School of Engineering
Brown University
Box D
184 Hope Street
Providence, RI 02912

Object Detection Based on GPU Programming

Jiacheng Guo (Jiacheng_guo@brown.edu)
Tianlun Liu (Tianlun_liu@brown.edu)
Tianxiong Wang (Tianxiong_wang@brown.edu)

Proposal submission date: November 20, 2016

Proposed completion date: December 18, 2016

Contents

Glossary of Terms

CUDA: CUDA is a parallel computing platform and application programming interface (API) model created by NVIDIA. It allows software developers and software engineers to use a CUDA-enabled graphics processing unit (GPU) for general purpose processing – an approach termed GPGPU (General-Purpose computing on Graphics Processing Units). The CUDA platform is a software layer that gives direct access to the GPU's virtual instruction set and parallel computational elements, for the execution of compute kernels.

SIFT: Scale-invariant feature transform (or SIFT) is an algorithm in computer vision to detect and describe local features in images. Applications include object recognition, robotic mapping and navigation, image stitching, 3D modeling, gesture recognition, video tracking, individual identification of wildlife and match moving.

SURF: Speeded Up Robust Features (SURF) is a local feature detector and descriptor. It can be used for tasks such as object recognition, image registration, classification, or 3D reconstruction. It is partly inspired by the scale-invariant feature transform (SIFT) descriptor. The standard version of SURF is several times faster than SIFT and claimed by its authors to be more robust against different image transformations than SIFT.

ORB: Oriented FAST and rotated BRIEF (ORB) is a fast-robust local feature detector, first presented by Ethan Rublee et al. in 2011, that can be used in computer vision tasks like object recognition or 3D reconstruction. It is based on the FAST key point detector and the visual descriptor BRIEF (Binary Robust Independent Elementary Features). Its aim is to provide a fast and efficient alternative to SIFT.

FAST: Features from accelerated segment test (FAST) is a corner detection method, which could be used to extract feature points and later used to track and map objects in many computer vision tasks. FAST corner detector was originally developed by Edward Rosten and Tom Drummond, and published in 2006. The most promising advantage of the FAST corner detector is its computational efficiency.

BRIEF: Binary Robust Independent Elementary Features, a faster method feature descriptor calculation and matching. It also provides high recognition rate unless there is large in-plane rotation. One important point is that BRIEF is a feature descriptor, it doesn’t provide any method to find the features.

BRISK: Binary Robust Invariant Scalable Key points, which tackles the classic Computer Vision problem of detecting, describing and matching image key points for cases without sufficient a priori knowledge on the scene and camera poses. BRISK relies on an easily configurable circular sampling pattern from which it computes brightness comparisons to form a binary descriptor string. The unique properties of BRISK can be useful for a wide spectrum of applications, for tasks with hard real-time constraints or limited computation power.

FREAK (Fast Retina Keypoint): as a fast, compact, and robust key point descriptor. A cascade of binary strings is computed by efficiently comparing pairs of image intensities over a retinal sampling pattern. Interestingly, selecting pairs to reduce the dimensionality of the descriptor yields a highly-structured pattern that mimics the saccadic search of the human eyes.

Brief Statement of Work

The overall goal of our project is object detection on graphics processing units, as a part of it, a system for object detection using NVIDIA CUDA was designed and implemented, allowing for graph object detection and image processing. Its contribution is mainly to study the options of NVIDIA CUDA technology and current graphics processing units for object detection acceleration. The project includes features extraction, classifier training and detection realization based on GPU programming.

Significance of the Opportunity

This project has significant impact on computer vision and image processing, and these two related fields have great and profitable outlook.

Computer vision is a simulation of biological vision for computers and related equipment. Its main task is through the collection of pictures or video processing to obtain the corresponding scene information, just as humans and many other kinds of creatures do every day. Computer vision is a knowledge of how the camera and computer are used to obtain the data and information of the subject that we need. Vividly speaking, is to install the computer eyes (camera) and the brain (algorithm), so that the computer can perceive the environment. It is not difficult to imagine how broad the prospect of a machine with vision can be. Computer vision is not only an engineering field, but also a challenging and important research field in the field of science. Computer vision is a comprehensive discipline, it has attracted researchers from various disciplines to participate in its research. These include computer science and engineering, signal processing, physics, applied mathematics and statistics, neurophysiology, and cognitive science.

There are many subjects of research objectives and computer vision like or related to this. These disciplines include image processing, pattern recognition or image recognition, scene analysis, image understanding and so on. Computer vision, including image processing and pattern recognition, in addition, it also includes spatial shape description, geometric modeling and understanding of the process. The realization of image understanding is the overall goal of computer vision.

Opportunities in business field

1) Aerospace and aviation technology

Aerospace and aviation technology applications digital image processing technology in aerospace and aviation technology applications, in addition to JPL on the moon, Mars photo processing, the other application is in aircraft remote sensing and satellite remote sensing technology. Many countries send many reconnaissance aircraft every day to carry out a large number of aerial photography on areas of interest to the planet. The analysis of the resulting photographs, which used to employ thousands of people, and now use advanced computer equipped with an image processing system to interpret the analysis, not only saves manpower, but also speed up the speed, you can also extract from the photo manual Cannot find a lot of useful information.

2) Biomedical engineering

Digital image processing in biomedical engineering applications is very extensive, and very effective. In addition to the above-mentioned CT technology, there is a category of medical microscopic image processing analysis, such as red blood cells, white blood cell classification, chromosome analysis, cancer cell recognition. In addition, X-ray lung image enhancement, ultrasound image processing, electrocardiogram analysis, stereotactic radiotherapy and other medical diagnosis are widely used image processing technology.

3) Communication and Network engineering

The main direction of current communication is voice, text, image, and data combined with multimedia communication. The telephone, the television and the computer are transmitted on the digital communication network in a triple-play manner. One of the most complex and difficult to image communication, because the image data is very large, such as the transmission rate of color television signals up to 100Mbit / s or more. To transmit such high-speed data in real time, it is necessary to use a coding technique to compress the bit amount of the information.

4) Industrial engineering

Image processing technology is widely used in industrial and engineering applications, such as the quality of parts in automated assembly lines, the classification of parts, the inspection of printed circuit boards, the stress analysis of elastic mechanical photographs, the drag and lift of hydrodynamic pictures Analysis, automatic sorting of postal mail, recognition of the shape and arrangement of objects and objects in some toxic and radioactive environments, industrial vision in advanced design and manufacturing techniques, and more. It is worth mentioning that the development of intelligent robots with visual, auditory, and tactile functions, will bring new incentives to the industrial and agricultural production, has been in industrial production in the painting, welding, assembly be effectively used.

5) Military

In the military image processing and recognition is mainly used for missile precision terminal guidance, a variety of reconnaissance photo interpretation, with image transmission, storage and display of military automated command system, aircraft, tanks, and warship simulation training system; Interpretation analysis, fingerprint identification, face identification, incomplete image restoration, and traffic monitoring, accident analysis.

6) Cultural and artistic aspects

At present, such applications include digital editing of TV pictures, animation production, electronic graphic games, textile handicraft design, fashion design and production, hair design, reproduction and restoration of cultural relics photographs, athlete movement analysis and scoring, video and multimedia systems.

At present, the television production system is widely used in image processing, transformation, synthesis, multimedia systems, still images and dynamic image acquisition, compression, processing, storage and transmission.

7) Scientific visualization

Image processing and graphics in close connection with the formation of scientific research in various fields of new research tools.

8) E-commerce

In the current high voice of e-commerce, image processing technology is also promising, such as identity authentication, product security, watermarking technology. In short, a wide range of image processing technology applications, has been in national security, economic development, daily life as an increasingly important role on the national economy and the people's livelihood cannot be underestimated.

Technical Objectives

The overall goal of this project is to detect objects in a loaded image. To achieve the goal, it can be divided into several technical objectives.

1. Extract features from images

Object detection typically use extracted features and learning algorithms to recognize instances of an object category. Feature extraction is a general term for methods of constructing combinations of the variables to get around these problems while still describing the data with sufficient accuracy.

The most exploited approach to object detection and mainly face detection is the appearance based. In general, these rely on methods from machine learning or statistical analysis to develop a model for the sought object. Such methods can be characterized from a statistical point of view, where a sample from an image classified by a Bayesian classification as an object or a non-object.

2. Classify objects

After extracting features from an image dataset, these features can be used to train a classifier. When training the classifier, we should choose a proper algorithm. To make the classifier efficient, only the most informative Haar-like features are selected from a highly over-complete set of these features by this algorithm, which also creates a classifier.

3. Detect objects based on GPU programming.

The problem of object detection by statistical classifiers (from the point of view of CUDA implementation) can be divided in these steps:

1) Loading and representing the classifier data
2) Image pre-processing
3) Object detection
4) Retrieving results

Technical Challenges

Typically, extracting features can be realized by following methods.

1) Knowledge-based methods. These methods encode knowledge of a certain object and capture relationships between its features. For a face this means capturing relationships between the positions of eyes, mouth, nose, or ears.
2) Feature invariant approaches. These algorithms try to find similarities which don't depend on lighting, viewpoint or pose.
3) Template matching. Patterns to describe an object are stored. Correlations between the input image and the patterns are then computed to detect the object.
4) Appearance-based methods. Models are learned from a set of training images, which capture a variability of the object. Such set of training images contains positive and negative samples.

To guarantee the quality of detection result, the method of extracting features from an image must be optimal. Since the extracted features may vary from different methods. Since the project focuses on object detection in an image, a set of training images would be established and the appearance-based method could be adopted.

Technical Approach

1. Extract features using the appearance-based method

1) Haar-like features

A method originally used by Viola and Jones [1] in their object detection framework. Haar-like features are described in figure 2.1, where the sum of the pixels in the black area is subtracted from the sum of the pixels in the white area. The calculation can be accelerated by using integral images, which store the image as already precalculated sums.

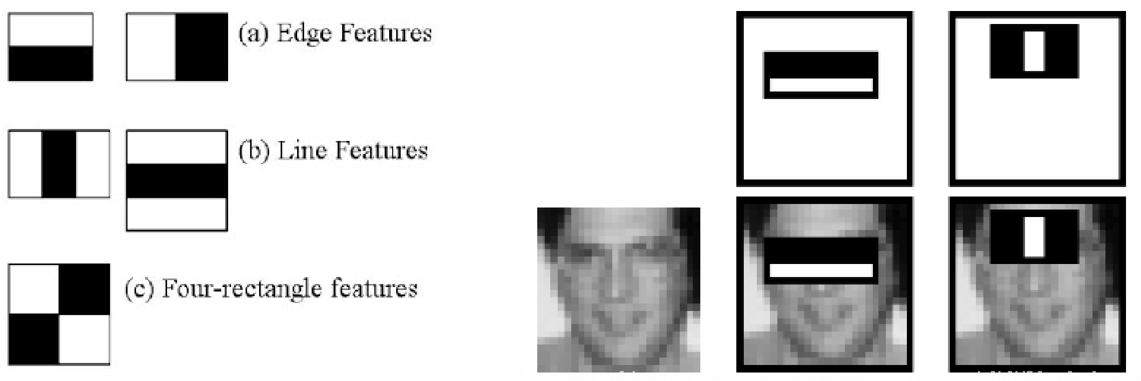
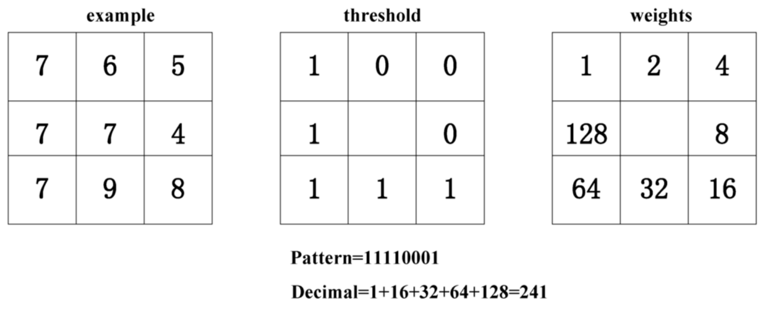


Figure 2.1 Haar-like features [2]

2) Local Binary Pattern

Another method to describe image features, which is used in the implementation, are Local Binary Patterns (LBP). They are based on encoding local intensities of an image with 8-bit codes. In their elementary form, they take a 3×3 area as an input and compare intensity values of all the pixels with the central one as shown in figure 2.2.



example

7	6	5
7	7	4
7	9	8

threshold

1	0	0
1		0
1	1	1

weights

1	2	4
128		8
64	32	16

Pattern=11110001

Decimal=1+16+32+64+128=241

Figure 2.2 Evaluation of LBP

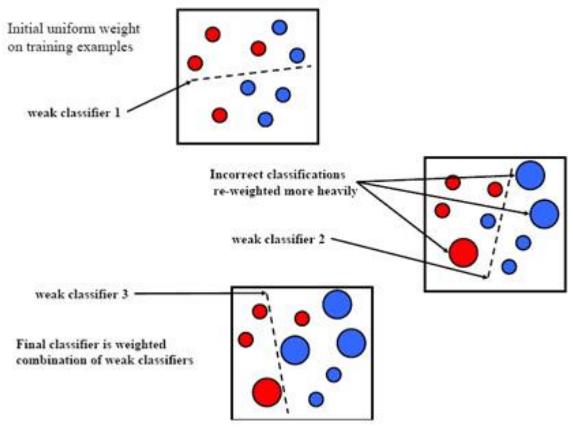
LBPs can be extended to be used not only for single pixels and thus 3×3 areas, but also for larger areas. When comparing larger areas, the sum of the areas is compared instead of single pixel values. For example, a 2×2 LBP would sum up areas of 4 pixels and compare them with the sum in the middle.

LBP features are invariant to lighting changes, because even though the image is lighter or darker, the sign of intensity differences stays the same. On the other hand, they are not invariant to geometrical transformations such as scale or rotation.

2. Classifying an object using the WaldBoost algorithm

1) AdaBoost

One such algorithm to analyze a sample is AdaBoost, an algorithm originally introduced in 1995 by Viola, Jones. The main idea is combining several weak classifiers into a single strong classifier. This means, that every weak classifier decides to which class the given sample belongs to. In object detection, this is usually a yes or no decision, and after processing all the weak classifiers a final decision is made, also a called a strong classifier. The strong classifier is evaluated as sum of decisions made by the weak classifiers. AdaBoost is a general boosting algorithm, which greedily selects weak hypotheses (inaccurate classifiers) and combines them into a strong classifier—a weighted majority of votes. Viola and Jones used AdaBoost for feature selection by keeping the weak hypotheses very simple and each based only on a single Haar-like feature.


Initial uniform weight on training examples
weak classifier 1
Incorrect classifications re-weighted more heavily
weak classifier 2
weak classifier 3
Final classifier is weighted combination of weak classifiers

$$H(x) = sign(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x))$$

Figure 2.3 AdaBoost

2) WaldBoost

Another algorithm and one, which we will discuss in more detail is WaldBoost, because it is used for the implementation. WaldBoost is an algorithm, which combines AdaBoost and Wald's Sequential Probability Ratio Test (SPRT). SPRT is a strategy to determine what class a sample belongs to, based on a series of measurements.

When realizing object detection, the principle of WalBoost algorithm can be explained as follow. The detector scans an image using a scanning window, usually the size of 24×24 or 26×26 pixels, for every possible location and searches for the object. Such scanned region of interest is called a sample and to make a decision, whether it's the given object or not, it is processed by a strong classifier (produced by the WaldBoost algorithm).

The algorithm processes weak classifiers and for each one of them compares them to a constant A. Such weak classifiers are trained based on a given method such as LBP, LRD and so on. After processing, all the weak classifiers, the sample is classified as an object or gets discarded throughout the process.

3. Realize object detection based on GPU programming.

The object detector is to be implemented in C++ with OpenCV and CUDA - the library for writing NVIDIA GPU code with a CUDA C interface. It can be implemented as an easy-to-use class, which takes a dataset as the input and it outputs detections. The following figure is an example of operations.

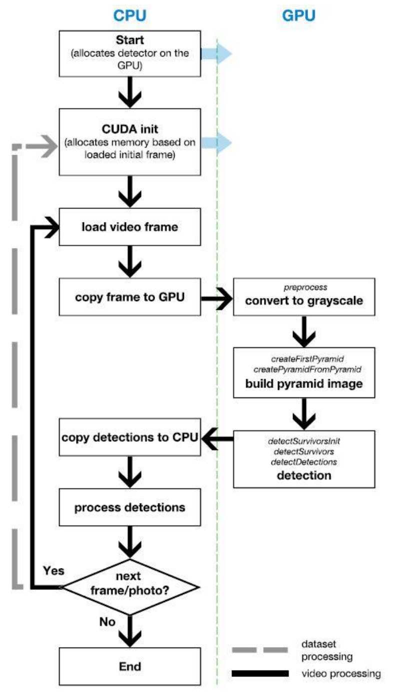


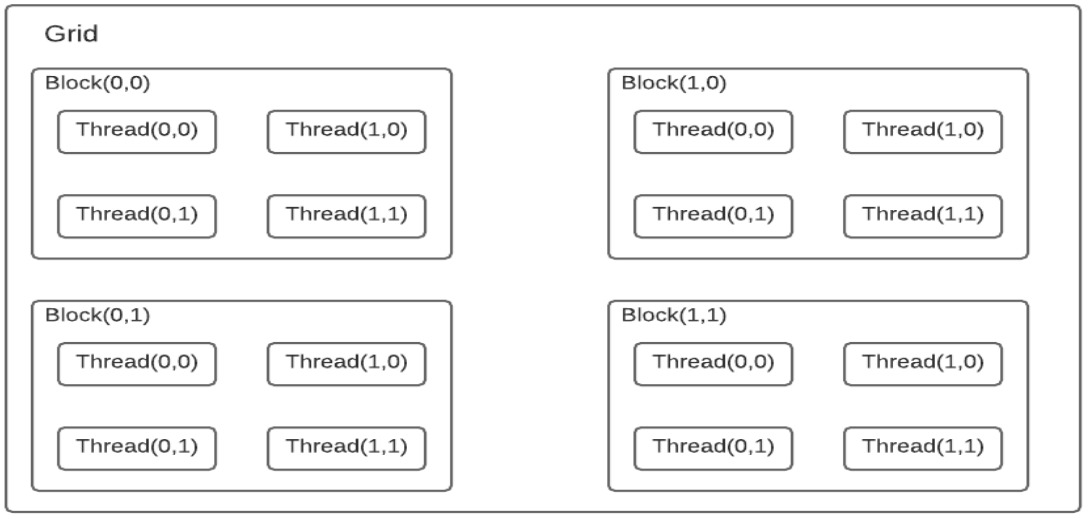
Figure 2.4 GPU Operations

Depending on what detector is trained on, anything from human to signs can be detected.

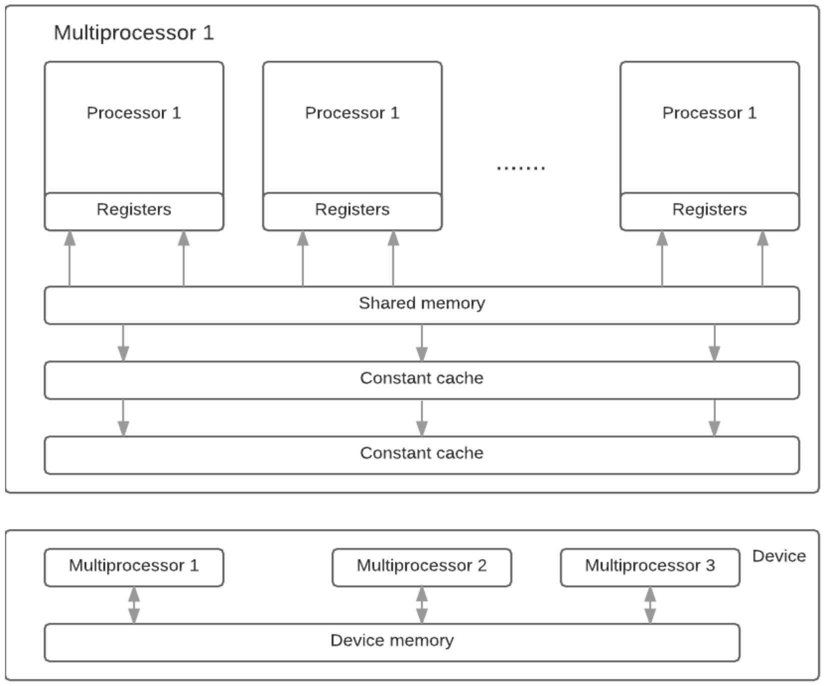
Metrics:

1) Achieved occupancy
2) Memory load/store throughput
3) Atomic and L2 throughput (for atomic requests)
4) L2 hit rate (texture reads)

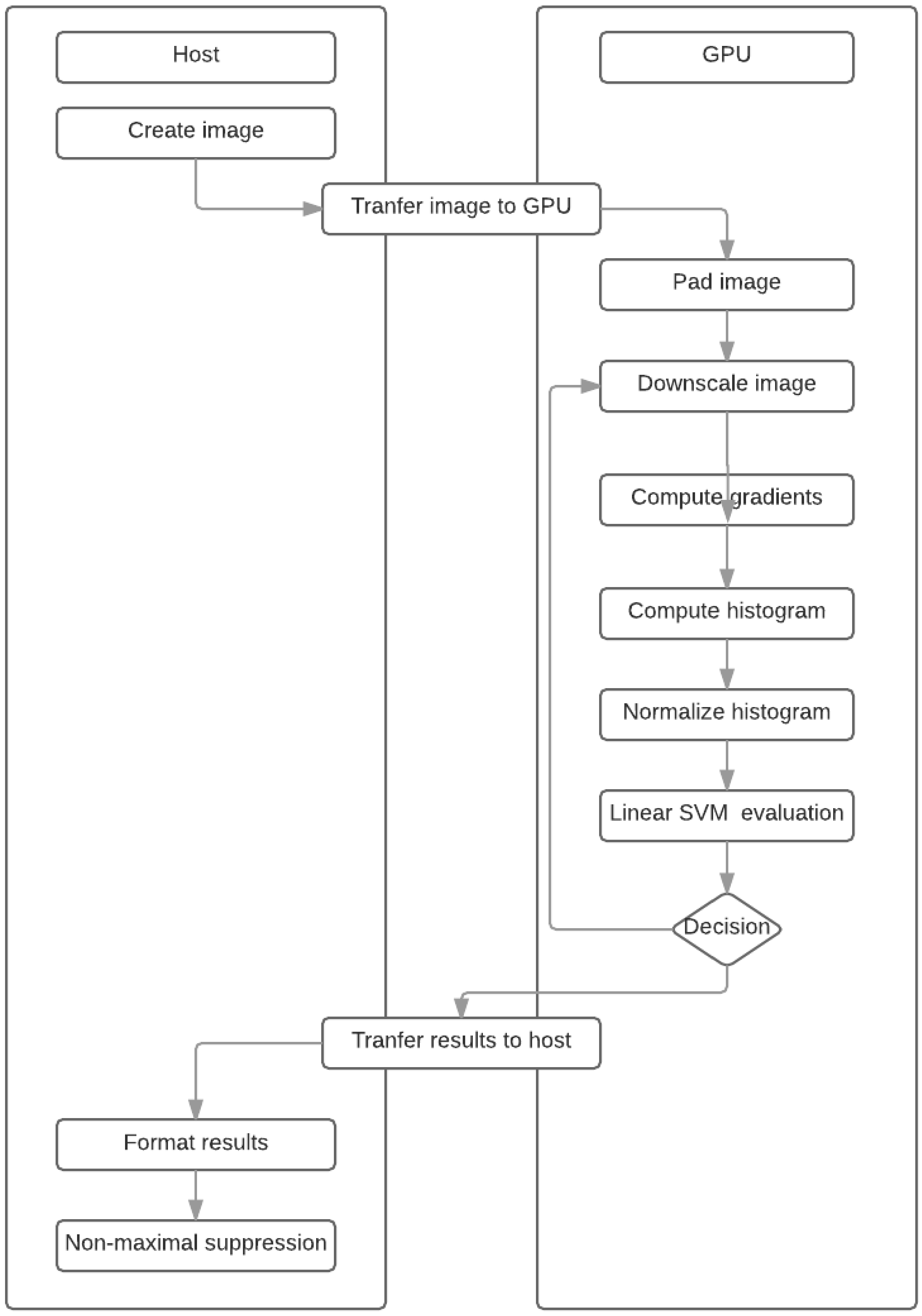
Proposed Software Architecture



CUDA - thread architecture



CUDA - thread architecture



Object detection based on GPU

Project Schedule and Milestones

Week 1: by Nov. 20, Team assignment 2, CUDA and OpenCV installation, background knowledge about image processing, object detection and computer vision.
Week 2: by Nov 27, Image processing including color image to greyscale image conversion, HDR tone mapping, Red-eye removal, Seamless image composition
Week 3: by Dec 4, Feature extraction, Feature Accelerated Segment Test algorithm, HOG descriptor
Week 4: by Dec 11, Object detection descriptor and matching algorithm.
Week 5: by Dec 18, Debug and optimization of project and Final report of project.

Related Project

1. **Detecting Objects on GPU** [3]
   by Pavel Macenauer

With high demand for real-time image processing, computer vision applications and a need for fast calculations in the scientific world, general-purpose computing on GPU (Graphics Processor Units), also known as the GPGPU, has become a popular programming model to accelerate programs traditionally coded on the CPU (Central Processing Unit). The goal of this project was to design a high-performance application or a library for object detection on the GPU. The topic of WaldBoost object detection on the GPU using LBP is well researched, on the other hand there is much less research done on the implementations themselves, and so the focus was to research how different options of the NVidia CUDA Toolkit affect object detection.

2. **GPU Implementations of Object Detection using HOG Features and Deformable Models** [4]
   by Manato Hirabayashi, Shinpei Kato

Vision-based object detection using camera sensors is an essential piece of perception for autonomous vehicles. Various combinations of features and models can be applied to increase the quality and the speed of object detection. A well- known approach uses histograms of oriented gradients (HOG) with deformable models to detect a car in an image. The paper has presented GPU implementations of HOG-based object detection using deformable part models. Our implementations are based on an analysis of performance bottlenecks posed by an introduction of deformable models in HOG-based object detection to accelerate appropriate computational blocks of the program. Detailed experiments using commodity GPUs showed that our implementations speed up HOG-based vehicle detection program tailored to the deformable models by 3x to 5x over the traditional CPU implementations. Given that this performance improvement is obtained from the entire program execution rather than an algorithm within the program, this is a significant contribution for real-world applications.

3. **GPU Accelerated Face Detection** [5]
   by Jussi Mäkelä

Graphics processing units have massive parallel processing capabilities, and there is a growing interest in utilizing them for generic computing. One area of interest is computationally heavy computer vision algorithms, such as face detection and recognition. Face detection is used in a variety of applications, for example the autofocus on cameras, face and emotion recognition, and access control. In this thesis, the face detection algorithm was accelerated with GPU using OpenCL. The goal was to gain performance benefit while keeping the implementations function- ally equivalent. A review of the development and APIs in GPGPU, and specifically OpenCL was made. Notable advances in GPGPU and face detection were discussed. Also, the research regarding performance differences between OpenCL and CUDA was reviewed. An OpenCL version of the Haar-based face detector was designed and implemented. The challenges and different optimization strategies were discussed.

Reference

[1] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features, 2001.

[2] http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html

[3] https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=115030

[4] https://www.computer.org/csdl/proceedings/cpsna/2013/0798/00/06614255.pdf

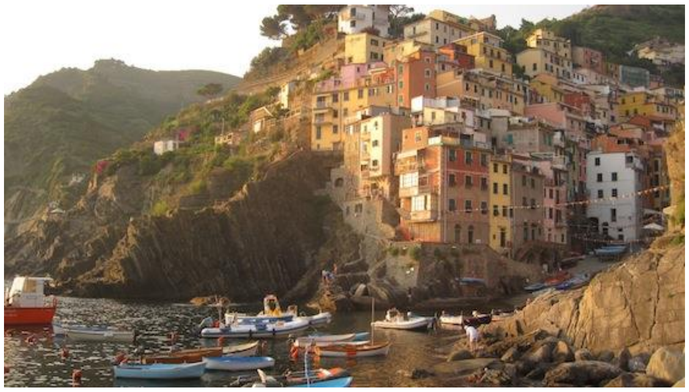
[5] http://jultika.oulu.fi/files/nbnfioulu-201303181103.pdf

Programming Task

Convert color image to greyscale image on Cuda C++

```
GpuTimer timer;
timer.Start();
//call the GPU code
__global__
void rgba_to_greyscale(const uchar4* const rgbaImage,
            unsigned char* const greyImage,
            int numRows, int numCols)
{
    //The output (greyImage) at each pixel should be the result of
    //applying the formula: output = .299f * R + .587f * G + .114f * B;
    //First create a mapping from the 2D block and grid locations
    //to an absolute 2D location in the image, then use that to
    //calculate a 1D offset
    int n=threadIdx.x;
    int m=blockIdx.x;
    uchar4 rgba=rgbaImage[numCols*m+n];
    greyImage[numCols*m+n]=.299f * rgba.x + .587f * rgba.y + .114f * rgba.z;
}
void rgba_to_greyscale(const uchar4 * const h_rgbaImage, uchar4 * const d_rgbaImage,
            unsigned char* const d_greyImage, size_t numRows, size_t numCols)
{
    //currently only one block with one thread is being launched
    const dim3 blockSize(numCols, 1, 1);
    const dim3 gridSize(numRows, 1, 1);
    rgba_to_greyscale<<<gridSize, blockSize>>>(d_rgbaImage, d_greyImage, numRows, numCols);
    cudaDeviceSynchronize(); checkCudaErrors(cudaGetLastError());
}
  timer.Stop();
cudaDeviceSynchronize();
checkCudaErrors(cudaGetLastError());
```

Original Image:



Output Image: