Harris algorithm

Contents

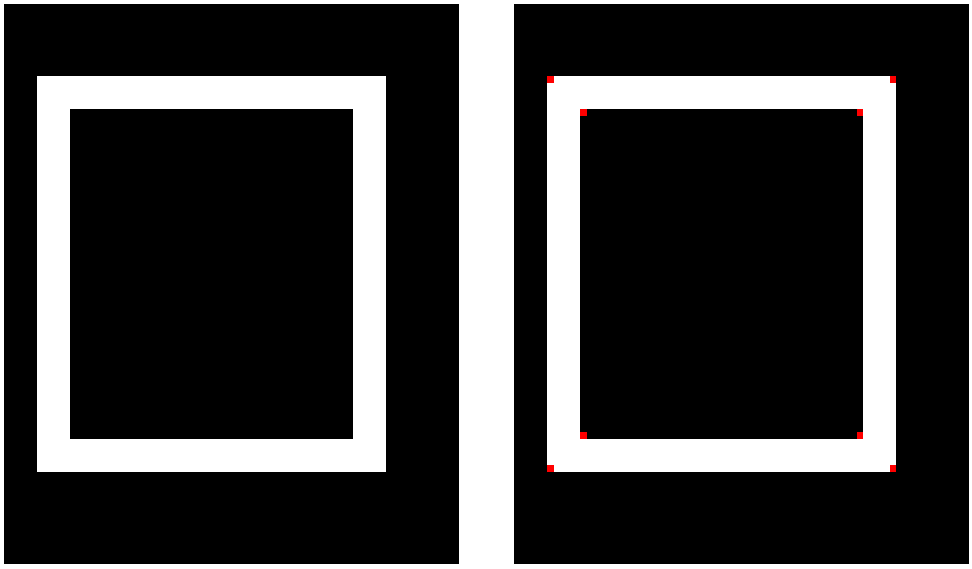

Figure 1. Applying Harris algorithm to input image (on the left) and the output image (on the right).

1. Abstract

Recently the new area of computer usage has been developing rapidly - computer vision and recognition systems. The idea is not new - make a technical system identify necessary attributes, but earlier the concept of real-time processing of a huge data volume was unrealizable. Nowadays we can handle images size of several millions pixels in a split second, enabling security systems and motion control systems orientate finely in a rapidly changing situation. Harris corner detection algorithm [1] is considered to be one of the primary algorithms, allowing one to mark singular points on image,importanceof which is estimated by special function of pixel brightness.

Also there are plenty of variations and add-ons for Harris algorithm such as Shi-Tomasi [3] corner detectng algorithm, Trajkovic-Hedly [4] detector and Moravec operator [2]. The majority of them performs the assessment of brighness function with some window and activation function. Such algorithms require preliminary image processing, for example Gaussian blur. No one algorithm takes into account CUDA features, but all of them could be easily parallelized on GPUs because each pixel is handled separately,however, it needs values from adjacent pixels.

The most simple case, image size of M*N requires performing an addition of 9 pixels for each of M*N points, that shows a large data overlap and suggests a texture memory usage for caching.

2. Problem definition

Given:

- Source image size of M $\times$ N;
- Threshold of detector function, determining whether the point is a part of singular points set.

It is necessary to perform corner detection and save output image with marked points to file. For image edges, missing pixels should be copied from nearest pixels.

3. Proposed method

The following method is proposed to implement Harris algorithm interpolation:

1. Copy data to device memory;
   $A(x) = \sum_{x,y} \omega(x,y) \begin{bmatrix} I_x^2(x) & I_x I_y(x) \\ I_x I_y(x) & I_y^2(x) \end{bmatrix}$
   $\omega(x,y) = g(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$
   $R = det(A) - \alpha trace^2(A) = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$
2. Compare values in all pixels of image with threshold and determine singular points:
   $R(x_c) > t_{threshold}$
3. Select only local maxima from set of singular points and display them on source image, save it to file.

4. Implementation requirements

4.1. Input data

- algorithmś input data shall be grayscale image and threshold value;
- image shall be read from file;
- the result should be stored to a different file to check the correctness of algorithm work;
- input data parameters are to be set when the program starts;
- input data parameters include:
  - path to a source image file;
  - threshold value.

4.2. Output data

- execution time using GPU;
- execution time without GPU;
- report about coincidence or mismatch of results using and without GPU;
- result – output image in case of reading source picture.

4.3. Implementation

It is necessary to use texture memoryб it enables to reduce reading time losses (because of the caching), implement hardware clamping of the image when threating the image boundaries (allows to decrease branching inside warps.

5. The expected result

1. Getting familiar with methods of image recognition.
2. Improve CUDA programming skills, get experience using textures.

References

[1] Harris affine region detector – http://en.wikipedia.org/wiki/Harris_affine_region_detector.

[2] Moravec corner detection – http://kiwi.cs.dal.ca/d̃parks/CornerDetection/moravec.htm.

[3] Wei Shi, Thomas W. Sun, and Richard D. Wesel. Optimal binary distributed detection.

[4] Trajkovic corner detection – http://kiwi.cs.dal.ca/d̃parks/CornerDetection/trajkovic.htm.