**CHAPTER 1**

**INTRODUCTION**

Haze in a scene degrades details and other information in the scene. For instance, the haze present in an outdoor scene which is characterized by heavy smog, fog, drizzle, smoke or other airborne particulate matter can obscure, and in extreme cases, hide objects in the scene. Distant objects suffer from the effects of haze more than closer objects since haze tends to be additive with distance. Thus, outdoor images tend to suffer from the effects of haze more than indoor images. As a result, in these images of hazy scenes, information regarding the scenes might be degraded or even lost.

Haze-induced images loss causes the performance of some image processing applications to suffer. For example, object recognition applications may require more processing time, more robust algorithms, multiple images (or frames thereof for sequential images) of the captured scene to function properly. Thus, whereas a face-recognition application might recognize detect, identify, track, etc. a face with one particular frame of a haze-free sequential image, it might take that same algorithm several (or more) frames from a similar, but hazy, sequential image to recognize the face. Haze also affects the performance of remote sensing applications, surveying applications, and other geo-physical applications because of the predominance of outdoor images processed by these applications. Should the haze be of sufficient thickness, these algorithms might fail to perform their intended functions altogether.

Haze removal is vital in images for applications mentioned above to function properly and produce the intended output. Various algorithms have been proposed to remove the haze and improve the quality of the image by many authors from past two decades based on statistical observation and various other assumptions. Recently, a Dehazing technique was proposed based on Dark Channel Prior, which not only removes the haze from a single image but also gives a depth map of the image as a by-product. This technique takes into consideration only a single image to dehaze in comparison to other techniques which takes two or more images of the same scene in different angles to remove the haze.

* 1. **ABOUT THE PROJECT**

Our project intends to remove the haze present in the image based on a key observation - most local patches in outdoor haze-free images contain some pixels whose intensity is very low in at least one color channel. Using this prior with the haze imaging model, we can directly estimate the thickness of the haze and recover a high-quality haze-free image.

Fig 1.1.1Hazy ImageFig 1.1.2Haze-Free Image

The Fig 1.1.1contains haze which degrades the image by hiding the details such as the trees, buildings, roof tops etc. which can be clearly seen in the haze-free image on the Fig 1.1.2. This degradation of images due to the addition of haze, mist and fog makes it less appealing and also unfit for vision based applications to successfully extract the properties of the image that it requires. For ex. Pictures taken from satellites are prone to addition of haze present in the atmosphere and processing of such images does not yield the required output as the details of the image are disturbed by the haze. Hence De-hazing of images is a very important task to be taken place before processing the image.

The aim of the project is to remove the haze present in the image and obtain a high-quality haze free image.

**1.1.1 PRACTICAL APPLICATIONS**

**UAV path detection:**

If haze is present in the atmosphere, then AI Algorithm's used to detect the path for landing the UAV may fail because of the presence of the haze in the image, which obscures the details in the image. At these situations we can dehaze the captured hazy image at runtime and provide dehazed image as the input image to the AI algorithm so that they can still detect the path and work correctly even in the bad weather conditions such as presence of fog, mist, haze etc

**Satellite surveying**:

The presence of haze in the atmosphere degrades the scene details captured by the satellite image capturing systems. Due to this satellite surveying is not possible all the time due to the bad weather conditions in the atmosphere. At these time when the image is degraded by the presence of haze, fog, smoke and other particles in the atmosphere, we can dehaze the captured hazy image, and still can perform the accurate surveying through satellite thereby improving the availability of the satellite surveying at all times even in the presence of the bad weather conditions.

**Detection of characters in the number plate of over speeding vehicle through CC cameras :**

Due the presence of the heavy haze in the atmosphere, the characters present on the number plate may not visible in the captured image of the over speeding vehicle. At these times the algorithms may not work properly and may fail to recognize the characters present in the image, since the presence of haze reduces the clarity of the image. Hence we can dehaze this captured hazy image and provide this dehazed image as the image to algorithm which detects the characters present in the number plate.

**Driverless Cars:**

Today Driverless cars are being developed where they depend on the AI algorithms to reach the destination and to detect the traffic signals, vehicles around them and others. In the heavy hazy condition the AI algorithms used to detect the vehicles in front of them may fail to detect the image at a larger distance, thereby leading to accidents. Hence we can incorporate this dehazing method to detect the traffic lights, vehicles moving in front of them at a larger distance and leading to accurate working of the algorithms even in the bad weather.

**Face recognition:**

Face recognition application may fail to detect face if the image is captured in the hazy, and thereby requiring more than one image to detect the face correctly. At this situation we can dehaze the hazy image and then provide the dehazed image as its input, thereby the application working correctly even the presence of the haze.

**Dehazing feature camera:**

As a feature in camera which dehazes the captured image after the hazy image is captured.

**1.2 EXISTING METHODS**

Research on removing the haze from images is taking place in a rapid pace from past two decades because of its importance in vision based applications and researchers are able to de-haze an image to a certain extent , although no algorithm is able to remove the haze completely which would otherwise result in loss of the originality of the image.

Most existing methods take into account many images of the same scene in different angles to know the thickness of the haze and the location of it.

For ex. Instant Dehazing of Images Using Polarization.

The main drawback is that, the exact number of images for a single scene cannot be determinate and hence may require quiet a number of images to remove the haze and is time consuming.

One notable haze removal technique from a single image is that proposed by Rannan Fattal, where de-hazing is achieved by interpreting the image through a model that accounts for surface shading in addition to the scene transmission. In this method, insufficient signal-to-noise ratio or the absence of multiplicative variation in significant portions of the image will cause the method to fail.

**1.3 PROPOSED METHOD**

A prior called the dark channel prior is proposed for single image haze removal. This prior is based on the statistics of outdoor haze-free images. In most of the local regions that do not cover the sky, some pixels (dark pixels) often have very low intensity in at least one color channel (RGB). In hazy images, the intensity of these dark pixels in that channel is mainly contributed by the airlight. Therefore, these dark pixels directly provide an accurate estimation of the haze transmission. Combining a haze imaging model and a soft matting interpolation method, we can recover a high-quality haze free image and produce a good depth map. This method can handle distant objects in heavily hazy images. The different steps to be carried out in this method are:

* Finding the dark channel
* Estimating the airlight
* Estimating the transmission map
* Refining the transmission map
* Obtaining the scene radiance

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 Instant Dehazing of Images Using Polarization**

This approach is based on the fact that usually air light scattered by atmospheric particles is partially polarized. This method works under a wide range of atmospheric and viewing conditions. The image formation process is analyzed taking into account atmospheric scattering. The image is then inverted to remove haze from images. The method can be used with as few as two images taken through a polarizer at different orientations.

In this paper[1][2][3] image formation process is modeled taking into account the polarization effects of atmospheric scattering. This model is used to recover the dehazed scene and also obtain information about scene structure and atmospheric properties. The image is composed of two unknown components – the scene radiance in the absence of haze, and airlight (the ambient light scattered towards the viewer). To recover these two unknowns we need two independent images. We easily obtain these images because usually airlight is partially polarized. The method only requires that the airlight induces some detectable partial polarization.

Image analysis that follows acquisition of polarization filtered images can remove the visual effects of haze. This approach enables dehazing when the problem cannot be solved by optics alone. In addition to the dehazed image, the method also yields information about scene structure and about the density and size distribution of the atmospheric particles. These results can form the basis for useful tools in photography and remote sensing**.**

Although this method is model based, no knowledge about the actual scattering particles is necessary. However, due to its basic principle, only Rayleigh scattering can properly be eliminated. Since Mie scatters polarize the light in a different way if any (larger the particle, the less the polarization). Also clear day recording is preferred for this method, since then the airlight polarization can properly be separated from the direct transmission.

**2.2 Single Image Dehazing**

In this paper[11] new passive method is used for recovering a haze-free image given a single photograph as an input. This is achieved by interpreting the image through a model that accounts for surface shading in addition to the scene transmission.

Here the airlight-albedo ambiguity is resolved by deriving an additional constraint that requires the surface shading and medium transmission functions to be locally statistically uncorrelated. This requires the shading component to vary significantly compared to the noise present in the image. In this method it does not assume the haze layer to be smooth in space, i.e., discontinuities in the scene depth or medium thickness is permitted.

In this method insufficient signal-to-noise ratio or the absence of multiplicative variation in significant portions of the image will cause our method to fail.

**2.3 Visibility in bad weather from a single image**

Tan presented a single image based dehazing method[4] in 2008. This proposal is based on the optical model.

The key observations made by Tan are:

* The output image must have better contrast compared to the input image.
* The variation of the values of atmospheric light A for the color components is dependent solely on the depth of the images of the objects, implying that objects with the same depth will have the same value of A, regardless their reflectance. Thus, the values of A for the neighboring pixels tend to be the same. Moreover, in many situations A changes smoothly across small local areas. Exception is for pixels at depth discontinuities whose number is relatively small.
* The input images that are plagued by bad weather are normally taken from outdoor natural scenes.

This algorithm is applicable for both color and gray images. However, it does not recover the scene’s original color. The computational time for 600x400 images, using double processors of Pentium 4 and 1 GB memory, takes approximately five to seven minutes. Also, this method has some flaws compared to other methods in terms of image quality.

**2.4 Improved Single Image Dehazing Using Geometry**

In outdoor surveillance applications, cameras are usually placed high in the air, and the depth of each pixel changes only slightly depending on whether a foreground object is present or not. As a result, one can enhance a series of images using the same depth map, as long as the atmospheric properties remain constant. However, errors in the depth map will become quite obvious, as the results are now being applied to a set of data that is different from that used for the estimation process.

This paper[6] assumes that neighboring pixels should have similar depths. The key observation is that weather degradation occurs in outdoor scenes, which means the majority of the images should exhibit the geometry of the camera located above a ground plane.

The previous work on fog enhancement has focused on statistical models for estimating the depth of each pixel. But this paper explores that the camera geometry can be exploited to improve the results of any statistical estimation technique. The expected monotonic trend can be implemented as a soft constraint within an energy minimization framework, which leads to a preference but not an absolute requirement. The geometric model is not limited to fog enhancement, but can be incorporated to other depth estimation techniques.

**2.5 Bayesian Defogging**

In this paper[7] a new method is used for estimating the optical transmission in hazy scenes given a single input image. Based on this estimation, the scattered light is eliminated to increase scene visibility and recover haze-free scene contrasts. In this new approach we formulate a refined image formation model that accounts for surface shading in addition to the transmission function. This allows us to resolve ambiguities in the data by searching for a solution in which the resulting shading and transmission functions are locally statistically uncorrelated. A similar principle is used to estimate the color of the haze. Results demonstrate the new method abilities to remove the haze layer as well as provide a reliable transmission estimate which can be used for additional applications such as image refocusing and novel view synthesis.

This can be achieved by interpreting the image through a model that accounts for surface shading in addition to the scene transmission. Based on this refined image formation model, the image is broken into regions of a constant albedo and the airlight-albedo ambiguity is resolved by denism, based on the transmission, for suppressing the noise amplification involved with dehazing. A user interactive tool is used for removing weather effects. This method requires the user to indicate regions that are heavily affected by weather and ones that are not, or to provide some coarse depth information.

This new approach is physically sound and relies only on the assumption that the transmission and surface shading are locally uncorrelated. The use of robust statistics allows us to cope with complicated scenes containing different surface albedos and the use of an implicit graphical model makes it possible to extrapolate the solution to pixels where no reliable estimate is available. Despite the challenges in resolving the ambiguities involved in this problem, the images produced by this method offer a lucid view of the scene and regain contrast that is typical to haze-free scenes.

**2.6 Bilinear Interpolation Dynamic Histogram Equalization For Fog Degraded Image Enhancement**

In this paper[8], a fog degraded image contrast enhancement method based on Bilinear Interpolation Dynamic Histogram Equalization is proposed in order to enhance the contrast of fog degraded image.

The main purpose of the Bilinear Interpolation Dynamic Histogram Equalization is to eliminate the domination and control the enhancement of the image detail. To eliminate the domination, the original histogram is divided into some sub-histograms. And then, a dynamic gray level range is distributed for each sub-histogram. At last, the gray levels are mapped by Histogram Equalization.

When the new dynamic gray level ranges are allocated, the loss of detail will be avoided by the Bilinear Interpolation as compared to Dynamic Histogram Equalization.

Results show that the BIDHE method can remove fog effect and enhance the definition and contrast of images effectively. This method also can be applied to the video sequences.

**2.7 Improved Dark Object Subtraction**

The Dark Channel Subtraction[13] is a method widely used in multi-spectral remote sensing systems and an improved version of it was inspiring for the dehazing method in the next section, the Dark Channel Prior, which is why DOS(Dark Object Subtraction) will now be summarized briefly. The DOS was developed many years before the improved version, but is now almost entirely used in its improved version today, as it was introduced by Chavez in 1988. The man idea behind this method is that in a scene at least one dark object exists, that has zero reflectance(is black). Also any measured radiance is attributed to atmospheric path radiance only. This also means, that the atmospheric transmittance is not corrected. The method needs at first to identify a dark object in the scene and then estimate the atmospheric light, which is additive term to the otherwise black pixel(if input image were haze free). This constant is then subtracted from all pixels in the image in order to remove the first-order scattering component. However in satellite imaging, where this technique is mostly used, the imaging sensor usually provides multi spectral data, such as from the Landsat Thematic Mapper, thus in the improved version the spectral bands are no longer treated separately from the scattering properties but dependent on the specific wavelength-scattering relationship, since atmospheric scattering is highly wavelength-dependent. This leads to much better results when using color images, since each color band will have its own dehazing iteration.

Chavez’s method “allow the user to select a relative atmospheric scattering model to predict the haze values for all the spectral bands from a selected starting band haze value. The improved method normalizes the predicted haze values for the different gain and offset parameters used by the imaging system”. This was also one of the first dehazing methods that does not need further information about the scene than those already contained within the image data. Difficulties arise in scenes where no dark object is present, which is a rare case in satellite images with large amounts of pixels, where it is statistically very likely to have also dark pixels, arising from shadowed areas. However these scenarios where dark objects are missing do exist.

**CHAPTER 3**

**SOFTWARE REQUIREMENT SPECIFICATION**

**3.1 REQUIREMENT ANALYSIS**

**3.1.1 SCOPE**

The scope of this project is to remove the undesirable effects such as haze, fog, smog or hue present in the given input to a high-quality haze-free image. These images are used by most of the vision based applications, as they require images with less or no unwanted properties present in the image.

**3.1.2 FUNCTIONAL REQUIREMENTS**

The functional requirements, also known as the functional specification are statements of services the system should react to particular inputs and how the system should behave in particular conditions.

* The images given as input contain haze, smog or hue.
* For every image, a sequence processing of the image is done.
* If the haze is present in the system, then it must be effectively removed.

**3.1.3 NON-FUNCTIONAL REQUIREMENTS**

The non functional requirements are constraints on services or functions offered by the system. They include timing constraints, constraints on development process and standards. They often apply to system on a whole.

* Reliability – The system must be reliable and must give accurate outputs for removing the haze
* Efficiency – The system should be efficient to work on many cases and hence reducing failure rates.
* Usability – The system must be user friendly and must be simple to use and implement.
* Robustness – The system must be able to restart after a failure.

**3.2 INPUT/OUTPUT REQUIREMENTS**

**3.2.1 INPUT CHARACTERISTICS**

* The input image is of .jpeg/.jpg format.
* Distribution of haze is random, hence can be present anywhere within the image.
* The concentration of the haze in the image can also be of different densities depending on where the picture was taken.

**3.2.2 OUTPUT CHARACTERISTICS**

* The output image can be saved in .jpeg/.jpg, .png, .bmp formats depending on the user requirements.
* The output image is haze-free and the details of the image can be found.
* There will be more clarity in the obtained high-quality haze free image.

**CHAPTER 4**

**SYSTEM DESIGN**

**4.1 ARCHITECTURAL DIAGRAM**

Compute Dark Channel

Compute Airlight

Upload Hazy image

Compute Transmission Map

GUI

Compute Refined Transmission Map

Obtain Scene Radiance

Display Dehazed Image

Fig 4.1 Architectural Diagram

The Fig 4.1 illustrates the overview of the project where the user uploads the hazy image and the step by step process is carried out by the user interface to obtain a dehazed image as the output.

**4.2 DATA FLOW DIAGRAM**

It is the graphical representation of the stepwise workflow of the system.

DARK CHANNEL FOR EACH PATCH

AIRLIGHT

TRANSMISSION MAP

REFINED TRANSMISSION MAP(BACKGROUND PROCESS)

OBTAIN SCENE RADIANCE (DEHAZED IMAGE)

Fig 4.2 Data Flow Diagram

Fig 4.2 gives the schematic outline of the process carried out by the user interface where the thickness of the haze and exposure to airlight are estimated and the dehazed image is obtained.

**4.3 SEQUENCE DIAGRAM**

Sequence diagram is a pictorial representation of the step by step process of the various actions performed by the actors in the system to get the desired output.

**Transmission map**

**Refined Transmission map**

**Dark channel**

**Airlight**

**Browse**

**User**

**Display Dehazed Image**

Choose image

Gray image

Density of haze

Alpha map

Refined map

Dehazed image

Fig 4.3 Sequence Diagram

In Fig 4.3 the user chooses the image to be dehazed and then the thickness of the haze and airlight is estimated. The haze is being removed and finally the image is refined to get the dehazed image.

**4.4 USE-CASE DIAGRAM**

The usecase diagram is a simple representation of the interaction between the user and system.

IMAGE DEHAZING

USER

USER

Fig 4.4 Use case diagram

In Fig 4.4 the user will browse the image to be dehazed and further refining steps are carried out by the user interface to get the dehazed image as the output.

**CHAPTER 5**

**SYSTEM IMPLEMENTATION**

**5.1 OVERVIEW**

FIND DARK CHANNEL FOR EACH PATCH

AIRLIGHT

CREATE TRANSMISSION MAP

RECOVER SCENE RADIANCE

REFINED TRANSMISSION MAP



Fig 5.1 Overview

The Fig 5.1 shows a sample output of how the hazed image is converted to get the dehazed image as the desired output.

The goal of the programming or implementation phase is to translate the design of the system produced during the design phase in to code in a given programming language, which can be executed by a computer and that performs the computation specified by the design, implementation of any software is preceded by important decisions regarding selection of the platform, the language used, etc. These decisions are often influenced by several factors such as real environment in which the system work, the speed that is required, the security concerns, other implementation specific details etc.

There are three major implementation decisions that have been made before the implementation of this project, they are as follows:

* Selection of the platform.
* Selection of the programming language for development of the dissertation work.
* Selection of the image processing toolbox for Image Matching.

**5.1.1 LANGUAGE**

* **MATLAB** is chosen for coding the whole project.
* **Image Processing ToolBox** is used to make use of the various in-built functions available that is prominently used in our project of Dehazing of images.

**Features**

* Matlab is a general purpose programming language with features economy for expression, modern flow control and data structures, and a rich set of operators.
* Matlab is a powerful and flexible language.
* Matlab is a portable language. Portability means that a matlab program written from ones computer system (an IBM pc foe ex.) can be compiled and run on another system(a DEC VAX system, perhaps) with little or no modification.

**5.1.2 PLATFORM/OPERATING SYSTEM**

An operating system is software that manages computer resources and provides users with an interface used to access those resources. An operating system performs basic tasks such as controlling and allocating memory, prioritizing system requests, controlling and internal system resources as a service to users and programs of the system.

The system under development works in a very restrictive environment. The security concerns are large and require that the system being developed be robust and safe from attack. Windows XP analyzes the performance impact of visual effects and uses this to determine whether to enable them. Windows XP operating systems can fix problem and add features by using service pack. The service pack is a superset of all previous service packs and patches so that only the latest service pack needs to be installed. an operating system process system data and user input and responds by allocating and managing tasks as to prevent the new functionality from consuming excessive processing overhead.

We can also use Windows Vista for as a platform because it is more secure than Windows XP. Even newer version of Matlab can be used in Windows 7.

**5.2 DETERMINING THE DARK CHANNEL**

The dark channel prior is based on the key observation on haze-free outdoor images: in most of the non-sky patches, at least one color channel has very low intensity at some pixels. That means, the minimum intensity in such a patch has a very low value. For an image J, we define

**Jdark(x) = min c ε (r,g,b) ( min y ε Ω(x) (Jc(y)))**

where **Jc** is a color channel of **J** and **Ω(x)** is a local patch centered at x. Observations says that except for the sky region, the intensity of **Jdark** is low and tends to be zero, if J is a haze-free outdoor image. We call **Jdark** the dark channel of **J**.

* Pad the pixels around the edges of the hazy image because the pixels at the edges are also considered to determine the dark channel.
* 15x15 patches are constructed around each pixel in the original image using the padded image.
* Minimum intensity pixel for every color channel in each patch is found.
* The result of the above procedure is the dark channel of the input hazy image.

Fig 5.2.1 Hazy image J Fig 5.2.2 Dark channel Jdark

The Fig 5.2.1 shows the original hazy image and Fig 5.2.2 corresponding dark channel of the image(Dark channel Jdark). The idea behind this is that, on haze-free outdoor images, at least one color channel has very low intensity at some pixels. That is, the Jdark for a haze-free image must almost be zero. In other words, the dark channel of the haze image will have higher intensity in regions with denser haze due to the addition of airlight.

The dark channel prior is partially inspired by the well known dark-object subtraction technique widely used in multi-spectral remote sensing systems where the spatially homogeneous haze is removed by subtracting a constant value corresponding to the darkest object in the scene.

**5.3 ESTIMATING AIRLIGHT**

The estimation of the airlight ***A*** is done by making use of the computed dark channel of the hazy image, since the dark channel of a hazy image approximates the haze denseness. Hence we can make use of the dark channel to detect the most haze-opaque region and to improve the estimation of the atmospheric airlight as compared to other methods of estimation of atmospheric airlight.

* Initially we pick the top 0.1% percent brightest pixels in the computed dark channel of the Hazy input image.
* These brightest pixels in the dark channel are the most haze-opaque region of the dark channel.
* Among these brightest pixels in the dark channel, the pixel with the highest intensity in the input image I corresponding to the brightest pixels in the dark channel is selected as the atmospheric light A of the hazy input image I.

The airlight for the above image is found to be 241 241 233.

**5.4 DETERMINING TRANSMISSION MAP**

Transmission map indicates the portion of the light that is not scattered and reaches the camera. Regions with more haze scatters light more and hence the transmission tends to zero, which in images makes such regions appear dark and the regions where haze is less relative to the hazy regions appear white or nearly white depending on the concentration of the haze in such regions.

* Rewriting the dark channel equation,

**Jdark(x) = minc ε (r,g,b) (min y ε Ω(x) (Jc(y)))**

* Since Jdark  tends to zero and as Ac , the corresponding channel of the atmospheric light is always positive, the equation may be written as,

**Jdark(x) = min c ε (r,g,b) (min y ε Ω(x) (Jc(y)) / Ac) = 0**

* This can be used to estimate the transmission for that patch by putting the above equation into the image formation model, however now in combination with the minoperator,

**minc (min y ε Ω(x) Ic(y)) = t~(x) minc (min y ε Ω(x) Jc(y)) + (1 - t~(x)) . Ac**

where t~(x) denotes the transmission in a local patch.

* Thus the transmission map can be found using

**t~(x) = 1 – minc ( min y ε Ω(x) Ic(y) / Ac)**

* As the atmosphere is not always completely free from dust particles or haze, a small amount of haze is left in the image in order to retain the originality of the image. This is done by keeping a small amount of haze for distant objects by introducing a constant parameter ω (o < ω < 1) into the above equation thus changing it to

**t~(x) = 1 – ω minc ( min y ε Ω(x) Ic(y) / Ac)**

Fig 5.4.1 Hazy Image J Fig 5.4.2 Transmission t~(x)

The Fig 5.4.2 shows us the transmission map for the hazy image.

**5.5 REFINED TRANSMISSION MAP**

Transmission map indicates the portion of the light that is not scattered and reaches the camera. Regions with more haze scatters light more and hence the transmission tends to zero, which in images makes such regions appear dark and the regions where haze is less relative to the hazy regions appear white or nearly white depending on the concentration of the haze in such regions.

The step by step method to compute the refined transmission map is as follows:

* Initially we find the laplacian matrix using the hazy image.
* First we build the 3\*3 patch around each pixel and then we compute the mean and covariance in that window.
* Then we form a diagonal matrix of size m\*n, for which the jth diagonal element takes the constant value of 1 and other diagonal elements of zero.
* Also we compute the vector alpha\* of lenght n, for which the jth element equals the already known alpha value of pixel j if j belongs to the window that is constructed around the pixel.
* Once the value of laplacian is computed then we are resizing the laplacian into the size of the original image i.e., m \* n .
* Once the laplacian is obtained then we solve the sparse linear system using the already known parameters of the equation where the regularization parameter is set to .0001.
* The refined transmission process is computed as a m\*n matrix at the background.

**5.6 OBTAINING SCENE RADIANCE**

The final step is to obtain the dehazed image using the above found parameters such as the airlight A, refined transmission map t(x) and the observed intensity I.

This is done using the equation

**J(x) = ( ( I(x) – A ) / max( t(x), to ) ) + A**

Scene radiance can be recovered using the transmission t(x), but t(x) can be very close to zero. The directly recovered scene radiance J is prone to noise. Therefore, we restrict the transmission t(x) to a lower bound to, which means that a small amount of haze is preserved in very dense haze regions.

Since the scene radiance is usually not as bright as the atmospheric light, the image after haze removal looks dim. Hence, we increase the exposure of J(x) for display.



Fig 5.6.1Hazy image Fig 5.6.2Haze-free image

The Fig 5.6.2 shows us the haze free image for the corresponding hazy image.

**CHAPTER 6**

**CONCLUSION AND FUTURE WORK**

**6.1 CONCLUSION**

The dark channel prior approach proposed to remove haze from a single hazy image is based on the statistics of outdoor images. When this approach is applied to the proposed model haze removal from a single image becomes simpler and more effective. The dark channel prior cannot be used for particular images where objects are similar to atmospheric airlight and have no shadow cast on them as the transmission for these objects will be underestimated.

This approach also works for gray scale images, if there are shadow regions in the image. The proposed method is proved to be better than the techniques previously used to remove haze from a single image. A numerical evaluation can be carried out between the existing technique and that of dark channel prior. the visibility is found to be better in a single image in which the haze is removed using dark channel prior.

**6.2 FUTURE WORK**

This proposed model will be enhanced to remove haze from image where objects are similar to atmospheric light and no shadow is cast on them. The haze removal based on models such as sun’s influence on sky region and the bluish hue near the horizon need to be investigated in our future work to enhance and add features to our model

**CHAPTER 7**

**REFERENCES**

**7.1 REFERENCES**

[1] S. G. Narasimhan and S. K. Nayar. Chromatic framework for vision in bad weather. CVPR, pages 598-605, 2000.

[2] Yoav Y. Schechner, Srinivasa G. Narasimhan and Shree K. Nayar. Instant dehazing of images using polarization, 2001.

[3] Srinivasa G. Narasimhan and S.K. Nayar.Vision and atmosphere. IJCV, 48:233-254, 2002.

[4] Robby T. TanVisibility in bad weather from a single image. CVPR, 2008.

[5] Kaiming He, Jian Sun and Xiaoou Tang. Guided Image Filtering, 2010.

[6] Peter Carr and Richard Hartley. Improved single image dehazing using Geometry. (CVPR) 2009.

[7] Ranaan Fattal.Bayesian defogging . Technical Report DU-CS-10-05 2010.

[8] Zhiyuan Xu and Xiaoming Liu. Bilinear Interpolation Dynamic Histogram Equalization for Fog-degraded Image Enhancement. (CVPR) 2010.

[9] Jing Yu and Qingmin Liao. Fast single Image fog removal using Edge-Preserving smoothing. IEEE 2011

[10] Louis Kratz and Ko Nishino. Factorizing Scene Albedo and Depth from a Single foggy image. Computer vision, IEEE 2009.

[11] Ranaan fattal. Single Image Dehazing. In SIGGRAPH, pages 1-9, 2008.

[12] Haoran Xu,Jianming Guo,Qing Liu and Lingli Ye Fast Image Processing using Improved Dark Channel Prior, IEEE 2012.

[13] Chavez. Improved Dark Object Subtraction, IEEE 1988

**CHAPTER 8**

**APPENDICES**

**8.1 SOURCE CODE**

**gui.m**

function varargout = gui(varargin)

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @gui\_OpeningFcn, ...

'gui\_OutputFcn', @gui\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

% --- Executes just before gui is made visible.

function gui\_OpeningFcn(hObject, eventdata, handles, varargin)

% Choose default command line output for gui

handles.output = hObject;

% Update handles structure

guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.

function varargout = gui\_OutputFcn(hObject, eventdata, handles)

% Get default command line output from handles structure

varargout{1} = handles.output;

% --- Executes on button press in browse\_button.

function browse\_button\_Callback(hObject, eventdata, handles)

% (EDIT TEXT <== SELECTED IMAGE ROW)

[filename,pathname,filterindex] = uigetfile({'\*.\*'},'Select the Hazy image');

Browsestring = 'Browse ';

set(hObject, 'String', Browsestring);

final\_path=strcat(pathname,filename);

I = imresize(imread(final\_path), 0.5);

for i = 7 : 12

subplot(2,6, i) ;

cla;

axis image off;

end

handles.hazy\_image=I;

subplot(2, 6, 7), imshow(I), title('Hazy image');

set(hObject, 'String',final\_path);

set(handles.editor\_airlight,'String', '---');

set(handles.status,'String', 'Image selected');

guidata(hObject,handles);

disp(get(hObject, 'String'));

return;

% --- Executes on button press in dark\_channel.

function dark\_channel\_Callback(hObject, eventdata, handles)

set(handles.status,'String','Computing Dark channel');

I = handles.hazy\_image;

J = find\_darkchannel(I);

handles.darkchannel = J;

subplot(2, 6, 8), imshow(J), title('Dark channel');

guidata(hObject,handles);

set(handles.status,'String','Dark channel computed')

% --- Executes during object creation, after setting all properties.

function name\_path\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

% --- Executes on button press in est\_airlight.

function est\_airlight\_Callback(hObject, eventdata, handles)

% find\_airlight function

set(handles.status,'String', 'Computing Airlight');

DC = handles.darkchannel;

I = handles.hazy\_image;

dims = size(I);

disp(dims);

dims = size(DC);

disp(dims);

brightest\_pixel = find\_airlight(DC, I);

A\_matrix = zeros(dims(1), dims(2), 3);

A\_matrix(:,:,1) = brightest\_pixel(1);

A\_matrix(:,:,2) = brightest\_pixel(2);

A\_matrix(:,:,3) = brightest\_pixel(3);

handles.airlight = double(A\_matrix) ./255;

set(handles.editor\_airlight,'String', num2str(brightest\_pixel));

guidata(hObject,handles);

set(handles.status,'String', 'Airlight Computed');

% --- Executes on button press in trans\_map.

function trans\_map\_Callback(hObject, eventdata, handles)

set(handles.status,'String', 'Computing Transmission map');

I = double(handles.hazy\_image) ./ 255;

A\_matrix = handles.airlight;

transmission\_map = find\_transmission\_map(I,A\_matrix );

subplot(2, 6, 9),imshow(transmission\_map), title('Transmission Map');

handles.transmission\_map = transmission\_map;

handles.hazy\_image = I;

guidata(hObject, handles);

set(handles.status,'String', 'Transmission map Computed');

% --- Executes on button press in refined\_trans.

function refined\_trans\_Callback(hObject, eventdata, handles)

set(handles.status,'String', 'Computing Refined Transmission map');

trans\_est = handles.transmission\_map;

im = handles.hazy\_image;

handles.ref\_trans\_map = find\_refined\_transmission\_map(im, trans\_est);

subplot(2, 6, 10),imshow(handles.ref\_trans\_map), title('Refined Transmission Map');

guidata(hObject, handles);

set(handles.status,'String', 'Refined Transmission map computed');

% Following code is associated with the push buttom "Dehazed Image"

set(handles.status,'String', 'Obtaining Scene Radiance');

I = handles.hazy\_image;

A\_matrix = handles.airlight;

transmission\_map = handles.ref\_trans\_map;

final\_image = find\_SceneRadiance(I, A\_matrix, transmission\_map);

subplot(2, 6, 11), imshow(final\_image), title('Dehazed image');

handles.dehazed\_image = final\_image;

imshow(handles.hazy\_image), title('Hazy Image');

imsave;

imshow(handles.dehazed\_image), title('Dehazed Image');

imsave;

guidata(hObject, handles);

set(handles.status,'String', 'Scene Radiance Recovered');

**find\_darkchannel.m**

function[dc] = find\_darkchannel(I)

[x y ~] = size(I);

patch\_size = 15;

I = padarray(I, [floor(patch\_size/2) floor(patch\_size/2)], 'symmetric');

for m = 1:x

for n = 1:y

patch\_r = I(m:(m+patch\_size-1), n:(n+patch\_size-1),1);

patch\_g = I(m:(m+patch\_size-1), n:(n+patch\_size-1),2);

patch\_b = I(m:(m+patch\_size-1), n:(n+patch\_size-1),3);

J(m,n,1) = min(patch\_r(:));

J(m,n,2) = min(patch\_g(:));

J(m,n,3) = min(patch\_b(:));

dc(m,n) = min(J(m,n, :));

end

end

**find\_airlight.m**

function [brightest\_pixel] = find\_airlight(DC, I)

[ r c v] = find(DC);

sorted\_v = sort(v, 'descend');

[x y ~] = size(DC);

no\_of\_pixels = x \* y;

top = ceil(.001 \* no\_of\_pixels);

mymin\_in\_vec = sorted\_v(top);

[nr nc nv] = find(DC >= mymin\_in\_vec);

no\_of\_elem = size(nr);

brightest\_pixel\_r = I(nr(1), nc(1), 1);

brightest\_pixel\_g = I(nr(1), nc(1), 2);

brightest\_pixel\_b = I(nr(1), nc(1), 3);

for m = 1 : no\_of\_elem

if (brightest\_pixel\_r < I(nr(m), nc(m),1))

brightest\_pixel\_r = I(nr(m), nc(m),1);

end

if (brightest\_pixel\_g < I(nr(m), nc(m),2))

brightest\_pixel\_g = I(nr(m), nc(m),2);

end

if (brightest\_pixel\_b < I(nr(m), nc(m),3))

brightest\_pixel\_b = I(nr(m), nc(m),3);

end

end

brightest\_pixel = [brightest\_pixel\_r brightest\_pixel\_g brightest\_pixel\_b];

disp(brightest\_pixel);

**find\_transmission\_map.m**

function [tranmission\_map] = find\_transmission\_map(TI, TA\_airlight)

w = 0.95;

tranmission\_map = 1 - w \* find\_darkchannel(TI ./TA\_airlight);

**find\_refined\_transmission\_map.m**

function t = find\_refined\_transmission\_map(im, trans\_est)

win\_size = 3;

win\_els = win\_size.^2;

l\_els = win\_els.^2;

win\_bord = floor(win\_size./2);

e = 0.000001;

[m,n,c] = size(im);

numpix = m\*n;

k = reshape(1:numpix, m, n);

U = eye(win\_size);

D = eye(win\_els);

num\_els = l\_els\*(m-2\*win\_bord)\*(n-2\*win\_bord);

ind\_i = ones(1,num\_els);

ind\_j = ind\_i;

els = zeros(1,num\_els);

count = 0;

for x = (1 + win\_bord):(n - win\_bord)

for y = (1 + win\_bord):(m - win\_bord)

wk = reshape(im(y-win\_bord:y+win\_bord,x-win\_bord:x+win\_bord,:), win\_els, c);

w\_ind = reshape(k(y-win\_bord:y+win\_bord,x-win\_bord:x+win\_bord), 1, win\_els);

[i j] = meshgrid(w\_ind, w\_ind);

i = reshape(i,1,l\_els);

j = reshape(j,1,l\_els);

ind\_i((count\*l\_els + 1):(count\*l\_els+l\_els)) = i;

ind\_j((count\*l\_els + 1):(count\*l\_els+l\_els)) = j;

win\_mu = mean(wk)';

win\_cov = wk'\*wk/win\_els-win\_mu\*win\_mu';

dif = wk' - repmat(win\_mu,1,win\_els);

elements = D - (1 + dif(:,1:win\_els)'\*inv(...

win\_cov + e./win\_els.\*U)\*dif(:,1:win\_els))...

./win\_els;

els((count\*l\_els + 1):(count\*l\_els+l\_els)) = ...

reshape(elements,1,l\_els);

count = count + 1;

end

end

L = sparse(ind\_i, ind\_j, els, numpix, numpix);

lambda = .0001;

t = (L + lambda .\* speye(size(L))) \ trans\_est(:) .\* lambda;

t = reshape(t, size(trans\_est));

**find\_SceneRadiance.m**

function final\_img = find\_SceneRadiance(TI, TA\_matrix, transmission\_map)

dims = size(TI);

final\_img = TI;

t0 = 0.1;

for m = 1 : dims(1)

for n = 1 : dims(2)

for k = 1 : 3

final\_img(m,n, k) = (((TI(m,n, k) - TA\_matrix(m, n, k))) / max(transmission\_map(m,n), t0)) + TA\_matrix(m, n, k);

end

end

end

**8.2 FUNCTIONS**

* imread

A = imread(FILENAME) reads a grayscale or color image from the file specified by the string FILENAME. If the file is not in the current directory, or in a directory on the MATLAB path, specify the full pathname. The return value A is an array containing the image data. If the file contains a grayscale image, A is an M-by-N array. If the file contains a truecolor image, A is an M-by-N-by-3 array.

* uigetfile

filename = uigetfile displays a modal dialog box that lists files in the current folder and enables you to select or enter the name of a file. If the filename is valid and if the file exists, uigetfile returns the filename as a string when you click **Open**. Otherwise uigetfile displays an appropriate error message, after which control returns to the dialog box. You can then enter another filename or click **Cancel**. If you click **Cancel** or close the dialog window, uigetfile returns 0. Successful execution of uigetfile does not open a file; it only returns the name of an existing file that you identify.

* imresize

B = imresize(A, scale) returns image B that is scale times the size of A. The input image A can be a grayscale, RGB, or binary image. If scale is between 0 and 1.0, B is smaller than A. If scale is greater than 1.0, B is larger than A.

* subplot

H = subplot(m,n,p), or subplot(mnp), breaks the Figure window into an m-by-n matrix of small axes, selects the p-th axes for the current plot, and returns the axes handle. The axes are counted along the top row of the Figure window, then the second row, etc.

* disp

disp(X) displays an array, without printing the array name. If X contains a text string, the string is displayed.

Another way to display an array on the screen is to type its name, but this prints a leading "X=," which is not always desirable.

* imshow

imshow(I) displays the image I which can be either RGB, gray or binary image

* imsave

imsave creates a Save Image tool in a separate figure that is associated with the image in the current figure, called the target image. The Save Image tool displays an interactive file chooser dialog box (shown below) in which you can specify a path and filename. When you click **Save**, the Save Image tool writes the target image to a file using the image file format you select in the Files of Type menu. imsave uses imwrite to save the image, using default options.

* padarray

B = padarray(A, padsize) pads array A with 0's (zeros). padsize is a vector of positive integers that specifies both the amount of padding to add and the dimension along which to add it. The value of an element in the vector specifies the amount of padding to add. The order of the element in the vector specifies the dimension along which to add the padding.

* sort

B = sort(A) sorts the elements along different dimensions of an array, and arranges those elements in ascending order.

* find

ind = find(X) locates all nonzero elements of array X, and returns the linear indices of those elements in vector ind. If X is a row vector, then ind is a row vector; otherwise, ind is a column vector. If X contains no nonzero elements or is an empty array, then ind is an empty array.

* eye

Y = eye(*n*) returns the *n*-by-*n* identity matrix.

Y = eye(*m*,*n*) or Y = eye([*m* *n*]) returns an *m*-by-*n* matrix with 1's on the diagonal and 0's elsewhere. The size inputs *m* and *n* should be nonnegative integers. Negative integers are treated as 0.

Y = eye(size(A)) returns an identity matrix the same size as A.

* meshgrid

[X,Y] = meshgrid(x,y) transforms the domain specified by vectors x and y into arrays X and Y, which can be used to evaluate functions of two variables and three-dimensional mesh/surface plots. The rows of the output array X are copies of the vector x; columns of the output array Y are copies of the vector y.

[X,Y] = meshgrid(x) is the same as [X,Y] = meshgrid(x,x).

* size

[M,N] = size(X) for matrix X, returns the number of rows and columns in X as separate output variables.

* strcmp

TF = strcmp('str1', 'str2') compares the strings str1 and str2 and returns logical 1 (true) if they are identical, and returns logical 0 (false) otherwise. str1 and str2 can be character arrays of any dimension, but strcmp does not return true unless the sizes of both arrays are equal, and the contents of the two arrays are the same.

* strfind

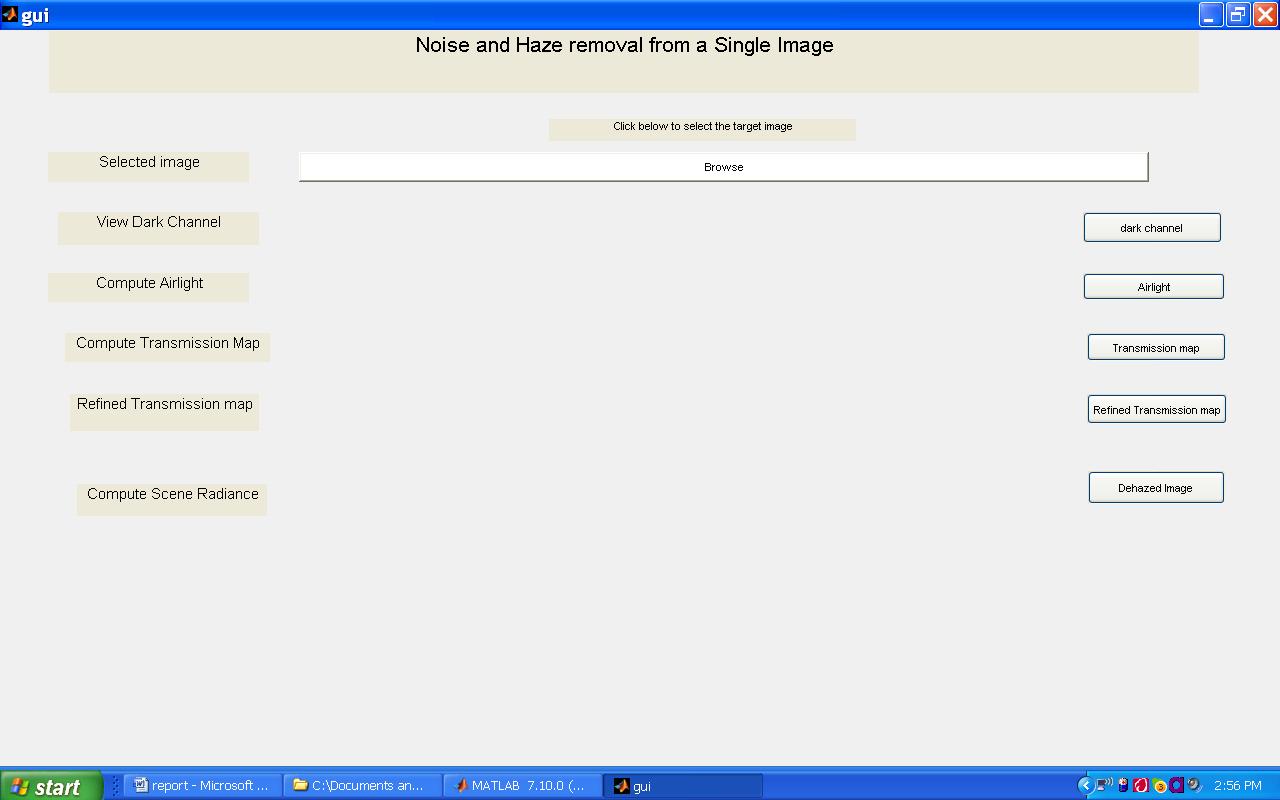
k = strfind(str, pattern) searches the string str for occurrences of a shorter string, pattern, and returns the starting index of each such occurrence in the double array k. If pattern is not found in str, or if pattern is longer than str, then strfind returns the empty array [].

**8.3 SNAPSHOTS**

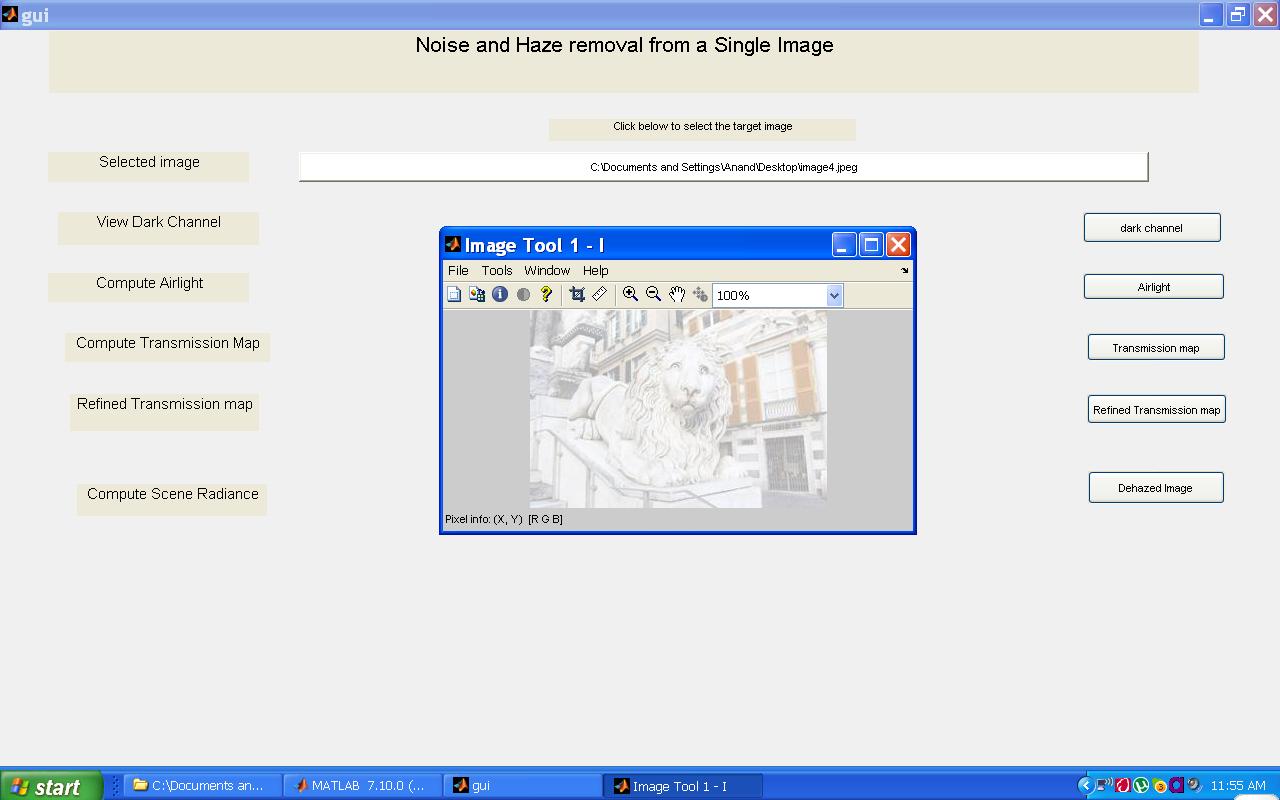
**SOME SAMPLE RESULTS**

**Sample result: 1**

* The snapshot below displays the graphical user interface designed to dehaze various hazy images.

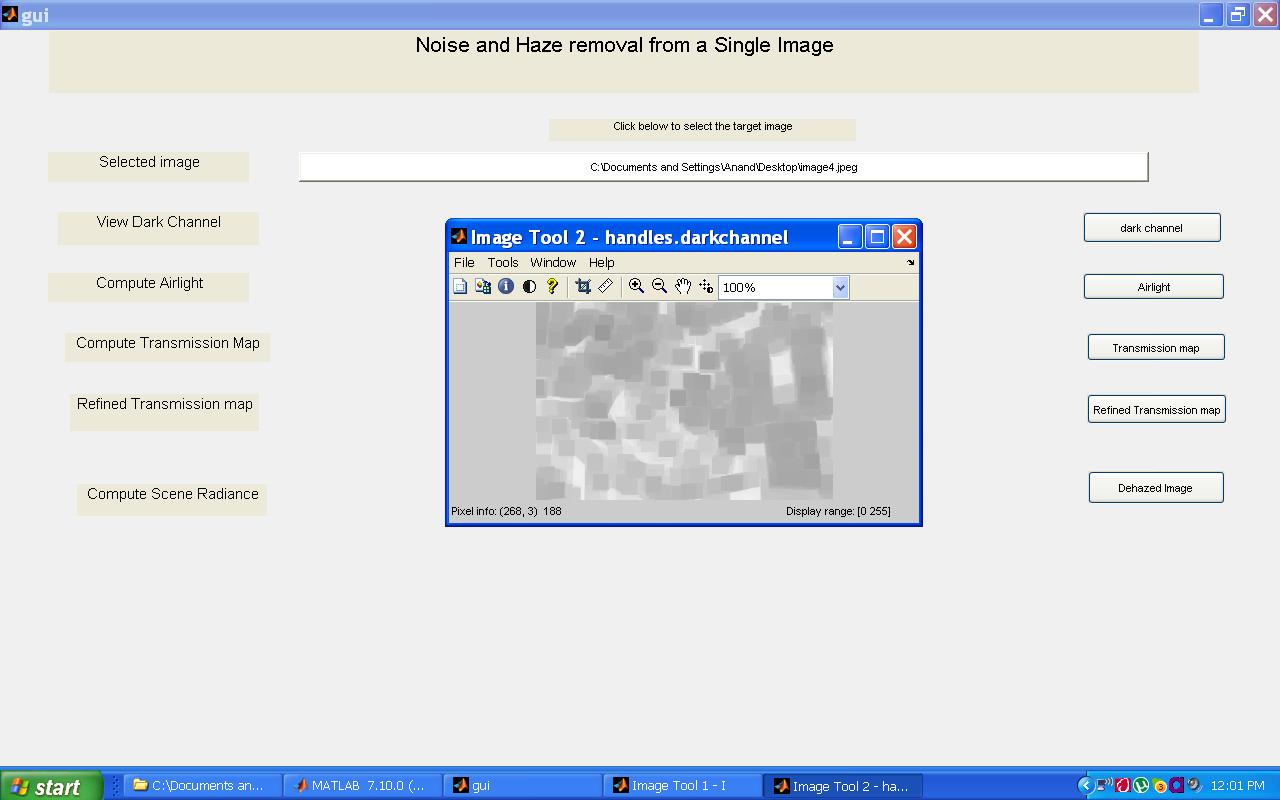


* The browse option is used to upload a hazy image from the system.



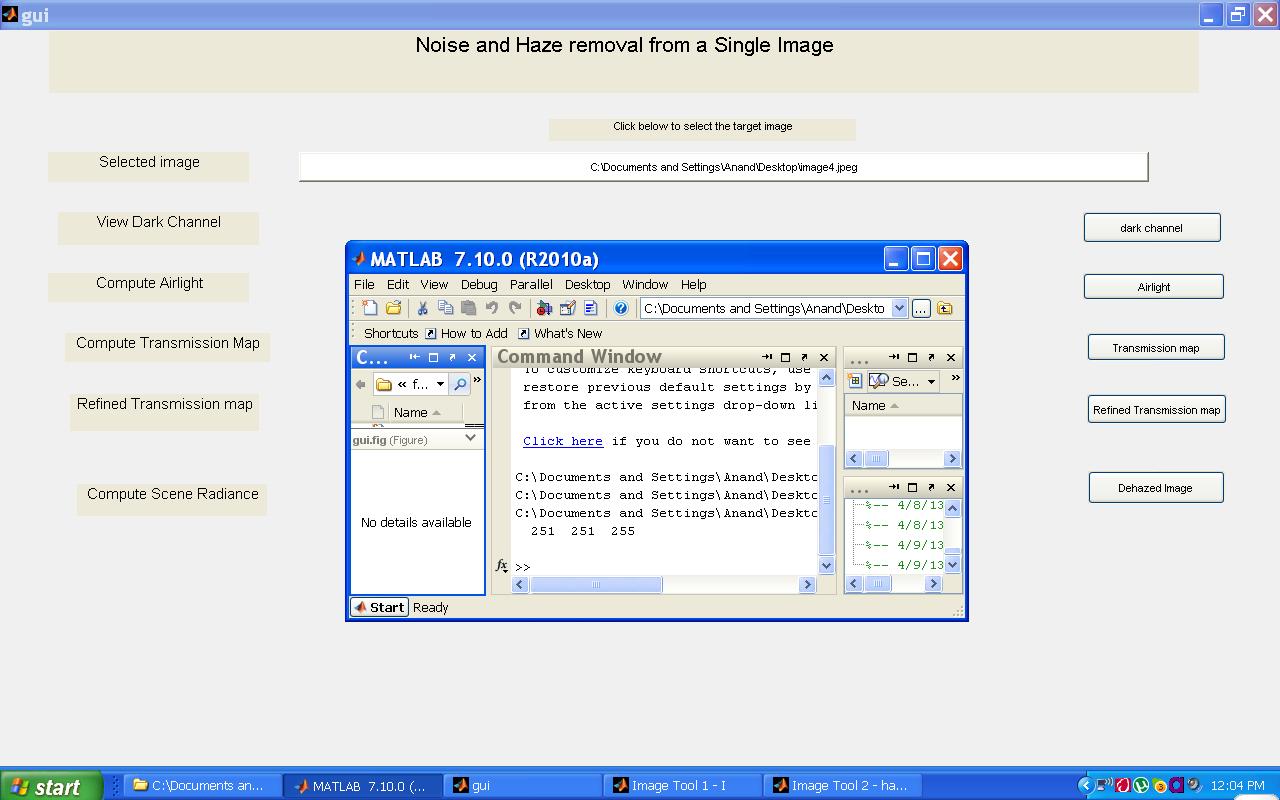
The uploaded hazy image is displayed in the graphical user interface.

* The snapshot below show the dark channel for the respective uploaded hazy image from the system.



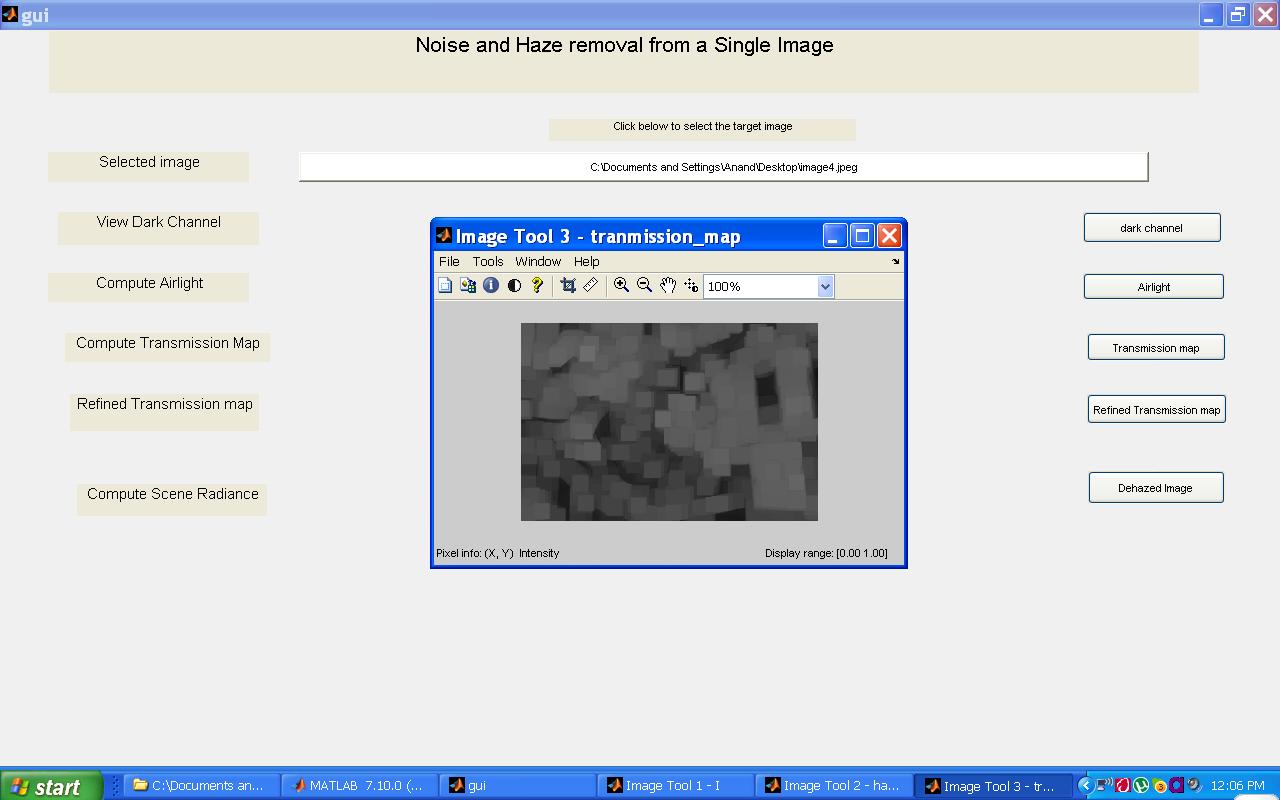
The dark channel of the hazy image is displayed in the graphical user interface.

* The snapshot below show the airlight for the respective uploaded hazy image from the system .

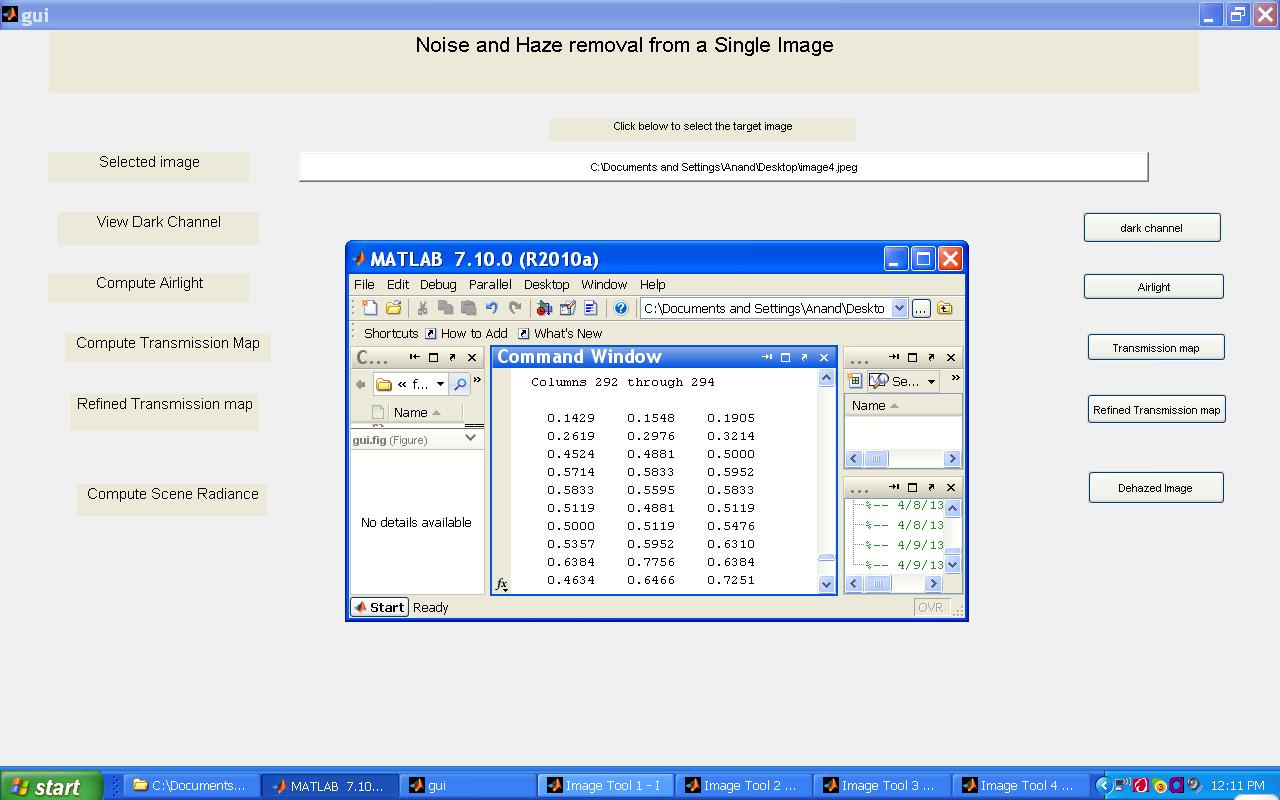


The result of airlight are values for colour channel red, green ,blue which is shown in the drop down box below airlight push button*.*

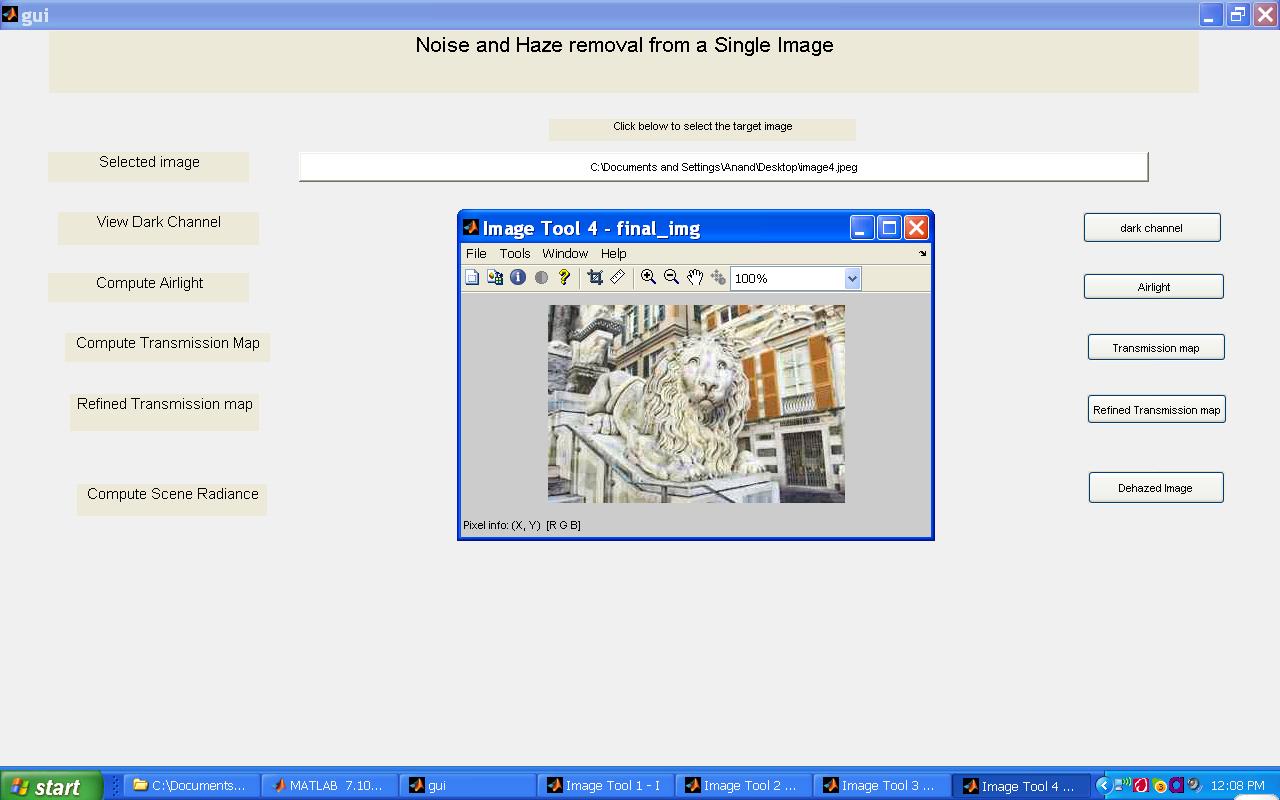
* The snapshot below shows the transmission map for the hazy image.



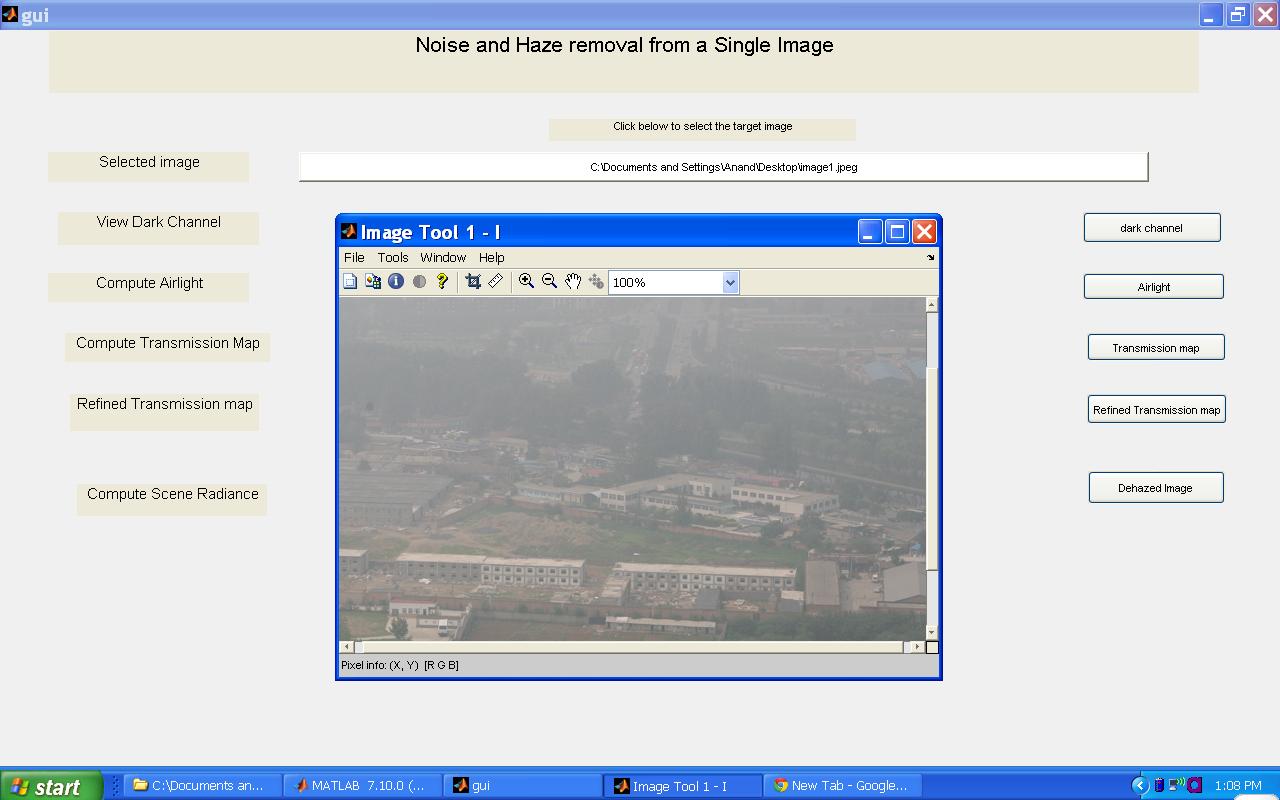
* The snapshot below shows the refined transmission map for the hazy image.

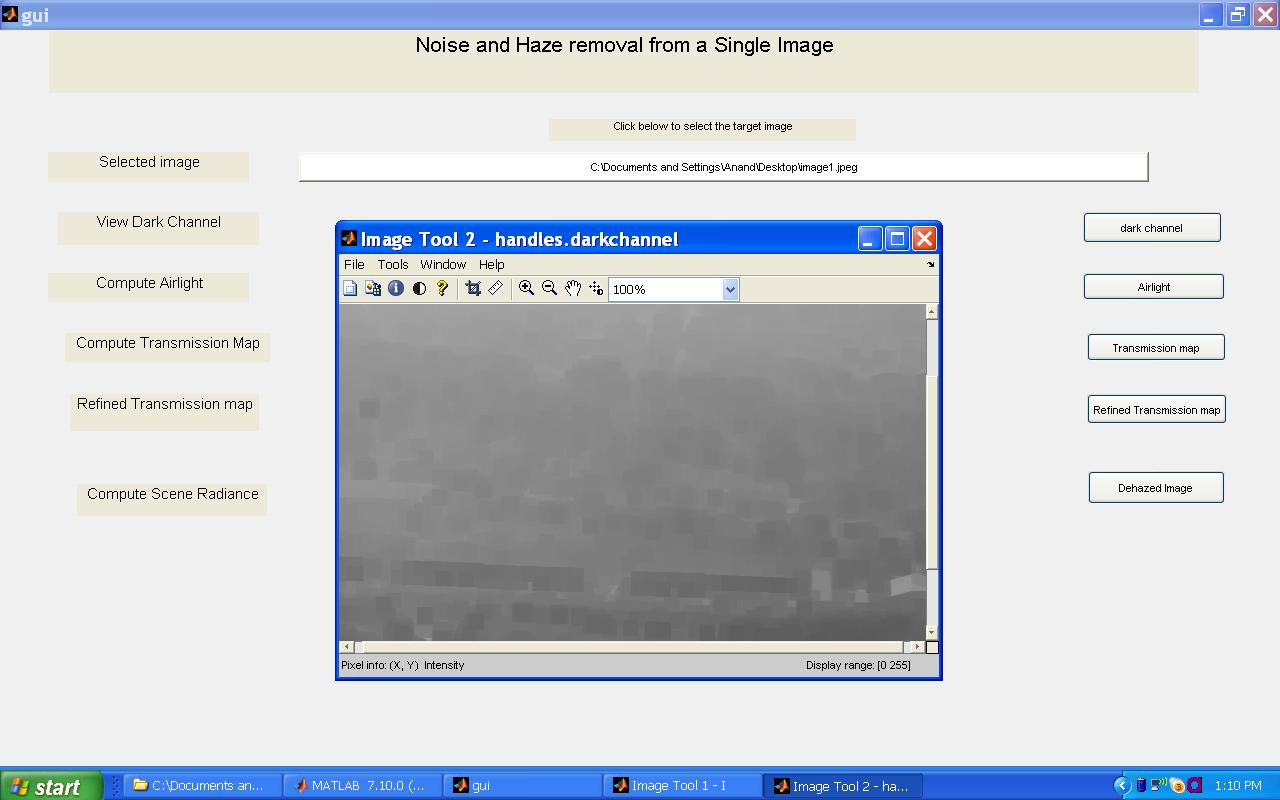


* The snapshot below shows the dehazed image.

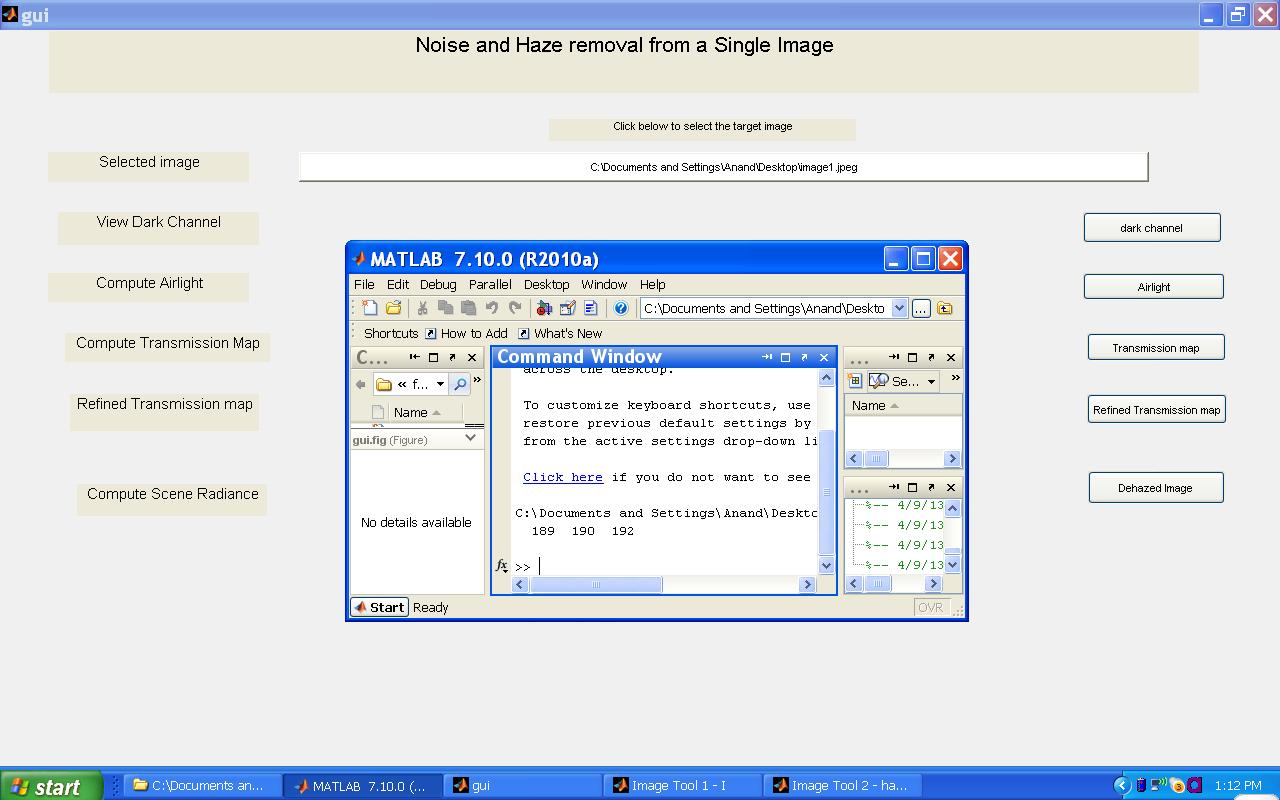


**Sample result: 2**

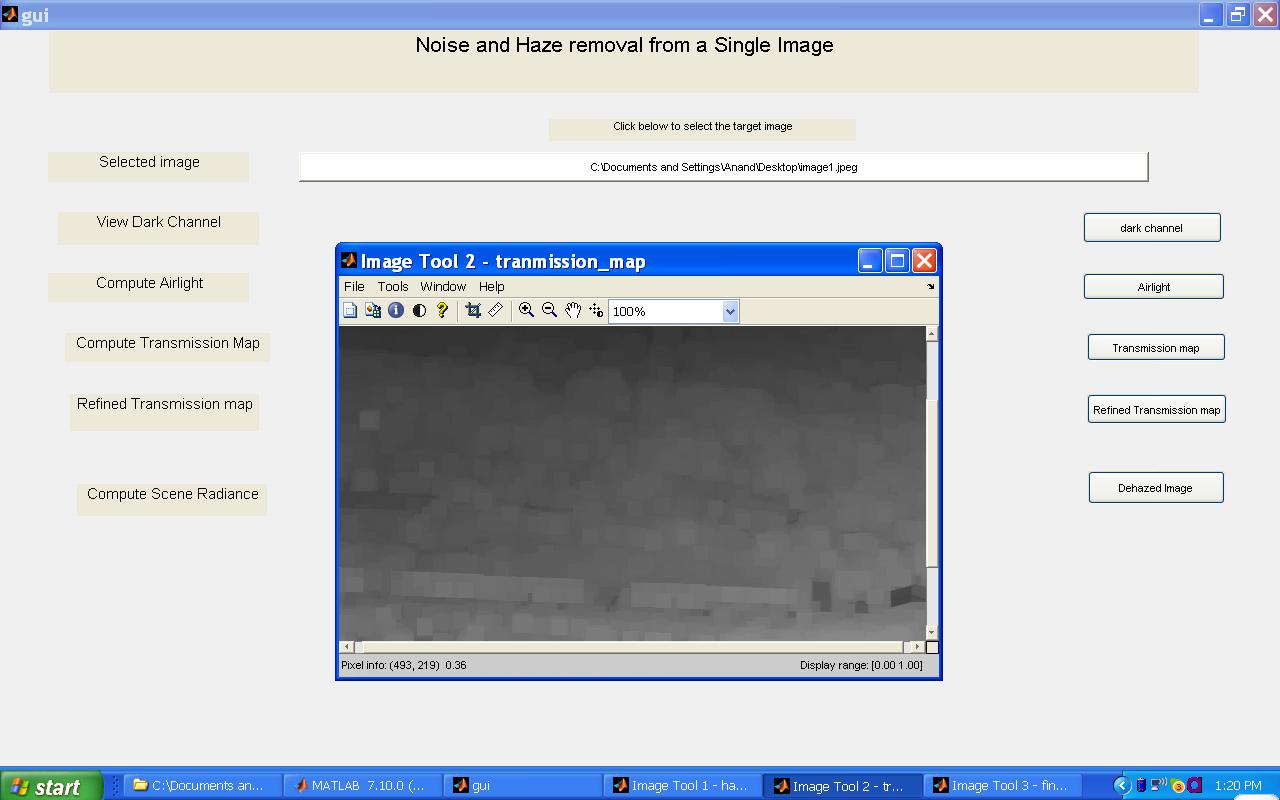
* The snapshot below displays the graphical user interface designed to dehaze various hazy images****
* The snapshot below show the dark channel for the respective uploaded hazy image from the system.



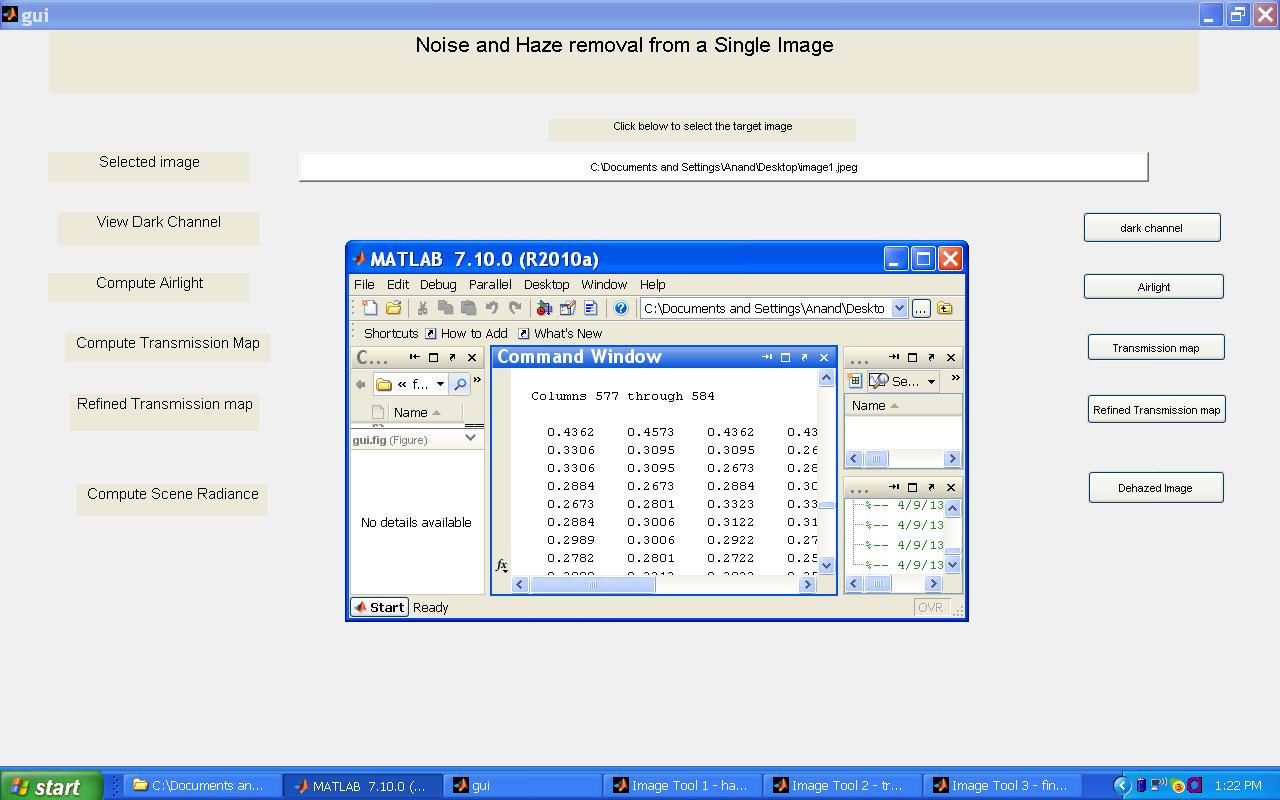
* The snapshot below show the airlight for the respective uploaded hazy image from the system .



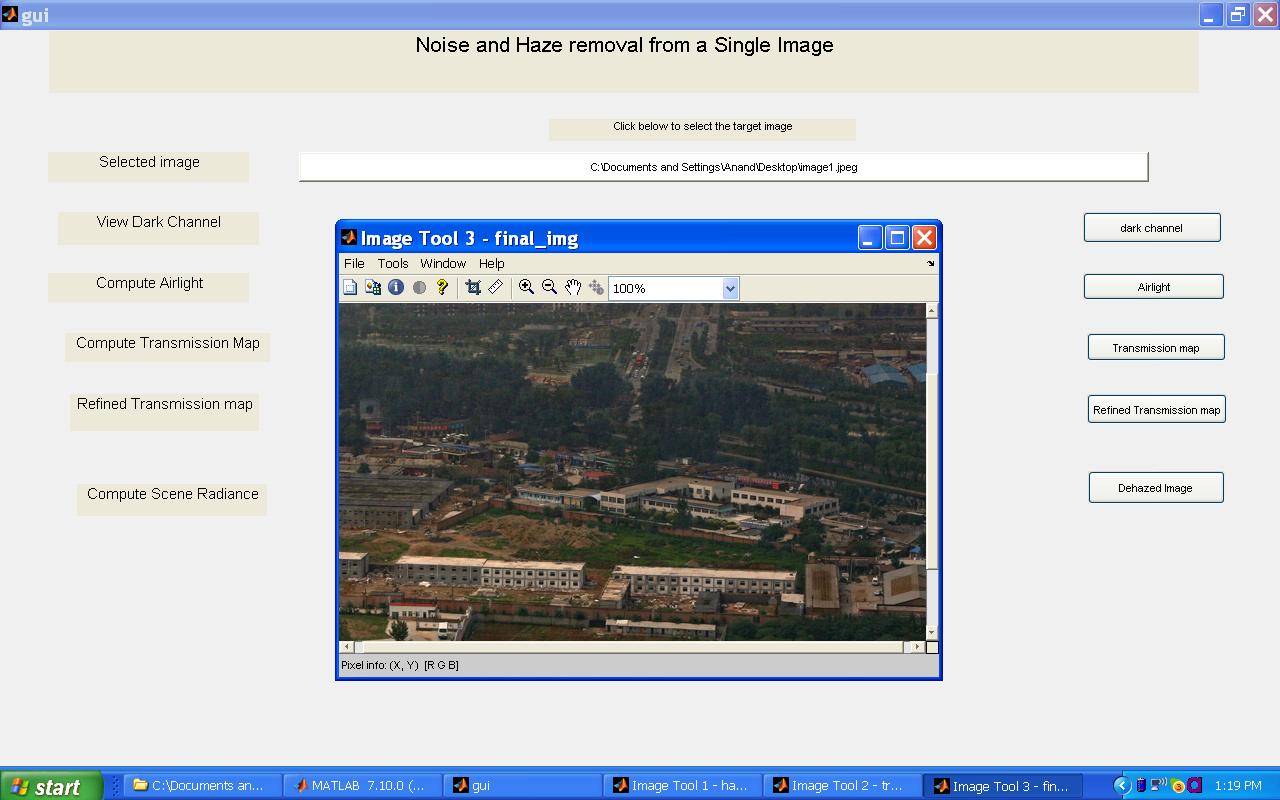
The result of airlight are values for colour channel red, green ,blue which is shown in the drop down box below airlight push button*.*

* The snapshot below shows the transmission map for the hazy image.

* The snapshot below shows the refined transmission map for the hazy image.

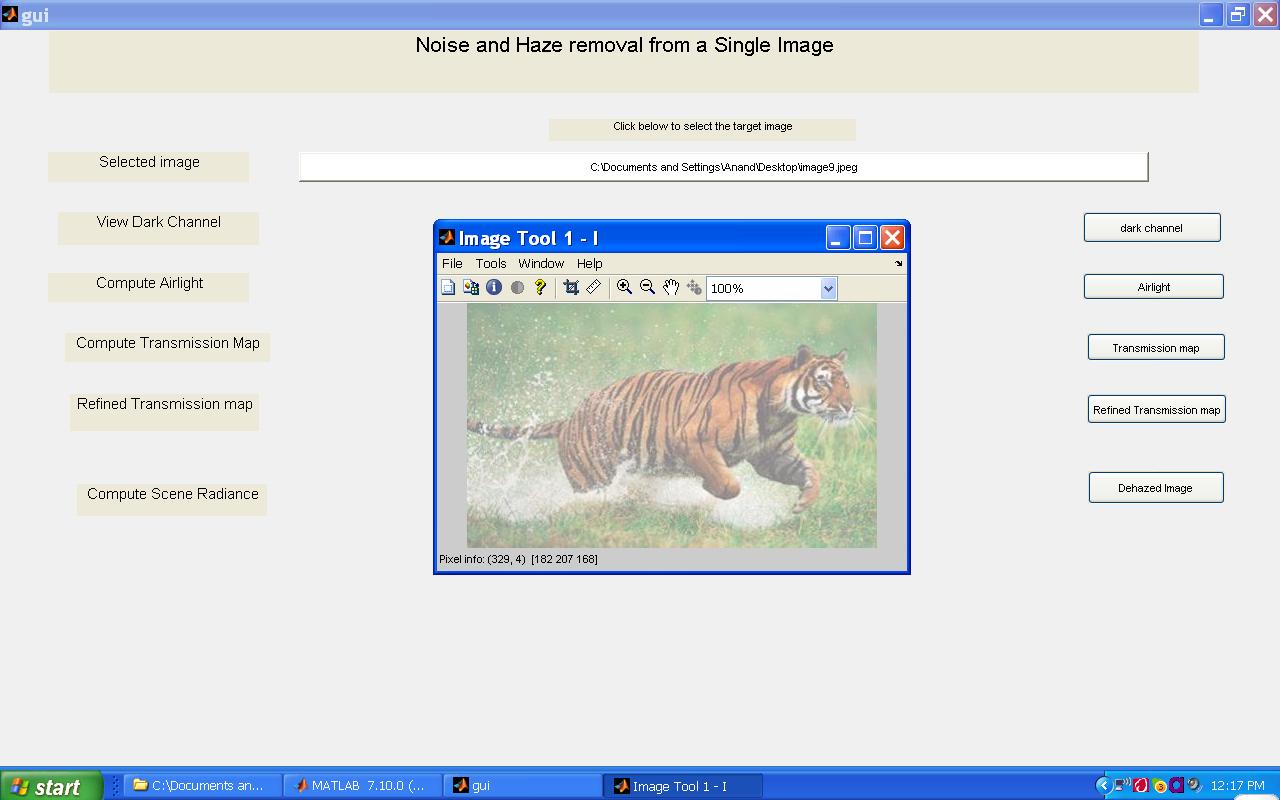


* The snapshot below shows the dehazed image.

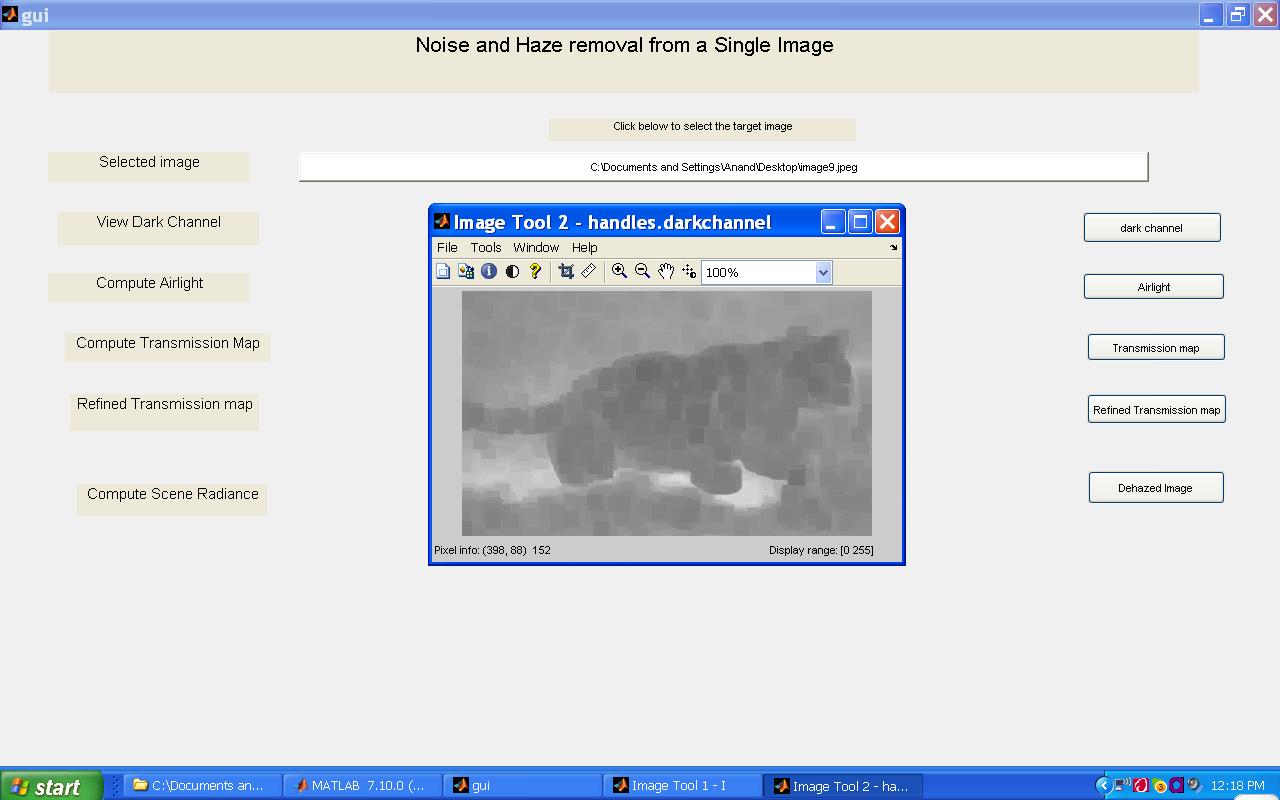


**Sample result: 3**

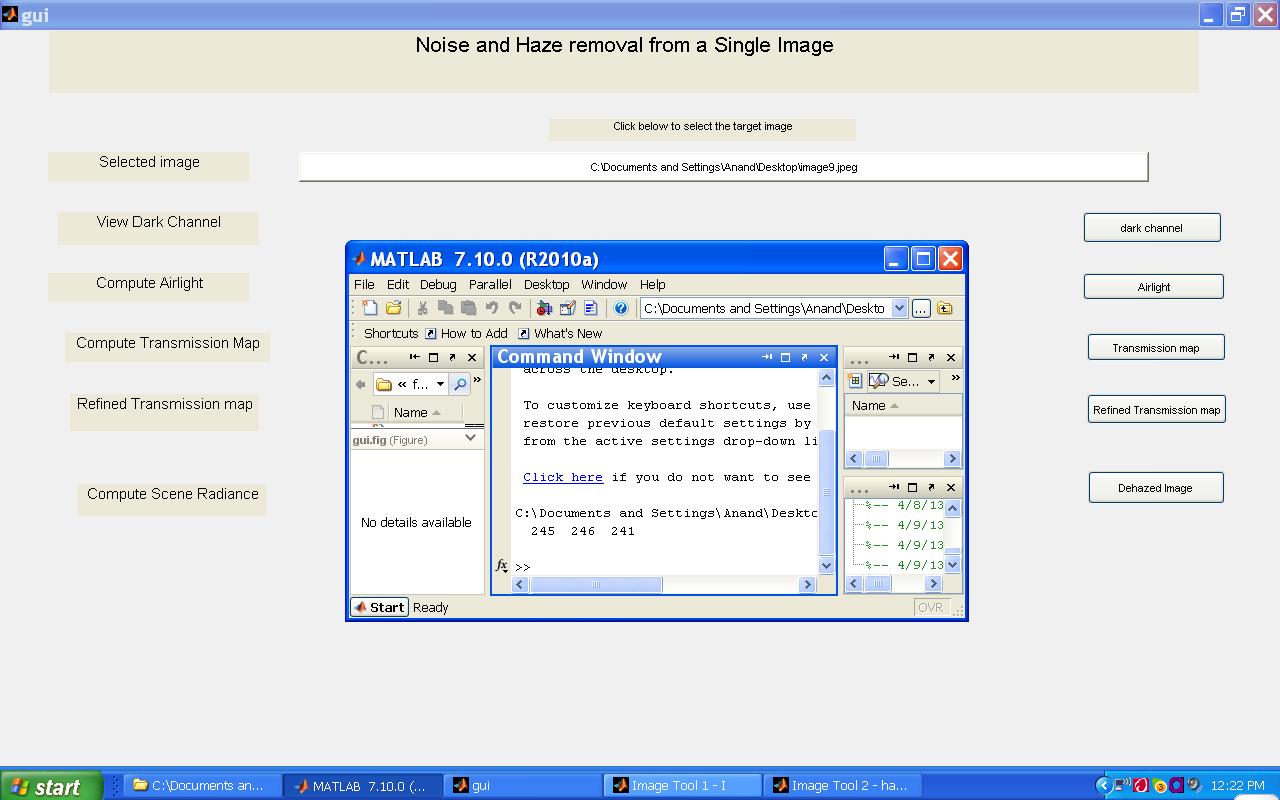
* The snapshot shows the graphical user interface for removing haze from a single image.



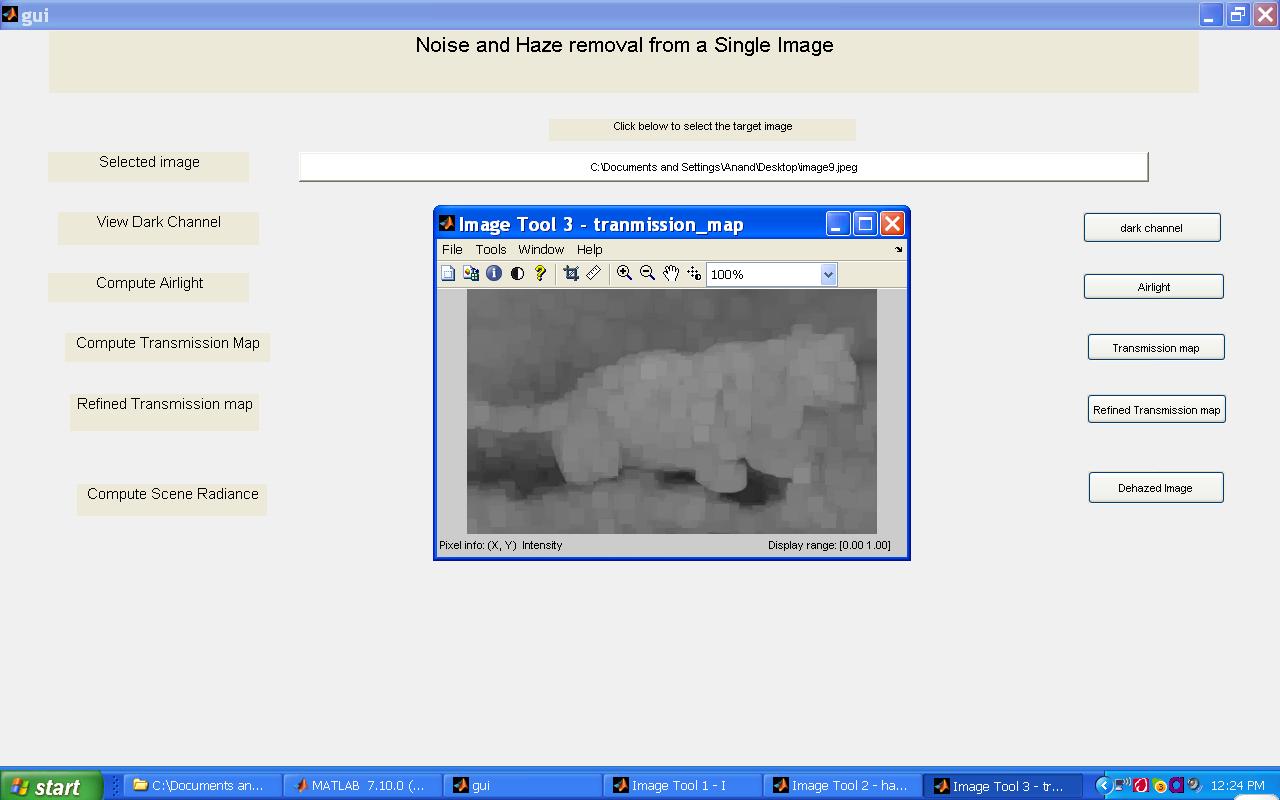
* The snapshot shows the dark channel for the uploaded hazy image.



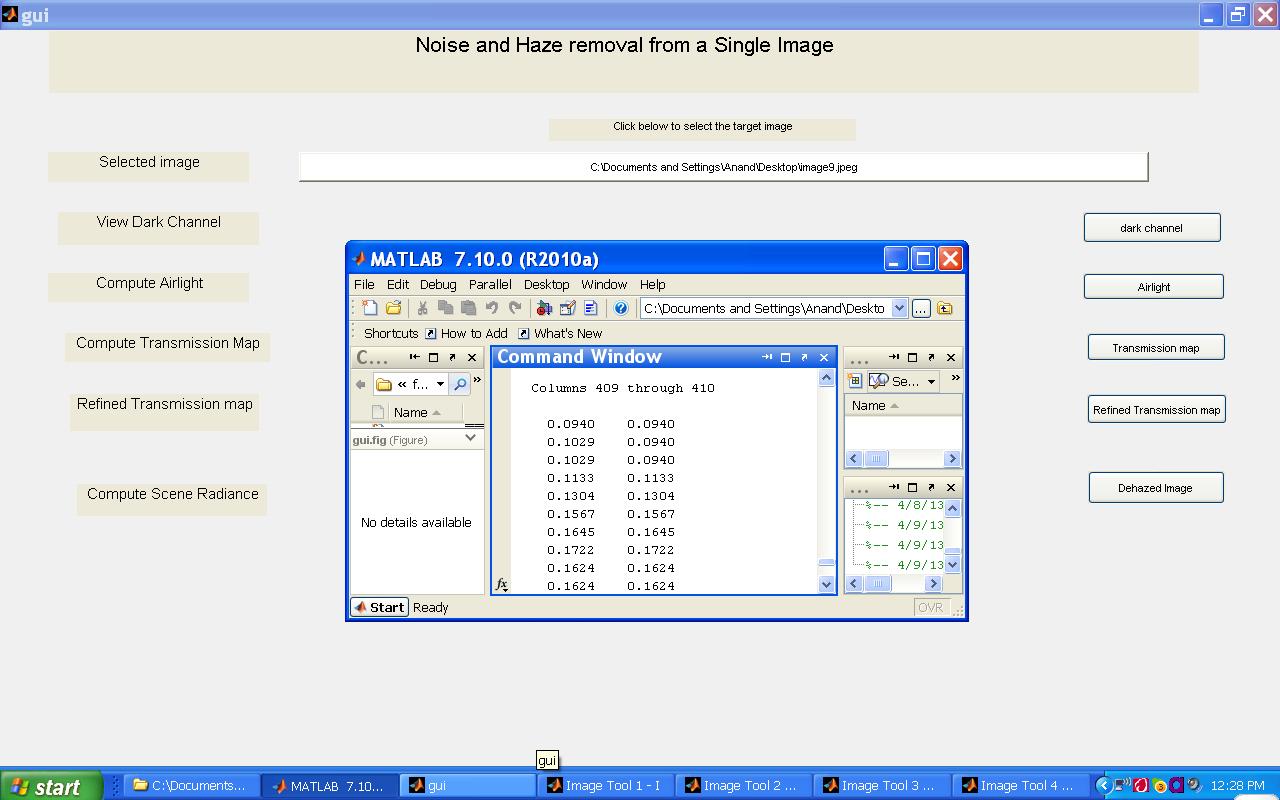
* The snapshot below show the airlight for the respective uploaded hazy image from the system .



* The snapshot below shows the transmission map for the hazy image.



* The snapshot below shows the refined transmission map for the hazy image.



* The snapshot shows the dehazed image.

