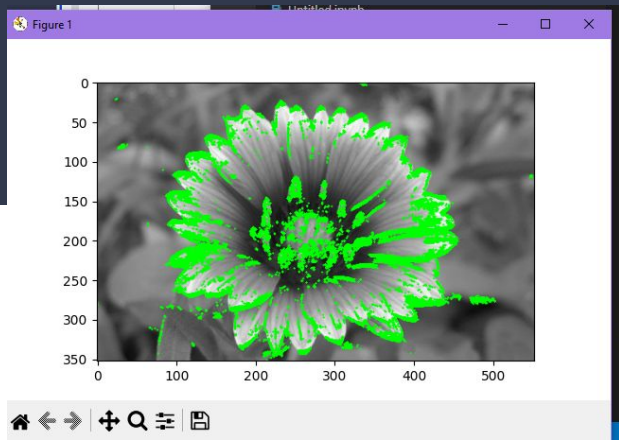


Specs

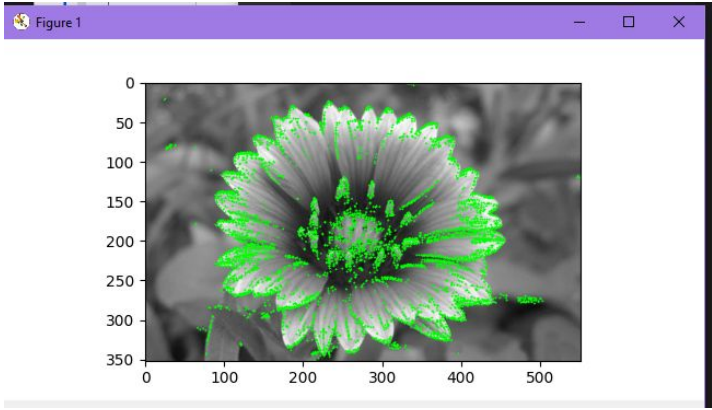
- Intel Core i7 10750H Comet Lake @ 2.60 GHz
  - 6 Cores, 12 Threads
  - RAM: 16 GB
- Base Image:
  - 552 x 352 Image of a Flower

Results (Multiprocess)

Speedup Factor (For no NMS):
31.56022/12.99571 = 2.42851064

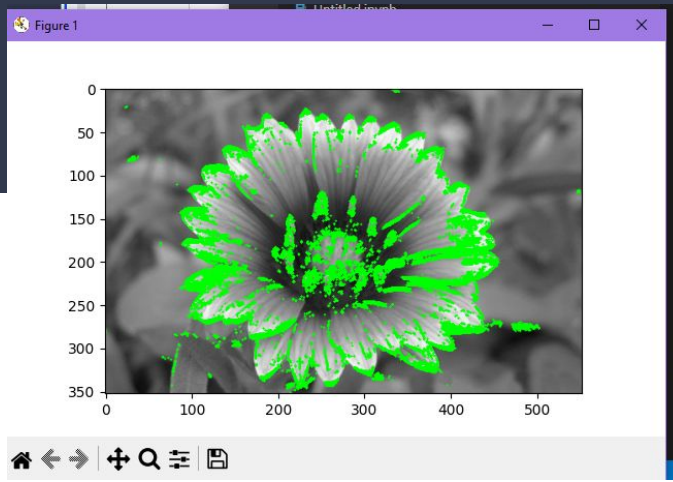



```
Program Start: Multiprocessing.

Checkpoint 1: Processes are all running for FAST calculations.

Checkpoint 2: Processes finished Running for FAST calculations. Merging data from processes...

Contiguous Requirement: 9
Threshold: 16
Radius: 3
Time Elapsed: 12.99571 seconds
Total Keypoints without nonMaxSuppression: 12918
```

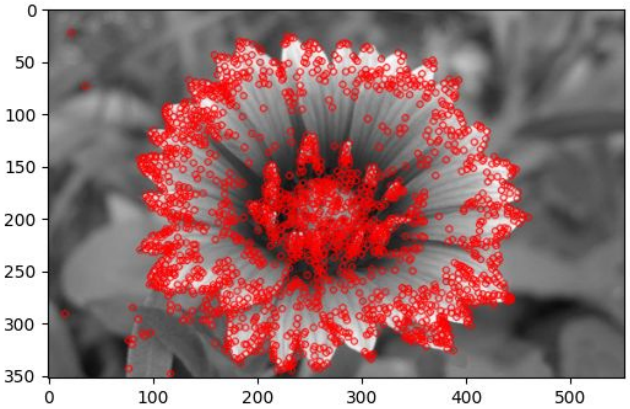



```
Contiguous Requirement: 9
Threshold: 16
Radius: 3
Time Elapsed: 31.56022 seconds (not calculating NMS)
Time Elapsed: 69.51619 seconds (w/ Calculating NMS)
Total Keypoints with nonMaxSuppression: 4743
Total Keypoints without nonMaxSuppression: 12918
```

Results




```
Program Start: Multiprocessing.

Checkpoint 1: Processes are all running for FAST calculations.

Checkpoint 2: Processes finished Running for FAST calculations. Merging data from processes...

Contiguous Requirement: 9
Threshold: 16
Radius: 3
Time Elapsed: 12.99571 seconds
Total Keypoints without nonMaxSuppression: 12918
```




```
Threshold: 10
nonmaxSuppression:True
neighborhood: 2
Total Keypoints with nonmaxSuppression: 1910
Total Keypoints without nonmaxSuppression: 7970
```

Future Direction

- **Processing Thousands of Images**
  - The original goal of Image Processing is to be able to process hundreds or thousands of images as efficient as possible. While my program does not come close to OpenCV's implementation of FAST, more research can be done to achieve what is done.
- **Using Quadrants Instead of Rows**
  - An image can be divided into smaller quadrants instead of rows. Having smaller quadrants can allow a processor to pick up the chunk that is not computed yet instead of allocating by set number of processes.

Works Cited

[1] E. Rosten and T. Drummond, “Machine Learning for High-Speed Corner Detection,” in Computer Vision – ECCV 2006, A. Leonardis, H. Bischof, and A. Pinz, Eds., Berlin, Heidelberg: Springer, 2006, pp. 430–443. doi: 10.1007/11744023_34

[2] P. Fink, “Optimized Implementation of a Feature Detector for Embedded Systems Based on the Accelerated Segment Test,” Apr. 17, 2014. Available: https://elib.dlr.de/113188/. [Accessed: May 03, 2024]

[3] “VPI - Vision Programming Interface: FAST Corners Detector.” Available: https://docs.nvidia.com/vpi/algo_fast_corners_detector.html#:~:text=Features%20from%20Accelerated%20Segment%20Test,computational%20efficiency%2C%20thus%20its%20name. [Accessed: May 03, 2024]

[4] buckmasterinstitute, Introducing the FAST algorithm, (Nov. 15, 2021). Available: https://www.youtube.com/watch?v=Vqtf0iVUqHg. [Accessed: May 03, 2024]

[5] A. P. Andriyandi, W. Darmalaksana, D. S. Maylawati, F. S. Irwansyah, T. Mantoro, and M. A. Ramdhani, “Augmented reality using features accelerated segment test for learning tajweed,” TELKOMNIKA, vol. 18, no. 1, p. 208, Feb. 2020, doi: 10.12928/telkomnika.v18i1.14750. Available: http://telkomnika.uad.ac.id/index.php/TELKOMNIKA/article/view/14750. [Accessed: May 03, 2024]

[6] D. Tyagi, “Introduction to FAST (Features from Accelerated Segment Test),” Medium, Apr. 07, 2020. Available: https://medium.com/@deepanshut041/introduction-to-fast-features-from-accelerated-segment-test-4ed33dde6d65. [Accessed: May 03, 2024]

[7] R. G. Panagiotis, M. Evdoxia, and D. Minas, “Parallelization of the hierarchical search in Python for high performance embedded systems,” in 2016 5th International Conference on Modern Circuits and Systems Technologies (MOCAST), May 2016, pp. 1–4. doi: 10.1109/MOCAST.2016.7495138. Available: https://ieeexplore.ieee.org/document/7495138. [Accessed: May 03, 2024]

[8] “Introduction to Multithreading and Multiprocessing in Python,” KDnuggets. Available: https://www.kdnuggets.com/introduction-to-multithreading-and-multiprocessing-in-python. [Accessed: May 03, 2024]

[9] A. K. Yadav, “Multiprocessing and multiprocessing manager to share an object with Processes in Python,” Medium, Feb. 17, 2023. Available: https://medium.com/@amitkumaryadav27/multiprocessing-and-multiprocessing-manager-to-share-an-object-with-processes-in-python-946b88552b84. [Accessed: May 03, 2024]

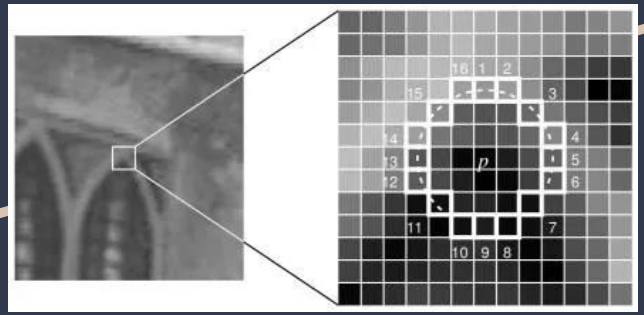
GitHub

https://github.com/clbadoy/CS4610-FAST

What is FAST

- Features from Accelerated Segment Test
  - Used to find key points of an image
    - A pixel has a well-defined position
    - Pixel has high local information content
  - In computer vision, used to obtain feature points and have the ability to track and map objects in an image
- Applications
  - Image Matching
  - Object Recognition
  - Tracking

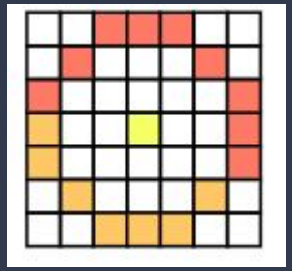
Steps of FAST



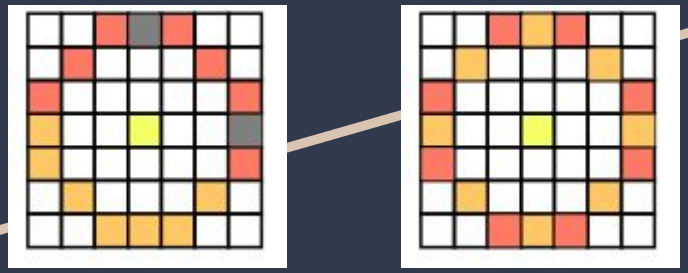
1. Convert the image from RGB to Grayscale
2. Select a candidate pixel *P*
3. Pick a certain threshold value *T*
4. Generate a Bresenham circle of *M* pixels around pixel *P* to test
    a. Let these pixels lying on the circle be considered pixels P→x
5. Calculate Pixel Intensity values and thresholds, denoted as
    a. I_P for candidate pixel's value
    b. $I_{P \rightarrow x}$ for pixels lying on the circle's value
    c. Upper Bound Threshold = $I_P + T$

Steps of FAST

Assume
N = 9
Red = Brighter
Orange = Darker
Grey = Similar



Keypoint Pixel



Not a Keypoint Pixel

$$S_{p\to x} = \begin{cases} d, & I_{p\to x} \le I_p - t & \text{(darker)} \\ s, & I_p - t < I_{p\to x} < I_p + t & \text{(similar)} \\ b, & I_p + t \le I_{p\to x} & \text{(brighter)} \end{cases}$$

- 6. Compute the circle pixel's Intensity State $S_{P\to x}$
  - Pixels lying on the circle P→x under this test can have 3 states
  - Brighter, Darker, or Similar
- 7. Test to see if there exists *N* # contiguous pixels on the circle with same state
  - If x of contiguous pixels >= N, then candidate pixel *P* is considered a keypoint
- 8. Re-iterate steps 1-7 for all pixels on the image

Limitations of FAST

- Noise Sensitivity
  - False/Miss Key Points
- Lack of Scale & Rotation Invariance
  - Nature of FAST and dependency on Bresenham Circles
- Lack of Subpixel Accuracy
  - Difficulty to conduct object stitching and recognition
- Inability to handle high-speed motion objects

Personal Challenges

1. Fully Understanding the Algorithm
    a. Recommended to read articles/papers involving or similar to the Algorithm
    b. Watching Youtube video tutorials on the Algorithm
2. Drawing conclusions from papers with diagrams
    a. Understand the Diagram & its axes/labels
    b. Consider the context & Methodology of the diagram
    c. Analyze the patterns/trends of the diagram
3. Comprehending the Mathematics
    a. Read up on the definitions of each term involved in the Mathematics and jot it down
    b. Try to find an application example of the math involved

Sources

VPI - Vision Programming Interface: FAST Corners Detector (nvidia.com)

Introduction to FAST (Features from Accelerated Segment Test) | by Deepanshu Tyagi | Medium

OpenCV: FAST Algorithm for Corner Detection

Introducing the FAST algorithm - YouTube

AV1FeaturefromAcceleratedSegmentTest.pdf