Trabajo Fin de Grado
Grado en Ingeniería de Tecnologías Industriales

Robot coordination to create collaborative panorama images

Autor: María Teresa Araúz Pisón
Tutor: José María Maestre Torreblanca

Dep. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2016

Trabajo Fin de Grado
Grado en Ingeniería de Tecnologías Industriales

Robot coordination to create collaborative panorama images

Autor:
María Teresa Araúz Pisón

Tutor:
José María Maestre Torreblanca
Profesor contratado doctor

Dep. de Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2016

Agradecimientos

A Pepe, mi tutor, por despertar en mí la ilusión por aprender a entender la realidad con otro lenguaje a través de la Informática durante los primeros meses de mi andadura en esta Escuela; por acompañarme en los arduos momentos que conlleva el adentrarse en esta aventura; y por su confianza y ayuda incondicional en este Trabajo Fin de Grado, que cierra una etapa pero que ha acabado de avivar en mí la pasión por la Ingeniería.

Abstract

In this work, we deal with the problem of coordinating a set of autonomous agents equipped with cameras whose goal is to obtain a panoramic image. To this end, we optimize a multiple-objective cost function that accounts for the number of matched points between the images, the distance between the agents or the distance from the visual central point. Some solutions for this problem are presented considering two different approaches: centralized and distributed coordination of the agents. For each one of them, the results are obtained using optimization methods and heuristic methods. The comparisons between all of them are done from the simulations of the algorithms developed.

Resumen

En este proyecto se trata el problema de la coordinación de un conjunto de agentes autónomos, cada uno de ellos equipado con una cámaras, cuyo objetivo es la obtención de una imagen panorámica. Para este fin, se optimiza una función objetivo de costes múltiples que representa, por ejemplo, el número de puntos emparejados entre las imágenes, la distancia entre los agentes o la distancia hasta el punto central de visión. Se presentan algunas soluciones para de problema basándonos en dos enfoques diferentes: las coordinaciones centralizada y distribuida de los agentes. Para cada uno de ellos, los resultados se obtienen utilizando tanto métodos de optimización como métodos heurísticos. Las comparaciones entre todos los resultados obtenidos se realizan a partir de las simulaciones de los algoritmos que se desarrollan durante este proyecto.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The main problem to be solved is the coordination of a series of robots equipped with a camera with the goal of generating a global panoramic image with certain properties based on the local images [32]. This is an issue with an enormous practical interest due to the results of this research project can be transferred for its implementation in fleets of Unmanned Aerial Vehicles (*drones*), which currently enjoy an enormous popularity because of its numerous applications, e.g., messenger service [10] and geographic information systems among many others. In addition, this topic presents a research challenge that has been already identified by the scientific community [27].

Nowadays, panoramic image generation has become a standard feature in most commercial cameras. As it has been indicated in the previous paragraph, this is achieved by means of a procedure known as image stitching, which consists on the combination of multiple images with overlapping fields of view to generate panoramic images. To this end, the detection of distinctive features in the set of images is used as a mean to determine the pixel coordinates between different images that can be related and used to estimate the corresponding alignments.

In parallel to the consolidation of image stitching techniques, we also assist to an explosion in the drone market with a myriad of potential applications, e.g.: surveillance, packet delivery, construction, etc. Hence, it is straight forward to envisage an application in which several drones collaborate to obtain panoramic images. As will be seen throughout this bachelor thesis, this is a complex problem that can be casted as a multiple objective multi-agent optimization problem. Issues such as how to coordinate the different agents or how to react in case one of the agents behaves faulty are important in this context. The autonomous agents should be relocated regarding the desired result or in the case of error in any robots. This relocation will be carried out optimizing a cost function related with the objective. The

cost function can include aspects as: i) the cost of getting determined points in common between images; ii) the cost of getting a panoramic image with an specific number of pixels; or iii) the cost of moving the robots. We believe that this problem can be interesting in applications such as aerial photography or even military applications, where a set of drones can be assigned to provide the maximum image information from a certain target.

Mainly, the problem is studied supposing the most simple situation, where the motion of autonomous agents is allowed only in a plane parallel to the desired panoramic image, i.e., assuming that there are no differences in angles of captured images between the various cameras. Future studies can get closer to a real world scenario, e.g, by considering the three space dimensions of the motions of agents or the difference of angles mentioned before. But this is not included in the actual project.

Firstly, the issue is studied from a centralized point of view, which mainly consists of solving the problem of coordination between agents considering that an external leader has all the information involved in the situation and makes all the required decisions basing on this information. Next, the same issue is treated from a distributed point of view, that is, there are agents that make the decisions and communicate to collect some information; i.e., global information about the system state is not available for each agent; therefore, the decisions have to be made by each agent with partial information.

Regarding the development of this project, Matlab will be used as work tool. Tests will be carried out by this program.

A noteworthy point is that the developments done as a result of this project may be implemented in a real test bed located at Tokyo. For that purpose, the involved researchers of the Tokyo Institute of Technology offer its laboratories, which can be seen in this article [14].

The outline of the rest of the bachelor thesis is divided in the following manner: Chapter 2 presents some techniques from literature used for Image Stitching and an algorithm developed to create panoramic images once the individual images have been captured. Chapter 3 is focused on Centralized Algorithm Methods to solve the issue of agents coordination, and presents some algorithms and the corresponding simulations. Chapter 4 presents Distributed Algorithm Methods, and includes a general vision of how to implement these methods and examples. Finally, Chapter 5 presents concluding remarks and hints about future work.

Chapter 2

Image Stitching

2.1 Introduction

Image stitching is one of the main problems that we deal with in this work. Image stitching is the name used to describe the process in which a panorama image is obtained from a set of images. The procedure followed is an extension of feature based image registration, but instead of registering a single pair of images, multiple image pairs are successively registered relative to each other to form a panorama.

Since the beginning of the century, researchers have made efforts to improve the existing methods or even to develop new ways to obtain an image panorama. All of these methods follow the same general structure, and the differences between them are principally related to the specific operation used to get the same objective. For example, in [28], V. Rankov proposed an image stitching algorithm for microscopic images based on the clinical necessity of acquiring an image of large regions but retaining microscopic resolution. With this purpose, the method is focused on overcoming both intensity discrepancies and geometric misalignments between the stitched images. Another example can be shown in [26], where the possibility of stitching image with the presence of moving objects is studied. The technique uses heuristic seam selection in the intensity and gradient-domains to choose which pixels to use from each image.

In general, image stitching comprises two steps: image matching and image blending. Image matching consists of the operations performed to obtain the connection between images. In order to achieve this operation, there are two different ways: direct method [33, 21], and feature detection method [7, 25].

The reason of the name 'direct methods' is due to the direct minimiza-

IMAGE STITCHING	
Step 1: Image matching	
Option A: Direct method	Option B: Feature detection method
	1. Feature extraction
	2. Feature correspondence
	E.g.: SIFT and SURF Features
Step 2: Image blending	

Table 2.1: Image Stitching Procedure

tion of the image-based misregistration measure, without special algebraic or geometric transformations, and because they are usually based on the direct minimization of intensity errors [33]. Using direct methods, it is possible to achieve very accurate registration because they use all the available image data [5], but it has the disadvantage that a high quality image is always needed [22]. In most cases, this type of images is difficult to achieve, and because of that, feature detection method is often used.

Feature detection methods have the aforementioned advantage, but they also possess the problem that invariance properties are needed to enable reliable matching of an arbitrary panoramic image sequence [5]. Generally, feature detection involves two procedures: feature extraction and feature correspondence. Many image segmentation techniques are used for feature extraction, such as the Canny operator in [30] or the classification method in [34]. Feature correspondence is the most challenging problem at present, and its performance depends in the characteristics of the features detected. As it is explained in [9], a robust algorithm to establish control-point correspondences is one of the most important tasks in automated image registration, and some of the existing feature-matching algorithms are referenced in that article as well.

A variety of algorithms are used for feature detection, but two of them, based on SIFT and SURF features, which are both broadly explained in next section, are the most commonly used for feature detection because of their robustness. Firstly, SIFT was developed in 1999 for object recognition [23], and in 2004 presented for image stitching [24]. SURF appeared two years later as an improvement of SIFT applied to image matching [2].

The second part of image stitching is blending. The purpose of these methods is to reduce the difference of intensities in the connection of the images by making the edges invisible [28]. Ideally, this step would not be necessary, but in practice the edges of the overlapping images can be distinguished. For this reason, the choice of a good blending strategy improves

(a) Original images

(b) Panorama image

Figure 2.1: Example of panorama image with bad blending technique.

the quality of the final panorama image. As it is shown in Figure 2.1, where a image panorama is obtained from some images using Matlab, the edges of the images in the panorama are clearly visible. This is the reason why the selection of a good blending algorithm is essential to avoid the situation represented in the figure.

There are many algorithms used for image blending, e.g.: watersheds blending [18], multi-band blending [5, 4], multi-resoluion spline blending [26] or gradient domain blending [28]. A discussion of some of these and other techniques is given by R. Szeliski in [32].

To sum up, the principal steps and procedures of image stitching have been collected in Table 2.1.

2.2 Feature detection methods

These kind of methods are used in order to solve the task of finding correspondences between two images of the same scene or object. As it has been said in the previous section, most of feature detection methods require two main steps. Firstly, feature extraction, which refers to the detection of the features of all the images. *Feature* are represented by a vector that collects the relationship between each element and its neighbourhood. These elements usually represent the interest points of the image, which are selected by its location, e.g., high-contrast regions of the image, such as object edges. Feature vectors have to be distinctive and, at the same time, robust to noise, detection errors, and geometric and photometric deformations, in order to ease Feature correspondence, which is the next step. It consists of matching features by finding the relation between feature vectors of all the involved images. This matching is often based on the distance between vectors. More information about feature detection methods can be found in [20] and [17].

There are many techniques used to carry out these steps, and there are some complete methods that collect all of them. In the following sections, two famous methods of feature detection, SIFT and SURF, are broadly explained.

2.2.1 SIFT

SIFT corresponds to the abreviation of "Scale-invariant feature transform", as it transforms image data into scale-invariant coordinates relative to local features. SIFT is one of the most famous algorithms used in computer vision to detect and describe local features in images. SIFT was developed for object recognition applications by Lowe in [23], but some years later, the same author included a new application for SIFT: image matching [24].

The principal characteristic of SIFT features is that they are invariant to uniform scaling and orientation, and also partially invariant to affine distortion and illumination changes. These points are relevant for the goal of matching images, because these invariations are essential for searching the relation between features of different images. It also has the appealing property of being distinctive and relatively fast, which is crucial for on-line applications. However, its drawback appears when high dimensionality is required along with short processing time. In these occasions, other methods have to be introduced to solve the problem but sacrificing some of the requests.

SIFT mainly includes four major stages: scale-space extrema detection, keypoint localization, orientation assignment and calculation of keypoint descriptors [24]. Basically, the algorithm computes a histogram of local oriented gradients around the interest point, the so-called keypoints, and store them in a 128-dimensional vector [2]. One of the image feature generation methods most used is Lowe's one [24], because it transforms an image into a large collection of feature vectors, and ensure that the keypoints are more stable for matching and recognition. SIFT descriptors are obtained by considering pixels around a radius of the keypoints [24].

The principal drawback of this method is the velocity required to get the result. This is the reason why SURF is implemented in some works, where the time requirements are most relevant than accuracy [22, 20].

2.2.2 SURF

Speeded-Up Robust Features, SURF, is a feature detection method based on SIFT presented by Herbert Bay and his coworkers in 2006 [2]. As it is detailed in that article, the goal of their work was to develop a new method focusing on scale and image rotation invariant detectors and descriptors, which is a compromise between feature complexity and robustness to deformations. A novel detector-descriptor method was developed and designated as SURF. The detector is known as Fast-Hessian Detector and is based on the Hesian matrix and integral images. The SURF descriptor represents the distribution of Haar-wavelet responses within the interest point neighbourhood, taking only 64 dimensions, and hence involving a reduction of computation time compared to SIFT. In mathematics, the Haar wavelet is a sequence of rescaled "square-shaped" functions which together form a wavelet family or basis. Wavelet analysis is similar to Fourier analysis in that it allows a target function over an interval to be represented in terms of an orthonormal basis [19].

Both, detector and descriptor, are based on integral images in order to reduce computation time. An integral image, which can be placed at any

location, represents the sum of all pixels in the image of a rectangular region formed by the origin and the point of its location. Having these results, the time required for calculating the sum of the intensities is reduced [2].

The descriptor is based on SIFT, but it takes two steps. First, orientation assignment, where a reproducible orientation of the interests points is detected in order to be invariant to rotation. Then, the components of the SURF descriptor are extracted from a square region centered around the interest point and aligned to the selected orientation.

To identify the elements of the descriptor, the selected region is split up into smaller 4 x 4 square sub-regions. For each one, a four dimensional descriptor vector is defined as: the first and second elements are the sum of the Haar wavelet response in horizontal and vertical respectively, and the other two elements are the sum of these absolute values, in order to have information about the polarity of the intensity changes. After computing all these vectors for the four sub-regions, the result is a SURF descriptor of length 64, as mentioned before.

Many modifications of SURF have also been carried out, e.g., in [22], where a modified SURF is combined with other methods such as K-NN and RANSAC to achieve the final panorama image.

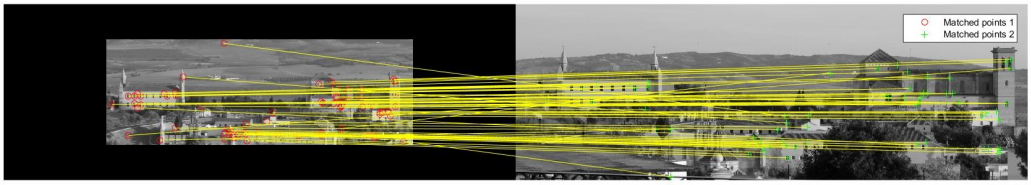
In the work considered in this thesis, SURF features are going to be used, due to the fact that better results are obtained in agreement between computing speed and accuracy. Note that all of these methods are included in Matlab and hence the real implementation of this algorithm is not necessary for this work, and not included in this document. See [2, 22, 20] for more information about these methods.

Using these functions included in Matlab, it is possible to check the behaviour of the algorithm explained for SURF features. One of the results obtained from Matlab is shown in the Figure 2.2, where the tested images are slightly different. As it is seen is that figure, there is a good quantity of matching points. The other example performed consists of the comparison of an image its rotated version. The resulting image, represented in Figure 2.3, allows us to check that the SURF features are invariant to rotation, getting similar results than the previous example without rotation.

2.3 Algorithm for Image Stitching

In this section, we define the algorithm used to generate the panoramic images with the robots considered in this bachelor thesis. As it was said, the functions used in the algorithm are included in Matlab's library and no attention is paid to them.

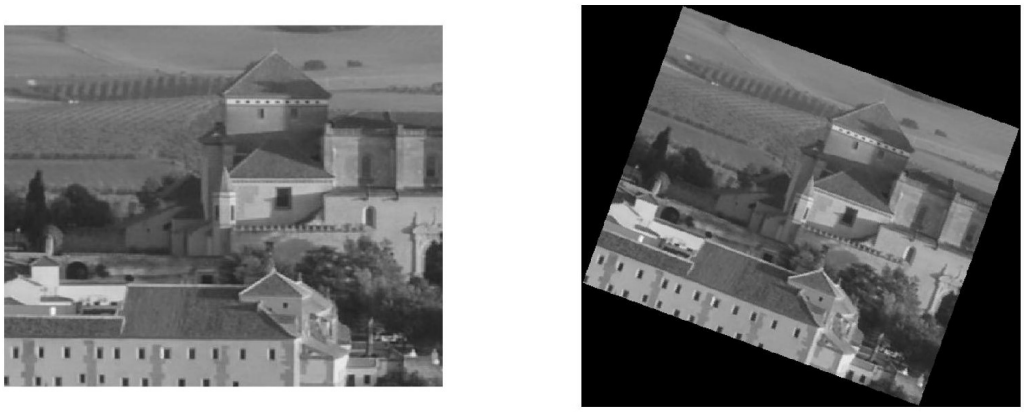
(a) Original images



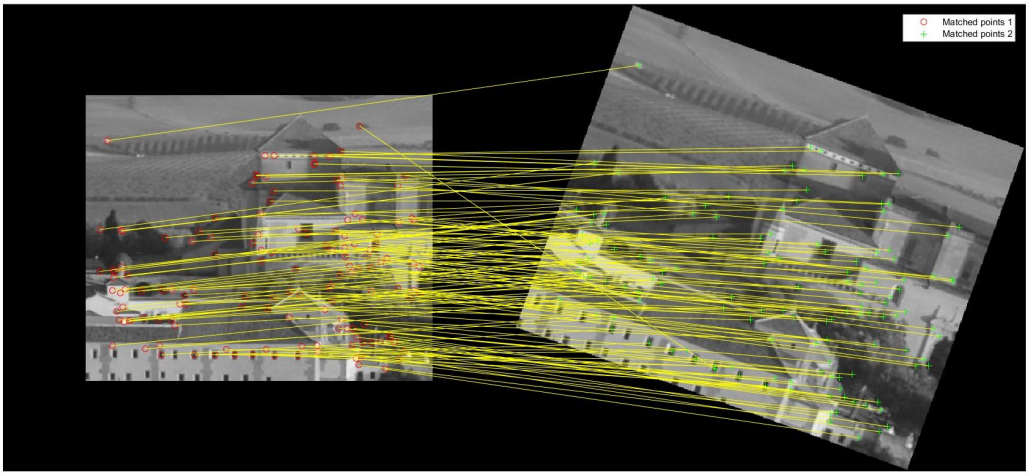
(b) Selected matched features in images

(c) Overlapped images with matched features

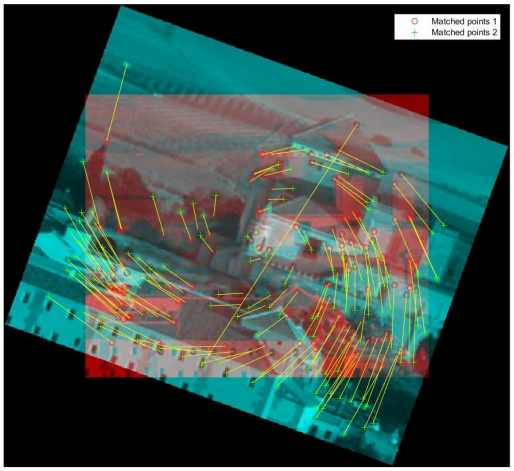
Figure 2.2: Example of matching SURF Features of similar images



(a) Original images



(b) Selected matched features in images



(c) Overlapped images with matched features

Figure 2.3: Example of matching SURF Features of rotated images

The algorithm created is divided in two parts: the first one dedicated to search matched points between images, and the second one, to join them using these matched points. The steps followed by the algorithm used are now detailed:

1. Search matched points:
   (a) Initialize all the transforms to the identity matrix using the Matlab function *projective2d*.
   (b) For each agent, find the neighbor with the highest number of matching points.
      i. Using the Matlab function *matcheFeatures*, calculate the number of matched points between the corresponding images, and record it only in the case of being greater or equal than 20. Otherwise, set the number of matching points to zero and discard the neighbor as a candidate for matching.
      ii. Calculate the average matched points, and discard the images with a lower number of matched points. Order the remaining agents in ascending order of matched points.
      iii. Look for the agent with highest number of matched points that has not been previously selected. In the case of all the agents has been selected already, select that with the maximum number of matched points.
   (c) Once a pair of agents is matched up, the transformation between both images is estimated using the Matlab function *estimateGeometricTransform*.
   (d) Finally, when all the agents have been matched up, calculate the transformation final matrix as the multiplication of the corresponding image and the images of the previous agents.
2. Combine single images:
   (a) Compute the output limits for each transformation in X and Y axis using the Matlab fuction *outputLimits* and find the image that is in the center of each axis.
   (b) Apply the center image's inverse transform to all the others for X and Y axis.
   (c) Initialize the panorama as an empty panorama image. Use the *outputLimits* method to compute the minimum and maximum output limits over all the transformations. These values are used to automatically compute the size of the panorama.

(d) Create the panorama, combining the images one by one:

i. Create a 2D spatial reference object by defining the size of the panorama, using the Matlab function *imref2d*.

ii. Transform the image considered into the panorama and create the mask for the overlay operation converting it to a binary image. Use for both operations the Matlab function *imwarp*.

iii. Overlay the image onto the panorama. Use the Matlab functions *step* and *vision.AlphaBlender* to overlay images together.

Finally, Algorithm 1 presents the program used for the simulations of the next chapters:

Algorithm 1: Image Stitching

Initialize all the objects for each robot and its image, including its SURF features and points;
Initialize all the $tforms$;
Initialize id_robMP as 0.
for $n = 1, ..., Nrobots$ **do**
  for $id\_prue = 1, ..., Nrobots$ **do**
    if $id\_prue \neq n$ **then**
      $indexPairs\_prue = matchFeatures(R(n).features, R(id\_prue).features)$;
      $total\_MP = size(indexPairs\_prue)$;
      if $total\_MP < 20$ **then** $total\_MP = 0$;
      end if
    end if
  end for
  Calculate $average$ of $total\_MP$;
  $possible = total\_MP > average$;
  $total\_possible = sum(possible)$;
  if $total\_possible = 1$ **then**
    for $id\_prue = 1, ..., Nrobots$ **do**
      if $possible(id\_prue) = 1$ **then** $id\_selecc = id\_prue$;
      end if
    end for
  else
    $total\_MP = possible. * total\_MP$;
    Keep in $possible\_min\_max$, $total\_MP$ in ascending order;
    Initialize $id\_selecc = 0$ and $i = 0$;

25: **while** $id\_selecc = 0$ **and** $i < total\_possible$ **do**
26: $id\_prue = possible\_min\_max(Nrobots - i)$;
27: **if** $selecc(id\_prue) = 0$ **then** $id\_selecc = id\_prue$;
28: **end if**
29: Update $i = i + 1$;
30: **end while**
31: **if** $i = total\_possible$ **then**
32: $id\_selecc = possible\_min\_max(Nrobots)$;
33: **end if**
34: **end if**
35: Update $selecc(id\_selecc) = selecc(id\_selecc) + 1$;
36: Update information of $R(n)$
37: $tforms(n) = estimateGeometricTransform(R(n).SURFpoints, R(id\_selecc).SURFpoints)$;
38: **end for**
39: Update $tforms.T = tforms(R.id\_robMP).T * tforms.T$ for all the robots;
40: Calculate $xlim$, $ylim$ for all $tforms$;
41: Calculate $centerImageIdx$ and $centerImageIdy$;
42: Calculate $Tinvx$ and $Tinvy$ as the center image's inverse transform
43: Apply $Tinvx$ and $Tinvy$ to all the $tforms$;
44: Calculate $xMin, xMax, yMin, yMax$ as the limit over all $tforms$;
45: Initialize *panorama*
46: Initialize *blender*
47: **for** $n = 1, ..., Nrobots$ **do**
48: $warpedImage = imwarp(Image, tforms)$;
49: $warpedMask = (imwarp(ones(size(Image), tforms)) >= 1$;
50: $panorama = step(blender, panorama, warpedImage, warpedMask)$;
51: **end for**

Chapter 3

Centralized Algorithm Methods

In this chapter we deal with the problem of coordinating multiple agents. The multi-agent coordination problem requires to integrate the movements of several agents, with the goal of maximizing their overall performance [16]. The concept of *agent* makes reference to any mobile device whose movement is monitored. Some examples can be found in [29]: mobile robots, unmanned air vehicles (UAVs), autonomous underwater vehicles (AUVs), satellites, aircraft, etc. The principal question that has to be solved is: How can groups of agents be intelligently coordinated with this purpose?

Many researchers have worked on this topic. For example, T. Siméon presented in 2002 a geometry-based approach for multiple mobile robot motion coordination [31]. In that article, the author tried to solve the problem of coordinating the motion of several robots that move along fixed independent paths to avoid mutual collisions. In contrast, various researchers investigated on what autonomous mobile robots can do in distributed coordination problems [13]. Another different example can be found in [27], where the collaboration is studied between a group of UAVs (microdrones), principally for cooperative aerial imaging based for disaster managements applications.

Robot coordination is necessary for water, earth and aerial robots and can be carried out in indoor or outdoor environments. In many situations, it is necessary to build maps of the environment, simultaneously with the use of a position estimator such as Extended Kalman Filter (EKF) [8]. In contrast, outdoor applications typically use the technique known as SLAM, (Simultaneous Localization and Mapping). SLAM deals with the problem of placing a mobile robot at an unknown location in an unknown environment and incrementally building a consistent map of this environment while simultaneously determining its location within this map. More information about SLAM method can be found in [1, 11].

In this paper, we work with UAVs. The principal benefit of using several

drones is the possibility of achieving an image of a large area with high resolution. There are many applications in which aerial views can be really helpful, e.g., disaster situations, due to the overview of the environment achieved with this coordinated system [27].

There are other advantages of using such multiple drone setting [27]: i) Firstly, the possibility of covering a much larger area; ii) more information can be gathered if necessary; and iii) the possibility of failing is lower because the system is formed by various elements, i.e., the system is more redundant and robust against failures.

For this project, we assume that the UAVs control techniques are known and available to use. Hence no attention is paid to this topic. It is also considered that the drones are equipped with enough sensors and actuators to achieve the desired goals [27]. Therefore, the rest of the thesis is mainly dedicated exclusively to the way of coordinating the agents of the system considered in our work. There are two different approaches for multi-agent coordination: the centralized and distributed. This chapter deals with centralized coordination and the next one with distributed coordination.

As its name indicates, *Centralized Coordination* consist of a central operator that has complete information about all the system, and it is the same operator that makes the decisions about each drone movements. This central operator can be one of the elements of the system, and it is also called the leader. It is the responsible for sending to each element the corresponding information about the motion [6].

Therefore one of the principal drawbacks of centralized coordination is the lack of robustness when there are failures in the communication and/or incorrect operations of the leader [12]. It is possible to solve this problem by using a different kind of centralization, but remaining it. One option is explained by A. Farinelli in [12], who weakly expounds centralized systems, characterized by the fact that the leader is not chosen a priori, but it is selected dynamically during the mission depending on the current situation of the team and the environment.

3.1 Problem setting

We assume that there is a set $\mathcal{N} = \{1, \ldots, N\}$ of mobile agents, each one equipped with a camera. For simplicity, in this work, we assume that the agents can move only in the XY plane. In addition, it will be assumed that all the cameras have the same orientation perpendicular to the XY plane, and takes images of the same fixed size.

The objective of the agents is to create a panoramic image from the images

that each one provides. To this end, they can move freely in the XY plane and exchange information. Notice that there are different ways in which this objective can be achieved. For example, one possibility is to promote the minimization of the distance between the agent position and the selected area as more relevant, or the distance between the agent position and the previous agent position. Likewise, the minimization of the overlap between the images can be another goal, but in that case, the number of matched points between images has to be considered too. Moreover, robustness is another relevant characteristic in the generation of the panoramic image: a certain level of redundancy eases the redistribution of the agents in case of agent failures. Therefore, to cover these and other possibilities we propose the optimization of the following cost:

$$J(x, y) = \alpha \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} \mathrm{MP}(i, j) + \beta \mathrm{NP}(x, y) + \gamma \mathrm{DP}(x, y) + \theta \mathrm{DR}(x, y).$$

where x and y are vectors that aggregate the corresponding local coordinates, i.e., $x = [x_i]_{i \in \mathcal{N}}$ and $y = [y_i]_{i \in \mathcal{N}}$; $\mathrm{MP}(i, j)$ stands for the matching points between agents i and j; $\mathrm{NP}(x, y)$ represents the number of pixels of the panoramic image; $\mathrm{DP}(x, y)$ represents the distance in relation to the vision central point and $\mathrm{DR}(x, y)$ the distance to the previous agent. Finally, α, β, γ and θ are weights that can be tuned according to the designer goals.

The main idea is the following: define a hot region in the image, so that those pixels have priority to be targeted. Likewise, it is possible to define a time stamp for the distinctive points found in each image. Time stamps can be used to determine which agent has the preference to remain in its current position (older time stamps). But in this project, time stamp is not considered.

3.2 Algorithms

The principle of this approach has been explained: there is a single operator that has global information about the system state and who is the responsible for the movements of each agent.

In this section, we offer two different approaches to solving the proposed problem, both considering centralized coordination between agents. In the first place, we show an algorithm whose result leads to the optimum robots positions. Next, we present another possible algorithm based on heuristic methods, i.e., the solution found may not be the optimum one.

3.2.1 Optimal algorithm

The main idea of this algorithm resides on the fact that all the agents are located at the same time, finding the combination of the coordinates of all the agents that minimizes the cost function. The number of variables of the optimization problem increases by two each time a new agent is considered. Each position is formed by two coordinates, x and y, for the height (coordinate z) is maintained constant. For simplicity, the number of agents considered in here is two. Therefore, the number of variables involved in the optimization problem is 4, which is the minimum for this purpose.

The incentive that generates the motions of the agents looking for a better result is that there is a visual central point defined where more attention has to be given. This point is selected by the user, so it is always considered as an initial parameter in our work. Regarding this point, a matrix represents the available area and each element holds a value referred to the distance between this point and the visual central one.

The cost function is calculated as the addition of some elements, which are obtained each time a new option is evaluated. These elements are:

- The addition of the values of both images depending on the distance to the visual central point.
- The absolute value of the distance between both agents.
- The inverse of the total number of matched points between images captured by both agents.

The algorithm created for searching the optimum of our problem can seem very simple, but as it will be seen in the section of results, the computational time required is excessive, making this solution not very suitable for our purpose. The steps followed in the algorithm are provided below:

1. Initial definition of required parameters:
   a) Size of the total available area of movements.
   b) Size of captured image by agents.
   c) Number of agents involved in the situation and an object in Matlab for each one to save all its information.
   d) Definition of the central vision point.
2. Creation of the matrix containing the absolute value of the distance of each pixel to the central vision point.

3. Position of the agents in the available area:

   a) Define the variables of coordinates for the agents.

   b) Using four structures 'for' in Matlab, examines all the possible combinations for all the agents positions. Make sure that the captured images get inside the available area by considering the required margin for both coordinates (i and j).

   c) Calculate the cost function for each combination. If it is less than the minimum cost value previously recorded, update the solution information, considering this combination as the new potential optimum result.

4. Generate the panoramic image using the algorithm detailed in the section of Image Stitching.

Finally, Algorithm 2 details the program.

Algorithm 2: Centralized method with optimum solution

Initialize the size of the area of movements: $Mmax$ and $Nmax$;
Initialize the size of captured images: Mr and Nr;
Initialize $Nrobots$ as 2 and the vector R with their respective objects;
Define the vision central point $(i0, j0)$;
Create the matrix *map* with values depending on the distance to $(i0, j0)$;
for $i1 = Mr/2, \ldots, Mmax - Mr/2$ **do**
    for $j1 = Nr/2, \ldots, Nmax - Nr/2$ **do**
        for $i2 = Mr/2, \ldots, Mmax - Mr/2$ **do**
            for $j2 = Nr/2, \ldots, Nmax - Nr/2$ **do**
                if $|i1 - i2| >= Mr/4$ **and** $|j1 - j2| >= Nr/4$ **then**
                    Calculate Val_pixel1 and Val_pixel2 from *map*;
                    $Val\_pixel = Val\_pixel1 + Val\_pixel2$;
                    $Dist\_rob$ equal to distance between robots;
                    Capture $image1$ and $image2$;
                    Obtain $features1$ and $features2$ of $image1$ and $image2$;
                    Calculate $numMP$ between $features1$ and $features2$;
                    if $numMP >= 20$ **then**, calculate $COST$;
                    end if
                    if $COST <$ *cost_opt* **then**
                        Update *cost_opt*;
                        Update $i1\_opt$, $j1\_opt$, $i2\_opt$ and $j2\_opt$;
                        Update information in $R(1)$ and $R(2)$;

```
                    end if
                end if
            end for
        end for
    end for
end for
Create panorama using Algorithm 1;
```

3.2.2 Heuristic method

In order to overcome the drawbacks of the previous algorithm, and despite the solution is the optimum, other algorithm has been developed in order to improve some of those issues. This algorithm is based on heuristic methods, meaning that the certainty of obtaining the optimum result is lost, being this the strongest disadvantage of this new solution.

There are many changes introduced between the previous algorithm and the new one. Some of the principal differences are following detailed:

- The principal benefit is the possibility of increasing the number of agents, as well as the size of the total area in which the movement of agents takes place, maintaining the computational time in a reasonable range. Hence, the size of the images taken by the agents can also be enlarged, getting the situation closer to the real world.

- The agents are now positioned one by one, meaning that once an agent is located, it can not be moved later. Next agents have to find their best position having all the previous ones positioned.

- The cost function contains now the distance from the position of the actual agent to the visual central point, instead of the distance between agents, as it was in the other algorithm.

- The first agent is located in the visual central point and no cost function is calculated for this agent. The rest of the agents are positioned around it.

- A new map is created in order to account for the positions that have already been captured. At the beginning of the program, the map is completely defined as not captured positions; and every time a new agent is positioned, the points captured by this agent are selected as occupied.

- Another novelty incorporated is related with the overlapping degree. Given that the agents are located one by one, the overlapping between images has to be considered.

- Excluding the first agent, the rest of the agents are positioned following the same idea. Each agent examines the neighboring positions of the area in the map selected as occupied, i.e., the area already captured by previous agents. Moreover, for each one of these positions, all the possibilities of overlapping are studied, selecting the degree of overlapping that gives the best results regarding cost function. At the end, the final position is selected between all the possibilities in accordance with the one that optimizes the cost function.

- The neighboring positions are selected from the sharp curve around the occupied area in the map, separated by a distance equal to half of the size of the image captured by an agent, from the line that differentiates both zones. The positions are recorded going across this line with an increase equal or less of the size of the image captured by an agent. After all the neighbored positions are selected, the agent moves around each one modifying the overlapping degree, in order to find the one that optimizes the cost function.

- Once all the agents are positioned, the panorama image is created by merging all the captured images using the same technique of image stitching of the previous algorithm.

Having into account all of these considerations we proceed to outline the steps followed by this algorithm:

1. Initial definition of the required parameters:

   a) Size of the total area of movements.

   b) Size of the captured image by agents.

   c) Number of agents involved in the situation and an object in Matlab for each one to save all its information.

   d) Initialize the matrix where localize the already captured pixels.

   e) Initialize the vectors including the overlapping degrees that are going to be studied.

   f) Definition of the central vision point.

2. Creation of the matrix containing the absolute value of the distance of each pixel to the central vision point.

3. Positioning of all the agents in the available area:

   a) Initial positioning of all the agents in the edges of the area.

   b) Position the first agent in the vision central point, capture its grayscale image, update the matrix of captured pixels and extract its SURF features.

   c) Position the rest of agents following the same steps:

      i. Find the position of the previous agent.

      ii. Using the matrix of captured pixels, find the sharp curve around the occupied area in the map, separating it from the line that differentiates both zones, a distance equal to half of the size of the image captured by an agent.

      iii. Go through the curve selecting the neighbor positions with an increase between them equal or less of the size of the image captured by an agent.

      iv. Initialize all the required structures to examine all the neighbor positions: vector with position coordinates for each one, vector to save the overlapping degree that generates the best result for each one, the value of cost function as infinite, and some elements of the object defined for each robot.

      v. For each neighbor position, examine all the possibilities of overlapping degree, calculating the value of the cost function and at the end, save the one that minimizes its value.

      vi. Select between all the neighbor positions with the best overlapping degree, the one that optimizes the value of the cost function.

      vii. Each time a new agent is located: capture the image, extract its SURF features, update the new information in the matrix of captured pixels and also the one of distance dividing the value of pixels captured by the overlapping degree.

4. Generate the panoramic image using the algorithm detailed in the section of Image Stitching.

Finally, Algorithm 3 presents the program used for the purpose explained.

Algorithm 3: Centralized method using heuristic method

1: Initialize the size of the area of movements: $Mmax$ and $Nmax$;
2: Initialize the size of captured images: Mr and Nr;
3: Initialize $Nrobots$ and the vector R with their respective objects;

Initialize the matrix map_pos as 0;
Define the vision central point $(i0, j0)$;
Define the vectors ki and kj with overlapping degrees;
Create the matrix map with values depending on the distance to $(i0, j0)$;
Position the $Nrobots$ in the edges randomly;
Position robot 1 in $(i0, j0)$ and update information in $R(1)$;
Update matrix map_pos;
for $robot = 2, \ldots, Nrobots$ **do**
    Update i_ant and j_ant from $R(robot - 1)$;
    Calculate the matrix im_edg containing the dilated edges of map_pos;
    Make sure that im_edg contains a single pixel wight sharp curve;
    Find $i1$ and $j1$ from the curve of im_edg;
    while $i \neq i1$ **and** $j \neq j1$ **do**
        Find i_next or j_next by increasing until Mr and Nr pixels following the curve of im_edg;
        Update matrix P including the new positions (i_next, j_nex);
    end while
    Calculate $Nvec$ from vector P;
    for $case = 1, \ldots, Nvec$ **do**
        for all possibilities of overllaping **do**
            Calculate Val_pixel from map;
            $Dist\_rob$ equal to $(i0, j0)$;
            Capture $image$;
            Obtain $features$ of $image$;
            Calculate $numMP$ between $features$ and $R(robot - 1).features$;
            if $numMP >= 20$ **then**, calculate $COST$;
            end if
            Update matrix mat_cost with $COST$;
        end for
        Keep in $vector\_ij(:, case)$ the position with minimum value in mat_cost;
        Keep in $COST\_case(case)$ the minimum one obtained;
    end for
    Select $case\_min$ of $vector\_ij$ with minimum $COST\_case$;
    Update information in $R(robot)$;
    Update matrix map_pos;
end for
Create $panorama$ using Algorithm 1;

KEY PERFORMANCE INDICATORS		
Computational Time	Number of pixels	Distance to central point
2384.645 s	111600	$1.5623x10^7$

Table 3.1: Key Performance Indicators of the simulation of the optimal centralized algorithm described in Algorithm 2 (results in Figure 3.2).

After testing this algorithm in simulations, its advantages are clearly verified, but some disadvantages are also discovered, e.g., the increase of computational time when the number of agents raises. However, some modifications have been introduced to improve these drawbacks, and they will be discussed and assessed in next section.

3.3 Simulation results

3.3.1 Results of the algorithm with optimum solution

The algorithm has been applied to just two agents, as it has been mentioned in the Section 3.2.1. Firstly, all the possible combination of the positions of the two agents were checked. However, due to the long computational time required, it was decided to decrease the number of possibilities by increasing the coordinates in 10 units each time a new combination needed to be tested. Therefore, the result obtained is closer to the optimum with the accuracy big enough for our purposes.

The value of the initial parameters defined for the simulation were the followings: 2 agents, the size of the area was 397x483, the size of captured images was 200x200 and the visual central point is located in $i = 250$ and $j = 250$, as shown in Figure 3.1.

Some *Key Performance Indicators* (KPIs) have been calculated in each simulation. Three KPIs have been defined in this project: i) computational time, ii) the total number of pixels of the generated panoramic image, and iii) the addition of the distances from all the captured pixels to the visual central point.

The results obtained from the simulation are shown in Figure 3.2. The corresponding values of KPIs are presented in Table 3.1. These results will be used for comparison with the algorithm that will be shown in the next section.

Figure 3.1: Central vision point for simulation of the optimal algorithm.

3.3.2 Results of the heuristic algorithm

The results of this algorithm are shown and compared to others. In order to clarify its benefits and drawbacks, some of them are already outlined.

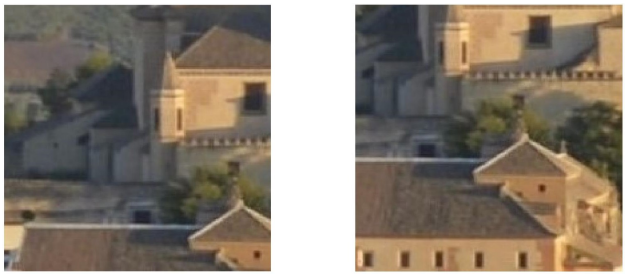
First, the comparison is made with the optimal results, pointing out the differences between both algorithms with the same initial value of the parameters. Next, the initial values of the algorithm are changed to make the results closer to a real world implementation and compared with those obtained by modifying some parts of this algorithm.

Comparison between the algorithm and the optimal solution

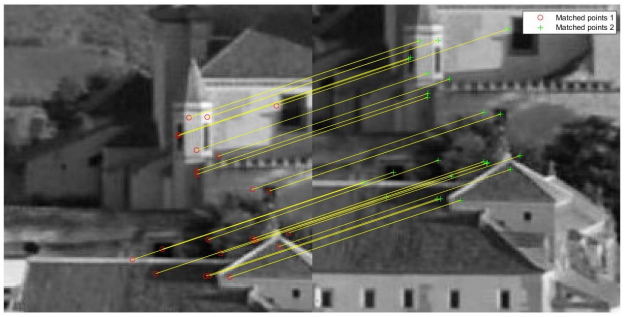
The first simulation has been done with the same initial parameters: 2 agents, the size of the area 397x483, the size of captured images 200x200, and the same visual central point shown in Figure 3.1. With this algorithm, it is necessary to define the overlapping degree range for the simulation. In this case, set it in the percent range [10,70], but the increase have been chosen of 1% and 10% in two different simulations.

The results obtained for both simulation considering the two increase of overlap range are exactly the same, and only one is presented in Figure 3.3. Its values of KPIs are different, so both are presented in Table 3.2, together with the ones of the optimal solution.

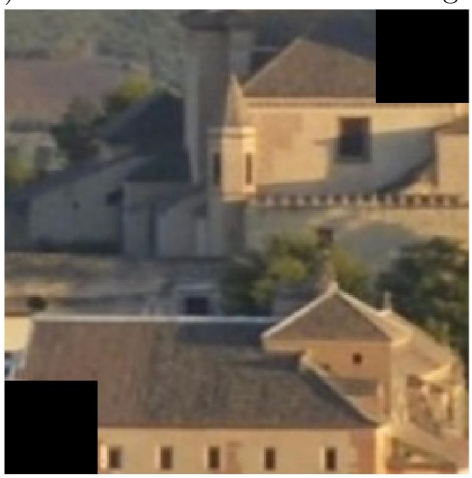
(a) Available area represented by the whole image in gray scale and in color.



(b) Images captured by the agents.



(c) Matched features between images.



(d) Panoramic image obtained.

Figure 3.2: Result of the optimal centralized algorithm described in Algorithm 2.

(a) Available area represented by the whole image in gray scale and in color.

(b) Images captured by the agents.

(c) Matched features between images.

(d) Panoramic image obtained stitching images captured.

Figure 3.3: Result of the centralized heuristic algorithm described in Algorithm 3.

KEY PERFORMANCE INDICATORS			
	Comp. Time	No. of pixels	Distance
Optimal solution	2384.645 s	$111,600$	$1.5623x10^7$
Heuristic solution 1	81.806 s	$60,893$	$6.4129x10^6$
Heuristic solution 2	4.655 s	$60,893$	$6.4129x10^6$

Table 3.2: Key Performance Indicators of the simulation of the optimal centralized algorithm described in Algorithm 2, compared with the simulations of heuristic centralized algorithm described in Algorithm 3 (results in Figure 3.2 and 3.3).

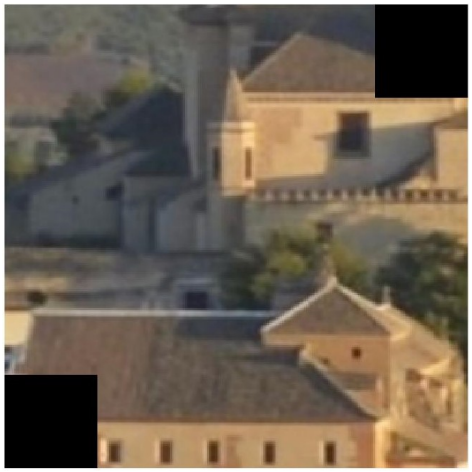
The difference between optimum and heuristic solutions is clearly seen at a glance if we compare both panoramic images, represented together in Figure 3.4. The first one, corresponding to the optimum result, is centered around the visual central point. However, the second one obtained by heuristic methods is only centered on that point the image captured by the first agent, instead of the whole panoramic image. But if we look at the values of the KPI referred to the distance of pixels to the center, it seems the opposite. However, this difference is due to the lower number of pixels of the panoramic images obtained by the heuristic method. This decrease of pixels necessarily causes a decrease in the sum of distances. Therefore, the comparison between both panoramic images is the best evidence that the heuristic methods do not converge to the optimum.

Another meaningful difference between both algorithms is the KPI related to the computational time required. The algorithm based on heuristic methods is between 29.15 and 512.27 times faster than the optimum one depending on the overlap range set.

Comparing the KPIs of both results of the heuristic method, the only difference is the computational time required. In this case, the decrease in the overlap range causes a huge increase in computational time, giving exactly the same result. Therefore, for next simulation in this section, the overlap range will be the second, i.e., using the increase of 10%.

Finally, it is necessary to make some comments about the results that we would obtain in the case of considering more than 2 robots in both cases. Based on the results obtained, we can expect that the difference in computation time would become more relevant each time a new agent is included mainly because the first algorithm adds two new variables each time a new agent is included, and consequently there is a combinational explotion in the search space of the solution. However, the centralization around the visual central point is worse in Algorithm 3, while in Algorithm 3 remains constant.

(a) Central vision point for simulations.



(b) Panoramic image obtained with optimal algorithm described in Algorithm 2.

(c) Panoramic image obtained with heuristic algorithm described in Algorithm 3.

Figure 3.4: Comparison between panoramic images obtained with optimal and heuristic algorithms.

The designer has to evaluate which characteristics are more relevant for the application being considered and choose the most suitable algorithm.

Simulation with various agents

The relevant drawback of the optimal algorithm, as it has been discussed in the previous point, is the long computational time when the agents are more than two.

Due to the characteristics involved in our project, the precision of the solution is not as important as the speed in which the solution is obtained. Our real scheme is composed of various agents, more than two, flying through an area with huge dimensions, and capturing images with size larger than the one considered before. Therefore, the optimal algorithm is not valid for our purpose, and the algorithm selected is the one based on the heuristic approach.

Next, we test this algorithm with more agents and increasing the size of the image. Some of the solutions obtained are collected and presented here.

The new values of the parameters are the following: the size of the area was 4000x6016, the size of captured images 1400x1400, and the central point was $i = 2100$ and $j = 2100$, as shown in Figure 3.5. The overlapping range available for the simulations is defined as in the first case: it goes from 10% to 70%. The increase is selected of 10% due to the reasons explained in the previous part.

The algorithm has been simulated with 2, 3, 4, 5, 6, 7, and 8 agents, and their results are presented in Figures **??**, and 3.12 respectively. The available area is the same for all the cases, and it is only presented in Figure 3.6 in the image (a). There is a small difference between the simulations with 2 agents to 4 and those with 5 agents to 8, because in order to get a continuous panoramic image, the weight of the cost function elements was changed, giving more relevance to the distance to the visual central point when the number of elements is higher. This can be clearly appreciated when comparing the captured images of 4 agents with those of 5 agents. Moreover, for simplicity, for the simulation of 6 agents to 8 we only present the images captured by all the agents and the panoramic image i.e., the images containing the matched features are omitted for those cases.

The KPIs of all of these results are shown in Table 3.3. The values of the three KPIs considered increase at the same time that the number of agents is also increased. It was evident that the total sum of distance to the center would become higher for two reasons: i) the total number of pixels has to be increased when using more agents, and ii) when more agents are considered some of them have to be necessarily positioned farther to the

Figure 3.5: Central vision point for simulation of the heuristic algorithm with higher dimensions situation.

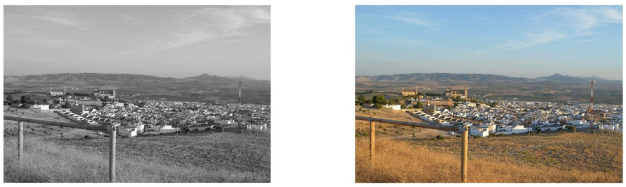
center. Therefore, the results obtained are as it was expected.

From this information, it is possible to make a comparison between the simulations of the previous section with this algorithm with 2 agents but with smaller dimensions than here. In Table 3.4 the values of the KPIs of both simulations are presented. Considering the difference between both values of computational time, we conclude that, although the algorithm based on heuristic methods is better for our purpose than the optimal one, it is still needed to introduce some improvements in order make it suitable for our necessities, i.e., short computational time for real-time applications.

Therefore, the main drawback of this algorithm is the increase of the computational time required when the dimensions involved in the application are considerably high. In order to solve this problem, some modifications have been introduced. We have performed some simulations for the sake of comparison.

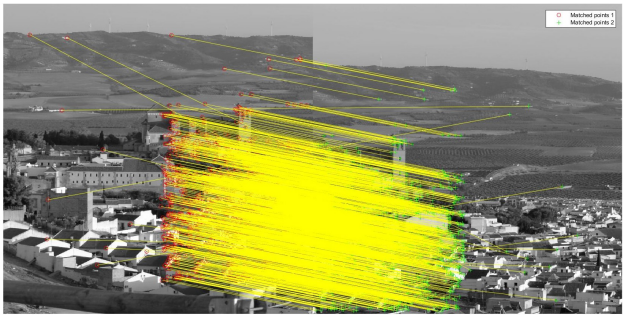
Comparison with some modification of this algorithm

The first modification of the algorithm is focused on the idea of minimizing the number of searching positions to be examined. In this case, the problem is solved ignoring the area of the map selected as captured pixels. Before, this



(a) Available area represented by the whole image in grayscale and in color.

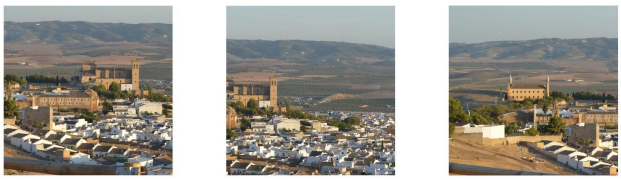
(b) Images captured by the agents.



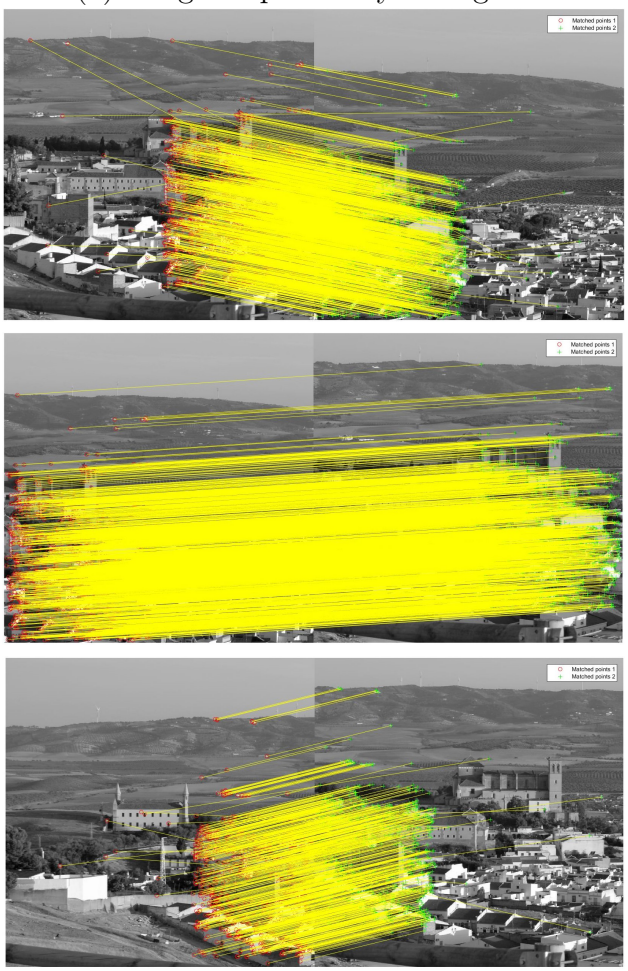
(c) Matched features between images.

(d) Panoramic image obtained.

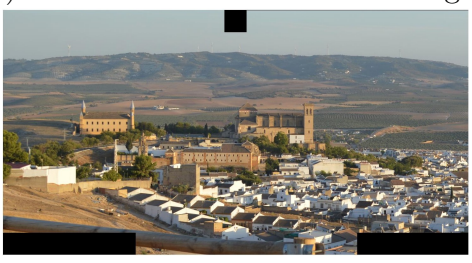
Figure 3.6: Result of centralized heuristic algorithm described in Algorithm 3 using 2 agents.



(a) Images captured by the agents.



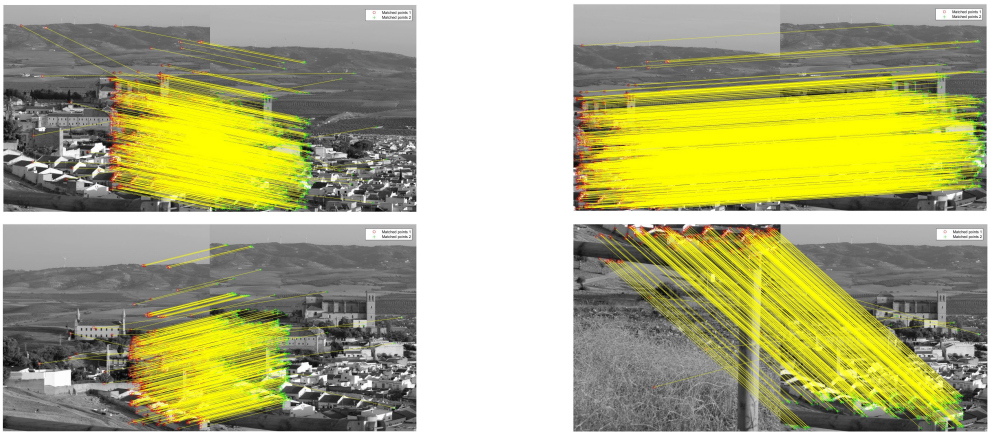
(b) Matched features between images.



(c) Panoramic image obtained.

Figure 3.7: Result of centralized heuristic algorithm described in Algorithm 3 using 3 agents.

(a) Images captured by the agents.

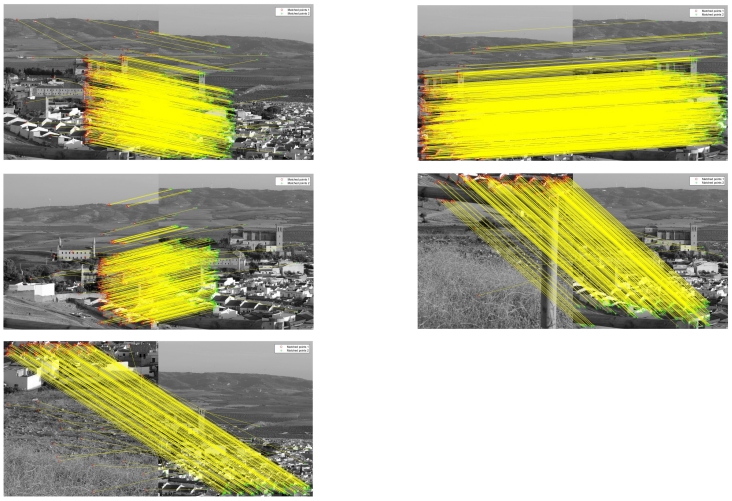


(b) Matched features between images.

(c) Panoramic image obtained.

Figure 3.8: Result of centralized heuristic algorithm described in Algorithm 3 using 4 agents.

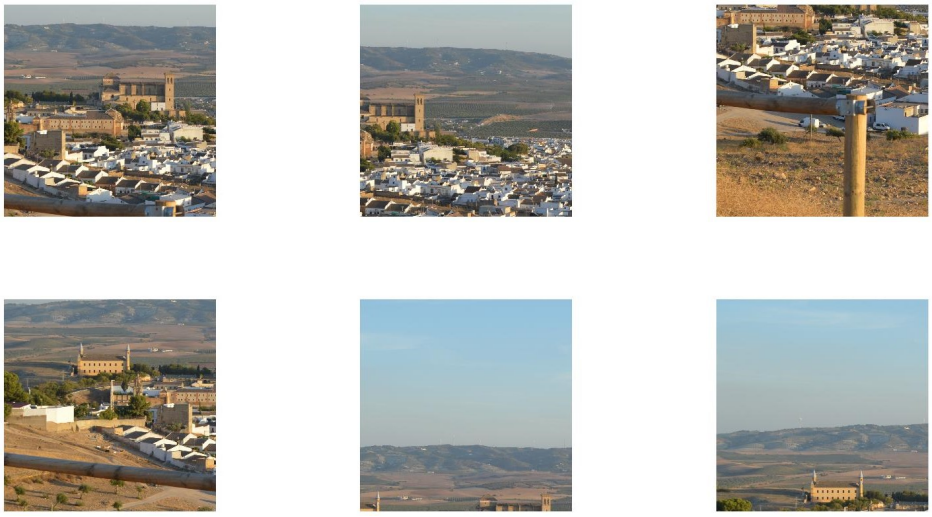
(a) Images captured by the agents.



(b) Matched features between images.

(c) Panoramic image obtained.

Figure 3.9: Result of centralized heuristic algorithm described in Algorithm 3 using 5 agents.



(a) Images captured by the agents.

(b) Panoramic image obtained.

Figure 3.10: Result of centralized heuristic algorithm described in Algorithm 3 using 6 agents.

(a) Images captured by the agents.

(b) Panoramic image obtained.

Figure 3.11: Result of centralized heuristic algorithm described in Algorithm 3 using 7 agents.

(a) Images captured by the agents.

(b) Panoramic image obtained.

Figure 3.12: Result of centralized heuristic algorithm described in Algorithm 3 using 8 agents.

KEY PERFORMANCE INDICATORS			
	Comp. Time	No. of pixels	Distance
2 agents	204.739 s	$3,037,862$	$2.2461x10^9$
3 agents	456.272 s	$4,038,720$	$3.3788x10^9$
4 agents	713.848 s	$5,231,801$	$4.8042x10^9$
5 agents	913.295 s	$6,343,962$	$6.1594x10^9$
6 agents	1105.306 s	$7,123,200$	$7.4157x10^9$
7 agents	1281.992 s	$7,713,160$	$8.3081x10^9$
8 agents	1477.277 s	$8,379,840$	$9.3428x10^9$

Table 3.3: Key Performance Indicators of the simulation of the simulations of the heuristic centralized algorithm described in Algorithm 3 (results in Figures from 3.6 to 3.12).

KEY PERFORMANCE INDICATORS			
	Comp. Time	No. of pixels	Distance
Small dimensions	4.655 s	$60,893$	$6.4129x10^6$
High dimensions	204.739 s	$3,037,862$	$2.2461x10^9$

Table 3.4: Key Performance Indicators of Algorithm 3 results with 2 agents.

(a) Images captured by the agents.

(b) Panoramic image obtained.

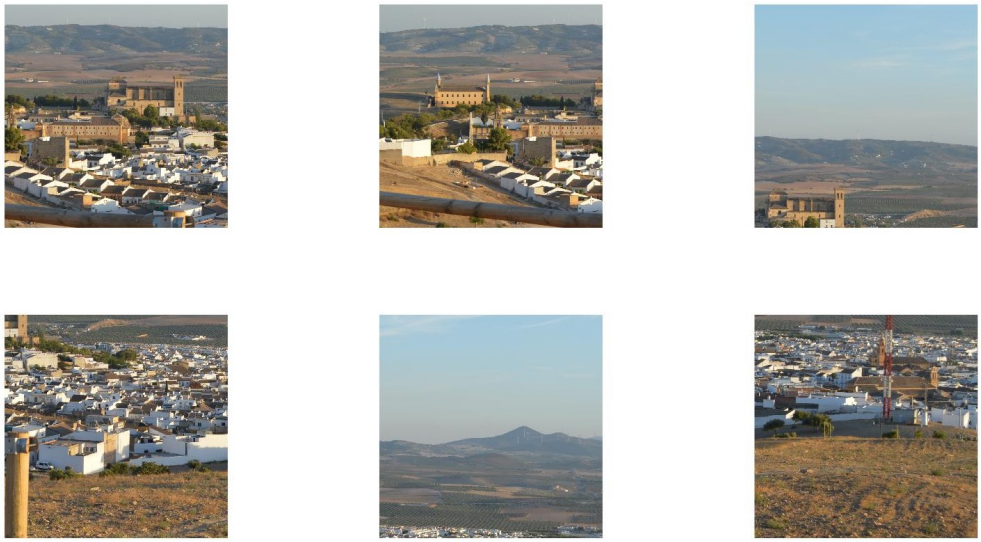
Figure 3.13: Result of simulation of the first modification of Algorithm 3 using 4 agents.

area was considered to create the sharp curve in which the neighbor positions were selected. Now, the map of captured pixel is not created. Therefore, the neighbor positions are selected in another way.

The simplification consists of each time a new agent has to be positioned, the neighboring positions to examine are only the eight that are around the area captured by the previous agent. This way, the previous agent area captured is the only one that has to be considered, decreasing the information to be recollected each time a new agent needs to be positioned.

This new algorithm was tested and simulated with 4, 6 and 8 agents, maintaining all the initial parameters with the same value as in the previous one. The results are shown in Figures 3.13, 3.14 and 3.15 respectively.

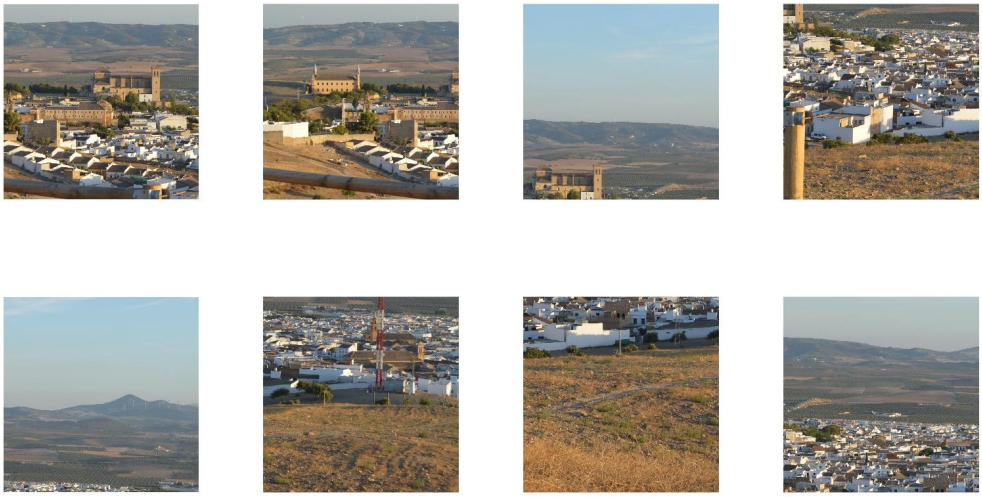
The values of the KPIs of the results are presented in Table 3.5 together



(a) Images captured by the agents.

(b) Panoramic image obtained.

Figure 3.14: Result of simulation of the first modification of Algorithm 3 using 6 agents.



(a) Images captured by the agents.

(b) Panoramic image obtained.

Figure 3.15: Result of simulation of the first modification of Algorithm 3 using 8 agents.

KEY PERFORMANCE INDICATORS			
	Comp. Time	No. of pixels	Distance
Results of Algorithm 3			
4 agents	713.848 s	$5,231,801$	$4.8042x10^9$
6 agents	1105.306 s	$7,123,200$	$7.4157x10^9$
8 agents	1477.277 s	$8,379,840$	$9.3428x10^9$
Results of the first version of Algorithm 3			
4 agents	164.135 s	$5,840,800$	$5.8548x10^9$
6 agents	233.243 s	$9,643,200$	$16.4220x10^9$
8 agents	291.686 s	$11,309,200$	$20.0690x10^9$

Table 3.5: Key Performance Indicators of the results obtained with the original version of Algorithm 3 and its first modification simulated with 2, 6 and 8 agents.

with the ones of the previous part. The principal benefit of this version is the great decrease of computational time, comparing with the previous algorithm results for 4, 6 and 8 agents. But in consequence, the sum of the distances to the center increases in a higher proportion than the number of pixels does. This fact can be translated in a loss of accuracy, i.e., the agents are more aleatory distributed in all the available area, not around the vision central area. Therefore, as it can be also verified in Figure 3.16, the final panorama image is not continuous using the modified algorithm. This problem takes more relevance when the number of agents increases, as it is possible to see in the same Figure 3.16.

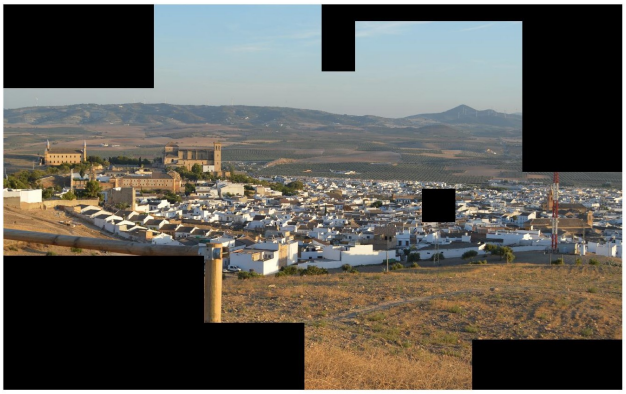
The other simplification made is related to the overlapping degree and consists of setting it as an initial parameter defined by the user. The neighboured positions are only the eight considered in the previous simplification. By this way, the total number of studied positions is decreased even more than before, because of each position has only one possibility of overlap in contrast to the initial algorithm, where several overlaps are considered for each position and selected the one that optimizes the cost function. This algorithm just studies the eight neighboured positions considering only the overlapping degree indicated at the beginning.

The algorithm has been tested using 4, 6 and 8 agents, and the results obtained from simulations are shown in Figures 3.17, 3.18 and 3.19 respectively. The overlapping degree is set in 40% for all the simulations. Regarding the cases of 4 and 6 agents, it would be possible to choose a lower percentage, but it is not the case for 8 agents because a higher number of matched

Original algorithm First simplification algorithm


(a) Panoramic images with 4 agents.


(b) Panoramic images with 6 agents.

(b) Panoramic images with 8 agents.

Figure 3.16: Comparison of panoramic images obtained with original version of Algorithm 3 and its first modification.

points between images is necessary to join with the rest the captured images by agent 7 and 8. Therefore, it has been selected 40% in order to set the same overlap for all the simulations. This is one of the main drawbacks of this simplification, that all the captured images have to consider the same overlapping degree.

If we compare the panoramic images obtained with 8 agents using the three algorithms, presented in Figure 3.20, it is obvious to state that the improvement in computational time implies a worsening in the characteristic of the panoramic image obtained. It can seem that the second modification gives a panoramic image with better characteristics, but in fact, it is worse because the overlapping degree is fixed, so the captured images taken are always the same, independently of the scene in which the agents are moving. Therefore there are probably some better solutions that are disregarded in this algorithm.

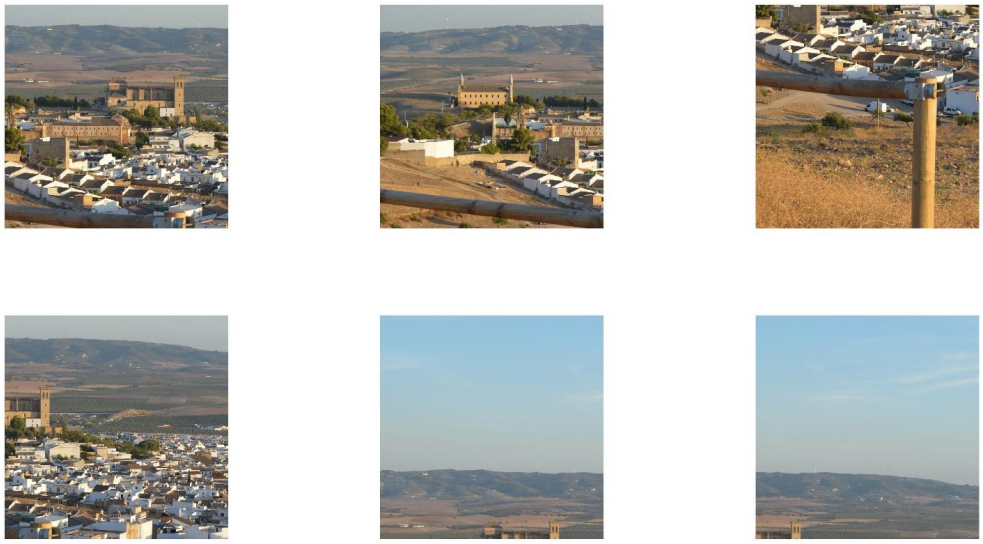
In Table 3.6 the values of KPIs of all the results of the 3 algorithms have been gathered to make possible the comparison. Looking at these values, the affirmation done in the previous paragraph can be based on them. Regarding the computational times, this solution is by far the one which most reduces the computational time required. But if we consider the total distance to the center related to the total number of pixels, the last results are better than the previous ones but not than the ones of Algorithm 3. In this point the main drawback of the last version done has to be considered, that makes it not as good as it can seem as it has been explained previously.

Having reached this point, the decision of which one choosing belongs to the designer once more time, depending on its necessities for the specific application and evaluating which one is the most important.

(a) Images captured by the agents.

(b) Panoramic image obtained.

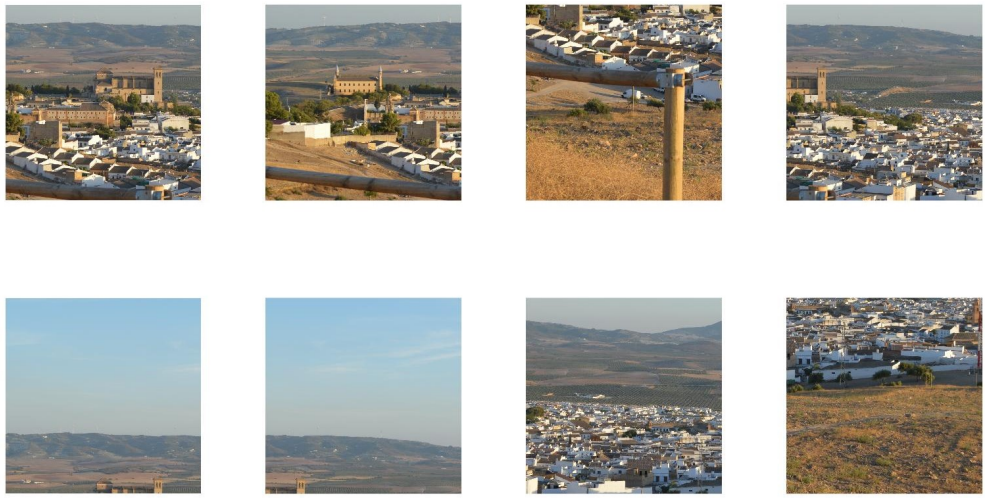
Figure 3.17: Result of simulation of the second modification of Algorithm 3 using 4 agents.



(a) Images captured by the agents.

(b) Panoramic image obtained.

Figure 3.18: Result of simulation of the second modification of Algorithm 3 using 6 agents.



(a) Images captured by the agents.

(b) Panoramic image obtained.

Figure 3.19: Result of simulation of the second modification of Algorithm 3 using 8 agents.

(a) Panoramic image of original version of Algorithm 3.

(b) Panoramic image of first modified algorithm.

(c) Panoramic image of second modified algorithm.

Figure 3.20: Comparison between panoramic images obtained with original version of Algorithm 3 and its first and second modifications using 8 agents.

KEY PERFORMANCE INDICATORS			
	Comp. Time	No. of pixels	Distance
Results of Algorithm 3			
4 agents	713.848 s	$5,231,801$	$4.8042x10^9$
6 agents	1105.306 s	$7,123,200$	$7.4157x10^9$
8 agents	1477.277 s	$8,379,840$	$9.3428x10^9$
Results of the first version of Algorithm 3			
4 agents	164.135 s	$5,840,800$	$5.8548x10^9$
6 agents	233.243 s	$9,643,200$	$16.422x10^9$
8 agents	291.686 s	$11,309,200$	$20.069x10^9$
Results of the second version of Algorithm 3			
4 agents	32.285 s	$5,488,000$	$5.2551x10^9$
6 agents	41.971 s	$7,369,600$	$7.7868x10^9$
8 agents	57.289 s	$10,192,000$	$14.736x10^9$

Table 3.6: Key Performance Indicators of the results obtained with the original version of Algorithm 3 and its first and second modifications simulated with 2, 6 and 8 agents.

Chapter 4

Distributed Algorithm Methods

In the virtual world, it can seem that centralized coordination is the easiest way to achieve our goal, but it is really unlikely having a system with those characteristics in the real world. Therefore, distributed coordination is the technique commonly use for real applications, and its principal difference between centralized coordination is that instead of having a leader who receives all the information from the agents and makes all the decision, each agent receives some information from its closer area and only makes the decision referred to itself.

Most articles found in the literature are related to this type of coordination and specifically focused on the way of how to solve the problems that it involves, which are no a few ones. For example, in the article [29], Ren, Beard and Atkins discuss problems appeared when the algorithm designed is applied to the real systems of UAVs, e.g. communication delays, relative information uncertainty or equilibrium state of the solution. In [13], the system studied is formed by a set of autonomous mobile robots which have only programmed their behaviors as a set of laws that guides the robot to react to environmental stimulus so that there is no explicit goal programmed in. The cooperation and the goals simply emerge as the computation goes on, and also the robots have no memory from the past. Therefore the instant visibility of each agent takes relevance, and the authors of this article studied the problem of having unlimited or limited visibility, comparing both results.

Distributed coordination has many fields of application, not only the one explained here for aerial robots. One example of mobile robot coordination can be found in [31], where the coordination is required in order to make possible the movement of many robots in the same space. Each robot has defined fixed independent paths, and the problem is to coordinate these mo-

tions to avoid collisions, i.e. path coordination. However, the article [27] is dedicated to the applications of aerial robots: environmental monitoring, surveillance, disaster management or mission planning among others. One of the fields cited in that paper is the cooperative aerial imaging, that is the same as we focus in this project.

In addition, some authors have solved this problem of coordination by applying some mathematical functions in order to get a determined geometric figure formed by the positions of the robots, as a simple line [15] or a circular shape [3]. More details of the processes can be found in the indicated articles.

4.1 Problem Setting

We assume that there is a set $\mathcal{N} = \{1, \ldots, N\}$ of mobile agents with the same characteristic than the ones explained in Section 3.1. The unique difference between the problem set in that Section and the one here is that some information about the system state is unknown at the moment of making a decision by each agent. There is no a unique operator, each agent is responsible for itself. Therefore, the agents have to reappraise the cost function each time a new information is exchanged with another robot.

The cost function considered for distributed approach has to be related to each agent, i.e., no global cost function can be calculated as a complete function, but each agent calculates its own cost function. Therefore, the elements involved in that function can only be referred to the information that each agent can obtain from its proximities.

Therefore, regarding these considerations we propose the optimization of the following cost:

$$J(x, y) = \alpha \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} \mathrm{MP}(i, j) + \beta \mathrm{NR}(x, y) + \gamma \mathrm{DP}(x, y) + \theta \mathrm{DR}(x, y).$$

where x and y are vectors that aggregate the corresponding local coordinates, i.e., $x = [x_i]_{i \in \mathcal{N}}$ and $y = [y_i]_{i \in \mathcal{N}}$; $\mathrm{MP}(i, j)$ stands for the matching points between agents i and j; NR represents the number of agents found in its proximities; $\mathrm{DP}(x, y)$ represents the distance between the actual agents and all the other agents found by it; and $\mathrm{DR}(x, y)$ the sum of the captured pixels values calculated as the distance from the central visual point. Finally, α, β, γ and θ are weights that can be tuned according to the designer goals.

4.2 Algorithm

The principle of this approach is different to the centralized one. In the centralized coordination, there was a single operator that had global information about the system state and who was the responsible for the movements of each agent. Whereas, in the distributed approach, there is not only a single operator, i.e., each agent is the unique responsible for its motion, and therefore, the one who makes the required decisions. This way, the agents involved in the distributed algorithms exchange its own information with the others, in order to get the most complex information of the situations as possible.

In this section, we offer two different distributed approaches to solving the proposed problem. The algorithms done are similar to the ones presented in Section 3.2 but referred to distributed coordination. In Subsection 4.2.1 an algorithm close to the optimal one is presented, and in Subsection 4.2.2 the algorithm developed is based on heuristic methods.

4.2.1 Optimal Algorithm

The original idea of creating this algorithm was to develop a general program whose result were the optimum. But considering distributed approach, this algorithm involves more complexity, and even some impossibilities sometimes. Regarding these considerations, this algorithm has been developed as similar as possible to the optimal one, although it is called the optimum in order to differentiate it with the one of the next subsection.

Two agents are only involved in the algorithm. Once both has been found by the other, their goal is to find the position that minimizes the cost function by starting to exchange information and calculating in both the value of the cost function each time a robot is moved. For this purposes, the agents are fitted with the required sensors to detect the presence of other agents in a determined space range and with the necessary system to exchanging information with the previously detected agents. Once the agents can communicate to each other, a coordination of motions is started in order to examine the cost each time an agent moves.

The algorithm is divided into two general parts. The first one is focused on the goal of looking for other agents in its proximities. The progress of the agent motion is forward the visual central point and this motion is stopped when the image captured by the agents can be merged. Then the second part starts, which tries to remove the agents looking for the combination with the lowest value of the cost function. The algorithm ends when none of the neighborhood combination of the studied positions of agents can minimize more the cost function. Therefore the actual combination is considered as

the optimum.

The cost function is calculated as the addition of some elements, in the same way of in the distributed algorithms. These elements are:

- The addition of the values of both captured images depending on the distance to the visual central point and considering increase of the values of the pixels recorded by the other agents.
- The absolute value of the distance between both agents.
- The inverse of the total number of matched points between images captured by both agents.

Each element is multiplied by a weight. Varying these value of possible to transform all the elements to the same number of integer digit in order to give the same important to all of them, or to give more relevance to just one element.

The algorithm just explained follows the steps outlined below:

1. Initial definition of the required parameters:
   a) Size of the total area of movements.
   b) Size of the captured image by agents.
   c) Number of agents involved in the situation and an object in Matlab for each one to save all its information.
   d) Initialize the matrix that represents the available space for simulation.
   e) Definition of the central vision point.
   f) Definition of the initial positions of agents (in opposites corners).
2. Simultaneously positioning of the agents in the available area:
   a) Initialization point:
      i. Define initial values of the coordinates variables of the agents.
      ii. Mark in the matrix of the space the initial positions of agents using their identifiers.
      iii. Creation of the matrix containing the absolute value of the distance of each pixel to the central vision point for each robot.
   b) Part 1: Motion of agents until their captured images can be merged:
      i. Each agent inspects its close area and records it.
      ii. Mark if any agent has been found in its close area:

A. If the agent is found:
   - Capture images by each agent, and continue only if their sizes are equal (in the other case they can not be merged with Algorithm 1). Extract the features of both images.
   - Update the values of the matrix that contains the value depending on the distance to the central point. Consider the position of the new agent found, varying gradually the value of the captured pixel by this agent: highest value for the ones of the center of the images, and the lowest for the edges.
   - Try to merge images. Only in the case of possible, calculate the value of cost function.
   - If the images can not be merged, update the position of both agents continue going through the central point. Keep the actual values as the previous one before changing. Update also the information in the matrix of the space, changing the old positions by the new ones.

B. If no agents are found: update the coordinate values in the same way as in the previous point when the merge of images has not been possible.

iii. Repeat these points the necessary times until the images can be merged. In this case, continue with next part.

c) Part 2: Look for the optimal solution:

i. Keep the actual position that is going to be tested as the previous values.

ii. Define the vector with possible position around the actual one for each agent. Consider only the actual position and the four ones of its neighborhood that only change one of the two coordinates an already defined increase. Therefore each vector contains 5 pairs of coordinates.

iii. Considering the vectors of neighborhood positions, calculate the value of the cost function for each one of the 25 possibilities, following the same steps as in the previous part for trying to merge images.

iv. If one of the possibilities studied obtains a lower value of the cost function than the original one, the values of coordinates are changed to these ones, and the steps repeated but for this new solution. In the case of the value of the cost function of the original position is the lowest, the optimum solution is this one and

no more calculations are done, updating the information contained by each robot.

3. Generate the panoramic image using Algorithm 1 detailed in Section 2.3.

Finally, Algorithm 4 details the program.

Algorithm 4: Distributed method with optimum solution

Initialize the size of the area of movements: $Mmax$ and $Nmax$;
Initialize the size of captured images: Mr and Nr;
Initialize $Nrobots$ as 2 and the vector R with their respective objects;
Initialize the matrix of real space mat_map as zeros.
Define the vision central point $(i0, j0)$;
Set $R(1).Pos\_ini$ and $R(2).Pos\_ini$ in opposite corners.
Initialize value of $i1$, $j1$, $i2$, and $j2$ as initial positions of agents.
Localize the agents in mat_map with identifiers 1 and 2.
Create the matrix $R(1).mat\_val\_ori$ and $R(2).mat\_val\_ori$ with values depending on the distance to $(i0, j0)$;
while $flag = 0$ **do**
    Define vision limits i_minV, i_maxV, j_minV, and j_maxV for both agents.
    Store in $R(1).Map\_rob$ and $R(2).Map\_rob$ the visible area of each agent. Functions for calculate $R(1).Num\_vec$ and $R(2).Num\_vec$ are defined in the robot objects.
    if $R(1).Num\_vec > 1$ **then**
        Obtain $image1$ and $image2$, and Im_size1 and Im_size2
        if $Im\_size1(1) = Mr$ **and** $Im\_size1(2) = Nr$ **and** $Im\_size2(1) = Mr$ **and** $Im\_size2(2) = Nr$ **then**
            Extract $features1$ and $features2$ from $image1$ and $image2$.
            Update in $R(1).Mat\_val\_prue$ and $R(2).Mat\_val\_prue$ the values of pixel captured by the other robot considering the distance to the central pixel of respective image.
            Try to match $features1$ with $features2$ and obtain $numMP$.
            if $numMP >= 10$ **then**
                Set $id\_enlace = 1$.
                Calculate $COST$.
            end if
        end if
        if $id\_enlace = 0$ **then**
            Set $i1\_prev = i1; j1\_prev = j1; i2\_prev = i2; j2\_prev = j2$.
            Update $i1$, $j1$, $i2$, and $j2$.

Update mat_map.
end if
else
Set $i1\_prev = i1; j1\_prev = j1; i2\_prev = i2; j2\_prev = j2$.
Update $i1$, $j1$, $i2$, and $j2$.
Update mat_map.
end if
if $id\_enlace = 1$ **then**
Set $flag = 1$.
end if
end while
Initialize P_vec1, P_vec2, $SOLUTIONS$, C_vec.
Define $incre\_i$ and $incre\_j$.
Set $case = 1$.
while $flag = 0$ **do**
Set $i1\_prev = i1; j1\_prev = j1; i2\_prev = i2; j2\_prev = j2$.
Define vectors P_vec1 and P_vec2 for each robot.
for $r1 = 1, \ldots, 5$ **do**
Set $i1 = P\_vec1(1, r1)$ and $j1 = P\_vec1(2, r1)$.
for $r2 = 1, \ldots, 5$ **do**
Set $i2 = P\_vec2(1, r1)$ and $j2 = P\_vec2(2, r1)$.
Update mat_map.
Define vision limits i_minV, i_maxV, j_minV, and j_maxV for both agents.
Store in $R(1).Map\_rob$ and $R(2).Map\_rob$ the visible area of each agent. Functions for calculate $R(1).Num\_vec$ and $R(2).Num\_vec$ are defined in the robot objects.
Obtain $image1$ and $image2$, and Im_size1 and Im_size2
if $Im\_size1(1) = Mr$ **and** $Im\_size1(2) = Nr$ **and** $Im\_size2(1) = Mr$ **and** $Im\_size2(2) = Nr$ **then**
Extract $features1$ and $features2$ from $image1$ and $image2$.
Update in $R(1).Mat\_val\_prue$ and $R(2).Mat\_val\_prue$ the values of pixel captured by the other robot considering the distance to the central pixel of respective image.
Try to match $features1$ with $features2$ and obtain $numMP$.
if $numMP >= 10$ **then**
Set $id\_enlace = 1$.
Calculate $COST$.
end if

60: **end if**
61: Update mat_map.
62: Set $C\_vec(case) = COST$.
63: Set $SOLUTIONS(:, caso) = [i1; j1; i2; j2]$.
64: Update $case$.
65: **end for**
66: **end for**
67: Set $cost\_min = Inf$.
68: **for** $case = 1, \ldots, 25$ **do**
69: **if** $C\_vec(case) < cost\_min$ **then**
70: Set $cost\_min = C\_vec(case)$.
71: Update $i1$, $j1$, $i2$, and $j2$.
72: **end if**
73: **end for**
74: **if** $i1 = i1\_prev$ **and** $j1 = j1\_prev =$ **and** $i2 = i2\_prev$ **and** $j2 = j2\_prev$ **then**
75: Set $flag = 1$.
76: **else**
77: Initialize P_vec1, P_vec2, $SOLUTIONS$, C_vec.
78: Set $case = 1$.
79: **end if**
80: **end while**
81: Store final information in $R(1)$ and $R(2)$.
82: Create *panorama* using Algorithm 1;

Regarding the algorithm is possible to check that the result obtained can not be the optimum. The real optimal solution would be found if all the possible combinations are studied. In the case of the centralized algorithm, explained in Subsection 3.2.1, all the possible combinations of positions were tested due to a global operator directed the motions. But in the case of this distributed algorithm, doing the same is more complicated. This process designed as the optimum considers the possible combinations around a potential solution, searching if any of these combinations have a better solution than the already found. Therefore, this is not a real optimal algorithm even calling it as *the optimal algorithm*.

4.2.2 Heuristic method

The optimum algorithm has only been created for 2 agents due to the increase in complexity involved when we use more agents. In order to make possible

the study of a system with more than 2 agents, an algorithm based on the heuristic method has been developed. Using the heuristic method implies that the result obtained may not be the optimum one, but usually, its solution is good enough for a specific application.

There are some differences between the optimum algorithm detailed in Section 4.2.1 and the heuristic one developed in this Section. The main ones are the following:

- Now, the agents are located one by one, instead of simultaneously. Therefore, each agent is positioned exchanging information with the agents previously located.
- There is the possibility that one agent image merges with more than one images of already positioned robots. Therefore, the number of agents found is included in the cost function.
- Each time an agent can merge its image with any other, it continues moving through the central points looking for other position with a lower cost value.

Therefore, regarding these considerations the algorithm developed follows the next steps:

1. Initial definition of the required parameters:
   a) Size of the total area of movements.
   b) Size of the captured image by agents.
   c) Number of agents involved in the situation and an object in Matlab for each one to save all its information.
   d) Initialize the matrix that represents the available space for simulation.
   e) Definition of the central vision point.
   f) Initially positioning of the agents in the edges of the area randomly.
2. Position the agents one by one following the same :
   a) Initialization point:
      i. Define the initial values of the coordinates of the agent.
      ii. Creation of the matrix containing the absolute value of the distance of each pixel to the central vision point for each robot.
   b) Motion of the agent until its captured image can be merged with other with minimum cost:

i. The agent inspects its close area and stores it.

ii. Mark if any agent has been found in its close area, and store them.

iii. If any agents are found, or the position studied is one of the neighborhood:

A. Capture image, and continue only if its size is equal to the standard (in the other case it can not be merged using Algorithm 1). Also, extract the features of the image.

B. In the case of any agents have been found, and for each of them:

- Update its own map of the space with the position of the new agent found
- Update the values of the matrix that contains the value depending on the distance to the central point. Consider the position of the new agent found, varying gradually the value of the captured pixel by this agent: highest value for the ones of the center of the images, and the lowest for the edges.
- Try to merge images. Only in the case of possible, calculate the value of cost function.

C. In the case of not studying neighborhood position and if the image has been merged with any other, start examining the four position of its neighborhood (up and down, and left and right positions) considering a fix pixels increase. All the previous steps are repeated for each one of these positions. Finally, the position of the agents is updated to the one of minimum cost.

D. If the image can not be merged with anyone, update the position of the agents by continuing going through the central point, or going back if it has captured an already captured pixel by another agent. Keep the actual values as the previous one before changing.

iv. If no agents are found: update the coordinate values in the same way as in the previous point when the merge of images has not been possible.

v. Repeat these points the necessary times until the robot can not move to a position with minimum cost in its neighborhood.

c) Update the information of the matrix that represents the real space.

d) Update the information of the actual robots.

3. Generate the panoramic image using Algorithm 1 detailed in Section 2.3.

Finally, Algorithm 5 details the program.

Algorithm 5: Distributed method with heuristic method

Initialize the size of the area of movements: $Mmax$ and $Nmax$;
Initialize the size of captured images: Mr and Nr;
Initialize $Nrobots$ and the vector R with their respective objects;
Initialize the matrix of real space mat_map as zeros.
Define the vision central point $(i0, j0)$;
Set $R(1).Pos\_ini$ and $R(2).Pos\_ini$ in the edges randomly.
for $robot = 1, \ldots, Nrobots$ **do**
    Initialize value of i and j as the initial position of the agent.
    Initialize its own map of the available space, $map\_surround$.
    Create the matrix mat_val with values depending on the distance to $(i0, j0)$;

    Initialize P_vec, C_vec, and Enl_vec.
    Define $incre\_i$ and $incre\_j$.
    Set $neighb = 0$, $cost\_min = Inf$ and $id\_enlace = 0$.
    while $flag = 0$ **do**
        Define vision limits i_minV, i_maxV, j_minV, and j_maxV. Store in map_rob the visible area of the agent.
        Calculate num_vec and vector id_robs.
        if $num\_vec >= 1$ **or** $neighb \neq 0$ **then**
            Obtain $image$ and Im_size.
            if $Im\_size(1) = Mr$ **and** $Im\_size(2) = Nr$ **then**
                Extract $features$ from $image$
                if $num\_vec >= 1$ **then**
                    for $case = 1, \ldots, num\_vec$ **do**
                        Update $map\_surround$ and mat_val if necessary.
                        Try to match $features$ with $R(id\_robs(case)).features$ and obtain $numMP$.
                        if $numMP >= 10$ **then**, Set $id\_enlace = 1$.
                        end if
                    end for
                    if $id\_enlace = 1$ **then**
                        Calculate sum_val.
                        if $sum\_val = Inf$ **and** $neighb = 0$ **then**, Set $flag\_val\_inf = 1$.
                        end if
                        Calculate $COST$.

34: **end if**
35: **end if**
36: **end if**
37: **if** $flag\_val\_inf = 0$ **then**
38: **if** $neighb = 0$ **and** $id\_enlace = 1$ **then**
39: Calculate elements of P_vec.
40: Update i_prev and j_prev, i and j, $cost\_min$ and $neighb$.
41: **else if** $neighb = 1$ **then**
42: Update i and j, $C\_vec(neighb)$, $Enl\_vec(neighb)$ and $neighb$.
43: **else if** $neighb = 2$ **then**
44: Update i and j, $C\_vec(neighb)$, $Enl\_vec(neighb)$ and $neighb$.
45: **else if** $neighb = 3$ **then**
46: Update i and j, $C\_vec(neighb)$, $Enl\_vec(neighb)$ and $neighb$.
47: **else if** $neighb = 4$ **then**
48: Update $C\_vec(neighb)$ and $Enl\_vec(neighb)$.
49: Calculate sum_enl and look for minimum value in C_vec.
50: Update i and j, and $neighb$.
51: **end if**
52: **end if**
53: **if** $flag\_val\_inf = 1$ **then**
54: Set $i\_prev = i$ and $j\_prev = j$.
55: Update i and j.
56: Set $flag\_val\_inf = 0$ and $id\_enlace = 0$.
57: **else if** ($neighb = 0$ **and** $id\_enlace = 0$) **or** ($neighb = 5$ **and** $sum\_enl = 0$) **then**
58: Set $i\_prev = i$ and $j\_prev = j$.
59: Update i and j.
60: **end if**
61: **if** $neighb = 5$ **then**
62: Initialize P_vec, C_vec and Enl_vec.
63: Set $neighb = 0$ and $sum\_enl = 0$.
64: **end if**
65: **else if** $neighb = 0$ **then**
66: Set $i\_prev = i$ and $j\_prev = j$.
67: Update i and j.
68: **end if**

69: **if** (i=i0 **and** j=j0) **or** (i=i_prev **and** j=j_prev) **then**
70: Set $flag = 1$.
71: **end if**
72: **end while**
73: Store final information in $R(robot)$.
74: **end for**
75: Create *panorama* using Algorithm 1;

4.3 Simulation results

4.3.1 Results of the algorithm with optimum solution

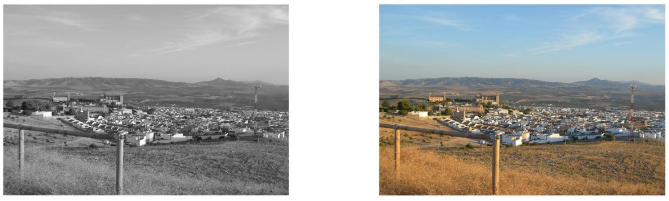
The algorithm has been developed for 2 agents, as it has been mentioned in Section 4.2.1. The values defined for the initial parameters are maintained the same as in Section 3.3.2 when using more than 2 agents, except the size of the captured images, which is smaller. This drop in the image size is due to more available motion space is needed in this algorithm to allow more possibilities of locations. All the values set are shown below:

- The size of the area: $4000x6016$.
- The size of captured images: $500x500$.
- The central point: $i = 2100$ and $j = 2100$, the same of Figure 3.5.

The agents considered for this process can exchange information with other. These agents have to be inside the visual area of the other agent to allow the communication. For these simulations, the size of this area is considered as double of the size of captured images: $8000x12032$.

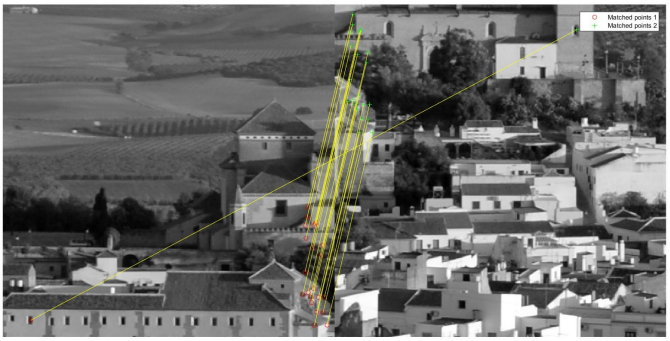
Some *Key Performance Indicators* (KPIs) have been calculated in each simulation. They are the same defined in the previous Chapter: i) computational time, ii) the total number of pixels of the generated panoramic image, and iii) the addition of the distances from all the captured pixels to the visual central point.

The results of the simulations are different if the weight of the elements included in the cost function change. If they are set in order to give the same importance to all the elements, the result obtained is presented in Figure 4.1 and its KPIs in Table 4.1. However, if the weights are set in order to give more importance to the total number of matched points between images, the result presented in Figure 4.2 differs to the previous one. Its KPIs are shown in Table 4.2.



(a) Available area represented by the whole image in gray scale and in color.

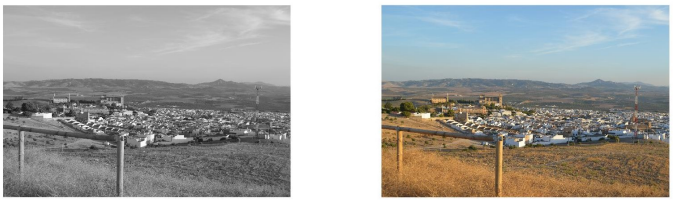
(b) Images captured by the agents.



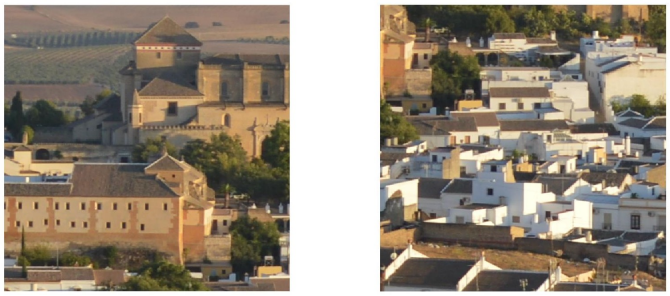
(c) Matched features between images.

(d) Panoramic image obtained.

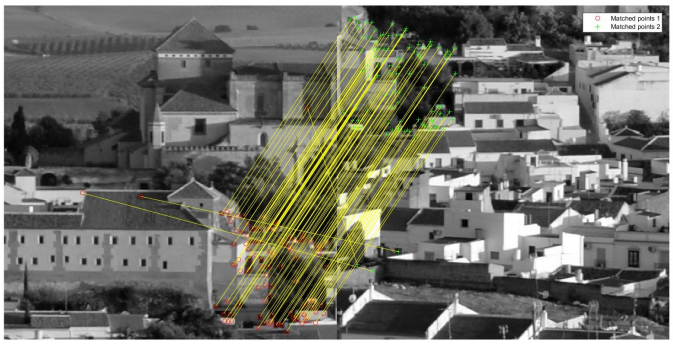
Figure 4.1: Result of the optimal distributed algorithm described in Algorithm 4 (first simulation).



(a) Available area represented by the whole image in gray scale and in color.



(b) Images captured by the agents.



(c) Matched features between images.

(d) Panoramic image obtained.

Figure 4.2: Result of the optimal distributed algorithm described in Algorithm 4 (second simulation).

KEY PERFORMANCE INDICATORS		
Computational Time	Number of pixels	Distance to central point
140.160 s	485720	$1.5269x10^8$

Table 4.1: Key Performance Indicators of the first simulation of the optimal distributed algorithm described in Algorithm 4 (results in Figure 4.1).

KEY PERFORMANCE INDICATORS		
Computational Time	Number of pixels	Distance to central point
227.853 s	459470	$1.4389x10^8$

Table 4.2: Key Performance Indicators of the second simulation of the optimal distributed algorithm described in Algorithm 4 (results in Figure 4.2).

Comparing the results obtained from both simulations, and specifically the value of its KPIs, it is possible to check that in the second case, the panoramic image obtained contains a lower number of pixels than the first one, i.e., more overlap is obtained using the second cost. This can be a benefit if considering the case of relocation when an agent faults or a disadvantage if considering the highest magnitude of the panoramic image as possible. Therefore, the weights of elements in the cost function have to be designed depending on the specific application necessities.

4.3.2 Results of the algorithm with heuristic method

Comparison between the algorithm and the optimal solution

Firstly, the heuristic algorithm has been tried using just two agents in order to compare its result with the optimum. The values of the initial parameters are the same of in the previous subsection. The initial position of the agents is supposed to be randomly, but in this case, is set always at the same points in order to make possible the comparison. These positions are:

- Robot 1: $i = 1733$; $j = 6016$.

- Robot 2: $i = 1$; $j = 5097$.

Two simulations have been done changing the values of weight in the cost function, as it has been done with the optimum algorithm. The result obtained giving the same importance to all the elements of the cost is presented in Figure 4.3. And the result of the second one where more importance is

KEY PERFORMANCE INDICATORS			
	Comp. Time	No. pixels	Distance
Optimal solution 1	140.160 s	485720	$1.5269x10^8$
Heuristic solution 1	64.144 s	480650	$1.6007x10^8$

Table 4.3: Key Performance Indicators of the first simulation of the heuristic distributed algorithm described in Algorithm 5 (results in Figure 4.3) compared with Table 4.1.

KEY PERFORMANCE INDICATORS			
	Comp. Time	No. pixels	Distance
Optimal solution 2	227.853 s	459470	$1.4389x10^8$
Heuristic solution 2	66.576 s	461192	$1.5389x10^8$

Table 4.4: Key Performance Indicators of the second simulation of the heuristic distributed algorithm described in Algorithm 5 (results in Figure 4.4) compared with Table 4.2.

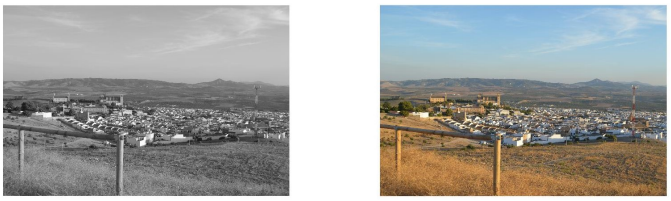
given to the number of matched points is presented in Figure 4.4. Their KPIs are shown is Tables 4.3 and 4.4 respectively, including also the values corresponding to the optimal result.

Regarding the obtained values of KPIs of each simulation, the difference between computational times of heuristic and optimum algorithms is the most significant in both tables, being lower the corresponding to the heuristic algorithm. The values of the other two KPIs are very similar to both algorithm, even obtained different panoramic images with each one.

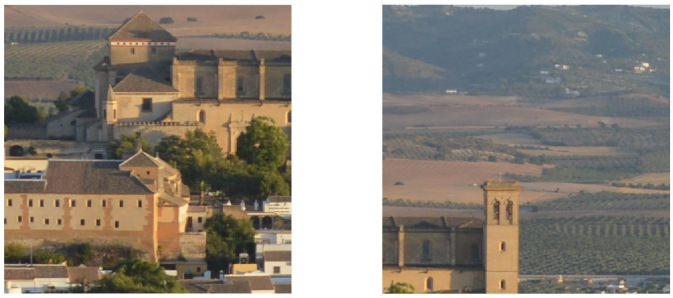
One notable aspect of the heuristic algorithm is that the computational time is maintained similar in the two simulations, whereas in the optimum algorithm there is an important increase of time when considering the number of matched points as more relevant for the cost. This is an advantage of the heuristic method: the time required to simulate is maintained in a short range no matter the cost function selected.

Simulation with various agents

The heuristic method developed has the advantage of considering the situations with more than 2 agents involved. Therefore, some simulations have been done increasing the number of agents to test its functionality. The initial positions of the agents are selected randomly, so each simulation considers the agents initially located at different points of the area edges. The



(a) Available area represented by the whole image in gray scale and in color.

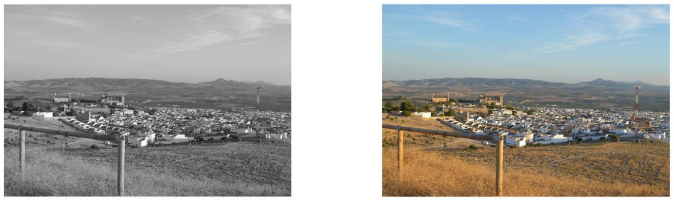


(b) Images captured by the agents.

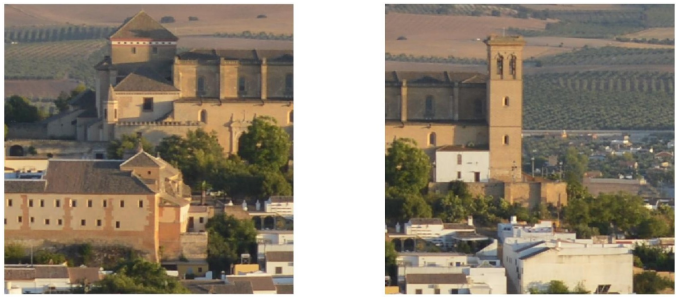
(c) Matched features between images.

(d) Panoramic image obtained.

Figure 4.3: Result of the heuristic distributed algorithm described in Algorithm 5 (first simulation).



(a) Available area represented by the whole image in gray scale and in color.



(b) Images captured by the agents.

(c) Matched features between images.

(d) Panoramic image obtained.

Figure 4.4: Result of the heuristic distributed algorithm described in Algorithm 5 (second simulation).

KEY PERFORMANCE INDICATORS			
	Comp. Time	No. pixels	Distance
2 agents	64.925 s	$782,192$	$1.8108x10^8$
3 agents	80.713 s	$726,739$	$3.1880x10^8$
4 agents	93.929 s	$920,715$	$3.8338x10^8$
5 agents	115.171 s	$1157,045$	$6.5399x10^8$
6 agents	124.288 s	$1372,443$	$10.7400x10^8$

Table 4.5: Key Performance Indicators of the heuristic distributed algorithm described in Algorithm 5 using different number of agents.

other parameters are set with the same values as before.

The cost function considered for all the simulation done is only the one that gives the same relevance to all the elements. No other cost has been tested due to the difference found in previous part between the two cost functions considered is not very relevant.

The results obtained are presented in Figures 4.5, 4.6, 4.7, and 4.8 for 3, 4, 5, and 6 agents respectively. And in Table 4.5 all the KPIs have been compiled together for all the cases studied, includying the case of 2 agents develop in the previous part.

Analyzing the results by looking at 4.5, the most relevant KPIs may be the sum of the distances of all the pixels to the central point. From the simulation of 5 agents, this value increases in a higher proportion than in previous ones. This increase could be due to the increase of the total number of pixel. However, considering also the panoramic images obtained in each case, it is possible to conclude that this higher increase may be because the last agents located are positioned farther from the visual central point. The other two KPIs maintain its values in a logical range when the number of agents increases.

Looking at the panoramic images of Figures 4.7 and 4.8, it is clearly seen that the heuristic algorithm developed is not very efficient in the case of having more than 3 or 4 agents. The agents move forward the visual central point direction and usually stop going when another agent is found. The initial position of the agents takes relevance in this algorithm because depending on this position, each agent will be moved in one direction. These positions are chosen randomly at the beginning, therefore the final panoramic image varies from each simulation, e.g., the panoramic image in Figure 4.8 looks like some robots have been initially positioned in close points, but other panoramic images would have been produced in case of father initial positions. Therefore, regarding this principal drawback of the heuristic algorithm developed, some

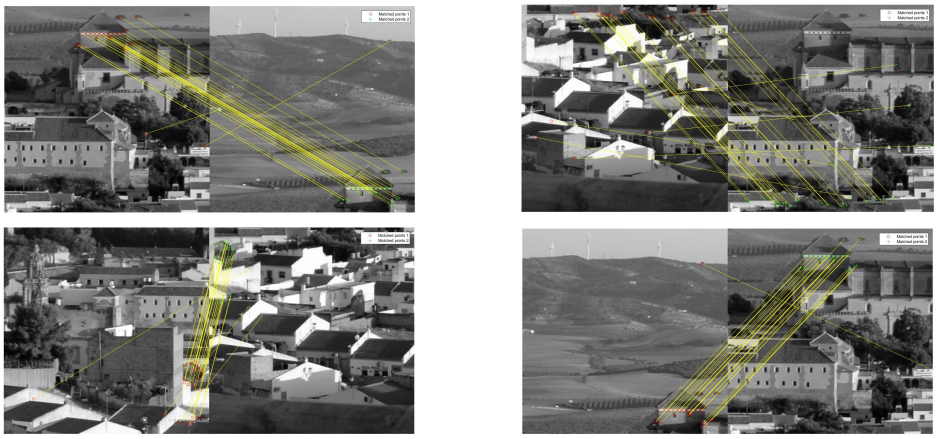
(a) Images captured by the agents.

(b) Matched features between images.

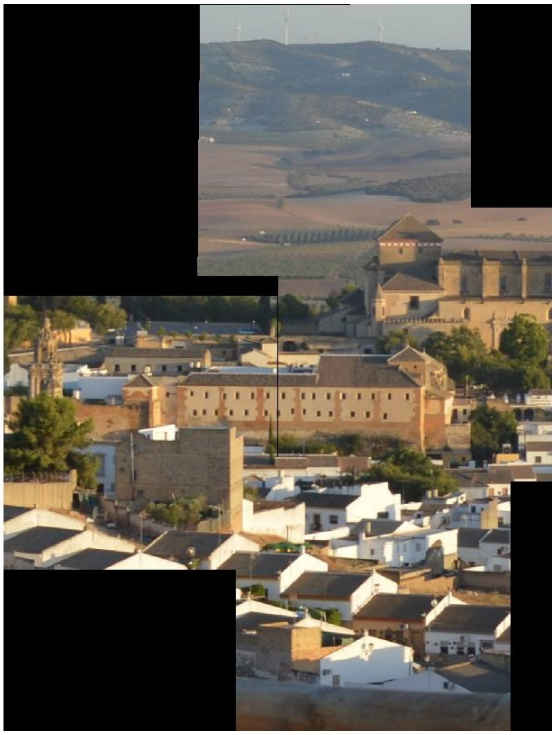
(c) Panoramic image obtained.

Figure 4.5: Result of the heuristic distributed algorithm described in Algorithm 5 using 3 agents.

(a) Images captured by the agents.

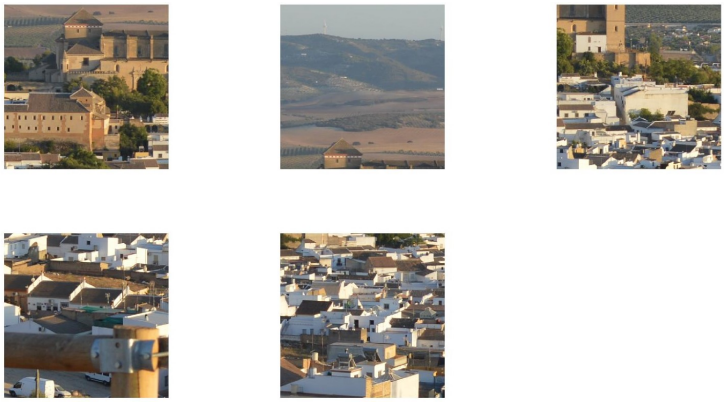


(b) Matched features between images.

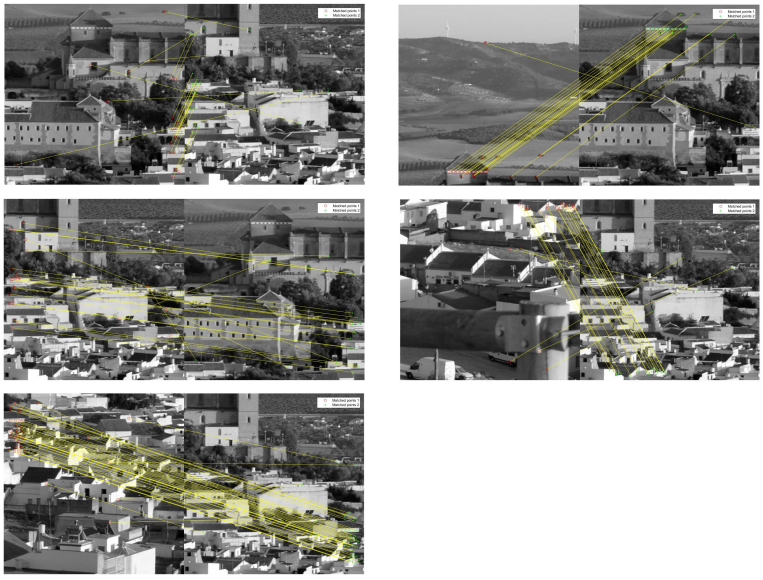


(c) Panoramic image obtained.

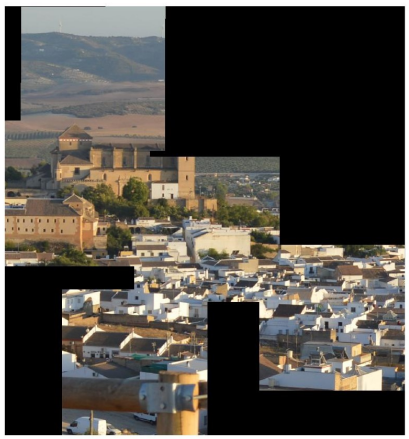
Figure 4.6: Result of the heuristic distributed algorithm described in Algorithm 5 using 4 agents.



(a) Images captured by the agents.

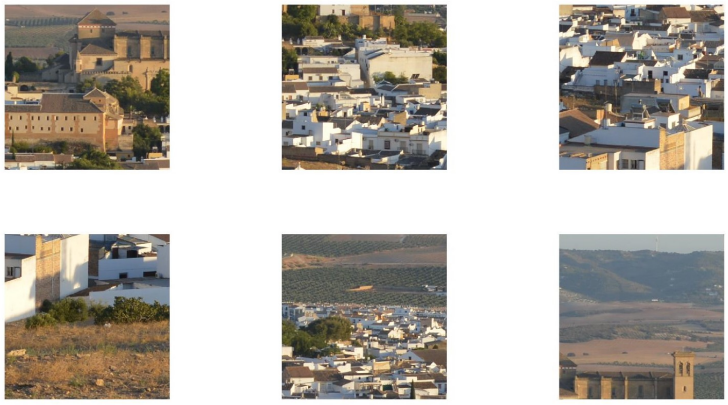


(b) Matched features between images.

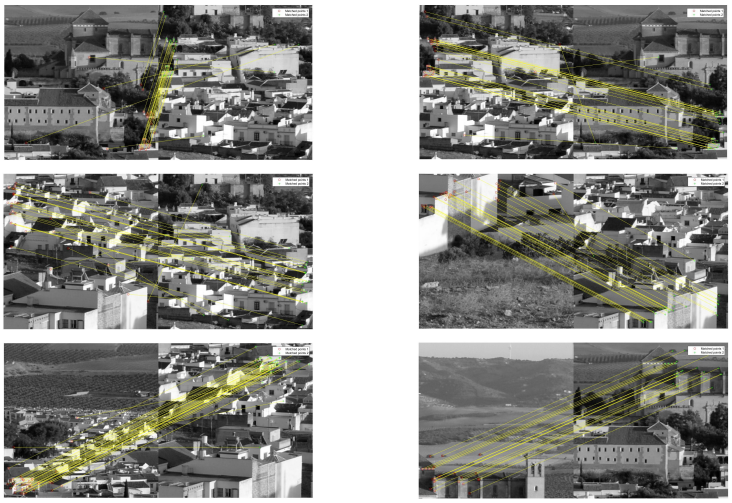


(c) Panoramic image obtained.

Figure 4.7: Result of the heuristic distributed algorithm described in Algorithm 5 using 5 agents.



(a) Images captured by the agents.



(b) Matched features between images.

(c) Panoramic image obtained.

Figure 4.8: Result of the heuristic distributed algorithm described in Algorithm 5 using 6 agents.

improvements could have been carried out in order to make it more suitable for various agents.

Chapter 5

Conclusions

In this work, we have presented some algorithms to achieve panoramic images from some images captured by individual coordinated agents. We have considered a system that consists of many aerial mobile agents, each one equipped with cameras, whose motions are restricted to XY plane. The main issue studied in this project has been the location of the agents in the available area in order to get the panoramic image with some specific characteristics.

Firstly, some algorithms have been developed using centralized coordination between the agents. Principally two results have been compared, the optimal and the heuristic results, concluding that the second one could be more suitable for our project due to the short computational time required. Some modifications of the heuristic algorithm have been done with the goal of continue decreasing the computational time, but finding that some characteristics had been gotten worse, e.g., centralization in the center point.

Secondly, the same problem has been solved using distributed coordination. The optimal and heuristic results have been compared in the same way of in centralized algorithm, even though the optimal algorithm developed was not exactly the optimum. Considering the specific characteristics required, different results have been obtained, concluding that for general purposes, the heuristic algorithm supplies better results than the optimal one if high accuracy is dispensable.

Future work

New algorithms for distributed coordination. In the real life, robots are generally coordinated using distributed approach. Therefore, future work on this topic would be interesting to bring the situation closer to real-world characteristics.

3D Systems. Another interesting point to be investigated is to transform all the work done for 2D to 3D system. The agents could move in XYZ space, including the new coordinate Z for each agent, and therefore, a new variable to be considered. Moreover, the change in the orientation of the cameras can be considered, allowing the agents rotation.

UAVs control. Before bringing this results to the real world, it would be necessary to research more about the way of controlling UAVs. In the present work, this point has been considered as known by the designer, and no attention has been paid to it.

Real Implementation. The final objective of this project would be the implementation of all the developed algorithms in a real test bed. This real implementation would be the crowning point of this project.

Bibliography

[1] Tim Bailey. *Mobile robot localisation and mapping in extensive outdoor environments*. PhD thesis, Citeseer, 2002.

[2] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[3] Lara Brinón-Arranz and Luca Schenato. Consensus-based source-seeking with a circular formation of agents. *European Control Conferenc e*, pages 2831–2836, 2013.

[4] Matthew Brown and David G Lowe. Recognising panoramas. In *ICCV*, volume 3, page 1218, 2003.

[5] Matthew Brown and David G Lowe. Automatic panoramic image stitching using invariant features. *International journal of computer vision*, 74(1):59–73, 2007.

[6] Wolfram Burgard, Mark Moors, Cyrill Stachniss, and Frank E Schneider. Coordinated multi-robot exploration. *IEEE Transactions on robotics*, 21(3):376–386, 2005.

[7] David Capel and Andrew Zisserman. Automated mosaicing with super-resolution zoom. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pages 885–891. IEEE, 1998.

[8] José A Castellanos, Juan D Tardós, and Günther Schmidt. Building a global map of the environment of a mobile robot: The importance of correlations. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 2, pages 1053–1059. IEEE, 1997.

[9] Xiaolong Dai and Siamak Khorram. A feature-based image registration algorithm using improved chain-code representation combined with invariant moments. *IEEE Transactions on Geoscience and Remote Sensing*, 37(5):2351–2362, 1999.

[10] Raffaello D'Andrea. Guest editorial can drones deliver? *IEEE Transactions on Automation Science and Engineering*, 11(3):647–648, 2014.

[11] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[12] Alessandro Farinelli, Luca Iocchi, and Daniele Nardi. Multirobot systems: a classification focused on coordination. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(5):2015–2028, 2004.

[13] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Peter Widmayer. Distributed coordination of a set of autonomous mobile robots. In *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*, pages 480–485. IEEE, 2000.

[14] Marc Forstenhaeusler, Riku Funada, Takeshi Hatanaka, and Masayuki Fujita. Experimental study of gradient-based visual coverage control on so (3) toward moving object/human monitoring. In *2015 American Control Conference (ACC)*, pages 2125–2130. IEEE, 2015.

[15] Samratul Fuady and Hideaki Ishii. Alignment forming in source seeking for multi-agent systems.

[16] Brian P Gerkey and Maja J Mataric. Sold!: Auction methods for multirobot coordination. *IEEE transactions on robotics and automation*, 18(5):758–768, 2002.

[17] Natasha Govender. Evaluation of feature detection algorithms for structure from motion. *Council for Scientific and Industrial Research, Pretoria, Technical Report*, 2009.

[18] Nuno Gracias, Mohammad Mahoor, Shahriar Negahdaripour, and Arthur Gleason. Fast image blending using watersheds and graph cuts. *Image and Vision Computing*, 27(5):597–607, 2009.

[19] Alfred Haar. On the theory of orthogonal function systems. *Math. Ann*, 69(3):331–371, 1910.

[20] Nagham Hamid, Abid Yahya, R Badlishah Ahmad, and Osamah M Al-Qershi. A comparison between using sift and surf for characteristic region based image steganography. *International Journal of Computer Science Issues*, 9(33-3):110–116, 2012.

[21] Michal Irani and P Anandan. About direct methods. In *International Workshop on Vision Algorithms*, pages 267–277. Springer, 1999.

[22] Luo Juan and Gwun Oubong. Surf applied in panorama image stitching. In *Image Processing Theory Tools and Applications (IPTA), 2010 2nd International Conference on*, pages 495–499. IEEE, 2010.

[23] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[24] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[25] Philip F McLauchlan and Allan Jaenicke. Image mosaicing using sequential bundle adjustment. *Image and Vision computing*, 20(9):751–759, 2002.

[26] Alec Mills and Gregory Dudek. Image stitching with dynamic elements. *Image and Vision Computing*, 27(10):1593–1602, 2009.

[27] Markus Quaritsch, Emil Stojanovski, Christian Bettstetter, Gerhard Friedrich, Hermann Hellwagner, Bernhard Rinner, Michael Hofbaur, and Mubarak Shah. Collaborative microdrones: applications and research challenges. In *Proceedings of the 2nd International Conference on Autonomic Computing and Communication Systems*, page 38. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[28] Vladan Rankov, Rosalind J Locke, Richard J Edens, Paul R Barber, and Borivoj Vojnovic. An algorithm for image stitching and blending. In *Biomedical Optics 2005*, pages 190–199. International Society for Optics and Photonics, 2005.

[29] Wei Ren, Randal W Beard, and Ella M Atkins. A survey of consensus problems in multi-agent coordination. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 1859–1864. IEEE, 2005.

[30] Eric JM Rignot, Ronald Kowk, John C Curlander, and Shirley S Pang. Automated multisensor registration: Requirements and techniques. *Photogrammetric Engineering & Remote Sensing*, 57(8), 1991.

[31] Thierry Siméon, Stéphane Leroy, and J-P Lauumond. Path coordination for multiple mobile robots: A resolution-complete algorithm. *IEEE Transactions on Robotics and Automation*, 18(1):42–49, 2002.

[32] Richard Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 2(1):1–104, 2006.

[33] Richard Szeliski and Sing Bing Kang. Direct methods for visual scene reconstruction. In *Representation of Visual Scenes, 1995.(In Conjuction with ICCV'95), Proceedings IEEE Workshop on*, pages 26–33. IEEE, 1995.

[34] AD Ventura, Anna Rampini, and Raimondo Schettini. Image registration by recognition of corresponding structures. *IEEE Transactions on Geoscience and Remote Sensing*, 28(3):305–314, 1990.

Coordinación de robots para la formación de imágenes panorámicas colaborativas

Resumen en Español

Autor: María Teresa Araúz Pisón
Tutor: José María Maestre Torreblanca

Sevilla, 2016

Resumen

En este proyecto se trata el problema de la coordinación de un conjunto de agentes autónomos, cada uno de ellos equipado con una cámaras, cuyo objetivo es la obtención de una imagen panorámica. Para este fin, se optimiza una función objetivo de costes múltiples que representa, por ejemplo, el número de puntos emparejados entre las imágenes, la distancia entre los agentes o la distancia hasta el punto central de visión. Se presentan algunas soluciones para de problema basándonos en dos enfoques diferentes: las coordinaciones centralizada y distribuida de los agentes. Para cada uno de ellos, los resultados se obtienen utilizando tanto métodos de optimización como métodos heurísticos. Las comparaciones entre todos los resultados obtenidos se realizan a partir de las simulaciones de los algoritmos que se desarrollan durante este proyecto.

Índice general

Capítulo 1

Introducción

El principal problema que se pretende resolver en este proyecto es la coordinación de una serie de robots, cada uno de ellos equipado con una cámara, con el objetivo de generar una imagen panorámica global con ciertas propiedades en función de las imágenes locales [29]. Este es un tema de enorme interés práctico debido a que los resultados de este proyecto de investigación pueden ser transferidos para su implementación en las flotas de vehículos aéreos no tripulados (*drones*), que actualmente gozan de una gran popularidad debido a sus numerosas aplicaciones, por ejemplo en servicios de mensajería [10] y en sistemas de información geográfica, entre muchos otros. Además, este tema representa un reto de investigación que ha sido ya identificado por la comunidad científica [24].

Hoy en día, la generación de imágenes panorámicas se ha convertido en una característica estándar en la mayoría de las cámaras comerciales. Como se ha indicado en el párrafo anterior, esto se consigue por medio de un procedimiento conocido como *fusión de imágenes*, que consiste en la combinación de múltiples imágenes con campos de visión superpuestos para generar imágenes panorámicas. Con este fin, la detección de rasgos distintivos en el conjunto de imágenes se utiliza como un medio para determinar las coordenadas de píxeles entre imágenes diferentes que pueden estar relacionados y utilizados para estimar las alineaciones correspondientes.

En paralelo a la consolidación de las técnicas de fusión de imágenes, también asistimos a una explosión en el mercado de aviones no tripulados con una gran variedad de aplicaciones potenciales, por ejemplo: vigilancia, entrega de paquetes, construcción, etc. Por lo tanto, esto conlleva a imaginar una aplicación en la que varios aviones no tripulados colaboran para obtener imágenes panorámicas. Como se verá a lo largo de este proyecto de investigación, es un problema complejo que puede ser moldeado como un problema de optimización multi-agente con objetivos múltiples. Cuestiones tales como

la forma de coordinar los diferentes agentes o cómo reaccionar en caso de que uno de los agentes se comporte defectuosamente son importantes en este contexto. Los agentes autónomos deben ser reubicados en relación con el resultado deseado o en el caso de error en cualquier robots. Esta reubicación se llevará a cabo mediante la optimización de una función de costes relacionados con el objetivo. La función de coste puede incluir aspectos como: i) el coste de conseguir determinados puntos en común entre las imágenes; ii) el coste de conseguir una imagen panorámica con un número específico de píxeles; o iii) el coste de mover los robots. Desde nuestro punto de vista, este problema puede ser interesante en aplicaciones tales como la fotografía aérea o incluso aplicaciones militares, en donde un conjunto de aviones no tripulados puede ser utilizado para proporcionar ela máxima información posible de una imagen con un cierto objetivo.

Principalmente, el problema se ha estudiado suponiendo la situación más simple, donde se permite el movimiento de agentes autónomos sólo en un plano paralelo a la imagen panorámica deseada, suponiendo que no existen diferencias en los ángulos de las imágenes capturadas entre las distintas cámaras. Los estudios futuros pueden estar destinados al acercamiento de la situación estudiada al mundo real, por ejemplo, teniendo en cuenta las tres dimensiones espaciales de los movimientos de los agentes o la diferencia de ángulos de giro mencionados antes. Sin embargo, no están considerados en el proyecto actual.

En primer lugar, el tema es estudiado desde un punto de vista centralizado, que consiste principalmente en la solución del problema de coordinación entre los agentes considerando que un líder externo tiene toda la información requerida de la situación actual y es quien toma todas las decisiones necesarias basándose en esta misma información. A continuación, el mismo problema se trata desde un punto de vista distribuida, lo que quiere decir que en este caso son agentes quienes toman las decisiones y se comunican entre ellos para recoger alguna información: la información global sobre el estado del sistema no está disponible para cada agente.Por tanto, las decisiones tienen que ser tomadas por cada agente con información parcial del sistema global.

En cuanto al desarrollo de este proyecto, se utilizará Matlab como herramienta de trabajo. Las simulaciones también se realizarán en este programa.

Otra característica importante de este proyecto es que todos los desarrollos que se lleven a cabo pueden ser implementados en un laboratorio real ubicado en Tokio. Para ello, los investigadores involucrados del Instituto de Tecnología de Tokio ofrecen sus laboratorios, los cuales se pueden ver en este artículo [14].

Los temas que se tratan a continuación en este proyecto se dividen de

la siguiente manera: El Capítulo 2 presenta algunas técnicas encontradas en la literatura utilizados para la fusión imágenes y el algoritmo desarrollado para la creación de imágenes panorámicas una vez que las imágenes individuales han sido capturados. El capítulo 3 se centra en métodos basados en algoritmos centralizados para resolver el problema de la coordinación de los agentes móviles, y presenta algunos algoritmos desarrollados para este fin y las simulaciones correspondientes. El capítulo 4 presenta métodos basados en algoritmos distribuidos, e incluye una visión general de cómo poner en práctica estos métodos y ejemplos. Por último, el capítulo 5 presenta las observaciones finales y sugerencias para posibles líneas de investigación futuras.

Capítulo 2

Fusión de imágenes

La fusión de imágenes es uno de los principales problemas que se tratan en este proyecto. *Image stitching* es el nombre en inglés usado para describir el proceso en el que se obtiene una imagen panorámica a partir de un conjunto de imágenes. El procedimiento seguido es una extensión de la búsqueda de puntos característicos similares de las imágenes, pero en lugar de trabajar con un solo par de imágenes, se ponen en relación sucesivamente varios pares de imágenes mediante estos puntos similares previamente encontrados para formar una panorámica entre todas.

Desde principios de siglo, los investigadores han hecho muchos esfuerzos para mejorar los métodos existentes o incluso para desarrollar nuevas formas de obtener una imagen panorámica. Todos estos métodos siguen la misma estructura general, y las diferencias entre ellos están principalmente relacionadas con la operación específica utilizada para conseguir el mismo objetivo. Por ejemplo, en [25], V. Rankov propuso un algoritmo de fusión de imágenes para imágenes microscópicas teniendo como base la necesidad clínica de la adquisición de una imagen de grandes regiones, pero conservando resolución microscópica. Para este propósito, el método se centra en la superación de las discrepancias de intensidad y desalineaciones geométricas entre las imágenes fusionadas. Otro ejemplo se puede encontrar en [23], donde se estudia la posibilidad de la unión de imágenes con la presencia de objetos en movimiento. La técnica utiliza la selección heurística de intensidad y el dominio de gradientes para seleccionar qué píxeles son mejores de cada imagen.

En general, la fusión de imágenes comprende dos etapas: emparejado de imágenes y mezclado de imágenes. El emparejado de imágenes consiste en unas operaciones realizadas para obtener la conexión entre las imágenes. Hay dos maneras diferentes para calcular esta conexión: el método directo [30, 18], y el método de detección de características [7, 22].

El motivo del nombre de 'métodos directos' se debe a la reducción a la

FUSIÓN DE IMÁGENES	
Step 1: Emparejamiento de imágenes	
Opción A: Método directo	Opción B: Método de detección de características
	1. Extracción de características
	2. Correspondencia entre caracteristicas
	E.g.: Características SIFT y SURF
Step 2: Mezclado de imágenes	

Cuadro 2.1: Proceso para fusión de imágenes

directa minimización de la medida de los errores de registro en las imágenes, sin transformaciones algebraicas o geométricas especiales, y debido a que por lo general se basan en la minimización directa de errores de intensidad [30]. Usando métodos directos, es posible conseguir un registro muy preciso, ya que utilizan todos los datos de la imagen disponibles [5], aunque tiene la desventaja de que requiere siempre de una imagen de alta calidad [19]. En la mayoría de los casos, este tipo de imágenes es difícil de conseguir, y es por eso por lo que a menudo se utiliza el método de detección de características.

Los métodos de detección de características tienen la ventaja ya mencionada, pero también poseen el problema de que son necesarias propiedades de invarianza para conseguir un emparejamiento de sucesivas imágenes panorámicas de forma fiable [5]. En general, la detección de características implica dos procedimientos: extracción de características y correspondencia entre estas características. Muchas de las técnicas de segmentación de imágenes se utilizan para la extracción de características, tales como el operador de Canny [27] o el método de clasificación [31]. La correspondencia entre características es el problema más difícil de resolver en la actualidad, y su rendimiento depende de las propiedades de las características detectadas. Como se explica en [9], una de las tareas más importantes en el registro automatizado de imágenes es la obtención de un algoritmo robusto para establecer correspondencias de puntos de control es, y algunos de los algoritmos usados para el emparejamiento de características son mencionados también en dicho artículo.

En la actualidad se usa una variedad de algoritmos para la detección de características, pero dos de ellos, basados en características SIFT y SURF, son los más comúnmente utilizado para la detección de características debido a su robustez. En primer lugar, las características SIFT fueron desarrollados en 1999 para el reconocimiento de objetos [20], y no fue hasta 2004 cuando se presentaron para fusiones de imágenes [21]. Las características SURF

aparecieron dos años más tarde como una mejora de las SIFT aplicadas al empareamiento de imágenes [2].

La segunda parte de la fusión de imágenes es el mezclado. El propósito de estos métodos es el de reducir la diferencia de las intensidades en la conexión de las imágenes haciendo principalmente que los bordes sean invisibles [25]. Idealmente no sería necesario este paso, pero en la práctica los bordes de las imágenes superpuestas suelen ser distinguidos fácilmente. Por esta razón, la elección de una buena estrategia de mezcla mejora la calidad de la imagen panorámica final. Como se muestra en la figura 2.1, donde usando Matlab se obtiene una imagen panorámica a partir de algunas imágenes individuales, los bordes de las imágenes en la imagen panorámica son claramente distinguibles. Esta es la razón por la que la selección de un buen algoritmo para el mezclado final de imágenes es esencial si se quiere evitar el resultado representada en la figura.

Hay muchos algoritmos utilizados para mezclado de imágenes, por ejemplo: 'watersheds blending' [17], 'multi-band blending' [5, 4], 'multi-resoluion spline blending' [23] o mezclado de gradientes de domino [25]. R. Szeliski presenta un estudio de algunas de estas y otras técnicas en [29].

Para resumir, se presenta la Tabla 2.1 donde se han esquematizado los principales pasos y procedimientos de la fusión de imágenes.

(a) Original images

(b) Panorama image

Figura 2.1: Ejemplo de imagen panorámica usando una técnica de mezclado básica.

Capítulo 3

Métodos basados en algoritmos centralizados

En este capítulo nos ocupamos del problema de la coordinación de múltiples agentes. El problema de coordinación multi-agente requiere de varios agentes para integrar los movimientos, con el objetivo de maximizar su rendimiento general [16]. El concepto de *agente* hace referencia a cualquier dispositivo móvil cuyo movimiento sea controlado. Algunos ejemplos se pueden encontrar en [26]: robots móviles, vehículos aéreos no tripulados (UAVs), vehículos submarinos autónomos (AUVs), satélites, aeronaves, etc. Por tanto, la cuestión principal que tiene que ser resuelta es: ¿Cómo pueden ser iteligentemente coordinados los grupos de agentes con este fin?

Muchos investigadores han trabajado sobre este tema. Por ejemplo, T. Siméon presentó en 2002 un enfoque basado en la geometría de coordinación de movimientos múltiples en robot móviles [28]. En ese artículo, el autor trató de resolver el problema de coordinar el movimiento de varios robots que se mueven a lo largo de caminos fijos independientes para evitar colisiones mutuas. Por el contrario, varios investigadores estudiaron lo que los robots autónomos móviles pueden hacer considerando la coordinación distribuida entre ellos [13]. Otro ejemplo diferente se puede encontrar en [24], donde se estudia la colaboración entre un grupo de UAVs (Microdrones), basadas principalmente en imágenes aéreas cooperativas para aplicaciones de resolución de desastres.

La coordinación entre robot es necesaria para robots en agua, tierra y aire, y se puede llevar a cabo en interiores o al aire libre. En muchas situaciones, es necesaria la construicción de mapas del entorno, al mismo tiempo que el uso de estimadores de posición como 'Extended Kalman Filter' (EKF) [8]. Por el contrario, las aplicaciones al aire libre suelen utilizar la técnica conocida como SLAM, ('Simultaneous Localization and Mapping'). SLAM se

encarga del problema de la colocación de un robot móvil en un lugar desconocido en un entorno desconocido y de manera progresiva se construye un mapa coherente del estado presente mientras se determina simultáneamente su ubicación dentro de este mapa. Más información sobre el método de SLAM se puede encontrar en [1, 11].

En este proyecto, se trabaja con vehículos aéreos no tripulados. El principal beneficio de utilizar varios drones es la posibilidad de conseguir una imagen de una gran área con alta resolución. Hay muchas aplicaciones en las que las vistas aéreas puede ser realmente útiles, por ejemplo, las situaciones de desastre debido a la visión general del entorno logrado con este sistema coordinado [24].

Hay otras ventajas en el uso de esta configuración múltiple de drones [24]: i) En primer lugar, la posibilidad de cubrir un área mucho más grande; ii) en caso necesario se puede recoger más información; y iii) la posibilidad de fallo es menor debido a que el sistema está formado por varios elementos, es decir, el sistema es más redundante y robusto contra los fallos.

En este proyecto, se supone que las técnicas de control de vehículos aéreos no tripulados son conocidas y están disponibles para su uso. Por lo tanto, no se presta más atención a este tema. También se considera que los aviones están equipados con los sensores y actuadores necesarios para alcanzar los objetivos deseados [24]. Por lo tanto, el resto del proyecto se centra principal y exclusivamente a la forma de coordinar los agentes del sistema considerado en nuestro trabajo. Para la coordinación de múltiples agentes hay dos posibles enfoques diferentes: el centralizado y el distribuido. Este capítulo se ocupa de la coordinación centralizada y el siguiente de la coordinación distribuida.

Como su nombre indica, la *Coordinación centralizada* consta de un operador central que tiene información completa de todo el sistema, y es el mismo operador quien toma las decisiones sobre cada uno de los movimientos de los aviones no tripulados. Este operador central puede ser uno de los elementos del sistema, y también se le llama líder. Es el responsable de enviar a cada elemento la información correspondiente sobre su movimiento [6].

Por lo tanto uno de los principales inconvenientes de la coordinación centralizada es la falta de robustez cuando hay fallos en la comunicación y/o funcionamientos incorrectos del líder [12]. Es posible resolver este problema mediante el uso de un tipo diferente de centralización, pero manteniendo su esencia. A. Farinelli explica una posible opción en [12], en la que débilmente expone los sistemas centralizados, caracterizados por el hecho de que el líder no se elige a priori, pero se selecciona de forma dinámica durante la misión dependiendo de la situación actual del sistema y el entorno.

Capítulo 4

Métodos basados en algoritmos distribuidos

En el mundo virtual, puede parecer que la coordinación centralizada es la forma más fácil de lograr nuestro objetivo, pero es poco probable tener realmente un sistema con esas características en el mundo real. Por lo tanto, la coordinación distribuida es la técnica de uso común para aplicaciones reales, y su principal diferencia con la coordinación centralizada es que en lugar de tener un líder que recibe toda la información de los agentes y toma todas las decisiones, cada agente recibe alguna información de su área más cercana y sólo toma decisiones referentes a sí mismo.

La mayoría de los artículos que se encuentran en la literatura están relacionados con este tipo de coordinación y se centran específicamente en la manera de cómo resolver los problemas que conlleva, que no son pocos. Por ejemplo, en el artículo [26], Ren, Beard y Atkins discuten los problemas que aparecen cuando el algoritmo diseñado se aplica a los sistemas reales de los vehículos aéreos no tripulados, por ejemplo retrasos en la comunicación, la relativa incertidumbre en la información o estado de equilibrio de la solución. En [13], el sistema estudiado está formado por un conjunto de robots móviles autónomos que sólo han programado sus comportamientos como un conjunto de leyes que guía el robot para reaccionar a los estímulos del entorno de manera que no hay un objetivo explícito programado. La cooperación y los objetivos simplemente emergen mientras la el cálculo se lleva a cabo, y además los robots no tienen memoria del pasado. Por lo tanto toma relevancia la visibilidad instantánea de cada agente, y los autores de este artículo estudian el problema de tener una visibilidad limitada o ilimitada, comparando ambos resultados.

La coordinación distribuida posee muchos campos de aplicación, no sólo el que se explica aquí con robots aéreos. Un ejemplo de coordinación de robot

móviles se puede encontrar en [28], donde se requiere esta coordinación con el fin de hacer posible el movimiento de muchos robots en un mismo espacio. Cada robot tiene definidos y fijados unos caminos independientes, y el problema es la coordinación de estos movimientos para evitar colisiones, es decir, la coordinación del camino. Sin embargo, el artículo [24] está dedicado a las aplicaciones de robots aéreos: la vigilancia del medio ambiente, la vigilancia de seguridad, la resolución de desastres o de planificación de misiones, entre otros. Uno de los campos citados en ese documento es la captación de imágenes aéreas de cooperación, que es el mismo campo en el que se centra este proyecto.

Además, algunos autores han resuelto este problema de la coordinación mediante la aplicación de algunas funciones matemáticas con el fin de obtener una figura geométrica determinada formada por las posiciones de los robots, como una simple línea [15] o una forma circular [3]. Más detalles de estos procesos se pueden encontrar en los artículos indicados.

Capítulo 5

Conclusión

En este proyecto, hemos presentado algunos algoritmos para lograr imágenes panorámicas de algunas imágenes capturadas por agentes individuales coordinados entre sí. Se ha considerado un sistema compuesto por muchos agentes móviles aéreos, cada uno de ellos equipado con una cámara, cuyos movimientos están restringidos al plano XY. El principal problema estudiado en este proyecto ha sido la ubicación de los agentes en el área disponible con el fin de obtener la imagen panorámica con unas características específicas.

En primer lugar, algunos algoritmos se han desarrollado utilizando una coordinación centralizada entre los agentes. Principalmente se han comparado dos tipos de resultados, los óptimos y los heurísticos; concluyendo que el segundo sería el más conveniente para nuestro proyecto debido al corto tiempo computacional requerido. Se han realizado algunas modificaciones del algoritmo heurístico con el objetivo de continuar disminuyendo el tiempo total de cómputo. Sin embargo, se pone de manifiesto que algunas características, como la centralización en el punto central, se empeoran.

En segundo lugar, se ha resuelto el mismo problema mediante coordinación distribuida. Los resultados óptimos y heurísticos se han comparado de la misma forma que en el algoritmo centralizado, a pesar de que el algoritmo óptimo desarrollado no era exactamente el óptimo. Se han obtenido diferentes resultados teniendo en cuenta las características específicas requeridas; concluyendo que para propósitos generales, los el algoritmo haurístico ofrece mejores resultados que el óptimo simepre que no sea indispnesable una alta precisión.

Líneas futuras de investigación

Nuevos algoritmos para la coordinación distribuida. En la vida real, los robots son generalmente coordinados usando el enfoque distribuido.

Por lo tanto, pensando en posibles investigaciones futuras sobre este tema, sería interesante esta ampliación para aproximar las características del sistema al mundo real.

Sistema en 3D. Otro punto que podría ser investigado es el que consiste en transformar todo el trabajo realizado para sistemas de 2D a 3D. Los agentes deberían moverse en el espacio XYZ, incluyendo la nueva coordenada Z para cada agente, y por lo tanto, una nueva variable para ser considerada. Por otra parte, también podría ser estudiado el cambio en la orientación de las cámaras lo que permitiría la rotación de los agentes.

Control de UAVs. Antes de acercar los resultados al mundo real, sería necesario investigar más sobre la forma de controlar los UAVs. En el proyecto actual, este punto se ha considerado conocido por el diseñador, por lo que no se le ha prestado atención.

Implementación real. El objetivo final de este proyecto sería la aplicación de todos estos algoritmos desarrollados en un laboratorio que cuente con las instalaciones necesarias. Esta aplicación real sería el punto culminante de este proyecto.

Bibliografía

[1] Tim Bailey. *Mobile robot localisation and mapping in extensive outdoor environments*. PhD thesis, Citeseer, 2002.

[2] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[3] Lara Brinón-Arranz and Luca Schenato. Consensus-based source-seeking with a circular formation of agents. *European Control Conferenc e*, pages 2831–2836, 2013.

[4] Matthew Brown and David G Lowe. Recognising panoramas. In *ICCV*, volume 3, page 1218, 2003.

[5] Matthew Brown and David G Lowe. Automatic panoramic image stitching using invariant features. *International journal of computer vision*, 74(1):59–73, 2007.

[6] Wolfram Burgard, Mark Moors, Cyrill Stachniss, and Frank E Schneider. Coordinated multi-robot exploration. *IEEE Transactions on robotics*, 21(3):376–386, 2005.

[7] David Capel and Andrew Zisserman. Automated mosaicing with super-resolution zoom. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pages 885–891. IEEE, 1998.

[8] José A Castellanos, Juan D Tardós, and Günther Schmidt. Building a global map of the environment of a mobile robot: The importance of correlations. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 2, pages 1053–1059. IEEE, 1997.

[9] Xiaolong Dai and Siamak Khorram. A feature-based image registration algorithm using improved chain-code representation combined with invariant moments. *IEEE Transactions on Geoscience and Remote Sensing*, 37(5):2351–2362, 1999.

[10] Raffaello D'Andrea. Guest editorial can drones deliver? *IEEE Transactions on Automation Science and Engineering*, 11(3):647–648, 2014.

[11] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[12] Alessandro Farinelli, Luca Iocchi, and Daniele Nardi. Multirobot systems: a classification focused on coordination. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(5):2015–2028, 2004.

[13] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Peter Widmayer. Distributed coordination of a set of autonomous mobile robots. In *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*, pages 480–485. IEEE, 2000.

[14] Marc Forstenhaeusler, Riku Funada, Takeshi Hatanaka, and Masayuki Fujita. Experimental study of gradient-based visual coverage control on so (3) toward moving object/human monitoring. In *2015 American Control Conference (ACC)*, pages 2125–2130. IEEE, 2015.

[15] Samratul Fuady and Hideaki Ishii. Alignment forming in source seeking for multi-agent systems.

[16] Brian P Gerkey and Maja J Mataric. Sold!: Auction methods for multirobot coordination. *IEEE transactions on robotics and automation*, 18(5):758–768, 2002.

[17] Nuno Gracias, Mohammad Mahoor, Shahriar Negahdaripour, and Arthur Gleason. Fast image blending using watersheds and graph cuts. *Image and Vision Computing*, 27(5):597–607, 2009.

[18] Michal Irani and P Anandan. About direct methods. In *International Workshop on Vision Algorithms*, pages 267–277. Springer, 1999.

[19] Luo Juan and Gwun Oubong. Surf applied in panorama image stitching. In *Image Processing Theory Tools and Applications (IPTA), 2010 2nd International Conference on*, pages 495–499. IEEE, 2010.

[20] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[21] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[22] Philip F McLauchlan and Allan Jaenicke. Image mosaicing using sequential bundle adjustment. *Image and Vision computing*, 20(9):751–759, 2002.

[23] Alec Mills and Gregory Dudek. Image stitching with dynamic elements. *Image and Vision Computing*, 27(10):1593–1602, 2009.

[24] Markus Quaritsch, Emil Stojanovski, Christian Bettstetter, Gerhard Friedrich, Hermann Hellwagner, Bernhard Rinner, Michael Hofbaur, and Mubarak Shah. Collaborative microdrones: applications and research challenges. In *Proceedings of the 2nd International Conference on Autonomic Computing and Communication Systems*, page 38. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[25] Vladan Rankov, Rosalind J Locke, Richard J Edens, Paul R Barber, and Borivoj Vojnovic. An algorithm for image stitching and blending. In *Biomedical Optics 2005*, pages 190–199. International Society for Optics and Photonics, 2005.

[26] Wei Ren, Randal W Beard, and Ella M Atkins. A survey of consensus problems in multi-agent coordination. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 1859–1864. IEEE, 2005.

[27] Eric JM Rignot, Ronald Kowk, John C Curlander, and Shirley S Pang. Automated multisensor registration: Requirements and techniques. *Photogrammetric Engineering & Remote Sensing*, 57(8), 1991.

[28] Thierry Siméon, Stéphane Leroy, and J-P Lauumond. Path coordination for multiple mobile robots: A resolution-complete algorithm. *IEEE Transactions on Robotics and Automation*, 18(1):42–49, 2002.

[29] Richard Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 2(1):1–104, 2006.

[30] Richard Szeliski and Sing Bing Kang. Direct methods for visual scene reconstruction. In *Representation of Visual Scenes, 1995.(In Conjuction*

with ICCV'95), Proceedings IEEE Workshop on, pages 26–33. IEEE, 1995.

[31] AD Ventura, Anna Rampini, and Raimondo Schettini. Image registration by recognition of corresponding structures. *IEEE Transactions on Geoscience and Remote Sensing*, 28(3):305–314, 1990.