Distributed Visual Processing Based On Interest Point Clustering

Xueyao Bai

May 24, 2015

Contents

Abstract

In this master thesis project, we study the problem in Visual Sensor Networks in which only limited bandwidth is provided. The task is to search for ways to decrease the transmitting data on the camera side, and distribute the data to different nodes.
To do so, we extract the interest points on the camera side by using BRISK interest point detector, and we distribute the detected interest points into different number of processing node by implementing proposed clustering methods, namely, Number Based Clustering, K-Means Clustering and DBSCAN Clustering.
Our results show it is useful to extract interest points on the camera side, which can reduce almost three quarters of data in the network. A step further, by implementing the clustering algorithms, we obtained the gain in overhead ratio, interest point imbalance and pixel processing load imbalance, respectively. Specifically, the results show that none of the proposed clustering methods is better than others. Number Based Clustering can balance the processing load between different processing nodes perfectly, but performs bad in saving the bandwidth resources. K-Means Clustering performs middle in the evaluation while DBSCAN is great in saving the bandwidth resources but leads to a bad processing balance performance among the processing nodes.

Chapter 1

Introduction

Today, as the wide deployment of wireless sensor networks (WSNs) has became more and more useful, the research of WSNs has gained plenty of attention. However, most of the attention have been concentrated on the collection of scalar data in terms of temperature, vibration, pressure, brightness, etc. In fact, as the technology growing fast and fast, merely scalar data is not sufficient for many applications.
As the processing speed of microchips has been explosively increased in the last decade, according to Moore's Law, the number of transistors in a dense integrated circuit doubles approximately every two years. Though this is only an observation, no one can deny the fast increasing of processing speed of microchips. Here the chance is, the traditional WSNs can extend the application to acquire and process multimedia signals such as still images and videos by using visual sensor networks(VSNs). The high possibility of implementing low-cost CMOS cameras enabled the chance of building low-cost VSN platforms which are able to capture, process and disseminate visual data collectively [1].

1.1 Visual Sensor Networks

VSNs are becoming more and more important due to the huge demand from the deployment of smart city. They highly support applications related to surveillance, tracking, and environmental monitoring. High-level analysis using object recognition and other techniques can intelligently track objects (such as people or cars) through a scene, and even determine what they are doing so that certain activities could be automatically brought to the operator's attention.
In the area of health protection, ambient assisted living and personal care applications of VSNs have great possibility in both commercial and social aspect. A variety of sensors (e.g., temperature, camera, sound detecter) and personal com-

puting devices, or even other objects such as TV or robot, would be connected in such applications. The sensors will record data and transmit to the user who can control the devices. In such VSNs, it is possible to improve life quality and also can help monitoring and assisting elderly or disabled people remotely
Other VSNs' applications can be for example, in terms of virtual reality, one can provide the service that the users can remotely visit interesting locations, such as museums deployed camera sensors via internet. Another possibility is the use of visual sensor networks in run time monitoring, where the network would automatically select the "best" view (perhaps even an arbitrarily generated one) of a live event.
VSNs also raise new challenges that have not been fully addressed by researches in WSNs. The use of high-resolution images and videos require more resources in both computational ability and transmission ability. Besides, In most cases, the VSNs will consume more power than WSNs, thus the study of energy saving in VSNs is necessary. Furthermore, the nodes (sensors) will also need to be capable of buffering lots of packets because the data flow in VSNs is larger than WSNs.

1.2 Thesis Objectives

As facing the challenge that the visual analysis is computationally intensive, and at the same time the transmission of the entire pixel information to a central server may need high transmission speed, the transmission and the processing needs to be optimized in the VSNs to realize high performance visual analysis.
Visual processing is typically performed in two steps. In the first step, the characteristic points, denote as *interest points* of the image are detected by the interest point detector, then the surrounding area of the interest point will be described by the descriptor.
Building on the hypothesis that interest points are clustered in the images, we will consider the scenario, where the camera performs the interest point detection, and transmits only the interest points of the image to the processing nodes. In this thesis, we will implement BRISK interest points detection.
Follow then, we will implement three clustering algorithms, namely Number Based Clustering, K-Means Clustering and DBSCAN Clustering to cluster the detected interest points and assign to different number of processing nodes.
Finally, we will evaluate the proposed clustering algorithms by considering metrics from Interest Point Imbalance, Pixel Processing Load Imbalance and Overhead Ratio, respectively.

1.3 Methodology

In this master thesis, we performed target-focused study in the preparing stage. That is, we've been always sticked to the analysis of different distribution schemes, thus we freed ourself from specifications of digital image process. In the implementation, we chose OpenCV 2.3 to process the images, and the dataset we utilized was taken from Nikon D3000 which contains 1161 pictures. The content of the images are arbitrary, which make the results of our study as general and reliable as possible. But it is also worth noting that the image content will affect the results dramatically, as the distribution of interest point highly depends on the image content.

Chapter 2

Visual Analysis

For most of VSNs, they consist of cameras, processing nodes, server nodes. as Figure 1 shows.
Runtime *visua analysis* thus is possible with the help of VSNs. In this chapter, we will give a overall review of *visual analysis*

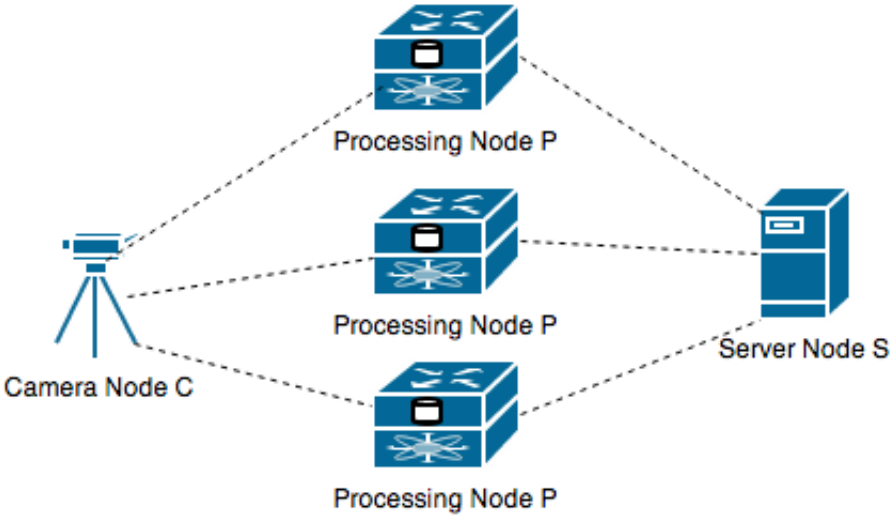


Figure 2.1: A typical VSN

2.1 Visual Analysis Tasks

Visual analysis cover a wide range of practical use. The visual analysis task can be generating cognitively useful visual representations of data, for example, diagrams

and graphs. 2D/3D coordinates often used to form the basis of visual designs in scientific data. In social media data, the relationship between news items or users is encoded by the network graphs. On the other hand, visual analysis of social media could open a wide range of promising applications, such as visual analysis could provide visual information for situational awareness for disaster response, or provide surveillance system that can tracking the object movement in the smart city, for both security purpose and life quality improvement purpose [2].
Another important visual analysis task is image retrieval. The image retrieval based applications can help us tremendously, such as they can help identify products quickly, compare items when we shopping, search information about movies, real estate, print media, or even artworks. In this master thesis, we will mainly focus on visual analysis based on image retrieval.
In the same time, there are also challenges posed by mobile image retrieval applications. Due to the limited processing resources on the camera node, and the limited bandwidth on the transmission node, it is critical to decide which part of the processing is suit to be performed on the camera node, and the others should be performed at the server, or even at the transmission node. On one hand, it is very bandwidth consuming to transmit a picture or frame with high resolution over a slow wireless link. Meanwhile, as mentioned before, even cheap camera nodes can handle some basic feature extraction of the salient image features.
A typical pipeline for image retrieval is shown in Figure 2.2. First the local features are extracted from the query image. The set of image features is used to assess the similarity between query and database images [1]. Second, the query features are quantized and sent to the database. In the database, the quantization cells is precomputed, a list of database images containing the quantized feature vector are associated with the quantization cells[3]-[6].
Finally, a geometric verification(GV) step is applied to the most similar matches in the database. The GV finds a coherent spatial pattern between features of the query image and the candidate database image to ensure that the match is plausible [1].

The performance of image retrieval can be measured by taking three metrics into account, namely, **retrieval accuracy**,**system latency** and **energy consumption**. In [1], the authors defined the percentage of query images correctly retrieved as *recall*, a good performance in terms of retrieval accuracy is indicated by a high recall value at a negligibly low false-positive rate. System latency consists three components, processing delay on client, transmission delay and processing delay on server. Processing delay depends on the algorithms used on the client and server side. The network situation determines the transmission delay, The transmission time in WLAN is very short compare to 3G network since WLAN has high bandwidth. Meanwhile, energy consumption varies a lot in different type of devices,

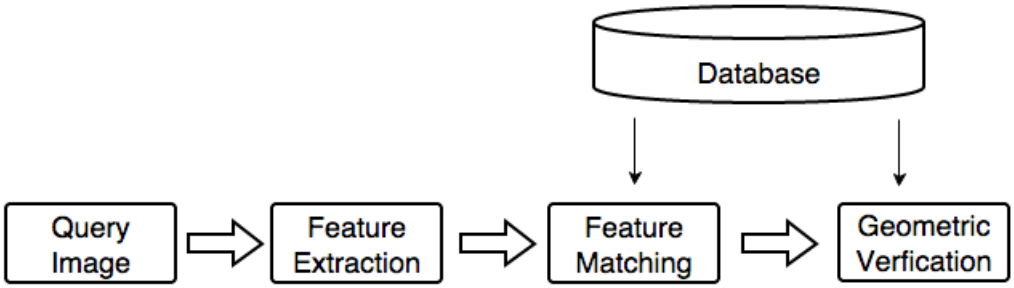


Figure 2.2: Pipeline For Image Retrieval

thus we somehow will not consider more on energy consumption in this thesis.

2.2 Feature Descriptor Based Visual Analysis

Feature descriptors can be obtained from three steps: interest points detection, orientation identification, descriptor extraction. The requirements that the descriptor should achieve are $scale-invariance$ and $rotation-invariance$. Scale-invariance ensures that the visual analysis would not be disturbed significantly by the distance between the camera and the objects. Meanwhile, rotation-invariance ensures that the visual analysis is free from the angle of camera to objects, together with scale-invariance, the robustness of feature descriptor based visual analysis thus can be achieved.

Interest Points Detection When we talk about interest points, we often refer to two kind of features in the image, one is corner and edge another is blob. Not only because these two features are visually noticeable, but also when performing pixel scanning, they are often with significant intensity changes.

There are several tools can be used as interest point detector. For example, Lowe's SIFT detector is widely used and proven as one of highest quality detector that provide promising distinctiveness and invariance. The combination of FAST interest point detector and BRIEF approach is suit for real-time applications since it can calculate fast when processing images. Another worth-mention method is SURF, because SURF has been proven to be a robust and fast method.

In this thesis, we are going to use Binary Robust Invariant Scale Key-points (BRISK) as the interest points detector. Because comparing with SURF, BRISK can also achieve comparable quality of matching while consume less computation time (less computation complexity) and generate short descriptors.

Orientation Identification Orientation identification is the method used to

achieve rotation invariance. To fulfill this task, a common way is changing the patch around each interest point to canonically oriented in the direction of the dominant gradient. But it is noticeable that BRISK does not do like this way. Instead, BRISK will identify the characteristic direction of each interest point to make the orientation-normalized descriptors. We will introduce more details in Chapter 5.

Descriptor Extraction The aim of descriptor extraction is to extract the salient features from the image to support the visual matching or image retrieval. A good descriptor first should be robust, in terms of scale-invariant and rotation invariant. It is easy to image that if two cameras in two different places captured the pictures from the same object, but the descriptors extracted from different pictures (i.e. different scale, different angle) are showing that "they" are not the same object, then the descriptors are totally worthless. Despite of robustness (i.e. scale invariant and rotation invariant), the descriptors in visual analysis are required to be discriminative since it would not be helpful if the descriptors occur in every image. There are mainly two categories of descriptors in use today. One of them belongs to non-binary descriptor such as SURF or SIFT, another is binary descriptor for example BRIEF or BRISK.

Chapter 3

Distributed Processing in VSNs

Distributed processing in VSNs is very important since it helps improving the performance of VSNs tremendously. For the camera nodes with limited processing ability, process most part of the image could take lots of time. Thus one would like to process the image on the processing nodes or on the server. On the other hand, the power resources and the transmission bandwidth is limited, so transmit only part of the image (interest area) would prolong the lifetime of VSNs and decrease the latency.

3.1 Delegation of Feature Computation

Delegation of feature computation has been proved to be a efficient method of improving the performance of VSNs. We can delegate processing steps from camera node C to processing node P in order to balance the work-load of VSNs. There are three ways of delegation as follow:

Area-Split In area-split, camera node C can delegate a part of image i, $G_{i,j}$ to a processing node $j \in P$. The $G_{i,j}$ is a cluster of interest points that is clustered by the clustering algorithms we proposed.

Scale-Split In scale-split, camera node C can delegate the octave parameters (effectively the scales) to a processing node of which can be used for detecting interest point. In such case, the interest points are detected through out the octave layers, which requires the transmission of the whole image.

A combination of area-split and scale-split can be the third way of delegation of interest point detection.

In general, for delegation of orientation identification and delegation of descriptor extraction, the pixel data of $A_{i,k}$ and $R_{i,k}$ are required to be transmitted, respectively. Since we are focusing on BRISK, so the $A_{i,k}$ and $R_{i,k}$ are the same in this

case.
The delegation of the processing steps affects the use of the computational and communication resources of the VSN nodes: the data transmission from C to P, the computational load of C and that of the nodes in P, and the data to be transmitted from P to S. These three are strongly coupled, and therefore the delegation needs to be optimized [3]. There are four schemes that can be used to off-load the camera node C:
No Detection/ No Extraction (ND/NE): In this case, the camera node C simply does not do anything about detection and extraction, thus the whole image will be transmitted to processing node. As illustrated in Figure 3.1.
Partial Detection/ Partial Extraction (PD/PE): Some of the interest points

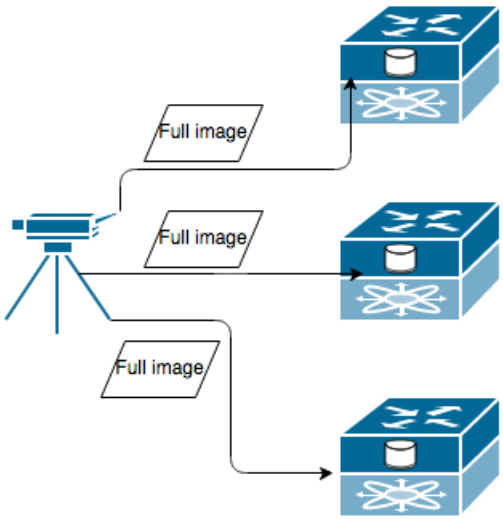


Figure 3.1: ND/NE scheme, the entire image is transmitted to processing node

are detected in C and also the related descriptors are extracted in C. The redundancy of $area - split$ can be decreased through detecting and extracting interest points at large scales. As shown in Figure 3.2.
Complete Detection/ No Extraction (CD/NE): In this case, all the interest points are detected in C and the descriptor extraction are delegated to the processing nodes, which means the camera node C only needs to send the pixel data for each interest area $A_{i,k}$. If we require the camera node to compute orientation, then only the pixel data of $R_{i,k}$ is required. The vectors of interest point locations and scales are required to be transmitted as well, as shown in Figure 3.3
Complete Detection/ Partial Extraction (CD/PE): In this case, the camera detects all the interest points and performs extraction for some of the descriptors.

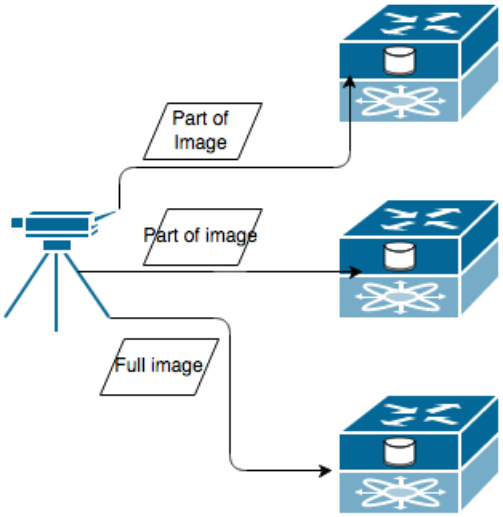


Figure 3.2: PD/PE scheme, part of the image is transmitted to processing node

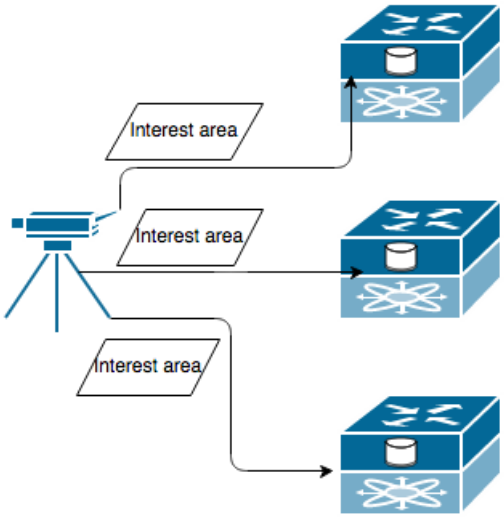


Figure 3.3: CD/NE scheme, interest areas of the image are transmitted to processing node

The set of interest points for which descriptors are extracted at the source should be chosen to minimize the remaining image pixels to be transmitted. Especially, when interest areas do not overlap, this scheme performs best.
For all these four schemes above, there are two factors can effect the performance.

First, from the topologic point of view, the computational and transmission resources of the VSN is critical. Second, from the point of view that the locations and scales distribution on the image.
The load of the processing nodes and the amount of data that need to be transmitted in the network are affected by the number of interest points K_i in an image. Based on the researches in [2], we find that the density of BRISK interest points decreases exponentially as a function of the detection threshold in the considered range, as illustrated in Figure 3.4.

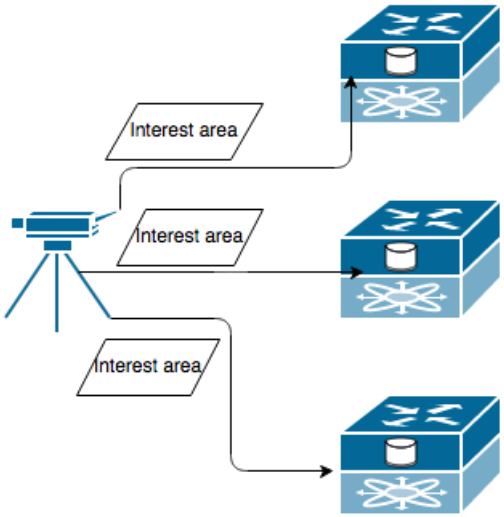


Figure 3.4: CD/PE scheme, interest areas of the image are transmitted to processing node, as well as some of the descriptors extracted by camera

3.2 Clustering based off-loading

In this thesis, we are going to consider the scenario that the interest points are somehow clustered in the image, and the camera performs the interest points detection, and only transmit the interest area to the processing node in order to perform descriptor extraction. Considering the delegation schemes aforementioned, CD/NE scheme is more likely to work well on our case. We will propose the algorithm and the give the evaluation in Chapter 7.

Chapter 4

Related work

Related work will be reviewed in this chapter, structured as follow: First, we will review related descriptor design for VSNs. In this thesis, we are going to use BRISK as the interest point detector and feature descriptor, while the SURF and other descriptors are also competitive and perform well in some situation, thus we will also give an introduction about SURF, and brief review about others.
Second, we will review networking performance of VSNs, with respect to the paradigms of visual analysis in VSNs.
Finally, a review of distributed processing will be presented.

4.1 Descriptor design for VSNs

4.1.1 Interest point detection

The approach of detecting salient interest points is searching for locations with pixel intensity changes. Blobs and edges are typic interest points.To realize the interest point detection, several interest point detectors have been proposed. In 1988, Harris corner detector was invented and soon became the most widely used detector in the worldwide[7]. In this detector, eigenvalues of the second moment matrix is used to support the corner detection. But in Harris corner detector, scale variety affects the performance dramatically. The concept of automatic scale selection was introduced by Lindeberg in [8]. It enables the detection of interest points in an image with the characteristic scale on their own.Detecting blob-like structures were the objective of Lindeberg's experiment. He made use of both the Laplacian and the determinant of the Hessian matrix in the detection. Latter on, this method was refined by Mikolajczyk and Schmid [9]. They created a robust, scale-invariant and with high repeatability detector which was based on Harris-Laplace and Hessian-Laplace. In their detector, a Harris measure or the

determinant of the Hessian matrix was used to select the location and the Laplacian was used to select the scale. Taking the speed as the first priority, a way of approximating the Laplacian of Gaussians (LoG) by a Difference of Gaussians (DoG) filter was proposed by Lowe[10].
There are also other proposed scale-invariant interest point detectors. Kadir and Brady[11] experimented the salient region detector that maximizes the entropy in the region. Jurie and Schmid[12] proposed an edge-based region detector.
Conclusion of the overall detectors can be made based on [13,14] that a Hessian-based detector is usually more reliable and repeatable than a Harris-based detector. Observation can also be made that approximations such as DoG can have a better trade-off between bringing speed up and lost accuracy.

4.1.2 Interest point description

Descriptor extraction can be performed as long as the interest points are detected. Interest point descriptors thus are required to perform descriptor extraction. There are even much more feature descriptors have been proposed so far. For example, Gaussian derivatives in [15], moment invariants in [16], complex features in [17], steerable filters in [18], phase-based local features in [19]. Especially, The distribution of smaller-scale features within the interest point neighborhood proposed by Lowe[20], has been proven that performs better than the others in [21]. The reason for that is the large amount of information regarding the spatial intensity patterns were captured and at the same time is robust against small deterioration or localization errors. Another important descriptor is Scale-Invariant Feature Transform (SIFT) proposed in [20]. A histogram of local oriented gradients around the interest point is computed and the bins with 128D vector was stored in SIFT.
Based on SIFT scheme, there are lots of refinements coming out through out the time. For example, a method of applying PCA on the gradient image around detected interest point was proposed in [22] by Ke and Sukthankar. This method is fast for matching since it only involve 36D vector. But it is shown to be not distinctive enough compare to the method proposed by Mikolajzcyk and Schmid[30]. And it shows that applying PCA will slow down the feature computation. Mikolajzyk and Schmid also proposed a variant of SIFT which name is GLOH, it has been proved to be reliable with the same number of dimensions. But the problem here again is GLOH uses PCA for data compression thus it is computationally expensive.
The SIFT descriptor is distinctive and fast thus it is used widely for online applications. Seet al. [23] made use of SIFT on a Field Programmable Gate Array (FPGA) and improved its processing speed. At the same time, Grabner et al. [24] used integral images to approximate SIFT as well. Compared with SURF, both

of them achieved the same processing speed while SURF can ensure a relatively high quality.

4.1.3 SURF: scheme and analysis

Interest point detection The SURF utilizes the basic Hessian matrix approximation. By using the integral images to fit in the more general framework, the speed can be improved dramatically.
Integral images We denote the entry of an integral image as $I_{\sum}(x)$, it represents that at a location $x = (x, y)^T$, $I_{\sum}(x)$ equals to the sum of all pixels in the image I within a rectangular region.

$$I_{\sum}(x) = \sum_{i=0}^{i \leqslant x} \sum_{j=0}^{j \leqslant y} I(i, j) \qquad (1)$$

The integral image is a very powerful kit because it makes it possible to releases the calculation time from the image size. One can easily do this by taking three additions to calculate the sum of the intensities over any upright, rectangular area. Figure 3 shows the calculation of integral image.

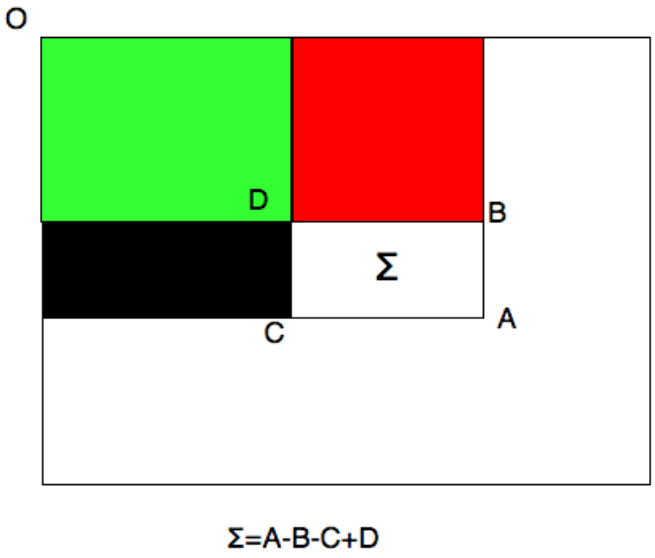


Figure 4.1: We can see that the sigma region can be easily calculated by the sum of A minus sum of B minus sum of C, because D is overlap twice, thus we plus D once at last as compensation

Hessian matrix interest points and SURF

SURF applies Hessian matrix on its detector because the Hessian matrix performs good in accuracy. Compared with Hessian-Laplace detector proposed by Mikolajczyk and Schmid[26], SURF detector mainly detects blob-like interest points where the determinant is maximum, and the determinant of the Hessian is used by SURF detector as well as for scale selection.
Scale space representation Both BRISK and SURF requires that the interest points should be detected at different scales. In this way, the scale-invariance can be assured. Scale spaces, both for SURF and BRISK, are implemented as an image pyramid. The scale spaces are divided into octaves. Consecutive filter response maps which are calculated by convolving the same input image with a filter of increasing size are represented by octaves.
Interest point localisation A non-maximum suppression in a 3 neighborhood then is applied in order to localize the interest points over scales in SURF. **Interest point description** In SURF, the distribution of the intensity content within the interest point area is described by the descriptors. Rather the gradient information extracted by SIFT, SURF use the distribution of first order Haar wavelet responses in x and y direction, make use of integral images for speed, and merely use 64D. This boosts the computation speed as well as bring the robustness. The steps of doing this is, first use the information from a circular area encompass the interest point to fix a reproducible orientation. Second, Extract the SURF descriptor from a square region corresponding to the chosen orientation. Last step, match the features between two images.

4.2 Networking performance of VSNs

4.2.1 Paradigms for image analysis in VSNs

Generally, the basic visual analysis is performed in two steps, as Figure 4.2 shows. First, the camera capture the image and compresses the image using some compressing algorithms, for example JPEG or H.265/AVC, in order to transmit efficiently over networks. Visual analysis is performed in the following. Although the compress-then-analysis approach has been applied on tons of applications successfully, but the flaw is the compression may impair the critical features when analyzed by the central. The reverse of traditional CTA paradigm is $analyze - then - compress$ paradigm, which first extracts features from original image, then compress the extracted features with a suitable coding scheme and transmit them to final destination, as illustrated in Figure 4.3.
The work in [x] compared the ATC and CTA with their rate-accuracy performance for image retrieval. The results in [x] shows that the performance of ATC and CTA

depends on the circumstance of the network. The ATC paradigm obtained good results at low bitrates, meanwhile the CTA performs well at high bitrates [27].

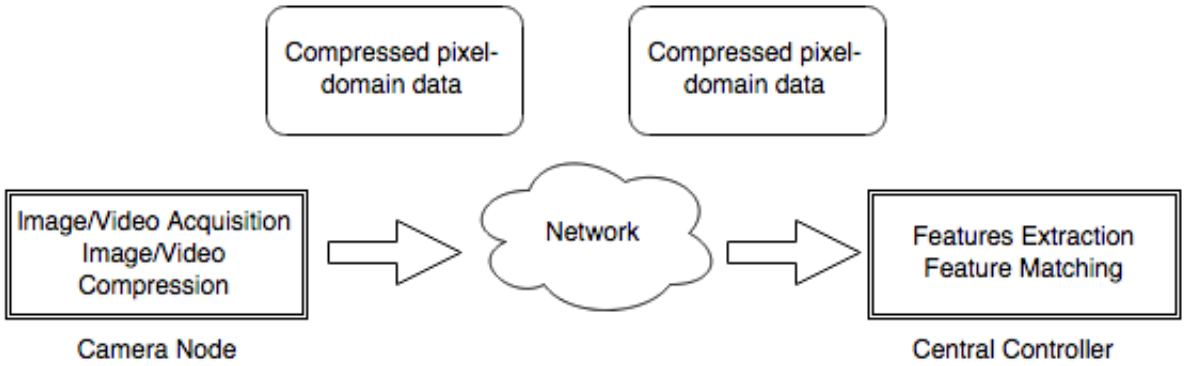


Figure 4.2: (a) Compress-then-analyze (CTA) paradigm

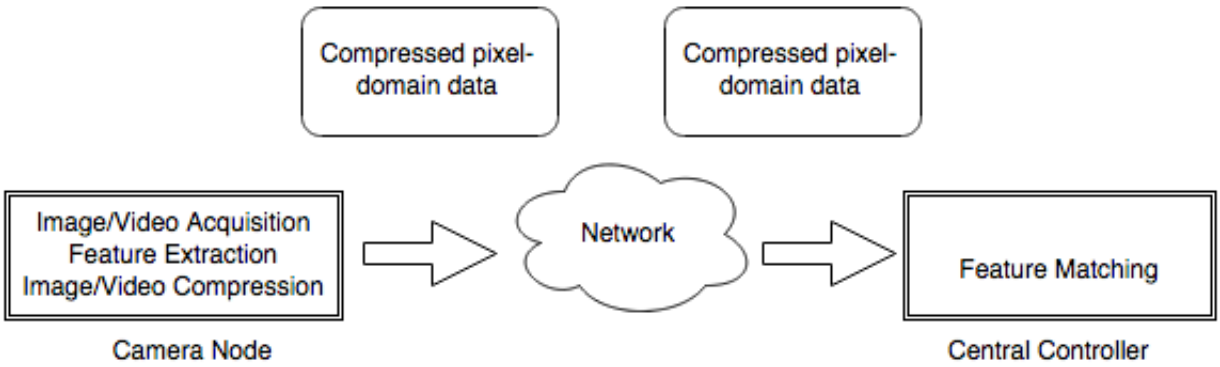


Figure 4.3: (b) Analyze-then-compress (ATC) paradigm

4.2.2 Distributed processing

Distributed systems today is a big concept that refers to the different modern network-based functional systems. Distributed processing plays an important role in such systems. There are three essential characteristics of a distributed system, namely *Concurrency*, *Synchronisationintime*, *Failures* [28].
Specifically, in VSNs, we consider more on the workload of each node of VSNs as the VSNs can provide only limited resource. In [4], Muhanmmad, Gyorgy, and Fodor proved that some delegation schemes can balance the workload of VSNs together with some processing schemes such as $area - split$ and $scale - split$.

Area-split: Spatial distribution of interest points

How the interest points distribute in an image is very important when we refer the efficiency of the $area - split$.

Characteristics of Spatial Distribution When looking into the interest points distribution in X-coordinates, the work in [2] revealed it almost obey the uniform distribution. If one knew the marginal distribution of the X,Y coordinates of interest points, one could perform area-split with equal number of interest points. To do so, [4] introduces the correlation between the X and Y coordinates. The results show that the average coefficients $\rho_{X,Y}$ is zero, which means there is no significant relation between X,Y coordinates. Thus, knowing the marginal distributions would not help balancing the processing load in $area - split$.

Processing load imbalance The research in [2] shows that it is useless to divide the image according to the marginal distribution of interest points. Another way of doing this is one can split the image into P pieces of subareas equally and delegate each area to a processing node. By defining imbalance value H, the results in [2] show that the most loaded processing node would take three times more workload over the average value.

Taking the Top-M extraction scheme into consideration for two cases, one is the ideal situation that the processing nodes P know the threshold Θ that can lead to the detection of interest point number is M. The results show that the number M of interest points increases lead to the decreases of the imbalance H, which suggest that more interest points M can do good to balancing the load in the network.

Another case is the practical case that the processing nodes P have no idea about the threshold that can detect the M interest points.Thus they need to calculate which of the interest points in its own area are the Top-M interest points.

In [4], researchers concluded that Area-split could become an efficient solution to delegate the feature extraction tasks to the processing nodes if the camera node could obtain the location of the interest points by performing complete detection, i.e., CD/NE and CD/PE. Alternatively, the camera node could try to predict the interest point distribution if the subsequent images are correlated, for instance, in the case of visual analysis of video sequences.

Scale-split: Octave distribution of interest points

Another method of analyzing interest points distribution is $scale-split$. In $scale-split$, one would consider the interest points over different octave layers the are detected at. When each processing node performs detection based on the scale space, this distribution plays an important role for understanding the feasibility of $scale - split$.

In ND/NE scheme, the parameters of the detection algorithm thus are needed to

support the delegation. The work in [4] use the default parameters of BRISK with 4 octave as well as intra-octave layers, denoted as c_0,d_0,c_1...d_3. The result shows that most of the interest points are detected at low octaves/layers, at c_0 in the case of BRISK. As the threshold θ increases, the share of interest points detected at low octaves/layers decreases.
The result in [4] also shows that the average and 5-95 percentile of the probability mass function (PMF) of the interest point distribution across the octave layers are very skewed. Further more, the distribution remains skewed even under the Top-M scheme.
Only using $scale - split$ is not suitable for balancing the load under ND/NE and PD/PE, as the consequence of the skewed octave layer distribution. But as being proved in [4], even when combined with $area - split$, the performance of $scale - split$ is not good.

Chapter 5

BRISK:Binary Robust Invariant Scalable interest points

BRISK is a novel method for resolving computer vision problem of detecting, describing and matching image interest points. Comparing with other methods mentioned in related work such as SIFT and SURF, BRISK provides a much more faster alternative while still keeping comparable matching performance. The feasibility of BRISK lies on the easy-configurable circular sampling pattern which computes brightness comparisons to form a binary descriptor string. Thus, BRISK is adaptable for the cases which require real-time constraints or in the situation of only limited computation power is available [31].

5.1 Methodologies and Schemes of BRISK

The key steps in BRISK are similar with other tools, but instead of orientation identification, BRISK takes a step named orientation normalizing to generate orientation normalized descriptors to achieve rotation invariance.

5.1.1 Scale-Space interest point Dectection

BRISK takes advantages of AGAST done by Mair et al [29] which is substantially an powerful extension of FAST. Based on that, BRISK searches for maxima in scale-space using the FAST score s as a measure for saliency rather than only searches in the image plane. Moreover, the BRISK detector uses continuous scale-space to estimate the real scale of each key point rather than in discrete scale axis. The scale-space pyramid layers in BRISK consist of n octaves c_i and n intra-octaves d_i, for $i = 0, 1, ..., n-1$,for most cases, $n = 4$. Above and below each octave c_i, there are intra-octaves d_i and d_{i-1} enclosing them. The scale space t is

represented as $t(c_i) = 2^i$. BRISK uses the 9-16 mask shape to detect potential key point. A point is considered to be key point if at least 9 consecutive pixels in the surrounding 16-pixel circle to be brighter or darker than the central pixel [31].
The process of detecting key points through out the octaves and intra-octaves is as follow: First, applying the FAST 9-16 detector on each octave and intra-octave using the same threshold T, to obtain the potential candidates of key point. Second, these candidates are constrained by a non-maxima suppression in sacle-space. The point needs to meet the maximum condition concerning its 8 neighboring FAST scores s in the same layer. The score s is the maximum threshold for judging if a point in the image should be considered as a corner [31].
BRISK considers that the image saliency as a continuous quantity not only through out the image but also across the scale dimension. For each detected maximum, a sub-pixel and continuous scale refinement are performed.

5.1.2 interest point Description

The BRISK descriptor is composed as a binary string by concatenating the results of simple brightness comparison tests out of a set of interest points. The characteristic direction of each key point is identified for the purpose of normalizing the orientation of descriptors thus to ensure that the the descriptor is rotation invariant [31].

Sampling pattern and rotation estimation

In BRISK, sampling the neighborhood of the interest point is the critical step which utilized the same pattern used in DAISY descriptor [30].
However, BRISK uses that pattern in a different way. The pattern as figure 5.1 shows, defines N locations which are located equally on circles concentric with the interest point.

BRISK applies Gaussian smoothing with standard deviation σ_i to the distance between the points on the respective circle around a point p_i in the pattern to avoid aliasing effects. Considering one of the $N \times (N-1)/2$ sampling-point pairs (p_i, p_j). The local gradient $g(p_i, p_j)$ can be estimated from the smoothed intensity values $I(p_i, \sigma_i)$, $I(p_j, \sigma_j)$ by

$$g(p_i, p_j) = (p_j - p_i) \times \frac{I(p_j, \sigma_j) - I(p_i, \sigma_i)}{||p_j - p_i||^2} \tag{5.1}$$

BRISK defines subsets of short-distance pairings S and long-distance pairings L respectively:

$$S = \{(p_i, p_j) \in A | \; ||p_j - p_i|| < \delta_{max}\} \subseteq A \tag{5.2}$$

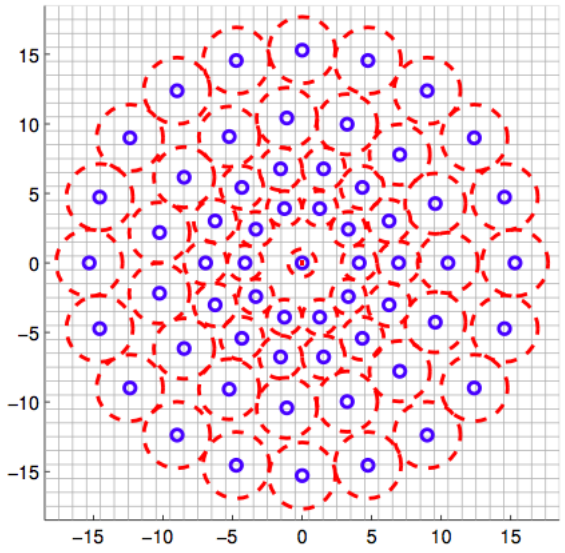


Figure 5.1: BRISK Sampling Pattern [31]

$$L = \{(p_i, p_j) \in A | \; ||p_j - p_i|| > \delta_{min}\} \subseteq A \tag{5.3}$$

where A is the set of all sampling-point paris:

$$A = \{(p_i, p_j) \in \mathbb{R}^2 \times \mathbb{R}^2 | i < N \wedge j < i \wedge i, j \in \mathbb{N}\} \tag{5.4}$$

Then the overall characteristic pattern direction of the interest point can be iterated through the point pairs in L by taking $\delta_{max} = 9.75t$ and $\delta_{min} = 13.67t$

$$g = \begin{pmatrix} g_x \\ g_y \end{pmatrix} = \frac{1}{L} \cdot \sum_{(p_i, p_j) \in L} g(p_i, p_j). \tag{5.5}$$

This estimation takes the long-distance pairs because in [31] it was confirmed that the local gradients annihilate each other thus does not influence the global gradient determination.

Descriptor construction

The descriptor of BRIKS is a binary string, denoted as d_k. By performing all the short-distance intensity comparisons of point pairs $(p_i^\alpha, p_j^\alpha) \in S$, each bit b in d_k corresponds to:

$$b = \begin{pmatrix} 1, & I(p_j^\alpha, \sigma_j) > I(p_i^\alpha, \sigma_i) \\ 0, & otherwise \end{pmatrix} \tag{5.6}$$
$$\forall (p_i^\alpha, p_j^\alpha) \in S$$

The noticeable characteristics of BRISK descriptor are, first BRISK takes a deterministic sampling pattern which yields a uniform sampling-point at a given radius encompassing the interest point. In consequence, The information content of brightness comparison will not be distorted by the Gaussian smoothing. Moreover, BRISK constraints the complexity of looking-up intensity values by using fewer sampling-points than pairwise comparisons. At last, the brightness variations are well limited (i.e. they are only need to be locally consistent) because the comparisons are constrained spatially [31].

5.2 Implementation of BRISK with OpenCV

The implementation of BRISK is based on OpenCV 2.3. The reasons we choose OpenCV as the implementation tool are from different point of view. The most important reason is OpenCV can work across platform, thus it is possible for us to immigrate to other platform in the future as the smart camera nodes may run on different platform. Another reason is OpenCV provides more functional image storage as we can use cv::Mat class as the container of the image. and we run BRISK in OpenCV with C++, thus C++ provide a nice container Vector to store detected interest point. Moreover, OpenCV will accelerate itself if Intel's Integrated Performance Primitives on the system is detected.

5.2.1 Brief Introduction of OpenCV

OpenCV, the open source computer vision library, is a library of programming functions mainly aimed at real-time computer vision, developed by Intel Russia research center in Nizhny Novgorod, and now supported by Willow Garage and Itseez [32]. It is free for use under the open source BSD license. The library is cross-platform. It focuses mainly on real-time image processing.

5.2.2 Implementation Introduction

The image database we use in this thesis is from Nikon D3000, which contains 1161 images and most of the image size is 2400×1900, storage size is between 2 MB to 4 MB, enabling us to observe more than 1000 keypints in most of cases. The general processing flow can be described as follow:
First, load the image using **cv::imread**, and store the image in **cv::Mat**. Second, we construct a BRISK detector instance using BRISK's constructor such like **cv::BRISK detector(threshold, octavelayers, patternScale)**. Finally, we store the detected interest points using **detector.detect(image, interestPointsContainer)**. As we mentioned before, C++ provides a good container

Vector to store the interest point. The basic function of BRISK thus can be realized by these steps.

Chapter 6

Clustering of BRISK Interest Points

6.1 Motivation of Interest Point Clustering

We are motivated by the work of [4], we first analyze the characteristics of interest point detected by BRISK as the trend function of detecting threshold, as figure 6.1 shows. As we can see, the number of interest point and density of interest point decrease along with the increasing of detecting threshold exponentially. This result suggests that increase an already high threshold value will not help decreasing the number of detected interest point sufficiently thus will not help decreasing the computation and transmission load in VSN. Considering the Top-M interest point scheme, we will need to gather sufficient interest point to calculate the descriptor, thus we are going to use $\Theta = 60$ as the default threshold.

Next, we consider the characteristic of $area - split$. As we have introduced $area - split$ in Chapter 3, the easiest way of realizing $area - split$ is splitting the image equally. In order to obtain the information of distribution of interest point in each of the sliced image, we use L as the interest point imbalance [4].
The definition of L is

$$L_i = \frac{\max_{1 \leq j \leq P} \quad n_{i,j}}{K_i / P} \tag{6.1}$$

Where $n_{i,j}$ denotes the number of interest point in subarea j for image i, P denotes the number of subarea and K_i denotes the total number of interest point in image i.
The value of L_i implies that when L_i closes to 1, the interest point in image i is almost balanced, e.g, each piece of image contains the same amount of interest point whereas $L = P$ indicates that all the interest points fall into the same sliced image $n_{i,j}$.

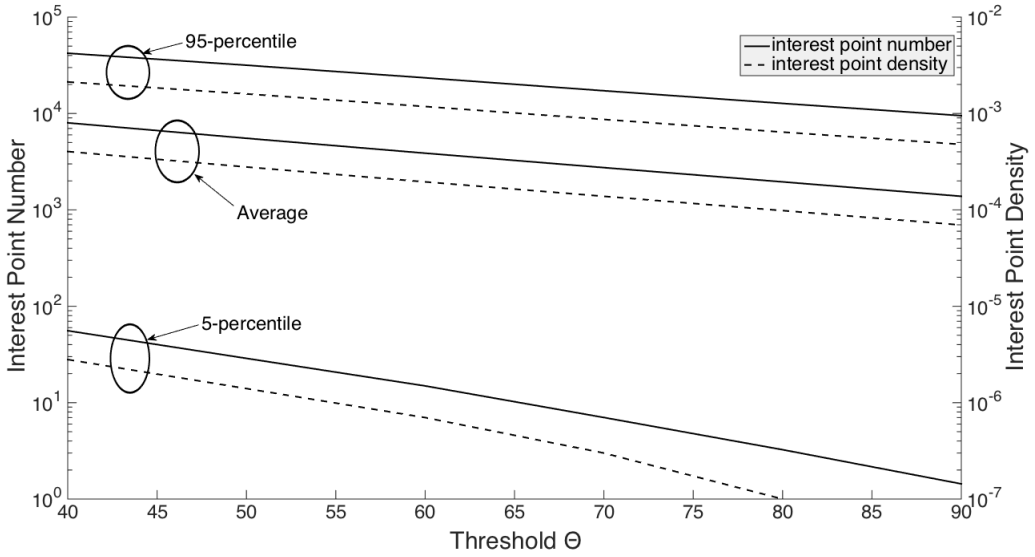


Figure 6.1: Number and Density of Interest Points varying with Threshold

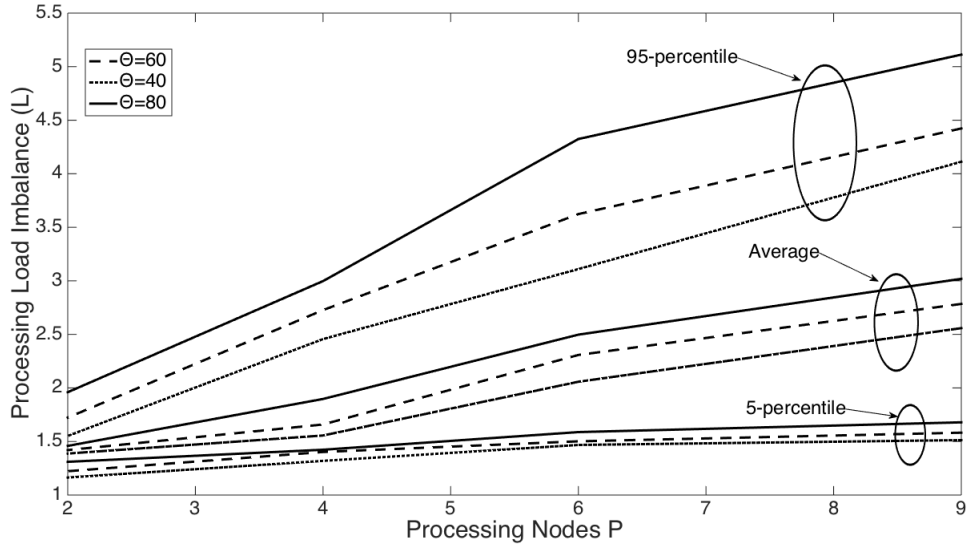


Figure 6.2: Load Imbalance as the Function of Processing Nodes

Figure 6.2 presents the statistic interest point imbalance of processing all the images from the image dataset. The result shows that the processing imbalance will increase if we cut the image into more pieces. The more we cut the image, the higher the processing imbalance will reach, suggesting that there are some parts of

the image that may contain more interest than other parts. Thus we can consider extracting images from the sliced image with higher number of interest points. Another factor that we could consider is we calculate the needed area of the image, and investigate how large of the empty area can cover over the original image. By doing this, we first define the needed area ratio R as

$$R = \frac{||\bigcup_{k=1}^{K} A_k||}{S} \tag{6.2}$$

Where A_k represents kth area of the interest point in the image. S is the size of the image. The results in Figure 6.3 shows that the mean empty area ratio is 0.2541, suggesting that almost 3/4 area of a random image is not covered by the interest area. Considering this phenomena, we can combine the processing load imbalance with it and cut the image into pieces in some manners to distribute the pixels to each of the processing nodes after detection on the camera node.

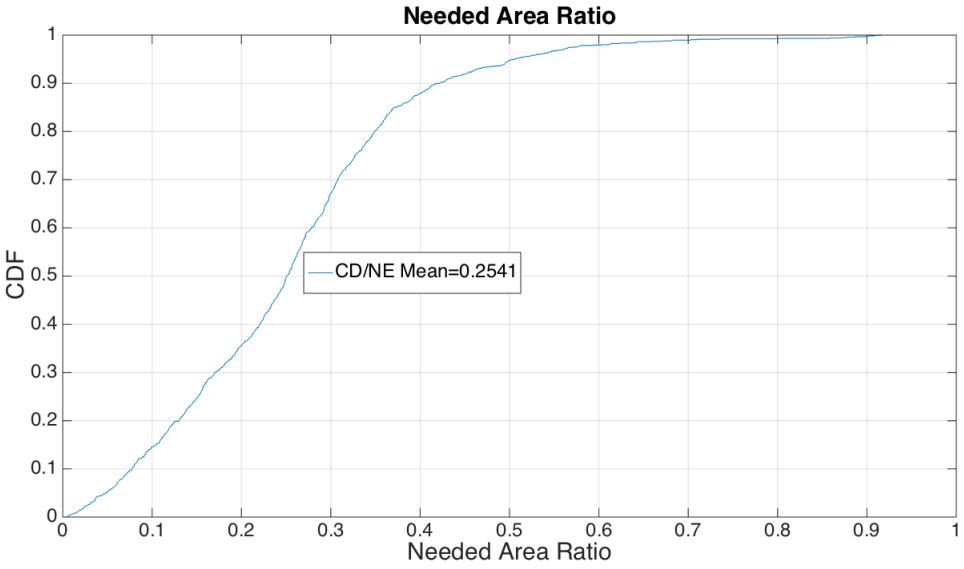


Figure 6.3: Needed Area Ratio of CD/NE

6.2 Interest Point Number Based Clustering

Motivated by the results we obtained above, we consider the scenario that we cut the image into pieces in which each of them containing the same number of the interest points, and we will examine two statistic properties.

6.2.1 Division Overhead

Consider the situation, we cut the original image into P pieces, and the *overhead* due to cutting is the pixels that belongs to many divisions, that needs to be transmitted many times.
Figure 6.4 demonstrates how we calculate the overhead. Here we divide the image into 1,2,3,4 four pieces, and the two interest area overlaps with each other, namely a,b,c,d. We define the overhead ratio as $R_{o,i}$ for image i, where

$$R_{o,i} = \frac{\sum_{j=1}^{m} O_{j,i}}{T_i \times P_i} \tag{6.3}$$

Where $O_{j,i}$ represents the jth overhead area of image i, T_i denotes total interest area of image i and P_i is the number of clusters of image i. One overhead area is selected if and only if it is an overlapping area and it is assigned to more than two process nodes. Thus in Figure6.4, the overhead area is a+b+c+d. Overhead ratio could indicate that whether if it worth to divide the image into more pieces. If the image is divided into n pieces, but yields out a high overhead ratio, then the gain from clustering interest points is impaired.

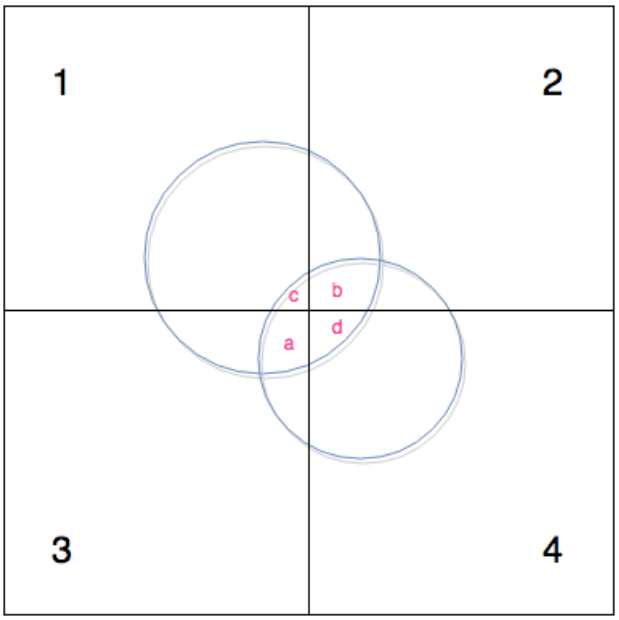


Figure 6.4: Overhead due to image division

6.2.2 Pixels Processing Load Imbalance

Even we distribute the the same number of interest point into each pieces of subareas, the number of pixels of interest area in each of the subarea is still not the same because of the differences of each interest area size. Thus, we consider the *pixels processing load imbalance* Z_i as

$$Z_i = \frac{max_{1 \leq j \leq P} N_{j,i}}{T_i / P} \tag{6.4}$$

Where $N_{j,i}$ stands for the pixels of interest area in subarea j of image i, T_i stands for the pixels that need to be transmitted of image i (total interest area), P represents the number of clusters. Z_i equals 1 indicates the ideal situation that each cluster contains the same needed-pixels, whereas the larger the Z_i reach, the more imbalances in the image processing.

6.2.3 Transmission Imbalance

For number based clustering, the dedicated metric that we would like to consider is Transmission Imbalance. We define the Transmission Imbalance D_i as the ratio that the size of the largest subarea to the average level. S_i is the size of an image i, P is the number of nodes and $W_{j,i}$ is the size of subarea j of image i.

$$D_i = \frac{max_{1 \leq j \leq P} W_{i,j}}{S_i / P} \tag{6.5}$$

6.3 K-means Clustering

Instead of simply dividing the interest points by numbers, there are different advanced clustering algorithms, most of them are designed for application in Big-Data analysis, Data Mining or Bioinformatics [33]. K-means Clustering and Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithms are two widely used clustering algorithms. We will introduce these two clustering algorithms briefly, and implement them in Chapter 7.

6.3.1 Data Clustering

Generally thinking of data clustering is the process that groups data objects, to reach the goal that make the objects in a given group similar to each other in some aspect. There is no accurate definition of clustering, rather, there are many definition of clusters. Depends on the data type, there are one or more definitions

of cluster can be useful to the specific case.
One of the popular definition in image processing is *centroid based* clustering. The main idea behind centroid based clustering is each datasets in a cluster has the closet distance between the center of that cluster. The measurement of the distance varies from different application and clustering algorithm.

6.3.2 The k-means Method

The objective of k-means method is, for all clusters, find a way to minimize the sum of all squared distances in each cluster. The objective function is defined as below:

$$arg \quad minS_i \sum_{i=1}^{k} (\sum_{x_j \in S_i} ||x_j - \mu_i||^2)$$

where x_j represents a data point in the dataset, S_i represents a cluster and μ_i is the center point of cluster S_i
One of the significant feature that k-means method holds is that the number of clusters in k-means method is predefined when clustering occurs. This feature can be both advantage and disadvantage, depending on the situation that the method works at. The positive side is that anomaly data point will not be included in new cluster, instead the anomaly data point will be sorted to some of existed cluster that close to the anomaly data point. This feature is especially suite for our demand, where the number of transmission node are known, thus we can set the k as the node number. However, the drawback is, in some cases it would be unwise to divide the data into certain clusters. For example, imaging we have a dataset that contains three blocks of datapoint that each datapoint in the block has the significant features close to each other. In such case we obviously hope the k-means method treat each block as a cluster so we have three clusters basically. But what if the k is not set to three? In that situation, the performance of k-means method will be poor.
To find a clustering configuration that minimize the objective function above has been proven to be difficult because the worst case time complexity is $2^{\sigma(n)}$ [34]. Thus heuristic algorithms are used in order to minimize the k-means function.

6.3.3 Lloyd's Algorithm

In 1957, Lloyd first proposed a heuristic algorithm to perform k-means clustering, and this algorithm was named in his name [33].
The idea behind Lloyd's algorithm is simple, one should only iteratively improve the position of the cluster centroids. But one should bear in mind that the initial

centroid positions are not generated by Lloyd's algorithm, instead Lloyd's algorithm only take the initial centroid positions as arguments and operating based on that. The initial centroid position will be introduced in Section 6.3.5.
By iterating the flowing two steps, the algorithm can improve the centroid positions.

- step one: $S_i = \{x_j : ||x_j - \mu_j|| \leq ||x_j - \mu_c|| \quad \forall 1 \leq c \leq k\}$
- step two: $\mu_i = \frac{1}{S_i}(\sum_{x_j \in S_i} x_j)$

In step one, the algorithm will perform so called *nearest neighbor search* to assign all data points to one of its closet centroid μ_i.
In step two, the algorithm calculate the mean of all data points in the cluster to update all centroids $\mu_1 \cdots \mu_k$.
The iteration of above two steps stops when some type criteria is met. Common criteria can be when there were no change in the cluster configuration, or the target function remains stable for a certain time because the change was below a certain threshold.

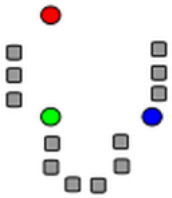
The time complexity is $O(ndk)$ for one iteration of the algorithm [35]. n stands for the number of data points, d represents the dimension of the data and k is the number of cluster centroids.

6.3.4 K-d Tree

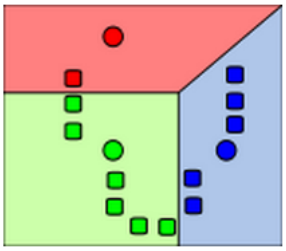
Pelleg and Moore examined a approach to improve Lloyd's algorithm by reducing the needed times to assign each data element to its closest cluster [35]. They use a space partitioning data structure called $k-d\ tree$. A k-d tree is a binary tree that every node consists of a hyperplane defined by a dimension and a point. Space is split by the hyperplane, each of them is represented by children of the node. Tree leaves store the data points.

K-d Center

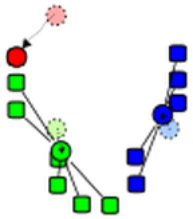
Nearest neighbor search is the necessary process for judging whether a data point belongs to a cluster in Lloyd's algorithm. The time complexity of the nearest neighbor search is nk. However, by building a k-d tree that contains all cluster centroids, it is possible to reduce the complexity of nearest neighbor search to $dk\log^2(k)+n\log(k)$. Because the required time to build a k-d tree is $dk\log^2(k)$ and $\log(k)$ time to find the nearest neighbor in a k-d tree containing k elements.



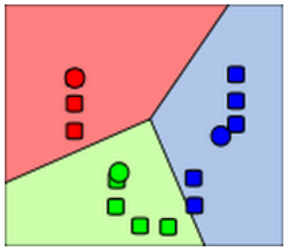
(a) k seeds (k=3) are randomly generated within the dataset



(b) k clusters are created by finding the nearest mean to each seed for each data point.



(c) change the new mean to each of the centroid in the cluster



(d) repeat step 2 and 3 until convergence has been reached

Figure 6.5: Convergence of Lloyd's algorithm.

Blacklisting

A k-d tree optimization is by using blacklisting to reduce the number of nearest neighbor searches. This optimization is based on the fact that several data points may share a common nearest cluster center. The k-d tree is traversed until a subspace with only one closest centroid was found to perform one nearest neighbor search. Thus the cluster center is configured with containing all data points in the subspace. All data points within a leaf is assigned to its closest cluster if a leaf is reached. This is similar to Lloyd's algorithm, the difference is the comparison only has to be performed for each data point within the closest centroid for corresponding leaf.
K-d tree can be used to improve the second step of Lloyd's algorithm since the data in the k-d tree retains between iterations. The update time of μ_i is linear with $|S_i|$ in Lloyd's algorithm. This brings the benefit that the calculation of step two will become linear with the number of subspaces assigned to the cluster instead of the number of data points.

6.3.5 Initial Centroid Position

Initial centroid position (seed) is the base of iteration in Lloyd's algorithm. How to find suitable seeds is critical when performing Lloyd's algorithm. There are several seeding methods, random seeding and partial clustering seeding are relatively efficient seeding methods and will be introduced as below.

Random Seeding

The simplest way of seeding is using random seeding. Random seeding is realized by choosing k number of data points randomly from the dataset as the initial cluster positions. It has been proved to be useful when the dataset is not in the pattern which has significantly special aggregation [36].

Partial Clustering Seeding

Partial clustering seeding is with higher seeding quality compare to random seeding. It seeks for the seeds by performing clustering on a subset of the dataset and use the results of the clustering as seeds for the entire dataset. This method provides more reliable results but impairs the calculation speed, thus partial clustering seeding requires a tradeoff between cluster quality and clustering time.

6.4 DBSCAN Clustering Algorithm

Density-based spatial clustering of applications with noise (DBSCAN) is a data clustering algorithm proposed in 1996 [37]. As its name suggests, DBSCAN is a density-based clustering algorithm. It is realized by grouping data points that are closely packed together, and excluding the points that stand lone in low-density regions. DBSCAN is also one of the most popular clustering algorithms.

6.4.1 Concepts of DBSCAN

DBSCAN defines a set of concepts that helps classifying data points. In the context of DBSCAN, a cluster is an area with high density of points. Noises are the points that lie in a low density area. A data point is considered to be a member of a cluster if and only if for a given distance, there are sufficient data points within the dedicated area. The distances used can be various of distance metrics, depending on the data type.
$Eps(\varepsilon) - neighborhood$:

$$N_{\varepsilon}(p) = \{q | distance(p, q) \leq \varepsilon\}$$

Where $N_{\varepsilon}(p)$ is the set of points q in dataset D that are distant from p by no more than Eps.
A *core point* is a point that its $Eps - neighborhood$ contains at least $MinPts$ points, denoted as $|N_{\varepsilon}(p)| \geq MinPts$.
A $density - reachable$ point is a point that is reachable from p if there is a path$p_1, ..., p_n$ with $p_1 = p$ and $p_n = q$, where each p_{i+1} is directly reachable from p_i.
A $directly\ density - reachable$ point is a point p that from point q with respect to ε and $MinPts$ if $p \in N_{\varepsilon}(q)$ and q is a core point.

Figure 6.6 shows the different types of point in DBSCAN for MinPts=2. Point A and other black points are core points since under radius of ε, there are at least two points around it. B and C are not core points, but they are reachable from core points. Therefore, core points, B and C consist a cluster. Meanwhile, point N is a noise point since it is neither a core point nor density-reachble.

6.4.2 DBSCAN:Implementation

Two important parameters for DBSCAN are ε and $MinPts$ as aforementioned. The algorithm starts with picking an unvisited starting point randomly, and retrieve the $Eps - neighborhood$ of this point. If there are sufficient points in it, a cluster is started. Otherwise, the point is marked as noise point. It is noticeable that this point is temporarily marked as nose unless the iteration is finished, that

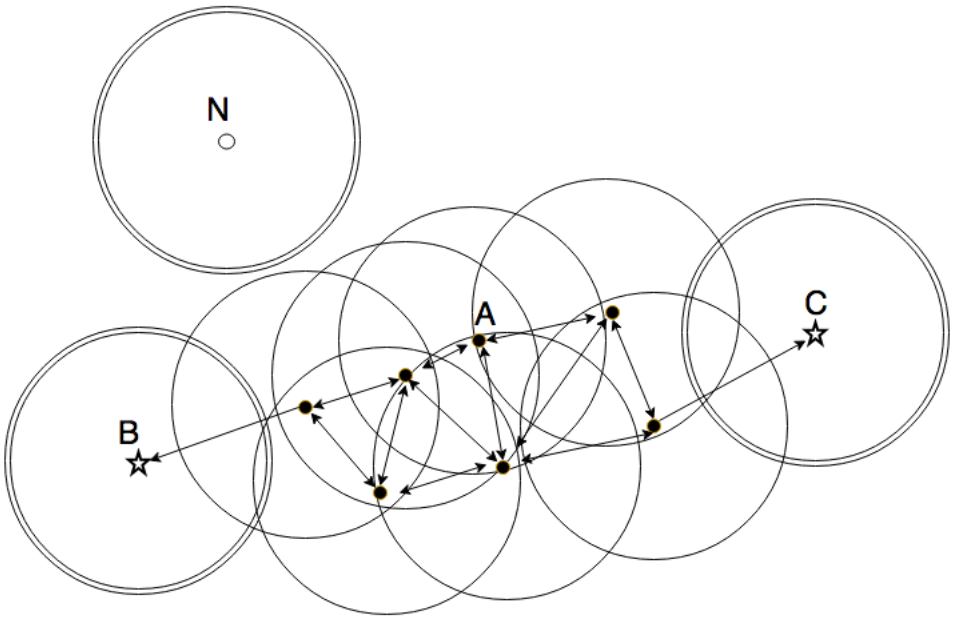


Figure 6.6: Different Types of Point in DBSCAN

is, the noise point still can be made part of a cluster if it can be found in a sufficiently sized ε of a different point.
The iteration happens by checking each point in dataset, if a point is in a dense part of a cluster, DBSCAN also marks its $Eps - neighborhood$ as part of the cluster. Thus, all points in the $Eps - neighborhood$ will be added. The iteration of finding one cluster stops until the density-connected cluster is completely found. After that, DBSCAN will choose a new unvisited point, and continues to find further cluster or noise.
Because DBSCAN needs to visit each datapoint several times, the complexity of DBSCAN is usually $O(n^2)$.

Chapter 7

Performance Evaluation of Clustering Gain

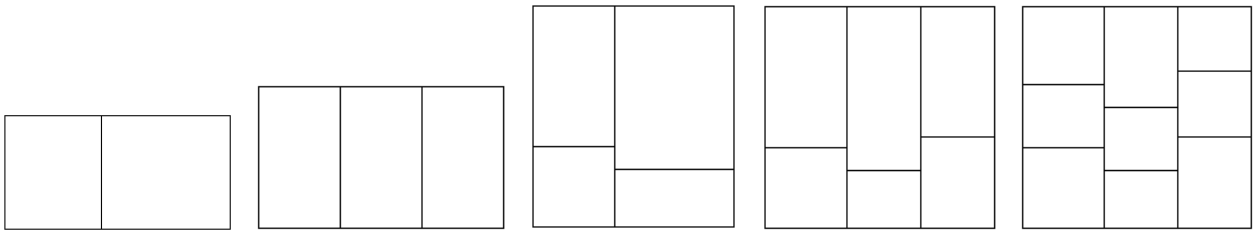
In this chapter, we will implement the proposed clustering algorithms, and give out the performance evaluation for each algorithm.

7.1 Number-Based Clustering

The number-based clustering, as we mentioned in chapter 6, of which we distribute the interest points to each processing node equally, is the first method to cluster the interest points for processing nodes. By doing this, we first select the top 1000 interest points according to their response value, and divide the original image into 2,3,4,6,9 pieces:

- Two clusters: one vertical cut so that the number of interest points are the same on both sides.
- Three clusters: two vertical cuts.
- Four clusters: one vertical cut, and then a horizontal cut in each of the clusters.
- Six clusters: two vertical cuts, and then each cut horizontally.
- Nine clusters: two vertical cuts, and each sliced pieces with two horizontal cuts.

As Figure 7.1 shows.



(a) two clusters (b) three clusters (c) four clusters (d) six clusters (e) nine clusters

Figure 7.1: Different ways of cutting image

We processed the image set through out 1161 images for each of the cutting schemes we proposed, and made the statistic evaluation according to evaluation criteria in terms of *Overhead Ratio*,*Pixel Processing Load Imbalance* and *Transmission Imbalance* in section 6.2.1, 6.2.2 and 6.2.3.

7.1.1 Number Based Clustering-Overhead Ratio

By implementing equation 6.3 for 1161 images and for five different cutting schemes, we obtained the overhead ratio results, as Figure 7.2 shows. As we divide the im-

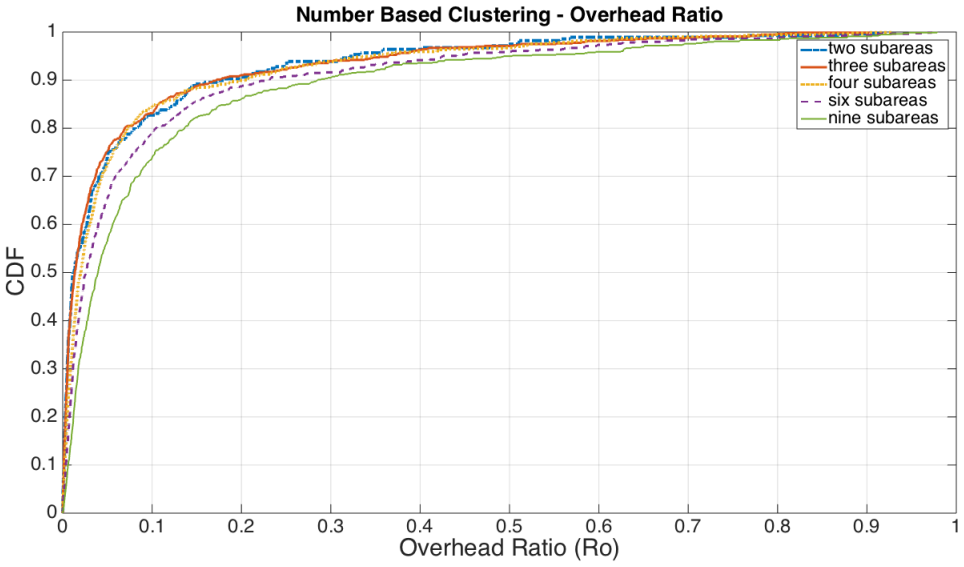


Figure 7.2: Number Based Clustering-Overhead Ratio of Different Division Schemes

age into more clusters, the overhead ratio increases accordingly. This is as what

we have expected, it is reasonable that if one image is divided into more pieces, the border lines will increase thus there will be more chance to produce the overhead area. And another noticeable thing is though the overhead ratio is increasing as we cut the image into more pieces, the amount of overhead pixels is still relatively small, mean value of overhead ratio for nine-clusters is less than 0.1.

7.1.2 Number Based Clustering-Pixel Processing Load Imbalance

As we have analyzed in section 6.2.2, for number based clustering, we distribute the interest point to each cluster equally, but this still lead to pixel processing load imbalance because the size of each interest area can vary. The pixel processing load imbalance in terms of the real pixels that been assigned to the processing node is defined in equation 6.4, the static results from processing 1161 images for 5 different cutting schemes is shown in Figure 7.3.

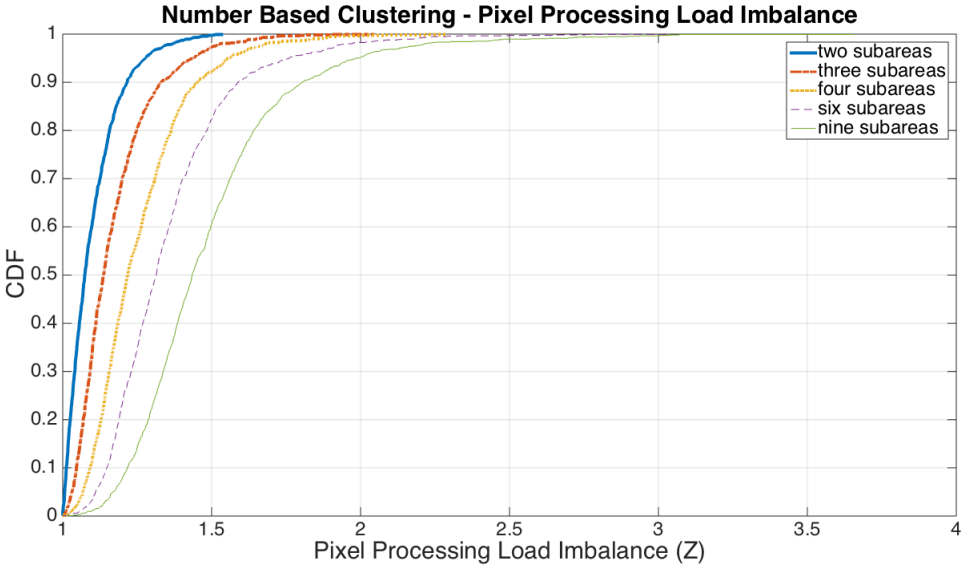


Figure 7.3: Number Based Clustering-Pixel Processing Load Imbalance of Different Division Schemes

The results show us that even if each processing node is assigned to process the same number of interest points, the actual processing load is still not balanced perfectly. Specifically, as the image is divided into more parts, the pixel processing load imbalance goes high. For the worst situation, e.g, the images were cut into 9 pieces, top 5% of the pixel processing load imbalance ratio Z are over 2, indicating processing node that with heaviest processing load need to process twice as the

average processing load.
However, it is noticeable that even the worst case can lead to twice pixel processing load imbalance, the average pixel processing load imbalance is still less than 1.5. For the 5% of the lowest cases, the pixel processing load imbalance is close to 1.

7.1.3 Number Based Clustering-Transmission Imbalance

For number based clustering, we were also interested in considering transmission imbalance since we perform number based clustering by cutting the image into rectangular areas. As defined in equation 6.5, the transmission imbalance can be indicated by counting transmission imbalance ratio D_i, which is the ratio that reveals the relation from the size of largest sub image to the size of average sub images. The result is shown in Figure 7.4. As we cut the image into more pieces,

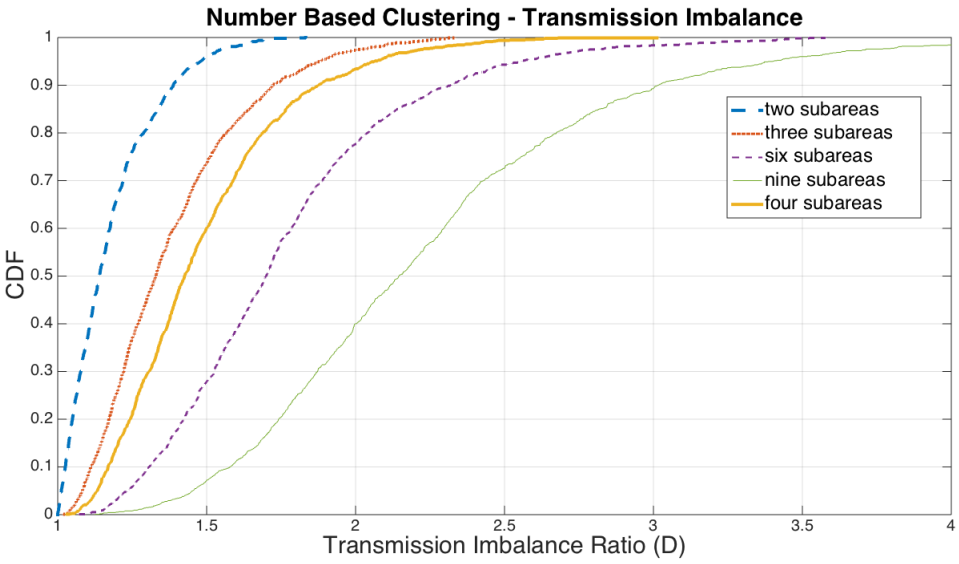


Figure 7.4: Number Based Image Division Imbalance Ratio

the image division imbalance ratio will increase, and the mean value range of different cut schemes is between 1.2 to 2.3.

7.2 K-means Clustering

We followed the k-means clustering algorithm introduced in Section 6.3. In our implementation, we utilized basic Lloyd's algorithm as the iteration algorithm, and combined it by using k-d tree. For the initial seeds selection, we adopted

the probabilistic means of initialization for k-means clustering proposed by Arthur and Vassilvitskii in 2007 [40]. The iteration termination criteria we chose was by utilizing the in-built stop criteria in OpenCV, that is, either by stoping the algorithm iteration if specified accuracy is reached, or by stoping the algorithm iteration if specified number of iterations are reached. We set the accuracy and iteration attempts to 1.0 and 10 respectively. To be consistent with the number based clustering, we assumed the number of processing node were the same as the situation in number based clustering and each processing node take a cluster of interest points as the processing load. Hence we set the number of clusters to 2,3,4,6,9 accordingly.

7.2.1 K-means Clustering-Overhead Ratio

We first consider the k-means clustering overhead ratio. We set the k value of the algorithm to 2,3,4,6,9 respectively according to the assumption aforementioned. Figure 7.5 shows the results. We can easily see the mean value of the overhead ratio decreased almost an order of magnitude compares to number based clustering. This is intuitive because k-means clustering algorithm can divide the interest points into more sophisticated shape, thus reduce the possibility that the edge of clusters intersecting with each other.
However, it is also noticeable that in Figure 7.5, the lowest overhead ratio comes from k=6 instead of k=2, suggesting that six clusters might more fit for clustering the interest points in images from our image dataset compared to other k values.

7.2.2 K-means Clustering-Interest Point Imbalance

As we set the k value of k-means clustering algorithm to 2,3,4,6,9, we obtained k-means interest point imbalance as Figure 7.6 shows. From the result, we can see that as we divide the interest points into more clusters, the interest point imbalance increases accordingly. Especially, when we set the k to 9, the interest point imbalance goes almost twice as much as six clusters and the average L reach to 7. This indicates that when we divide the interest points of an image into 9 clusters using k-means clustering, the processing node which is assigned with most of interest points will take seven times workload among average value.

7.2.3 K-means Clustering-Pixel Processing Load Imbalance

The same as we did in Interest Point Imbalance, we still divide interest points into 2,3,4,6,9 clusters. Figure 7.7 shows the results. The results compared to the number based pixel processing load imbalance are almost twice as much as it. Suggesting k-means clustering would bring almost twice pixel processing load

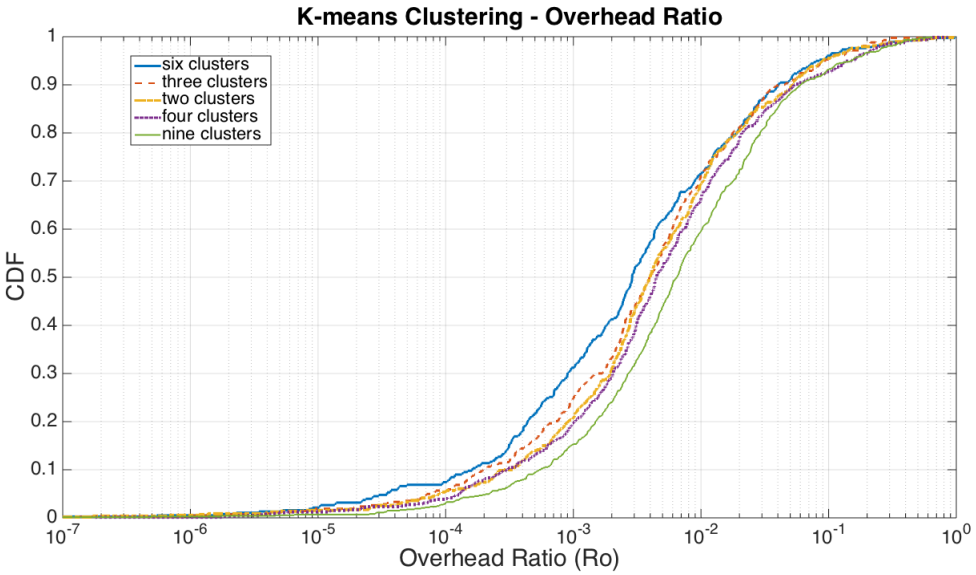


Figure 7.5: K-means Overhead Ratio

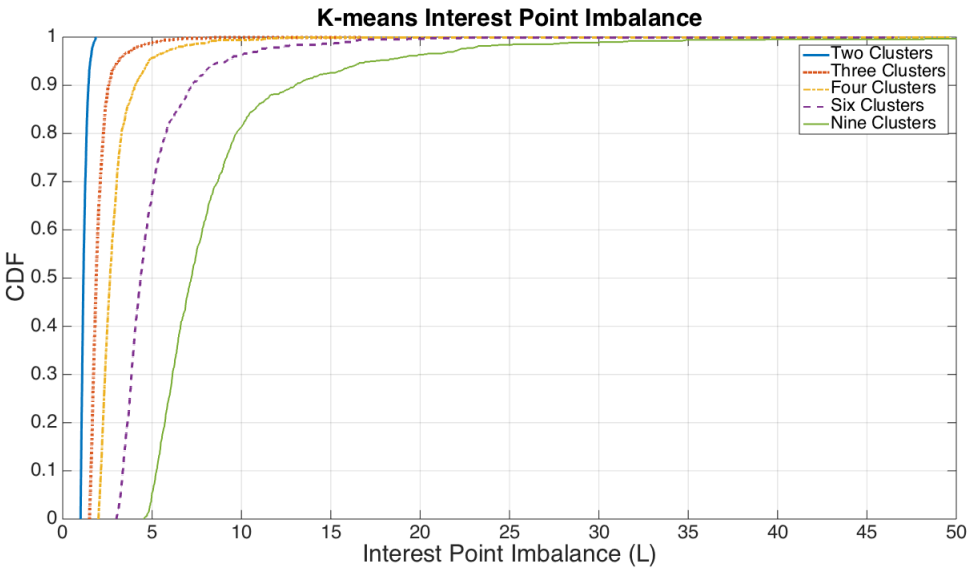


Figure 7.6: K-means interest point imbalance

imbalance over number based clustering.
It is also noticeable that compared to k-means interest point imbalance, the k-means pixel processing load imbalance is lower. This is due to characteristic of the interest point in an image. BRISK will detect a batch of dense but with small

radius interest points to describe a clear blob while it will use less density but with larger radius interest points to describe the relatively blurry blob. Hence this will lead to a cluster with dense interest points has smaller interest area compared to the cluster with less interest points.

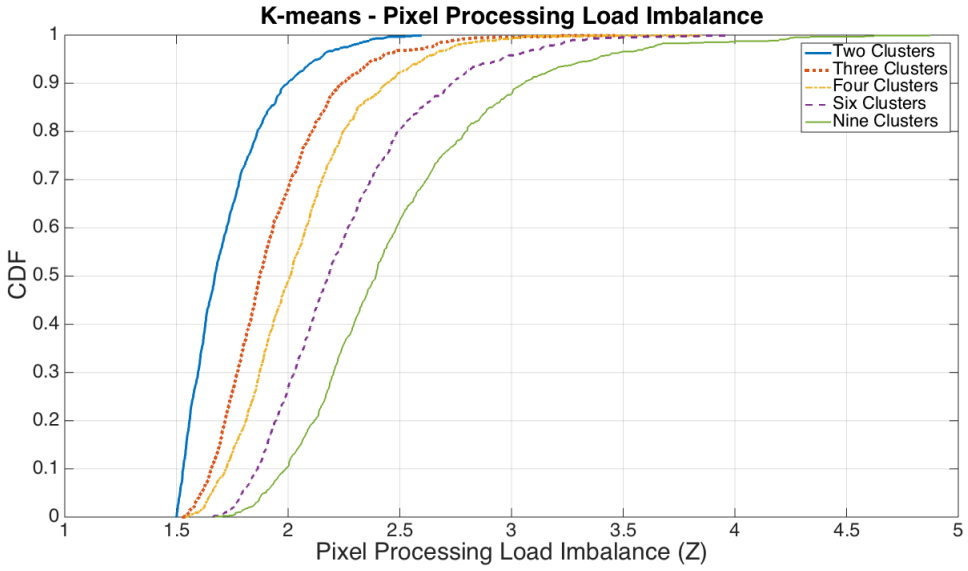


Figure 7.7: K-means Pixel Processing Load Imbalance

7.3 DBSCAN Clustering

We implemented the DBSCAN clustering algorithm based on the description in Section 6.4. Since DBSCAN clusters interest point into uncertain number of clusters, to be consistent with the evaluation of number based clustering and k-means clustering, we adjusted the parameters of ε and $MinPts$ to 100 and 10 respectively, to yield a reasonable range of clusters number of images in the image dataset to be constraint from 2 to 10.

7.3.1 DBSCAN Overhead Ratio

Figure 7.8 shows the results of overhead ratio of DBSCAN. The results we obtained is very intuitionistic. The DBSCAN reduced the overhead ratio more efficiently, compares to k-means and number based clustering. There is over 90 percent chance for DBSCAN to cluster interest points with overhead ratio less than 10^{-3}, and over 75 percent chance to cluster without any overhead area.

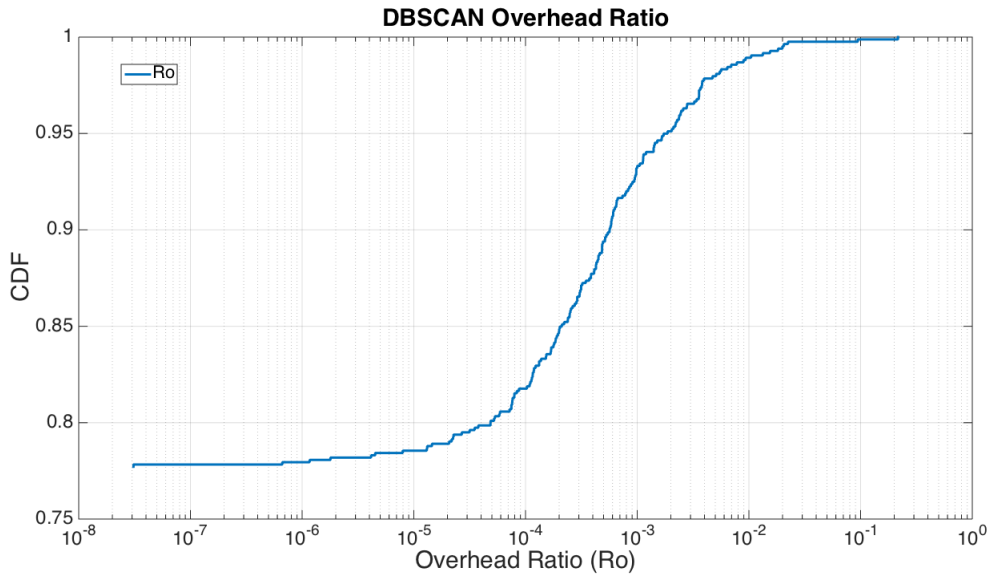


Figure 7.8: DBSCAN Overhead Ratio

7.3.2 DBSCAN Interest Point Imbalance

After calculation, we obtained the results of DBSCAN interest point imbalance as Figure 7.9 shows. The mean value of interest point imbalance is higher than the most unbalanced case in k-means clustering. This suggests that though DBSCAN can provide better performance in overhead ratio, it can not balance the interest points among each clusters nicely. That is, some of the clusters will contain lots of interest point while others only contain a few of interest point.

7.3.3 DBSCAN Pixel Processing Load Imbalance

The results of pixel processing load imbalance of DBSCAN are shown in Figure 7.10. The DBSCAN pixel processing load imbalance is almost the same as DBSCAN interest point imbalance, while we have already obtained that in k-means clustering, the pixel processing load imbalance is lower than interest point imbalance, because of the existence of some less dense points with larger radius.
To explain this, we looked into the details of DBSCAN algorithm. DBSCAN excludes the noise points but k-means algorithm does not. Therefore, the less dense interest points with large radius are not likely to be included in any clusters in DBSCAN, thus the pixel processing load imbalance is almost the same as the interest point imbalance for DBSCAN.

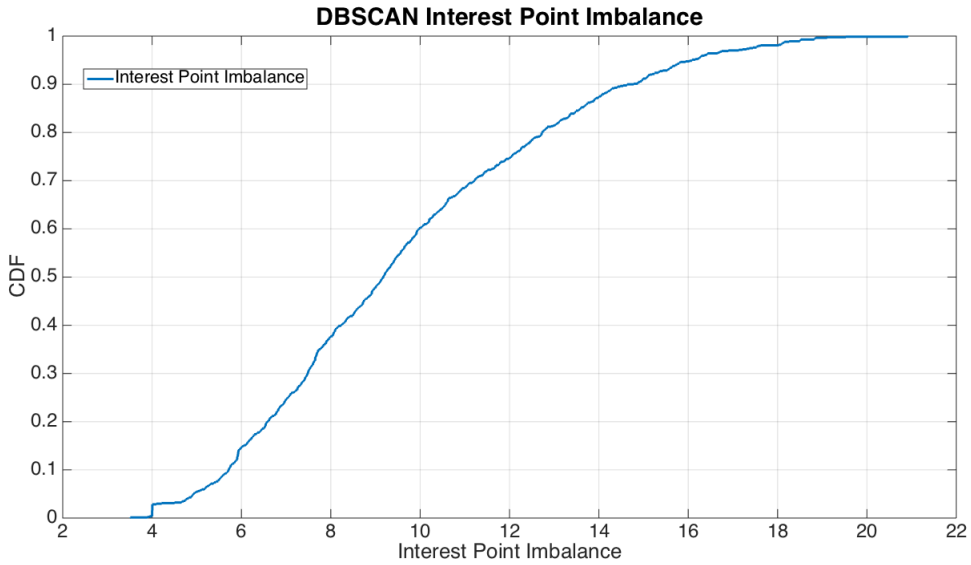


Figure 7.9: DBSCAN Interest Point Imbalance

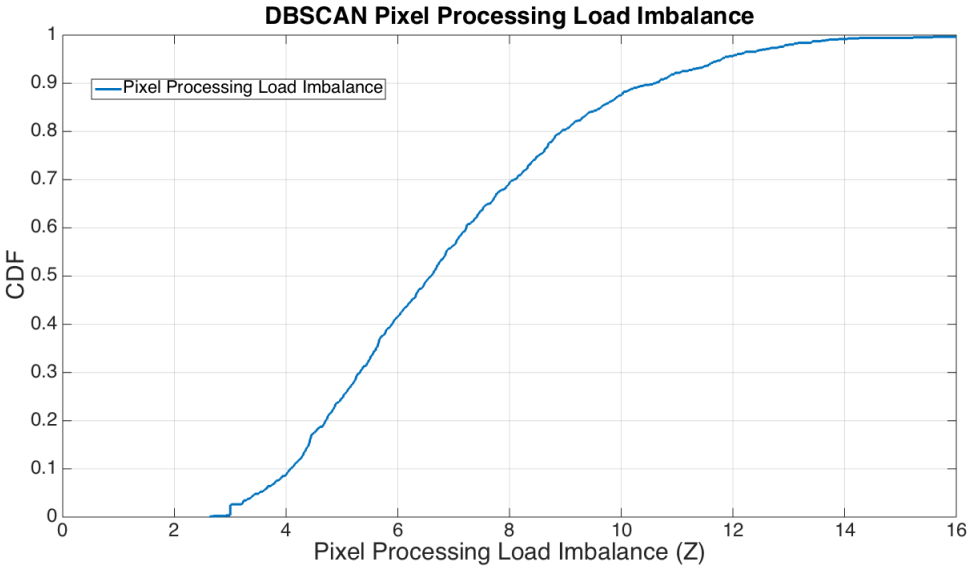


Figure 7.10: DBSCAN Pixel Processing Load Imbalance

Chapter 8

Conclusion and Future Work

In this master thesis project, we explored the critical challenge of visual sensor networks, that is, both high bandwidth and rich computational resources are required. In consequence, we were considering to extract the interest points on the camera side, and transmits only the extracted interest points to the server. After the calculation, we found the pixels of interest points in the image dataset is only a quarter on average. This suggests that to extract the interest points on the camera side can decrease the data flow in the visual sensor networks

To do so, we first extracted the interest points of an image by implementing BRISK detector, and a step further, we implemented different clustering algorithms (Number Based Clustering Algorithm, K-Means Clustering Algorithm, DBSCAN Algorithm) on detected interest points, we clustered the interest points into different clusters, and examined the characteristic of different clustering algorithms with respect to Overhead Ratio, Interest Point Imbalance and Pixel Processing Load Imbalance.

From the results, we found that none of the proposed algorithm is overpowering than others. Indeed, each of them has their own shining spot. Number Based Clustering can deliver the interest points to the fixed number of processing nodes nicely, because it was designed to assign interest points to the processing nodes equally. But Number Based Clustering will lead to highest overhead ratio compares to other algorithms. Thus Number Based Clustering suits for the scenario that requires the data flow to be balanced restrictedly, but will need to consider a high level overhead ratio as tradeoff.

DBSCAN algorithm, on the contrary, yields the highest interest point imbalance, due to its characteristic that it can not pre-set the number of clusters and DBSCAN can cluster the interest points into convex or concave shape. This makes DBSCAN not suit for the scenario that requires high performance on data-flow balancing. However, DBSCAN performs extremely good in decreasing the overhead ratio, that is, DBSCAN can cluster the interest points as much as avoiding

causing of overhead area. This makes the data transmission very effective, and thus suits for the scenario that the sources are limited.
K-Means Clustering Algorithm, comprises the strong and weak of other two algorithms. K-Means Clustering can pre-set the number of clusters, and thus have the middle performance regarding the interest point imbalance. And K-Means Clustering has a better performance in decreasing the overhead ratio compares to Number Based Clustering.
In our work, we calibrated the Eps and $MinPts$ carefully for DBSCAN to constraints the number of clusters that yields by DBSCAN been limited in the range from 3-10.
To sum up, we found extraction of interest points in a image can reduce the data that needed to be transmitted in the network a quarter on average. The clustering methods we proposed can help improving the performances with respect to overhead ratio, interest point imbalance and pixel processing load imbalance. But the clustering methods also brings extra computational consumptions on the camera side. We did not seek for whether there is any trade off between computational resources consumption due to the time limitation. Future work can be considering to assign several clusters to a processing node instead of one cluster for one processing node, thus to be consistent with the results of Number Based Clustering and K-means Clustering that can preset the number of clusters. Possibilities also could be searching for the trade off in between the computational resources consumption and transmission gain due to the interest points clustering.

Bibliography

[1] *A survey of visual sensor network.* Soro S, Heinzelman W (2009). Adv Multimed 2009, Article ID 640386

[2] *Visual Analysis of Social Media Data* Schreck, T. Keim, D. Computer (Volume:46 , Issue: 5) ,

[3] *Mobile visual search.* Girod, et al., IEEE Signal Processing Magazine, vol.28, no.4, pp.61,76, July 2011

[4] *Characterization of SURF and BRISK interest point distribution for visual processing in sensor networks* Dan, Khan, Fodor, submitted, confidential

[5] *Video Google: A text retrieval approach to object matching in videos* J. Sivic and A. Zisserman, in Proc. IEEE Int. Conf. Computer Vision (ICCV), Washing- ton, DC, 2003.

[6] *Scalable recognition with a vocabulary tree* D. Nist and H. Stewius, in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), New York, June 2006.

[7] *Hamming embedding and weak geometric consistency for large scale image search* H. Jegou, M. Douze, and C. Schmid, in Proc. European Conf. Computer Vision (ECCV), Berlin, Heidelberg, 2008.

[8] *Lost in quantization?Improving particular object retrieval in large scale image databases* J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), Anchorage, AK, June 2008.

[9] *A combined corner and edge detector* C. Harris, M. Stephens, in: ACCV, issue 1, 2006, pp. 918-927.

[10] *Feature detection with automatic scale selection* T. Lindeberg, IJCV 30 (2) (1998) 79-116.

[11] *Indexing based on scale invariant interest points* K. Mikolajczyk, C. Schmid, in: ICCV, vol. 1, 2001, pp. 525-531.

[12] *Object recognition from local scale-invariant features* D. Lowe, in: ICCV, 1999.

[13] *Scale, saliency and image description* T. Kadir, M. Brady, IJCV 45 (2) (2001) 83-105.

[14] *Scale-invariant shape features for recognition of object categories* F. Jurie, C. Schmid,in: CVPR, vol. II, 2004, pp. 90-96.

[15] *Scale and affine invariant interest point detectors* K. Mikolajczyk, C. Schmid, IJCV 60 (1) (2004) 63-86.

[16] *A performance evaluation of local descriptors* K. Mikolajczyk, C. Schmid, PAMI 27 (10) (2005) 1615-1630.

[17] *General intensity transformations and differential invariants* L.M.J. Florack, B.M. ter Haar Romeny, J.J. Koenderink, M.AViergever, JMIV 4 (2) (1994) 171-187.

[18] *Moment invariants for recognition under changing viewpoint and illumination* F. Mindru, T. Tuytelaars, L. Van Gool, T. Moons, CVIU 94 (1-3) (2004) 3-27.

[19] *Reliable feature matching across widely separated views* A. Baumberg, in: CVPR, 2000, pp. 774-781.

[20] *The design and use of steerable filters* W.T. Freeman, E.H. Adelson, PAMI 13 (9) (1991) 891-906.

[21] *Multi-scale phase-based local features* G. Carneiro, A.D. Jepson, in: CVPR, issue 1, 2003, pp. 736-743.

[22] *Object recognition from local scale-invariant features* D. Lowe, in: ICCV, 1999.

[23] *A performance evaluation of local descriptors* K. Mikolajczyk, C. Schmid, in: CVPR, vol. 2, June 2003, pp. 257-263.

[24] *PCA-SIFT: a more distinctive representation for local image descriptors* Y. Ke, R. Sukthankar, in: CVPR, issue 2, 2004, pp. 506-513.

[25] *Vision based modeling and localization for planetary exploration rovers* S. Se, H.K. Ng, P. Jasiobedzki, T.J. Moyung, in: Proceedings of International Astronautical Congress, 2004.

[26] *Fast approximated SIFT* M. Grabner, H. Grabner, H. Bischof, in: ACCV, issue 1, 2006, pp. 918-927.

[27] *Compress-then-analyze vs. analyze-then-compress: Two paradigms for image analysis* Alessandro Redondi, Luca Baroffio, Matteo Cesana, Marco Tagliasacchi. Dipartimento di Elettronica, Informazione e Bioingegneria P.zza Leonardo da Vinci, 32 - 20133 Milano (Italy)

[28] *Practical Distributed Processing* Phillip J. Brooke MA, DPhil, MBCS CITP, Richard F. Paige BSc, MSc, PhD .ISBN: 978-1-84628-840-1

[29] *Adaptive and generic corner detection based on the accelerated segment test.* E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger. In Proceedings of the Eu- ropean Conference on Computer Vision (ECCV), 2010. 2, 5

[30] *Daisy: an Efficient Dense Descriptor Applied to Wide Baseline Stereo.* E. Tola, V. Lepetit, and P. Fua. IEEE Transac- tions on Pattern Analysis and Machine Intelligence (PAMI), 32(5):815?830, 2010. 4

[31] *BRISK: Binary Robust Invariant Scalable Keypoints* Stefan Leutenegger, Margarita Chli and Roland Y. Siegwart Autonomous Systems Lab, ETH Zurich

[32] *Itseez leads the development of the renowned computer vision library OpenCV.* http://itseez.com/

[33] *Data Clustering: 50 Years Beyond K-Means.* Anil K. Jain, Michigan State University,Michigan

[34] *k-means Requires Exponentially Many Iterations Even in the Plane.* Andrea Vattani, University of California, San Diego

[35] *Accelerating Exact k-means Algorithms with Geometric Reasoning.* Dan Pelleg, Andrew Moore, Carnegie Mellon University, Pittsburgh

[36] *Refining Initial Points for K-Means Clustering.* P.S Bradley, Usama M. Fayyad. University of Wisconsin, Wisconsin

[37] *A density-based algorithm for discovering clusters in large spatial databases with noise.* Ester, Martin; Kriegel, Hans-Peter; Sander, J?rg; Xu, Xiaowei (1996). Simoudis, Evangelos; Han, Jiawei; Fayyad, Usama M., eds. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96). AAAI Press. pp. 226?231. ISBN 1-57735-004-9. CiteSeerX: 10.1.1.71.1980

[38] *k-means++: The advantages of careful seeding* Arthur D, Vassilvitskii S. Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, 2007: 1027-1035.