Daria A. Burminova

COOPERATIVE IMAGE ANALYSIS IN VISUAL SENSOR NETWORKS BASED ON NASH BARGAINING SOLUTION

Master of Science Thesis

Politecnico di Milano
October 2014

POLITECNICO DI MILANO
Electronics, Information and Bioengineering Department

Master of Science in TELECOMMUNICATION ENGINEERING
Major in COMMUNICATION NETWORKS

COOPERATIVE IMAGE ANALYSIS IN VISUAL SENSOR NETWORKS BASED ON NASH BARGAINING SOLUTION

Daria A. Burminova

Advisor: Engr. Matteo CESANA .....
Co-advisor: Dr. Alessandro REDONDI .....

Candidate: Daria A. BURMINOVA .....
Student ID number 796449

Academic Year 2013/14

Abstract

One of the most dynamically evolving areas in Telecommunications is the Internet of Things. This concept describes "a world-wide network of uniquely addressable and interconnected objects, based on standard communication" (ETP-EPOSS). The Internet of Things can find application practically everywhere starting from a wearable device monitoring steps to a space shuttle coffee machine.
This thesis research considers visual sensor networks (VSN), one of applications of the Internet of Things. VSNs enable a smart surveillance system that can improve areas of early emergency alarming, safety and health-state monitoring. Taking into consideration battery-limited lifetime of sensor network the topic of energy saving is of actual importance. This thesis work defines an outline, where a visual sensor network is densely populated with camera nodes, in case of surveillance of public areas application. In this reference scenario some of the cameras have an overlapped field of view. Furthermore each camera performs visual analysis task: acquire image, extract features, transmit compressed features over the network. Therefore camera nodes acquire, process and spend limited energy resources on the same (or partially the same) image. This work proposes cooperative framework in which camera nodes with (partially) overlapped field of view collaborate to reduce the overall energy consumption. Camera nodes collaborate upon a portion of image they have to process, playing a bargaining game. The optimal solution to this game is offered by the Nash bargaining solution (NBS). NBS implementation allows to obtain "efficient" image processing sharing policies.

Chapter 1

Introduction

In recent years extensive attention worldwide has been given to the Internet of Things. The key technologies include sensor networks, RFID technology, thread detection and nano-technology [1]. These technologies certainly impact industrial development, scientific and technological progress and the Internet of Things also helps facilitate human life.
Sensor networks, as a bright example of the Internet of Things concept, are dense wireless networks of small, low-cost sensors which collect and disseminate environmental data. Wireless sensor networks facilitate monitoring and controlling of physical environments from remote locations. They have applications in a variety of fields such as environmental monitoring and military purposes, gathering and sensing information in hostile locations. Visual wireless sensor networks include visual sensor nodes called camera nodes, which integrate the image sensor, the embedded processor and wireless transceiver. Visual sensor networks also find application in health and medicine, for example, for remote monitoring, diagnostics and additional support for disabled or monitoring movements and internal processes of small animals. Surveillance is a primary area for camera-based networks. They monitor large public areas using hundreds or thousands of camera nodes. Environmental application includes monitoring of remote areas over a long period of time, such as flood or fire detection. Also visual wireless sensor networks provide the scientists with tools for large scale earth monitoring and planetary explo-

ration. Whereas telepresence systems enable a remote user to "visit" some location that is monitored by collection of cameras, such as a virtual museum or gallery.

Every sensor network consists of a number of nodes. Each node has various energy and computational constraints due to its limited cost and size. If sensor networks include also camera nodes, then image-related operations imply larger energy consumption due to local visual analysis and data transmission. Each visual wireless sensor network can be characterized by a number of parameters: real-time performance; location and orientation information; local processing, that may include simple image subtraction or more complex object classification or scene reasoning. A visual sensor network is also characterized by autonomous node collaboration, implying information exchange and autonomous decision making.

The reference scenario of this thesis addresses a visual sensor network for surveillance of public areas, where camera nodes are densely located. In such a setting, some of the camera nodes most likely have an overlapped field of view among them, implicating that camera nodes in fact observe the same area and process the same (or partially the same) images. The reference surveillance application is based on visual analysis task such as object recognition or image retrieval in which camera nodes are required to acquire images, extract local features from the acquired images, and send a compressed version of such features across the network for a further processing. Keeping in mind that any sensor network is energy constrained, every time camera nodes process the same image it leads to unnecessary energy consumption and consequently shorter network lifetime.

To this extent, this work proposes cooperative framework in which camera nodes with (partially) overlapped field of view collaborate to reduce the overall energy consumption. In a nutshell, the collaborating cameras agree on which part of acquired image they have to process, thus reducing per-camera energy consumption, and, as a consequence, extending the network lifetime. The image sharing problem is represented as a bargaining game in which the players (camera nodes) bargain upon a portion of image they have to process. The solution to this game, that is Nash bargaining solution is then

used to obtain "efficient" image processing sharing policies. Research and analysis is conducted to understand the feasibility of the suggested solution, comparing it with alternative data sharing approaches. And finally a communication protocol is offered with the purpose of enabling NBS-based image sharing collaboration among nodes. This thesis has been developed within the framework of The GreenEyes European project, which has the final goal of empowering wireless sensor networks with vision capabilities [2].
First of all in Chapter 2 reviews visual wireless sensor networks, their application and characteristics description. Visual analysis is also discussed. It is an important component of NBS implementation. In Chapter 3 theoretical principles of the Nash bargaining solution concept are described, as well as a detailed account of its plausible interpretations, parameters and applications. Chapter 4 is dedicated to the problem statement, reference scenario and utility function experimental evaluation. The following Chapter 5 contains main components required for performance evaluation of NBS-driven algorithm, structure of analysis and obtained outcomes also take place in Chapter 5. Chapter 6 describes the communication protocol, its components and required energy budget. Chapter 7 concludes the thesis and proposes possible directions of future inquiry and development.

Chapter 2

Visual Sensor Networks

2.1 Introduction

This chapter reviews wireless visual sensor network's essential properties, functionalities, along with feature extraction algorithms that enable image analysis.

A concept of sensor network is usually interpreted as a network of spatially distributed autonomous sensors with a communications infrastructure allowed to monitor and record conditions, such as temperature, sound, pressure. Whereas visual sensor networks consist of tiny visual sensor nodes called camera nodes, which integrate the image sensor, embedded processor, and wireless transceiver [3]. Following the trends in low-power processing, wireless networking, and distributed sensing, visual sensor networks have been developed as a new technology with a number of potential applications, ranging from security and surveillance to environmental monitoring and telepresence. For example, the physiological data (heart rate, pulse or physical activity) about a patient can be monitored remotely by a doctor. It brings more convenience to a patient meanwhile remote management of the data becomes more opportune for a doctor. It is envisioned that in a future sensor networks will become an integral part of everyday life. Consisting of a large number of low-power camera nodes, visual sensor networks sup-

port a great number of novel vision-based applications. The camera nodes provide information from a monitored site and may perform distributed and collaborative processing of the collected data. On one hand the large amount of image data produced by the cameras leads to elevated process reliability, on the other hand the network's energy and transmission resources are constrained. Therefore a visual sensor network implementation requires exploring new means for data processing, communication and sensor management. Meeting these challenges of visual sensor networks an interdisciplinary approach, such as Nash bargaining solution concept is required. NBS enables more intelligent way of image processing.

2.1.1 Visual sensor networks applications

Sensor networks coupled with different types of sensors (such as magnetic, infrared, visual, radar etc.) performs monitoring of a great variety of environmental conditions, including the following [4]:

- vehicle movement
- lightning condition
- soil makeup
- presence or absence of certain type of objects
- object characterizations, such as speed or size

The alliance of micro-sensing, visual component and wireless connection of those nodes introduces new application areas.

Surveillance Surveillance is a primary area for camera-based network. Monitoring of large public areas (such as airports, subways, etc.) is performed by hundreds or thousands of security camera nodes [3]. Current efforts in visual sensor networking are concentrated toward advancing the existing surveillance technology by utilizing intelligent methods for extracting

information from image data locally on the camera node, thereby reducing the amount of data traffic. Visual sensor networks can be seen as a next generation of surveillance systems that are not limited by the absence of infrastructure, nor do they require large processing resources at one central server.

Interactive surveillance using multiple cameras Cameras could be used for social security and cooperative surveillance to ensure the security of entire community, for example, by tracking suspects over a wide area, using the cooperation of multiple cameras. Using interactive surveillance the traffic police can discover, track and catch vehicles involved in traffic offences [5].

Anomaly detection and analysis In some cases it is necessary to determine normal or abnormal behaviour of people or vehicles. For example, interactive surveillance placed in parking lots and supermarkets could analyse abnormal behaviour indicating theft. And automatic notification may follow either directly to police department or as a recorded public announcement.

Crowd flux statistics and congestion analysis Visual surveillance systems can automatically compute the flux of people at important public areas such as stores and travel sites, and analyse congestion in the area and accordingly assist with more flexible management of people flow.

Environmental application Can be used for monitoring remote and inaccessible areas over a long period of time. It is usually combined with other types of sensor networks, in that way camera activity can be triggered only when a certain event is detected. Some environmental applications of sensor networks include tracking of small animals, birds; macro-instruments for large-scale earth monitoring and planetary exploration; environmental monitoring in marine, soil and atmospheric context; forest fire or flood detection and studies of pollution.

Forest fire detection Sensor network nodes can be densely deployed in a forest, such that it is possible to locate exact origin of the fire and notify the end user and to prevent damaging fire spreading. They can be equipped with effective energy saving mechanisms and technologies (such as solar cells) considering that they can be left unattended for months or years.

Flood detection An example of a flood detection is the ALERT system deployed in the United States [6]. The sensors that measure water level and weather conditions supply information to the centralized database in a predefined way.

Health applications Some sensor networks find use as an interface for integrated patient monitoring, drug administration in hospitals, monitoring movements and internal processes of small animals, tracking doctors inside the hospital.

Telepresence systems Telepresence systems enable a remote user to "visit" some location that is monitored by a collection of cameras, for example, museums, galleries or exhibition rooms. The system is able to provide the user with any current view from any viewing point, and thus it provides the sense of being physically present at a remote location through interaction with the system's interface.

Commercial applications Some of commercial applications are monitoring of product quality, constructing smart office spaces, environmental control in office buildings, robot control and guidance in automatic manufacturing environments, factory process control and automation, monitoring disaster area, smart structures with sensor nodes embedded inside, detecting and monitoring of car thefts.

Detecting and monitoring car thefts Sensor nodes are being deployed to detect and identify threats within a geographical region and report these threats to remote end user via Internet for analysis.

2.1.2 Characteristics of Visual Sensor Networks

One of the main differences between visual sensor networks and other types of sensor networks lies in the nature of how the image sensors perceive information from the environment. Most sensors provide measurements as 1D data signals. In addition, a camera's sensing model is inherently different from the sensing model of any other type of sensor. Typically, a sensor collects data from its vicinity, as determined by its sensing range. Cameras, on the other hand, are characterized by a directional sensing model - cameras capture images of distant objects/scenes from a certain direction. The 2D sensing range of traditional sensor nodes is, in the case of cameras, replaced by a 3D viewing volume (called field of view, or FoV). Visual sensor networks are in many ways unique and more challenging compared to other types of wireless sensor networks.

Resource Requirements

The lifetime of battery-operated camera nodes is limited by their energy consumption, which is proportional to the energy required for sensing, processing, and transmitting the data. Given the large amount of data generated by the camera nodes, both processing and transmitting image data are quite costly in terms of energy, much more than for other types of sensor networks. Furthermore, visual sensor networks require large bandwidth for transmitting image data. Nash bargaining solution suggests a way of reducing image data processing cost, therefore meet intended reduction of energy consumption.

Real-Time Performance

Most applications of visual sensor networks require real-time data from the camera nodes, which imposes strict boundaries on maximum allowable delays of data from the cameras. The real-time performance of a visual sensor network is affected by the time required for image data processing and for the transmission of the processed data throughout the network.

Location and Orientation Information

In visual sensor networks, most of the image processing algorithms require information about the locations of the camera nodes as well as information about the cameras' orientations. This information can be obtained through a camera calibration process, which retrieves information on the cameras' intrinsic description, such as angle of view, focus length and SSD chip size and extrinsic parameters, such as geometrical location.

Local Processing

Local (on-board) processing of the image data reduces the total amount of data that needs to be communicated through the network, followed by a trade-off between cost to pay for local image data processing and cost to pay for data transmission.
Local processing can involve simple image processing algorithms (such as background subtraction for motion/object detection, and edge detection) as well as more complex image/vision processing algorithms (such as feature extraction, object classification, scene reasoning). In order to extract necessary information from different images, a camera node must employ different image processing algorithms. The aforementioned vision processing tasks require extracting features about an event, which in the case of energy constrained camera nodes can be hard or even impossible to achieve. Thus, although it is desirable to have high-resolution data features, costly feature extractions actually should be avoided.

Autonomous Camera Collaboration

Visual sensor networks are envisioned as distributed and autonomous systems, where cameras collaborate and, based on exchanged information, reason autonomously about the captured event and decide how to proceed. Through collaboration, the cameras relate the events captured in the images, and they enhance their understanding of the environment. This particular feature of visual sensor networks is exploited by Nash bargaining solution

framework. The collaborative capabilities employ the bargaining process, make it possible to implement the framework.

2.2 Reference visual analysis lineup

The aforementioned visual processing tasks require extracting features about an event. Visual analysis is performed by feature detection algorithms resulting in visual features description that can be used according to a user application. Primarily visual analysis is implemented enabling processing and evaluation of human or object movement/motion and behaviours.
In computer vision and image processing, visual features are used to encode the image structure in a spatial neighbourhood (the way image patch looks like) around each interest point of the image. Local image structure around interest points simplifies further processing in the vision system. In addition, the features are highly distinctive and, at the same time, robust to noise, detection errors, and geometric and photometric deformations. Those characteristics allow a single feature to be correctly matched with high probability against a large database of features [7].
Features are the result of an operation applied to generic neighbourhood. Such type of operation is called feature detection and feature description. *Feature extraction* involves simplifying the amount of resources required to describe a large set of data accurately. Fewer resources means also less computation power, less energy and less memory consumed. This operation is used to detect and isolate various desired number or shapes (features) of a digital image.
Feature detection makes decisions locally regarding presence of a feature of a given type at a given image point or not, seeking for image points with special characteristics, such as saliency or high contrast, which depends on the specific type of detector. Main basic known detectors are edge, corner and blob detectors. The result of the feature detection is often represented in terms of coordinate sets of the image points where a feature is detected.
Once a feature is detected, a local image patch can be extracted from the feature vicinity to characterize its surroundings region, known as feature de-

scription process. The result is known as a feature descriptor or feature vector, which is usually a high dimensional vector.
Cost of extracting these features is minimized by taking a cascade filtering approach, in which the more expensive operations are applied only at locations that has passed the first phase.
Visual feature vector, defined in terms of a specific structure in the image data, can be often represented in this way:

- Keypoint position: coordinates of the pixel location
- Scale: the scale of an interest point
- Orientation: one or more orientations are assigned to every keypoint location based on local image gradient directions
- Keypoint descriptor: is a descriptor vector (64 elements for SURF) for each keypoint such that the descriptor is highly distinctive and partially invariant to illumination
- Fast Hessian response: response of 'fast Hessian' detector, which relies on integral images to reduce computation time

Usually feature extraction algorithms belong to one of two types:

- Blob feature extraction algorithms
- Binary feature extraction algorithms

Vision processing tasks require extracting features about an event taking into account energy and memory constraints of the camera node.
A light-weighted algorithm that is widely used in computer vision to perform feature extraction and detection is called SURF. In the following Sections 2.2.1 and 2.2.2 we briefly review main characteristic of two detection/description algorithms, which are featured in this thesis work.

2.2.1 SURF - Speeded Up Robust Features

SURF is scale- and rotation invariant interest point detector and descriptor [8]. It is fast and robust algorithm for local, similarity and invariant image representation and comparison. SURF selects interest points of an image from silent features of its linear scale-space, and then builds local features based on the image gradient distribution.

The search for discrete image correspondences can be divided into three main steps. First interest points are selected at distinctive locations in the image, such as corners, blobs and T-junctions. Next, the neighbourhood of every interest point is represented by a feature vector. This descriptor has to be distinctive and, at the same time, robust to noise,detection errors, geometric and photometric deformations. Finally, the descriptors are matched between different images.

The detector is based on the Hessian matrix because of its good performance in computation time and accuracy. Gaussian filters are replaced by a simpler alternative - LoG approximation, that approximates second order Gaussian derivatives, and can be evaluated very fast using integral images independently of size. The performance is comparable to the one using the discretised and cropped Gaussian.

Scale spaces are usually implemented as image pyramids. The images are repeatedly smoothed with a Gaussian and subsequently sub-sampled in order to achieve a higher level of the pyramid.

In order to localize interest points in the image and over scales, a nonmaximum suppression in a 3 x 3 x 3 neighbourhood is applied. The maxima of the determinant of the Hessian matrix are then interpolated in scale and image space with the Brown's method. Scale space interpolation is especially important in our case, as the difference in scale between the first layers of every octave is relatively large.

The SURF descriptor is implemented by a set of steps. The first step consists of fixing a reproducible orientation based on information from circular region around the interest point. For that purpose, we first calculate Haar-wavelet responses in x and y direction and this in a circular neighbourhood. In keep-

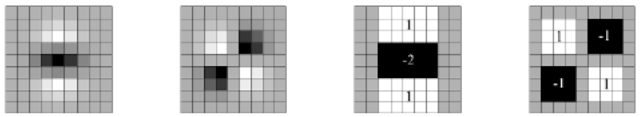

Figure 2.1: The (discretised and cropped) Gaussian second order partial derivatives

ing with the rest also the wavelet responses are computed at that current scale *s*. The wavelet responses should also be weighted with Gaussian and centred at interest point. The responses are represented as vectors in a space with horizontal response strength along the abscissa and the vertical response along the ordinate. The horizontal and vertical responses are summed and then yield to a new vector.

To determine the SURF descriptor, the first step is to construct a square region, see Figure 2.1 centred around the interest point, and oriented along the orientation selected previously. The region is split up into smaller square sub-regions. For each subregion we compute few simple features, and hence first set of entries for the feature vector is formed. Each sub-region has a four-dimensional descriptor vector **v**. The wavelet responses are invariant to a bias in illumination (offset). Invariance of contrast (a scale factor) is achieved by turning the descriptor into a unit vector.

The important difference between SURF and some other scale and rotation-invariant interest point detector and descriptor, is its fast computation of approximate differential operators in the scale-space, based on integral image representation and box filters, which enabling real-time application such as tracking and object recognition.

2.2.2 BRISK - Binary Robust Invariant Scalable Keypoints

Usually feature extraction algorithms need to determine a trade-off between description quality and computational requirement. BRISK feature description algorithm sets new milestone in determining the point of balance.

BRISK method represents different approach to extraction features, referring

to binary feature extraction algorithms. It is also fast and its speed lies in the application of a novel scale-space FAST-based detector in combination with the assembly of a bit-string descriptor from intensity comparisons retrieved by dedicated sampling of each keypoint neighbourhood [9].
Points of interest are generated from an image, structured as follows:

- **Scale-space point detection** Points of interest are identified across both image and scale dimensions using a saliency criterion. Keypoints are detected in octave layers of the image pyramid as well as in layers in-between that boosts efficiency. The location and scale of each keypoint are obtained in the continuous domain via quadratic function fitting.

- **Keypoint description** A sampling pattern consisting of points lying on appropriately scaled concentric circles is applied at the neighbourhood of each keypoint to retrieve gray values, processing local intensity gradients, the feature characteristic direction is determined.

BRISK keypoints can be matched more efficiently thanks to binary nature of the descriptor.
With the aim of achieving invariance to scale which is crucial for high-quality keypoints, enhanced by searching for maxima not only in the image plane, but also in scale-space using the FAST scores as a measure for saliency. The BRISK detection algorithm estimates the true scale of each keypoint in the continuous scale-space.
In the BRISK framework, the scale-space pyramid layers consist of n octaves c_i and n intra-octaves d_i, typically $n = 4$. The octaves are formed by progressively half-sampling the original image. Each intra-octave d_i is located in-between layers c_i and c_{i+1}, see Figure 2.2. The first intra-octave is obtained by downsampling the original image by a factor of 1.5, while the the rest of intra-octave are derived by successive halfsampling. There are several ways to choose a mask shape for keypoints. In BRISK is usually used the 9-16 mask, which essentially requires at least 9 consecutive pixels in the 16-pixel circle to either be sufficiently brighter or darker then the central pixel for the FAST criterion to be fulfilled.

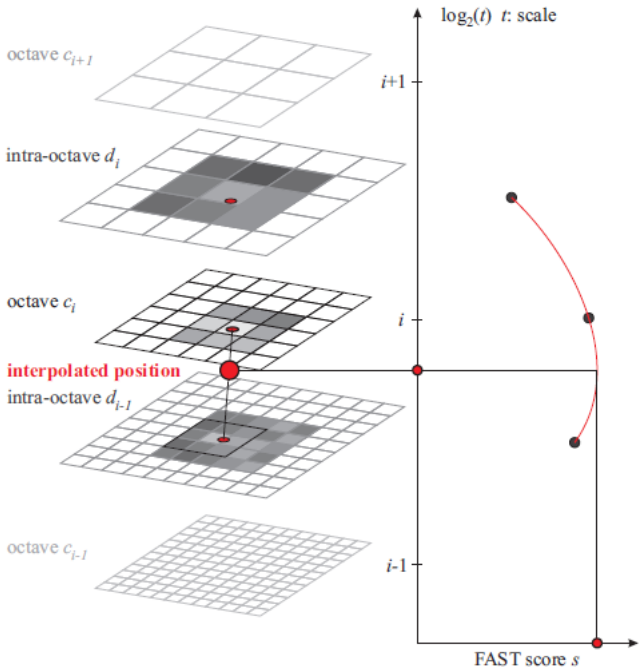

Figure 2.2: Scale space interest point detection

The FAST 9-16 detector is applied on each octave using the same threshold T to identify potential regions of interest. Next, the points belonging to these regions are subjected to a non-maxima suppression in scale-space. Firstly the point in question needs to fulfil the condition of maximum with respect to its 8 neighbouring FAST scores s in the same layer. The score s is considered as maximum threshold still considering an image point a corner. Secondly, the scores in the layer above and below need to be lower as well. Considering the image saliency as a continuous quantity not only across the image but also along the scale dimension, we perform a sub-pixel and continuous scale refinement for each detected maximum. Next these refined scores are used to fit a 1D parabola along the scale axis yielding the final score estimate and scale estimate at its maximum. As a final step we re-interpolate the image coordinates and between the patches in the layers next to the determined scale.

Given a set of keypoints (consisting of sub-pixel refined image locations and associated floating-point scale values), the BRISK descriptor is composed as

a binary string by concatenating the results of simple brightness comparison tests. First we need to identify the characteristic direction of each keypoint to allow use of orientation-normalized descriptors and hence to achieve rotation invariance which is the key to general robustness. Also we carefully select the brightness comparisons with the focus on maximizing descriptiveness. The pattern used for sampling the neighbourhood of the key is illustrated in Figure 2.3. It defines N location equally spaced on circles concentric with the keypoint. In order to avoid aliasing effect when sampling the image intensity of a point a Gaussian smoothing is applied. Positioning and scaling the pattern accordingly for a particular keypoint k in the image.
For the formation of the rotation- and scale-normalized descriptor BRISK applies rotation on the sampling pattern. BRISK uses significantly fewer sampling points than pairwise comparisons, limiting the complexity of looking up intensity values. The comparisons here are restricted spatially such that the brightness variations are only required to be locally consistent. With the sampling pattern and the distance threshold as shown above, we obtain a bit-string of length 512. Matching of BRISK descriptors is a simple computation of their Hamming distance.

BRISK and SURF algorithms are established leaders in the field of computer vision, which exhibit feasible performance along with computationally efficiency, therefore are used in this research work to evaluate the Nash bargaining solution performance.
In the next chapter a review on game theory, bargaining games and Nash bargaining solution concept are given. Knowing all the necessary bargaining game components and parameters it is comprehensible to implement the NBS concept within the framework of visual sensor networks.

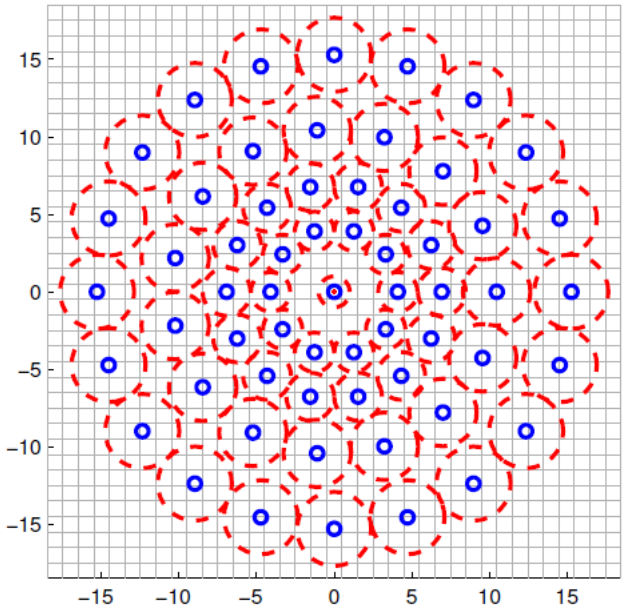

Figure 2.3: The BRISK sampling pattern, $N = 60$

Chapter 3

Game theory and Nash bargaining solution

Nash bargaining solution is a fundamental concept that has applications in different areas of technology and economy. It is also widely exploited for sensor networks. NBS is usually applied to overcome the energy constrains in a visual sensor network and provide longer network lifetime. This chapter describes in detail theoretical basis of Nash bargaining solution, in particular its characteristics, relation to game theory and cooperative games and continues with review of NBS applications.

3.1 Game Theory

Game theory is the name given to the methodology of using mathematical tools to model and analyze situations of interactive decision making [10]. These are situations involving several decision makers (called *players*) with different goals, in which the decision of each affects the outcome for all the decision makers. This interactivity distinguishes game theory from standard decision theory, which involves a single decision maker, and it is its main focus. Game theory tries to predict the behaviour of the players and sometimes also provides decision makers with suggestions regarding ways in which they can achieve their goals.

The foundations of game theory were laid down in the book *The Theory of Games and Economic Behaviour* published in 1944 by the mathematician John von Neumann and the economist Oskar Morgenstern. The theory has been developed extensively since then and today it has applications in a wide range of fields. The applicability of game theory is due to the fact that it is a context-free mathematical toolbox that can be used in any situation of interactive decision making. A partial list of fields where theory is applied, along with examples of some questions that are studied within each field using game theory, includes:

- Theoretical economics
- Networks
- Political Science
- Military applications
- Inspection
- Biology

Traditionally, game theory is divided into two major subfields: strategic games, also called noncooperative games, and coalitional games, also called cooperative games. Broadly speaking, in strategic games the players act independently of each other, with each player trying to obtain the most desirable outcome given his or her preferences, while in coalitional games the same holds true with the stipulation that the players can agree on and sign binding contracts that enforce coordinated actions. Game theory does not deal with the quality of justification of these enforcement mechanisms; the cooperative game model simply assume that such mechanisms exist and studies their consequences for the outcome of the game.

3.2 Bargaining Games

Bargaining games is a part of coalitional games subfield. A bargaining game model situation in which two or more players bargain towards an agreed-

upon outcome. The set of all possible outcomes is called *a feasible set* and each outcome can be attained only by unanimous agreement of all players. Different players typically prefer different outcomes, which explains bargaining aspect of the model. A default outcome called the *disagreement point*, is released if the players fail to reach an agreement.

A solution concept for bargaining games is a function that assigns to every bargaining game an outcome that can be looked at as the outcome that would be recommended to the players by an arbitrator or a judge. Solution concept could satisfy a list of desirable properties. There also exists a unique solution concept that satisfies all the properties, namely Nash solution for bargaining games. The model can be extended to bargaining games with more than two players.

Frequent case includes two or more players that need to conduct negotiations over an issue, with a pay-off to each player dependent on the outcome of the negotiation process. Influenced by variety of factors there is a range of outcomes available, if only the players can come to an agreement and cooperate. Sometimes negotiations do not lead to an agreement. Bargaining games for two players is modelled by using a set $S \subset \mathbb{R}^2$ and a vector $d \in R^2$. A point $x = (x_1, x_2) \in S$ represents a potential bargaining outcome expressed in units of utility, where x_i is player i's utility from the bargaining outcome. The set S thus represents the collection of possible bargaining outcomes and the vector d represents the outcomes in the case where no agreement emerges from the bargaining process. The model was first introduced and studied by Nash.

In case players fail to reach compromise, it is necessary to turn to an arbitrator, to determine fair agreement. Arbitrator in order to be called reliable needs to base his methods of choosing the agreement on principles agreeable to both players, and show how his proposed agreement follows from those principles. It is also desirable for the principles to determine an agreement in such a way that any other suggested agreement would fail to satisfy one or more principles.

3.2.1 Theoretical model

A bargaining game *is an ordered pair (S,d)* in which:

- $S \subset \mathbb{R}^2$ is non-empty, compact, and convex set, called the set of alternatives.

- $d = (d_1, d_2) \in S$ is called disagreement point (or conflict point).

- There exist an alternative $x = (x_1, x_2) \in S$ satisfying $x \gg d$.

For example, Larry and Sam can, by cooperating, attain a potential profit of \$100. They need to agree on dividing this sum between them. If they cannot come to an agreement, they will not cooperate, there will be no profit, and neither will receive any pay-off. If they come to an agreement, they will cooperate, and divide the money according to the agreement. However, Larry will be required to pay tax at a rate of 50% of his share of the profit, whereas Sam will be taxed at a rate of only 30% of his share. In this case the disagreement point d=(0,0)and the set of possible agreements is the interval S between (50,0) and (0,70), see Figure 3.1.

Denote the collection of all the bargaining games by F.
If two players agreed on an alternative $x = (x_1, x_2) \in S$, then Player 1's pay-off is x_1, and Player 2's pay-off is x_2. If the players cannot come to an agreement, the outcome of the game is d; Player 1's pay-off is d_1 and Player 2's pay-off is d_2.
The assumption appearing in the definition of bargaining games are justified as follows:

- The set of alternatives S is bounded; maximal and minimal outcomes of each player are bounded

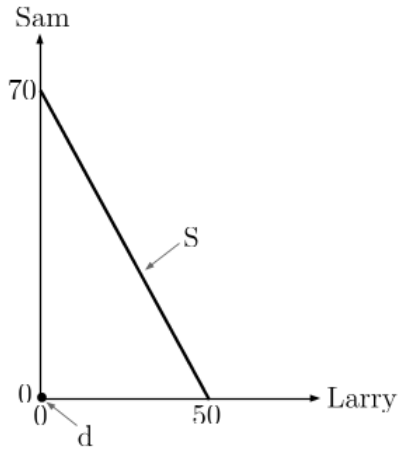

Figure 3.1: Graphic presentation of the bargaining game

- The set of alternatives S is closed, and therefore, the boundary of every sequence of possible outcomes in S is also in S. Without this assumption, it may be the case that there is no optimal solution

- The set of alternatives S is a convex

- We assume that there exists an alternative $x \in S$ such that $x \gg d$, to avoid dealing with degenerate cases in which there is no possibility that both players can profit from an agreement

A solution concept is a function φ associating every bargaining game $(S, d) \in F$ with an alternative $\varphi(S, d) \in S$.
The point $\varphi(S, d) \in S$ is the one that the arbitrator will propose and the players accept as an agreement.

3.2.2 Properties of Nash solution

The properties were proposed by John Nash in 1953, and they are mathematical expressions of principles that could guide an arbitrator who is called

upon to propose bargaining agreement. There exists a unique solution concept satisfying these properties.

Symmetry A bargaining game $(S, d) \in F$ is symmetric if the following two properties are satisfied:

- $d_1 = d_2$ the disagreement point is symmetric
- $x = (x_1, x_2) \in S$, then $(x_2, x_1) \in S$

Geometrically, symmetry implies that S is symmetric with respect to the main diagonal in $\mathbb{R}^2$, where the disagreement point is located.
A solution concept φ is symmetric if for every symmetric bargaining game $(S, d) \in F$ the vector $\varphi(S, d) = (\varphi_1(S, d), \varphi_2(S, d))$ satisfies $\varphi_1(S, d) = \varphi_2(S, d)$.

Efficiency An alternative $x \in S$ is called an efficient point of S if there does not exist an alternative $y \in S, y \neq x$ such that $y \geq x$.
A solution concept φ is efficient if $\varphi(S, d) \in PO(S)$ for each bargaining game $(S, d) \in F$. Where *PO(S)* the set of efficient points of S.

Covariance under positive affine transformations It is reasonable to require the solution concept be independent of the units of measurement.
Another possible property to adopt is *covariance under translation*. The property implies that if we add a constant to each one of a certain player's pay-offs, the solution will change by the same constant.
A solution concept φ is covariant under positive affine transformations if for every bargaining game $(S, d) \in F$, for every vector $a \in \mathbb{R}^2$ such that $a \geq 0$, and for

$$\varphi(aS + b, ad + b) = a\varphi(S, d) + b. \tag{3.1}$$

Independence of irrelevant alternatives (IIA) A solution concept φ satisfies the property of IIA if for every bargaining game $(T, d) \in F$ and every

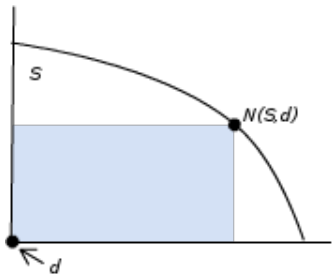

Figure 3.2: The Nash solution (the darkened rectangle is the rectangle of maximal area)

subset $S \subseteq T$,

$$\varphi(T, d) \in S \Rightarrow \varphi(S, d) = \varphi(T, d). \tag{3.2}$$

There exist a unique Nash solution concept N that satisfies properties of symmetry, efficiency, covariance under positive affine transformations and independence of irrelevant alternatives.

The solution *N(S,d)* of the bargaining game *(S,d)* is the individually rational alternative x in S that maximizes the area of the rectangle whose bottom left vertex is d, and whose top right vertex is x. See Figure 3.2

$$N(S, d) := argmax_{\{x \in S, x \geq d\}}(x_1 - d_1)(x_2 - d_2). \tag{3.3}$$

The point *N(S,d)* is called Nash agreement point (or the Nash solution) of the bargaining game (S,d). There are three theorems that support Nash solution concept feasibility:

- For every bargaining game $(S, d) \in F$ there exist a unique point in the set

$$argmax_{\{x \in S, x \geq d\}}(x_1 - d_1)(x_2 - d_2). \tag{3.4}$$

- The solution concept N satisfies the properties of symmetry, efficiency, covariance under positive affine transformations, and independence of

irrelevant alternatives

- Every solution concept φ satisfying symmetry, efficiency, covariance under positive affine transformations, and independence of irrelevant alternatives is identical to the solution concept N defined by Formula (3.3)

Consequently if one of the properties described above is left out, uniqueness is lost.

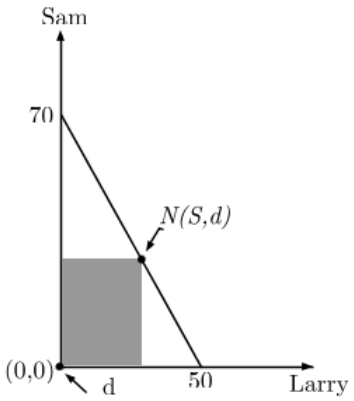

Figure 3.3: Nash solution to Sam and Larry problem

Recalling the example introduced earlier about Larry and Sam, who are trying to split 100$ between each other with the constraint that both need to pay different denomination taxes afterwards. We have defined a range where the solution is placed, according to Figure 3.3. The disagreement point is said to be 0$, meaning neither of the players get a pay-off in case they do not come to an agreement. The solution values are lied within the range described by the linear equation

$$y = -\frac{7}{5} * x + 70 \tag{3.5}$$

where x is pay-off range that is available for Larry, which is $0 \leq x \leq 50$ and

y accordingly the pay-off that is available to Sam, that is $0 \leq y \leq 70$. Then according to Nash bargaining solution concept we need to find such *(x,y)* that maximize the area under the line, see Figure 3.5.

$$\begin{aligned} max(x * y) \\ s.t. \\ 0 \leq x \leq 50 \\ y = -\frac{7}{5} * x + 70 \end{aligned} \tag{3.6}$$

That leads to the solution

$$\begin{aligned} x = 25\$ \\ y = 35\$ \end{aligned} \tag{3.7}$$

Implying that according to Nash bargaining solution concept Larry receives a pay-off equal to 25$ after paying all the taxes, and Sam receives pay-off of 35$. And this solution is optimal.

3.3 Application of Nash bargaining solution in visual sensor networks

The use of tools and techniques suggested by cooperative game theory and NBS in particular helps to address the challenges appointed by visual sensor networks and has a great improvement potential.

Resource management - Spectrum allocation and bandwidth allocation Resource management is an important consideration while designing a wireless sensor network. Impulse Radio-Ultra WideBand communication is a strong candidate for short-range biomedical wireless sensor networks. The problem of finding the fair sub-band schedule into a multi-objective integer programming problem is planed to be solved using NBS [11], [12].

Energy saving and power control Currently, the energy problem remains one of the major obstacles somehow preventing the complete exploitation of WSN technology. Energy saving and power control strategies should be devised at sensor nodes as well as in the network. For instance, the Game-theoretical Total Link algorithm a MAC scheme based on p-persistence slotted ALOHA is considered. In order to determine the value of the attempt probability p the authors constructed the p-persistence as a simple non-cooperative game which involves generalized pay-offs reflecting energy saving and throughput. They modelled the power allocation problem as a two-person bargaining game, and used Nash bargaining solution to achieve win-win strategy [13].

Topology optimization Game theory has played an active role in the aspect of routing algorithm and topology control. One of application scenarios considers a locally computable heuristic "recommendation algorithm" referring to as LocalHeur for convenience of incentive compatible topology control for selfish all-to-one routing which is modelled by a locally minimum cost forwarding game in the absence of complete global information for realistic scenarios [14].

As a following step I introduce an explanation and main functioning principles of game theory and Nash bargaining solution.

In the next chapter there will be explicitly explained how Nash bargaining solution can be used in Visual sensor networks framework, in particular, its application in local image processing performed by a camera node, the way to reduce energy consumption via collaboration of camera nodes in case they have overlapped field of view. Also the following chapter introduces all the components that are necessary to be preliminary determined, such as utility function, or camera calibration that defines overlapped field of view.

Chapter 4

Problem statement

At this point there have been given detailed reviews of visual sensor networks and NBS. In this chapter there will be introduced an outline that makes it possible for a visual sensor network to exploit NBS formula with the scope of energy saving. It is followed by the definition of utility function and its experimental evaluation for the established reference scenario.

4.1 Reference scenario

Consider one of the most widely used applications of visual sensor networks for smart surveillance of public areas. We can make an assumption that such a visual sensor network is densely populated with camera nodes, see Figure 4.1. Each of them carries out functions that are assigned to it and processes the acquired images. In a standard case image processing is performed by each camera independently and as if it is the only camera node in the area. However those camera nodes are most likely to have overlapped field of views among them, implicating that some of camera nodes in fact observe the same area and process the same (or partially the same) images, which leads to re-evaluation of relevancy of operations performed by each camera node and visual sensor network in general as well as it is most certainly undesirable energy consumption. This scenario is indicative for this thesis work.

As it was mentioned before energy saving remains priority for any sensor network because it is usually battery-charged. Hence any redundant operations that a sensor network performs lead to a shorter lifetime and inconvenience of usage. A solution for the aforementioned outline is provided by NBS.

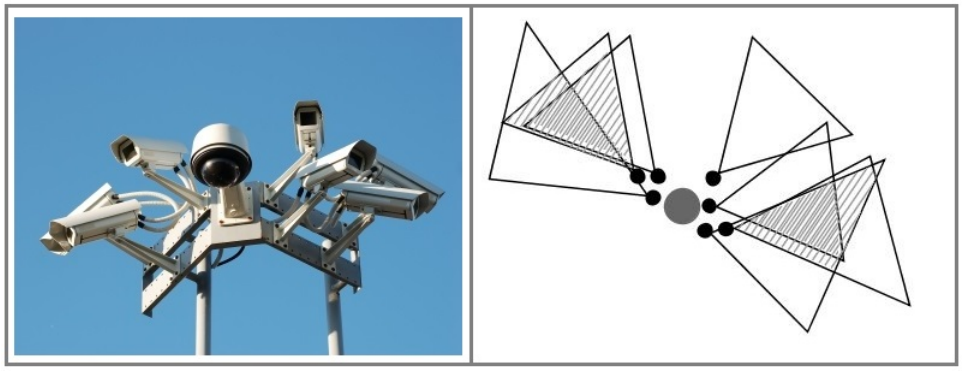

Figure 4.1: Public area, camera nodes, overlapped field of view

NBS-based strategy can be exploit in such a way to reduce the portion of image each of the camera nodes with overlapped field of view have to process and as a consequence of processing less - more energy saved for longer network lifetime.

For the reference scenario we fix number of camera nodes *N=2,3*, considering both completely and partially overlapped field of view. We give to each camera node only two feature description/detection algorithms: SURF, BRISK and consider that all the camera nodes are able to communicate parameters to one another. Parameter exchange include exchange of residual energy budget, power consumption, utility function coefficients as well as intrinsic camera parameters. Communicating the parameters is essential for performing bargaining game and implementing NBS. We feature every node with NBS-based strategy making it a collaborator that bargains with other camera nodes for a portion of image (image visible to all of the collaborators) to process. Each camera-collaborator can decide what portion it is ready to process based on its real-time parameters and parameters of neighbouring collaborators. Once the outcome of the bargaining game is obtained each camera establishes its portion of image to process (to apply BRISK or SURF

algorithms) while the remaining image portion is shared and processed by neighbouring bargaining participants. NBS is a distributed method meaning that each node calculates the share autonomously but the solution remains unique and coherent for all collaborating cameras. Therefore not only we gain longer network lifetime but we also do not require any topology changes to the existing sensor network.
The components needed to implement NBS-based strategy are described in Chapter 3, Section 3.2.1, utility function is among them. In our case it is obtained experimentally.

4.2 Utility function

First of all it is important to define the "currency" that every player prefers to bargain for. Referring to the example of Larry and Sam, they performed bargaining over 100$. In case of visual sensor network, the unit every camera node bargains for is unit of saved energy, amount of energy each camera saves, if it agrees to collaborate, the utility function is accordingly defined in units of energy. Initially I define a function that represents amount energy that is required for every camera node to process a portion of the image. Then the utility function for every camera node involved in bargaining equals to amount of energy that it does not need to process, meaning, summation of energies spent by other cameras to process their fraction of the image. The fraction of image that a camera does not need to process translates into a quantity of saved energy, and every bargaining game is played exactly over this quantity.
Sensor network characteristic component that requires the largest amount of energy is local processing of the image. Taking this into account, It is relevant firstly to determine the time it takes to apply one of the feature extraction algorithms (BRISK, SURF) described in Chapter 2. And from time it takes to process the image, the energy spent can be easily obtained by multiplying it by power consumption of the particular camera node under consideration.
Elapsed time - time it takes to apply feature extraction algorithm. It is de-

scribed by a curve, function of time from image size. The order and the shape of the curve is defined experimentally, setting up the proper environment and structure.

4.2.1 Development environment

Function curve describes time needed for a given feature extraction algorithm to process image of a fixed size. Curve order and coefficients are obtained experimentally. Series of tests are launched on two types of hardware, using OpenCV library that enables computer vision and feature extraction algorithms and the test images are part of dataset provided by IEEE surveillance evaluation workshop.
Software implementation of feature extraction/detection algorithms is done by OpenCV library, that has C/C++, Python and Java interfaces. It supports Windows, Linux, Mac OS, iOS and Android operating systems. The library provides set of tools for computer vision and it is free for academic use. Its main focus comprises real-time application use. The library finds its implementation in variety of areas from mines inspection and map stitching to advanced robotics.
Feature extraction/detection algorithms SURF and BRISK, that are introduced in Chapter 2 find their realization in OpenCV library. To be able to explore BRISK and SURF performance from different prospective and to define elapsed time function coefficients in different environments there were chosen two types of hardware:

- BeagleBone, minicomputer, running Unix-based OS
- Mobile device with Android operating system

The script implementation differs depending on hardware. In case of BeagleBone platform it is slightly easier to set up the script. BeagleBone runs Linux-based operation system and compiles C/C++ scripts, advantageously agrees with OpenCV library natively written in C++.
First, it is necessary to make sure the following libraries and tools are installed prior to OpenCV library installation:

- GCC 4.4.x or later
- CMake 2.6 or higher
- Git
- GTK+2.x or higher, including headers (libgtk2.0-dev)
- ffmpeg or libav development packages: libavcodec-dev, libavformat-dev, libswscale-dev

The latest stable OpenCV library version can be downloaded from SourceForge webpage.
In case when one performs script implementation working with Android OS it is recommended to install Android Developer Tool (ADT) Bundle, that includes Eclipse Integrated Development Environment and an Android Virtual Device Manager. An Android Virtual Device is an emulator configuration that permits modelling an actual device by defining hardware and software options to be emulated by the Android Emulator. Aside from ADT bundle it is a requirement to install JDK which is Java development kit provided by Sun Corporation as it is necessary condition because of Java programming language used for Android applications development. Finally considering that OpenCV library is natively written in C++ programming language while providing Java interface, one requires NDK (Native development kit) to be able to link OpenCV to current IDE and latest available OpenCV adopted for Android from Sourceforge webpage. In my case it is

```
OpenCV-2.4.9-android-sdk.zip
```

Linking OpenCV library

For BeagleBone platform OpenCV installation is conducted through following steps:

1. Create a temporary directory, where generated Makefiles will be located. Makefiles include project files as well the object files and output binaries.

2. Enter a created temporary directory where set of steps is the following:

```
cd ~/opencv
mkdir release
cd release
cmake -D CMAKE_BUILD_TYPE=RELEASE -D
                                CMAKE_INSTALL_PREFIX=/usr/local ..
make
sudo make install
```

This concludes the implementation stage and makes it possible to use OpenCV tools in the script testing elapsed time function on BeagleBone platform. Implementation process of OpenCV for Android, taking into account all the prerequisites that were listed earlier are satisfied keeps the following order of steps described below:

1. Create a new folder for Android with OpenCV development. I unpacked OpenCV SDK to the

   `dariab/Documents/workspace/`

2. Unpack the SDK archive into the chosen directory.

```
unzip ~/Downloads/OpenCV-2.4.9-android-sdk.zip
                                C:/dariab/Documents/workspace/
```

3. Open Eclipse and choose your workspace location, it is recommended to create a new location for Android + OpencCV projects.

4. Import OpenCV library through Package Explorer and Import function. OpenCV library is packed as a ready-for-use Android Library Project. Eclipse should automatically locate OpenCV library and samples.

Once Eclipse complete library initialization we are able to start with the application that tests Android and its elapsed time function.

4.2.2 Structure of the implemented function

A programming script that executes the experiment with the scope to obtain utility function parameters computes and collects image size values and corresponding elapsed time executing given feature detection algorithm (SURF/BRISK), then data is output to a file. Each file contains 2-column array with image size and elapsed time values for all the images from the dataset and has following structure:

```
 ---------------------------------------
| image size [pixel] | elapsed time [sec] |
 ---------------------------------------
|         ...        |         ...        |
 ---------------------------------------
```

The test images (taken from the dataset) all have a same fixed size. Hence to obtain values of elapsed time for greater variety of image sizes the script performs slicing of the input image, defining a Region of Interest which further feature extraction algorithms is applied for. Each stripe is set to a certain starting dimensions and gradually increasing the width with every cycle iteration until the stripe equals to initial size of the input image, see Figure 4.2. Slicing is performed both vertically and horizontally. In case C++-based script, I am able to work with stripes with starting minimal width of 20 pixels and increase iteratively the stripe width by 20 pixels. For Java-based script the stripes has width of 155 pixels (considering vertical slicing) and of 200 pixels (considering horizontal slicing). The stripe width in this case is limited due to exceptional combination of Java application, Android OS architecture and OpenCV library.

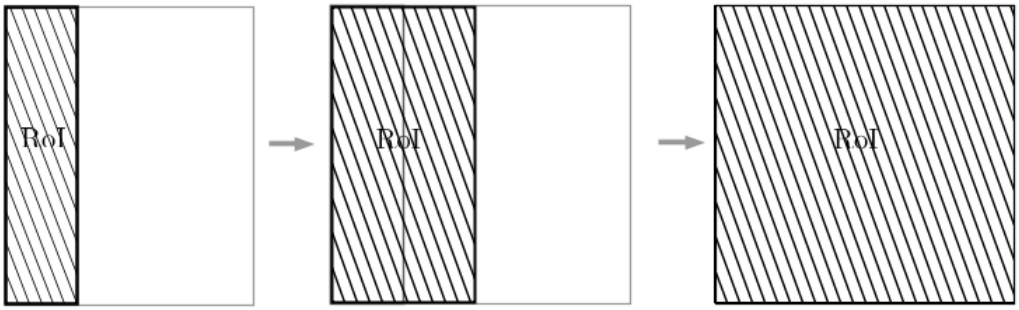

Figure 4.2: Image slicing, defining a RoI

After all the data is collected elapsed time values get averaged according to the size of the image. Considering the array and its first column that contains the different image sizes, find identical entries of image size and mean value is found for corresponding elapsed time values. This operation results into an array of smaller dimensions that contains several image size instances and their averaged elapsed times. The mean elapsed time values are input for a function that defines the curve that fits best the input values.
While part of the experiment that executes data collection is done either in C++ programming language (for BeagleBone platform) or in Java programming language (for Android OS), averaging of collected data and curve approximation is done using Matlab.

Data collection script has the following structure

```
load the image
define feature extraction algorithm
define the stripe width
initialize an array
for every stripe of the image
    * define Region of Interest (RoI) according to stripe width
    * calculate size of RoI in pixels,
      save it in an array in the first column
    * start a countdown
    * execute feature extraction algorithm on RoI
    * stop a countdown
    * save countdown value in the array in the second column
```

```
    * increase the stripe width
until stripe coincides with the size of the input image
output the array to a file
```

The set of images that is used in the data collection stage is provided taken IEEE Performance Evaluation of Tracking and Surveillance workshop data, year 2009. It contains 3120 image samples. A simple code written in bash feeds all the dataset to the the script. Hereafter I arrive to a point when there are 14 data files containing arrays with collected data. Each of the files is processed by Matlab script in order to obtain coefficients of the curve.
Matlab processing is performed according to the following scheme

```
* open the file
* load array in the working space
* find all the entries that correspond to
   the same image size (the first column)
* take corresponding values of the elapsed time
                              (the second column)
* find an average for elapsed time samples
* use Matlab function polyfit() to define the curve
                     that matches the averaged value
* display a chart with image size as x-axis
                  and elapsed time as y-axis
```

As an outcome of Matlab script there are coefficients that describes a linear function, as *linear* function happens to be the best fit based on data provided by conducted experiment. Hence the function that describes elapsed time from image size is linear and consequently utility function that is one of the components of Nash bargaining solution concept is linear and therefore the coefficients defined by the experiment can be applied to accomplish NBS-based algorithm implementation for visual sensor network.
It is worth commenting that I was only able to implement BRISK feature detection algorithm for Android platform because it is the only one available free-of-charge for academic purposes. Hence there are only two curves defined for Android-based hardware (considering Vertical and Horizontal slicing). Four curves are defined while script testing via workstation (Intel(R) Core(TM) i5 CPU 660 @ 3.3GHz, 4 GB RAM), four curves are defined for

BeagleBone working on 275 MHz CPU frequency and four curves are defined for BeagleBone working on 500MHz CPU frequency.

I mentioned before that elapsed time function, expressed by Formula (4.1) is needed to obtain amount of energy needed to locally process the image of a given size. As function is defined to be linear:

$$T_{el} = A * x + B \tag{4.1}$$

Hence the equation, describing the energy spent on image processing:

$$E_{cons} = T_{el} * P_{cons} \tag{4.2}$$

Where P_{cons} is a value of power consumption that depends on what what hardware is in use and can be found in technical specifications.
The Diagrams 4.3, 4.4, 4.5, 4.6, 4.7, 4.8 demonstrate sampled and averaged linear function based on collected data.

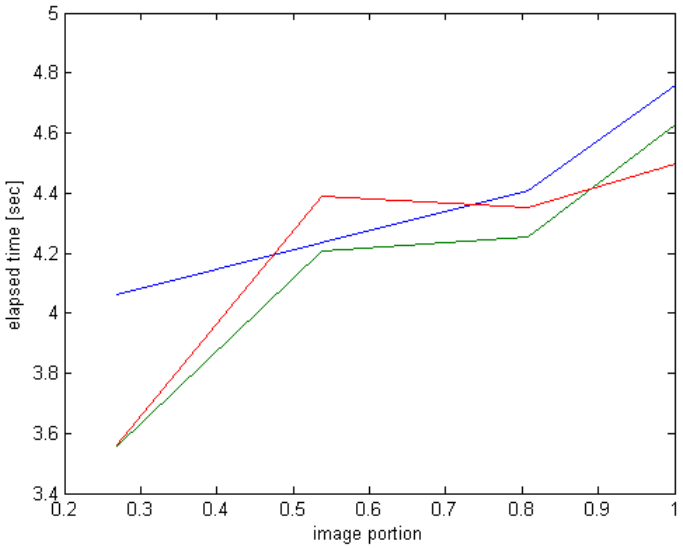

Figure 4.3: Linear function, sampling several generic images (Android OS, BRISK, horizontal slicing)

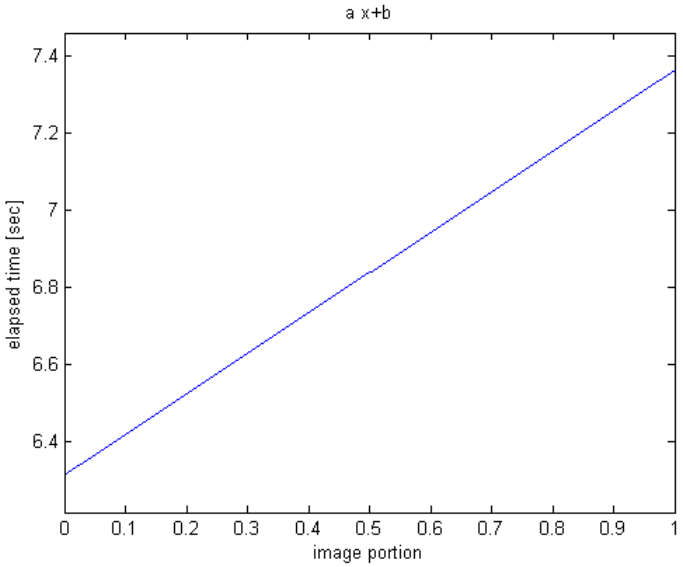

Figure 4.4: Linear function, obtained from averaged values (Android OS, BRISK, horizontal slicing)

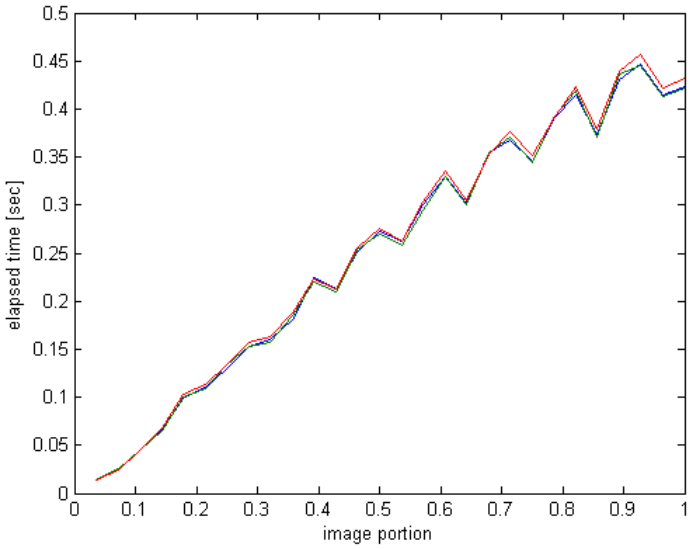

Figure 4.5: Linear function, sampling several generic images (BeagleBone 275MHz, BRISK, vertical slicing)

BRISK algorithm, Android vs. BeagleBone

I made two critical observation during testing stage of BRISK feature extraction algorithm on Android platform:

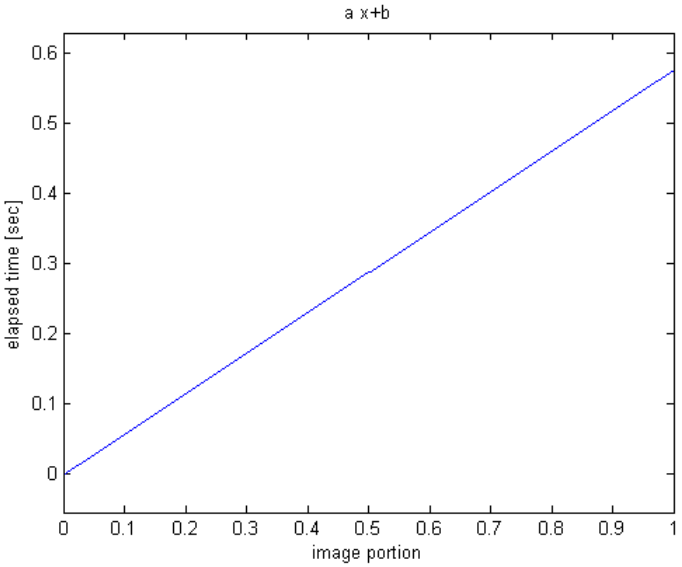

Figure 4.6: Linear function, obtained from averaged values (BeagleBone 275MHz, BRISK, horizontal slicing)

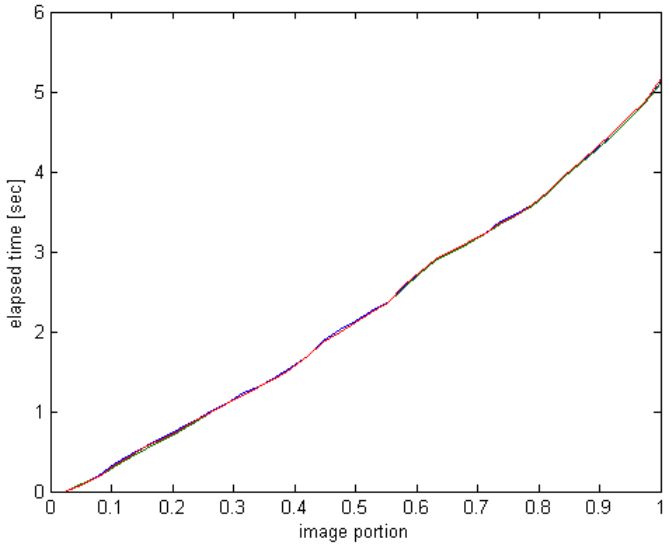

Figure 4.7: Linear function, sampling several generic images (BeagleBone 500MHz, SURF, vertical slicing)

- Parameters that control BRISK sensitivity cannot be customized or set by a user, they remain constant

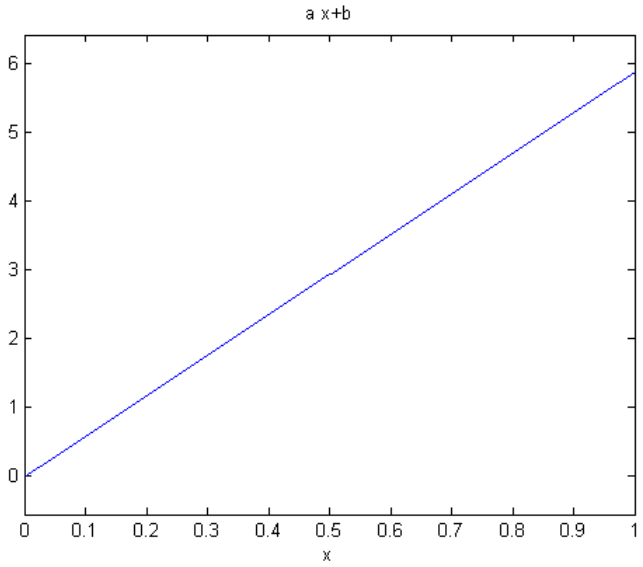

Figure 4.8: Linear function, obtained from averaged values (BeagleBone 500MHz, SURF, vertical slicing)

- Extraction/detection process elapses noticeably slower

An explanation for the latter can be found at official OpenCV library page and refers to Complex CV logic (that conducts many calls to OpenCV) and therefore will work slowly because of additional cost of Java Native Interface (JNI) calls.
An additional test should be made basing on these two critical observation. This test adjusts parameters of BRISK feature extraction algorithm used at BeagleBone platform. It customizes the parameters in such a way that BRISK at Android and BRISK at BeagleBone platform treats images equally. Subsequently using new values of elapsed time for BeagleBone we find the ratio that discloses the extent BRISK for Android slows down to.

$$\frac{Android \quad elapsed \quad time}{BeagleBone \quad elapsed \quad time} \tag{4.3}$$

BRISK algorithm is described by three parameters:

- Threshold level
- Octaves

- Pattern Scale

Firstly we need to locate the OpenCV library, that was installed beforehand at the previous stage of the research, see Section 4.2.1. Then look into the open source file, found at

```
OpenCV-2.4.9-android-sdk.zip\OpenCV-2.4.9-android-sdk\sdk\native
                    \jni\include\opencv2\features2d\features2d.hpp
```

And here we see how the parameters were fixed for BRISK algorithm:

```
CV_WRAP explicit BRISK(int thresh=30,
                       int octaves=3,
                       float patternScale=1.0f);
```

The next step is to assign these values to parameters of BRISK algorithm we used in the script that tests BeagleBone platform and run the test again. We also need to adjust the stripe width that was used in Android platform test:

- Horizontal slicing - 155 [pixels]
- Vertical slicing - 200 [pixels]

The values are forced by exceptional features of combined performance of Android platform and OpenCV library, that is adopted for it.
Accounting for all the parameters' updates the tests are run on BeagleBone platform both for 275 MHz and 500 MHz CPU frequencies resulting into four data files with new adopted elapsed time values. Using this data it is trivial task to find a performance ration of Android and BeagleBone platforms. The script that calculates the ratio can be expressed in pseudocode as follows:

```
for every pair of elapsed time values (one from Android data file,
another - from BeagleBone file)
that corresponds to certain image size:
   * find the ratio: Android elapsed time/BeagleBone elapsed time
   * find mean value
   * at this point we obtain ratios found for certain image size
     find mean value over these ratios
```

As a result we obtain following correspondence of platforms' performances:

$$\frac{Android \quad elapsed \quad time}{Divider*} \tag{4.4}$$

Horizontal striping

*	Ratio Value
BeagleBone elapsed time, 275 MHz	14
BeagleBone elapsed time, 500 MHz	14

Vertical striping

*	Ratio Value
BeagleBone elapsed time, 275 MHz	12
BeagleBone elapsed time, 500 MHz	12

That leads us to a conclusion that consequence of OpenCV library being natively C++-based library and once adopted for Java and Android platform shows **12-14** times slower performance.
In the next chapter there will be described what components define NBS formula in the context of visual sensor network, followed by the performance evaluation of the camera nodes, indicating lifespan of the sensor network according to different scenarios.

Chapter 5

NBS-driven visual analysis

In the Chapter 4 we introduced reference framework, described problem statement and suggested the way it may be resolved and furthermore experimentally characterized the utility function. Disagreement point is the next component that is needed to be defined before the NBS-driven visual analysis can be implemented.

5.1 Disagreement point

If I refer to a case with Larry and Sam, it is decided that no player receives the pay-off if they refuse to collaborate and this was their disagreement point in that case.
In case of visual sensor network when saved energy is a unit of measure the disagreement point should be forced to be such a state where no energy can be saved, meaning that each camera node simply process the entire image.

For this research I observed a case of 2 and 3 camera nodes involved in cooperation, having completely overlapped field of view, and having partially overlapped field of view.

5.2 Nash Bargaining Solution Implementation

For this thesis work it is decided to feature the performance evaluation with number of camera nodes *N=2,3*. However the NBS-driven framework can be easily extended to an arbitrary number of camera nodes.
Once all the components for Nash bargaining solution formula are defined, We may recall the concept that was introduced in Chapter 3. It is possible to redefine the Formula (3.3), tailoring it for visual sensor network.
I define the parameter of residual energy budget that each camera node possesses. The idea is to get two motes to exchange their current states regarding residual energy budget, field of view and power consumption and let Nash bargaining solution concept decide on image share assignment accordingly. One of the advantages of Nash bargaining solution is that the decision is made separately by every mote involved in collaboration but remains unique, hence there is no need to introduce a cooperator node that performs centralized assignment, the solution is calculated locally at each camera.
The concepts that were used in Chapter 3 can be efficiently adjusted for the case of visual sensor network and image processing.
Pay-off for each player that was described in theoretical part of Chapter 3 maps into energy that a camera has the means to save in case it participates in bargaining and reach an agreement. Feasible utility function and how to calculate it is described in Section 4.2 by Formula (4.2). Disagreement point that was introduced in theory of bargaining games is defined as penalty that will be applied in case when camera nodes involved in bargaining are not able to come to an agreement, implying that each camera will need to process the entire image by itself and utility function that indicates the amount of saved energy will be clearly equal to zero. Recalling Nash bargaining solution, expressed with Formula (3.3).

$$argmax_{\{x \in S, x \geq d\}}(x_1 - d_1)(x_2 - d_2). \tag{5.1}$$

And following Nash bargaining solution formula for the case with two camera nodes that collaborate to split image processing.

$$argmax_{\{x \in S, x \geq d\}}[((A(1 - x_1) + B)P_{c1} - d_1)((C(1 - x_2) + D)P_{c2} - d_2)]. \tag{5.2}$$

Where P_{c1}, P_{c2} are power consumption values for mote 1 and mote 2 accordingly. Approximate values of power consumption are given in Chapter 4, Section 5.3.1.
Values of d_1, d_2 define disagreement point, which in our case is penalty each mote pays if it does not collaborate. Penalty is expressed in units of saved energy and in case of disagreement saved energy equals 0.
Coefficients *A, B, C, D* are the coefficients of the curves that is described by the linear function of elapsed time, that is introduced in Section 4.2. The function multiplied by the power consumption parameter produces the value of energy spent processing the image of a given size. Consequently energy saved by the camera under consideration is summation of energy spent by other camera nodes to process their share of the image.
While x_1, x_2 are fraction of an image that each of the motes processes in case they reach an agreement. Therefore we enforce the following assumption $x_1 + x_2 = 1$ to be able to make further simplification:

$$x_2 = 1 - x_1 \tag{5.3}$$

And denoting

$$\begin{aligned} x_1 &= x \\ x_2 &= 1 - x \end{aligned} \tag{5.4}$$

We redefine Nash bargaining solution formula for 2 motes as follows:

$$argmax_{\{x \in S, x \geq d\}}[((A(1 - x) + B)P_{c1})((C(x) + D)P_{c2})]. \tag{5.5}$$

From this point as the formula is quite trivial, it is possible to demonstrate step by step a solution for x-share. We can notice that the product is a

polynomial of second order and also it has a parabolic shape and if we neglect coefficients B and D because they represent a simple offset and expand the product we obtain the following polynomial

$$y = -AC * x^2 + AC * x. \tag{5.6}$$

For this equation maximum can be calculated with formula $x_{max} = -b/2a$ and hence

$$\begin{aligned} x_{max} &= -\frac{(AC)}{-2 * AC} \\ x_{max} &= \frac{1}{2} \end{aligned} \tag{5.7}$$

Arriving to a conclusion that the result of sharing is always half for each of the collaborating motes despite values of coefficients.

5.2.1 Nash bargaining solution, generalized version

At this point Nash bargaining solution may loose its essential importance because we simply assign a share of 0.5 to each of the motes. The current formula also has one more essential omission: it does not account for residual energy budget which is crucial factor for the image sharing process to be fair and flexible. Thereupon exists an extended version of Nash bargaining solution that is denoted generalized Nash bargaining solution and expressed by following formula

$$argmax_{\{x \in S, x \geq d\}}(x_1 - d_1)^{\alpha_1}(x_2 - d_2)^{\alpha_2}. \tag{5.8}$$

As it can be noticed the formula includes exponents α_1, α_2. From theoretical point of view the exponents have quite intuitive explanation [15].
In a given N-player bargaining problem *(S,d)* with $N \geq 2$, it is useful to give some additional weight, power to the players. In our case some of the motes have larger residual energy budget at the time of bargaining and for such motes saving energy might be not as vital as for a mote with small residual

energy budget. In order to account for this fact, one can assign for every mote i a value $\alpha_i \in [0, 1]$ which represents the bargaining power of i. The bargaining-power values are chosen such that

$$\sum_{i=1}^{N} \alpha_i = 1 \tag{5.9}$$

The idea of bargaining power allows us to give some weight to the negotiation capabilities of every player. A player having higher bargaining power would thus be a candidate to obtain an advantage in the final outcome of the bargaining process. In such a bargaining problem, if we drop the symmetry axiom of Nash, we can define generalized Nash bargaining solution. The use of the bargaining power allows the maximization to become more biased towards the mote having higher bargaining power α_i. Furthermore, one can see that, whenever the weights α_i are equal for all $\alpha_i = 1, .., N$ then reduces back to standard Nash bargaining solution concept. When we adopt Formula (5.5) to (5.8), we obtain an expression below:

$$argmax_{\{x \in S, x \geq d\}}[((A(1-x)+B)P_{c1})^{\alpha_1}((C(x)+D)P_{c2})^{\alpha_2}]. \tag{5.10}$$

Careful attention should be paid when one assigns bargaining powers to motes in the sensor network. In this case the reason for α-exponent should be based on residual energy budget that is main bargaining 'currency' for the motes. Taking into account the fact the utility function is defined in units of saved energy, hence the larger the bargaining power of the mote the greater its interest in collaboration and sharing image processing, meaning expression of the bargaining power is always inversely proportional to the residual energy budget.
For further consideration there were defined 3 possible ways to calculate α-component.

1.

$$\begin{aligned} \alpha_1 &= \frac{1}{\frac{E_{r1}}{(A+B)P_{c1}}} \\ \alpha_2 &= \frac{1}{\frac{E_{r2}}{(C+D)P_{c2}}} \end{aligned} \tag{5.11}$$

2.

$$\begin{aligned} \alpha_1 &= \frac{1}{\frac{E_{r1}}{(Ax+B)P_{c1}}} \\ \alpha_2 &= \frac{1}{\frac{E_{r2}}{(C(1-x)+D)P_{c2}}} \end{aligned} \tag{5.12}$$

3.

$$\begin{aligned} \alpha_1 &= \frac{1}{E_{r1}} \\ \alpha_2 &= \frac{1}{E_{r2}} \end{aligned} \tag{5.13}$$

Where E_{r1}, E_{r2} denote residual energy budget of a camera node and P_{c1}, P_{c2} denote power consumption of a camera node accordingly.
It is also needed to normalize α-component formulas to stay coherent with the Constraint (5.9).

$$\begin{aligned} \alpha_{1,norm} &= \frac{\alpha_1}{\alpha_1 + \alpha_2} \\ \alpha_{2,norm} &= \frac{\alpha_2}{\alpha_1 + \alpha_2} \end{aligned} \tag{5.14}$$

Once the concept of bargaining power is introduced and defined in terms of residual energy budget, Nash bargaining solution formula is in its complete and final form. This form is used in the next stage of the research work and incorporates performance analysis of Nash bargaining solution comparing it

with alternative approaches to perform image sharing with respect to variable initial conditions.

5.2.2 Minimize the summation principle

It is important to see not only the performance of NBS-based cooperation but also to see its advantages or disadvantages regarding other methods of image processing assignment or no cooperation case where every mote processes all the images by itself and does not communicate with neighbouring motes. Analysis of NBS performance is compared against another feasible strategy that also allows to perform image sharing, that is denoted as *minimize the summation principle*. The main idea is to assign image shares following the rule to minimize the summation (for all the nodes involved in collaboration) of spent energy. The formula will be referred as min-sum formula or approach in the future.

$$min(\sum(Ax_1 + B)(Cx_2 + D)) \tag{5.15}$$

5.2.3 Minimize the maximum principle

Another strategy that dictates the share of image that should be assigned to each of the camera nodes is according to the rule to minimize the maximum value of spent energy among nodes-collaborators. This approach will be referred as min-max in the future.

$$min(max(Ax_1 + B, Cx_2 + D)) \tag{5.16}$$

An another important assumption that is made for this analysis stage is that the image that motes are trying to share, shared without overlapping, hence instead of using notation x_1, x_2 that each defines a fraction of the image that is assigned to mote 1 and mote 2 accordingly in such a way that $x_1 + x_2 = 1$. it is possible with the following notation

$x_1 \rightarrow x$

$x_2 \rightarrow 1 - x$

5.3 Performance analysis

For the purpose of the analysis it is necessary to identify performance evaluation criteria. This criteria describe relative success of implemented strategy. In my case I measure NBS performance in terms of lifespan: number of times a camera node with implemented image sharing strategy is capable of participating in cooperation, and perform bargaining games before it runs out of energy.

5.3.1 Complete view overlap

The analysis evolves in two potential scenarios: the first scenario describes the case where motes have a complete overlap of view. The second scenario describes performance of the motes in case field of view is only partially overlapped.

Two motes collaboration

First of all it is fundamental to define factors that steer the analysis, starting with number of motes that participate in bargaining/image division.
The starting point is the easiest case of two motes. It is also necessary to define what type of hardware and software (feature extraction algorithms) they run.
In the chapter 4 we defined 14 possible variants of the utility function that can be assigned to a camera node. Each one of those 14 variants is characterized by 2 coefficients, that are coefficients of a linear function from image size (measured in pixels) to elapsed time (time it takes to extract features from a camera shot). For analysis stage there were chosen number 12 and number 14, that correspond to BeagleBone at CPU frequency 500MHz as hardware type and both running BRISK feature extraction algorithm, but the coefficients were obtained applying vertical and horizontal slicing accordingly. A crucial requirement that allows them to proceed with image sharing and collaboration is using the same type of feature extraction algorithm.
Also let's introduce value of initial energy budget for each of the motes. The

budget can be calculated using data from technical specification for AA type battery that is used as standard power supply for BeagleBone. From the specification it follows that the initial budget:

$$2100[mAh] * 1.5[V] = 3100[mWh] * 3600[sec] = 11340[J] \tag{5.17}$$

On the other hand device power consumption can be obtained by looking at BeagleBone specification. Worst case scenario is when a devices operates at the highest CPU frequency hence requesting larger power:

$$350[mA] * 5[V] = 1750[mW] \tag{5.18}$$

Once the value of power consumption is determined the last missing component is time it takes to process the image. Thereupon the product of time and power consumption is consumed energy. Heretofore driving factors for the forthcoming NBS analysis are as follows:

- Coefficients to define utility function
- Initial energy budget of a mote
- Power consumption of a mote

Those factors enables 8 possible cases to study, 8 different development scenarios to evaluate. The cases are described in the Table 5.1. The columns 2-5 contain coefficients of the function that describes amount of energy each camera node spend to process the image of a given size, the column 6 and 7 contain the parameter of power consumption for each of the nodes, and lastly column 8 and 9 contain initial energy budget of each camera.
For example, if we observe the case number 3 (take a look at the third row), we can say that both camera nodes have equal energy consumption parameter as both utility functions are described by the same coefficients, but also each of them has different power consumption, and equal initial energy budget.
The following step is to take a closer look at the formula of generalized NBS we have suggested and possible interpretations of $\alpha_{\mathbf{i}}$-exponent, see Formula

#	COEFFICIENTS, NODE 1		COEFFICIENTS, NODE 2		POWER CONSUMPTION, NODE 1	POWER CONSUMPTION, NODE 2	ENERGY BDUGET, NODE 1	ENERGY BDUGET, NODE 2
	A	B	C	D				
1	12	12	12	12	1.75	1.75	11470	11470
2	12	12	12	12	1.75	1.75	11470	5500
3	12	12	12	12	1.75	0.9	11470	11470
4	12	12	12	12	1.75	0.9	11470	5500
5	12	12	14	14	1.75	1.75	11470	11470
6	12	12	14	14	1.75	1.75	11470	5500
7	12	12	14	14	1.75	0.9	11470	11470
8	12	12	14	14	1.75	0.9	11470	5500

Figure 5.1: Parameter combination for the case of 2 cameras collaboration

(5.10). That leaves us with 3 feasible forms of Nash bargaining solution formula to analyse.

The Figures 5.2, 5.3, 5.4 show lifespan value for 3 approaches that evaluate load share among 2 motes in each one of 8 cases, bases on initial energy budget, individual power consumption and time it takes to extract features from the image. Different $\alpha_\mathbf{i}$-exponent is used each time:

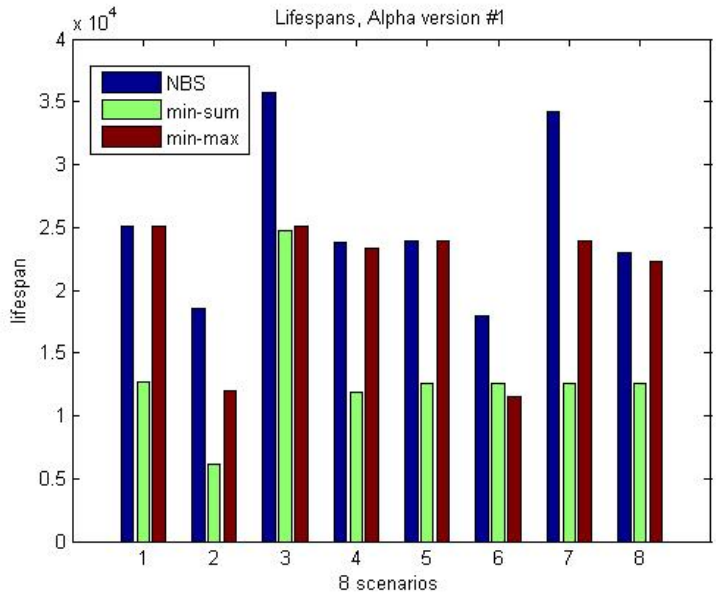

Figure 5.2: Lifespan, $\alpha_\mathbf{i}$-exponent, type 1

The Figure 5.5 provides us with knowledge about different ways of calculating α_i-exponent. And from the test it should be clear that all three

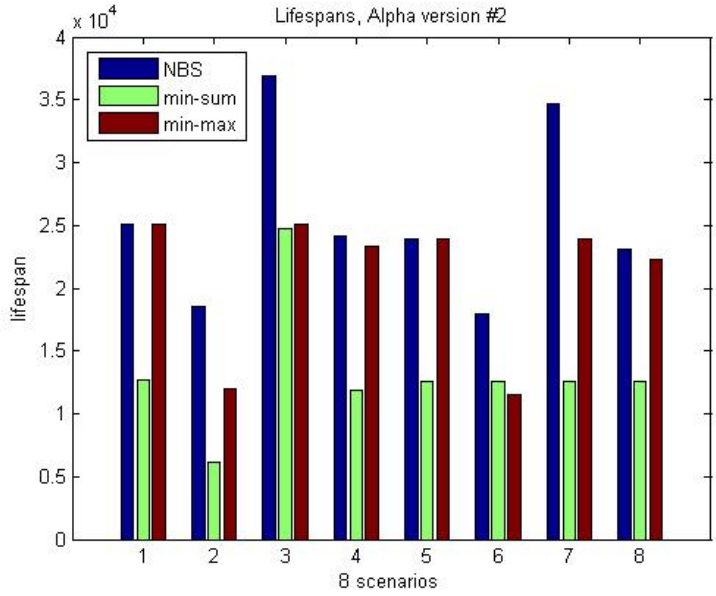

Figure 5.3: Lifespan, $\alpha_{\mathbf{i}}$-exponent, type 2

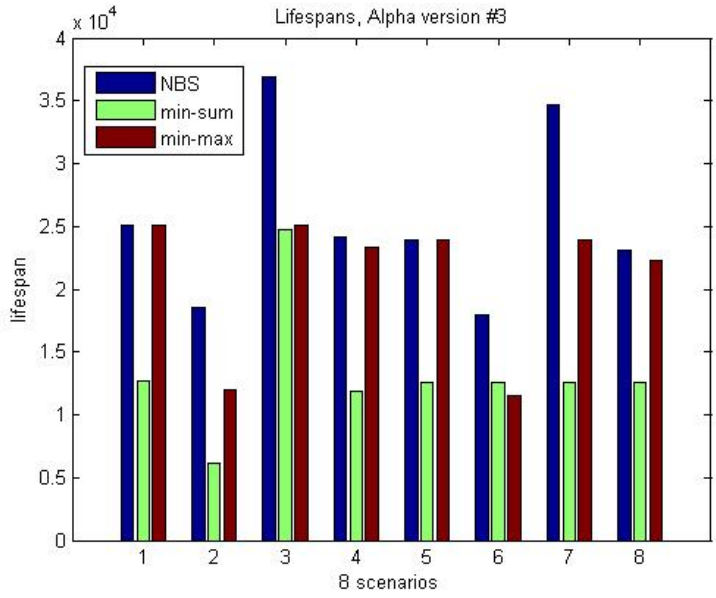

Figure 5.4: Lifespan, $\alpha_{\mathbf{i}}$-exponent, type 3

versions have similar response and to determine which one should be used in the generalized NBS formula it is necessary to refer to its complexity recalling one of the main focuses to minimize amounts of used energy, the decision

leans towards the most simple way to calculate α_i-exponent. And just from looking at the Formulas (5.11),(5.12),(5.13) that calculate α_i-exponent it is clear that the third type requires the least amount of computation and the fewest number of parameters, hence is accepted as the best fit for NBS formula.

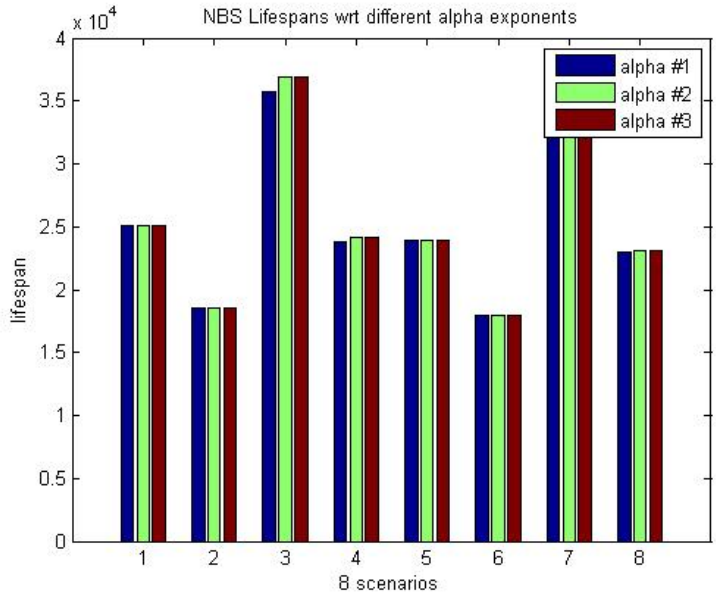

Figure 5.5: 3 α_i-exponent similarity

Regarding the Figure 5.6 that demonstrates quite clearly convenience of collaboration and load sharing when it comes to energy consumption issues and ways to encounter them. We may notice obvious advantage of using generalized NBS formula and that it brings the highest gain.

Min-sum and min-max formulas, extended versions The next stage is to go through the same analysis considering the same factors but adding one important update. In the previous stage min-sum and min-max formulas of image sharing approach have not taken into account residual energy while calculating a share, whereas generalized NBS always considers amount of residual energy by including it into α_i-exponent estimation, that makes it more responsive in terms of share assignment. Extended formula for min-

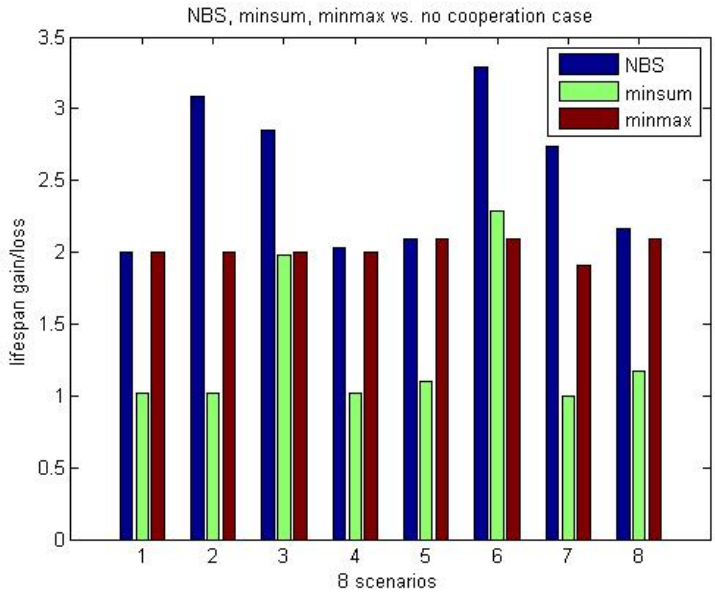

Figure 5.6: Gain in lifespan with respect to no collaboration

sum approach looks as following

$$max \sum_{x} [(E_{r1} - (Ax + B)P_{c1}) + (E_{r2} - (C(1 - x) + D)P_{c1}] \tag{5.19}$$

The idea behind the formula is to share image between the motes in a way that maximizes residual energy for entire system (summation of 2 motes' residual energies).
The same logic is used to extend min-max formula, that now is converted into maximization of minimal value of residual energy among motes participating in image sharing.

$$max[min((E_{res1} - (Ax + B)P_{c1}), (E_{res2} - (C(1 - x) + D)P_{c2})] \tag{5.20}$$

The Figures 5.7, 5.8, 5.9 give an idea about performance of all the three image sharing approaches, considering extended formulas of max-sum and max-min. As before we consider 8 possible cases and regarding 3 different versions of α_i-exponent.

The Figure 5.10 displays gain in terms of lifespan we obtain, using ex-

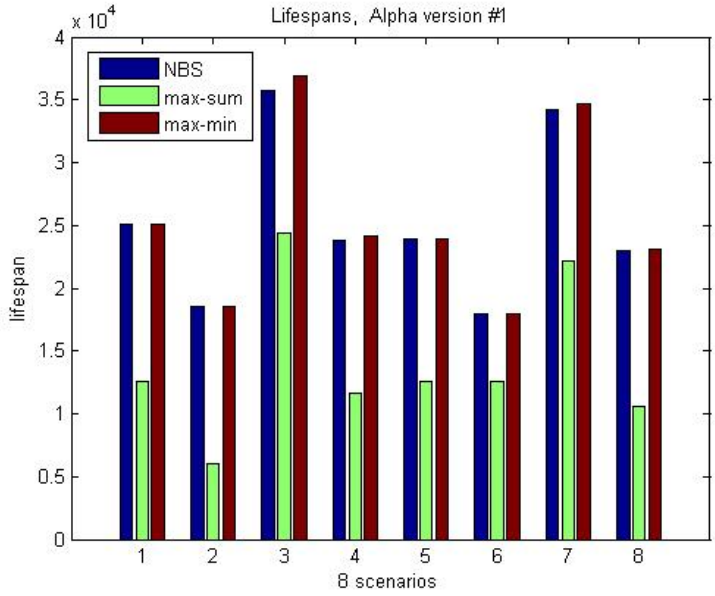

Figure 5.7: Lifespan, α_i-exponent, type 1

tended formulas of min-sum and min-max approaches:

And finally Figure 5.11 displays how lifespan gain has changed once we start using enhanced formulas max-sum and max-min for image sharing.

Concurrently we also can take a closer look at how shares are assigned in the most unlucky case (that is the one that leads to the smallest lifespan value) and the most fortunate case, when sensor network lifespan value is the largest. As example there was taken a case where we have 2 motes with the same hardware and software characteristics, same power consumption parameter but different starting energy budget (case 2), see Figure 5.12, 5.13.

Three motes collaboration

In this section performance analysis continues and we include into observation the third mote and assume all three cameras have a complete overlap of view.

It is important to notice that that performance of Nash bargaining solution is always better when simple (initially assumed) formulas if min-max and

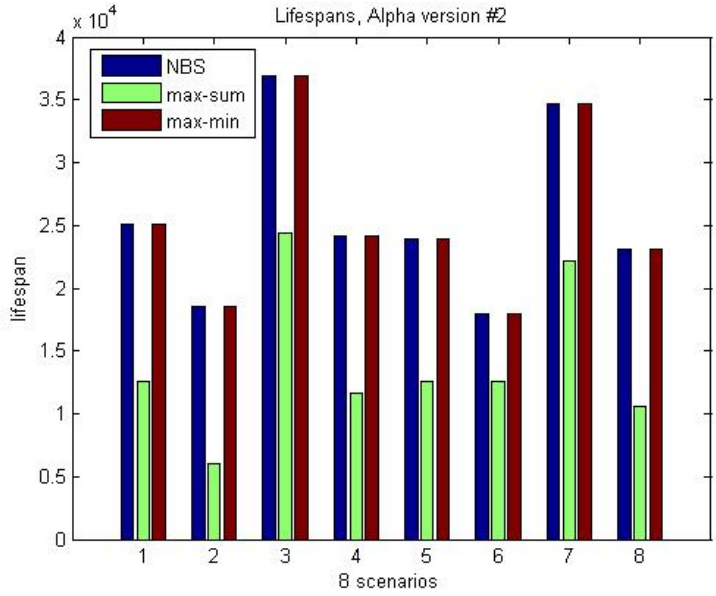

Figure 5.8: Lifespan, α_i-exponent, type 2

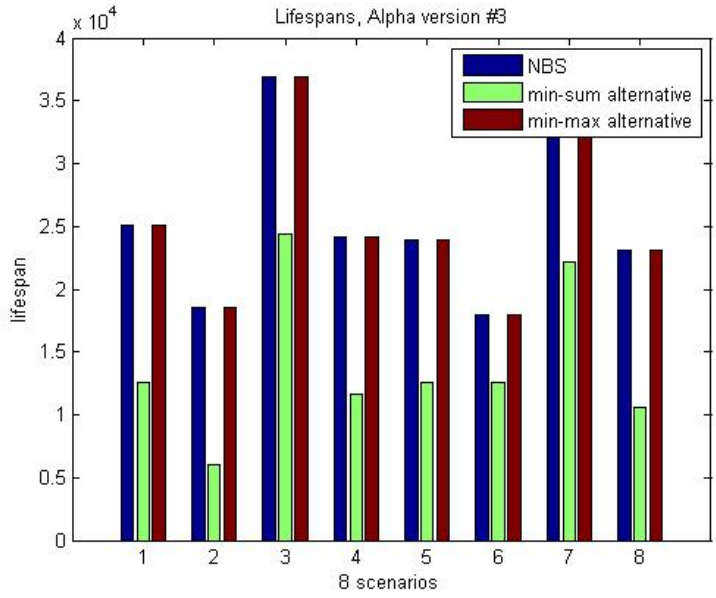

Figure 5.9: Lifespan, α_i-exponent, type 3

min-sum are described. It is true also for the case with three cameras, and the same tendency as for 2 motes case is valid in terms of performance of max-sum and max-min formulas. Max-min approach improves its perfor-

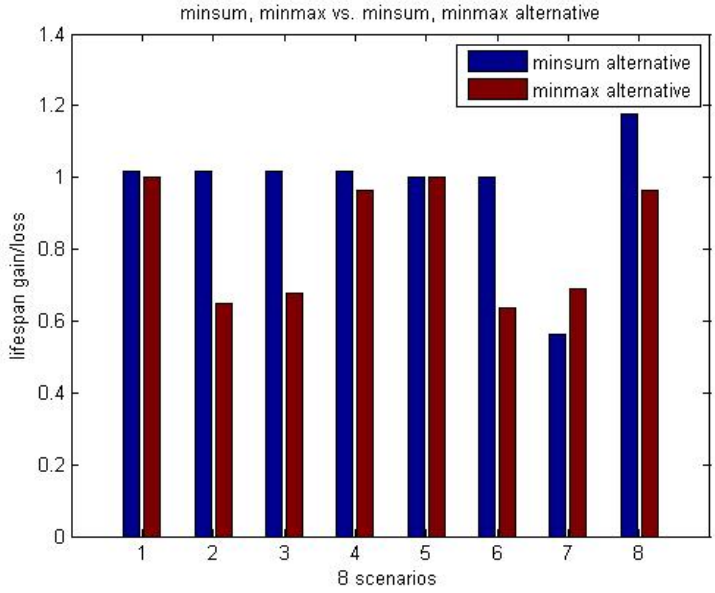

Figure 5.10: Lifespan improvement adopting max-sum and max-min

mance significantly.

For three motes there are more factors we can combine together, therefore 52 distinct performance cases are adopted and explored. But for the sake of visibility and legibility the diagrams and figures only include 8 cases out of 52, according to the Table 5.14.

For instance, if we take a look at case number 19, we see that in this performance evaluation case all three nodes have the same initial energy budget, but one of the nodes has different power consumption, and different time it takes to process the image.

There can be found standard performance Figures 5.15, 5.16, 5.17, that demonstrate realization of three processing sharing algorithms regarding two α_i-exponent. Comparison of Nash bargaining solution and max-sum and max-min formulas are displayed. One of the versions of α_i-exponent was overlooked due to its poor performance contribution as well as its complexity.

The Figure 5.18 gives an idea what improvement takes place when instead

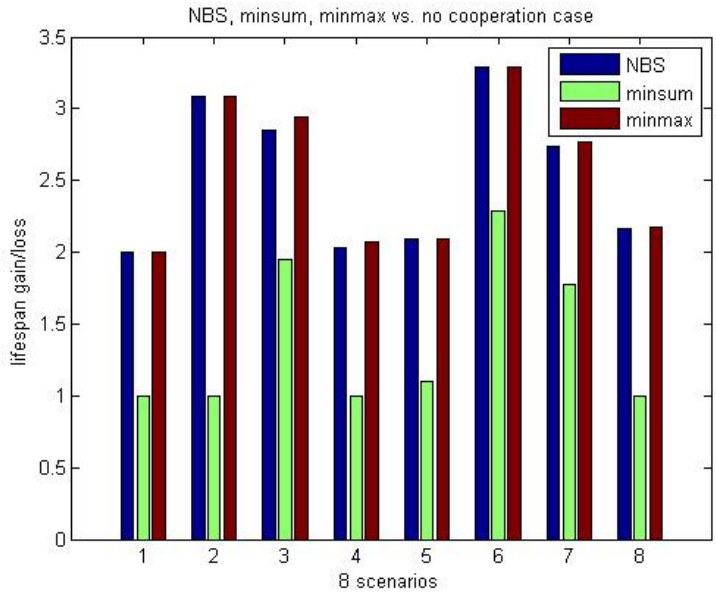

Figure 5.11: Gain in lifespan with respect to no collaboration

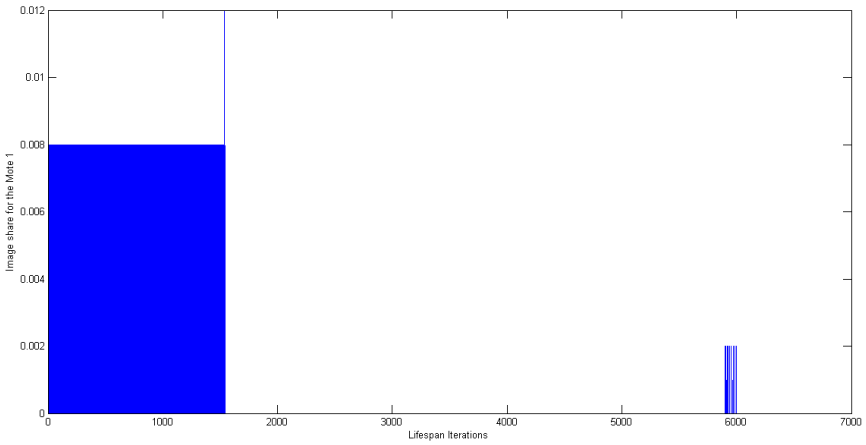

Figure 5.12: Image share of Mote 1, max-sum approach, poor performance

of plain extended versions of max-min and max-sum are used.

Taking a closer look at how image shares are assigned in the most unlucky case (that is the one that leads to the smallest lifespan value) and the most fortunate case, when lifespan value is the largest.

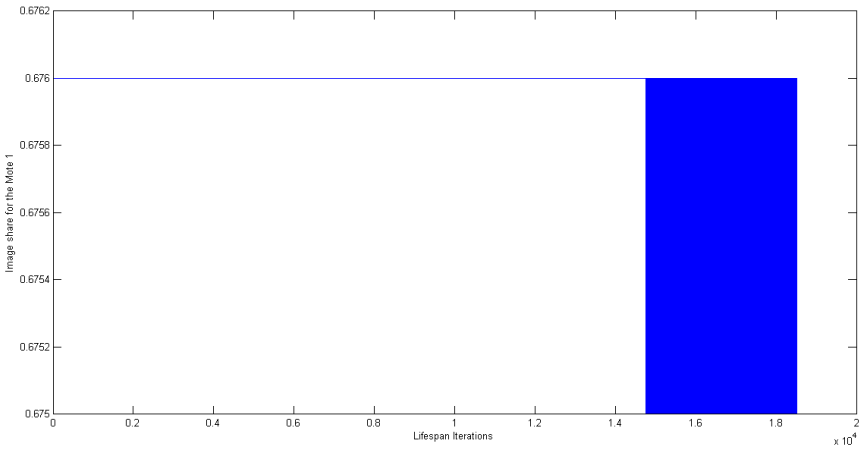

Figure 5.13: Image share of Mote 1, NBS approach, fortunate performance

#	COEFFICIENTS, NODE 1		COEFFICIENTS, NODE 2		COEFFICIENTS, NODE 3		POWER CONSUMPTION, NODE 1	POWER CONSUMPTION, NODE 2	POWER CONSUMPTION, NODE 3	ENERGY BDUGET, NODE 1	ENERGY BDUGET, NODE 2	ENERGY BDUGET, NODE 3
	A	B	C	D	E	F						
1	8	8	8	8	8	8	1.75	1.75	1.75	11470	11470	11470
7	8	8	8	8	8	8	1.75	1.75	0.9	11470	5500	5500
13	8	8	8	8	8	8	0.9	0.9	0.9	11470	11470	11470
19	8	8	8	8	12	12	1.75	1.75	0.9	11470	11470	11470
25	8	8	8	8	12	12	0.9	0.9	0.9	11470	11470	11470
31	8	8	12	12	12	12	1.75	1.75	0.9	11470	11470	11470
37	8	8	12	12	12	12	0.9	0.9	0.9	11470	11470	11470
52	12	12	12	12	12	12	0.9	0.9	0.9	5500	5500	5500

Figure 5.14: Selected combinations of parameters, 3 camera nodes collaborating

To demonstrate poor performance we observe case 4, where 3 motes has the same hardware and software characteristics and just one of the motes has different starting energy budget, higher than 2 other motes, see Figures 5.19, 5.20, 5.21, 5.22.
The case 34, where all 3 motes has same initial energy budget, but one of the modes uses different type of software and has higher power consumption parameter. Max-min approach shows excellent performance, see Figures 5.23, 5.24.

Important notice The decision to update the existing Formulas (5.15) and (5.16), and change the concept of the approach from minimizing spent energy to maximizing residual energy triggered performance improvement and make the approaches more sensitive in terms of image share assignment.

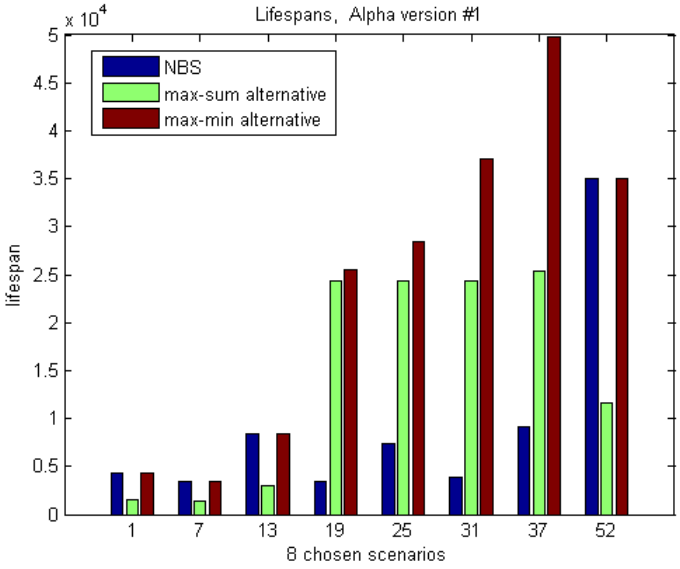

Figure 5.15: Lifespan, α_i-exponent, type 1

Referring to the diagrams we can observe that the performance max-min now reaches the same lifespan value as NBS, and in some cases even larger value. The lifespan for max-sum has improved as well but not as noticeably.
I now can establish that both NBS and max-min algorithms are feasible to implement as a strategy for visual sensor network. However max-min algorithm is a centralized method, meaning that a node is not able to make the decision on the image share to process locally, and a coordinator instance is required to calculate and assign the shares. The prerequisite to implement the coordinator node implies extra cost in terms of energy consumption and also require extra monetary resources. Whereas NBS is a distributed type of the algorithm, implicating that once all the needed parameters are obtained each node is capable of calculating its share locally, and the value will be coherent with the values of the camera nodes that are involved in cooperation. Hence the strategy that implements NBS does not require any topology changes or hardware upgrade, which is an important property, that leads to vital for visual sensor network resource preservation. Hence considering two image share approaches with analogous performance, NBS prevails max-min approach by being distributed and less resource-consuming.

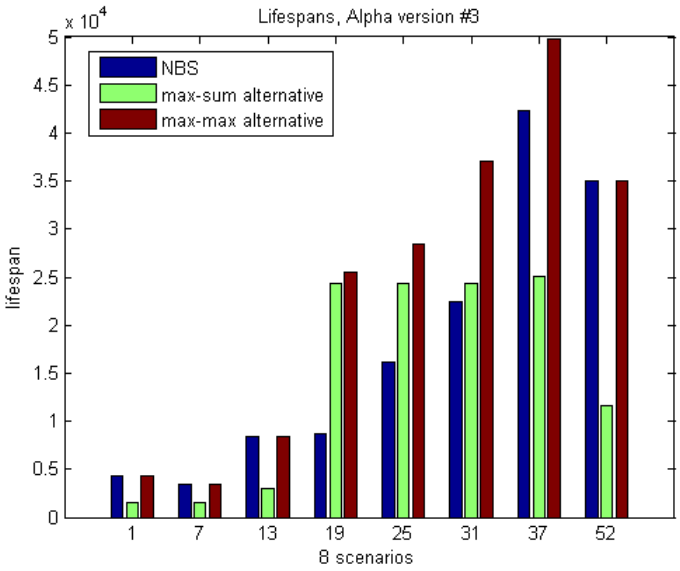

Figure 5.16: Lifespan, α_i-exponent, type 3

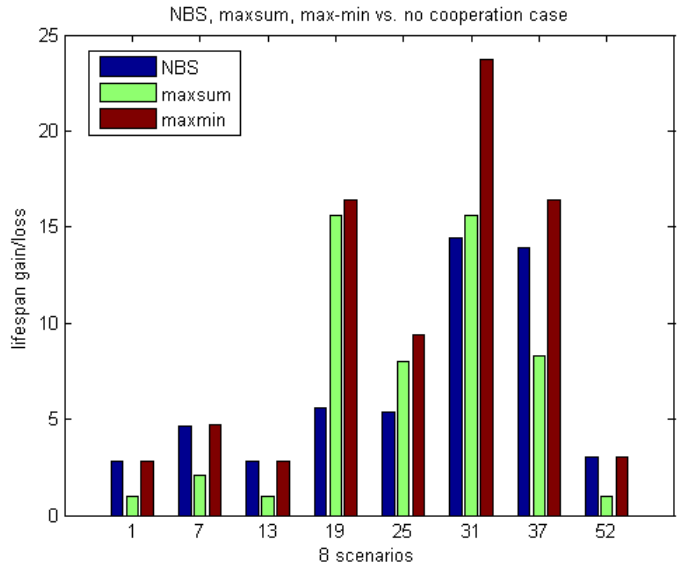

Figure 5.17: Gain in lifespan with respect to no collaboration

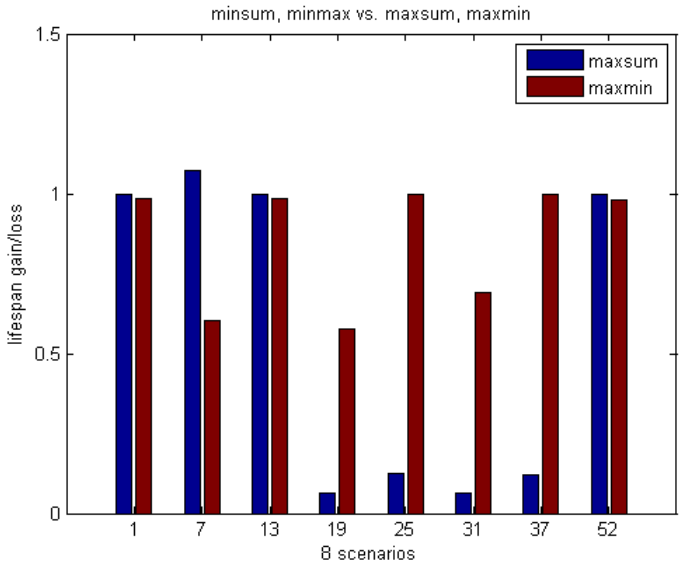

Figure 5.18: Lifespan improvement adopting max-sum and max-min

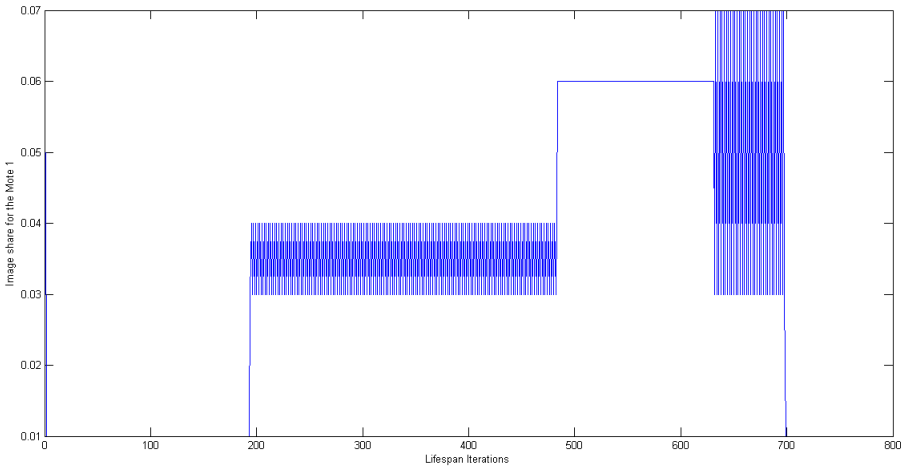

Figure 5.19: Image share of mote 1 through lifespan iterations, max-sum, poor performance

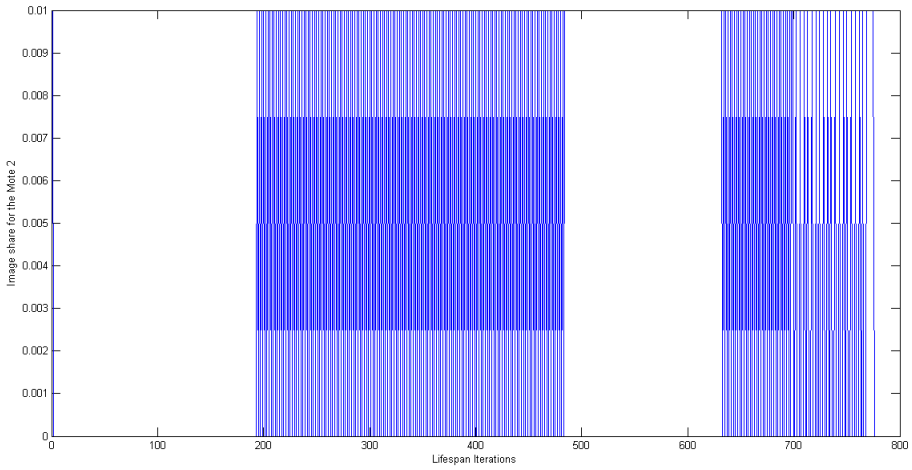

Figure 5.20: Image share of mote 2 through lifespan iterations, max-sum, poor performance

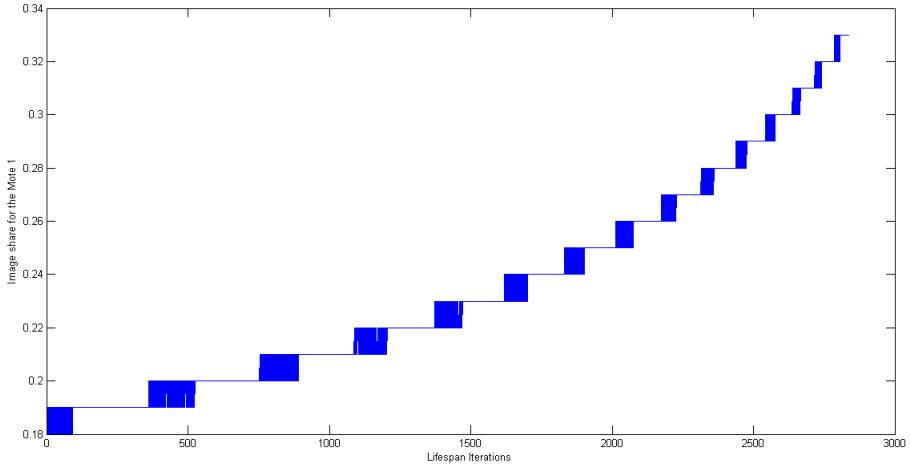

Figure 5.21: Image share of mote 1 through lifespan iterations, NBS, fortunate performance

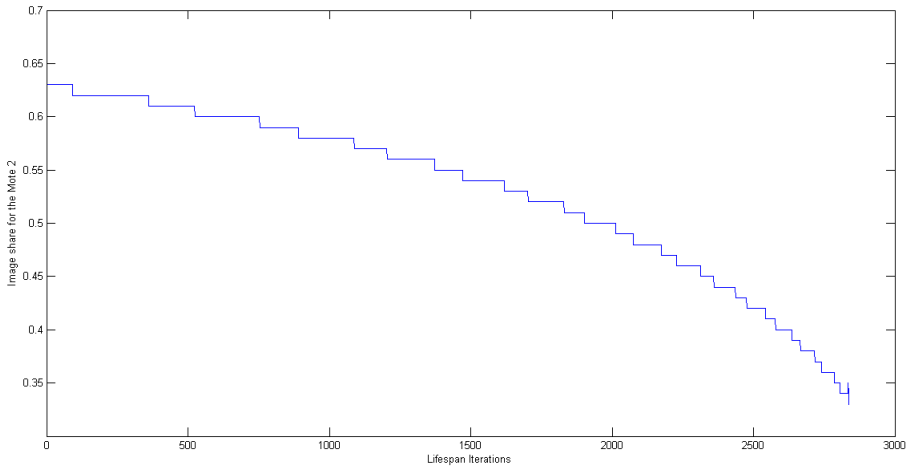

Figure 5.22: Image share of Mote 2 through lifespan iterations, NBS, fortunate performance

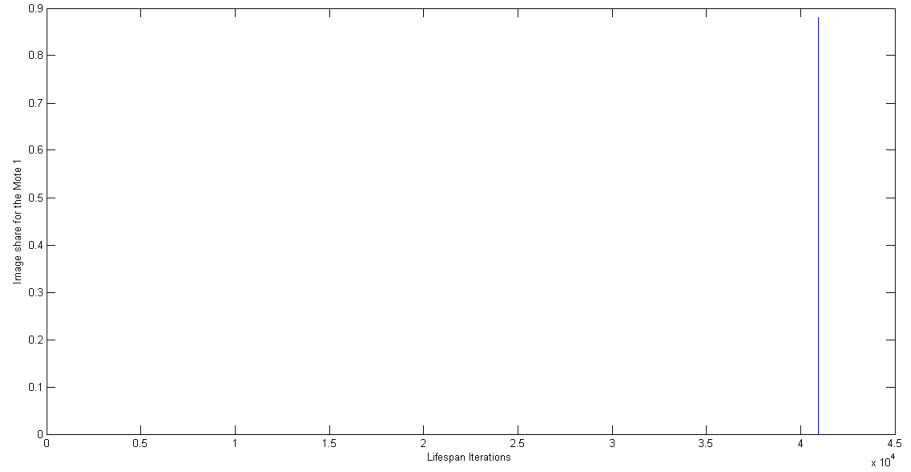

Figure 5.23: Image share of mote 1 through lifespan iterations, max-min, excellent performance

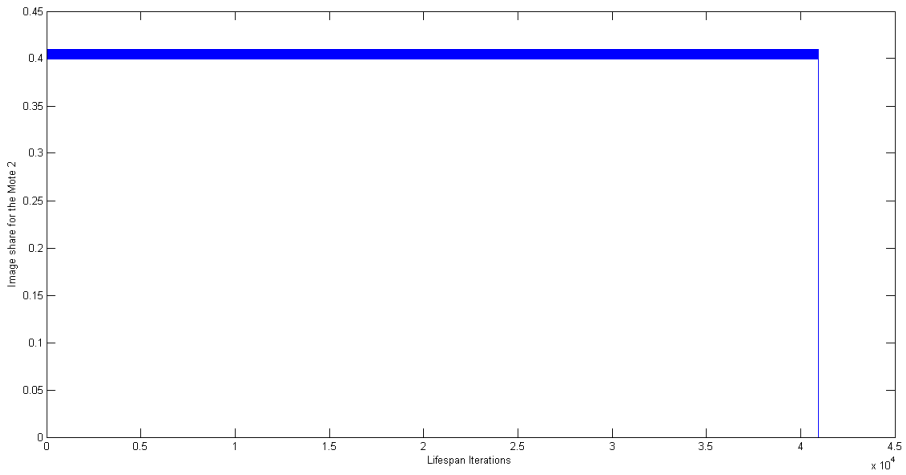

Figure 5.24: Image share of mote 2 through lifespan iterations, max-min, excellent performance

5.3.2 Partial view overlap

This part of analysis provides results regarding scenario where motes that are involved into the collaboration only have partial field of view overlap. This fact leads to an offset that should be embedded into the algorithm. This offset denotes a portion of the image that each mote is required to process alone and a portion that it can bargain on.

Field of view detection techniques

Prior to NBS-driven node collaboration an overlapped field of view should be calculated for all camera nodes in the sensor network. There are two main approaches that describe how motes define overlapped field of view.

Approach 1 Defining an overlapped field of view for 2 cameras is based on computational geometry technique called *Voronoi tessellation* [16]. In two dimensions a Voronoi tessellation of a set of points is the partition of the plane into convex polygons such that all the polygons contain a single generating point and all points within a polygon are the closest to the corresponding generating point. Each reference point in the camera is contained within a cell, with all points in a cell closest to the corresponding reference point. Given a skewed distribution of reference points, it follows that densely situated points will be contained within smaller polygons, and sparsely situated points in larger polygons. Since the size of each polygon is related to the density of the points in the neighborhood, it can be used as an approximation of the area represented by each point. Voronoi tessellations can be extended to points in three dimensions, with each point contained with a 3D cell instead of a polygon. Using Voronoi tessellation, each reference point is assigned a weight that is approximately equal to volume of the cell that it lies in. Note that Voronoi tessellation requires the coordinates of reference points in order to partition the viewable region into cells or polygons.

To be able to determine the region of overlap for each camera, we first assume

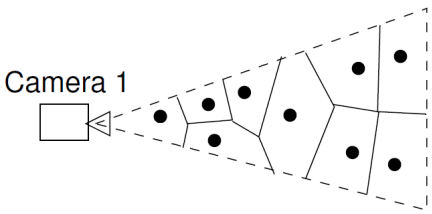

Figure 5.25: k-overlap estimation with distribution of reference points, weighted approximation

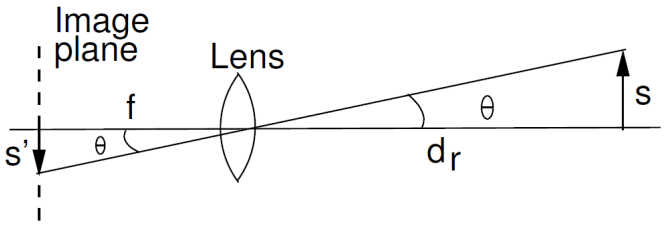

Figure 5.26: Object and image relation in 3D

a reference object placed at randomly chosen locations. Using this points: first Voronoi tessellation of the viewing area is obtained for each camera. A Voronoi tessellation of a set of points is the partitioning of the plane into convex polygons such that all polygons contain a single generating point and all points within a polygon are closest to the corresponding generating point. Figure 5.25 shows a skewed distribution of reference points in the 2D viewing area of a camera and the corresponding Voronoi tessellation. Each

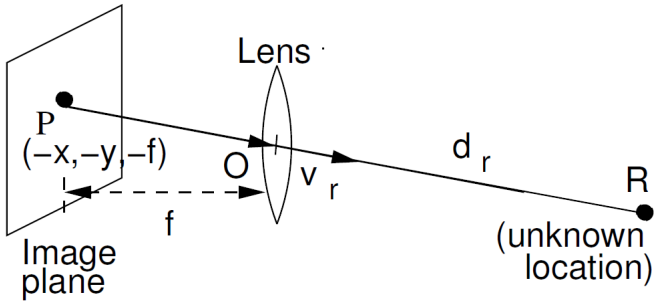

Figure 5.27: Estimation of reference point location

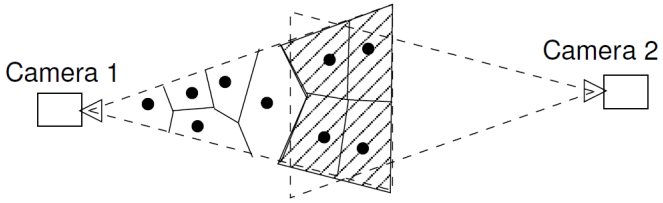

Figure 5.28: Region of overlap using Voronoi tessellation

reference point in the camera is contained within a cell, with all points in a cell closest to the corresponding reference point. The region of overlap for any two cameras C_i and C_j is simply the union of cells containing all reference points simultaneously visible to the both cameras. Figure 5.28 shows the Voronoi tessellation of the 2D viewing region of camera 1, the reference points viewable by cameras 1 and 2, and the approximate region of overlap (shaded region) for (C_1, C_2). Thus, our approximate tessellation is used to determine region of overlap.

Estimating reference point locations. The tessellation process that lies in the core of region of overlap detection, requires the location of reference points. Since no infrastructure is available, there still exists a technique to estimate this location using principles of optics. A key insight is that if each camera can determine the coordinates of visible reference points relative to itself, then tessellation is feasible and absolute coordinates are not required. Assuming the origin lies within the center of the lens, the relative coordinates of a point are defined as (d_r, v_r) where d_r is its distance from the origin, and v_r is a vector from the origin in the direction of the reference point that defines its orientation in 3D space.

It is assumed that the size of the reference object is known a priori and it is s. The focal length f is also known. Then the camera first estimates the size of the image projected by the object - this can be done by computing the bounding box around image, determining the size in pixels and using the size of the CMOS sensor to determine the size of those many pixels. If s' denotes

the size of the image projected by the reference object on the camera, then from Figure 5.26 the following condition holds

$$tan(\theta) = \frac{s}{d_r} = \frac{s'}{f} \tag{5.21}$$

Since *s,s',f* are known d_r can be computed.
Next to determine the orientation of the reference point relative to the camera, assume that the reference object projects an image at pixel coordinates *(x,y)* on the image plane of the camera. Then the vector v_r has the same orientation as the vector that joins the centroid of the image to center of the lens (i.e. origin). As shown in Figure 5.27, the vector *PO=(x,y,f)* has the same orientation as v_r, where O is the origin and P is the centroid of the image with coordinates *(-x,-y,-f)*. Since *(x,y)* can be determined by processing the image and f is known the relative orientation of the reference point can be determined.

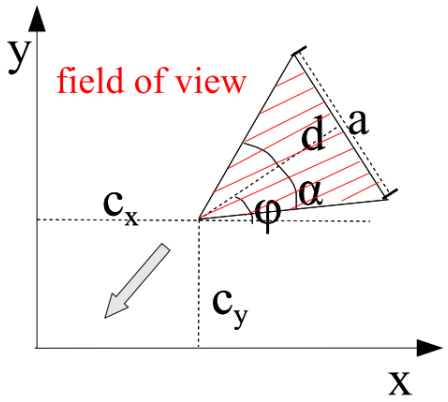

Figure 5.29: Field of view

Approach 2 The field-of-view of a camera is described by a triangle as shown in Figure 5.29. The parameters of this triangle are calculated given the (intrinsic) camera parameters, such as CCD sensor size, lens focus and angle-of-view using well known geometric relations. Once a the field-of-view

is determined by a triangle it is possible to define the area covered by a camera at position (c_x,c_y) and with pose φ by three linear constraints. In order to obtain the area visible to both camera nodes the first step is to translate location coordinates in such a way that directional sensor (mote) are placed along the x-axis of the coordinate system:

$$y' = y - c_y \tag{5.22}$$

Field of view of a directional sensor can be calculated according to the formula:

$$FOV_i = d * 2 * arctan(\frac{AOV_i}{2}) \quad [m] \tag{5.23}$$

Where d is a distance a sensor is able to reach with FOV and AOV (denoted α) is the angle of view, provided by technical specification of the directional sensor.
To be able to visualize the case and understand basic working principles the following assumptions are made:

1. Rectangular room. A room where directional sensors are placed
2. Sides of the room are known
3. 2 directional sensors, that are placed along the same side of the room
4. x-coordinates (placement along the side) are known
5. φ_1, φ_2, pose of mote 1 and 2 respectively

The area that a mote captures is projected on the opposite side, considering we have acceptable critical direction angle. Critical direction angle permits two motes FOVs have an observed area in common. We need to determine the area each mote captures, in particular the borders of this area. To find those borders observe another triangle defined by bisection line, perpendicular line to the opposite side (where we expect the projection), see Figure 5.30. We obtain right-angled triangle. Knowing the x-coordinates of the sensor the perpendicular line intersecting with opposite side would give

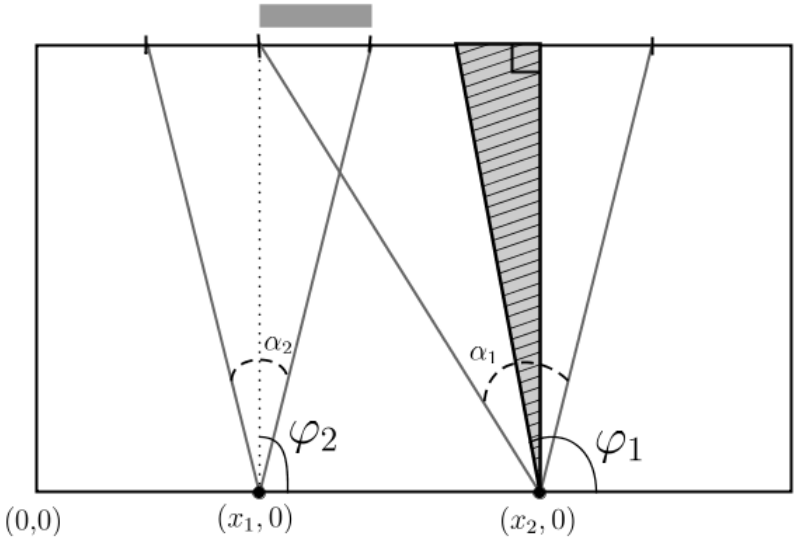

Figure 5.30: Detecting overlapped field of view

us the same x-coordinate value and cathetus would represent the offset from that point and the exact location where the bisection line is pointing out, meaning from this point FOV/2, see Formula (5.23) to the right and FOV/2 to the left the borders can be found. Then we find the intersecting part and calculate the percentage from the entire image size that is located in the intersection and include this parameter to calculate Nash bargaining solution in case of partial overlap.

At this point in the analysis the fraction of image where an overlap takes place is represented by a number between zero and one. This number or its complement is the offset that is introduces in all the NBS-related formulas and has an essential influence on further bargaining concerning image share among camera nodes in collaboration. Every note that is involved in bargaining sets a portion of image that it is inevitably forced to process, this the part that is not included in the overlapped FOV. Every mote has its own image capturing characteristics, the most relevant in this case is angle of view, that basically describes how wide is the acquired image. Starting from an image portion that a note needs to process by itself it may be interested in smaller or larger portion of the image to bargain for. In the Figure 5.31

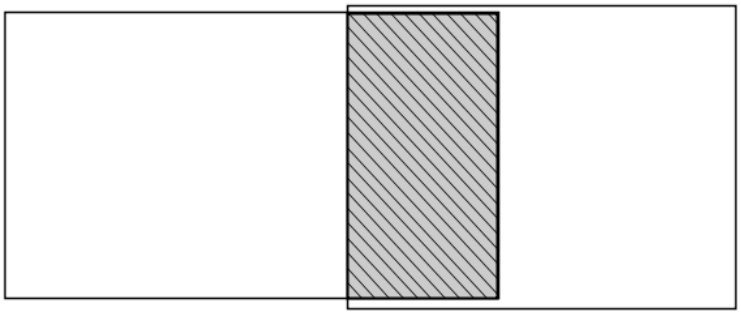

Figure 5.31: Overlapped field of view, 2 motes

it is shown what portion each mote has to process by itself and what part is a subject for further bargaining. Keeping in mind the performance analysis and its scope to save maximum amount of energy the offset changes future bargaining outcome.

Two motes collaboration

The Figures 5.32, 5.33 display the usual case of performance analysis of three image sharing approaches with the first type of the α_i-exponent and extended max-min and max-sum formulas used.

Three motes collaboration

The very same logic is applied for the case of partial overlap of views for three motes.

The Figures 5.35, 5.36 display the usual case of performance of three processing offload approaches with the third type of the α_i-exponent and extended max-min and max-sum formulas used. And the performance is consequently observed in eight chosen cases that are produced by combining steering parameters.
Once the NBS strategy is defined within visual sensor network the next step

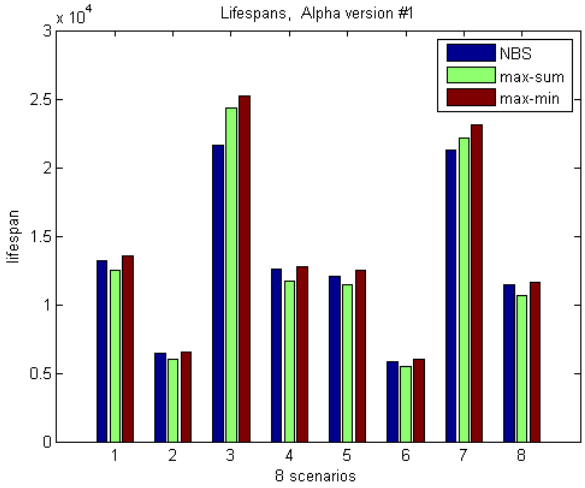

Figure 5.32: Lifespan, $\alpha_{\mathbf{i}}$-exponent, type 1

is to implement a communication protocol that ensures the stable parameter exchange and communication that completes the NBS implementation and allows to fully exploit the NBS concept with the scope of energy saving. The next chapter contains suggested description of the communication protocol.

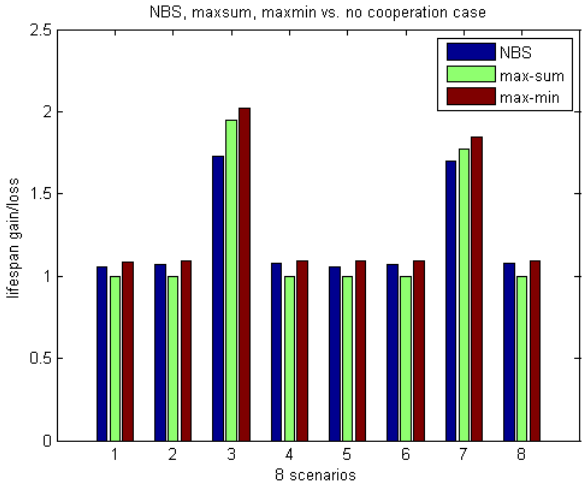

Figure 5.33: Gain in lifespan with respect to no collaboration

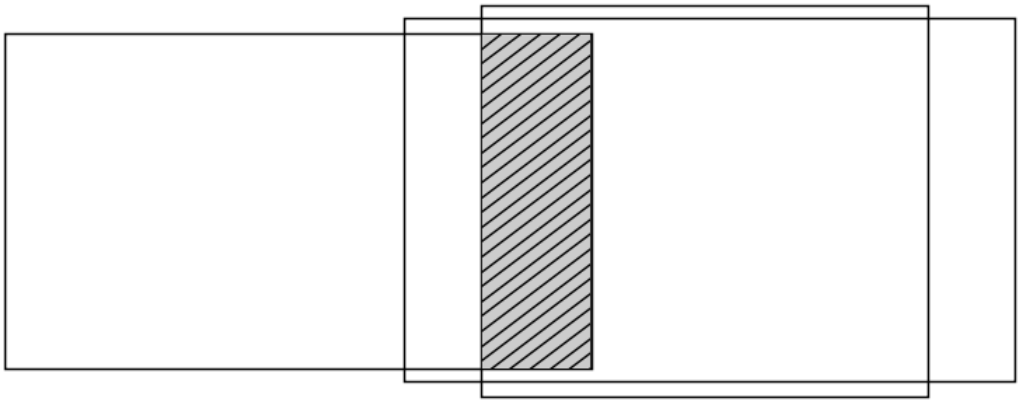

Figure 5.34: Overlapped field of view, 3 motes

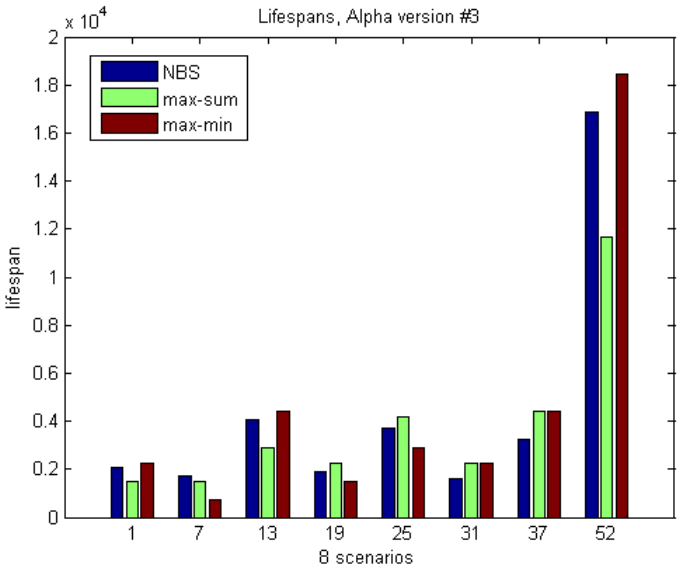

Figure 5.35: Lifespan, α_i-exponent, type 3

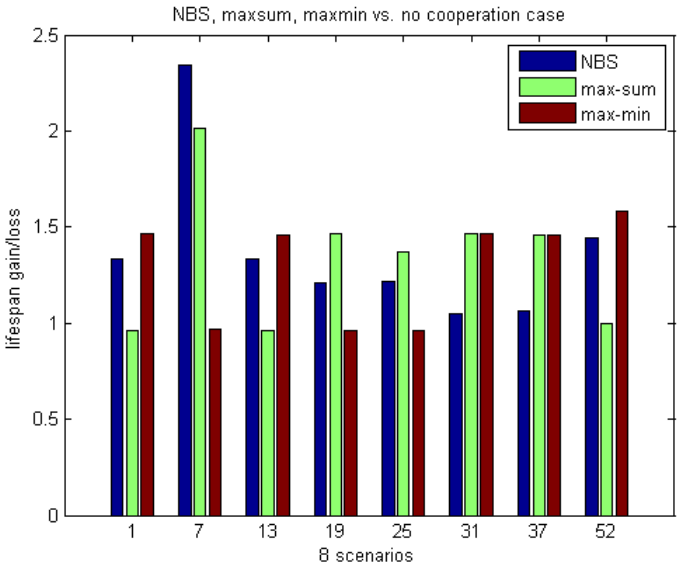

Figure 5.36: Gain in lifespan with respect to no collaboration

Chapter 6

Communication protocol

6.1 Components

A communication protocol is a system of digital rules for data exchange within or between nodes of the network. The protocol described in this Chapter empowers NBS-based strategy to share image for local processing in visual sensor network.
The protocol is executed by two processes that are conducted in parallel. The first one is exchange of the parameters (including residual energy budget update). Beacon frame in broadcast is used for parameter exchange process. Based on the received beacon frames from neighbour nodes, each camera creates a table. The table contains information relevant for establishing nodes collaboration.
The beacon frame has structure, that is shown in Figure 6.1.
Parameters in the payload of the beacon frame allow to calculate overlapped field of view for each node and add to the table accordingly.

Only the addresses of the nodes whose overlapped field of view (with the node under current consideration) is determined to be greater than 0 is placed in the table. Beacon frames also keep all the parameters that are needed to calculate NBS. It is crucial for the expected saved energy estimation. The value of expected saved energy is the value that is obtained in case the nodes

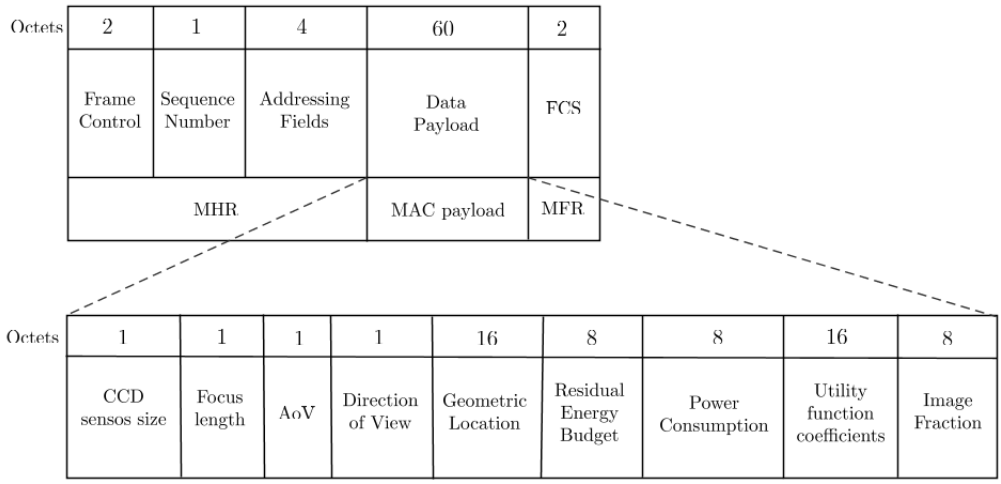

Figure 6.1: Frame structure, camera and NBS parameters in payload

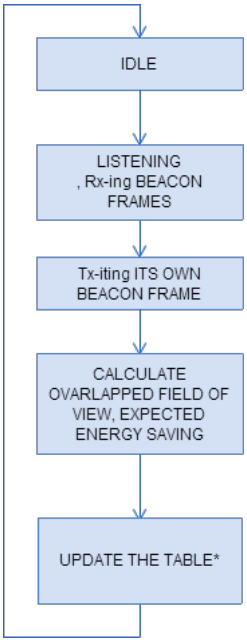

Figure 6.2: Flowchart, continuous exchange of beacon frames

decide to collaborate, and this value is also part of the table.
The table has following columns:

| camera address | overlapped FOV % | expected energy saving |

Another process that can occur simultaneously with the continuous exchange

of beacon frames is the one triggered by image acquisition request sent to a camera. In this case the following set of rules is activated.
The first step is to determine the number of entries in the table:

- If 0 - camera processes the image by itself
- If 1 - send a collaboration request to the camera address, contained in the entry
- If more than 1 - more elaborated scenario with several branches should be applied

The second step is to check the most promising collaboration possibility, the one that might bring us the biggest energy saving, we check the column of the table that contains the values of expected energy savings and choose the maximum. We take the address of the corresponding camera and send a collaboration request.

In such a way each camera potentially feasible for collaboration expresses

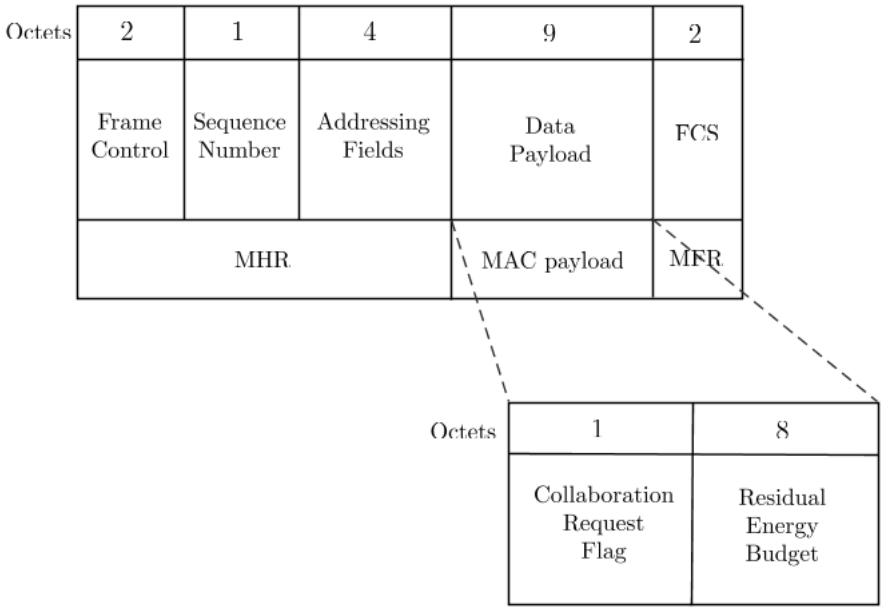

Figure 6.3: Frame with collaboration requests in payload

its preferences to the neighbouring cameras. In case we receive in response

a preference frame from camera address we chose, then a notification about establishing collaboration is broadcasted, NBS is calculated using current values of residual energy budget, the solution of NBS is share of the image that the node need to process locally.

In case when there is more than 1 entry in the table, and furthermore

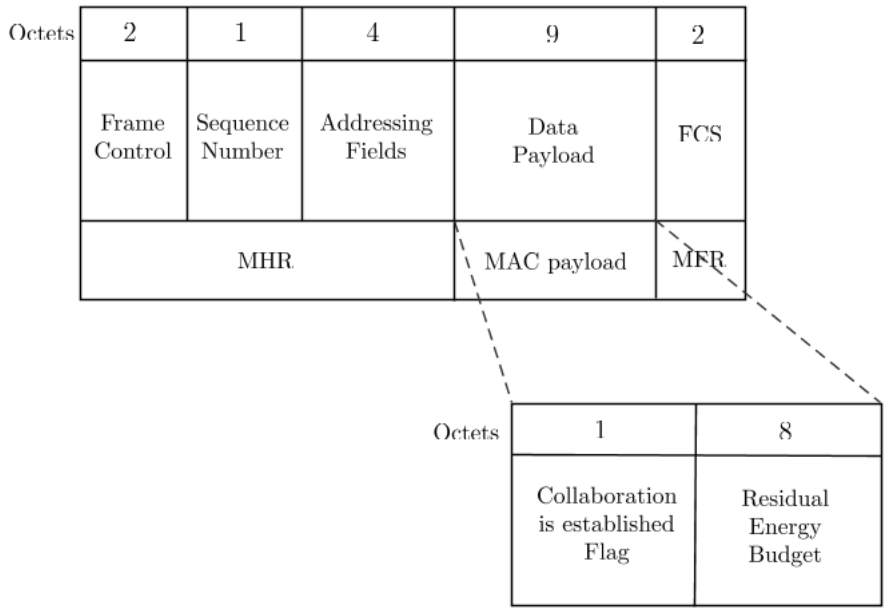

Figure 6.4: Frame structure, established collaboration notification

there is an overlap of 3 cameras then this case should be stated in the table precisely - a union of 2 camera addresses should be placed there, this can be done with a pointer, such that when the entry with the pointer is chosen the collaboration request packet will be send to both nodes' addresses and hence the preference reply should be received from both camera addresses, otherwise a mismatch is called. In such a way when no preference is received, then its the case where other 2 pair of nodes will continue collaboration and image sharing (and they exchange notification frames), or we have complete mismatch of preferences and in this case a feasible solution is to choose the second maximum value of expected energy saving from the table and start the process of establishing collaboration from the step where the packets with

preferences are broadcasted.
The Flowchart 6.5 expresses the sequence of steps taken.

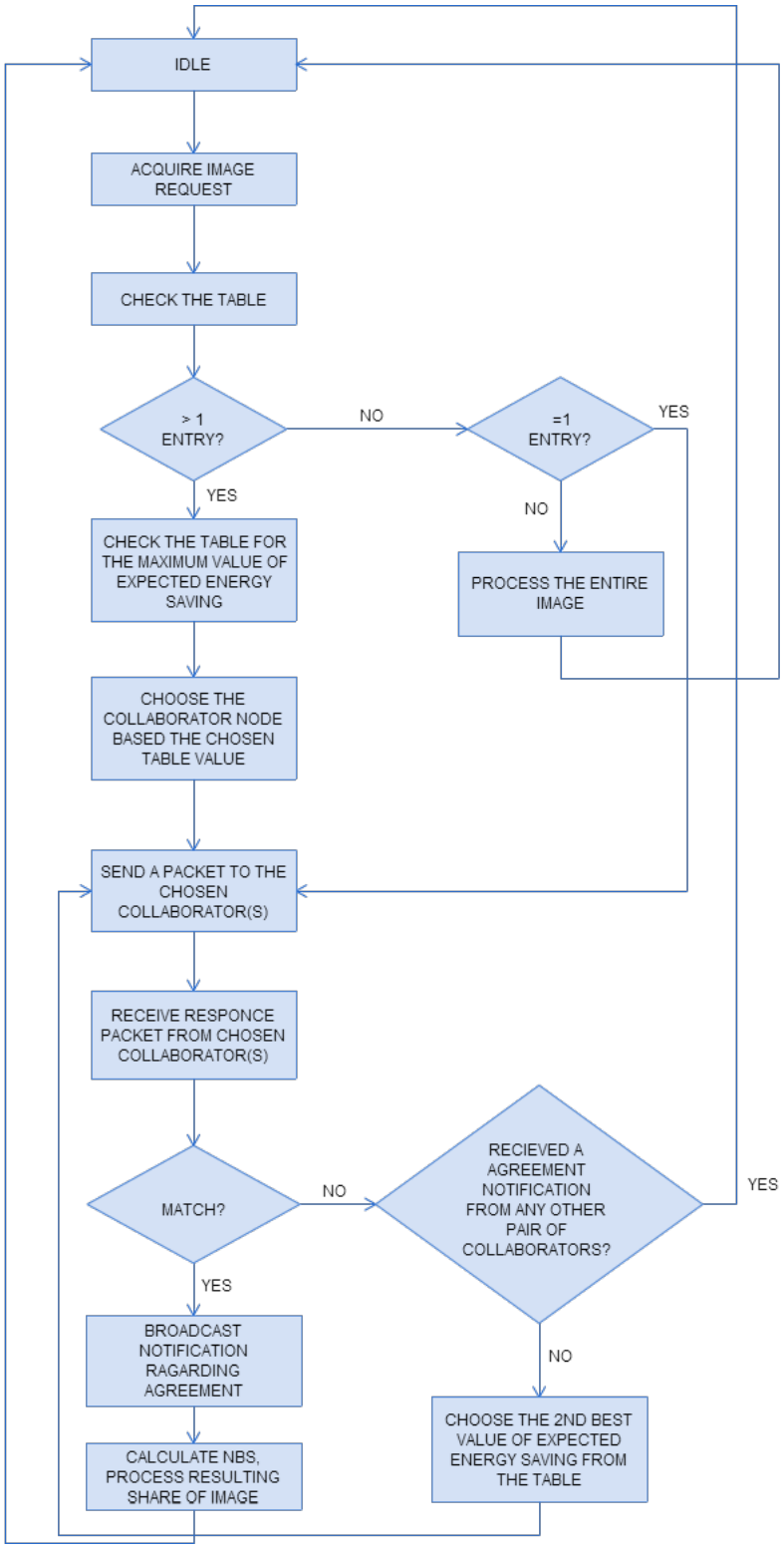

Figure 6.5: Flowchart, steps towards establishing collaboration

Communication protocol timeline is represented by the Sequence Diagram 6.6.

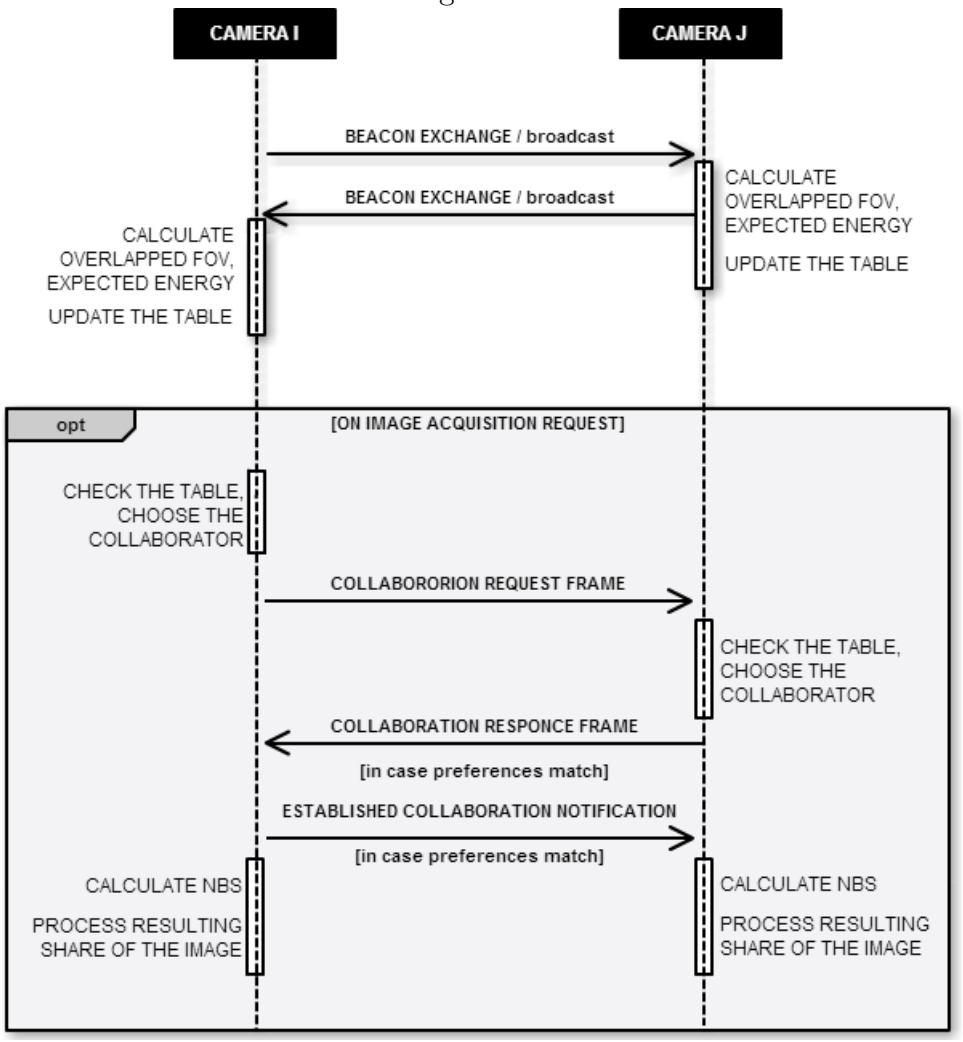

Figure 6.6: Timeline, communication protocol

6.2 Protocol energy consumption

At this point it is necessary to determine the amount of energy required by the communication protocol to operate properly. The values of energy consumption are defined experimentally, assuming the framework based on 2 wireless communication standards IEEE 802.11g and IEEE 802.15.4. Each of the standards is characterized by a power consumption, voltage parameter and typical transmission rate. The value of energy spent on communication protocol depends on the frame size that is in transmission. In this case we have 3 types of frames the protocol exploits:

- Beacon frame, size = 69 octets
- Collaboration request frame, size = 18 octets
- Collaboration notification frame, size = 18 octets

The parameters are taken from the datasheets of the chipsets:

Parameter	IEEE 802.11g	IEEE 802.15.4
Power consumption in Tx (mA)	80	17.4
Power consumption in Rx (mA)	110	18.8
Voltage (V)	1.8	3.3
Transmission rate	1 Mb/s	250 Kb/s

Table 6.1: Protocol energy consumption parameters

We also assume that number of nodes k in the network equals 10. Then using the formula to calculate transmission time:

$$T_{tx} = \frac{N}{Rate} \tag{6.1}$$

Where N is the number of bits to transmit/receive, that is the frame size expressed in bits, and *Rate* is the transmission rate, typical for each of the communication standards. Then the energy required for the beacon exchange stage is calculated according to the equation:

$$E_{beacon} = T_{tx} * (P_{tx} + (k-1) * P_{rx}) \tag{6.2}$$

And energy required for actual collaboration request/response stage:

$$E_{collab} = T_{tx} * (2 * P_{tx} + P_{rx}) \tag{6.3}$$

Total amount energy is a summation of energies required at both stages:

$$E_{total} = E_{collab} + E_{beacon} \tag{6.4}$$

And as a result spent energy values are in the range:

- for IEEE 802.11g: $E_{total} = 0.0011[J]$
- for IEEE 802.15.4: $E_{total} = 0.0014[J]$

Whereas the amount of energy required to locally process the image is on average 0.4793 [J] that implicates the difference order of approximately 400, meaning that energy required to support the protocol is negligible with respect to energy needed to process the image. That permits us to perform the analysis, disregarding the value of protocol energy consumption.

Chapter 7

Conclusions and future developments

Energy saving and longer network lifetime are key issues in visual wireless sensor networks. The solution that was proposed in this thesis work is based on Nash bargaining solution and allows to save overall energy of the network and increase its lifetime. Camera nodes with overlapped field of view collaborate and decide on a portion of image each one processes.

First I described a framework that is provided by visual sensor network, its components, parameters and constraints. Then I introduced basic principles of Nash bargaining solution, identifying parameters that were required for the implementation in the framework of visual sensor network.

One of the crucial parameters was utility function, that was defined experimentally. Experiment involved studying of visual analysis, in particular time it took to process the image at each camera node with respect to given parameters. The function that described the time from image size was linear and consequently the utility function of NBS was linear.

Once the framework was defined series of tests were performed, analysing behaviour of NBS-based algorithm to share the image to process. The analysis included two subcases: one considered a case where cameras have a completely overlapped field of view, and the second one considered a case where cameras only had a partially overlapped field of view. Each of subcases

involved either two or three camera nodes in collaboration. The obtained outcome showed that the lifetime increased by 350% in case of complete overlap and two collaborating cameras and 2300% in case of three camera nodes collaboration. In case of partial overlap - up to 250% for both two and three cameras collaborating.

During analysis stage NBS solution was compared to a case of no collaboration, alternative approach where it was required to maximize the summation of energy spent on image processing by the cameras involved in bargaining and maximize the minimum energy spent on image processing accordingly. And the latter algorithm demonstrated the outcomes in terms of lifespan being very close or in certain cases even better than suggested NBS-based algorithm. This can be commented as follows: throughout the research the standard min-max approach was enhanced by including the parameter of residual energy of each camera node. This component ensured more flexible algorithm and as a consequence longer lifetime of the network. But despite great performance results, the min-max algorithm remaines centralized, meaning requiring a cooperator node to assign and manage the participant nodes in collaboration. Whereas NBS-based algorithm is distributed and in this case each node makes a decision to collaborate or not autonomously and this is its great advantage.

Analysis followed by the communication protocol that enabled parameter exchange and collaboration process. I presented types of frames that used for each stage of collaboration, as well as energy consumption estimation, needed to understand the protocol was light-weighted and the main energy consumed through local processing. And also flowcharts describing order of the requests and according frames.

As a possible next step it could be useful to construct a test-bed, that emulates the behaviour of suggested NBS-solution and communication protocol in a framework where there are more than three nodes or the number and location of nodes is periodically changing. The algorithm may also be explored for more feature extraction algorithms, when for this research there were explored only two examples of such. As well as for more types of hardware and camera models, that would make the research comprehensive. And

ultimately the suggested NBS-based algorithm should find implementation in a real life sensor network, followed by an extensive performance evaluation. In such a way the thesis may continue its development in several directions.

Bibliography

[1] W. Baoyun, "Review on internet of things," *Journal of Electronic Measurement and Instrument*, vol. 23, no. 12, pp. 1–7, 2009.

[2] Politecnico di Milano, "Greeneyes project." `http://greeneyesproject.eu/`, 2014.

[3] S. Soro and W. R. Heinzelman, "A survey of visual sensor networks," *Advances in Multimedia*, 2009.

[4] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[5] W. Hu, T. Tan, L. Wang, and S. J. Maybank, "A survey on visual surveillance of object motion and behaviors," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 34, no. 3, pp. 334–352, 2004.

[6] R. Nelson, "Alert - automated local evaluation in real time." `http://www.alertsystems.org/`, 2012.

[7] R. Iacono, "Collaborative object recognition in a wireless visual sensor network," *Master Thesis*, 2012.

[8] H. Bay, T. Tuytelaars, and L. J. V. Gool, "Surf: Speeded up robust features," in *9th European Conference on Computer Vision (1)*, pp. 404–417, 2006.

[9] S. Leutenegger, M. Chli, and R. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *13th International Conference on Computer Vision*, pp. 2548–2555, 2011.

[10] M. Maschler, S. Zamir, and E. Solan, *Game Theory*. Cambridge University Press, 2013.

[11] K. Ma, Q. Han, C. Chen, and X. Guan, "Bandwidth allocation for cooperative relay networks based on nash bargaining solution," *International Journal of Communication Systems*, vol. 25, no. 8, pp. 1044–1058, 2012.

[12] G. Zhang, H. Zhang, L. Zhao, W. Wang, and L. Cong, "Fair resource sharing for cooperative relay networks using nash bargaining solutions," *IEEE Communications Letters*, vol. 13, no. 6, pp. 381–383, 2009.

[13] K. Pandremmenou, L. P. Kondi, and K. E. Parsopoulos, "Optimal power allocation and joint source-channel coding for wireless ds-cdma visual sensor networks using the nash bargaining solution," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 2340–2343, 2011.

[14] H.-Y. Shi, W.-L. Wang, N.-M. Kwok, and S.-Y. Chen, "Game theory for wireless sensor networks: a survey," *Sensors*, vol. 12, no. 7, pp. 9055–9097, 2012.

[15] Z. Han, *Game Theory in Wireless and Communication Networks: Theory, Models, and Applications*. Cambridge books online, Cambridge University Press, 2012.

[16] P. Kulkarni, P. J. Shenoy, and D. Ganesan, "Approximate initialization of camera sensor networks," in *4th European Conference on Wireless Sensor Networks*, pp. 67–82, 2007.