Survey of Techniques in Parallel Image Feature Extraction

Anuroop S
Department of CS&E,
Adhichunchanagiri Inistiture of Technology,
Chikkamagalur, India.

Dr. M.G. Suraj
Department of CS&E,
Adhichunchanagiri Institute of Technology,
Chikkamagaluru, India

Abstract **- With tremendous increase in multimedia processing, sometimes it is necessary to extract some points of interests or key points in image for comparison with other images or for further processing. This entire process is called image feature detection and description. Currently in there are two very popular Image Feature extraction algorithms (IFEA) namely SIFT and SURF. But these algorithms are designed in serial manner and cannot utilize the full power of parallel processing in modern computers. In this paper we present different techniques for parallel image feature detection and description, without affecting the functionality of the IFEA's.**

Key words: SIFT, SURF, IFEA's, feature extraction, parallel processing.

I. INTORDUCTION

Image feature extraction is the method of extracting interesting points or key points in an image as a compact feature vector. Feature detection, extraction and matching are often combined to solve common computer vision problem such as object detection, motion tracing, image matching and object recognition in an image scene. As David Lowe [1] mentions about SIFT key points "The features are invariant to image scaling and rotation, and partially invariant to change in illumination and 3D camera viewpoint. They are well localized in both the spatial and frequency domains, reducing the probability of disruption by occlusion, clutter or noise".

Fig 1: Object matching in an image scene

As shown in the above Fig 1 by using SIFT algorithm we are able to find the object on the left in an image scene in the right regardless of the scale, illumination and orientation.

II. SIFT and SURF Overview

SIFT, SURF [1][3] are 2 most well-liked and sturdy LFAs (Local Feature Algorithm). Each of them carries with it an initialization stage, feature detection stage and have description stage. An outline of their work flow is shown in Fig 2 and it is as follows:

- Initialization stage: This stage will do some initialization work, including loading the image, obtaining the intensity of every image component.
- Feature detection: This stage detects feature or key points in an image. First it builds a scale space pyramid of *m*x*n* which

is guaranteed to be scale invariant. Different methods are used to build the scale space pyramid in SIFT and SURF. In SIFT the scale space is built as shown in Fig 3, using difference of Gaussian and applying the Gaussian filter repeatedly. SURF alters the filter size to effectively construct the scale space. After the pyramid construction, it is compared with the neighboring 26 pixels in 3x3x3 cube. If the pixel or point is a maxima value the point is taken as key point. To ensure the quality of extracted key points the points along the low contrast and edges are discarded. The remaining points are taken as feature points

- Feature Description: In this stage, each identified point will be described by a *n*

dimensional vector (where *n* = 64 for SURF and n = 128 for SIFT). In order to ensure rotation invariant, orientation is calculated based on the pixels around it. And finally a descriptor window is constructed based on orientation information and the value feature is normalized to keep the feature point illumination invariant. But this process of feature extraction goes in serial manner and cannot utilize the parallel processing power of modern computers. However, there are several approaches to parallelize the feature extraction process and they are discussed in next section.

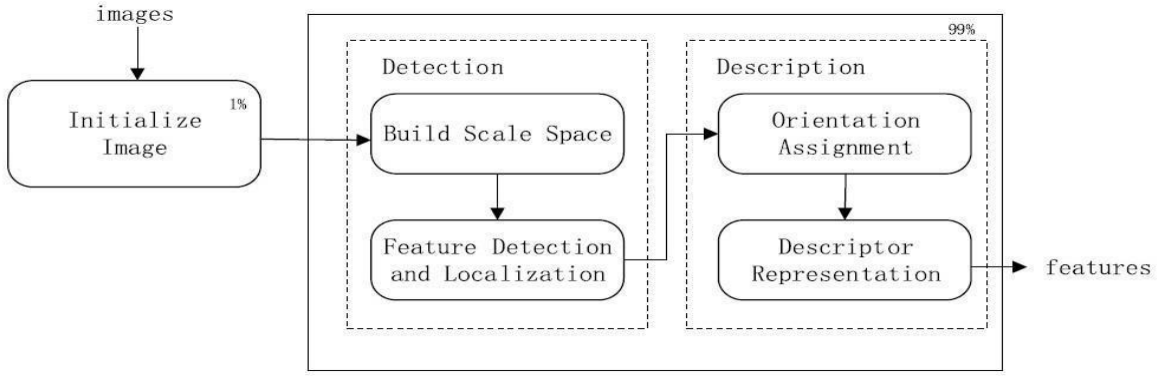

Fig 2: Work flow of SIFT/SUR

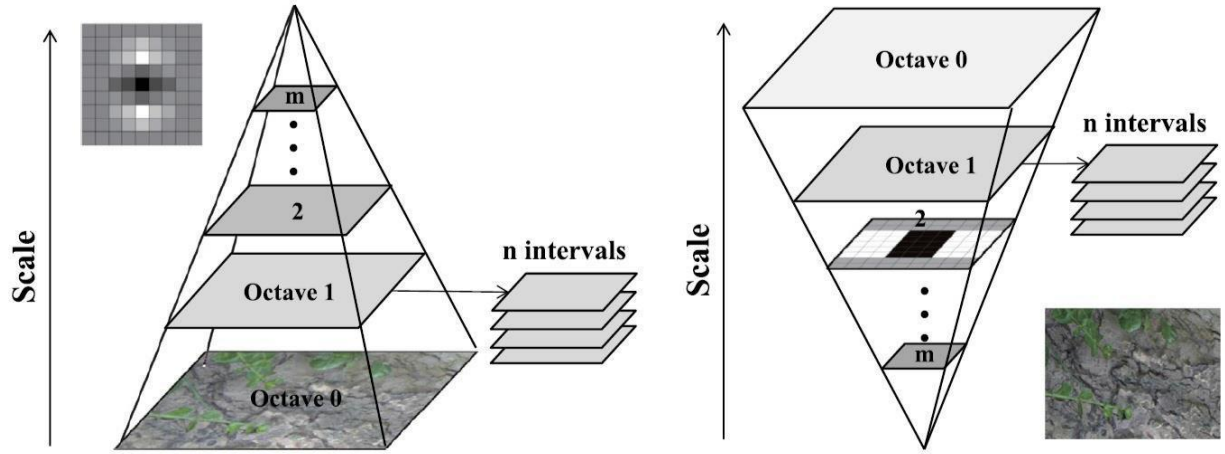

Fig 3: Scale space of SIFT

III TECHNIQUES

There are several techniques to parallelize the SIFT/SURF and some of them are discussed below.

Block Level

On block level we can split an input image into number of sub images and assign each sub images to threads as sown in Fig 4. [2] This can be done dynamically or statically.

In dynamic image splitting, we split the image into blocks depending upon the number of hardware threads available on the target CPU.

In Static image splitting we split the images into number of equal blocks and put them in queue. As soon as one thread finishes the process of one block of image it takes another block from the front of queue and process that block. This process will be completed if there are no more items in the queue.

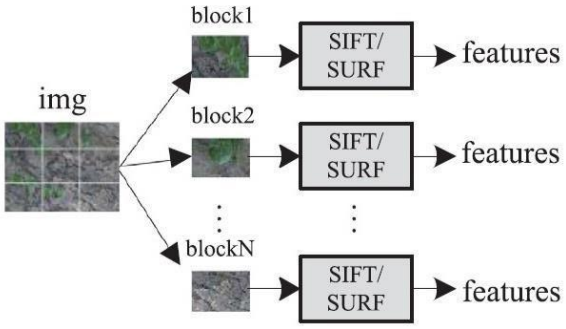

Fig 4: Block level Parallelism

Pros: Quick and easy, not much thread overhead.

Cons: Because of image splitting some key points will be lost in the split. The more the number of splits the more the feature or key points will be lost as show in Fig 5 as given in [2] ref. Hence it is necessary to find the balance in the split of image.

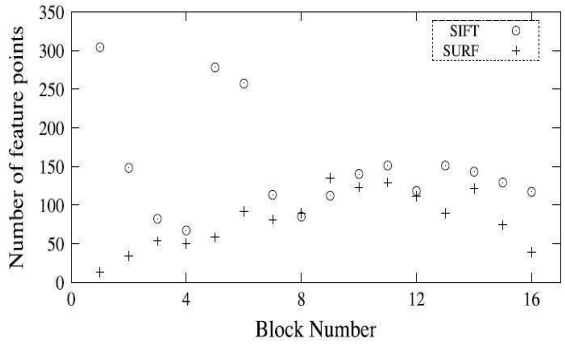

Fig 5: Block Level imbalance

Hence it is useful to use block level parallelism if a few missing feature points does not make the difference in the work.

Image level

In Image level parallelization, several images are divided into several groups and each group is assigned to threads as shown in Fig 6. We can divide the image into static groups or dynamic group.

In static grouping, each group is assigned to a particular thread and left to process. In dynamic grouping, images are taken from other groups as soon as the thread finishes and remains idle.

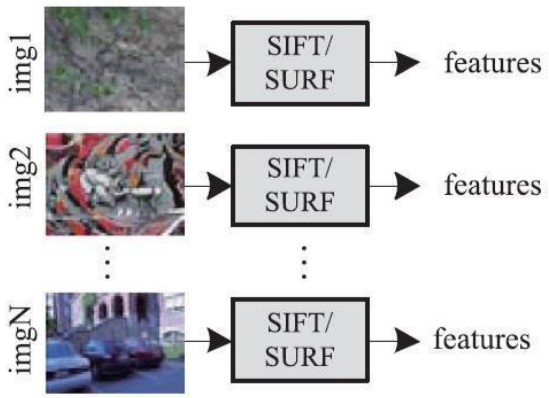

Fig 6: Image level parallelism

Pros: Feature points are not lost, depending upon the implementation, efficient use of processors can be achieved.

Cons: Some group might have a very high resolution image which has more feature points than others this leads to imbalanced workloads as shown in Fig 7 [4].

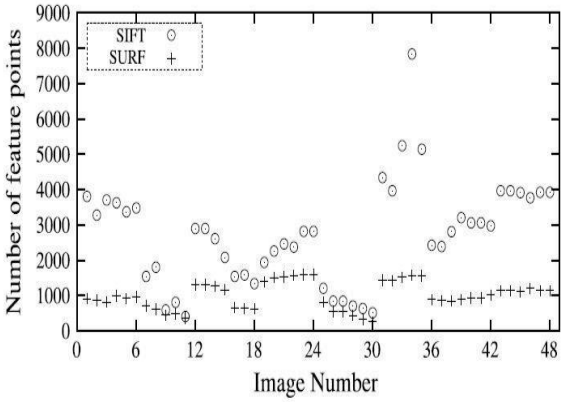

Fig 7: Image level imbalance

Image level parallelism is useful when we have to process large number of images in data centers image processing and feature extraction or video feature extraction.

Scale Level

Since scale space pyramids are constructed first and then feature points are extracted from the pyramid [2][4]. It is possible to extract key points from the scale space independently. Thus scales can be processed concurrently as show in workflow in Fig 8.

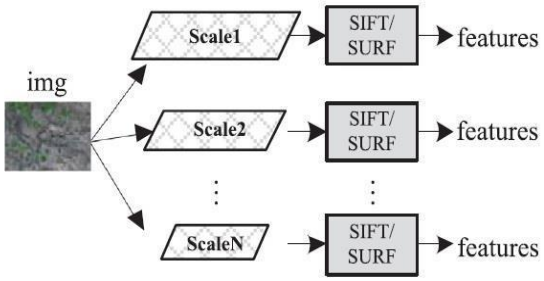

Fig 8: Scale Level parallelism

During building of the scale space pyramid, the workload of each scale are also imbalanced, hence it is not recommended to use this method as it is complex in design and execution.

Pipeline

Since SIFT/SURF has two main stages detection and description, these two stages can be distributed among different threads and the data from one stage of one thread can be pipelined to another stage on another thread. Input images are treated as flow from one thread to another, here the pipeline can be a form of queue or stack as shown in Fig 9.

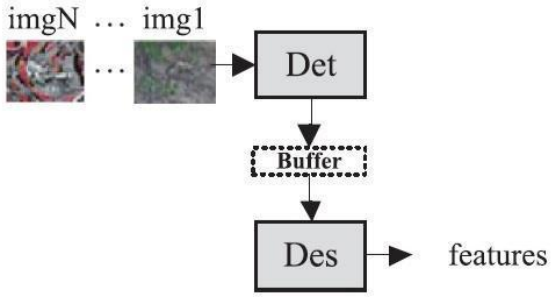

Fig 9: Pipeline parallelism

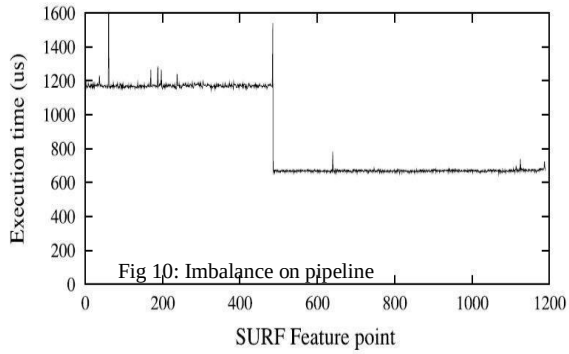

Fig 10: Imbalance on pipeline

Pros: This when combined with block level parallelism or image parallelism can achieve good results.

Cons: This has an extra overhead of thread creation, locks and thread synchronization. Hence the imbalance as shown in Fig 10 [2].

As shown above by assigning dedicated threads to dedicated tasks it is possible to achieve best performance. Hence this type of parallelism is recommended for use.

IV CONCLUSION

This paper presents different approaches to parallelize IFEA along with their pros and cons. The above mentioned techniques each have their own strength and weakness and are best used in combination with other techniques. Apart from these techniques other algorithms apart from SIFT/SURF, like ORB and LBP can also be explored.

REFERENCES

[1] David G Lowe, “Distinctive image features from Scale Invariant Key points”,
International Journal for computer vision, vol. 60, no. 2, pp. 91-110, 2004.

[2] Yunping Lu, Yi Li, Bo Songa, Weihua Zhanga, Haibo Chene, Lu Pengf, “Parallelizing image feature extraction algorithms on multi-core platforms”, Journal on Parallel and Distributed Computing, Elsevier, vol. 92, pp 1-14, 2016.

[3] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool, "Speeded Up Robust Features (SURF)”, European Conference on computer vision, 2006.

[4] K. Aniruddha AcharyaR. Venkatesh Babu Sathish S. Vadhiyar, “A real-time implementation of SIFT using GPU”, Journal of Real time image processing, Springer, pp. 1-11, 2014