# SOFTWARE SOLUTION FOR HIGH RESOLUTION IMAGE VISUALIZATION

**Andrés Coronado Giménez**

**Zagreb professor: Davor Petrinovic**

**Valencia professorr: Jose Manuel Mossi**

Curso 2017-18

Zagreb, 1 de julio de 2018

This thesis is the last project for obtaining the title that corresponds with the postgrad studies of telecommunication engineer. It has been developed during the exchange program Erasmus+, in the Faulty of Electronics of Zagreb (FER).

## Abstract

As part of the graduate master thesis, it is necessary to devise, realize and test a program solution for dynamic visualization of high resolution still images in the form of an animated video. As a rule, the standardized resolution of a typical display system is significantly lower than the resolution of digital still cameras (e.g. SLR) because commercial screens have a resolution of 2 to 8 million pixels, while cameras have a resolution of up to 50 million pixels. Because of this limitation, the image can only be displayed with significant resolution reduction or by cropping a small detail of the original digital still image. There is a need to develop a software solution that simultaneously enables both: to produce a video from a single image in which the individual frames are obtained by scaling, cropping, panning, and rotation of a high resolution original photo. In addition, the program must enable merging or transitions from one photograph to another for applications when the same object is photographed by cameras with different optical parameters and different angular widths (stored) and resolutions. The system needs to be developed, implemented and evaluated using the Matlab program environment.

## Resumen

Como parte de la tesis de final de máster, es necesario diseñar, realizar y probar un programa para la visualización dinámica de imágenes fijas de alta resolución en forma de un video animado. Como regla general, la resolución estandarizada de un sistema de visualización típico es significativamente menor que la resolución de las cámaras digitales (por ejemplo, SLR) ya que las pantallas comerciales tienen una resolución de 2 a 8 millones de píxeles, mientras que las cámaras tienen una resolución de hasta 50 millones de píxeles. Debido a esta limitación, la imagen solo se puede visualizar con una reducción de resolución significativa o recortando un pequeño detalle de la imagen fija digital original. Es necesario desarrollar una solución de software que simultáneamente permita ambas cosas: producir un video a partir de una sola imagen en la que los marcos individuales se obtienen escalando, recortando, panoramizando y rotando una foto original de alta resolución. Además, el programa debe permitir la fusión o transiciones de una fotografía a otra para aplicaciones cuando el mismo objeto es fotografiado por cámaras con diferentes parámetros ópticos y diferentes anchos angulares (archivados) y resoluciones. El sistema debe desarrollarse, implementarse y evaluarse utilizando el entorno del programa Matlab.

## Resum

Com a part de la tesi de final de master, és necessari dissenyar, realitzar i provar un programa per a la visualització dinàmica d'imatges fixes d'alta resolució en forma d'un video animat. Com a regla general, la resolució estandarditzada d'un sistema de visualització típic és significativament menor que la resolució de les càmeres digitals (per exemple, SLR) ja que les pantalles comercials tenen una resolució de 2 a 8 milions de píxels, mentre que les càmeres tenen una resolució de fins a 50 milions de píxels . A causa d'aquesta limitació, la imatge solament es pot visualitzar amb una reducció de resolució significativa o retallant un xicotet detall de la imatge fixa digital original. És necessari desenvolupar una solució de programari que simultàniament permeta ambdues coses: produir un video a partir d'una sola imatge en la qual els marcs individuals s'obtenen escalant, retallant, panoramizando i rotando una foto original d'alta resolució. A més, el programa ha de permetre la fusió o transicions d'una fotografia a una altra per a aplicacions quan el mateix objecte és fotografiat per càmeres amb

diferents paràmetres òptics i diferents amples angulars (arxivats) i resolucions. El sistema ha de desenvolupar-se, implementar-se i avaluar-se utilitzant l'entorn del programa Matlab.

# Index

# CHAPTER 1. Introduction

The human visual system is one of the most powerful image processing mechanisms that exist. It is without a doubt the most used sense by the human being, so the images have taken a fundamental role in our society throughout the last years. The great capacity of the images to transmit information, as well as the easy access to high quality cameras and terminals for their visualization have helped this great growth. This growth was mainly due to an advance of great importance, the digitalization of the images, which allows both the storage and the ability to process them to obtain information or create different effects. As a result of this feature, many applications have been developed that use images with very diverse purposes such as medical applications, tracking, object detection ...

One of the fields where this processing is especially necessary is in the spatial photographs, that is, photographs taken to the space in order to obtain some kind of information from the light that comes from the spatial elements. The processing is necessary because the light that comes to us comes from an infinity of stars, which are at very distant distances, which causes the information to be confused when the light of different elements is mixed or modified. The blur that light undergoes as it passes through the atmosphere. This project seeks to achieve an improvement in photographs of high quality space, more specifically by eliminating unwanted effects introduced by different types of lenses, which due to its spherical geometry each lens deforms the image in some way.

# CHAPTER 2. THEORY

## 2.1 Image processing

In general, the processing of an image consists of altering the visual information to obtain better results or to isolate some particular characteristics of the images. The impact of this discipline has been huge and affects practically all areas such as telecommunications, industrial processes, entertainment and much others. A digital image can be defined as the numerical representation of an object, for which an image is discretized in both spatial coordinates and brightness. The numerical values of these values are stored inside a matrix, which is divided into small pieces known as pixels. Each of these pixels has a luminosity value and another 3 of colours that represent the 3 primary colours. With this, we can define the digital processing of images as the set of operations carried out on these pixels to achieve the desired objective.

## 2.2 Processing types

It can be said that the first attempts to manipulate and transmit images arose in 1920, when a system of transmitting photographs through a transatlantic cable was developed using telegraphic codes. This system allowed the codification of the image in 5 levels of grey, later it was improved up to 15 levels in 1929. With this system, the transmission time of quality images was reduced for publication from 2-week newspapers (shipping by ship) to 15 minutes.

The processing and analysis of images has been developed in response to three of the biggest problems concerning images: the digitization and coding of images that facilitate the transmission, representation and storage of images, the improvement and restoration of an image to interpret more easily its content and the description and segmentation of images for applications of robotic vision or artificial vision.

Today we have many methods of image processing, all to improve certain characteristics and to evaluate some statistical aspects, which can be divided into 3 large groups:

Algorithms in the spatial domain. It refers to methods that process an image pixel by pixel, or also considering a set of neighbouring pixels.

Algorithms in the frequency domain. Frequently, these methods are applied to the resulting coefficients of the Fourier Transform of an image.

Feature extraction algorithms. Unlike the two previous groups, feature extraction algorithms are focused on the analysis of images for the extraction of attributes and regions of interest, separation of objects from the background, detection of edges or shapes, among others.

## 2.3 Spatial domain methods

They include all methods that are based on the processing of a pixel (called the current pixel) from a relationship that can include neighbouring pixels

### *2.3.1 Transform types*

Depending on the relationship of the output pixel with the neighbours of the current pixel, the transformations of an input image in a processed image can be classified as follows:

• Point transformations. They are those in which the pixel resulting from the operation depends only on the value of the input pixel. Typical point operations include the manipulation of pixels one by one, for example, binarization, segmentation, colour correction, tone, saturation, gamma, etc.

• Local transformations. In this case, to obtain the output pixel, the contributions of the neighbouring pixels in the operation are used. Many operations are local, for example, smoothing, media, morphological operations, edge enhancement. They are classified into linear filters, such as the average, and non-linear filters, such as the median. For these spatial operations, an image is taken as input, and run through each of its pixels using a window of neighbourhoods of size NxN. In this way they process an image through all its elements, and apply a transformation on them

• Geometric transformations. They are made considering the positions of the pixels in the image, and geometric operations are applied to them. Typical examples are rotation, translation, scale changes, rectification, and radiometric transformations of the pixels. This type of transformation is the one used in this project, which is why it is developed widely in section 2.6.

## 2.4 Frequency domain methods

These algorithms are based on filters that process an image working on the frequency domain trough Fourier Transform of the image. Because the image is considered a finite and discrete two-dimensional function, there is its Discrete Fourier Transform (DFT). To obtain the DFT, the original image is modified following the Convolution Theorem.

• Transformations in the Fourier domain: A filter transfer function H (u, v) is one that acts on the Fourier transform of an image F (u, v), and allows certain frequencies to be suppressed while leaving others without any change. Low frequencies are responsible for most grey levels of an image over soft areas. While the high frequencies have to do with the details of the image, such as edges and noise.

• Transformations based on Histogram: The histogram of an image is a graphic representation of the frequency with every grey level appear on it (or intensity levels in each colour plane, in case of a colour image). It is a fundamental tool for the analysis of digital images, since it allows to "condense" a lot of information about the image (probabilities of each level of grey) although the spatial location is lost. Its dynamic range is the set of grey levels present.

## 2.5 Characteristics extraction methods

Contrary to the methods seen so far, feature extraction is a method that takes an image as an input and extracts interesting attributes from it, such as coordinates of objects that meet certain characteristics, detection of curves and shapes, component labelling, among others. The extraction of characteristics enters fully into the field of image analysis, constituting the first stage in the intelligence of an artificial vision system.

## 2.6 Geometric transforms

Geometric operations modify the spatial relationships between the pixels of an image and can be of three basic types: translation, scaling and rotation. Unlike other operations, geometric transformations change the projection of the image on the plane that contains it, which implies a change in the distribution of the pixels with respect to a coordinate system. The resulting image differs in size and perhaps in shape from the original. This type of transformation is very important in the analysis of images, especially for the recognition of patterns. Depending on the position of the objects in space, when captured by a camera they can (and usually) modify their shape, transforming for example the circles into ellipses, the squares into rhomboid forms. Therefore, in this type of situation, before comparing patterns and detecting forms a previous stage of transformation of the image is necessary.

In a digital image there are no intensity values between the values of the x and y coordinates, which coincide with the intersections of the horizontal and vertical values. This is because the digital image is of a discrete nature. When a transformation occurs in the grid, either by a displacement, a turn or an approach, the new pixels no longer have to coincide with intersections and will usually be located on intermediate points, as shown in figure 1, where the continuous grid shows the conventional arrangement of the pixels of the original image, while the discontinuous grid reflects how the original grid would look after turning it at a certain angle. Since the pixels must be projected over the final image, with a structure similar to that of the original image, it is necessary to convert these coordinates, and also obtain the values of the final pixels as a function of the transformed ones.

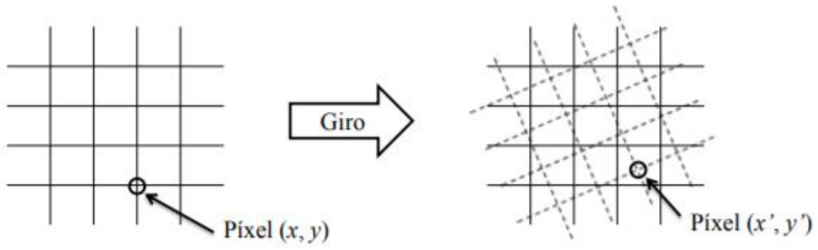


*Figure 1: Mismatch due to geometric transform*

For this, on the one hand it is necessary to determine the coordinates of each pixel (x, y) in the transformed grid (in broken lines), as shown in Figure 1. In general, the pixels (x ', y') obtained after the transformation will not be integer values and, therefore, will not coincide with pixels of the destination grid. On the other hand, it is necessary to calculate the values of the pixels (x ', y') in the target grid from the known values of pixels (x, y) between them. The first step depends on the transformation to be performed, while the second one corresponds to an interpolation operation. Once a geometric transformation has been applied, it is necessary to obtain the intensity values associated with the transformation carried out, so that the original image appears geometrically transformed, but with the intensity values obtained from the corresponding ones in the original image. It is in this step where the modification of the intensity values of the original image is generated. In this section we will first study the concept of interpolation and the way to calculate the intensity values associated with a geometric transformation, to then see different types of transformation functions.

### *2.6.1 Interpolation*

The interpolation can be seen as the calculation of the intensity value of a pixel, in any position, as a function of the surrounding pixels (occupying the entire positions of the destination grid). This calculation is necessary because when geometric transformations are applied to the images, changes in the position of the pixels are presented with respect to their initial position in the image plane and therefore numerical interpolation methods are required for the intensity values that will be assigned to the new positions in the image plane. There are three known methods that are the nearest neighbour, bilinear interpolation and bicubic interpolation.

• Interpolation by nearest neighbour

It consists in supposing that the pixel to be interpolated takes the same value as that of the closest pixel among the four that surround it. To decide which is the nearest pixel, the Euclidean distance is usually used. In Figure 2, it would take the value of the pixel p (i, j). Another option is to assign the average intensity associated with the two closest pixels, one in the horizontal direction x and the other in the vertical direction y. In this case, in Figure 2, it would take the mean value between p (i,j) and p (i+1, j).

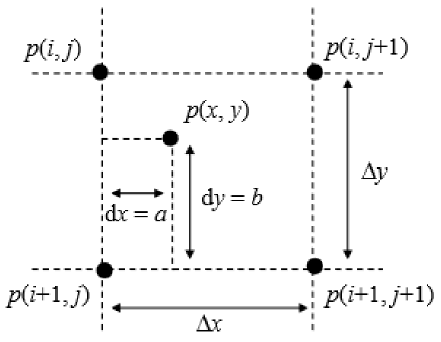


*Figure 2: Interopolation scheme*

• Bilinear interpolation

Another way to perform interpolation with better results than those obtained with interpolation by nearest neighbour, although at the cost of a greater computational load, is the bilinear interpolation, which assigns to each pixel object of the transformation a weighted average value of the intensities of the four pixels that surround it. The intensity value of the interpolated pixel, as a function of the four intensity values of the pixels in its environment, is calculated as follows

$$p(x,y) = a_1 p(i,j) + a_2 p(i,j+1) + a_3 p(i+1,j) + a_4 p(i+1,j+1)$$

where the weighting factors (a1, a2, a3, a4) are given by the distance between each pixel and those of its environment.

• Bicubic interpolation

A third way to perform interpolation with better results than bilinear interpolation is bicubic interpolation. In this case, 16 neighbouring points of the pixel that is being interpolated intervene. Since they are located at distances other than the pixel of unknown value, greater weight is given in the calculation to the nearest ones. The bicubic interpolation is the one that offers better results but in turn is the one with the highest computational load.

### *2.6.2 Geometric transforms types*

There are different types of geometric transformations for the image, which can be divided into 2 large groups: Linear or affine and non-linear. The affine type transformations include the basic transformations in which the coordinates of the pixels of the new image are expressed linearly in terms of the original pixels. This involves the conservation of straight lines, parallel lines and length ratios along a line. On the other hand, non-linear transformations use more complex equations, so the transformed image can change any of its characteristics.

There are 2 ways to perform or define the desired geometric transformation:

• Direct form: in case the desired transformation is known in advance, it can be defined directly by means of a transformation matrix, as will be seen below, to apply it to the original image. This method is usually used for the affine transformations and, therefore, for the realization of the video, which will consist of zooming, displacement and rotation on the image.

• Through control points: this method is used when the coordinates of certain pixels of the transformed image are known accurately in advance. In this way certain control points are established, which consist of pairs of points that relate the position of a point of the original image with its position in the transformed image. Thus, by means of several pairs of points, the transformation that best adjusts to these changes is calculated by means of a function. This transformation, which may be affine or non-linear, will be the one used to correct the effect introduced by the lenses.

#### *2.6.2.1 Affine transformations*

An affine transformation is one in which the coordinates (x ', y') of the image point are expressed linearly in terms of those of the original point (x, y). That is, the transformation is given by the equations:

$$x' = ax + by + m$$
$$y' = cx + dy + n$$

When m = n = 0 the above equations become the prototype of a linear transformation by multiplication by a matrix to the left, that is,

$$\begin{cases} x' = ax + by \\ y' = cx + dy \end{cases} \longrightarrow \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Therefore, it is not difficult to show that the related transformations transform a grid (or mesh) into another grid, characterizing them as the most natural to the images. In the study of the meaning of the coefficients of the matrix of the transformation, it can be proven that

$$T\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} SC_x & S_{Horiz.} \\ S_{Vert.} & SC_y \end{pmatrix}$$

where, a = $SC_x$ corresponds to scaling in x, d = $SC_y$ scaling in y, b = $S_{horiz}$ pressure or tilt in horizontal direction and c = $S_{vert}$ pressure or tilt in vertical direction.

There are also other actions of related transformations such as rotating a geometric region an angle θ around a fixed point and defined as:

$$T\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}$$

On the other hand, we know that in order to move the image on the x and y axes, it is only necessary to add or subtract (depending on the direction) the number of pixels that you want to displace the image. To add this function to the translation matrix, it is only necessary to add these coefficients in a new column and fill the matrix with a row that will be 0 0 1. Thus, different possible transformations can be defined using a single matrix. To combine these transformations, it is only necessary to multiply the matrices with the different types of transformations.

$$\begin{cases} x' = ax + by + m \\ y' = cx + dy + n \end{cases} \longleftrightarrow \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & m \\ c & d & n \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

This is how the transformation matrix ends up, where the scalars a, b, c, d have the same meaning as those of the previous equation, m corresponds to the horizontal displacement while n to the vertical displacement.

#### 2.6.2.2 Nonlinear transformations

The related transformations have the advantage that they are perfectly invertible, in addition to maintaining the parallel lines, length ratios ... However, in certain cases it is necessary to apply a more complex transformation to achieve other effects. This is where the non-linear transformations appear, which we can divide into projective and polynomials. On the one hand the projective do not maintain the parallel lines yet, however, they converge towards the same point. Polynomials on the other hand can modify any of the characteristics of the image.

Non-linear transformations are just an extension of the related transformations in which more elements are introduced, so we have more equations. Thus, the projective transformation adds a single equation, with what remains as follows:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Here we see how even non-linear transformations are defined by linear equations.

The same occurs for polynomials. Any polynomial transform can be expressed as:

$$\begin{aligned} u &= \sum_k \sum_l a_{kl}\, x^k y^l \\ v &= \sum_k \sum_l b_{kl}\, x^k y^l \end{aligned}$$

In this way we can define a related transformation as a polynomial of degree 1. To see this better, we develop the equations for a polynomial of degree 2:

$$\begin{aligned} u &= a_{20}x^2 + a_{02}y^2 + a_{11}xy + a_{10}x + a_{01}y + a_{00} \\ v &= b_{20}x^2 + b_{02}y^2 + b_{11}xy + b_{10}x + b_{01}y + b_{00} \end{aligned}$$

Which can be expressed in matrix form:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a_{20} & a_{02} & a_{11} & a_{10} & a_{01} & a_{00} \\ b_{20} & b_{02} & b_{11} & b_{10} & b_{01} & b_{00} \end{bmatrix} \begin{bmatrix} x^2 \\ y^2 \\ xy \\ x \\ y \\ 1 \end{bmatrix}$$

It can be verified in a simple way as in the case of a20 = a02 = a11 = b20 = b02 = b11 = 0 this transformation becomes an affine type. As we can see in the next image there are different types of distortion introduced by the lenses, which can be corrected with transformations both affine and polynomial type.

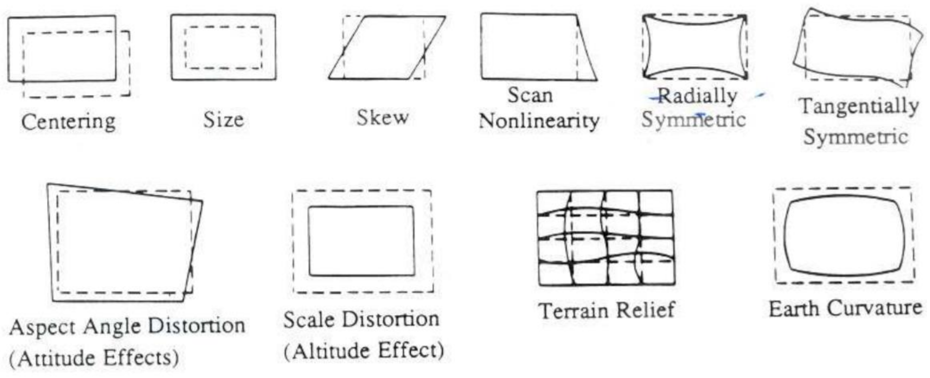


*Figure 3: Different tipes of distortion*

As seen in the previous image, there are a great variety of distortions introduced by the lenses, which require different transformations to be corrected. In the case of this project, an attempt is made to correct the effect known as radial symmetry, introduced due to the spherical shape of the lenses. A polynomial transform of 3rd degree will be used.

# CHAPTER 3. ALGORITHM

As has been mentioned previously, the objective of this project is to create a full HD video from a static image through different geometric transformations. Actually, we are going to make use of 2 images, but making the effect that there is only one. These two images show the same region of the space with different angular widths, therefore, captured with different lenses, which introduce a different radially symmetric distortion on each image. Due to this distortion, it's impossible to get a smooth transition between the 2 images without feel the differences because a deviation of only 2 pixels could be dramatic in an image with a huge number of elements, as our case, where we have thousands of stars in each image. This problem has to be solved making use of Matlab due to its huge capacity for working with matrices (the same element with which digital images are represented) and also big amount of image processing functions.

The problem is, at the time of merging from one image to another, the elements of the image (like stars) that are located near the edges will not be completely aligned due to the radially symmetric deformation, so in the transition there will be a mismatch. If we are capable to avoid this problem in some way, once corrected, it will be possible to make the video passing from one image to another when necessary without the change being perceived.

In this chapter, first it is explained the realization of the linear transformations, which we use for applying some transforms to make the video. Second, it will be explained how to infer a non-linear transformation and how we can take advance of them to solve the problem of the mismatch when the transition is carried out.

### *3.1.1 Affine transforms*

As the first step of the project, an algorithm capable of performing different linear geometric transformations was developed. We have worked with the 3 basic transformations that are: rotation, scaling and shifting, which can be applied at the same time by multiplying their respective transformation matrices. For apply these transforms a script has been done, which works with a matrix as input, in which all the transformations that we wish to apply are defined, also the number of steps for each transformation could be set. So, in the first row is defined the rotation to be applied in degrees, the second row means the zoom factor, the third and fourth for shifting in x and y axes (relative to the image) respectively and, for the last row, the number of steps between the original image and the transformed one. In this way, each column represents a transformation, composed of the 3 basic transformations, which will be carried out from the result of the previous transformation. To make these transformations in Matlab, 3 main steps are needed:

* <u>Creation of the transformation matrix</u>: As we have seen in the section of affine transformations, these are defined by a matrix, which relates the position of the new pixels to their original position. To achieve the 3 types of transformations it has been necessary to create 3 different matrices, which are finally multiplied to apply the 3 transformations at the same time. In addition to the creation of the matrices, it is necessary to make certain adjustments for getting the desired behaviour:

In the case of rotation, the transform is carried out by default around the pixel (0,0), which refers to the upper left corner. The solution for the rotation to be carried out from the centre is to perform a first displacement to bring the centre of the image to point (0,0) of the current coordinate system, with which a rotation is achieved on the centre of the image. Later it is necessary to apply the reverse displacement to visualize the desired part of the image:

```
w = size(I_b, 2); %Image width
h = size(I_b, 1); %Image height

Rdefault = imref2d(size(I_b)); % The default coordinate system

tX = mean(Rdefault.XWorldLimits);
tY = mean(Rdefault.YWorldLimits);
tTranslationToCenterAtOrigin = [1 0 0; 0 1 0; -tX -tY,1];
tTranslationBackToOriginalCenter = [1 0 0; 0 1 0; tX tY,1];
```

In this part of the code it is shown how to apply the rotation matrix from the centre of the image. First, an imref2d object encapsulates the relationship between the intrinsic coordinates anchored to the rows and columns of a 2-D image and the spatial location of the same row and column locations in a world coordinate system. The XWorldLimits field as well as the YWorldLimits correspond to the coordinate system that will be used in the output image, which starts at 0 unless otherwise indicated. Knowing this computing the mean of the vectors x and y, we obtain the centre of the new coordinate system. The input is the size of the original image, so the size and the ratio of the transformed image will be the same as the original. Finally, the matrices responsible for moving the image are constructed so its centre coincides with the point (0,0). For this it is worth moving the image half of its pixels in the left direction and up. A similar effect occurs when applying the zoom, where it's also needed to make a movement to display the desired part.

In the next code we can see how the transformation matrix are constructed and how the shifting is computed.

```
     %%ROTATION MATRIX
formRotation = [cosd(rotVec(i)) -sind(rotVec(i)) 0 ; sind(rotVec(i))
cosd(rotVec(i)) 0 ; 0 0 1]';
TRot =
tTranslationToCenterAtOrigin*formRotation*tTranslationBackToOriginalCenter;

%%ZOOMING MATRIX
%Compute displacement:
x0 = w/2 - zoomVec(i)*center_x;
y0 = h/2 - zoomVec(i)*center_y;
TZoom = [zoomVec(i) 0 0 ; 0 zoomVec(i) 0 ; x0 y0 1];

%%SHIFTING MATRIX
Tshift=[1 0 XshiftVec(i) ; 0 1 YshiftVec(i) ; 0 0 1]';
```

* <u>Obtaining the tform</u>: With this step, the structure that will later be the input of the transformation function is built.

This process is carried out by the function affine2d.m, which receives as input the transformation matrix to return a structure with the necessary elements to carry out the transformation. These are:

T - 3x3 matrix representing forward affine transformation

Dimensionality - Dimensionality of geometric transformation

* Application of the transformation: To perform the transformation of the image there are several functions that use different methods. In this case imwarp.m has been used, which receives as input the image to be transformed, the geometric transformation defined by the tform structure as well as several optional parameters, among which is the coordinate system that we want to use in the new image to return the transformed image, the type of interpolation desired etc.

```
%Combined matrix + tform
TZyR = TZoom*TRot*Tshift*lastT;

tform = affine2d(TZyR); %Needed by imwarp

J = imwarp(I, tform, 'OutputView', Rdefault,'FillValues',255, 'Interp',
'cubic')
```

Below there is one example of these transformations on the image with greater angle of view, such as a rotation of 45º plus a displacement on the x axis of 20%, zoom keeps 1 and there is no displacement in y axis. The new image will be cropped in those parts that are mapped out of the coordinate system used and, in addition, the empty parts of the image will be filled in white white due to the field 'FillValues' = 255. The reason for this cut is that the creation of the video in Matlab is done by obtaining the frames of the last figure shown, and these frames must all have the same resolution, and in case we want to show the complete rotated image, it should be decimated to fit in the same coordinate system:

```
>> input = [ 45 ; 1 ; 20 ; 0  ; 1 ];
>> videoAuto(input)
```

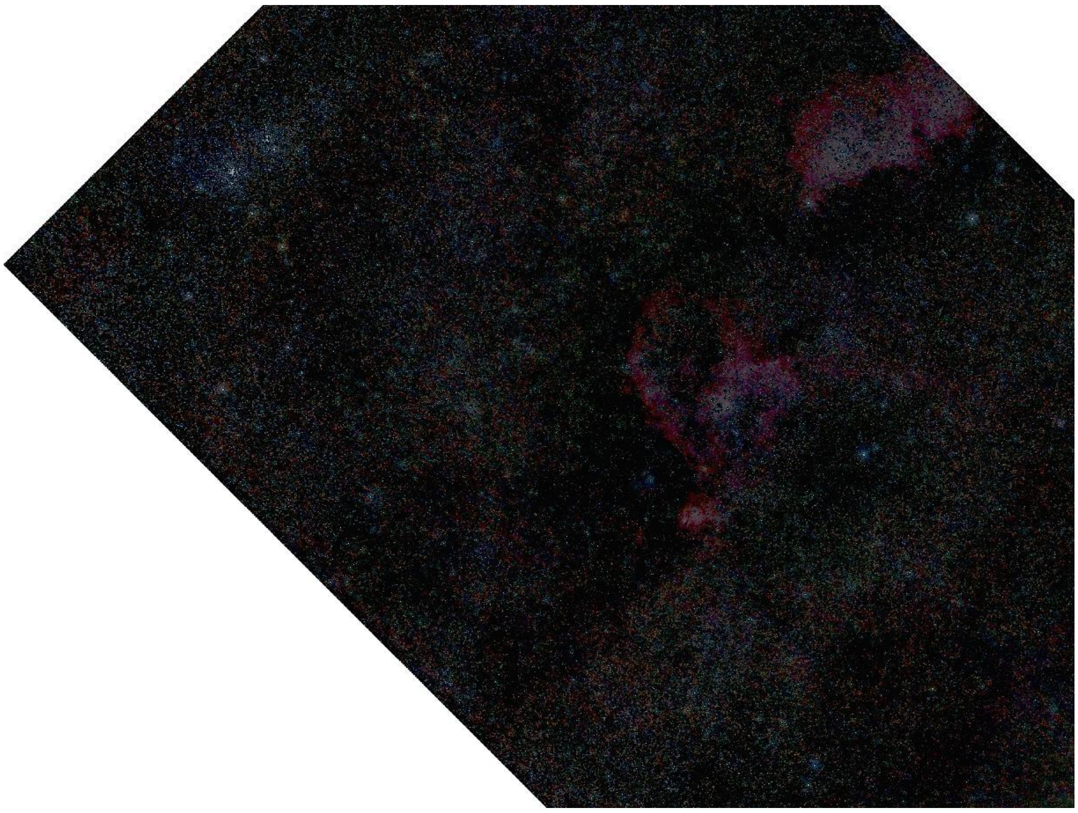


*Figure 4: Image rotated 45º and shifted 20% in x axis*

The problem appears when it's needed to visualize one element in more detail, for which is needed to apply a large zoom factor. This causes that, from a certain level of expansion, it is necessary to perform an interpolation to continue visualizing the image with the same resolution, with which it spatial resolution and, therefore, information is lost.

It is at this point that the transition to the other image must be made. This new image has been taken by another tele objective with a greater focal length and therefore a lower angle of view. When having the 2 images the same resolution, what we have are 2 images with the same number of pixels but showing a part of the sky of smaller size. This is due to the relationship between focal length and the angle of view in the cameras. We can see it in the next figure:

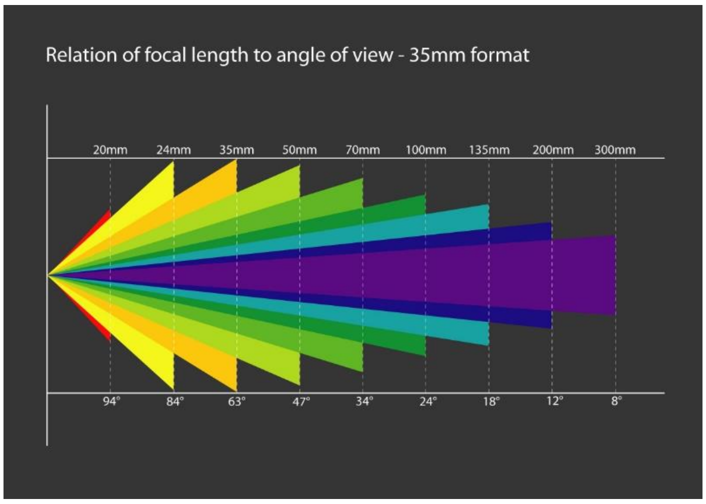
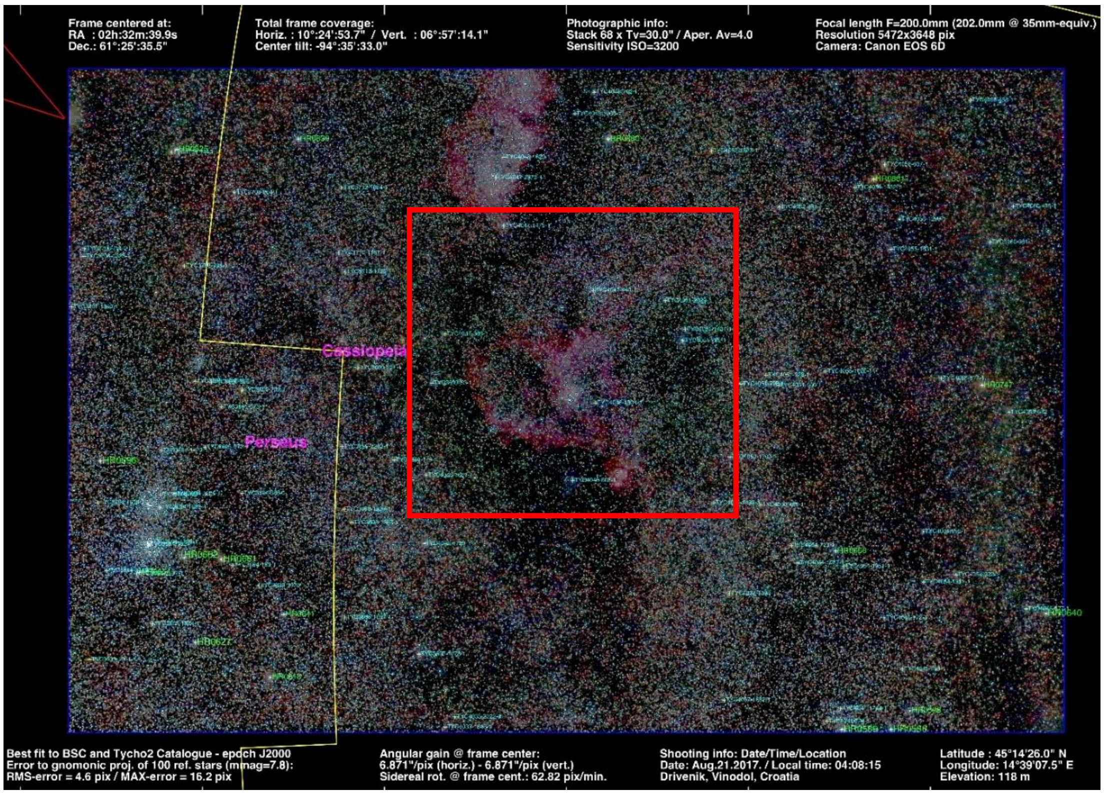


*Figure 5: Relation between focal length and angle of view*

We see how a small focal length implies a large angle of vision, so the images represent a large part of the scene at the expense of losing spatial resolution, while with a large focal length the effect is the opposite, it has a higher spatial resolution, but a smaller part of the scene is collected. In this case we have 2 images taken with focal lengths of 200mm and 500mm, so their viewing angles are 12º and 5º respectively. We can see these images below:


Frame centered at:
RA : 02h:32m:39.9s
Dec.: 61°:25':35.5"

Total frame coverage:
Horiz. : 10°:24':53.7" / Vert. : 06°:57':14.1"
Center tilt: -94°:35':33.0"

Photographic info:
Stack 68 x Tv=30.0" / Aper. Av=4.0
Sensitivity ISO=3200

Focal length F=200.0mm (202.0mm @ 35mm-equiv.)
Resolution 5472x3648 pix
Camera: Canon EOS 6D

Best fit to BSC and Tycho2 Catalogue - epoch J2000
Error to gnomonic proj. of 100 ref. stars (mmag=7.8):
RMS-error = 4.6 pix / MAX-error = 16.2 pix

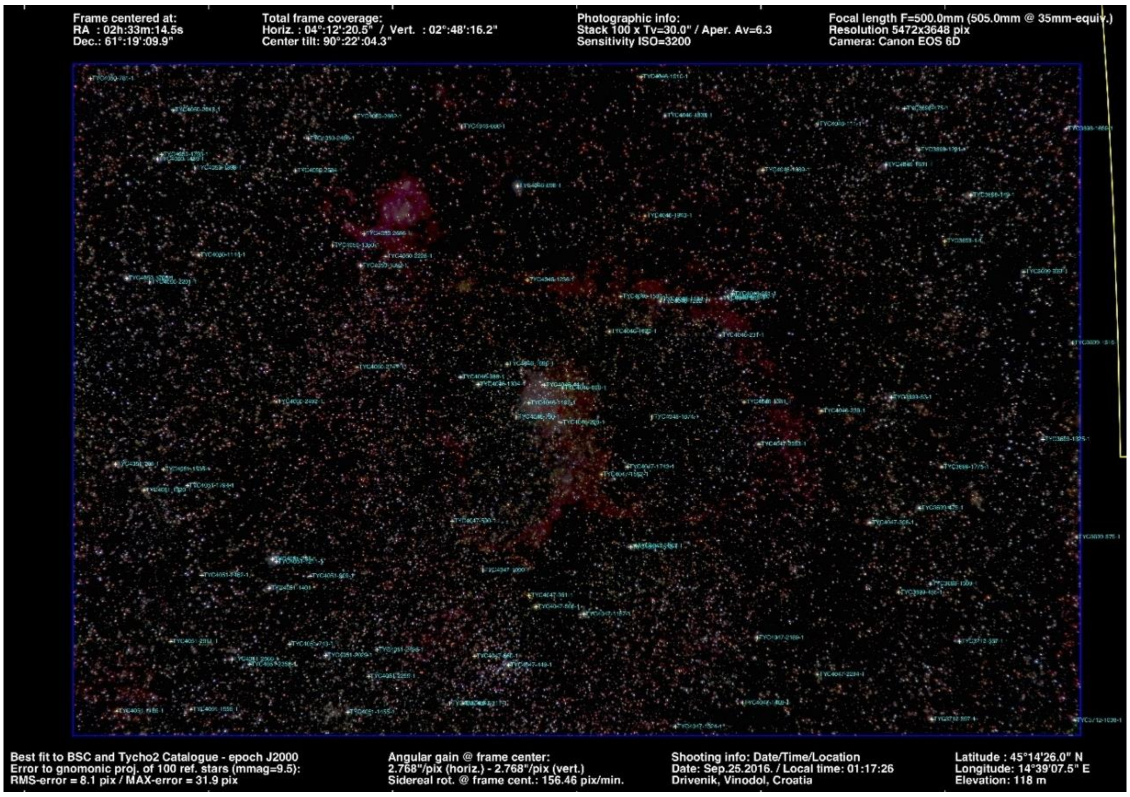
Angular gain @ frame center:
6.871"/pix (horiz.) - 6.871"/pix (vert.)
Sidereal rot. @ frame cent.: 62.82 pix/min.

Shooting info: Date/Time/Location
Date: Aug.21.2017. / Local time: 04:08:15
Drivenik, Vinodol, Croatia

Latitude : 45°14'26.0" N
Longitude: 14°39'07.5" E
Elevation: 118 m

*Figure 6: Image with lower focal length (with annotations)*

Above we have the image with bigger angular width and, therefore, that show the biggest region of the sky. The red square indicates the region that is shown in the image with less angle of view, which is shown next.



Frame centered at:
RA : 02h:33m:14.5s
Dec.: 61°:19':09.9"

Total frame coverage:
Horiz. : 04°:12':20.5" / Vert. : 02°:48':16.2"
Center tilt: 90°:22':04.3"

Photographic info:
Stack 100 x Tv=30.0" / Aper. Av=6.3
Sensitivity ISO=3200

Focal length F=500.0mm (505.0mm @ 35mm-equiv.)
Resolution 5472x3648 pix
Camera: Canon EOS 6D

Best fit to BSC and Tycho2 Catalogue - epoch J2000
Error to gnomonic proj. of 100 ref. stars (mmag=9.5):
RMS-error = 8.1 pix / MAX-error = 31.9 pix

Angular gain @ frame center:
2.768"/pix (horiz.) - 2.768"/pix (vert.)
Sidereal rot. @ frame cent.: 156.46 pix/min.

Shooting info: Date/Time/Location
Date: Sep.25.2016. / Local time: 01:17:26
Drivenik, Vinodol, Croatia

Latitude : 45°14'26.0" N
Longitude: 14°39'07.5" E
Elevation: 118 m

*Figure 7: Image with greater focal length (with annotations)*

These two images are showing the same part of the sky, but the second one has a focal length much bigger so what we see is the central part of the large image with a magnification of 2.5 (500mm and 200mm).

In addition, it can be verified that the image with the smallest angle of view is rotated 180º with respect to the largest one. Knowing this data, a linear transformation can be applied to the large image to show only the part that corresponds to the small one. The transformation matrix is defined for a 180º rotation and a zoom factor of 2.5, and after passing it to our function as an input, the result is as follows. In figure 8 it can be seen the image with greater focal length transformed in order to show exactly the same region as the other image.

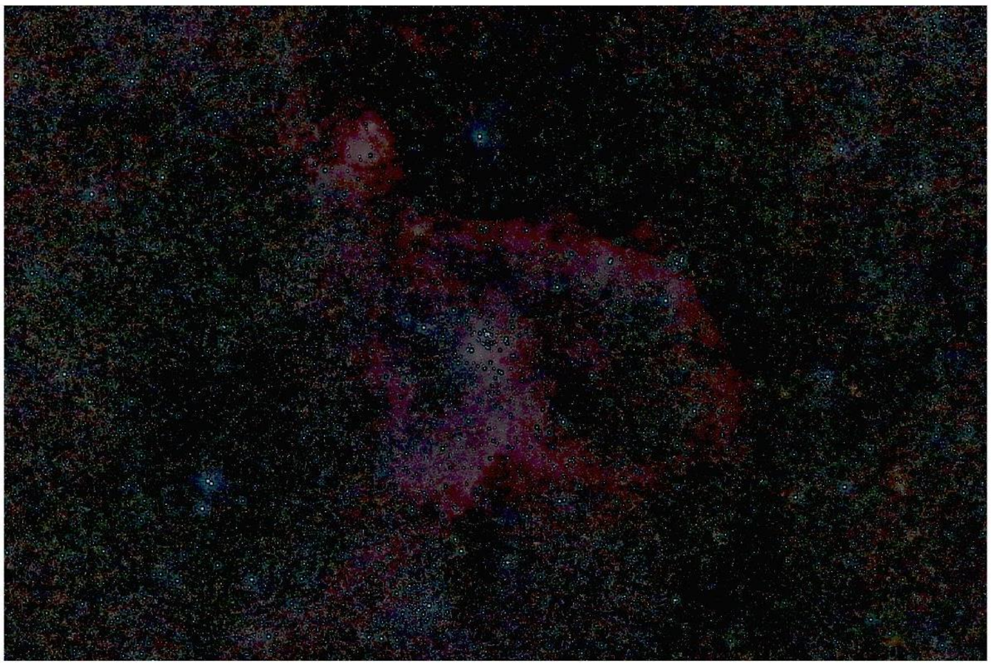


*Figure 8: Image transformed for showing the same part as the other*

Now, the original smaller image is shown in figure 9, so we can see, as indeed, how both images are showing exactly the same region of the space. The differences between these 2 images are because they have been taken with different camera configurations. In the first one (bigger), a long-time of exposure time has been set so we can see more elements and at the same time with more brightness while, in the second, the exposure time is shorter, so we have a darker image but a better spatial resolution with which it is easier to distinguish some elements from others. This is the reason why we want to change from one image to the other when we want to see one element in detail.

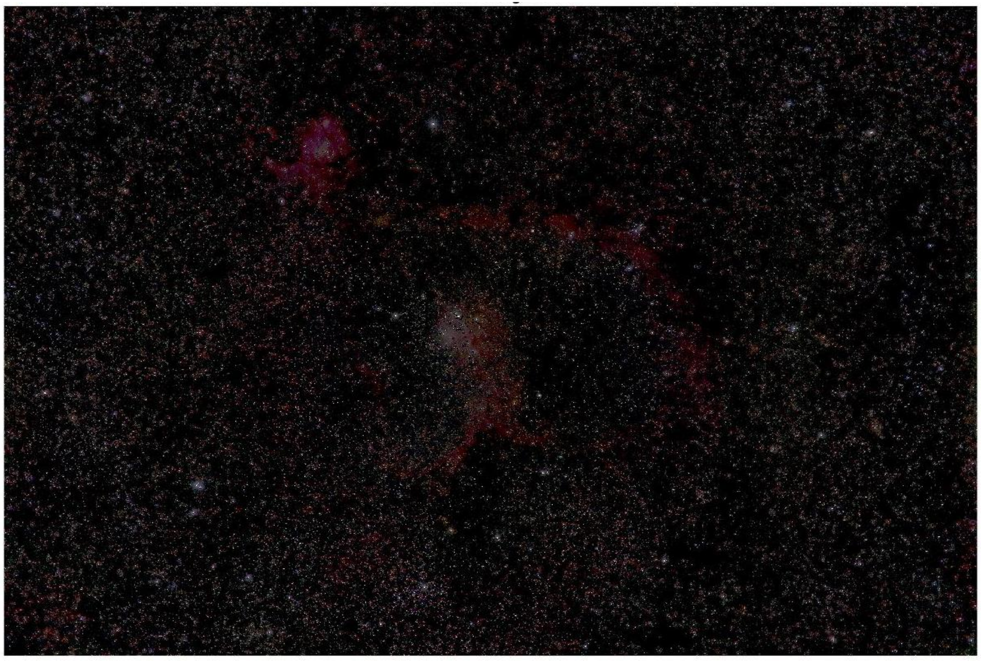


*Figure 9: Smaller image original*

The problem in making this transition is that, in photography, the optic distortion is associated with zoom lens, particularly large-range zooms, and depends on focal distance. We are working with tele objectives of long range and with huge focal distances, so this problem is especially critical in our case. Each of the lenses introduce a different distortion as we can see in the characteristics given by the manufacturers. In the following figure is represented this distortion, known as radially symmetric, which is provided for the 200mm lens.

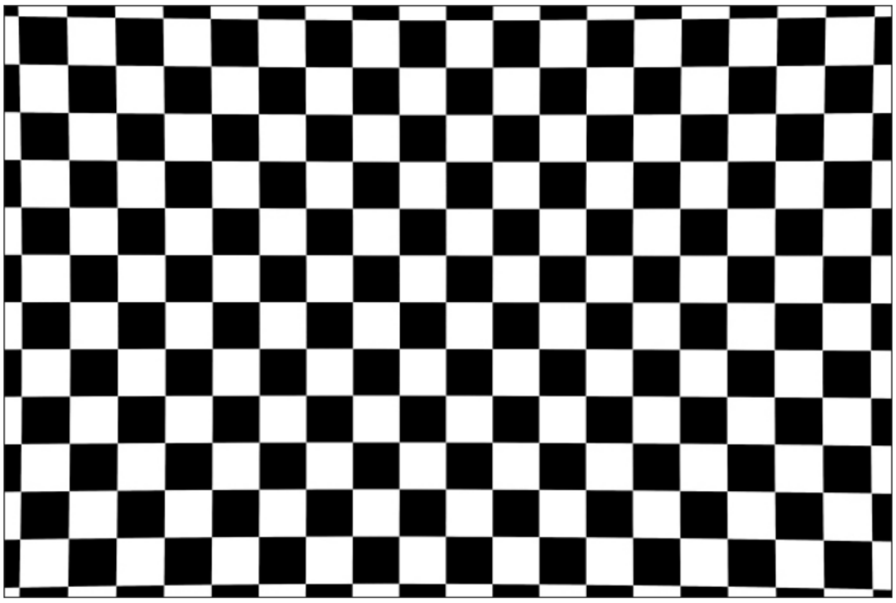


*Figure 10: Distortion introduced by the lens*

As we can see, the lines keep parallel in the centre, but when the lines are close to the corners start to bend. This effect is caused by the spherical shape of the lenses and to avoid this phenomenon, non-linear transformations is needed.

### *3.1.2 Non-linear geometric transform*

The method chosen to correct the images in order to avoid the deformation has been to perform a polynomial transformation by means of control points that transform the image of the largest angular width for showing the same region as the smallest, thus, the distortion of one of the images, specifically the bigger one, is modified to be the same as the other. With this operation we get a new image that although still have some deformation due to the lenses, but this distortion will be the same in the 2 images, so all the elements of the image will match in the same points and we will get a smooth transition.

This transformation will be a third-degree polynomial type and will be calculated by a function through a series of control points. The method of control points consists in establishing a series of points distributed throughout the original image, which we know exactly in which position they will be mapped in the new image. To establish the control points, the most characteristic elements of the image are used: the stars. If we know the exact position of enough stars that are present in both images it is possible to establish a relationship between the positions of these elements of the image that we want to transform and the positions of the stars of the reference image. With this data, the fitgeotrans.m function obtains the tform that best fits the new positions using least squares technique. This transformation can be any of the ones explained in previous sections, both affine and non-linear.

Another important point is the method to obtain the exact centre of each star, which is necessary for the control points to be as accurate as possible because the more precise is the position of the centre the more accurate is the transform. So, the centres were obtained through fitting a gaussian function. After the calculus of the centres were done, another function called GetControlPoints.m was implemented whose purpose is to get a matrix where in each row will be the position of each common element in both images. It's necessary to find out which stars are present in both images in order to establish the control points. There is a catalogue of the whole sky with information about all the stars like the ID, the position etc. where it's possible to realize a query only in certain region. GetControlPoints realize this query for the regions that correspond with our images and give the identifier to each star of our image. Once we have all the stars identified with their respective ID, we can check which stars appear in both images and then establish the control points.

Once the positions of the stars in our image are known, as well as the real positions where they should be, a commitment is established according to the number of CP used. With a small number of them, there will be areas that are not covered by any point, so the transformation in these areas can adopt undesirable behaviours that do not conform to the desired transformation. On the other hand, an excess of CP would make very difficult to find the exact solution by running the least-squares method, and the transformation would not be as accurate as desired. Taking this into account, it has been chosen to work with 100 CPs, with which the whole image is covered while a very approximate solution can be found.

At this point a checking was made to verify that the 3rd degree polynomial transform produces better results than the affine. For this, a transformation was applied to the image with the largest arc to show only the part that corresponds to the image with the smallest arc. This transformation was applied 2 times, the first linearly and the second by polynomial of 3rd degree.

```
obje = 'IC1805';
r_min = 163;
r_max = 400;

Rdefault = imref2d(size(I_b)); % The default coordinate system used by
imrotate

CP = GetControlPoints( obje,r_min,r_max );
u = CP(:,1);
v = CP(:,2);
x = CP(:,3);
y = CP(:,4);

tformAffine = fitgeotrans(CP(:,1:2),CP(:,3:4),'similarity');
tformNoLinear = fitgeotrans(CP(:,1:2),CP(:,3:4),'polynomial',3);

[um, vm] = transformPointsInverse(tformAffine,x,y);
errorLinear = sum(sqrt((u-um).^2+(v-vm).^2))/length(u);

[um2, vm2] = transformPointsInverse(tformNoLinear,x,y);
errorNoLinear = sum(sqrt((u-um2).^2+(v-vm2).^2))/length(u);
```

Once obtained, the inverse operation was performed by the function transformPointsInverse.m, which consists in obtaining the positions of the pixels of the original image from the calculated pixels trough an inverse transformation. Once the new values have been obtained, it is only necessary to calculate the mean square error with respect to the original positions, which should coincide if the transformation were exact.

Thus, with the linear transformation the error is about 2.3 pixels on average while with the polynomial it is less than 0,1 pixels. That means the polynomial transform works much better than the linear, in fact, is almost perfect.

### *3.1.3 Video realization*

Once this point is reached, the video is made, which will be composed of frames obtained through the transformations of the original image.

This video should be displayed in a full HD monitor without any previous transformation like down sampling or interpolation for avoiding the loss of information. Due to this feature, the video must have a full HD resolution, whose size is 1920x1080 pixels. This resolution does not correspond to the resolution of our images, which are much larger: 5472x3684 pixels. This difference forces the player to make a decimation in the image so that it can be adjusted in the Full HD monitor, which produces a loss of vital information in the spatial images due to many small elements can be found on it, so the loss of pixels means a loss of resolution that can carry out the complete or partial elimination of some elements. To avoid this reduction of the image, we have opted to visualize only a part of the image, with the full HD size, so it can fit perfectly in a full HD monitor.

For the creation of the video, the Matlab object VideoWriter has been used, whose function is to create a video file from an array or Matlab movie. The object contains information about the video and the properties that control the output video. With the videoWriter object created, it’s needed to get the frames for the video, and the function getframe.m is in charge of it. This function captures the current axes as it appears on the screen as a movie frame at the same size that it appears on the screen, so, if the image showed has full HD resolution, the video will have the same.

Finally, the video will have a resolution of 1902x1033 due to Matlab doesn’t allow to visualize the images at full screen, and it’s from these images where we get the frames.

Before proceeding to the realization of the video, the improvement introduced thanks to the polynomial transformation of 3$^{rd}$ degree is verified. For this, the original image and the transformed image will be shown overlapped, that is the effect that will take place when the transition between images will be carried out in the video. In this way, the difference between the 2 images is checked and, therefore, the effect that will be perceived when viewing the video. The more different are the images, the less aligned the elements will be when overlapping, which will create a blurring effect in the image and will be translated as a movement effect in the video. First, the original image will be shown, overlapped with the linearly transformed to see the problem to be solved. Subsequently, the original image and the transformed image through polynomials will be shown at the same time to see the difference and, therefore, the improvement introduced:

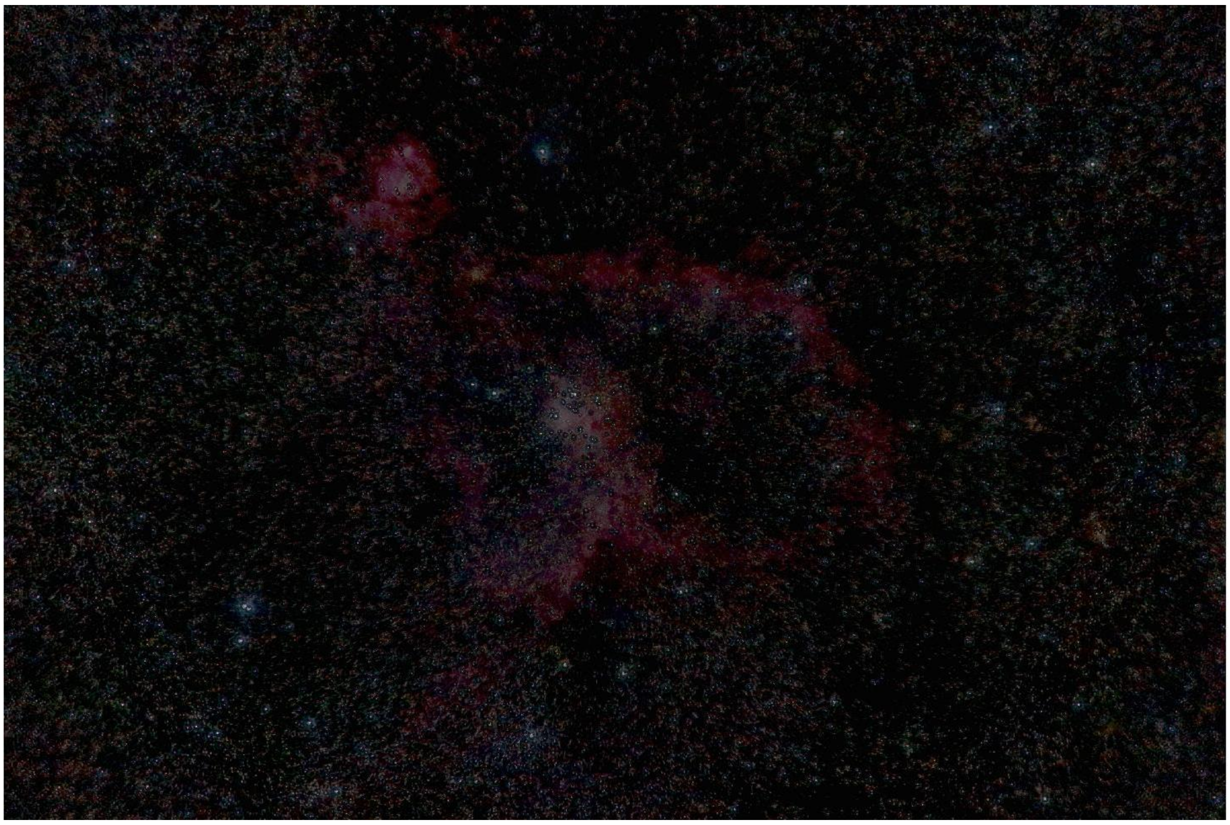


*Figure 11: Original and linearly transformed overlapping*

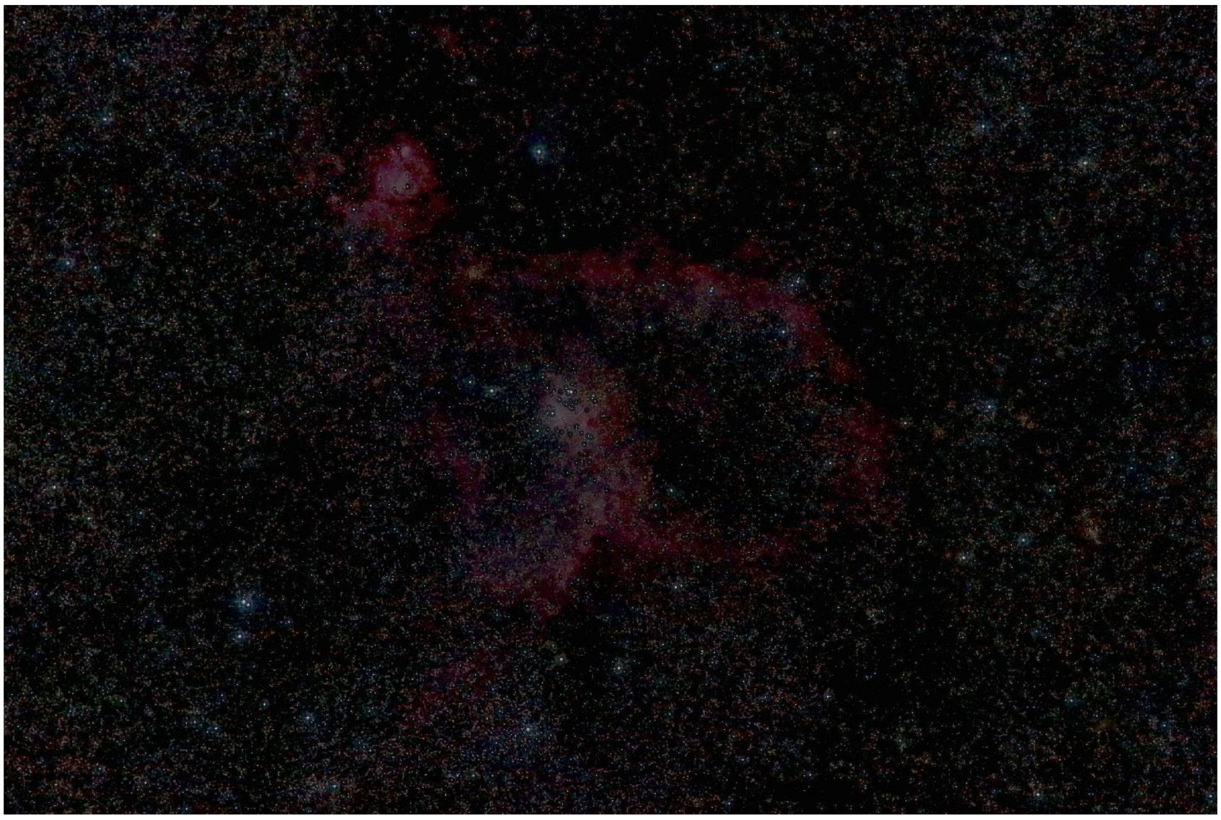


*Figure 12: Original and polynomial transformed overlapping*

The difference between these two images is observed at first glance, especially in the areas near the corners, where the effect introduced by the lens is more pronounced. While in the first image, corresponding to the original image overlapped with the linearly transformed, it is observed how near the corners the image appears blurred, it is difficult to distinguish some elements, this effect being greater as the smaller the element is. On the other hand, in the second image, corresponding to the overlapping between the original image and the one transformed through polynomial, these zones near the edges are seen much more clearly, being able to distinguish each element independently of its size.

To check this improvement in more detail, 2 more comparisons are made, as in the previous 2 images, but this time showing just one part of the image to see this effect with greater clarity. First, the improvement introduced in the centre of the image (where the central nebula is located) will be checked, and then, near the left down corner. These areas are boxed in the following image:

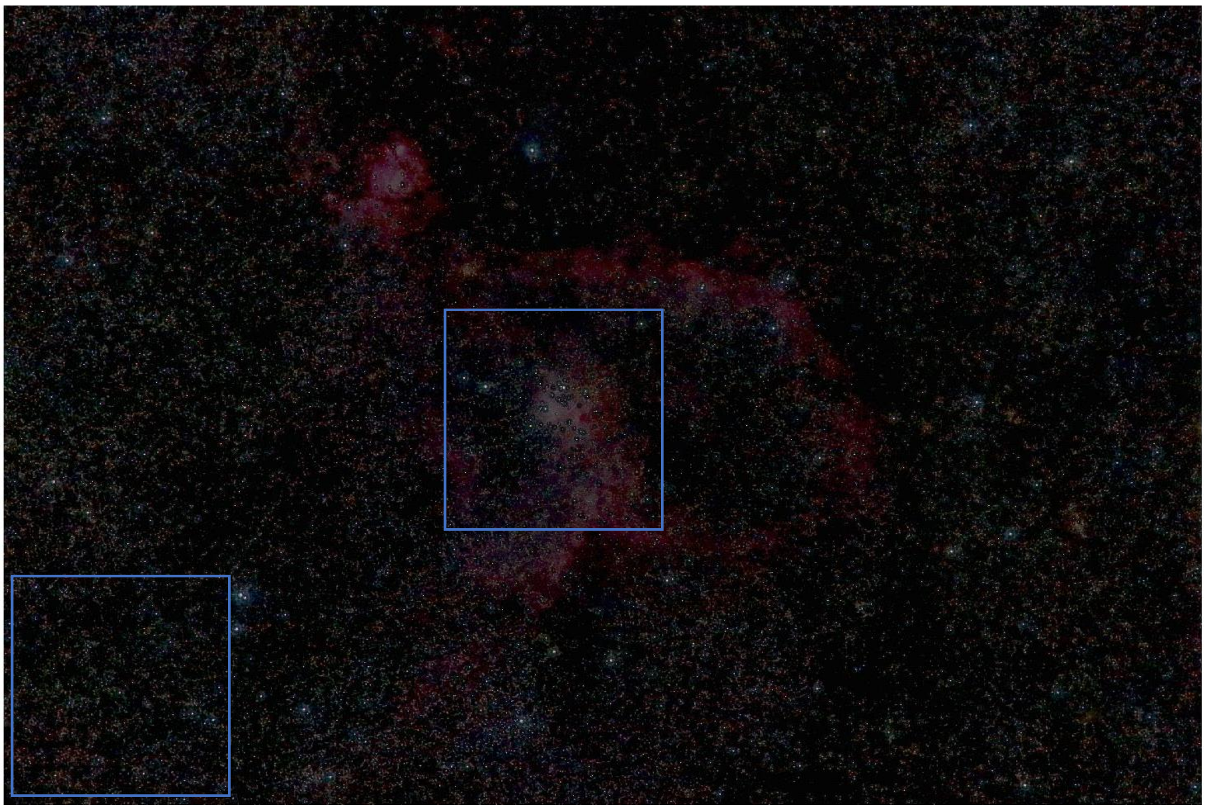


*Figure 13: Zones to see more in detail*

In principle, symmetrical radial deformation should not have a large impact on the central area as explained above. We proceed to make the comparison of this area:

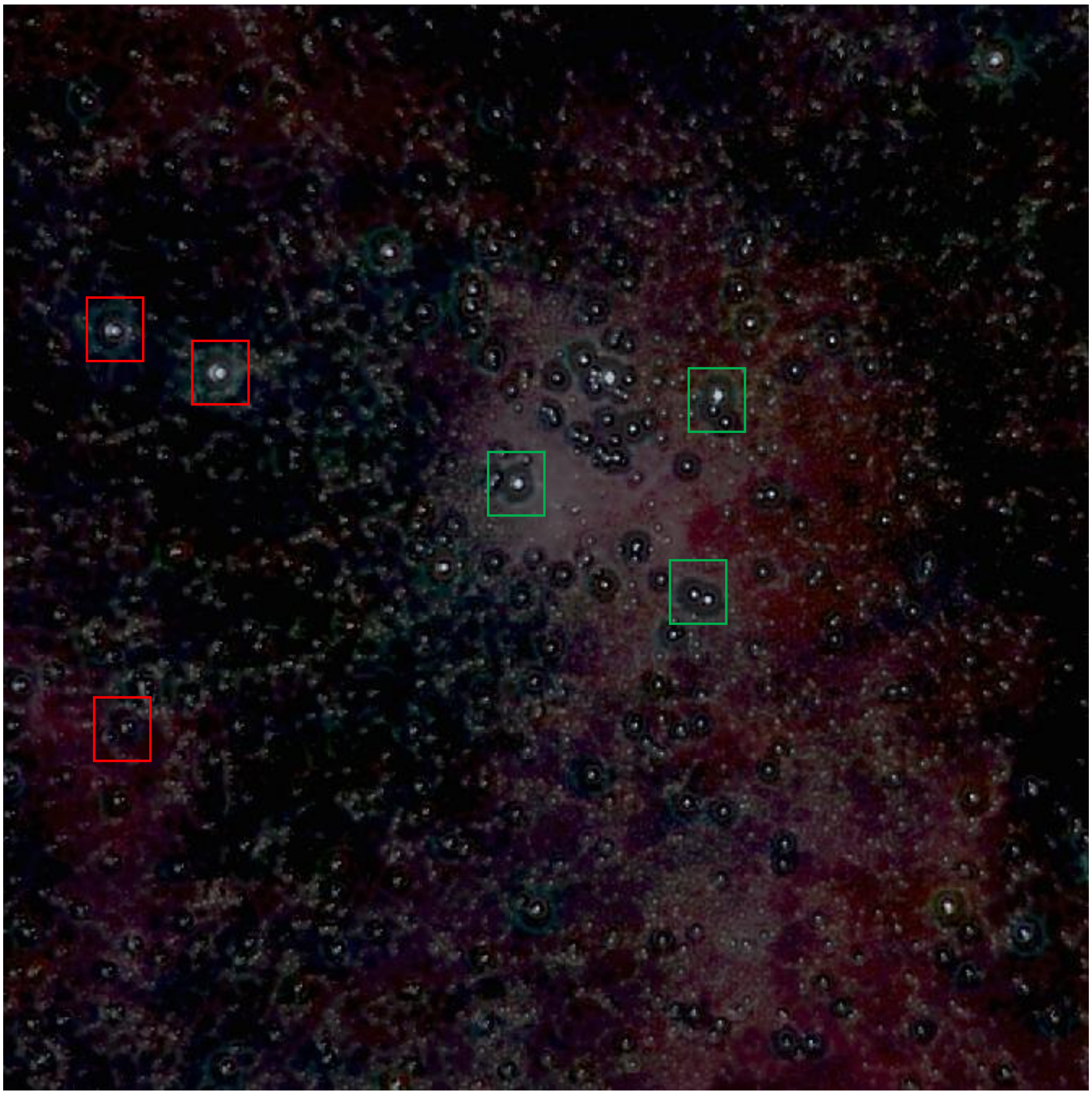


*Figure 14: Central zone of the original image overlapped with linearly transformed*

Here, it could be seen the central area of the image transformed linearly. It is clearly seen how the green boxed elements do not suffer any kind of distortion, so their position and size can be clearly seen. However, certain undesired effects begin to appear in this area, as can be seen in the elements indicated in red. When these elements do not coincide in the same position of the image, a partial duplication of some stars takes place, which translates into a loss of information. In addition, this effect will be clearly visible when the transition between the images is made in the video, since a change of position of certain stars will be observed, thus eliminating the desired continuity effect. We now proceed to show the result after the polynomial transformation.

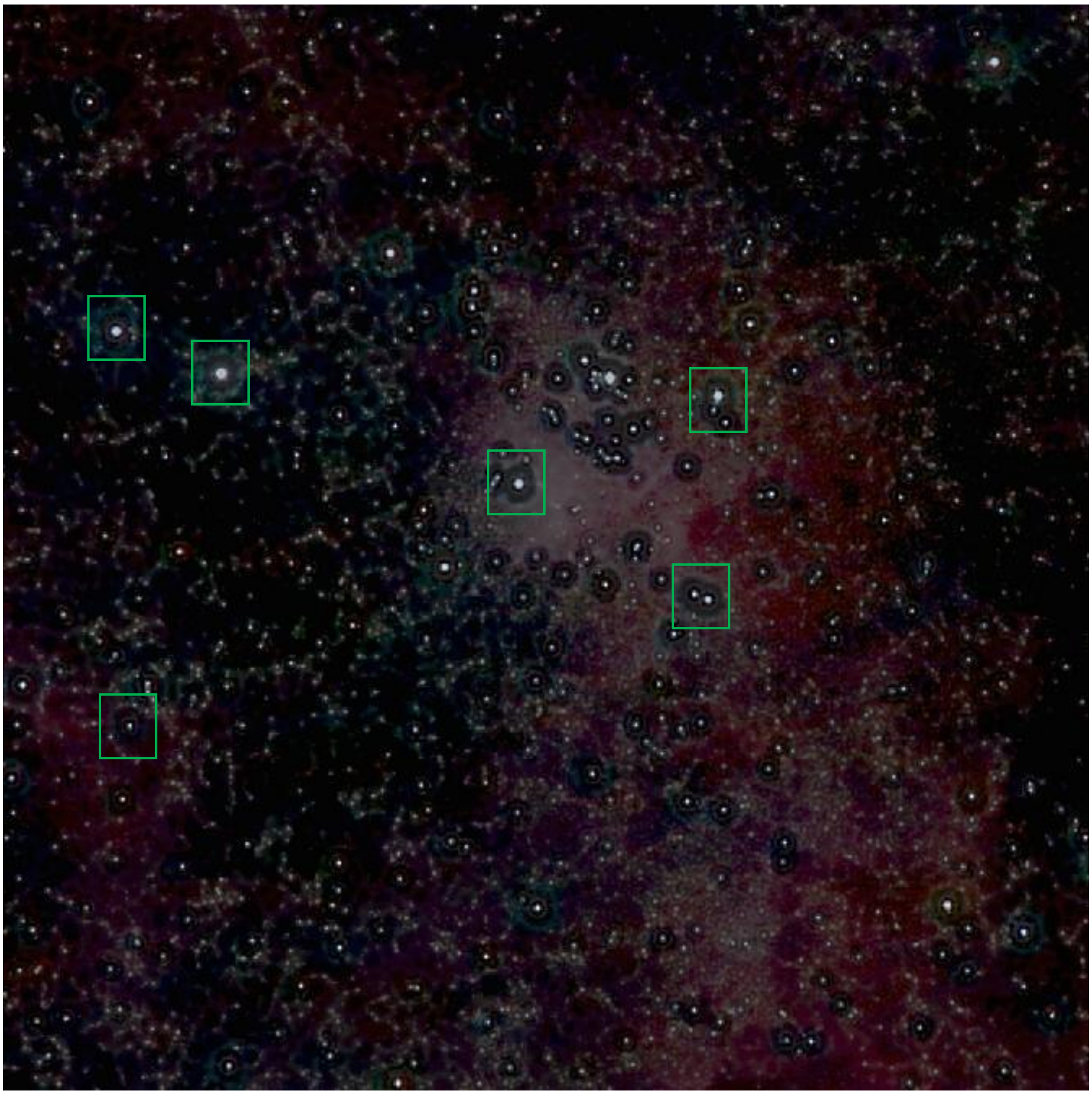


*Figure 15: Central zone of the original image overlapped with transformed through polynomial*

Figure 15 shows clearly the improvement introduced. The elements are all now marked with green due to they have no distortion. Now, all the elements of the image are shown unaltered, keeping their position and size despite being formed by 2 overlapped images. We now proceed to perform the same check, but in the lower left corner, where the distortion is more pronounced.

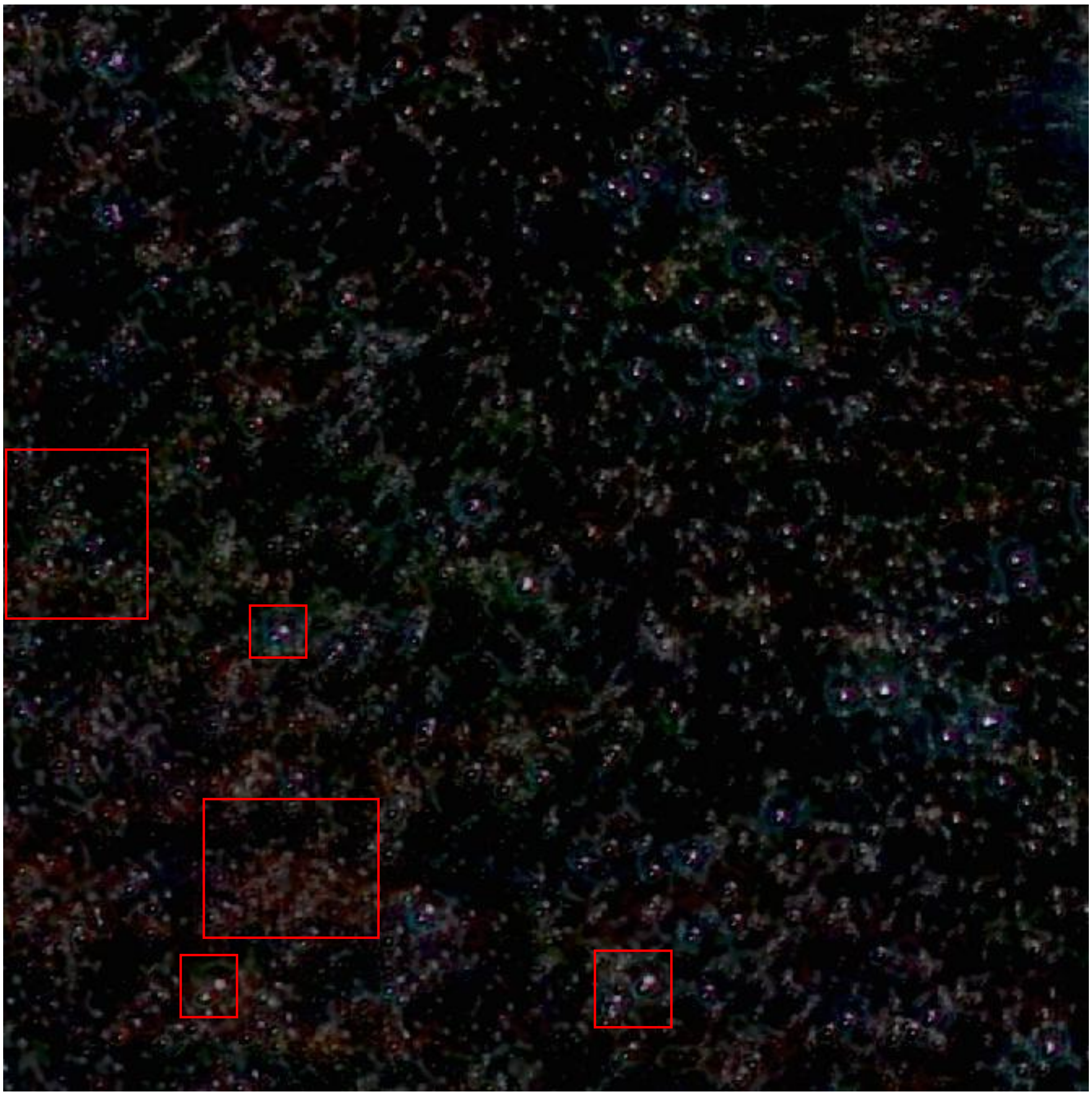


*Figure 16: Corner of the original image overlapped with the linearly transformed.*

Figure 16 shows some elements of the image pointed with small boxes that, as in the previous example, appear duplicated or blurred. However, the biggest problem in this area mostly affects star clusters or nebulae. These are large accumulations of stars produced in a small space, so the information that can be extracted is confusing because the light of some stars influences the others, which leads to a loss of quality. This effect is much more critical in this case due to the distortion. The misalignment of the elements causes them to mix completely, with which all the possible information is lost since the result is a simple spot where nothing can be clearly distinguished. In the following image it can be seen the result after applying the transformation using polynomials.

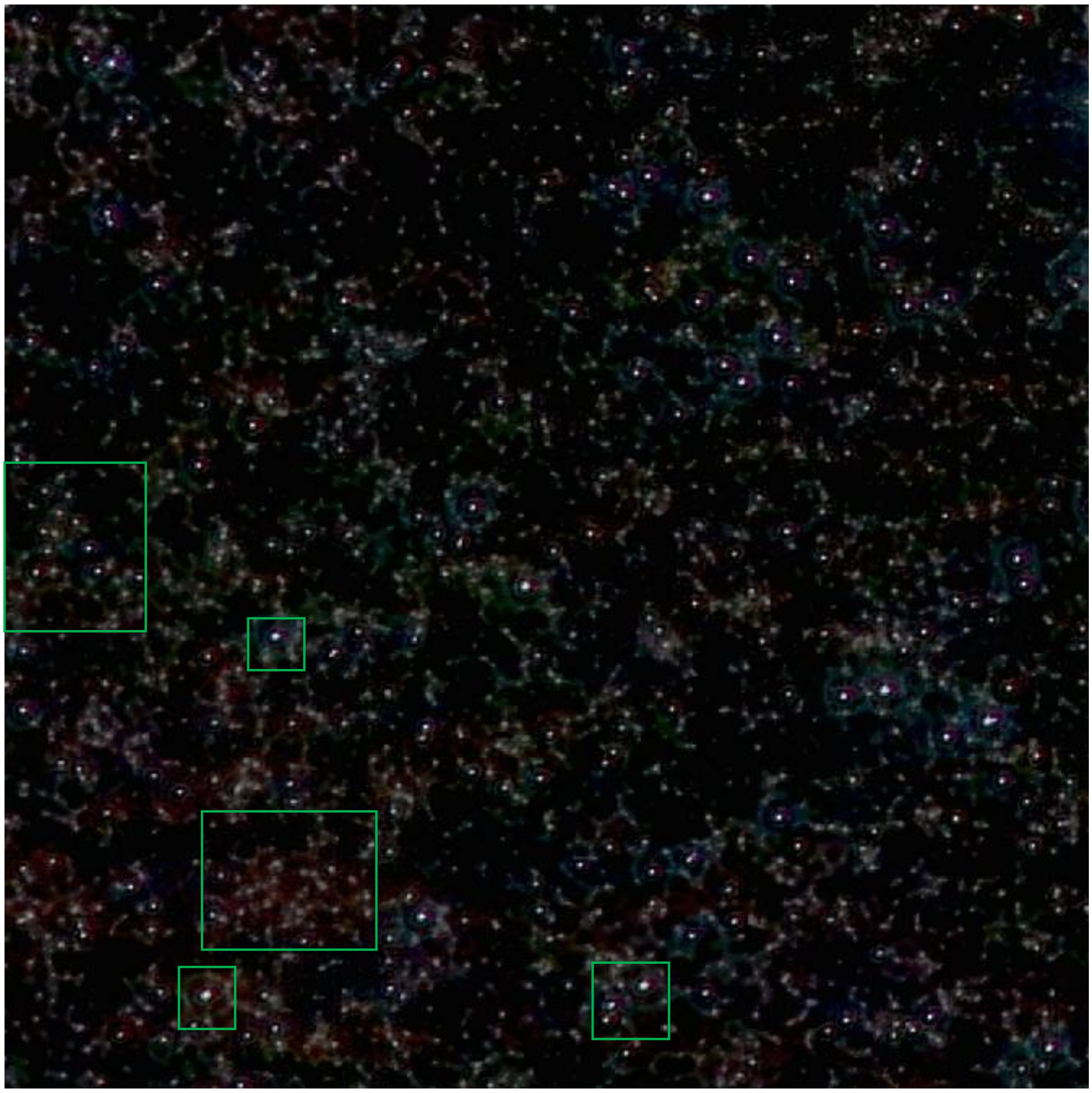


Figure 17: Corner of the original image overlapped with the transformed through polynomial.

In this case, as in the example corresponding to the central part, the mismatches of the punctual elements have been solved, these being completely aligned, but also, in this case, the mismatches in the concentrated or nebulous elements have also been corrected. As seen in the larger green boxes, in this case it is possible to distinguish the vast majority of stars in the image, even when they are in a zone with high density of stars.

Now it’s all ready to make the video from both images. The function used is the same for the linear transforms, so with a matrix we can define all linear transforms we want to apply, but now, another row is added, which is composed of zeros or ones, and this will denote when we want the transition to the other image. In the moment that in the last row a 1 appears, some transforms will be applied in order to show the same part of the other image to make the transition.

The final video will consist, as if it were a presentation about the observable sky, in a tour through the image focusing on the most remarkable elements of this piece of sky. Only basic linear transformations will be used for this first part. Once the most interesting areas have been covered, an analysis of the area where the 2 photographs have been taken will be made. To do this, the whole image will be transferred to the analysis area, the transition will be made to the other image to observe that part in more detail, and finally, to show the complete image again. In the following figure is a diagram of the video's route, where each box indicates the zone of visualization at each moment. It starts showing the whole image.

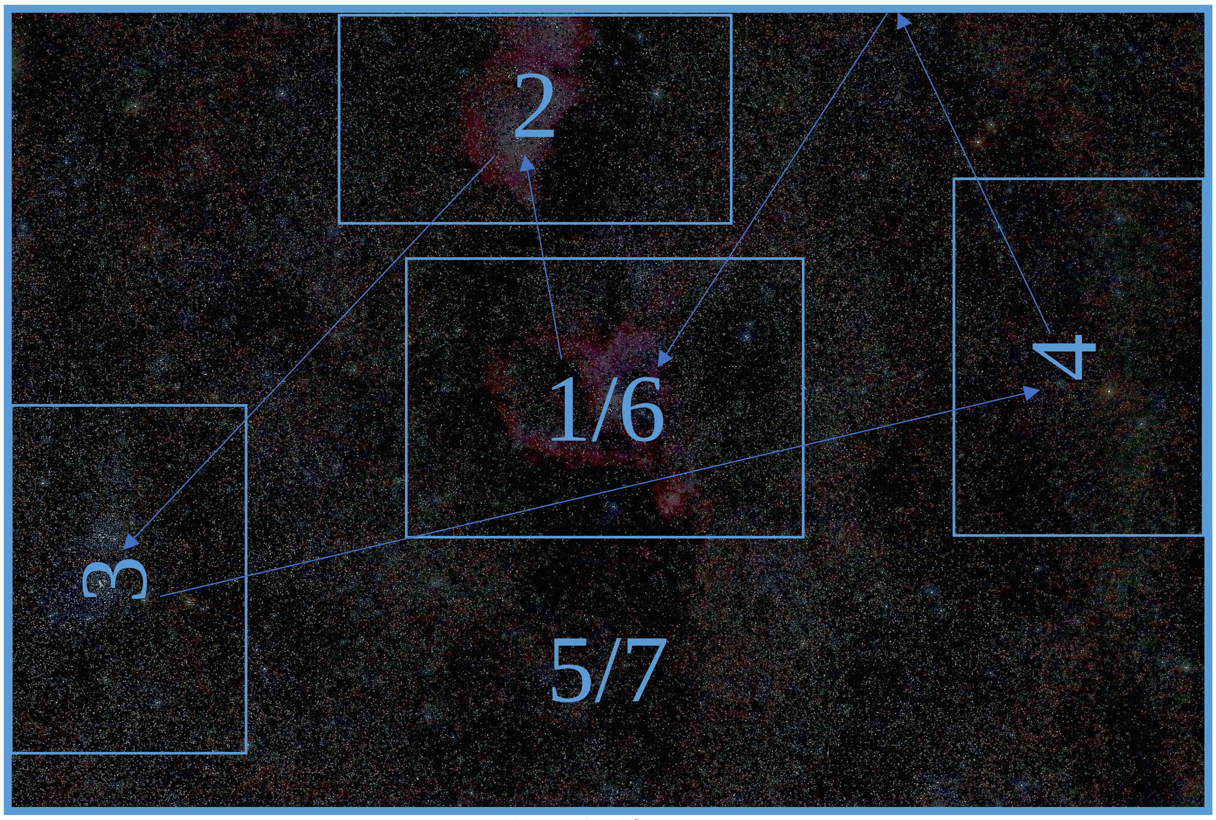


*Figure 18: Video route*

As we see in figure 15, the first action will be to apply a zoom to visualize the central nebula, which corresponds to the central part of the image, for which a zoom factor of 2.5 will be applied. The second step consists in a superior displacement to the left to visualize the nebula located in the upper part of the image (2). From this second zone, the image is transferred to zone 3 by means of a rotation of 90º and a displacement in both the X axis and the Y axis. This transition is made in 2 parts for a more comfortable visualization. From zone 3 the video will show zone 4, where several bright stars can be found, by means of a 180º turn. The next step is to visualize the complete photograph again, so at this point the first part of the video ends, having seen the most important elements in detail. At this time, we proceed to visualize the central area again, but this time with a rotation of 180º, in order to make it coincide with the other image with which we are working. Once placed in zone 6, the transition is made, in which no kind of movement is observed thanks to the geometric transformation by means of polynomials, to analyse that part more in detail. Finally, the last steps are undone to conclude the video showing the whole image again.

Below is the matrix used as input to carry out the realization of this video. As already explained above, each row corresponds to a transformation type, while each column indicates all the transformations carried out in each step. The program performed reads a column in each step and applies the indicated transformations before moving on to the next column. In this way, the first row represents the rotation in degrees, the second the zoom factor, the third and fourth refer to the displacement in X and Y respectively and the fifth row represents the number of steps in which you want to apply the transformation. In this case it has been chosen that all transitions are made in 80 steps to give a sense of continuity. The last row indicates when you want to transition between images.

```
>> input

                  input =

                    0    0   90    0  180   90  176  0  -176
                    2    1    0    2    1    0    2   1    0
                    0    8    0   35    0   36   -2   0    0
                    0   33    0    0    0  -12   -2   0    1
                   80   80   80   80   80   80   80   0   80
                    0    0    0    0    0    0    0   1    0
```

With the input matrix is possible to know that the realization of this video consists of 9 steps in total, in each one different transformations are applied. It is also known that all transformations will be made in 80 steps and that the transition will be made in the penultimate transformation. The name of the video is "videoFinal.avi" and it is attached to this report.

Now, different videos can be done to show, for example, different elements, focusing in one part of the image or applying a rotation for get another point of view etc. But the main point of the project has been already finished due to it has been possible to avoid the distortion introduced by different lenses. In next chapter some limitations of the algorithms and future improvements will be discussed.

# CHAPTER 4. CONCLUSIONS AND LIMITATIONS

Once the video is finished, we talk about the conclusions and limitations of this project.

After the completion of this project it has been shown how geometric image transformations can provide solutions to virtually all the needs related to images, the question is to know which one to use in each moment in what way.

We have also observed that symmetric radial distortion is more critical than initially expected, and that it can greatly affect depending on the type of image with which one is working, and the objective sought. In this case, this distortion was critical to our objective and the solution of the polynomials has managed to solve the problem perfectly.

On the other hand, the high degree of efficiency of non-linear transformations has been proven in addition to their versatility since they can be adapted to any desired transformation. Even if a 3rd order polynomial transformation was not enough, you can still use a polynomial of degree 4, which will have even more coefficients

The main limitation is that this video has been made using only 2 images since it has been done as an example of how this solution works. A really interesting improvement would be to make this software but with a much larger repository of images, with which we could get to make a dynamic map of all the visible sky, even adding other images with different angles of view, with what would be obtained a multi-level map.

Finally comment that this software could have interesting applications in the future if extended to all lenses, as these are always going to introduce a certain distortion, which can lead to many problems as we have seen.

# CHAPTER 5. REFERENCES

[1] González and Woods, Digital Image Processing, 3rd. Ed.

[2] https://es.mathworks.com

[3] http://dea.unsj.edu.ar/imagenes/recursos/Capitulo1.pdf

[4] Samuel Barreto Melo, Transformaciones geométricas sobre imágenes digitales, Facultad de Ciencias, Universidad Distrital Francisco José de Caldas.

[5] http://www.vision.uji.es/courses/Doctorado/FVC/FVC-T10-Geometricas-4p.pdf

[6] Mikkel B. Stegmann, Image Warping, Informatics and Mathematical Modelling, Technical University of Denmark