Probabilistic Time Lapse Video

Jorge Luis Reyes Ortiz

Master of Science

School of Informatics

University of Edinburgh

2008

Abstract

This research explores the construction of Time-Lapse Videos from cluttered consecutive images. It is focused on the analysis of the evolution of a construction project (The Informatics Forum). A capture process of 3 years produced a large image database which provides the basis for this work. Images were affected by different artifacts such as environmental and lighting changes, pedestrians, machinery and vehicles. Mechanisms have been developed to automatically render the images and reduce these adverse elements from the scenes to produce more 'truthful' videos which more accurately describe the building construction, thus contrasting with traditional techniques which only use captured raw images spaced at fixed intervals to produce the output video.

Reduction of noise, jitter removal and normalization of color levels are the methods employed in the preprocessing stage to clean the image database and provide a suitable starting point for the implementation of the time lapse video creation algorithms. Groups of images captured on the same day are processed to produce the frames of the final video which represent the background of the scene. Foreground elements and sudden weather transitions are removed. Two different methods to solve this are evaluated in this research. First, a multivariate median filter is employed as a deterministic approach to solve the 'day image' generation and processing. Then we employ a filter based on non-parametric kernel density estimation and a neural network classifier. Results show improvements on the time lapse videos. A visible reduction of the foreground elements and a higher stability in the image colors present a smoother version of the building construction.

Acknowledgements

I would like to thank my supervisor Bob Fisher for his unconditional guidance during the entire project. Our weekly meetings and email discussions really helped me clarify ideas and explore different alternatives to solve the problems. Special thanks to my family who have been supporting me at every stage of my personal and professional career despite being thousands of miles away.

I am also grateful to my two sponsors, Colfuturo and the Alban Programme, which gave me the opportunity to continue my studies for a pleasurable and challenging year in Edinburgh. Finally I want to thank my friends Tony who helped to proofread this work and Sandhya with whom I discussed ideas related to the project and spent many hours in the labs finishing assignments.

Supported by the Programme Alban, the European Union Programme of High Level Scholarships for Latin America, scholarship No. E07M403765CO.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Jorge Luis Reyes Ortiz*)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Time-Lapse Video.

In image and video processing, Time-lapse video (TLV) is a visual technique in which frames from an image sequence are captured at a lower rate than when they are played back. It is a useful tool to visualize events that change imperceptibly slowly. This technique has been widely used in different fields; Motion of celestial bodies, cloudscapes, growth of plants, and observations of natural phenomena have been its classical topics.

Nowadays, the application of time-lapse photography has been extended to several areas such as video surveillance, the evolution of construction projects, scientific studies of natural phenomena and visual effects in cinematography. Recent improvements in technology have made the creation of time lapse videos easier by introducing low cost digital video cameras, software applications for video editing and webcam remote access through the Internet.

The generation of time-lapse can be improved if different elements related to the scene and the observer are taken into account. For instance, when a person observes a rapid still image sequence (with rates above 15 frames/second), the human eye will perceive it as an integrated moving image, the flicker between images becomes undetectable and the brain has the illusion of smooth motion. These features are exploited in video technology. We can create the same effect with time-lapse images when they are played back at the frame rates stated before.

Unfortunately, if some of the changes that appear in the scene lack continuity or have periodic occurrence, then the results do not look realistic and give the effect of a motion scene that is difficult to comprehend. Examples of these scenes are time-lapse videos of crowded outdoor locations during a long period of time (e.g. weeks).

Short term events are pedestrians and passing vehicles, longer events are day and night, weather changes and stationary cars.

One of the big challenges of a Time-Lapse Video is to preserve the most important events occurring in a scene whilst undesired elements can be omitted. Many of the time-lapse videos are produced in controlled environments where factors such as the scene illumination, the position or motion of the camera are very stable. This helps to create realistic effects and diminish the amount of video processing. An example of these videos is the capture of images of flowers opening indoors without being affected by wind, sun, rain, etc. The resulting TLV can be a smooth depiction of the flowering transition in just a few seconds (refer to Figure 1.1 for an example). In contradistinction to the previous case, some other TLV are produced with data affected by different noise sources which have to be considered and removed during the video rendering. This research deals with these artifacts to produce improved time-lapse videos of the evolution of a construction project.

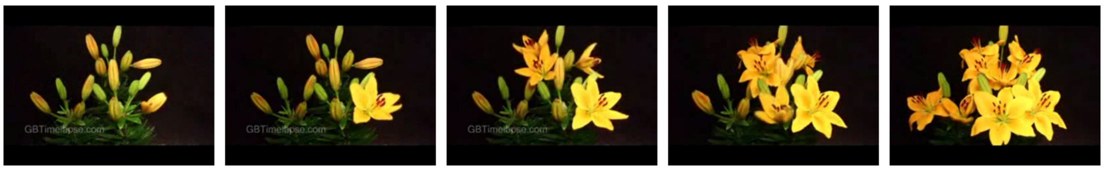


Figure 1.1: Example of a time-Lapse video. A bunch of lilies opening over a period of about two hours in just eight seconds. Video source [GBTimelapse, 2007].

1.2 Case Study: The Informatics Forum.

The University of Edinburgh has captured still images of its new Informatics Forum over the past three years at a frequency of 1 frame/minute. These images can be linked together to produce a time-lapse video of the building construction over the entire project execution. Images from the building were obtained from three different cameras. An example of the construction process at five different times is depicted in Figure 1.2.

A traditional technique to make a TLV would take a set of images from the database partitioned at fixed time intervals. The risk of doing this is the effect of random appearance of moving objects in the scene such as people and machinery, combined with the evident differences in the weather and illumination during the entire construction. This approach results in a visually complex and distracting time-lapse video. Figure

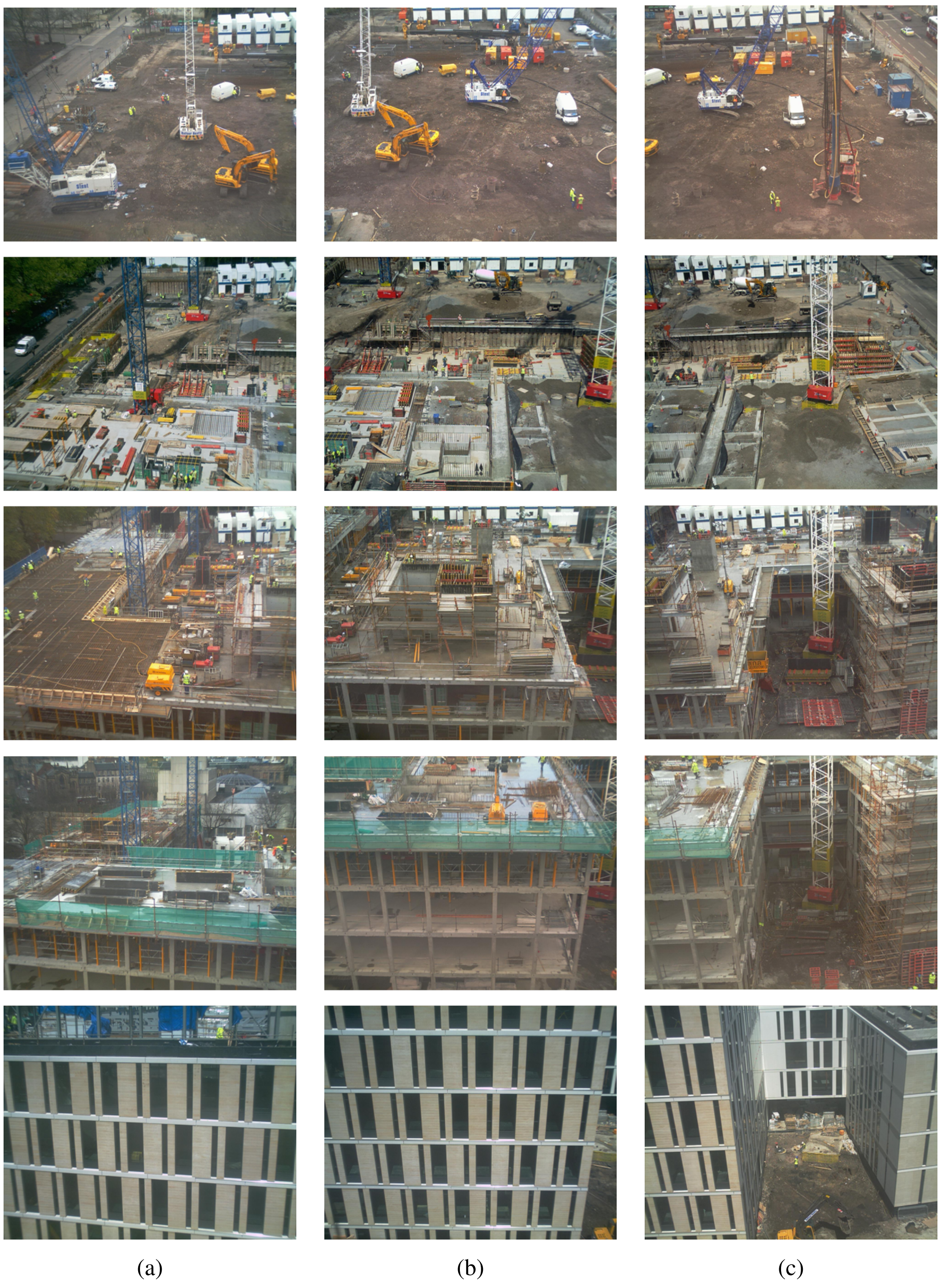


Figure 1.2: Image Sequence of the construction process taken from three cameras: (a) Doric. (b) Corinthian. (c) Ionic.

1.3 shows an example of three consecutive frames from a traditional time-lapse video. Note that several elements have been removed or added to the scene; additionally the light conditions have evidently changed from one day to another.

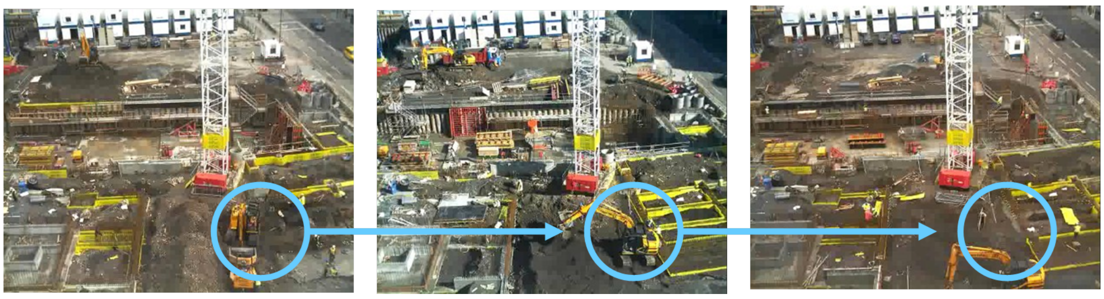


Figure 1.3: Three consecutive frames of a traditional time lapse video. The detail depicts a sample of variations between them.

Generally, short term events can appear as noise in the rendered video because its duration takes only very few frame rates. Hence, if the intention of the video is to create a smooth progression of events, then it is required to analyze the video data and remove the elements that are not required for a better video representation. Additionally, strong environmental changes such as snow, rain and fog which are already part of the image data can also be made less marked in the final video. All these modifications allow for a better understanding of the visual information for an observer.

1.3 Problem Description and Objectives.

This research investigates the construction of Time-Lapse Videos from cluttered consecutive images. We aim to produce novel mechanisms to automatically render the image data of the evolution of a construction project. This required developing deterministic and probabilistic approaches that make use of prior knowledge of the data to analyze the information from a time-lapse image database.

The end result is to create videos containing the most representative elements of the construction progress; the building will be partially isolated from the external variations such as weather conditions and moving objects. We have available data for one particular case which is the Informatics Forum and this will be the focal point of our research. Even though this approach can be applied to different fields, the utilization of specific knowledge on this case might not allow the generalization of the procedures here implemented for the creation of all types of TLV.

These are the main objectives in this research:

- Develop methods for producing time-lapse videos that improve the visual representation of the evolution of a construction project given a set of cluttered images from the Informatics Forum.
- Construct a technique to automatically create the most likely scene background of an entire day where temporary elements such as moving objects and lighting changes are removed.
- Investigate approaches to preprocess the raw image data and remove artifacts such as jitter, lighting and color changes that affect the image processing.
- Evaluate and compare the results of the implemented methods to make decisions about their effectiveness and applicability.

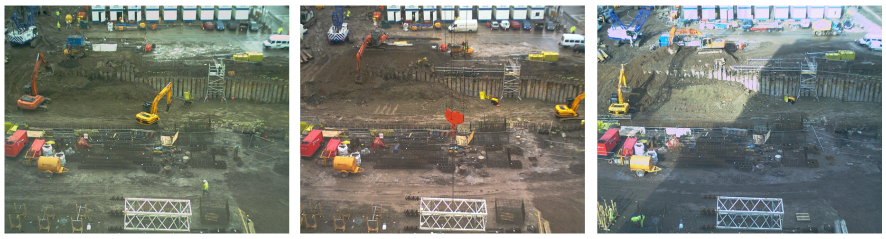
This research developed a series of techniques for time-lapse video generation. They have dealt with different elements that affect the scene such as Jitter, Color unbalance, unnecessary Foreground, weather changes, etc. These mechanisms require the processing of large amounts of image data to finally produce short frame sequences. Their evaluation is depicted in this document. An example of the steps to produce time-lapse video using the deterministic approach (multivariate median filter) is shown in Figure 1.4. It depicts how from the initial image dataset we obtained the last sequence of frames for video rendering. Note that some objects present in the images of the first row are removed. In addition, the image colors of consecutive frames are normalized after preprocessing.

To provide a better understanding of the techniques employed in this research, a multimedia library containing the time-lapse videos is available online at the following url [Reyes and Fisher, 2008]:

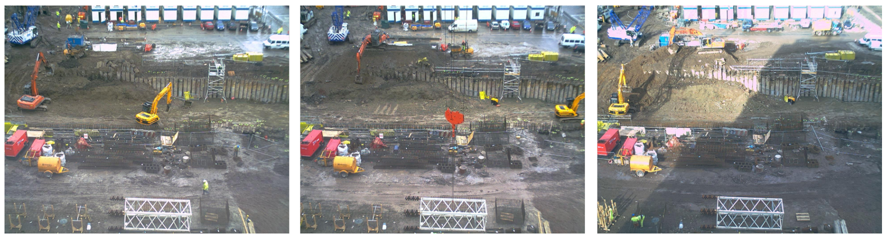
`http://groups.inf.ed.ac.uk/vision/BUILDING/REYES_VIDEOS/.`

1.4 Outline of the Research.

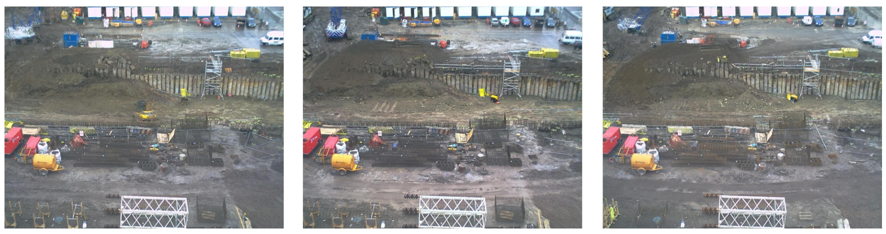
Previous work related to the production of time-lapse videos has been analyzed to have a clear understanding of the state of the art in this area. It has been focused on the idea of the persistence of intrinsic elements on the images and the possibility of extracting representative video segments or features from the data. This analysis is delineated in



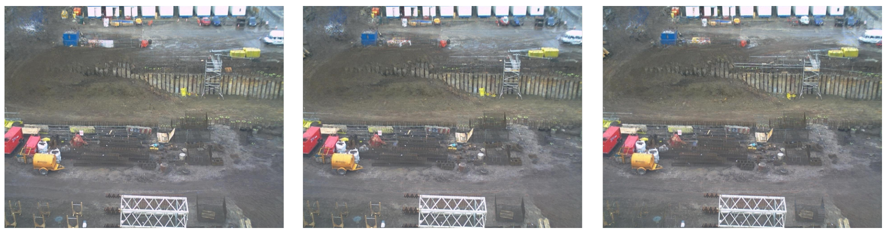
(a)



(b)



(c)



(d)

Figure 1.4: Multivariate median time-lapse video example (3 frames). (a) Raw images. (b) Preprocessed images (Noise reduction, Jitter Correction and Color balance). (c) Multivariate median filter applied on images from the same day. (d) Multivariate median of temporally neighboring images.

Chapter 2 together with a brief depiction of the data acquisition process. The latter reveals how the images from the Informatics Forum were captured (hardware) and stored (software and database).

Once the images were available for our use, we realized that these had several unfavorable elements that could be improved to finally obtain better results during the image processing stages. For instance, a clearly visible vibration between consecutive frames (jitter) and an alteration of the true color values of some images due to defects in the capture devices. Mechanisms such as noise reduction, jitter removal and color normalization that reduce the artifacts on the raw images before the TLV construction are described in detail in the image preprocessing chapter (3).

At that point we assumed that the images were clean and ready for the processing. One fundamental idea of our video construction process is that images that represent each day (day image generation) are obtained initially using the available data and then processed to create the final videos (day image processing). This concept is applied to three different approaches for TLV production. We start with the implementation of the traditional method which was produced for comparison purposes. Then, a deterministic method based on median estimations in a three-dimensional space of the video images is implemented. And finally we experimented with a probabilistic approach which applies a kernel density estimator in day image generation and two classifiers (Naive Bayes and neural networks) for posterior day image processing. The explanation and evaluation of these methods is presented in Chapter 4. A general overview of the complete reconstruction process is depicted in Figure 1.5. This shows how the raw data go through the different phases till the final video is produced.

As a final point, we conclude showing the contribution of this work and analyze the difficulties found during the experimentation. Additionally, some improvements of the implemented techniques are proposed as future work. This can be found in Chapter 5.

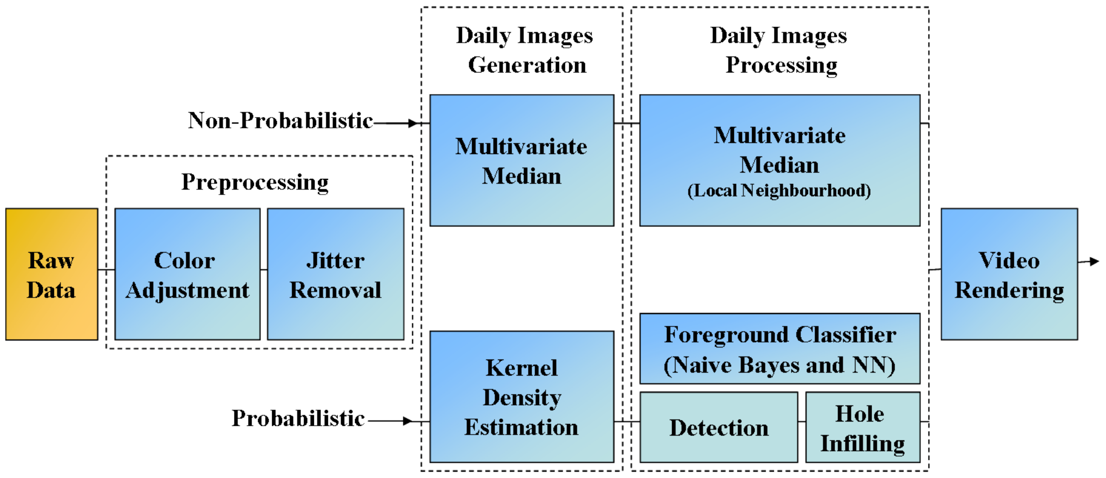


Figure 1.5: Reconstruction stages for the generation of time-lapse videos.

Chapter 2

Background

The computer vision community has worked the problem of playing back video at faster frame rates than their original capture speeds. Some areas have been focused in the interpretation of this data to extract relevant information. Time-varying imagery deals with the problem of dynamic scene analysis and the processing of sequences of images to retrieve information that can not be obtained independently from each frame. Video summarization looks for short elements in video data to best represent the entire data set. For instance, a film can be condensed in a very short time but a non-linear sampling allows grabbing groups of sequential frames instead of having rapid changes. In this section a description of the current research in time-lapse video analysis is depicted. Additionally, a description of the image acquisition process is presented. This includes details in relation to the hardware utilized during the registration and a depiction of the image database that has been employed to produce time-lapse videos.

2.1 Time-lapse video analysis

The concept of intrinsic image model was introduced by [Tappen, 2005] where inherent characteristics of the images such as illumination and surface perception can be extracted from visual data. The scenes can be expressed for instance in terms of the combination of these characteristics. This concept was extended by [Weiss, 2001]. They use time-lapse video to estimate a single reflectance image with multiple illuminations in a static scene. Some other researchers have applied these intrinsic image ideas in different fields.

The reflectance and illumination images are extracted from surveillance video in [Matsushita et al., 2003]. Similarly to our study case, this work deals with changes in illumination conditions due to weather and time of day. These artifacts make the processing task more difficult when video surveillance systems are employed in outdoor locations. They use techniques for image normalization to remove shadows and color changes from the images. Differently to our approach they work with a static background and image data at higher frame rates, hence this allows to create a model of the scene and subsequently apply object tracking algorithms for robust video surveillance.

In [Koppal and Narasimhan, 2006], a technique is proposed for scene analysis to cluster an image into regions with similar light changes according to the surface normals of the scene points. This is done after using smoothly moving light sources in the scene with unplanned patterns and analyzing the derivates of the reflectance distributions on the images. Pixels with similar color intensity variation are considered as part of surfaces with similar normals. Similarly, in [Sunkavalli et al., 2007] Sunkavally developed a method called Factored Time-Lapse Video to obtain time lapse video from outdoor and clear sky scenes. He obtained plausible results after separating the light sources and shadows from the video data. He estimated the shadows for all the pixels in the image using time-varying intensity profiles. The whole sequence (spatio-temporal volume) was factored in a skylight image, a sunlight image, two basic curves that describe the temporal characteristics of skylight and sunlight and a shift image that represent the offset of each pixel respect to the basic curves. Figure 2.1 shows an example of image decomposition in their factors using this method. This last two researches differ of our model because even though the sun is the main light source falling on the Informatics Forum images and it always has a known motion pattern, It is influenced by the clouds or weather. Hence a real motion model cannot be always applied to this particular application; also, as it will be described in Section 2.4, the number of frames that we have available for the processing is not the adequate amount for this purpose. Finally, the background is constantly changing at unpredictable locations.

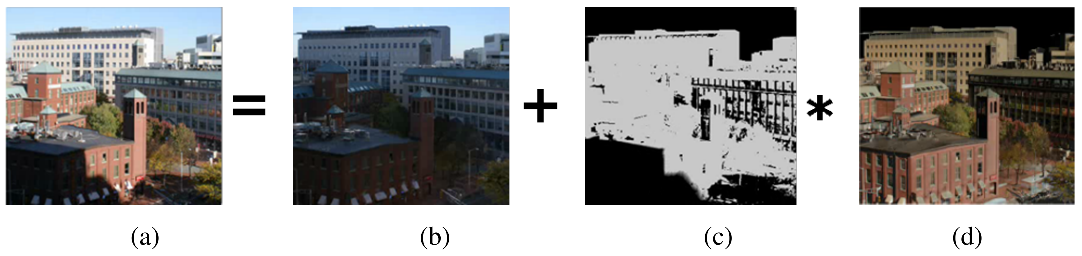


Figure 2.1: Factored time-lapse video [Sunkavalli et al., 2007]. The spatio-temporal volume (a) is separated into: (b) Sky component and (c) Sun component, which is modulated by. (d) The shadow volume.

In [Bennett and McMillan, 2007] Bennett presents a method for generating time lapse videos from video-rate footage. They use a non-uniform sampling method based on dynamic programming to process the source data into the time lapse video. They are able to retain, remove and resample events based on the user requirements such as the visual objectives and specified video duration. Depending on the desired effect and type of output, they use different metrics for the optimization. The non-linear sampling is combined with a virtual shutter to create longer time exposures by employing several images. In our method for generating Time-lapse videos, only uniform image sampling is considered for the rendering which means that consecutive frames of the final set are spaced by the same time interval. A non-uniform sampling can be considered if it was desired to visualize parts of the video at different reproduction speeds. Our raw data is not video footage but for the generation of each frame of the final video, we have to analyze a subset of images. Something similar takes place in this approach.

2.1.1 Video Summarization

In [Smith and Kanade, 1997] Smith worked with multimedia video summarization. Short representative video segments retain the essential information from the input videos. With the increasing size of collections of video databases, tools are required to evaluate in short times the contents of these libraries. They propose a method to produce videos which are synopses of the original. This is done by analyzing the video and audio information with image and language understanding techniques such as motion analysis, scene segmentation to detect changes, object detection (faces, text, etc.).

Another approach related with video summarization is the generation of video synopsis. This is a tool developed by [Rav-Acha et al., 2006] used for browsing and indexing long time-lapse videos. It provides a short representation of the events that occur in a long period of time by showing activities that occurred at different times simultaneously. The final output provides a compact video representation and preserves the essential activities of the original video. Foreground, which is detected by a measure of activity on the images, is isolated and immersed in the same spatio temporal volume with other foreground elements that happened in different instances in contradistinction to the conventional video abstraction. An example of this method is the summarization of the event events occurring in an airport. Refer to Figure 2.2.

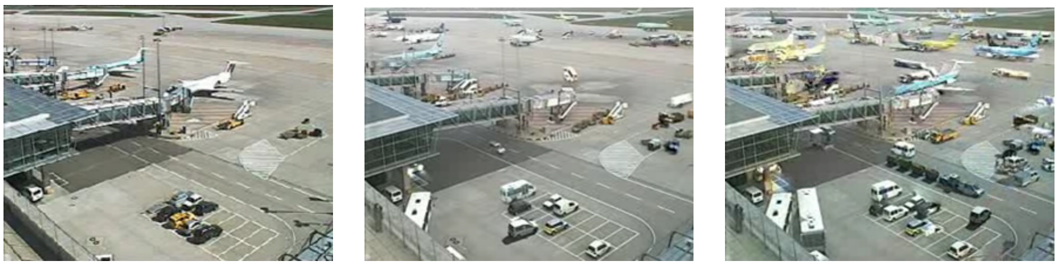


Figure 2.2: Video synopsis and indexing. Airport surveillance video. (Taken from [Yael Pritch and Peleg, 2008]) (a) Typical image. (b) Image from Synopsis video. (c) Video Synopsis at a smaller duration (more collisions between elements).

2.1.2 TLV in Cellular Biology

Cellular biology is another area where this technique has been applied. Time-lapse microscopy imaging provides an important method to measure the cycle progression and morphology of cells. When this analysis is required in large populations of cells, it is unreasonable to use a manual analysis of the data. In [Yang et al., 2006] an automated mechanism to study large volumes of time-lapse microscopy images is proposed. This is based on Mean-Shift [Comaniciu and Meer, 2002a] and Kalman Filter methods for a robust cell movement and division tracking. These measurements are important in understanding for instance drug effects on cancer cells.

Additionally, in [Althoff et al., 2006] an in vitro cell migration analysis is described. This was focused on the creation of a mechanism to differentiate neural stem/progenitor cells. A Hidden Markov Model is a powerful tool to describe stochastic non stationary processes such as cell migration.

2.1.3 TLV applied in Construction

In [Ferguson., 2007], a classical video recovery algorithm was employed by Ferguson to generate a time-lapse video of the building. It was based on a temporal median filtering method. Only video data from a fixed time day interval was chosen for the analysis. The results obtained with this approach showed the appearance of a phenomenon called ghosting where shadows of the foreground were immersed in the rendered video. Additionally, the images were blurry and the light conditions kept changing. This previous work will be used as reference for the evaluation of this project. A detailed explanation of this approach can be found in Section 4.2.1.

Time-lapse images can be utilized in the detection of the status of a construction process. In [Abeid and Arditi, 2002] time-lapse imagery is linked with dynamic scheduling of construction operations. They use a time-lapse technique to visualize the construction performance in a short period and also make daily registration of the progress from a user interface. Then the events and activities are associated with the image data. The disadvantage of this approach is that requires human supervision. On the other hand in [Lukins et al., 2007] a fully automated method for measuring the progress of a construction is proposed and depicted with a simplified test case. Based on image processing techniques and advanced reconstruction tools, they compare a virtual 3D model of a building with the current project status to determine the feedback of the progress. This research can be applied in project management to enable better control and efficiency in the project execution.

2.2 Background modeling

Other work has been focused on the background and foreground analysis. This consists in the separation of the visual information in two basic elements. The moving objects in a scene (foreground) can be detected by comparing each frame with a model of the scene background. Several elements that have to be considered to create this model; they include illumination changes and non-stationary background such as moving trees branches or waves in the sea, etc. Another important alteration that can take place in the background is the addition of new elements that modify the geometry in the scene (such as new structures in a building). These changes are initially foreground but then the background model has to be capable of incorporate them. This is called dynamic background generation.

A conventional method to estimate background is pixel intensity in terms of a probability density function (pdf); this is usually modeled as a Gaussian distribution. The background probability for a new pixel in new observations is estimated based on this distribution and a threshold. This model can be adapted to smooth changes in the environment. This basic method has been used in [Wren et al., 1997].

Some scenes cannot be modeled with a simple Gaussian distribution. For instance, pixels located near tree branches in an image cannot be defined as a single Gaussian because they can change from the light blue sky to the green leaf color. Hence for this a mixture of distributions (usually Gaussian) can be used to model each pixel. In [Grimson et al., 1998] a mixture of Gaussians distributions is used to model the color pixel variations in an outdoor scene. It is relatively complex to determine the exact number of Gaussians required for each particular pixel. Many researchers use a predefined number as the maximum number of Gaussians to describe each particular background pixel. These distributions are weighted according to their appearance. In [Lee, 2005] three Gaussians were used to model the background.

Sudden illumination changes were not considered in the previous approaches. For instance, in [Taycher et al., 2005] a HMM model has been used to model states of the environment in an outdoor scene. These states represented light levels such as cloudy and sunny skies.

In [Elgammal et al., 2002] Elgammal reconstructs a robust statistical model of the background based on a non-parametric kernel density estimation for outdoor visual surveillance. This does not assume a probability density function but it is estimated in terms of the most recent observations from the data. They model complex natural scenes including changing light conditions, shadows and small movements in the background. A similar method is implemented in [6]. Additionally in [Mittal and Paragios, 2004] Mittal uses another non-parametric approach to model scenes with dynamic behavior in the background. Their model measures the optical flow in the images and it uses this as a feature of the background. For instance, they are able to detect foreground objects in a beach where the waves constantly appear (as part of the background). They compare their technique with several methods such as Gaussian mixture and basic non-parametric approaches. Figure 2.3d shows an example of this method in a beach scene and is compared with a Gaussian mixture and a non-parametric model, note that the waves do not appear in the motion-based model (Figure 2.3).

In our approach is required to remove foreground elements and replace them with the most likely representation of the background model. In contrast to previous models, we deal with a highly dynamic background and the estimation of a truth model with a small amount of data has not a straightforward solution. Nevertheless the methods here implemented depict a plausible representation of the scene background.

The process of foreground removal implies the emergence of empty zones in the images that have to be filled with data from the neighboring regions in space and time. This can be done for instance by placing the data of the previous day in these zones or utilizing a set of nearby images. Both approaches are taken in consideration in this project. Another possible solution to the occluded background infilling is the use of a non-parametric method for texture synthesis similar to the one proposed in [Efros and Leung, 1999]. This process grows an image from an initial seed and preserves its local

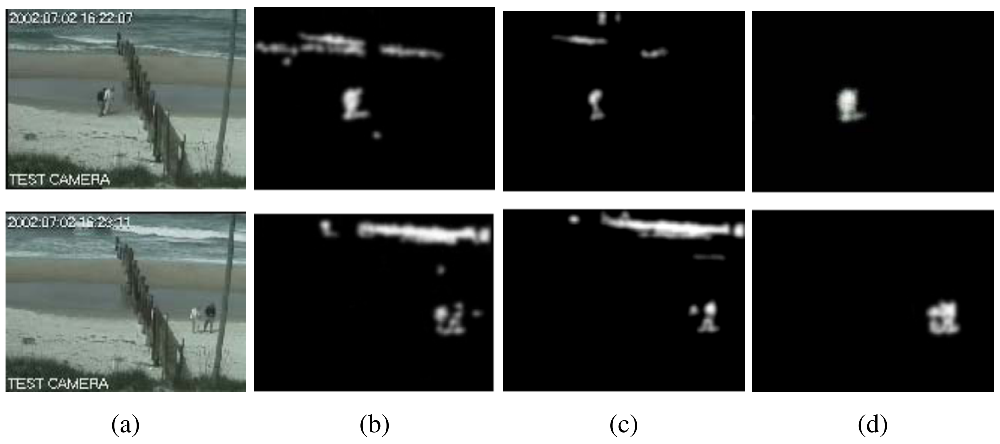


Figure 2.3: Motion-based background example and comparison with two techniques (source [Mittal and Paragios, 2004]): (a) Original images. (b) mixture of Gaussians model. (c) non-parametric model. (d) Motion based background model).

structure. The seed in this case would be the data surrounding the foreground, then this data is modeled as a generalized markov chain in two dimensions to determine its probability distribution and predict samples of occluded regions.

2.3 Hardware

Corinthian, Doric and Ionic are the names of the network cameras that have been installed in Appleton tower (the building next to the Informatics Forum) to register the project evolution. They have been located deliberately to capture most of the details of the construction. The cameras utilized in the project have a similar appearance to the one shown in Figure 2.4a. Their localization in the building is depicted in Figure 2.4b.

These devices have an Ethernet network interface which makes them flexible for installation. Additionally, they have a resolution of 1.2 Megapixels which is suitable for high quality video and image capture. In our case only images were stored but from an internet address [Blunsden, 2008], live feeds were available during the construction. The most relevant specifications of the cameras are shown in Table 2.1 and a complete description of their characteristics can be found in [axi, 2006].

We had some difficulties with the image acquisition process that affected the quality of the images. These are described as follows:

- Doric and Ionic cameras have been moved manually in different occasions. This was done essentially to provide a wider view of the construction once it was

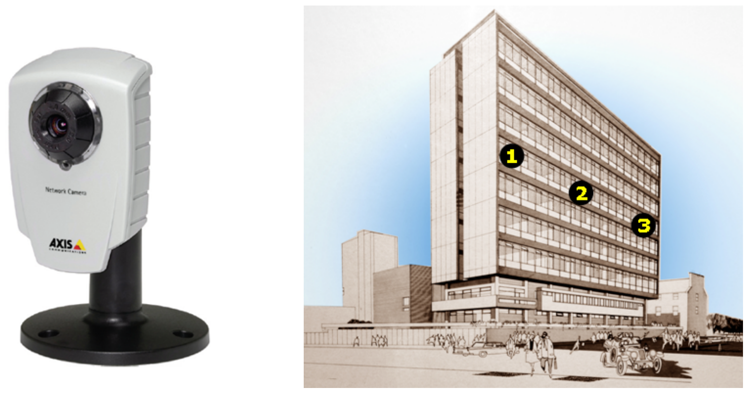


Figure 2.4: (a) Network camera 3/4 view. (b) Localization of the cameras in Appleton Tower. 1. Ionic, 2. Corinthian, 3. Doric.

Feature	Detail
Model	Axis 207
Resolution	1280x1024 pixels
Sensor	RGB CMOS
Lens	3.6 mm (0.5 m - infinity)
Frame rate	12fps

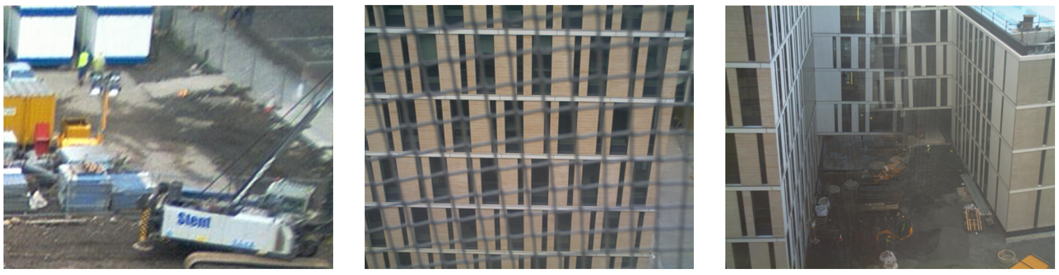
Table 2.1: Network Camera Specifications

detected, when the construction was already in progress, that higher floors of the building were not going to be visualized by the cameras. As will be mentioned later, in Section 3.2.3.2 this will create a difficulty on the jitter correction process. Corinthian camera was never manually adjusted but its stability was affected by external factors which produced on it a high vibration. In the Jitter Removal Results Section 3.2.4 it will be shown that the vibration of this camera is more than the other two. In Figure 2.5c two images, one before and other after the adjustment, are blended to illustrate the change on the viewpoint of the camera.

- Some images had a focusing problem. There is not really an alternative to solve this issue thus some regions of the final video will have blurred sections. An example of this is the top right region of the Corinthian camera as depicted in Figure 2.5a which looks more blurred than the rest of the image. These cameras are suitable for objects located at infinite, but this one had a defect.

- The cameras were affected by dust on the windows of the building. This created some dark and blurred images. There was not a simple method to clean the windows because of their high location.

- Finally, Appleton Tower was disturbed by strong winds during February 2008. This affected the stability of the windows and obliged the installation of a protection mesh to prevent any accident. The mesh covered the entire building including the places where the cameras were installed. Figure 2.5b shows an image of the mesh installed. This clearly caused harm to our database; therefore none of the images from this period were taken for the processing.



(a) (b) (c)

Figure 2.5: Difficulties during image acquisition process. (a) Blur in image portion from Corinthian camera. (b) Mesh for strong winds protection. (c) Manual adjustment of the camera. (Note the incorrect overlapping of images before and after the adjustment)

2.4 Image Database.

The capture of images from the Informatics Forum has been a process of approximately 2.5 years. This was a completely automated process and little human supervision was required. It was necessary occasional inspection of the cameras and servers operation to verify their continuous operation. Most of this maintenance could be done remotely through an internet connection and access to the university Linux system except for manual adjustments to the position of the camera that required physical intervention. The result of this capture process generated an very large amount of image data, and this is essentially the input of our project.

Images were stored at a rate of 1 frame per minute (fpm). This created approximately 600 daily images and 3000 weekly images taking into account the capture time was from 8:00 am to 6:00 pm and only weekdays. However, not all the images were suitable for the processing because their lightness was low depending on the time of day; hence they were not entirely clear. They are dark early in the morning and at the end of the day. Due to this cause, we are going to take only the brightest images from each day for the processing. This was done heuristically by using images taken close to noon; this is the period from 11:00 am to 2:00 pm. The frame rate was also reduced to 0.1 fpm which is equivalent to 6 images per hour.

Additionally there is a linear relationship between the number of images utilized and the computational cost. Therefore, a reduction of the number of frames was then convenient to produce results in a suitable amount of time. It is important to remark that even though cameras have same frame rate, there is not exact synchronization between them. The images do not have the same timestamps.

Table 2.2 specifies the most relevant characteristics of the images from the database.

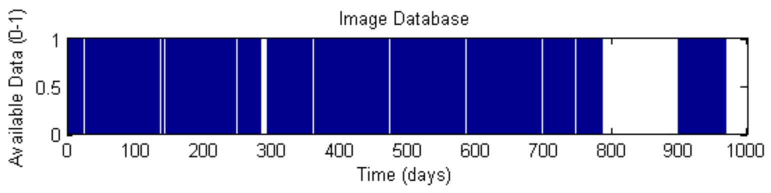
Note that the average size of the images from the database is about 185 Kb. In contrast, there was a big size difference between the images. When the construction started the image size was around 170kb, only a small amount of elements were on the scene, it consisted mostly of unoccupied terrain. Then in the middle of the construction process, several elements were taking part of the scene and the image size increased to a maximum of 320 Kb. Finally when the building façade was completed the image size was reduced again to approximately 100Kb due to the redundancy of windows and walls.

The image capture process had some gaps during the construction. This was generally because of problems with the server and/or the network cameras. This does not

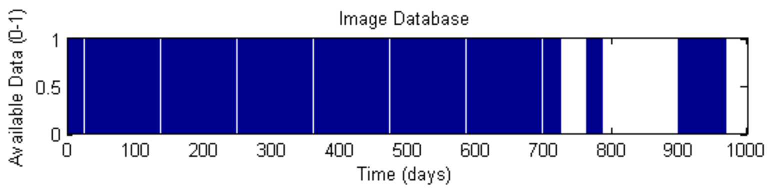
Feature	Units	Database	
		Total	Employed
Capture Time	hr	8–16	11–14
Frame rate	fpm	1	0.1
Resolution	Mpx	1.2	1.2
Number of Images	units	824.070	30.097
Image Size (Avg)	Kb	185	185
Storage space	Gb	143	5.58

Table 2.2: Image database characteristics.

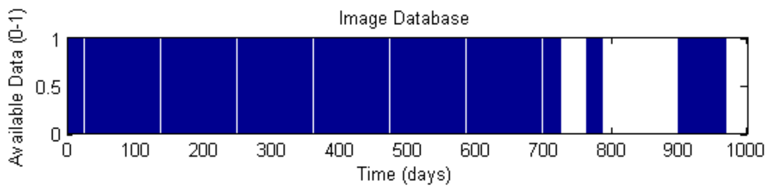
necessarily affect the results of the video but taking into account that these have been occurring at different times for the three cameras the final videos will be out of phase to some extent. Figure 2.6 shows a plot of the available data from the database for the three cameras. Note that there is not available data for the days with zero value, including weekends. These last are not clearly visible because the resolution of the plot. The last 100 days are not available for processing because these are the ones affected by the mesh installed on Appleton Tower.



(a)



(b)



(c)

Figure 2.6: Image Database. Available data - blue, missing data - white. (a) Doric. (b) Corinthian. (c) Ionic.

Chapter 3

Image Preprocessing

The input of the entire video reconstruction process is the image database from the Informatics Forum. These images have various artifacts such as noise, jitter between consecutive frames and problems with the color gain. In this chapter, the methods that have been developed to preprocess the data are described. Their main objective is to reduce effects on the raw images that can be harmful for the good performance of the image processing. These are described as follows:

- **Noise Reduction:** Capture devices and the compression process introduce noise in the images. This is decreased by applying non-invasive image filters that do not affect the image quality.
- **Jitter Removal:** The vibration of the cameras was a distinctive factor during the entire project. This was induced by motion sources near the hardware location. A method to align the frames and reduce this effect is implemented.
- **Color Adjustment:** Several images have incorrect color gains. Even though they are taken in similar conditions, many differences are noticeable between each other. This is mostly related with two factors: 1. The variety of weather changes during the project execution and 2. The mechanism utilized by the camera to adjust the gains of the visual input to produce good quality color images. This last is called Automatic White Balancing (AWB).

3.1 Noise Reduction

Raw data is affected by a variety of noise sources. Most of them are related with the electronic circuitry of the cameras (CCDs), dust on the lenses and also the artifacts produced by jpeg compression. Before working with the images, we evaluated the possibility of reducing this noise to improve the results of the experiments. Different filters were tested on the image to check their effectiveness. We tried initially with linear filters such as convolution using a Gaussian kernel of different window sizes but it was found that these smoothing filters blur the images and reduce the detail, especially in the edges, and this is not very appropriate for this application. As an alternative, a 3x3 Wiener filter [Washizawa and Yamashita, 2006] was employed. This filter is based on a statistical approach that depends on the local variance of the images, increasing the blurring when variance is small and decreasing it otherwise. Among linear operators this is the one that restores better the signal with respect to the squared error when it is averaged over the noise and original signal. This method is a commonly used approach to remove additive noise (Gaussian white noise).

Non-linear filters work using information from neighboring values and make a decision about the central pixel value. For instance, a median filter [Davies, 2004] preserves the image details. The output pixel is the result of applying the median of the brightness of the pixel window. We have also utilized a non-linear filter for noise reduction called conservative smoothing [R. Fisher and Wolfart., 2003]. This method attempts to make the pixel values consistent with their neighbors, hence it guarantees that the intensity of the pixels will be always bounded within the neighborhood pixel intensities. If any of the pixel intensity is outside these bounds the pixel is then shifted to the highest or lower values of the neighborhood. This filter is very effective for removing spikes and salt and pepper noise from images. The application of conservative smoothing on the image set is done separately for each color channel. The results of these two filters on the images are an appropriate combination for preserving image detail and improving image quality; the non-linear filter is applied first.

Figure 3.1 shows a synthetic example of a color image in which salt and pepper noise has been added and also a real example from the dataset where both filters (Weiner and conservative smoothing) have been applied. Observe the noise reduction after utilizing this method. Table 3.1 shows statistics about the effect of the non-linear filter in the images.

3.2 Jitter Removal

In this section, a method to stabilize the images based on statistical cross-correlation is described. It assumes that consecutive images are highly correlated and have small

Conservative Smoothing	
Average corrections per image	68103
Percentage of total number of pixels	5.19%
Difference between window limit values and central pixel	
Mean (px)	2.4742
Standard deviation (px)	2.2411

Table 3.1: Conservative smoothing statistics.

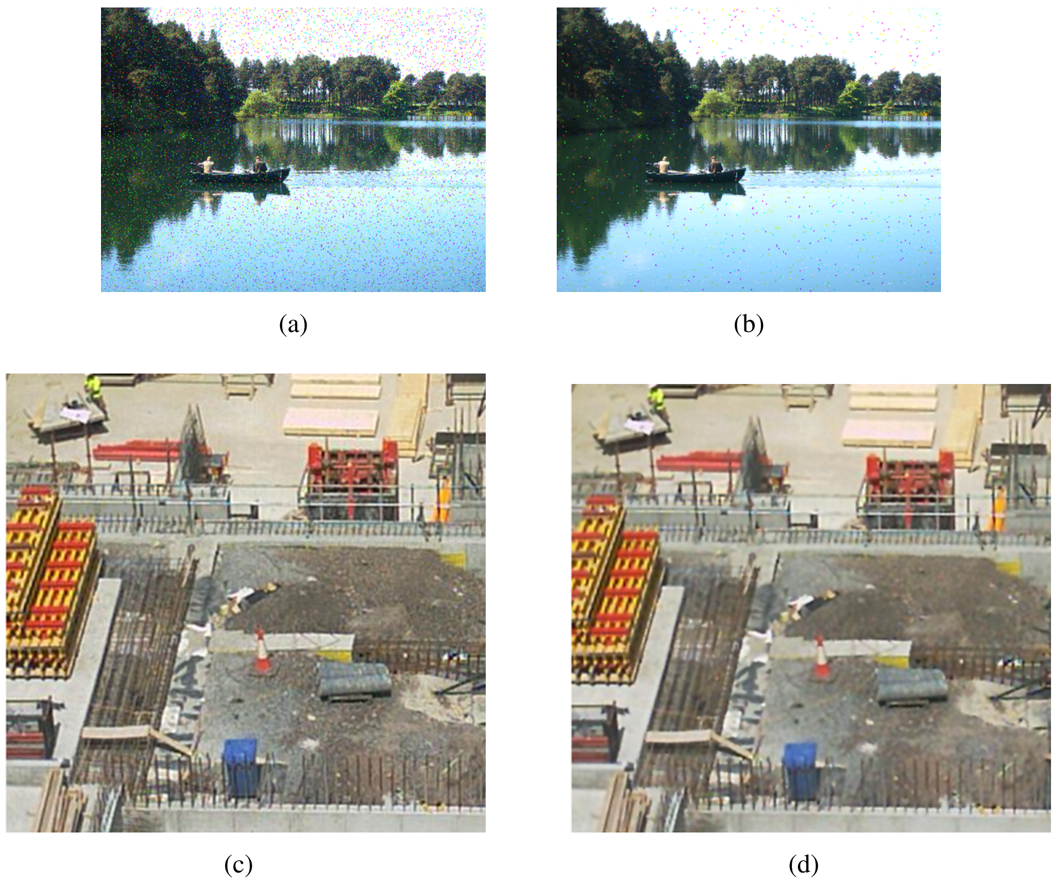


Figure 3.1: Conservative Smoothing (a) Lake image with salt and pepper noise added. (b) Results after applying the conservative smoothing filter. Close up of Forum image before noise reduction (c) and after using Wiener and conservative smoothing filters (d).

changes in their configuration such as the addition of a few elements in the scene and changes in the weather. In principle, we aim to find the shift between images and then arrange them in an optimal position. In Figure 3.2 we can see the jitter between two consecutive frames. A section of the images is enlarged to visualize the phenomenon which is in this case approximately 5 pixels difference in the vertical direction.

There are different jitter sources which affect the position of the cameras. It is possible that in our scenario these devices are being affected by strong winds, nearby machinery or defects in the mounting. As a consequence, the viewpoint is slightly different from frame to frame. These disturbances are noticeable when the video is played back. An image stabilization mechanism is required to solve this problem and also improve the quality of the image processing because this increases the correspondence between pixels of consecutive frames.

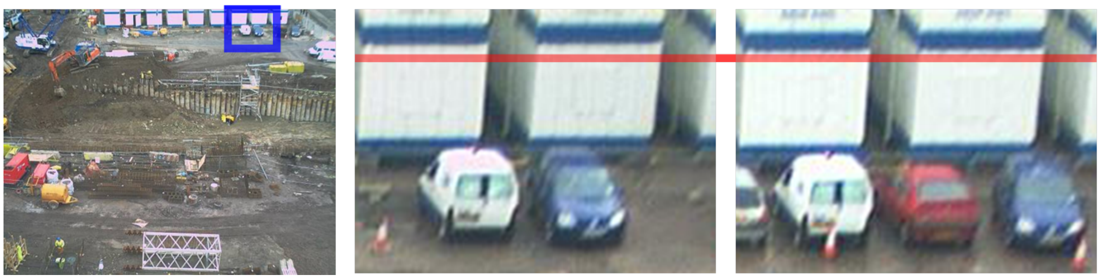


Figure 3.2: Jitter example between consecutive day images from Corinthian camera images. Red line shows the vertical movement of the camera.

3.2.1 Statistical Cross-correlation

Different alternatives have been considered to find relationships between images. Some of them are: (Weighted) Euclidean distance, Mean Squared Error [2], Histogram-based metric [3], Manhattan distance, etc. For this project, we have chosen Statistical Cross-correlation $(ccor)$ as an appropriate measure of similarity between images. It is a very useful tool to find matches because it is robust to noise and invariant to changes in lightness which makes it suitable for our necessities. This measure has results within a limited range $[-1,1]$. The highest cross-correlation is obtained when there is an exact match between images, this is equivalent to $ccor = 1$. Images are uncorrelated when $ccor \approx 0$ and they are inversely correlated when $ccor = -1$. In Figure 3.3 an example of the estimation of the $ccor$ for three different examples is depicted. The reference is the landscape image.

Fisher [Fisher and Oliver, 1995] discusses a method that generalizes the monochrome image cross-correlation to a multivariate case. Experiments in MRI and RGB color spaces showed the effectiveness of the approach. The mean of the correlation coefficients of individual channels is a good estimator of similarity for high-dimensional imagery. We will follow this measure of similarity to find the jitter of the RGB images from the image database. Subsequently the Cross-correlation of color images A and B is defined in 3.1 as:

$$ccor(A,B) = A \otimes B = \frac{1}{3 \cdot size(A)} \cdot \sum_{c \in \{r,g,b\}} \sum_{(i,j) \in A} \frac{\left(A_{i,j,c} - \mu_c^A\right)}{\sigma_c^A} \frac{\left(B_{i,j,c} - \mu_c^B\right)}{\sigma_c^B} \tag{3.1}$$

where μ_c^A and σ_c^A are the mean and standard deviation of image A in color channel c, $size(A)$ is the number of pixels in image A. We assume that $size(A) = size(B)$. When normalized data is used, $ccor$ reflects the true linear Pearson correlation.

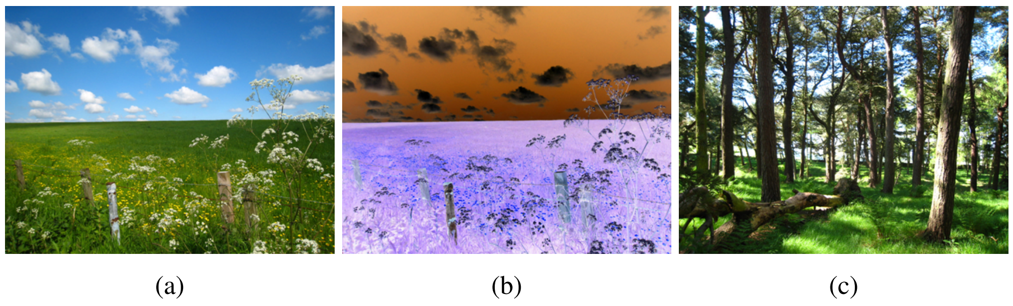


Figure 3.3: Cross-correlation values for color images between reference image and : (a) Landscape image . $ccor = 1.0$. (b) Landscape negative image. $ccor = -1.0$. (c) Forest image. $ccor = 0.1216$.

One of the advantages about this measure of similarity is that it is insensitive to color level variations. We have noticed in the images that the lighting levels change substantially from one day to another and there are additionally effects generated by the automatic color balance mechanism of the cameras. This approach will neglect these deviations.

A complete description of the relationship between consecutive frames in terms of an image projection consists of a series of different transformations such as 3D translations and rotations. But based on the fact that the elements of the scene are very far from the camera we can assume that they are close to infinite and the model can be simplified in this manner:

- The jitter can be described as a two-dimensional translation of the image.

- No rotation or shear of the image is considered.

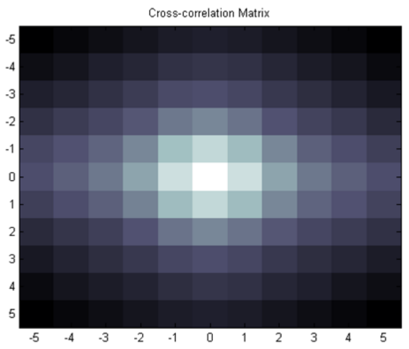
3.2.2 Cross-correlation Matrix

A cross-correlation matrix is used to find the optimal location of the images to reduce the jitter. The process consist of taking a pair of images A and B. These images are reduced in size by cropping the edges (e.g. 3 pixels $offset$). The reduced image A is centered with respect to the original and B is shifted horizontally and vertically according to the vector $\tilde{\Delta} = (\delta_x, \delta_y)$, $\delta \in \mathbb{Z}, -offset \leq \delta \leq offset$. The mathematical description of this matrix is shown in equation 3.2.

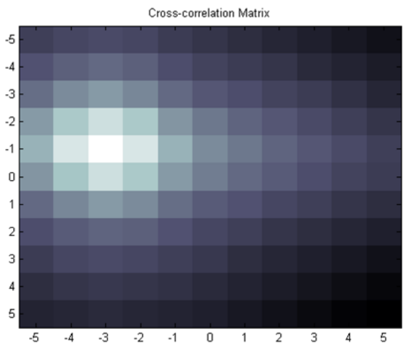
$$C(\delta_x, \delta_y) = \frac{1}{3 \cdot size(A)} \cdot \sum_{c \in \{r,g,b\}} \sum_{(i,j) \in A} \frac{\left(A_{i,j,c} - \mu_c^A\right)}{\sigma_c^A} \frac{\left(B(\delta_x, \delta_y)_{i,j,c} - \mu_c^B\right)}{\sigma_c^B} \tag{3.2}$$

3.2.2.1 Discrete shift

The position of the matrix element with the highest cross-correlation value is the shift required to align a pair of images. Figure 3.4 shows two cross-correlation matrices. This example has been set up manually. The brightest regions symbolize higher correlation. Figure 3.4a is the result of applying $ccor()$ between one image and itself. In Figure 3.4b the original image has been shifted three positions to the right and one position down, consequently the matrix is indicating that the highest correlation will be obtained if the image is moved back to the original position, this is $(-3, -1)$.



(a)



(b)

Figure 3.4: Cross-correlation matrix. (a) No shift, maximum $ccor$ in center pixel. (b) Shift between consecutive frames $(-3, -1)$.

The cross-correlation is estimated in a limited window, a 7x7 neighborhood. This has been preferred to reduce the computational cost which increases quadratically with the window size. In addition we assume that the vibrations of consecutive days are small and are approximately inside this region. There are just only a few examples which are outside this range but they can be solved manually or left without modifications because these shifts are excessively large and corrections might affect the final video frame size. This will be clarified in section 3.2.3.

3.2.2.2 Accurate Jitter Estimation.

In reality the jitter of the camera is not a discrete value. In fact this value is real number inside the cross-correlation matrix position boundaries. The values of the cross correlation matrix can be used to find accurate jitter estimation by assuming that these values have been generated from a distribution. We could then find the point which is the maximum of this function. Initially we tried to fit the data to a 2D normal distribution but the shapes found were not always 'Gaussian'. Moreover the values of the elements of the matrix were very close to each other suggesting a Gaussian function with a high variance. Finally we implemented a more robust method based on bicubic interpolation. This uses information from the pixels of 16 neighbors to estimate the cross-correlation intermediate values. This technique is commonly used on image resampling to find color values when the number of the pixels in an image is increased and it gives more realistic results than simpler methods such as nearest neighbors and bilinear interpolation [Świerczyński and Rokita, 2008].

In figure 3.5, an example of bicubic interpolation is depicted. Plot 3.5a shows the cross-correlation matrix in a 3D space where x and y axis are the matrix positions and z is the $ccor$ value. As can be seen this plot is not smooth. After the interpolation of the matrix values, we found the intermediate values of the matrix by reducing the grid size as shown in 3.5b. This result is smoother that the original matrix and it was found that the maximum cross correlation using the interpolation method was equal to $(-0.1, 1.6)$ instead of using the discrete shift estimation which found a shift of $(0, 2)$ pixels.

This estimation is very useful to find the right amount of movement of the camera through time. The difficulty with this approach is that it is not possible to carry out the exact shift of the images without affecting the image data. This would imply the interpolation of the image pixel values. Subsequently, this will cause problems in the processing stage because of the weighted blending of the image colors using nearest

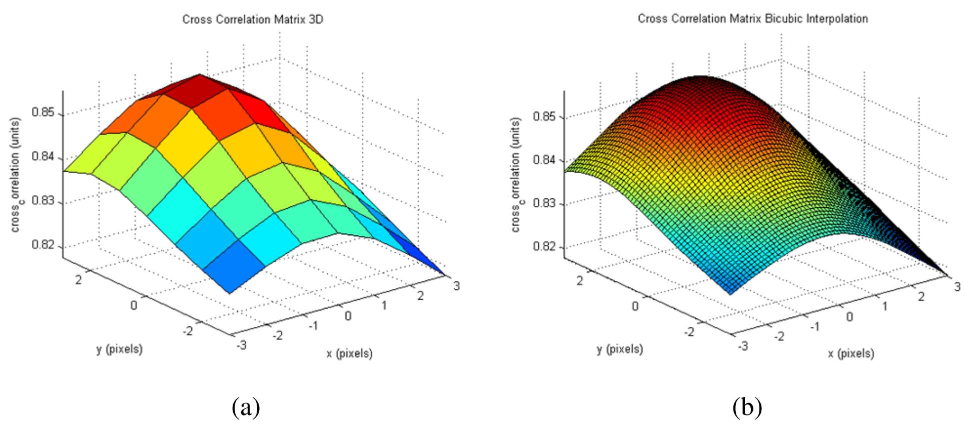


Figure 3.5: Cross-correlation matrix interpolation example. (a) $7x7$ $ccor$ matrix in 3D space. (b) Bicubic interpolation.

neighborhood data. The accurate method will be only used during the evaluation of the jitter correction but not for the image stabilization which used the rounded integer values of the shift.

3.2.3 Image Stabilization.

We have evaluated the jitter of the cameras through time and it was found that the vibration of the images is different for the three cameras. The highest jitter was found in the Corinthian camera and this can be easily observed in the rendered videos. Figure 3.7 shows the horizontal and vertical shift relating consecutive images for the complete set of images. This has been done employing the discrete shift method. The plot also takes into account the days without image capture to have the real time scale. This is why the plot reaches 800 days on the time axis whereas the total of available days is near 600. In these cases the x and y positions are kept constant.

3.2.3.1 Image Alignment

This has been done in two stages. As will be discussed in section 4, we have first to estimate representative images for each day (day image), then process them to obtain the final video. We found that the jitter can occur in images from the same day and certainly from different days. As a result, the jitter correction is applied first to one day samples to produce day images and then to the resulting image set. The procedure is similar for both cases and it is detailed as follows:

- The horizontal and vertical shift of the camera is estimated for a given set and

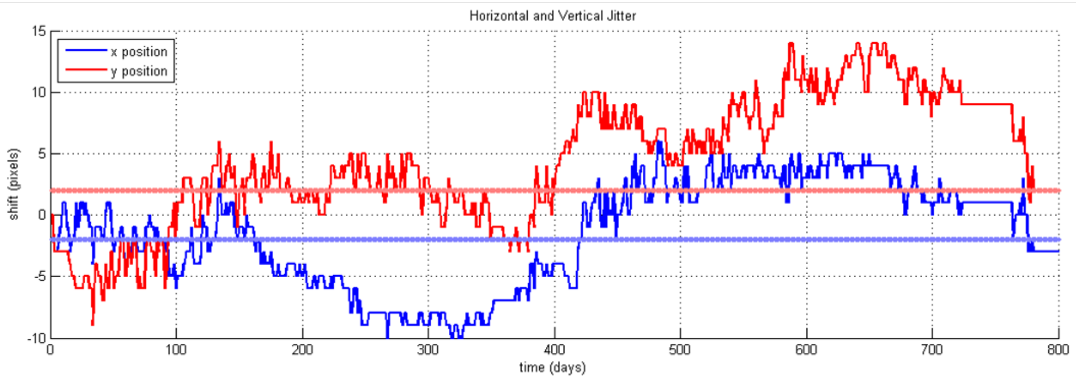


Figure 3.6: Horizontal and vertical shift of Corinthian camera over the entire building construction. Red and blue lines correspond to the alignment position with the least loss of data from image edges.

a (x, y) position is associated to each frame representing the cumulative shift starting from the first image at (0,0) as depicted on figure 3.6.

- A motion trail is generated from the jitter estimations to describe the camera displacement through time.
- The mean location of this trail is chosen as the center of the shift (see horizontal red and blue lines on figure 3.6). This will correspond to the position with the least loss of data from the image edges.
- All the images are shifted to the center in discrete steps.
- The edges without content are removed. This clearly produces a reduction of the image. The size reduction is calculated using the maximum shift of the images with respect to the alignment position.

The process of image alignment is depicted in figure 3.7. Orange bars represent images of consecutive days. After the jitter estimation they are shifted according to the cross-correlation matrix and aligned correctly.

3.2.3.2 Manual adjustment.

As was mentioned before, we had assumed only 2D translations of the images. There are some special cases were the jitter is better described as a rotation rather that a translation. The current method for jitter estimation finds a reasonable position but

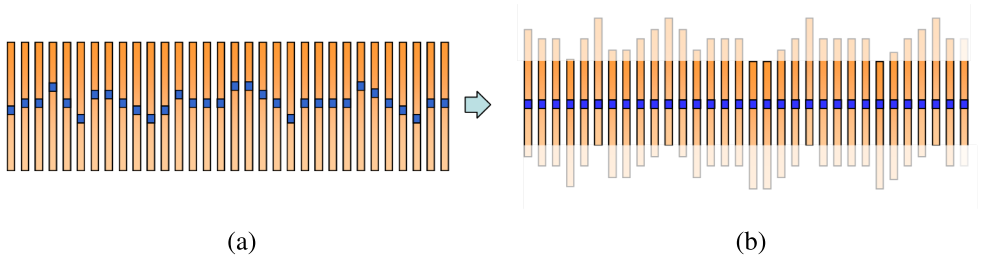


Figure 3.7: Jitter correction process. (a) Raw images. The blue squares show the shift between consecutive frames. (b) Image alignment. Image size is reduced to remove empty edges.

sometimes a manual adjustment of the image position can produce a perceptually better result.

Additionally, some other adjustments were done when the jitter exceeded the neighborhood window size. Even though the algorithm reduced the amount of jitter in these frame it was not enough to correct it entirely. This occurred approximately 2 times for each camera. When manual movements of the camera were registered, it was not possible to apply any shift because these were very large and the image size reduction after alignment would have been inappropriate. These cases were left without shift modifications, so it will be noticeable during the video playback one or two large jumps in the camera viewpoint.

3.2.4 Jitter Removal Results

Once the image stabilization is implemented, the jitter level can be measured on the processed images to find the amount of error. Afterwards we can compare the results of the approach with the original data. For this, we have used the method based on bicubic interpolation because it still finds errors in the jitter correction that can not be sensed using the discrete method.

In Figure 3.8 the results of the jitter correction are depicted. The blue plot is the Euclidean distance of the shift vector $\|(\delta_x, \delta_y)\|$ for each image pair of consecutive days using raw data. Similarly the red color shows the distance of the processed daily images. These results are for the three cameras and they clearly show an improvement in the stability of the images and the effectiveness of the method. Additionally, we can observe that there is still a small amount of vibration which is also perceived in the final video. As a further improvement, this can be solved for instance by shifting the images to match the exact position, but this requires interpolation of the pixels. The

Camera	Dataset	Mean (px)	Std (px)	SSE (px^2)
Corinthian	Original	1.3501	1.0044	1593.17
	Processed	0.0869	0.0855	8.37
Doric	Original	0.3944	0.4373	207.21
	Processed	0.2524	0.2169	66.21
Ionic	Original	0.8731	0.7715	788.15
	Processed	0.31184	0.2012	80.00

Table 3.2: Jitter Removal Results

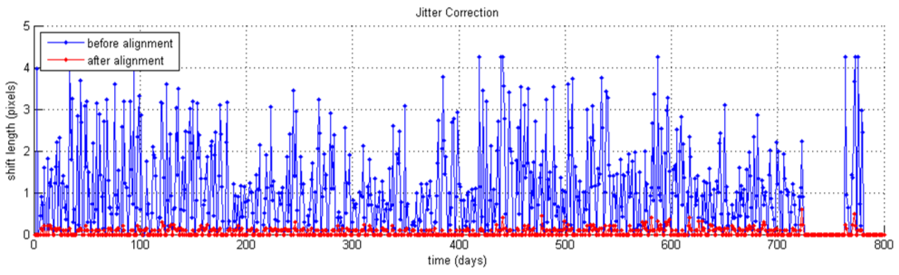
method can be implemented after the images have been processed and no additional modifications are required.

Corinthian was the camera with the strongest jitter. This is seen in Figure 3.8a and in Table 3.2 where the mean, standard deviation, and the sum of squared error (SSE) of original and processed images are illustrated. Note that the mean jitter is much higher in Corinthian than in Doric and Ionic. It is clear that some external vibration was affecting the position of Corinthian camera. After the processing all the cameras have a small deviation which is no more than 0.22 pixels. This is certainly the effect that remains on the final video.

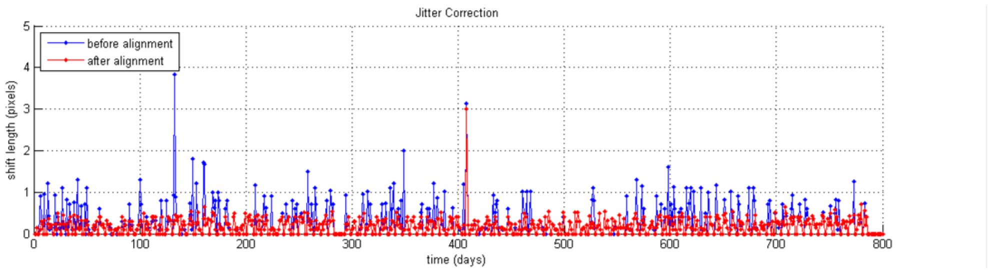
3.3 Illumination Change Normalization

Another relevant finding on the raw images is the high range of variations that can be found in the pixel color values. Apart from the expected normal changes of the construction process such as the addition of new structures, machinery, etc., other factors affect the images and make them appear very distant from each other even if they are continuous in time. Some examples of these changes are depicted in Figure 3.9. The main causes of these variations are described as follows:

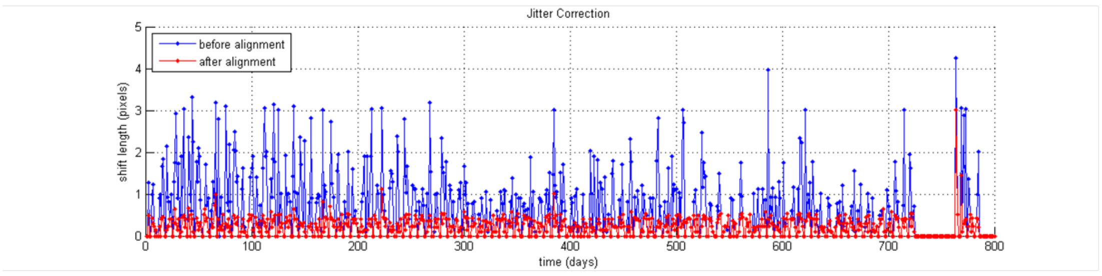
- Weather changes. We have captured all types of weather during the entire construction. They include days with sun, fog, snow, rain, clouds, etc. (refer to Figures 3.9a, 3.9b and 3.9c). This obviously modifies the scene conditions and the color distribution of the images.
- The previous conditions come together with the emergence of shadows and highlights that darken or brighten the images. For instance, the shadow of Appleton



(a)



(b)



(c)

Figure 3.8: Jitter correction results. Blue and red curves show camera jitter before and after alignment. (a) Corinthian. (b) Doric. (c) Ionic.

tower over the Forum construction in sunny days occurs regularly and produces big dark regions in the images. In the rendered video it is possible to see how the shadow shifts its position depending on the time of the year.

- The cameras have integrated an Automatic White Balancing (AWB) system that chooses the appropriate gains for each color channel (RGB) to represent more realistically the images. There are different methods of doing AWB. Examples of this are using edge detection [Lin, 2006], adjacent RGB channels [Lam et al., 2004] and neural networks [Chen et al., 2007]. We have no knowledge about the type of transformation that the camera utilizes to balance the color values. The main problem is that we have detected many erroneously balanced images which affect the processing. An example of this problem can be found in Figure 3.9d. Notice that there are many of these cases in the dataset. Sometimes the images look more bluish or reddish than expected. A particular point related with this is that this balancing adjustment is a problem of the three cameras. For instance, in Figure 3.10, we see how the three cameras have registered simultaneously blue scenes when the image previously taken was clearly fine.

These aspects have consequences on our implementation. Abrupt color level transitions from one day to the next causes flicker that makes the video look confusing. A good video should have gradual changes and stability on the images. Apart from the different lightness levels, we have color shifts that make strong differences between consecutive days. These two elements are not suitable in a good TLV. Finally these differences on the images affect the creation of day images, especially when using methods such as multivariate median filter. This creates the emergence of ghosts which be clarified in Section 4.2.

3.3.1 Histogram Based Techniques

To have an idea of what was happening with the images in terms of color variations, an evaluation of different image properties such as luminosity, color shift, standard deviation and average color values was implemented. This was done taking samples images from each day. In theory we expected to find a smooth variation in these values. Here we assumed that the only change in the images was the addition or subtraction of elements from the scene.

The Michelson contrast measure [Damera-Venkata et al., 2000] was employed to evaluate the deviation of the image levels through the dataset. This is defined in Equa-

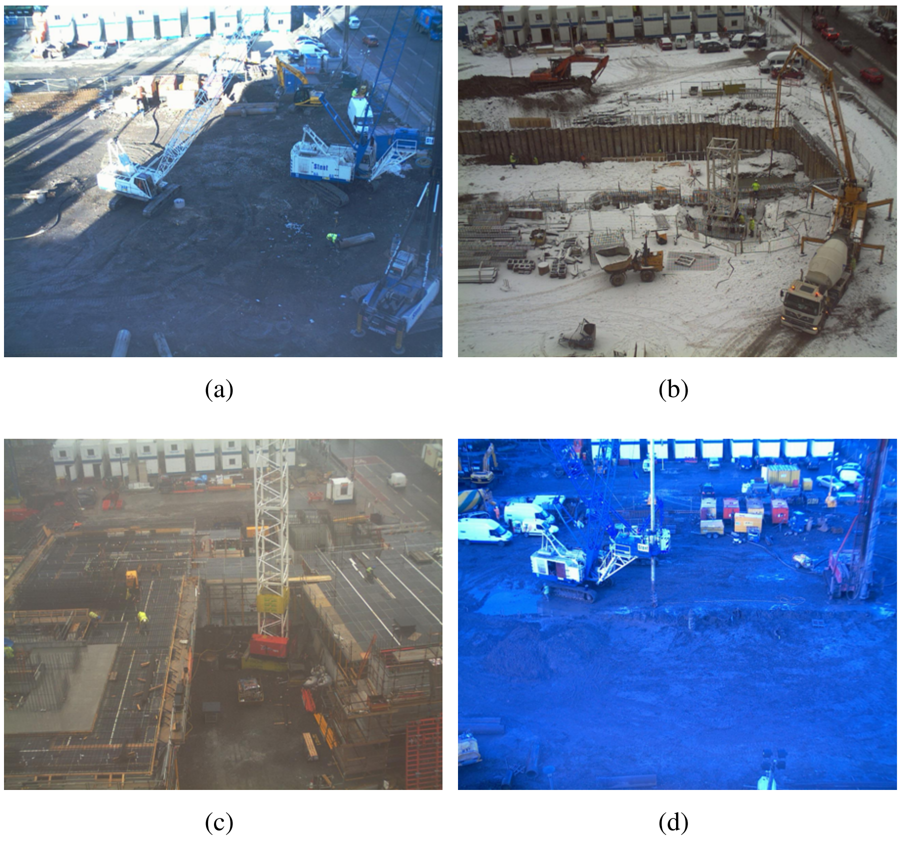


Figure 3.9: Color changes due to the weather and camera. (a) Sun (shadows and highlights). (b) Snow. (c) Fog. (d) Automatic White Balancing.

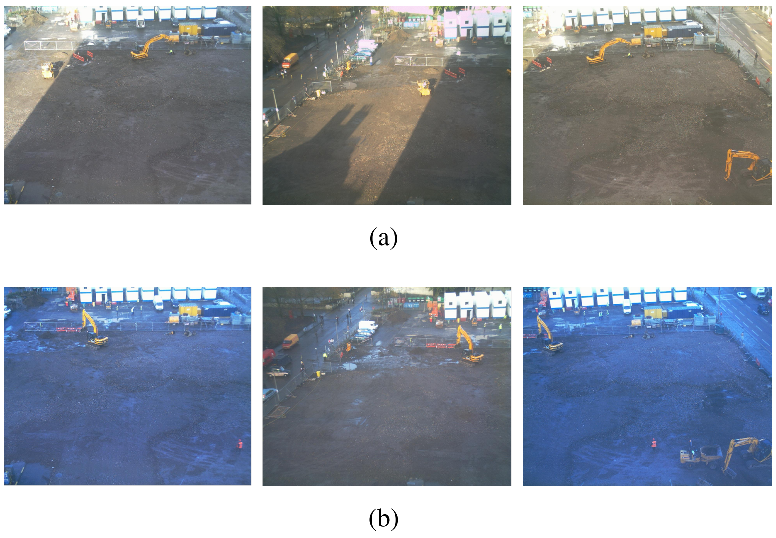


Figure 3.10: Automatic White Balancing Problems with the three cameras. (a) 2005-Nov-02 Images with right color values. (b) after 10 minutes, the images turn bluish almost simultaneously.

tion 3.3 where C_M is the Michelson contrast, L_{min} the minimum and L_{max} the maximum luminance values. The instability of this measure between consecutive frames gives an idea about the evident flicker in the video. For this, raw images where taken from each day (the one captured at noon). The Luminance image $L_{(i,j)}$ was calculated by averaging the RGB channels $(R+G+B)/3$.

$$C_M = \frac{L_{max} - L_{min}}{L_{max} + L_{min}} \tag{3.3}$$

L_{min} and L_{max} are in our case the lower and upper limits that represent the 10% and 90% of the image histogram values, with this, we wanted to remove outliers with low and high intensities from the contrast calculation. The results of the contrast measure for the Ionic camera are shown in Figure 3.11. They show how the light changes extensively. This is a sample of the instances that makes the time lapse video production more difficult. All these elements should be reduced at the end of the whole process.

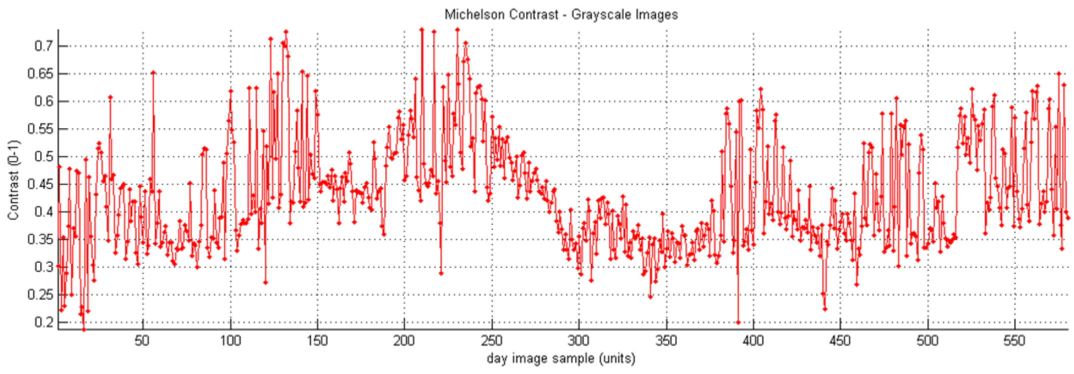


Figure 3.11: Michelson contrast measure for Ionic Images.

Three different methods based on histogram information have been explored to solve issues with the changes in illumination. They are described as follows:

3.3.1.1 Image Adjustment.

The first approach to color normalization tried to standardize the image assuming that the histograms distributions were similar but they had different scales and shifts. The idea was essentially that with one rule it was possible to take all images to a reference point. The solution was to utilize an image adjustment method which modifies the histogram in such way that the result makes use of all the available histogram range, from 0 to 255. This was done in two different ways:

3.3.1.1.1 Contrast Stretching.

Similarly to what we used in Michelson contrast estimation we assumed that 2% (1% top and bottom) of the data were outliers, we reduced the percentage to avoid saturation of many pixel intensities after stretching. These elements were removed to find the appropriate limits of the histogram distribution. Then, a linear transformation was applied to the images according to Equation 3.4.

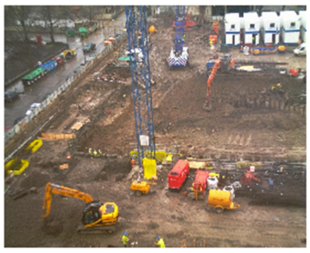
$$I'_{(i,j)} = \begin{cases} 0 & I_{(i,j)} \leq V_{min} \\ 255 \cdot \frac{I_{(i,j)} - V_{min}}{V_{max} - V_{min}} & V_{min} < I_{(i,j)} < V_{max} \\ 255 & I_{(i,j)} \geq V_{max} \end{cases} \tag{3.4}$$

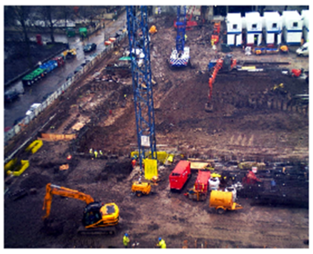
where I and I' are the image before and after the stretching respectively; V_{min} and V_{max} are the color values equivalent to the upper and lower boundaries of the distribution 1% and 99%. We worked initially in the RGB color space and produced the results depicted in Figure 3.12b.

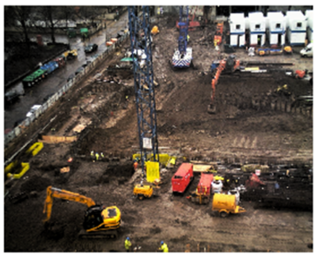
One of the issues with this method is that the stretching was performed independently for each color channel meaning that it could create eventually color values that were not originally in the image. The solution for this was to transform the images to chromaticity coordinates and lightness component (rgs) and implement the histogram stretching only on the lightness channel, hence the results modified the intensity values of the color channels simultaneously removing the possibility of the emergence of new colors. After this, it was possible to return to RGB space. The results of this other approach are depicted in Figure 3.12c and the relation between RGS and RGB spaces is shown in Equation 3.5.

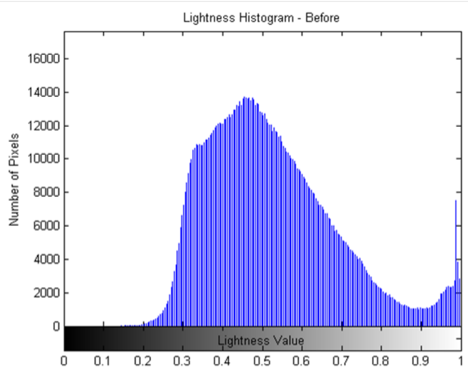
$$r = \frac{R}{R+G+B}, \qquad g = \frac{G}{R+G+B}, \qquad s = \frac{R+G+B}{3} \tag{3.5}$$

This approach worked relatively well in many of the images from the set. Note that the results of the RGB domain are not particularly right. The image in Figure 3.12b has a purple tone. On the other hand, the other method preserves the original colors and increases the details. Some issues with this method appeared because some images had different histogram shapes and the stretching was not really the best solution to normalize them. For instance, sunny images have a big amount of saturated data on 255 so the approach was not doing any improvement in the right side of the histogram because more than 1% of the data was already stretched.

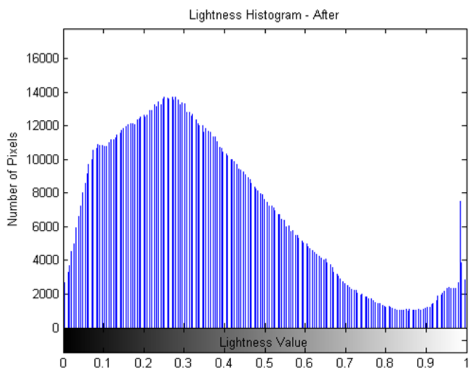

(a)


(b)


(c)



(d)



(e)

Figure 3.12: Image stretching. (a) Original image. (b) Image stretching in RGB domain (each channel separately). (c) Image after luminosity channel Stretching. (d) Luminosity Histogram of original Image. (e) Luminosity Histogram of stretched Image.

3.3.1.2 Histogram equalization

The second histogram-based approach to normalize image values consisted of a non-linear transformation of the image color values to match a reference histogram. This is very useful when images are being compared and have been captured under different conditions such as weather and AWB gains. For instance, if we take the histogram of a reference image H_{ref}, then we can take a second image and modify each color value by a factor. This is performed independently on each color channel and it certainly fixes many differences between the images. An example of this stretching is shown in Figure 3.13, Notice that the color values of the images now look more similar. An analysis of image equalization techniques can be found in [Delon, 2004].

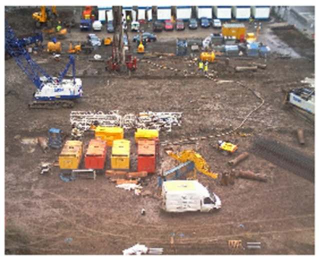
One of the advantages of this method is that we can fix the problem of the automatic white balancing of the camera once we have a good reference image. Several alternatives were implemented to find the reference image. One of them was to obtain the median histogram given a set of images captured in a reduced gap of time and assume that the variations on the images were not excessive. It is clear that the AWB problem does not occur in all the images; hence this median histogram can be used as a reference.

This method was implemented to normalize images from the same day by taking the median histogram. One difficulty with this is that the lightness conditions for each day were very different, so the equalization not necessarily works similarly between equalized images of consecutive days. This problem can be solved by producing images with information of the neighboring images. This approach will be discussed in section 4.2.4.

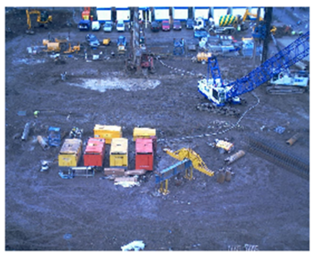
3.3.1.3 Linear Transformation and Corresponding Pixels

We can use apriori knowledge from the images to improve the results of the normalization of the image color values. In this approach we have assumed that consecutive images have just a small amount of changes and most of the elements in the image stay without modifications. Hence there might be a transformation between the intensity of most of the pixels from an image to a consecutive one. Here we also assumed that the images were already aligned using the jitter removal method.

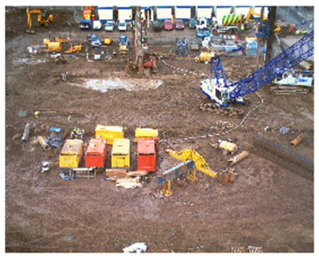
This third method considers that the relationship between values of an image pair is a linear transformation. We aim to find a method to take image B and transform it to match the colors values of image A in the form $I'_{(i,j)} = m \cdot I_{(i,j)} + b$. This will reduce



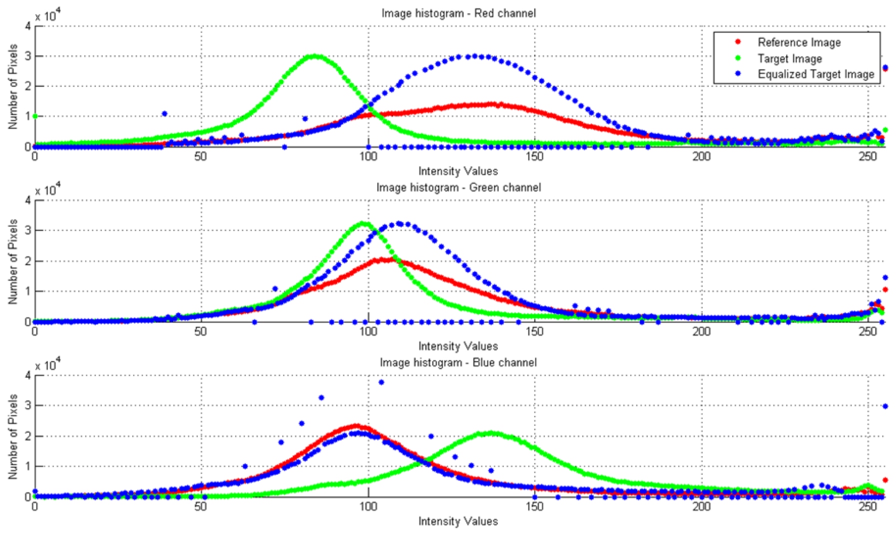
(a)



(b)



(c)



(d)

Figure 3.13: Image equalization. (a) Reference Image . (b) Target Image (it has white balancing problems). (c) Equalized target image. (d) Histograms in RGB channels for the three images, note how the blue histograms which is the equalized images matches the reference image histogram.

to some extent the problems when big differences between consecutive frames appear because the linear transformation is applied to all the elements of the images and it is not possible to change radically the images (differently to previous method). This can be seen as a regularization method.

The process is based on the idea that most of the pixels with the same color intensity in image A will change to the same color intensity in image B. The process is described as follows:

- For each pixel value $v = \{1, 2, \ldots, 255\}$, the x and y positions $I_{(i,j)}$of the pixels with this particular value are obtained.
- Taking into account the no-jitter and image similarity assumptions, most of the pixel positions values that had the same value on image A, will have the same value in B (but not necessarily equal to the one in A). So we can link the intensity of pixels $I_{(i,j)}$ in A to the median of intensity values in B.
- Once the relationship for all levels is known, we can see the problem as an optimization which can be solved to find the coefficients m and b that reduce the error between both images. The error criterion was least square error (LSE) as shown in Equation 3.6.

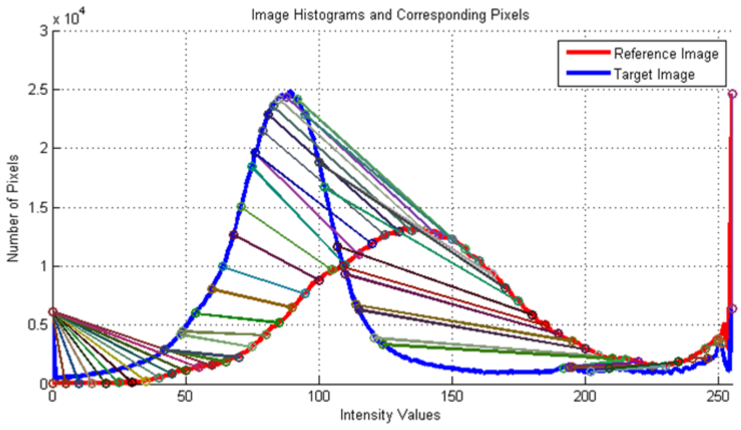
$$(m,b) = \underset{(m,b)}{\arg\min} \sum_{i=1}^{255} (v - (m \cdot f(v) + b))^2 \tag{3.6}$$

where $f(v)$ is the median intensity of all pixels in image B whose corresponding pixels in image A was value i.

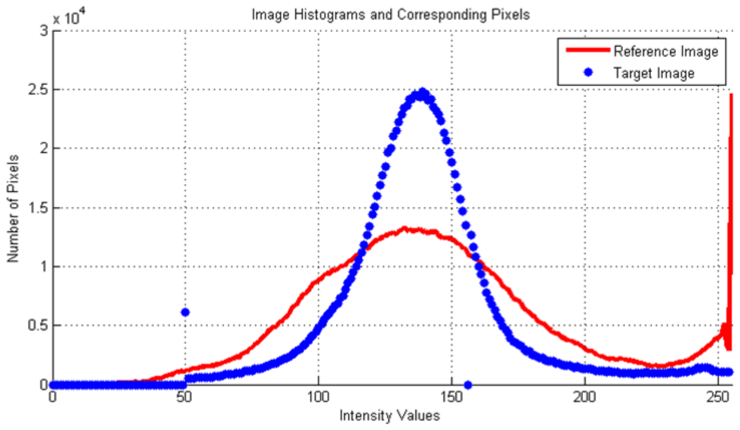
The results of this method are depicted in Figure 3.14. The connections in Figure 3.14a represent the correspondence of the values between the images and Figure 3.14b shows the result of the linear transformation after applying optimization method. Note that the position of the new histogram matches better the reference.

3.3.2 Grey World Assumption

Previous histogram-based techniques worked relatively well. The main issue with these methods was the selection of the reference image. It means that the process works very accurately when we have identified a pair of images (target and reference). These alternatives have lack of generalization over the whole dataset. This suggests that we need a more general approach to find a global solution to the image color



(a)



(b)

Figure 3.14: Corresponding pixels method. (a) Histogram of red channel of two consecutive images A and B, last one with unbalanced whites. The lines that connect the histograms are the corresponding pixel values. (b) Histogram of reference image A and transformed image B after the LSE minimization.

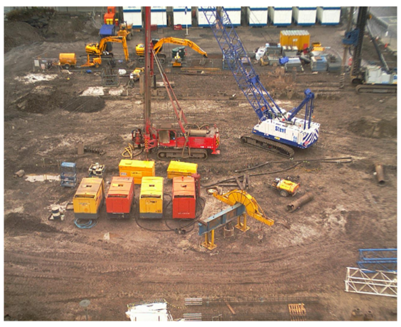
problem. This approach is based on a widespread method that can be applied in this instance because it satisfies many of its assumptions. It is probable that histogram based methods work better than this for an image pair but not in the whole set and this is essentially what we require.

This section describes the Grey World Assumption (GWA) and some variances of it to normalize the images based on a general idea about the behavior of the data. This assumption states that given an image with enough amounts of color variations, the average of the color components should be equal to a common gray value. There are several algorithms in the market used to do AWB. Some of them are: Using adjacent channels [Lam et al., 2004] and based on edge detection [Lin, 2006]. They modify the image color properties using the signal coming directly from CCD or CMOS sensors. They try to balance the gains for each color channel and make the image look more realistic. This is because there are different types or light sources with different frequency components such as sunlight, incandescent, fluorescent, etc. hence the gains of the sensors have to be moved to depict the true color of the elements on the scene. In our data we are being affected in most situations by solar light. However this changes with the weather conditions, time of day, number of clouds, etc.

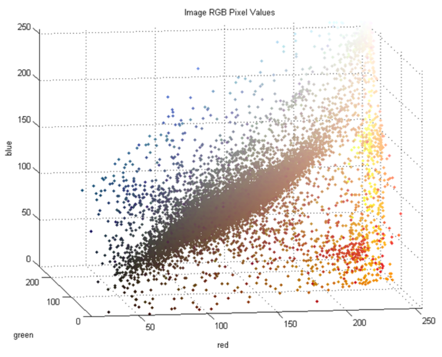
As was shown in Section 3.3, the camera is not doing a proper white balancing and this is affecting the images quality. One particular difficulty with our data is that we do not have the complete data from the CCD sensor but only a limited range of intensity values from 0 to 255. If the values from image are saturated (close to the bounds), then they are probably outliers whose original position was at a different intensity. Nevertheless we have taken the GWA assumption as our image normalization method. There are some characteristics of our images that make it appropriate to be used. These are:

- The images from the dataset are composed by a large amount of color values.
- The color of several elements in the building is 'grey' (this mean values near to $V_{(r,g,b)} = (128, 128, 128)$)
- It can be applied in a general form to all the images from the database and no reference image is required.

These assumptions can be clarified better in Figure 3.15b. The pixels of a sample image have been plotted in RGB space. Because of the large band of pixels lying near the main diagonal of the cube, they clearly show that the image is composed by several color values and that most of them are concentrated in the grey region. Most of the images have a distribution like this one but shifted in position and scale. The Grey World Assumption tries to take the clusters of all these images and put them in the same region. Additionally, taking into account that the images just change slightly from one frame to the next, this method solves many of the issues with AWB of the camera.



(a)



(b)

Figure 3.15: Validity of GWA in Forum Images. (a) Image sample from Corinthian Camera. (b) Scatter plot of pixel values in RGB space. Each point of the graph has associated its true RGB color.

In the GWA the gains of the three color channels (RGB) are adjusted independently to satisfy the following criteria:

$$\bar{R} = \bar{G} = \bar{B} \tag{3.7}$$

This method does not work appropriately if there is a small amount of colors, but this does not occur on the images from our database. Generally, the value to which the means of each channel is shifted is the average of the three color means $\bar{R} + \bar{G} + \bar{B}/3$. In our approach we have considered that this is not really suitable if we want to create a relationship between the colors of consecutive images. For this, the mean RGB values to which the histograms are updated is fixed for all the images. The transformation implemented on the images is described in Equation 3.8.

$$A'_{i,j,c} = \frac{grey\_value}{\bar{A_c}} . A_{i,j,c} \tag{3.8}$$

where $A'_{i,j,c}$ is the resulting intensity for a given pixel (i, j) and its color channel c. $A_{i,j,c}$ is the original image intensity, $\bar{A_c}$ is the mean of the image values for each color and $grey\_value$ is the desired mean intensity.

Even though this approach is not entirely realistic, it will guarantee a smoother transition on the color levels of the images. For instance, we could fix the mean to $(\bar{R}, \bar{G}, \bar{B}) = (128, 128, 128)$ but we can also change the proportion of each color channel to a different value to make the final colors more realistic. For instance, if the amount of red is increased respect to the blue and green, it will create in the images a warmer effect.

Figure 3.16 shows the gray world assumption applied to a set of images from the same day. Some of the original images have a strong imbalance with their color. They look bluish. After the method is applied all the images have a similar color distribution.

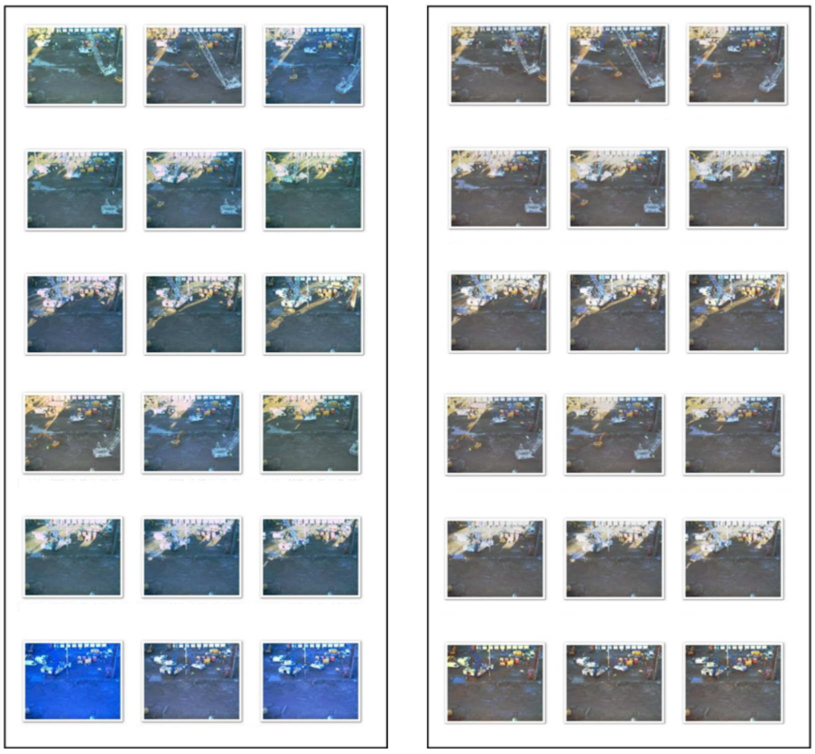

(a) (b)

Figure 3.16: Grey world assumption example. The method is applied to 18 images of the same day. (a) Raw Images. (b) Images after GWA is applied.

3.3.2.1 Improved Gray World Assumption.

There are some issues with the original GWA that produce unexpected results. The main reason is that we do not have the data from the camera CCD sensor but from the images produced by the camera after an unknown AWB and a color normalization method was applied. After an inspection of the data in terms of RGB and rgs histograms, some patterns were found which can be utilized as prior knowledge with respect to the camera behavior to improve even further the image values.

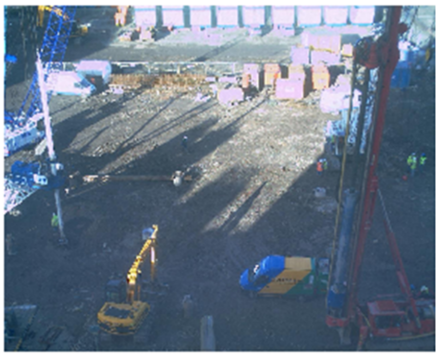
The cameras were mostly affected in the estimation of the correct red and blue values. This was less evident in the green channel. This was noticed after visualizing the histogram in chromaticity coordinates. The normalized green channel is regularly constant and inside a narrow window. The other two colors change constantly in shape and position. For instance, when days are very sunny the normalized red histogram shifts to the left and the blue to the right whereas the green preserves its position. Additionally the R, G, and B histograms have similar shapes but with a different stretching (see Figure 3.17c). We also saw that the scaling of red values was producing a high number of saturated elements. Notice in the graph that the red histogram looks like enlarged when compared to the others, hence after scaling, many pixels will have values above 255 but they are truncated due to the available range. Moreover, some blue values before scaling are already saturated.

Another conclusion from the data was that there is a high correlation between the R, G, and B histograms of each image, but it seems not to be true in images with balancing problem. This observation was utilized to improve the initial GWA and produce a new method which first uses the green histogram as a reference for the Red and Blue channels. These two are stretched to match the green channel. Once this has been done it is possible to apply the GWA.

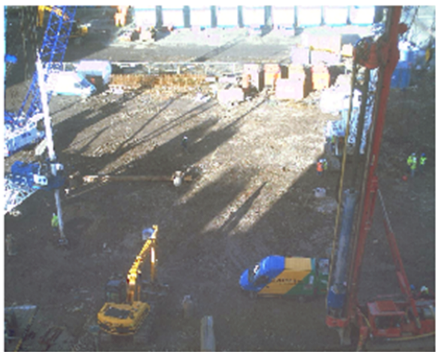
Moreover the ’grey’ color was modified to a smaller value because the mean of the images was smaller than $(128,128,128)$ hence every time GWA was applied the histograms were brightened producing new saturated elements.

The results of the improved method are depicted in Figure 3.17. The final color normalization algorithm is described as follows:

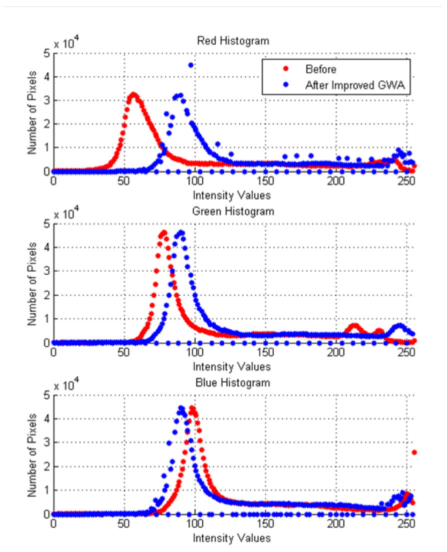
- Calculate the green histogram H_g from a sample image I.
- Smooth histogram H_g using a moving average filter of span equal to 5 to reduce its local variance.
- Stretch Red and Blue image channels (I_r and I_b) to match the smoothed green histogram Hs_g using the equalization method described in Section 3.3.1.2.
- Apply the grey world assumption to the image composed of the green channel I_g and the equalized red and blue channels (Ieq_r and Ieq_b) using Equation 3.8 and a $grey_value$ equal to 115.



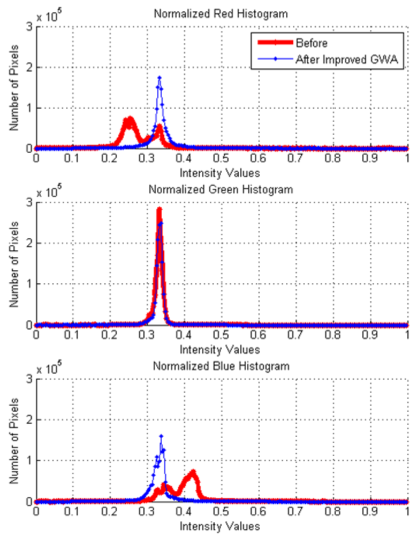
(a)



(b)



(c)



(d)

Figure 3.17: Improved Grey World Assumption. (a) Original Image. (b) Images after GWA. (c) RGB histograms. (d) Normalized RGB Histograms.

Chapter 4

Image Processing

In this chapter a general description of the methods implemented to produce time-lapse videos is presented. Once the raw images were preprocessed according to the stages described in Section 3, they were available for further processing. The time lapse video implementation has two main stages. First, day images are created. These are the most likely background estimation of each day. Then, the process is followed by the combination of these day images with their neighbors to produce a new set of frames for the final video. The transition of consecutive video frames is smoother after this step. For instance, moving obstacles such as vehicles that took more than one day on the scene can be removed from the video. These were not seen as foreground during the creation of day images.

We have implemented deterministic and probabilistic approaches to solve the problem. The deterministic approach uses essentially median filters and the probabilistic is based on different alternatives. For generating day images, it uses a non-parametric kernel density estimator and for processing two classifiers: Naive Bayes and a neural network.

4.1 Traditional Time-lapse

This method is based on the basic idea of time-lapse video generation. Initially we assumed we had to produce a representation of an event in a shorter period of time. As was mentioned before, the building blocks of the final video are images that represent one day. We took one image from each day to be part of the time-lapse video. All these images were selected at the same time (noon) because the light conditions between days are similar and generally appropriate. The images were selected from the database and arranged for the video generation. They were not processed to remove elements equalize colors, align frames, etc.

We evaluated the time required for the video representation. The selection of the time rate was influenced by the number of frames and also by the desired video playback time. The standard transmission rate in Europe is 25 fps. With this reference and taking into account that we have around 540 images for the video (this number is slightly different for each camera and does not consider images with the mesh), the final video will take 21.6 seconds. The length of the video can be changed to slower frame rate to produce longer videos. Sometimes this is a suitable solution because it is possible to watch in more detail the transition of events during the building construction. The duration of the video given by the number of images and the frame rate is $length = num\_images/framerate$, where $length$ is in seconds and $framerate$ in fps.

An example 6 consecutive images using the traditional approach is shown in Figure 4.1. Notice how the images look very distinct from one day to the next. Snow, solar glares and problems with the colors are clearly visible. The playback of this video will shows a confusing transition of images, and it is for this reason that further processing is required to improve the video quality. We have applied other techniques. The final frame rate of the videos was changed to 15 fps to visualize better the details of the construction. This is equivalent to a 36-second video sequence.

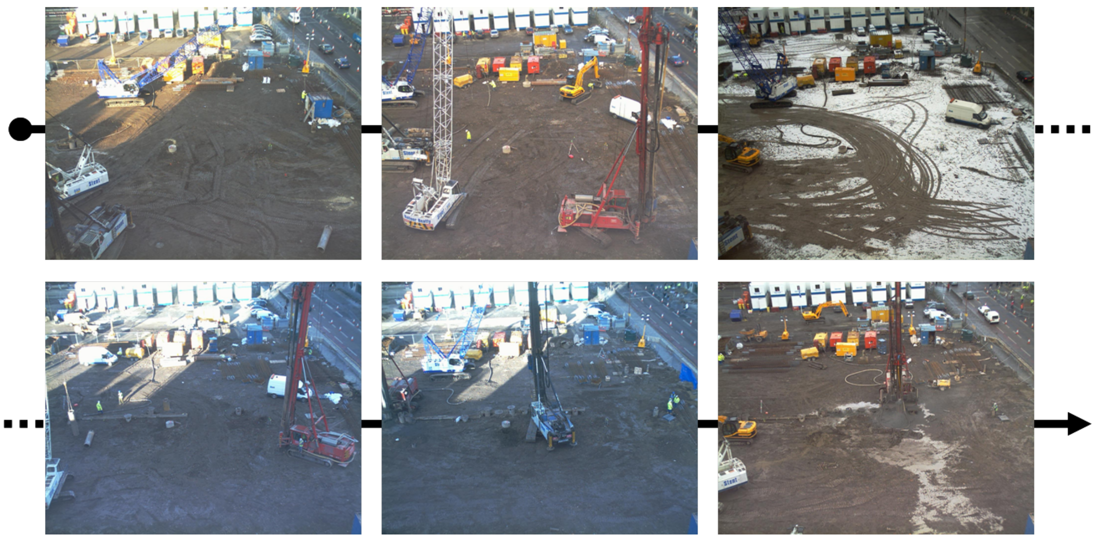


Figure 4.1: Traditional time-lapse video. Segment of 6 frames.

4.2 Deterministic Approaches

The techniques discussed in this section show how to improve time-lapse videos with information from several images. One of the purposes is to do a cleaning of all the elements that can be noise for the observers and that are not real objects of the building. These are machines, people, and weather transitions.

Day image generation is the most computationally expensive part of the processing because it requires processing a set of 18 images to create a single one and this has to be done for each day. In Section 2.4 we clarified that even with a high number of images available we could not take them all because the processing would have been unmanageable. The solution for a faster processing was to take several computers in parallel (8) to run the same process for different days and then combine the resulting data. Day image processing was faster than the first stage so this just required a single processor to produce the final TLV.

4.2.1 Median Filter

The median (m) in probability theory is the number that divides the population of elements in two groups of equal size. It is also the value that minimizes the average of absolute deviations $E(|X-m|)$. The idea with this measure is that given a set of values we could be able to estimate a representative value that should be robust in the presence of outliers. The mean as an arithmetic average of the elements in a dataset is not useful for this case. Outliers in our data are the elements that appear in the set that have small occurrence. For instance, if a person appears in the scene for a short period of time and it is captured by the camera in one or two frames, it is clear that this element is not part of the building and should be discarded from the background estimation.

The concept of median is then applied to image processing. Given a set of images $I_k, \{k=1,2,...,N\}$ we have to find an image Im which is the median of the entire set. The median filter is applied to each pixel's RGB color which is a 3D vector as shown in Equation 4.1. Notice that this is implemented separately for each color channel.

$$Im_{(i,j,c)} = \arg\min_{m} \left(\frac{\sum_{k=1}^{N} \left| x_{(i,j,c)}^{k} - m \right|}{N} \right). \tag{4.1}$$

where N is the number of images, (i,j) is the row and column position of the pixel in the image and (c) the color channel.

This is a pixel by pixel approach and no information from the adjacent pixels is used for the estimation. This method was previously employed in [Ferguson., 2007] and was reproduced to have a comparison point for the evaluation of the results. One of the biggest issues with this method is that each channel is worked independently and this can generate unwanted results such as the emergence of colors that are not part of the dataset because the R, G and B medians can be located in a position different from the data cluster. This is generally true when the number of elements in the data is small and also when the data is concentrated in distant clusters. This effect is called color bleeding [Davies, 2004] because it gives the impression of new colors appearing in the image. In the next section a comparison of this and other methods will be depicted.

4.2.2 Multivariate Median Filter.

This approach starts with the idea that we cannot estimate the median independently for each channel each color because there is a relationship between channels. The concept of the minimization of the average of absolute deviations in the median definition is extended to the multidimensional space by employing the Euclidean distance between the RGB values. The method will always produce a median value within the dataset without the possibility of bleeding and it is also most robust to outliers that taking the mean. The mathematical description of the multivariate median (MVM) is shown in Equation 4.2.

$$Im_{(i,j)} = \underset{M \in \{X\}}{\arg\min} \left(\frac{\sqrt{\sum_{k=1}^{3} (X_{i,j}^{k} - M^{k})^2}}{N} \right) \tag{4.2}$$

An example of applying the method to a set of 18 images from one day is depicted in Figure 4.2. Most of the moving objects, the sun have disappeared and the final image looks clean.

In Figure 4.3 a comparison between the mean, median and multivariate median filters is depicted. A set of 176 [1] pixel color values from a fixed spatial position but different images from the same day is plotted in a RGB space (each point has its true color). In this particular example, the pixel was most of the time capturing the color of the ground floor of the construction and in a shorter period of time a yellow vehicle was parked in the same position (see yellow cluster). The grey cluster has an elongated shape due to the fog which was present during the morning making the ground color

[1] During early stages of the experiments we worked with a larger number of images.

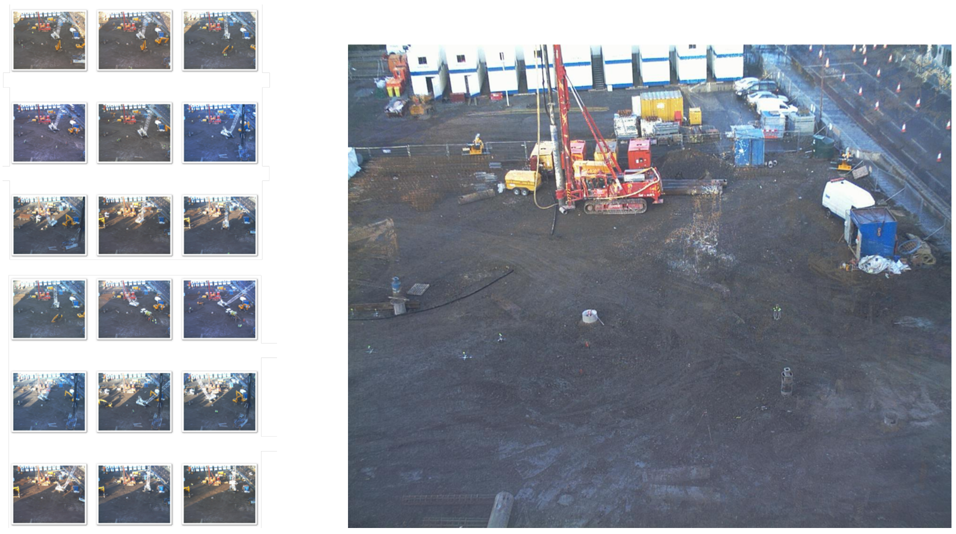
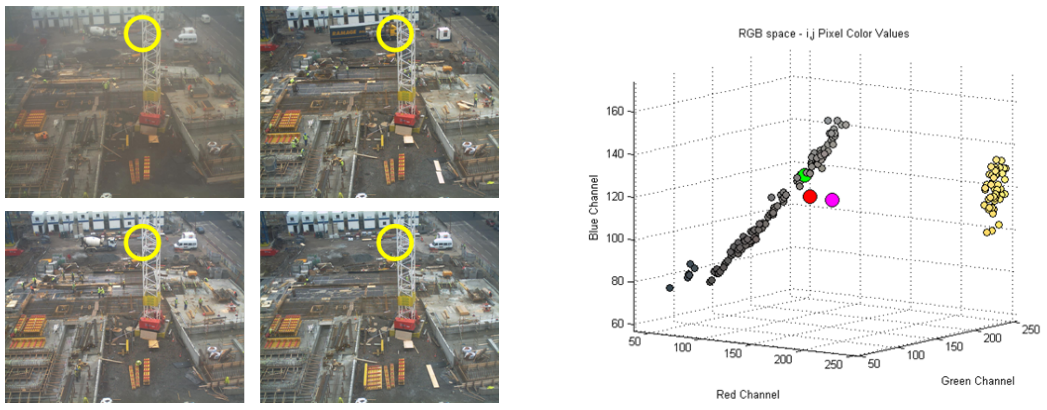

(a) (b)

Figure 4.2: Multivariate median example. (a) Set of 18 input images. (b) Day image after applying the filter.



(a) (b)

Figure 4.3: Comparison between median methods. (a) Sample images from 04-Jul-2006. The circles show the test region. Notice the yellow truck. (b) Color values from sample pixel $(r,c) = (105,711)$. Magenta = mean, red = median, green = multivariate median.

brighter. It is reasonable to think that the best color to represent the cluster is the one in the center of the grey cluster and to not take into account the yellow values, which can be considered in this case as an outlier. The mean, median and multivariate median of this dataset are plotted. Note that the mean has been biased by the yellow cluster the same as the median but in a smaller proportion. On the other hand the multivariate median remains as part from the dataset and it is located in the grey cluster which is close to what we expected.

The same technique was implemented in different color spaces such as LUV which attempts perceptual uniformity of the colors [Comaniciu and Meer, 2002b]. We did not find any particular improvement in these spaces and additionally we found that the method of transforming the image to one space and then taking it back to the original one might induce noise on the images. Finally we preferred to apply MVM in RGB space.

4.2.3 Multivariate Median Results

4.2.3.1 Temporary Foreground Removal

One of the notable improvements of this method over the traditional technique is the complete removal of wandering objects in the scene. This includes: people (pedestrians and workers), construction tools and passing vehicles. The position of these objects is generally changing in the scene and they are considered as outliers on the images. As an example refer to Figure 4.4 where it is shown the multivariate median image of one day when there were many working people. Notice in the results there is no vestige of them.

In general, many of the issues regarding to weather changes in the image as the appearance of sunlight during short periods of time, fog during the first images of the days and some shadows disappeared. Anyhow not all were removed. Especially days when the sun was constantly falling on some region of the image and then moving in a direction during the whole day. These types of problem were not totally removed but reduced in some proportion.

4.2.3.2 Ghosting.

Some of the images had traces of elements that were partially removed from the original images. It was possible to detect some shadows with the shapes of the removed elements. We called this effect ghosting. There were two main reasons for this result.

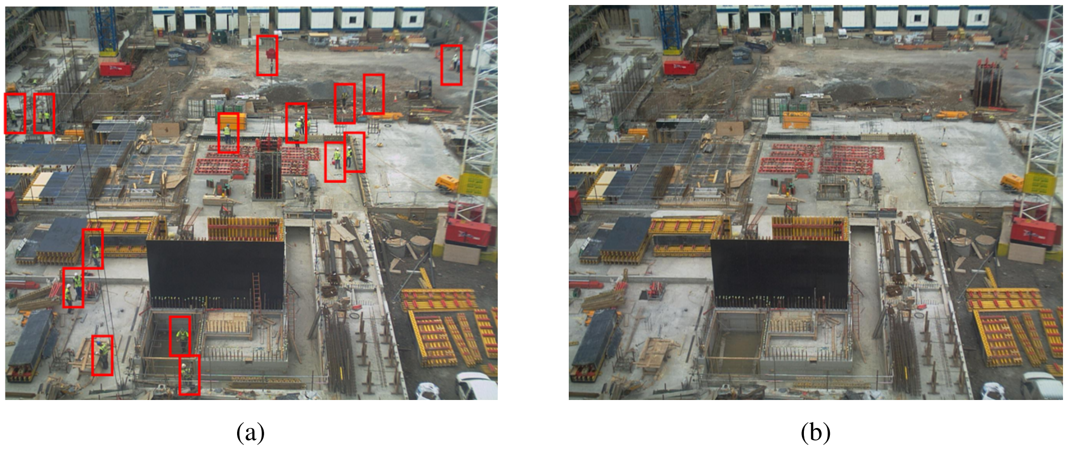


Figure 4.4: Temporary foreground removal. (a) Image Sample with moving objects closed on red squares. (b) Day image after applying multivariate median filter .

One was related with a bias of big outliers when the multivariate median filter was applied and the other was a problem with white balancing.

A big outlier refers to an object that appears in the scene for a considerable number of frames. If, for instance, in a set of 18 images we have an element that appears 7-8 times, then it is very likely that the MVM will choose one element from the biggest cluster but this value will be biased by the other cluster. In the example shown in Figure 4.3 the yellow cluster is affecting the estimation and making the region look brighter than if it was not there. Also an example of the median filters is shown in Figure 4.5. Notice that the ghost is less visible with the MVM estimation than with the median for each channel (there is no bleeding).

The second source of ghosts was the problems with the AWB discussed in Section 3.3 and weather changes. We initially tried to estimate the multivariate median from the raw images and noticed some color changes such as the one depicted in Figure 4.6d. This was mainly because some of the images in the data were unbalanced creating patches of different tones in the same image. Hence it was necessary to solve this before applying the filter (refer Figure 4.6e). The true colors of the images had to be adjusted because otherwise even the regions with no change were going to appear as a cluster with high variance. This problem in combination with real outliers would create more error in the processed images. Figures 4.6a, 4.6b and 4.6c show how the sun appearing in some regions of the image for long periods produced unwanted results.

Another alternative to improve the quality of the results was to increase the number of images from each day, so that the real background elements would appear more

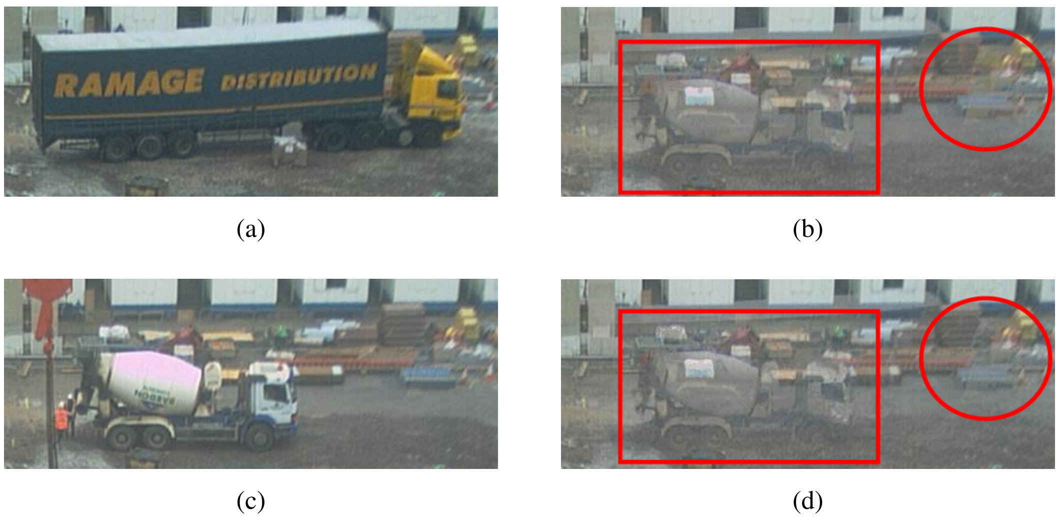


Figure 4.5: Ghosting. (a) and (c) are image segments with vehicles that cause the ghosting. (b) Independent color channel median. (d) Multivariate Median. Bounded regions show the defects. They are less evident in when MVM is used

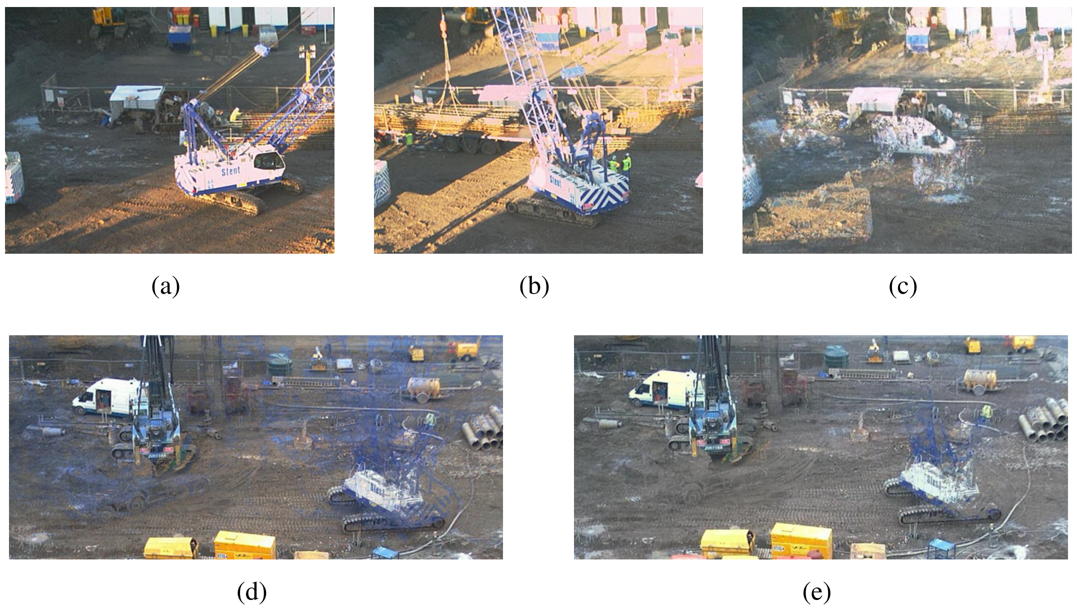


Figure 4.6: Ghosting. (a) and (b) are sample images of a partial sunny day. (c) The results of applying a MVM filter of these images. (d) MVM of images with AWB problems. (e) MVM after color normalization.

often than outliers. This is always linked with the computational cost of the algorithm.

4.2.3.3 Semi-stationary objects

Another problem was in regions of high activity or where vehicles like cranes, which have a stationary body and a flexible arm, create the wrong representation of the background. The results are generally a clear body with an unclear reconstruction of the arm in different locations. These effects can be lessened once the day image processing has been implemented because generally these objects change position from one day to the next, and posterior processing of day images can remove them. Refer Figure 4.7 for examples of these objects.

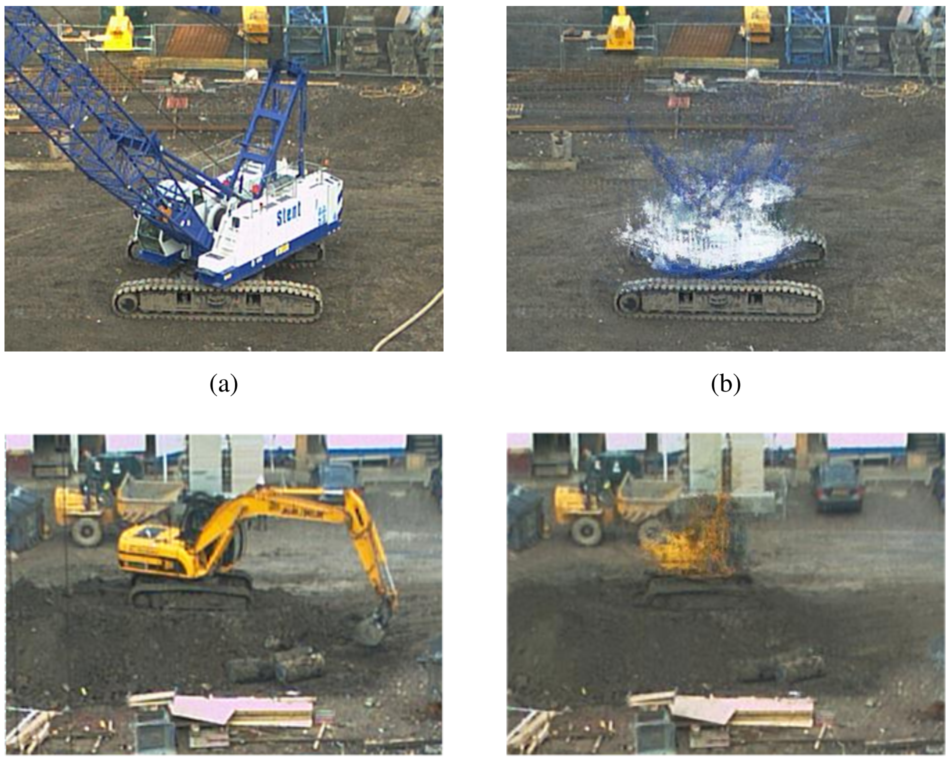

(a) (b) (c) (d)

Figure 4.7: Semi-stationary objects. (a) and (c), Images before processing. (b) and (d) Images after applying MVM

4.2.4 Multivariate Median of Temporally Neighboring images

Once the day images have been created, they are processed based on information of temporally neighboring images. The concept behind this approach is that neighbors can help to find foreground elements from the day images. Then, if there are elements that appear for a short period of time and their color values are distinct from the real background then they can be removed and replaced with the data coming from the same position of adjacent images. For instance, assume that a vehicle is located on the scene for one day. The most reasonable idea would be to remove that element from the scene and fill its space with information coming from nearby images.

A multivariate median filter was applied at each pixel to a subset of images coming from consecutive days. It was required to determine the right number of images to take, the smallest value will be a 3 size neighborhood which was not really good because the multivariate median of only 3 RGB pixel values is not meaningful. This temporal neighborhood size cannot be increased too much because then information from distant times is going to interfere in the estimations and produce a blending effect. We chose a neighborhood size of 5 images for the processing.

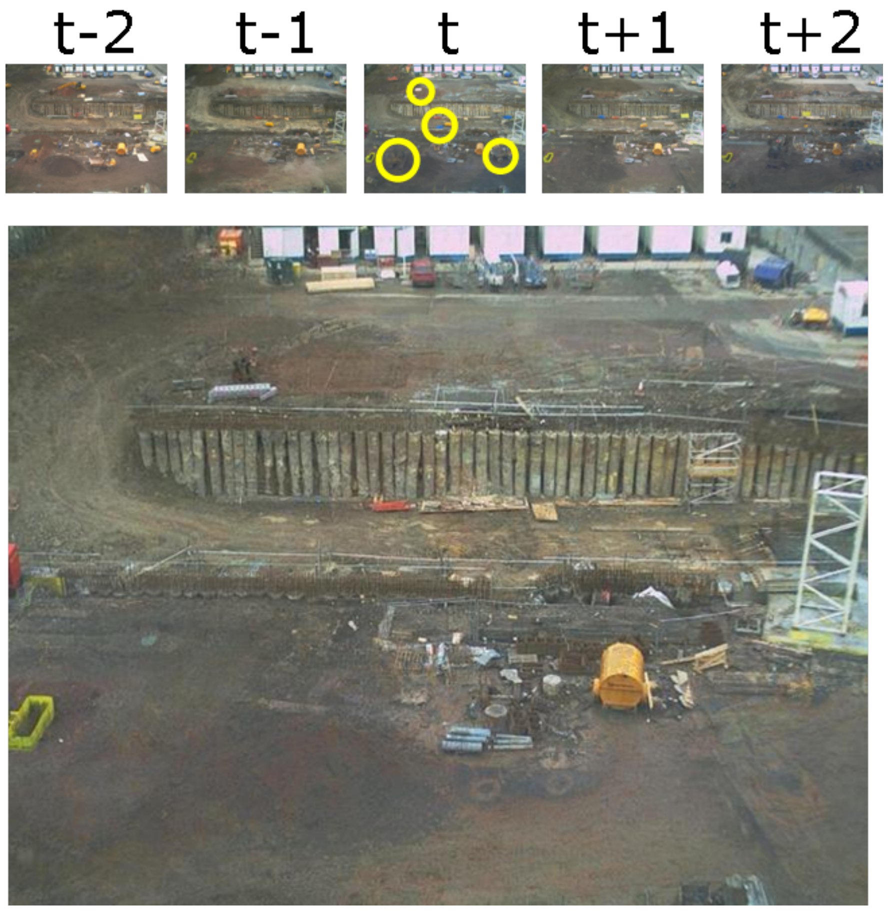


Figure 4.8: Multivariate median of five consecutive days. Note that objects inside yellow circles in the image at time t have been removed.

Figure 4.8 shows how the general approach works. Each image I_k is replaced by the multivariate median of itself and the two previous and two subsequent neighbors. Note that the foreground elements appearing in the image at time t were removed after the filtering. Similarly to the process implemented in the Day image generation, a mean filter would have blended all the frames in an unrealistic way.

Given the set of images $G = \left\{I^{t-2}, I^{t-1}, I^{t}, I^{t+1}, I^{t+2}\right\}$ we can compute the multivariate median image $J_{i,j}$ at time t using Equation 4.3.

$$J_{i,j} = \underset{M \in \left\{G_{i,j}\right\}}{\arg\min} \left(\frac{\left\| I_{i,j}^{t+\delta} - M \right\|}{N} \right) \tag{4.3}$$

where $I_{i,j}^{t+\delta}$ is the RGB color intensity of pixel (i,j) from image I at time $t+\delta$, $\delta = \{-2,-1,0,1,2\}$.

4.3 Probabilistic Methods

This section describes the image processing methods for the generation and processing of day images based on a probabilistic approach. This is divided in two parts. The first one describes a filter based on non-parametric kernel density estimation that is used for creating day images. This exploits the idea of finding the regions with higher density given the image data to find the best representation of a day image. Then a technique for foreground detection and removal is implemented. This uses day images and local information of consecutive frames rather than only pixel color values. A Naive Bayes and a neural network classifier are evaluated in this part of the processing.

4.3.1 Non-parametric Kernel Density Estimation

One of the problems found in the multivariate median was the ghosting. This was due to the bias produced by outliers when they took place for a considerable number of frames. The final effect seemed like a blending of the images. It would be better to choose a value that represents better one of the clusters than an intermediate value of the whole set. In this approach we start with the assumption that the pixels $P_{i,j}$ of a set of images from the same day are generated from a probability density function (pdf). Then, we can find the region with the highest density and then take the value (part of the set) with the maximum value as the best estimate of whether it is foreground or background. This approximately estimates the mode of multivariate data. We are trying to find the value that occurs more frequently in the pdf.

The pixel color values in our images and can be distributed in several forms. Hence, we can not always fit the data to a known shape such as Gaussian, or mixture of Gaussian (in which case we would have to find the number of clusters). For this reason we have employed a non-parametric kernel density estimation (KDE) to produce an empirical description of the data. Each kernel will be generated with all the pixel RGB color values $(x_k,\ k = \{1,2,...,N\})$ at location (i,j) of the N images from the same day. The pdf is constituted by 3D Gaussian distributions $K()$ centered on each of the data samples. The bandwidth (h) of the Gaussians is defined by the 'Rule of Thumb' [Silverman, 1986]. The formula of the probability density function is given in Equation 4.4.

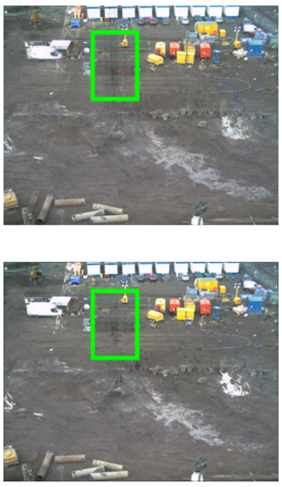
$$p(x) = \frac{1}{N \cdot h} \cdot \sum_{N}^{i=1} K\left(\frac{x - x_i}{h}\right) \tag{4.4}$$

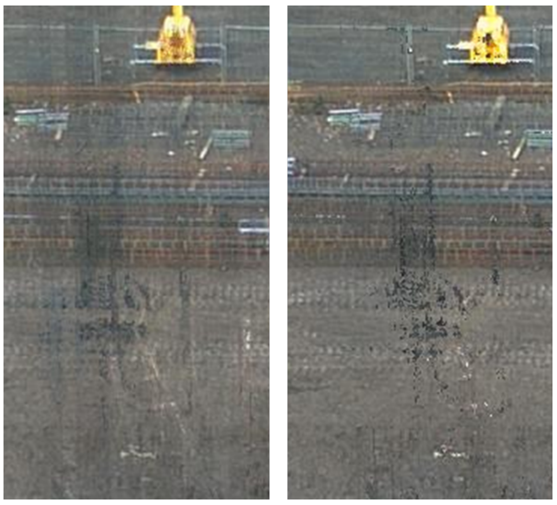
Once the pdf is obtained, we can find the color value with the highest probability density. The computation of the day image $Ik_{(i,j)}$ based on KDE is obtained by applying the following equation:

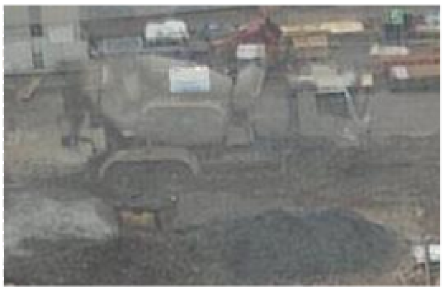
$$Ik_{(i,j)} = \arg\max_{x_k} \left(p_{i,j}(x_k)\right) \tag{4.5}$$

An example of the results after applying non-parametric kernel density estimation in the images is depicted in Figure 4.9. A comparison between the multivariate median filter and the KDE shows how the ghosting problem is reduced. Notice that now it can be seen that some noise appears on the image. This corresponds to the colors of the truck that was located in that position, rather than taken an intermediate color value between the ground and the truck. The filters choose the one that appears more often. There is actually no prior knowledge about what is or what is not background. As a result, elements that might not be the right background can appear in the filtered image.

If the number of images is increased then the accuracy of the results will be higher. Another approach that we tested to solve the problem was to increase not the number of images but the number of elements in the dataset. This was done by including for each pixel of the original dataset, its neighbors (3x3 size). The number of elements of the set changed from 18 to 162 increasing the accuracy of the distribution. The results showed a loss of detail in the images but the ghosting was even less evident. The algorithm implementation utilized functions extracted from the Matlab Kernel Density Estimation KDE toolbox.


(a)


(b)


(c)

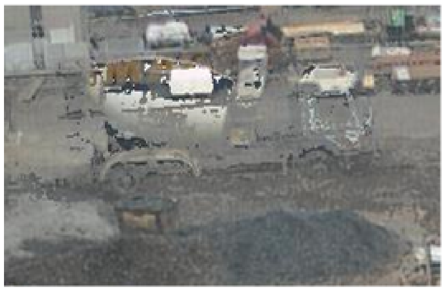

(d)

Figure 4.9: Non-parametric kernel density estimation results. (a) Day images obtained after applying MVM and KDE respectively. (b) Close ups of these images to show the details of the methods. Note that there is no blending with KDE. (c) and (d) Are the results of MVM and KDE of the ghost truck previously evaluated in Section 4.2.3.2.

4.3.2 Neighborhood Classification

By using KDE filter we produced the day images. These still have foreground elements from the scene that can be removed. For instance, elements that remains in the scene for periods longer than one day. In the day images they are identified as background but seen from a sequence of consecutive days, they can be detected as temporary elements. This brings the necessity of working with images of different days to improve single day images. In Section 4.2.4 we utilized 5 days neighborhoods and applied the multivariate median filter in a bounded window. For the probabilistic approach we take just the adjacent images. We will call them I_{t+1}, I_t, I_{t+1}representing the previous, current and next day respectively.

Rather than working with a pixel by pixel approach this method works at a neighborhood level to improve the processing. This gives more information about the data structure. We will work with these images to detect different types of events. Figure 4.10 shows five cases that have been considered to appear in the images. Each circle represents a neighborhood from the image at the three possible times, and a change in color (for instance from white to black) means that neighborhoods have are not similar.

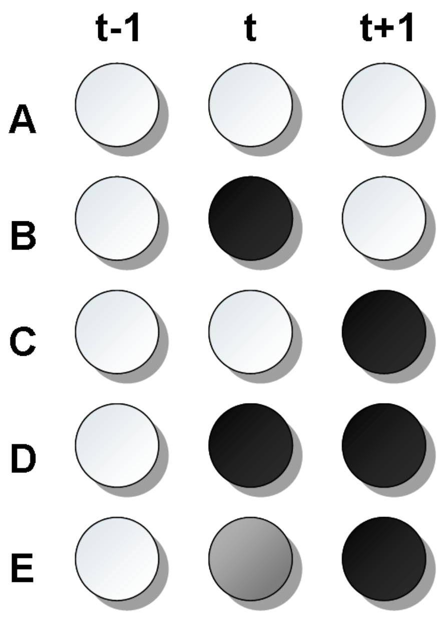


Figure 4.10: Five neighborhood cases at three times.

Case A represents no change in the neighborhood, Case B is when a temporary change occurred at time t. Case C shows that a new event will occur in time t+1 and that a permanent change is over, similarly, case D shows that an event can be probably permanent appears in the last two frames. Case E occurs when each neighborhood is different from the others. These are all the possible cases that can be found in a set of three images. Once we know this we can think about utilizing a model to classify these cases based on local information from the neighborhood.

There are some constraints that are considered for the classification of these events:

- Foreground and background regions have spatial connectedness. This explains that the detection of a pixel as foreground might have some physical connection with other pixels with the same characteristics. Objects in the scene have a minimum size (at least a couple of pixels) and can be bounded in a region, so the detection of isolated pixels in regions where the neighborhoods are classified differently might say that there is a mismatch in the classification. This suggests posterior processing of the classifier results.
- A measure of similarity is required to detect differences between neighborhoods and consequently do the classification. This should be independent of changes such as lighting, wetness, color tone, etc. Some measures of similarity will be discussed in the following section.
- We assume that images are aligned after the jitter removal (Section 3.2) and there is a right correspondence between the pixels of consecutive images.
- The images have been preprocessed and also the color levels have been normalized using the grey world assumption.
- The removed FG detection region can be slightly larger than the original foreground but this will not be harmful for the image quality because the spaces will be filled with background information from neighboring images. These changes in the size can be due to procedures such as dilation and erosion with the resulting data.

4.3.2.1 Naive Bayes Classifier

We defined the inputs as the neighborhoods at three different times and the output of the model which were the five possible cases. We can adapt the approach as a classification problem. A Naive Bayes classifier is the first approach implemented to solve it. This supervised learning method makes strong independence assumptions of the cases given the observations that in our case are a measure of similarity between the neighborhoods. It will be shown later that this simplified approach can be useful for the foreground detection.

Initially it is required to determine the apriori probabilities for the five cases. They were estimated manually by taking samples from the day images database and finding the frequency of appearance of each case. Experimentally we have obtained the following values: $P(A) = 0.37, P(B) = 0.15, P(C) = 0.14, P(D) = 0.13, P(E) = 0.21$. These five probabilities must sum up to one.

Now lets define $s(A,B)$ as a measure of similarity between consecutive frames, where A and B are neighborhoods with RGB color data. Let V_{t-1}, V_t and V_{t+1} be the neighborhoods at three different times. We can measure the similarity between pairs of them as:

$$\begin{aligned} d_{-0} &= s(V_{t-1}, V_t) \\ d_{0+} &= s(V_t, V_{t+1}) \\ d_{-+} &= s(V_{t-1}, V_{t+1}) \end{aligned} \tag{4.6}$$

We can estimate $p(X \mid d_{-0}, d_{0+}, d_{-+})$, the posterior probability of a class $X \in \{A,B,C,D,E\}$ given the observations d_{ij} by Bayes Rule:

$$p(X \mid d_{-0}, d_{0+}, d_{-+}) = \frac{p(d_{-0}, d_{0+}, d_{-+} \mid X) \cdot p(X)}{p(d_{-0}, d_{0+}, d_{-+})} \tag{4.7}$$

The denominator (evidence) can be ignored because it is common for all cases. The resulting case for each sample is the one with the highest posterior probability. To evaluate the likelihood we can use two models. One assumes that there is full independence between the measures of similarity given the observation and other that assumes high dependence. We have evaluated the first option. This is described in Equation 4.8.

$$p(d_{-0}, d_{0+}, d_{-+} \mid X) = p(d_{-0} \mid X) p(d_{0+} \mid X) p(d_{-+} \mid X) \tag{4.8}$$

Finally according to the five cases. Only case B can be considered as the detection of a foreground on the scene. Let Ω be all causal factors (observations) then the classification condition is: if $p(B \mid \Omega) > p(A \vee C \vee D \vee E \mid \Omega) = p(A \mid \Omega) + p(C \mid \Omega) + p(D \mid \Omega) + p(E \mid \Omega)$ then the pixel can be classified as foreground FG.

4.3.2.2 Cross-correlation

The model previously described can be implemented by utilizing different types of similarity measures between neighborhoods $d(A,B)$. Some of them might be more suitable for this particular application. Some common approaches are the Euclidean distance between images, mean squared error and histogram-based distances [Wang et al., 2000] and cross-correlation. A suitable measure for this research should be perceptually meaningful and independent of changes such as lightness and small shifts. For instance, the Euclidean distance performs poorly if it is applied in a shifted image such as the ones that we have due to the jitter problem. These requirements are needed because the data was classified manually and is very likely that the similarities found by the observer could not be really similar for a particular method, In addition, One of the main purposes of this algorithm is to create a reasonable representation for the observer who is eventually the final evaluator of the system performance.

Statistical cross-correlation which was also utilized for the jitter removal will be used as a measure of similarity between image neighborhoods. Its calculation is shown in Equation 4.9. We keep calculating it as the average of the cross-correlation of the three color channels. A second measure of similarity will be described in the next section.

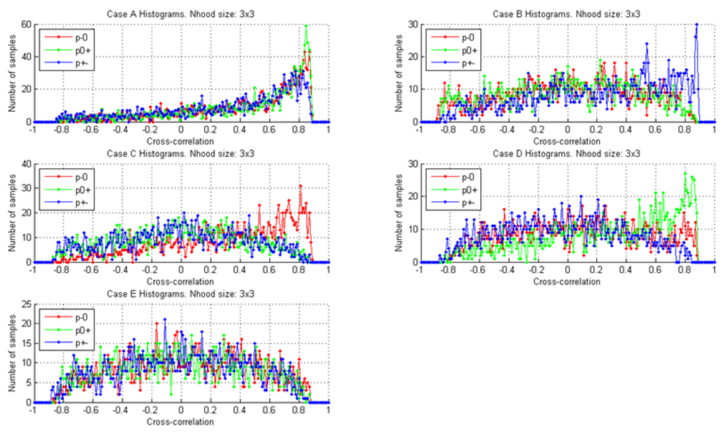
Given two image neighborhoods A and B of size NxN,

$$ccor(A,B) = \frac{1}{3 \cdot N^2} \cdot \sum_{c \in \{r,g,b\}} \sum_{(i,j) \in A} \frac{\left(A_{i,j,c} - \mu_c^A\right)}{\sigma_c^A} \frac{\left(B_{i,j,c} - \mu_c^B\right)}{\sigma_c^B} \tag{4.9}$$

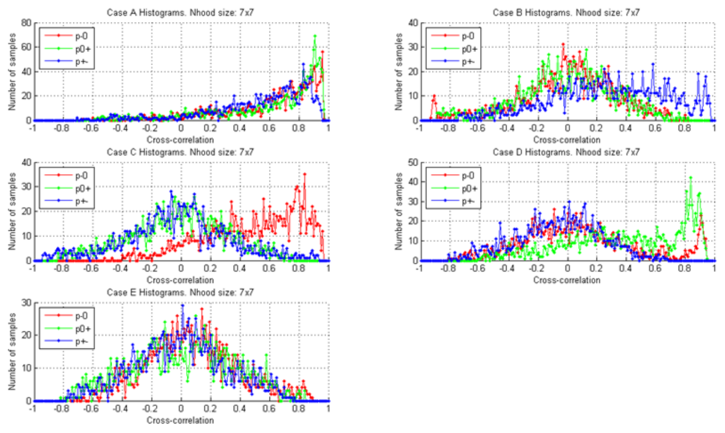
where μ_c^A and σ_c^A are the mean and standard deviation of image A in color channel c.

The definition of cross-correlation is shown in Equation 3.1. There is no change with the original approach except that the input now is different. Hence, $s(A,B) = ccor(A,B) = \rho_{AB}$. Remember that $ccor$ is highly independent of lightness and useful for matching objects with similar texture. Something that is appropriate for our experiments.

Training procedure: To create a model of the data we utilized images from the dataset to extract samples with the five cases. Each one consisted of three consecutive day images neighborhoods, its associated case A,B,C,D,E and the location (row and column) of the sample in the image. Then the cross correlation was estimated for image pairs ρ_{-0}, ρ_{0+} and ρ_{-+} for each sample with different neighborhood sizes (from 3x3 to 21x21). Additional samples were automatically generated by assuming that neighboring pixels to the ones already selected were the same case (spatial connectedness). Finally the cross-correlation histograms for each case and each neighborhood pair were calculated. The results are depicted in Figure 4.11. Each plot is the histogram for different neighborhood size and the three colors are the possible pairs.



(a)



(b)

Figure 4.11: Cross correlation histograms for 5 cases (A,B,C,D,E) and two different neighborhood sizes. (a) 3x3 window. (b) 7x7 window.

Note that the smoothing of the histogram increases with the neighborhood size. For small sizes such as 3x3 the histogram shapes are very unclear and it is not possible to see a real relationship between the data values but when size increases it starts to get better. The shapes of the histograms can be approximated to a probability density function. They seem to be smooth enough and also seem to give some information about the behavior or the data.

The plots suggest that a pdf that can be used to represent the data distribution on the histograms. Note that most of the distributions are bell shaped and the data is concentrated near zero similar to a normal distribution. This occurs with all the similarity measures between different neighborhoods. On the other hand, when the neighborhoods are similar, the histogram shape does look skewed to the right. It is clear that the maximum value that can be obtained from the cross correlation is one. According to this we have decided to fit the data to an exponential distribution as the shown in Equation 4.10. A gamma distribution was also taken into account but this distribution has a small value near to the edge (near to 1) which makes it unrealistic because the region of highest correlation has to be considered and can not have a low density.

$$p(x) = \begin{cases} \lambda e^{-\lambda(1-x)} & , x \leq 1, \\ 0 & , x > 1. \end{cases} \tag{4.10}$$

Considering all these factors, we have two types of distributions. Gaussians for the ones where the neighborhoods are uncorrelated and exponential when there is high correlation. Figure 4.12 shows the estimated pdfs for all the cases after fitting the data. The estimation of these elements was obtained by maximum likelihood estimation.

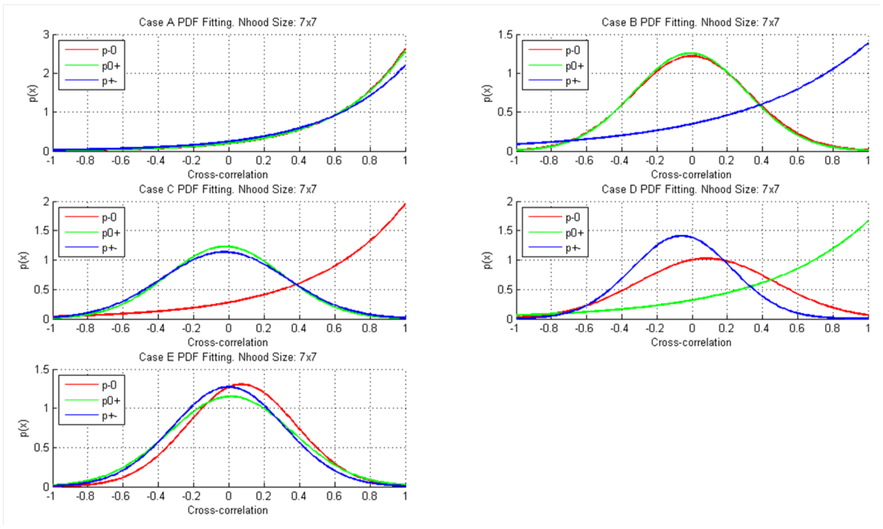


Figure 4.12: Estimated probability density functions for 7x7 neighborhoods.

Notice that the ρ_{-+} histogram for the B case is not as well defined as the other cases where the cross-correlation is high. It seems that the shape is not very clear and that does not fit correctly an exponential distribution. There are some reasons regarding these results. The selection of samples of B case was not as easy as other types of data, so it required more time to find the appropriate samples. As was mentioned, these were selected manually and the considerations are biased by human perception. It seems that some of the selections might have been incorrect. Additionally it is very likely that when foreground was located on the scene, it modified the scene itself. Such as vehicles when they were parked in the ground modifying the terrain and leaving the imprints of tires.

The cross-correlation method worked well for samples of case A. The three $ccor()$ shapes matched and the curves are clearly smooth. Something similar happened with C and D cases. This is likely because it was easier to detect the same attribute on the image of consecutive images than when an object was in the middle. Case E histograms show correctly the lack of correlation between neighborhoods and examples of this were very easy to find on the dataset.

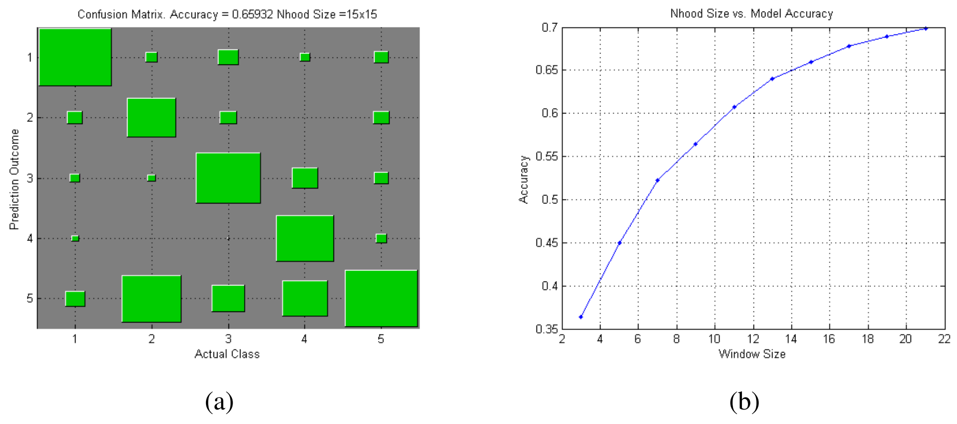
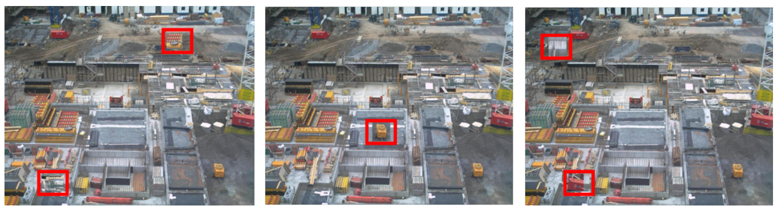


Figure 4.13: Naive Bayes Classifier. (a) Hinton graph of confusion matrix for a 15x15 window. (b) Accuracy of classification according to the neighborhood size.

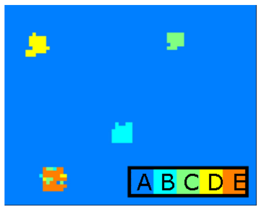
The accuracy of the classification was estimated using a test set. This was created in a similar manner to the training data by taking neighborhood samples from the image database and their associated case. The number of elements was equal to 3750. 150 of these were obtained manually and the rest were automatically generated assuming that nearby pixels had the same class (5x5 Neighborhood). The results show that the method employed is not the best approach for this type of data. Figure 4.13a shows the confusion matrix of the classification for a 7x7 window. Notice that the highest number of misclassification occurred with class E. Figure 4.13b shows how the accuracy increases with the neighborhood size. This means that this similarity measure is not suitable for small windows. Furthermore, we can not either take large neighborhood sizes because the local information will be lost and combined with other elements of the scene.

Once the pdfs that define the likelihoods were obtained it was possible to test the model. It was done using real images and also using synthetic images. Synthetic images were generated by taking a real image from the data and then modifying it by inserting and removing objects using an image editor, so it was possible to create samples of all the five available cases.

Figure 4.14 shows an example of the results of the implemented model on synthetic data. The first part is the original image which was modified in the parts indicated by the squares. The second image shows the results according to the classification; the color table is also shown. Notice that the results match the five cases properly. Case E is the one that seems more unclear because it has some errors in the classification confirming the results of the confusion matrix.



(a)



(b)

Figure 4.14: Naive Bayes Classifier tested on synthetic data. (a) Sample Image at three different times. Red boxes show the edited regions to generate the five cases. (b) Classification results. Color map is shown in bottom right corner.

After experimenting with real data we found that the results were really noisy and many misclassifications were obtained. Figure 4.15 shows the results of a series of images which have been labeled according to the five cases (same colormap as previous Figure). Notice the frames where the snow appears, the classification is able to detect it as foreground. This is not accurate enough to be applied to the entire video. The results suggested that a different classification method could be evaluated. This will be described in next section.

Some of the problems found with the cross-correlation were:

$ccor()$ is independent to lightness but it is also independent to color. This occurs because the color information is partially lost when the cross-correlations of each channel are averaged.

The estimation of the cross correlation requires normalization of all the images. This divides the pixel values by the standard deviation of the neighborhood. And because we were working with very small neighborhoods, some images had the same

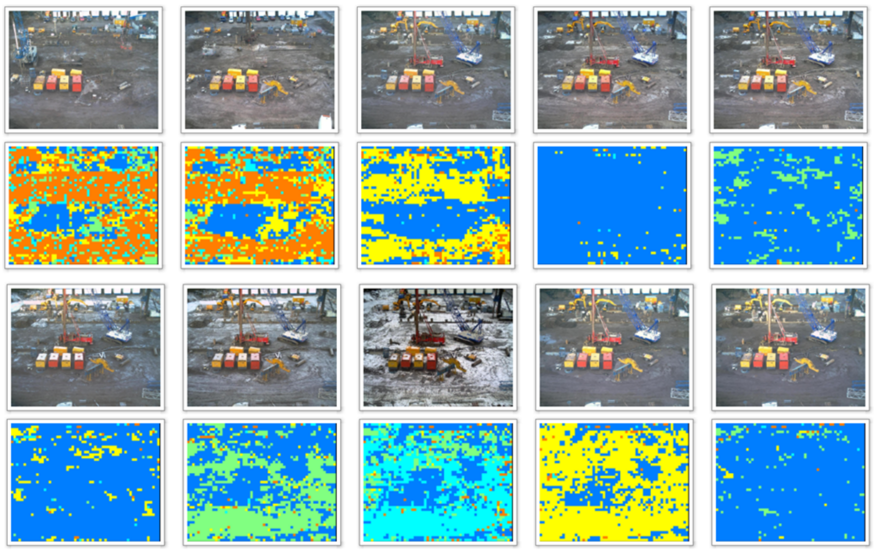


Figure 4.15: Naive Bayes classifier in real data. An example of an inaccurate classification can be seen in the first three images. The orange region which represents class E is incorrectly predicted because the ground of the scene did not change. However the images in the second row, show how the model detects properly the appearance of snow as class B (light blue) in the third image. One day before, these regions are classified as class C and the fourth day as D.

color value in all pixels producing neighborhoods with zero std which do not allow normalization. Examples of this are saturated regions where the sun is pointing and flat elements such as truck containers. A provisional solution was to take these cases only shift them to zero mean only and do not divide by the std. In any case, this showed that for an appropriate cross correlation estimation it is required to have images with high std and size. This was certainly true for the images from the dataset for the jitter estimation but not in this case.

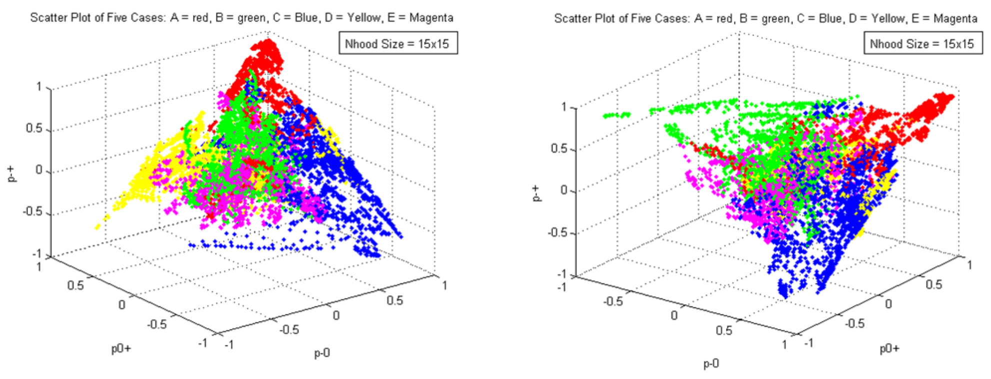
4.3.2.3 Neural Network Classifier

A different approach to solve the classification problem was to implement a neural network. It is likely that the distribution of the similarity measures can not be accurately represented as a mixture of Gaussian and exponential distributions. Hence the implementation of a Naive Bayes model might give unwanted results. It was possible to visualize the data in a 3D space to have an idea of the shapes of the cluster of each case. In Figure 4.16 this can be seen. Notice that there are 5 clusters, each one is represented by a different color. There are also some data samples that seem located

Neural Network	
Input type	$\rho_{-0}, \rho_{0+}, \rho_{-+}$
Image Window size	15x15
Number of inputs	3
Number of hidden units	4
Number of outputs	5
Hidden units activation function	$tanh()$
Output units activation function	$softmax$
Regularization term	6
Test data accuracy	0.6932

Table 4.1: Neural network specifications for 5 neighborhood cases classification

in the wrong place and mixed with incorrect classes. The central cluster is case E and is the one with the highest interference of other classes. This might explain why the worst classification was obtained with case E when it was tried with the synthetic data. Nevertheless the scatter plot suggests that a neural network approach might be suitable for the classification of the five cases.



(a) (b)

Figure 4.16: 3D scatter plot of 5 cases samples from two different views.

The implementation of this method consisted of a neural network with three inputs, each one corresponding to the similarity measure between neighborhood pairs. A hidden layer of 4 units which was experimentally estimated (this procedure will be described below) and five outputs each one corresponding to one of the five classes. Table 4.1 shows the specifications of the neural network.

We utilized the same training data from the Naive Bayes approach. The activation function of the output units is a softmax where the sum of all outputs must sum up to 1 and the one with highest value defines the class. The activation function of the hidden units is a sigmoid function. The procedure of estimating the right parameters of the neural network is described as follows:

- The cross-correlation of neighborhood pairs is a 3D vector input of the neural network.
- Each sample have an associated target $\{A, B, C, D, E\}$. (5 outputs)
- 50% of the data was utilized for training, 50% for validation and regularization term estimation.
- The weights of the NN were optimized using the Netlab toolbox for Matlab [513, 2002]. The optimization method was scaled conjugate gradients.
- Different networks configuration were tested. The number of hidden layers, the random seed and the regularization term were modified to obtain the least sum of squares error. Early stopping was not considered in the training, hence the number of cycles was enough to guarantee convergence (200).

The confusion matrix for the test data is depicted in Figure 4.17. It can be seen that the number of misclassifications is reduced when compared to the first method (Figure 4.13a) but most of the inaccuracies still occur in case E.

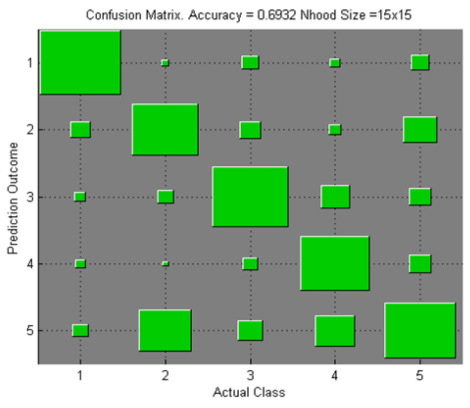


Figure 4.17: Hinton graph of confusion matrix for a 15x15 window using a Neural network as a predictor

The main goal of having 5 different cases was to be capable to replace the unwanted elements with the most probable background information. It is clear that all elements of size B were going to be removed form the image dataset, but then the resulting empty spaces had to be filled with some data. We decided to replace these spaces according to the classification of the surrounding areas of the foreground. For instance, if a foreground element was next of a case C region, then it made sense to replace the holes with data from time $t-1$. A similar idea was with neighborhoods of type D.

The results showed that a neural network approach was a better method to solve the classification problem than the Naive Bayes classifier. As a different alternative, we decided that the process could be focused only on the detection of elements of case B and assume all other cases as B'. We needed to build again the NN model with two outputs. We also changed the similarity measure because we had found some difficulties with the cross-correlation such as the normalization, neighborhood size and perceptual similarity. This measure is herewith explained and then the final model of foreground removal using two classes is depicted the following section.

4.3.2.3.1 Euclidean distance of scaled neighborhoods in Principal Component space Another suitable similarity measure which is perceptually meaningful and is not excessively expensive in terms of computational cost is the Euclidean distance of scaled neighborhoods in Principal Component space. It is a variant of the technique proposed in [Li et al., 2005]. This process consists of a combination of image resampling, a PCA transformation and Euclidean distance metric. The general process is described as follows:

- Take neighborhood samples and scale them to a smaller resolution. For instance, take 15x15 neighborhoods and rescale them to 3x3. Each new pixel is the result of averaging the values of a 5x5 window from the sample. This process reduces local noise and variance. In our case, the jitter problem does not affect this estimation (something that can be critical in the cross-correlation for small neighborhoods).

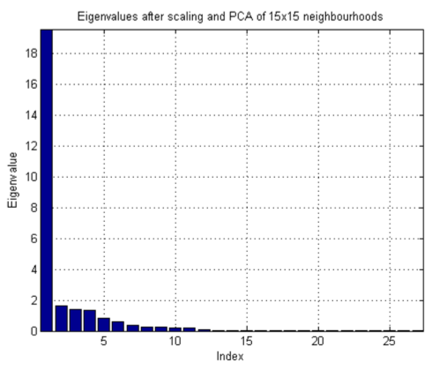
- The resulting elements are 27 dimensional vectors. This corresponds to (3x3x3), the window size and the three color channels which are considered separately. This vector is then transformed by PCA to 10 dimensions which correspond to the highest principal components of the covariance matrix of the data.

- Finally the Euclidean distance of the vectors for each transformed neighborhood is calculated. This similarity equation for a given neighborhood pair is given in Equation 4.11 taken from [Li et al., 2005].

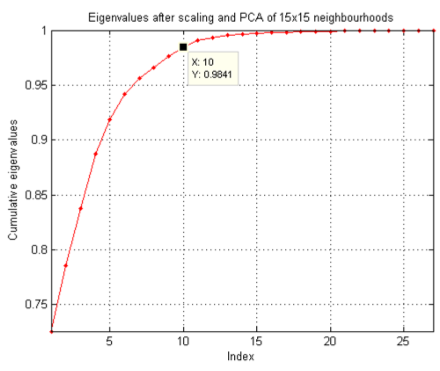
$$s(A,B) = \|T(D(A)) - T(D(B))\| \tag{4.11}$$

where D is the image resampling of the neighborhood to a smaller size and T is the PCA transformation.

The PCA transformation matrix was calculated by taking sample neighborhoods from the whole dataset. For each day, two images were selected and 10 neighborhood samples were chosen. This is equivalent to 12,000 samples. Hence we had enough data to find the appropriate transformation. The data was normalized to zero mean and one std before finding the covariance matrix. Then the eigenvectors were found and arranged according to their eigenvalues. Figure 4.18 shows the curve of the eigenvalues found. Note that most of the energy of the data is concentrated in the first 10 eigenvalues (98%). These were the ones selected for the transformation. Note that the first eigenvalue is much higher than the rest of the values. Variations of this value produce in reconstructed data samples from 10-dimensional eigenspace to a 3x3 RGB neighborhoods changes in color intensity. The cross-correlation is not very sensitive to changes in this property and this could probably explain the reasons of having poor results in the first approach.



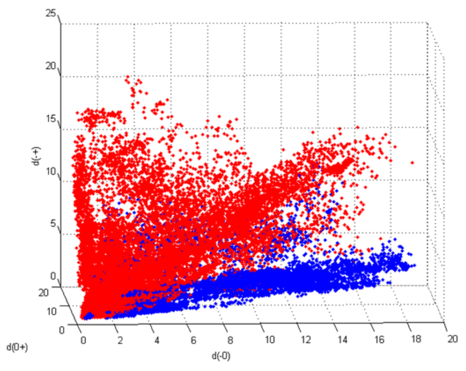
(a)



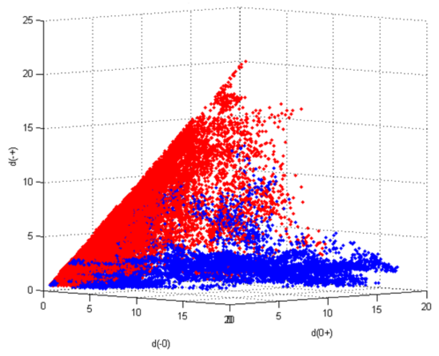
(b)

Figure 4.18: Eigenvalues of image neighborhoods after scaling and PCA. (a) Classified in decreasing order. (b) Cumulative graph.

We used a Similar procedure to visualize the distribution of the data like with the cross-correlation method. Notice in Figure 4.19 that two clusters representing each class are clearly defined and have only a small overlapping region. This shows that some other features of the data were captured with this metric that were not with the $ccor$.



(a)



(b)

Figure 4.19: Distribution of foreground and background samples using Euclidean distance of scaled neighborhoods. The blue cluster is constituted by all the samples classified as foreground (class B) and the red cluster represents the background.

4.3.3 Foreground Detection and Removal using Temporally Neighboring images

Foreground elements in image at time t can be detected given the information from the previous and subsequent images. This approach works at neighborhood level and the measure of similarity used is the Euclidean distance in PCA space as described in the previous section. Each pixel is classified as background or foreground. If the latter is true, then the pixel values are replaced by the ones from the image at time $t-1$. This decision was heuristically made assuming that any of the consecutive frames could be used to replace the foreground and the video frame rate was fast enough to make this operation undetectable for the observer.

A neural network was trained to solve the classification problem. The training and test data was the same utilized with the cross-correlation measure. Its difference was that the cases different to class B were included in a single class (B'). The training process was similar to the one explained in Section 4.3.2.3. The number of outputs changed to two (BG and FG) and the number of hidden units was equal to three. Table 4.2 shows the final configuration of the neural network and Figure 4.20 shows the confusion matrix for the test data. The accuracy of the best neural network was equal to 0.8902, which is much higher that the values obtained with the first method. This improvement is related with the reduction of number of cases and the utilization of the new measure of similarity.

The model was tested in real data. The example in Figure 4.21 show its operation:

Neural Network	
Input type	s_{-0}, s_{0+}, s_{-+}
Image Window size	15x15
Number of inputs	3
Number of hidden units	3
Number of outputs	2
Hidden units activation function	$tanh()$
Output units activation function	$softmax$
Regularization term	10
Test data accuracy	0.8902

Table 4.2: Neural network specifications for foreground and background detection.

A binary image is the result of the classification of each image pixel. Pixel values equal to 1 (white) are foreground and 0 (black) are background. Assuming that foreground pixels are spatially connected we process the binary image to remove small regions which might be wrong classifications. The image is cleaned up using two morphological operations: erosion to remove small regions and dilation to expand the foreground and connect nearby pixels. The final step is the infilling of the foreground regions with the data of the previous day. Note in the figure example that a large proportion of the white crane appearing in the image at time t is removed after this process. However some of the parts of the blue arm were not detected.

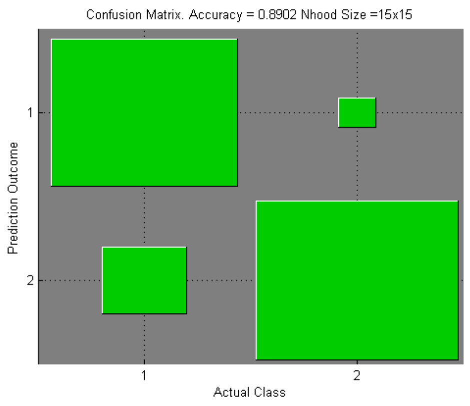


Figure 4.20: Hinton graph of confusion matrix for foreground and background detection. 1 is equivalent to foreground and 2 background.

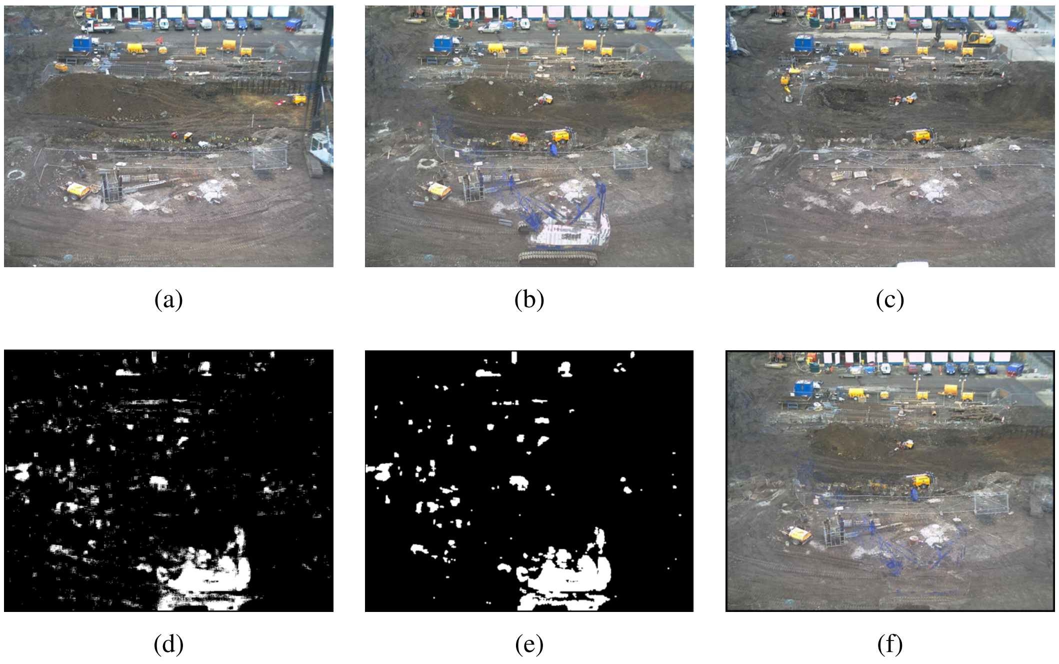


Figure 4.21: Foreground Detection and Removal using Temporally Neighboring images. (a), (b) and (c) are sample day images at times $t-1$, t and $t+1$ respectively. (d) is the BG and FG classification image at time t. (e) is obtained after a processing previous image to remove small foreground regions and noise. (f) is the result of replacing foreground elements in image t with the values taken from image $t-1$.

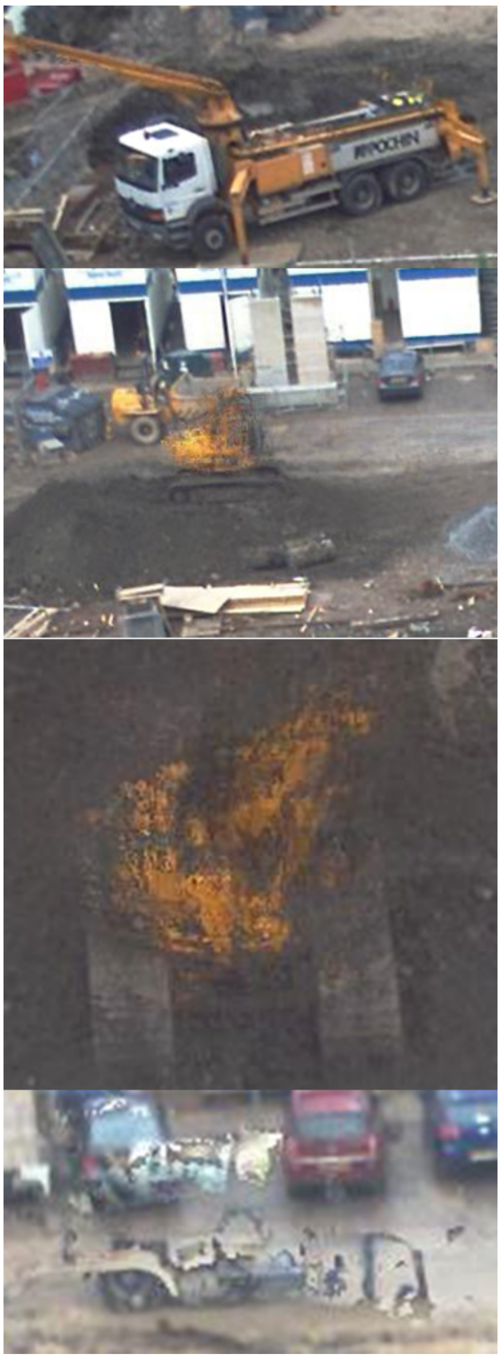
4.4 Illustrative Comparison

One difficulty in the evaluation of the video outcome was that we did not have a real background of the scene. This is an image that represents only the building without considering other elements. This image should also change constantly according to the project evolution. We tested the creation of background using segments selected manually from different images and placing them together with a graphics editing program (Photoshop) but the results were not really appropriate for the comparison. Another possible solution was to create artificial scenarios in laboratory conditions with a variety of elements, lighting conditions and capture devices similar to the ones used in the project. However it was not easy to create such variety of conditions to match real situations.

All the produced time-lapse videos are available online at [Reyes and Fisher, 2008]. The videos and processed images were utilized to evaluate and compare the performance of the methods. Our ability to understand visual images and videos was used to make a better assessment of the results rather than taking hand made background images that were not representing accurately the background.

The methods utilized in the processing stage received as input the preprocessed images. Both methods, the probabilistic and the deterministic, performed better than the traditional approach in terms of a reduction of artifacts in the scene. It was found that the most relevant problem with the MVM filter was the ghosting which is related to the presence of a high number of outliers in the data and the reduced number of elements in the set. Enlarged regions of images show the differences between both methods in Figure 4.22. The samples were taken from day images after implementing MVM and KDE. Notice that foreground objects seem blended with the building structure in the first case. In contrast, with the kernel density method the samples look like spots with sharp edges that separate the foreground with the real scene background. The reason for this behavior is that this last model does not have prior knowledge about the characteristics of the data and it only chooses elements that appear more often in the image whether it is background or foreground.

Figure 4.23 shows the stages of the multivariate median method for three consecutive days. The first row show preprocessed images. The second row shows the result of processing 18 images from one day to obtain the background representation (day image). The last process applied was the MVM of temporally neighboring images with span equal to five days. Similarly, Figure 4.24 depicts the probabilistic approach.


(a)

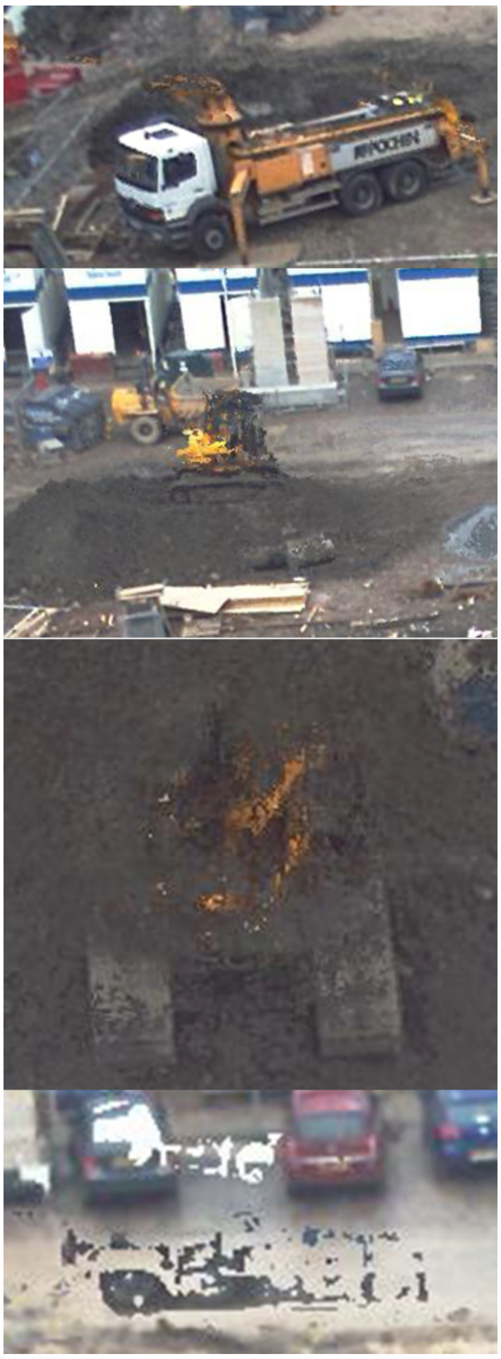

(b)

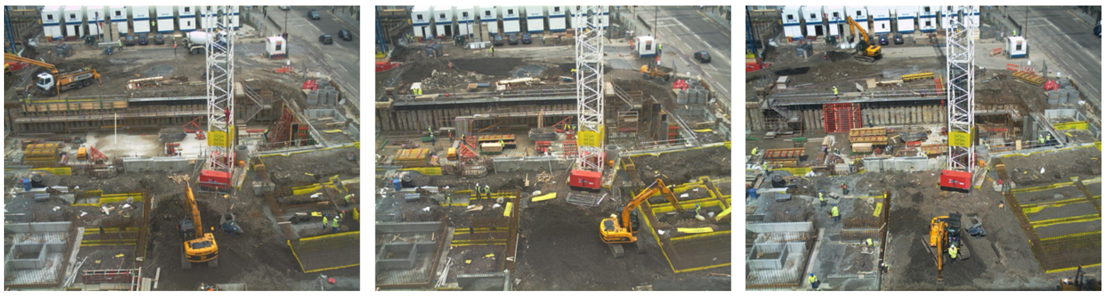
Figure 4.22: Enlarged day image segments for comparing processing methods. (a) Multivariate median filter. (b) Non-parametric Kernel density estimation filter.

Initially the kernel density estimation filter is applied to create day images, then the foreground detection and removal using neighboring images method is computed. Finally, the last frames are created using MVM of neighboring images. This last step was added to have a comparison point with the deterministic method. The results show that both techniques produced time-lapse videos perceptually better than the traditional approach and that can be useful in the processing of cluttered consecutive images. Further improvements in the results could be done if more assumptions about the source of the data are considered. An example of this is the weather prediction in each image. This will be described in section 5.2.1.

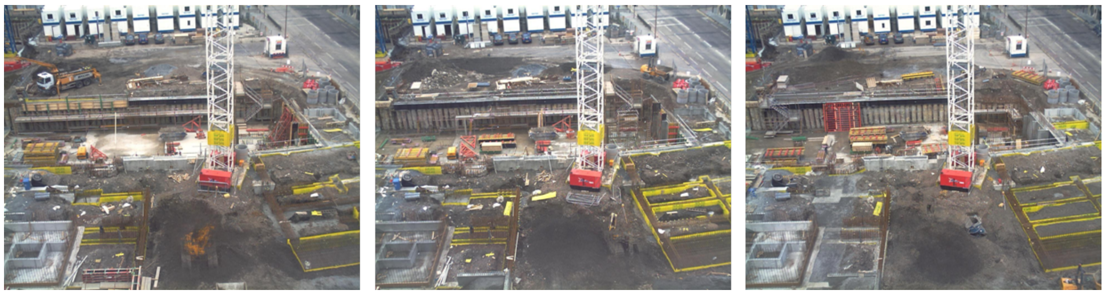
Depending on the application, it might be preferable to utilize a Kernel density estimation filter rather than a multivariate median filter. For instance, The foreground detection method would perform poorly if we use the multivariate median filter. KDE makes more visible the differences between background and foreground. Hence if we need to compare neighborhoods of consecutive days using any of the similarity measures (e.g. cross-correlation), it is better to find a high or low differences between them rather than intermediate values which are uncertain. Ghosting is not easily detected with our similarity measures because the color values in those regions are an intermediate color between the dataset values. This can create confusion between an pixel being background or foreground. Then KDE facilitates the detection of foreground when applied before the foreground detector using three consecutive frames.

4.5 Video Rendering

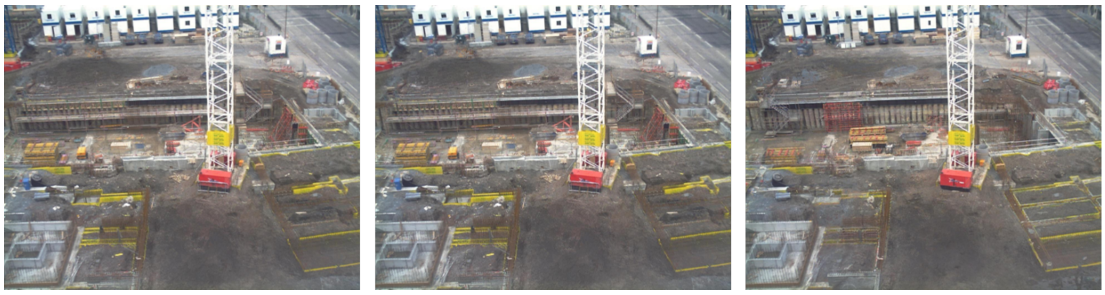
The last stage of the process is the generation of the video files using the frames produced after the processing. It was required to define some of their characteristics such as the number of frames per second, the resolution and the compression. These elements had to be considered for the rendering. The number of frames of our videos was given by the number of days of construction. Rather than choosing the frame rate (e.g. 25 fps), we defined the playback time. This was 30 seconds. In addition, we chose a standard resolution equal to 720x480 pixels of the images and Windows Media Video (WMV) as the compressed video file format. Table 4.3 shows the most relevant characteristics of the time-lapse videos.



(a)



(b)

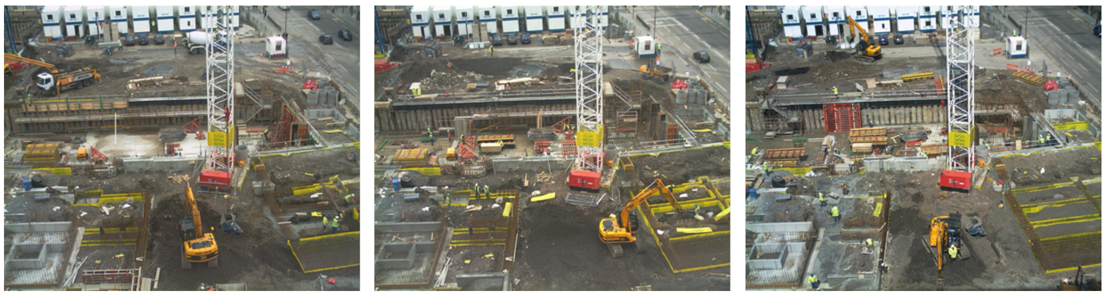


(c)

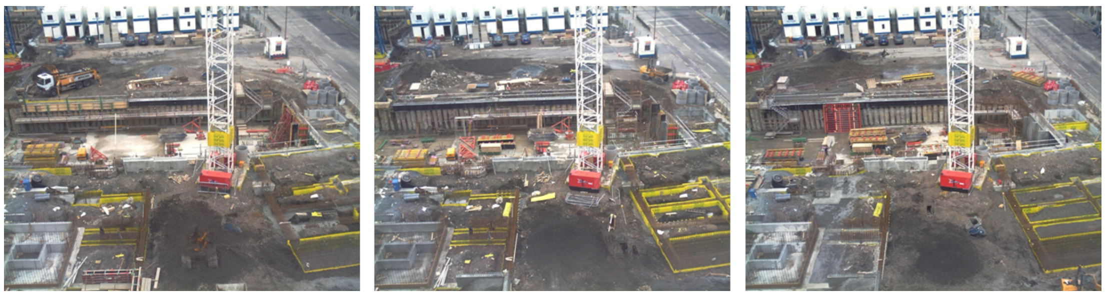
Figure 4.23: Deterministic processing approach. Sample of three consecutive days. (a) Raw Images. (b) Multivariate median filter. (c) Multivariate Median of Temporally Neighboring images.

Video Characteristics	
Frame rate	15 fps
Resolution	720x480
Compression Format	WMV

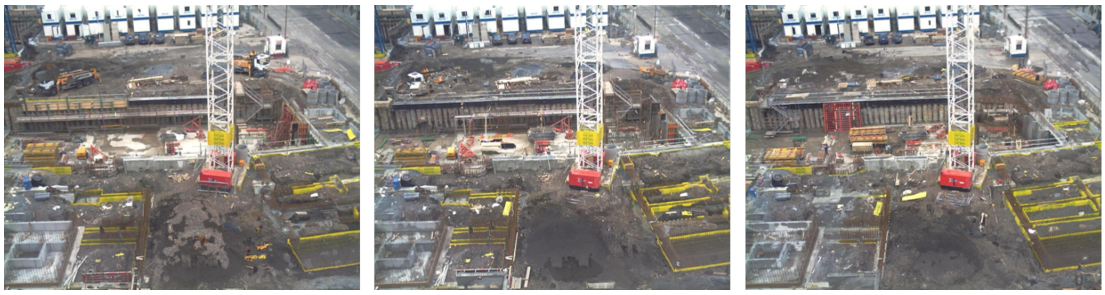
Table 4.3: Time-lapse video characteristics.



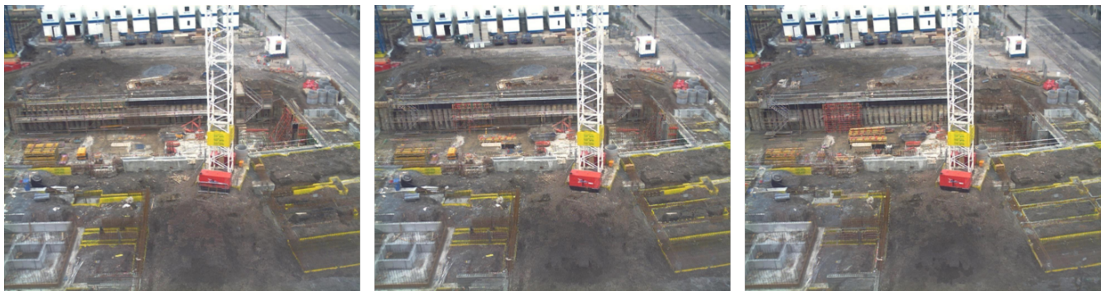
(a)



(b)



(c)



(d)

Figure 4.24: Probabilistic processing approach. Sample of three consecutive days. (a) Raw Images. (b) Non-parametric kernel density estimation filter. (c) Foreground Detection and Removal using Temporally Neighboring images (3 images) (d) Multivariate Median of Temporally Neighboring images (5 images).

Chapter 5

Conclusions.

5.1 Contribution

Time-lapse video is a useful tool for the visualization of events, such as the construction of a building. However, the capture process, which generally takes a long time, is susceptible to many disturbances that affect the quality of the images. This work has dealt with these problems to produce videos of higher quality and improved the user's perception of the scenes.

The preprocessing of the images was an appropriate solution to improve the results of the time- lapse video. It was the necessary primary stage before the image processing. More control could have been taken during the capture period to preserve the image data quality. For instance, by providing the video cameras with a better fixation mechanism to isolate them from external vibrations. In addition, an evaluation of the camera settings could have reduced problems with the color deviations. The white balance could have been left fixed rather than in automatic mode and the jpg compression could have been reduced (the latter produces an increment in the size of files). One of the principal aims of this piece of work is to reduce the number of artifacts that were present in our data

The jitter was one of the artifacts that deteriorated the video quality. It was reasonable to align the spatio-temporal volume before applying image processing methods that required pixel-to-pixel correspondence. Even though the jitter removal process reduced the shift between images to a value less than one pixel (0.1% of image size) , it is possible to detect this small vibration during the video play back. This shows that human vision is very sensitive to this kind of effect.

Changes in the lighting conditions and colors were also partially solved in the image preprocessing. The methods evaluated in this project showed the difficulties in finding a single approach from which is it possible to generalize for all the input cases. The assumptions that we used for the methods, such as the grey world, were suitable for most of the data samples and ultimately depicted a good normalization of the image colors. The images with AWB problems were almost totally corrected since the lack of balance of the color channels due to the camera was resolved. Normal images had a high correlation between the RGB histograms as opposed to imbalanced images. In addition, changes in illumination due to the weather were better normalized in normal images and sunny images with uniform distribution than in days with snow or fog. Images after the color normalization had a similar balance concerning pixel intensities and provided a better starting point for the processing of day images.

We developed two methods for producing time lapse videos. The deterministic model, consisting of the multivariate median filter works well to estimate the background of day images and remove most of weather changes and mobile objects. This included problem of ghosting which is related with the susceptibility of the algorithm to outliers. The same filter was used to work with consecutive day images within a local area. The amount of data available for this was very small for an accurate image improvement since it consisted of only five frames. Furthermore, taking into account that the jitter was not completely corrected, the results could be worse in this pixel-by-pixel approach. However, in the final reproduction of the video at 25 fps these types of artifacts are not easily perceived by the observer.

The probabilistic model for day background estimation based on KDE solved the problem of ghosting.. Nevertheless, due to the lack of knowledge about the source of the data, this method was prone to choosing elements that were not part of the building (background), and consequently these had a higher appearance rate in the images Therefore,, selecting elements from high density regions is not the only condition for their classification as background or foreground. It is also important for a local evaluation of the data (at the neighborhood scale) to make good use of the knowledge related to the scene behavior. The evaluation of the data showed that this particular set of images is very variable and that probabilistic approaches would improve any further processing technique on the data.

The neural network classifier was a reasonable approach to find foreground elements in the scene. The selection of small image regions is a better approach than using single pixels because it utilized information from the local variance of the images. The initial proposal , where five different cases were considered, was not suitable for this application. The reason could be related to a number of training elements, a similarity measure that does not capture the features of interest or the wrong selection of training samples biased by the user's interpretation of the scene. The classification of two types of elements was simpler than the multi-class system. An important methodological concern is related to the number of images required to detect the appearance of foreground. Only three images were used and this should be increased in order to make the methodology more robust.

One of the disadvantages of the algorithm implementation is the time needed to produce a complete video. This requires about two days of processing with five computers running simultaneously to create a complete video sequence. The most expensive stage is the day image generation, followed by the jitter removal and the day image processing. A possible solution to this could be to execute the algorithm in a different programming language. Matlab was the computing environment used to conduct all the experiments of this project. Languages such as C++ and Java provide a faster platform which can be a more suitable option since it improves the processing speed. A reduction in the processing time allows the addition of more images from the database. As a result, better background images can be obtained. Furthermore, they should be taken at hours outside the current range to provide more information about the real background and to reduce to probability of finding foreground elements in the same region.

5.2 Further Work.

5.2.1 Weather Classification

The images of the set can be better preprocessed by increasing prior knowledge about them. This approach seeks for the classification of images according to weather conditions. During the experiments it was observed that the preprocessing of the images worked better for some images than others and it was clear that this was related with the weather in the images. One proposal to improve the results of the TLV generation is to classify the images according to the weather changes and then apply dedicated processes to the data. For instance the images could be classified as: Cloudy,, fair,, fog, snow and rain.

Procedures that can be applied according to the class are for instance:

- Sunny days: Patches that are large and bright can be removed from the scene.

This means that if there is a region receiving direct sunlight and the values are saturated (seen as white), these regions could be discarded from the processing. Some of the flicker found in the video is due to the appearance of these patches for small time intervals. The images would be composed of pixels without values on these patches. These can eventually be infilled after posterior processing with data from other images. Some experiments were carried out to test the idea. Figure 5.1 shows examples of images where the bright spots were removed after looking for regions covering a large area and with pixel values close to the upper intensity value (255). Note that the process works very well when the images are sunny; however when this is not the case it will not necessarily take regions that are not bright. This confirms the need to classify these images.

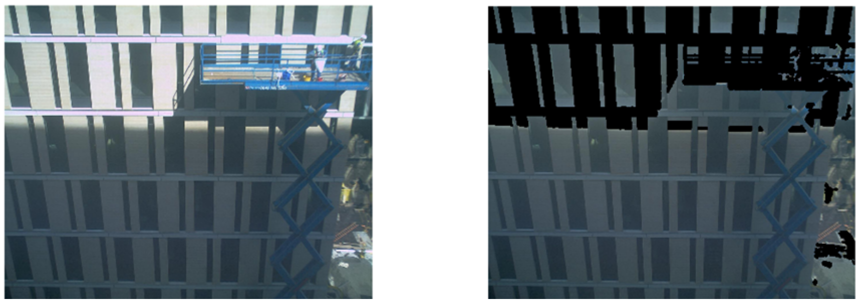
- Snowy days. After their detection, these days can be automatically removed from the set. If the purpose of the time lapse is to show a gradual depiction of the building evolution, without sudden transitions of events, then these images should not be included in the set. One idea regarding the difference between normal and snowy images that was found during the jitter removal process, was the cross-correlation between them was negative.

- Foggy days. A common characteristic of the images with fog is that they have low contrast. There are some available techniques for 'defogging' such as the one described in [Sun et al., 2004] which can be utilized to enhance images quality and make them look like bright days.

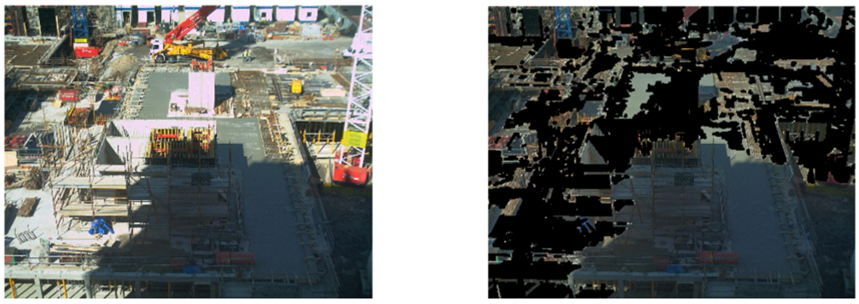
A suitable alternative to solve this problem is to implement a classifier similar to the one described in the processing section (4.3.2.3.1). The input would be a transformation of each image such as its histogram, or a combination of an image scaling and a linear transformation in PCA space. Even by simple inspection of the histograms it is possible to have an approximation of the weather in the scene, hence a method such as support vector machines or neural networks could be used in this approach.

5.2.2 Image Segmentation

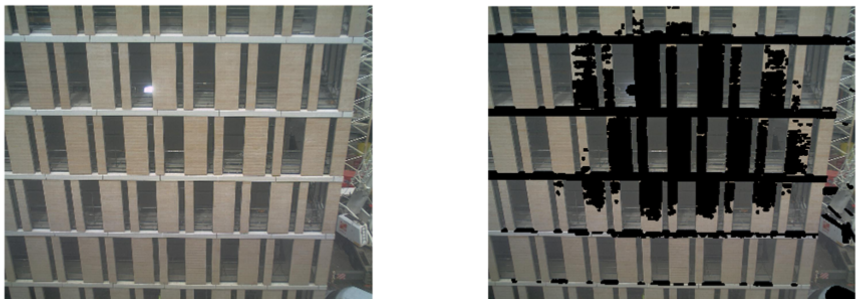
Images can be divided into multiple segments to find a more meaningful way of understand the data. We can find several regions in the images using segmentation and then determine precisely using groups of images (such as the ones from one day) if these regions represent background or foreground. Here we are concentrating on the idea of



(a)



(b)



(c)

Figure 5.1: Processing of sunny images. The bright patches are segmented from the images (black regions in right side images). Samples (a) and (b) are from sunny days. The same method is applied in (c) which is not a sunny image and the algorithm fails.

spatial connectedness of the objects in the scene. The objects (for instance stationary objects) can be tracked to see if they appear, or disappear, in the scene. Subsequently if it is detected that the object is foreground, then the algorithm for the estimation of the day background need not consider the image regions where the object was present. This method works with multiple shape regions rather than pixels or squared neighborhoods.

Additionally some of the elements in the scene such as trucks and cranes can be easily identified because of their color and shape. This information can also be taken into consideration when deciding whether an element is foreground. In this case, they can be easily removed from the data from the first stages of the image processing.

5.2.3 Background Model

Even though a model of the background is very difficult to obtain due to the variability of the scene, we can make an improvement in the background estimation if we take information from past days for the creation of day images. If the rate of change of image regions is small, then we can create a local background model. For instance, when the walls of the building were built, no relevant further changes occurred in the scene. In some of these images elements such as arms of cranes and sunrays appeared. These can be easily detected because the variability of those regions (walls) was small and these elements can be removed. Additionally we can use the model of previous days in order to make decisions about what is background when there is a high uncertainty; this will solve problems such as the ones found with the KDE where the decisions were based solely on the mode of the pixel values.

5.2.4 Grey World

In this project, 'gray world assumption' assumed that there was only a fixed RGB value representing the mean of color intensities in each image. A more realistic approach is to consider that this value changes progressively over time. Therefore, we can obtain a better color estimation of the images if we evaluate the mean of red, green and blue components directly from the images. For instance we can sample the means from image subsets and then define a variable 'grey' value for each day. This method must guarantee smoothness in the color transition and the values for each color channel are not required to be equal.

Bibliography

[513, 2002] (2002). *NETLAB: algorithms for pattern recognition.* Springer-Verlag New York, Inc., New York, NY, USA.

[axi, 2006] (2006). *AXIS 207Network Camera. UsersManual.* AXIS Communications. `http://www.axis.com/products/cam_207/index.htm`.

[Abeid and Arditi, 2002] Abeid, J. and Arditi, D. (2002). Linking time-lapse digital photography and dynamic scheduling of construction operations. *Journal of Computing in Civil Engineering*, pages 270–279.

[Althoff et al., 2006] Althoff, K., Degerman, J., Wahlby, C., Thorlin, T., Faijerson, J., Eriksson, R., and Gustavsson, T. (2006). Time-lapse microscopy and classification of in vitro cell migration using hidden markov modeling. *Acoustics, Speech and Signal Processing.*, 5:V–V.

[Bennett and McMillan, 2007] Bennett, E. P. and McMillan, L. (2007). Computational time-lapse video. In *ACM SIGGRAPH 2007 papers*, page 102, New York, NY, USA. ACM.

[Blunsden, 2008] Blunsden, S. (2008). Architectural reconstruction of the new informatics building. `http://homepages.inf.ed.ac.uk/s0346435/projects/small_proj/small_proj.html`.

[Chen et al., 2007] Chen, C.-Y., Chen, C.-J., Hunag, H.-C., Chen, Y.-J., and Hwang, R.-C. (2007). Automatic white balancing by using nn module. *Innovative Computing, Information and Control. ICICIC '07. Second International Conference on*, pages 269–269.

[Comaniciu and Meer, 2002a] Comaniciu, D. and Meer, P. (2002a). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619.

[Comaniciu and Meer, 2002b] Comaniciu, D. and Meer, P. (2002b). Mean shift: a robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619.

[Damera-Venkata et al., 2000] Damera-Venkata, N., Kite, T., Geisler, W., Evans, B., and Bovik, A. (2000). Image quality assessment based on a degradation model. *Image Processing, IEEE Transactions on*, 9(4):636–650.

[Davies, 2004] Davies, E. R. (2004). *Machine Vision: Theory, Algorithms, Practicalities*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Delon, 2004] Delon, J. (2004). Midway image equalization. *J. Math. Imaging Vis.*, 21(2):119–134.

[Efros and Leung, 1999] Efros, A. and Leung, T. (1999). Texture synthesis by non-parametric sampling. *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, 2:1033–1038 vol.2.

[Elgammal et al., 2002] Elgammal, A., Duraiswami, R., Harwood, D., and Davis, L. (2002). Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. *Proceedings of the IEEE*, 90(7):1151–1163.

[Ferguson., 2007] Ferguson., A. (2007). Method of the creation of time-lapse video of the informatics forum. The University of Edinburgh. Institute of Perception, Action and Behaviour.

[Fisher and Oliver, 1995] Fisher, R. and Oliver, P. (1995). Multi-variate cross-correlation and image matching.

[GBTimelapse, 2007] GBTimelapse (2007). Easter flowers. `http://www.gbtimelapse.com`.

[Grimson et al., 1998] Grimson, W., Stauffer, C., Romano, R., and Lee, L. (1998). Using adaptive tracking to classify and monitor activities in a site. *Computer Vision and Pattern Recognition Proceedings.*, pages 22–29.

[Koppal and Narasimhan, 2006] Koppal, S. J. and Narasimhan, S. G. (2006). Clustering appearance for scene analysis. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1323 – 1330.

[Lam et al., 2004] Lam, H.-K., Au, O., and Wong, C.-W. (2004). Automatic white balancing using adjacent channels adjustment in rgb domain. *Multimedia and Expo. 2004 IEEE International Conference on*, 2:979–982 Vol.2.

[Lee, 2005] Lee, D.-S. (2005). Effective gaussian mixture learning for video background subtraction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(5):827–832.

[Li et al., 2005] Li, Z., Schuster, G., and Katsaggelos, A. (2005). Minmax optimal video summarization. *Circuits and Systems for Video Technology, IEEE Transactions on*, 15(10):1245–1256.

[Lin, 2006] Lin, J. (2006). An automatic white balance method based on edge detection. *Consumer Electronics. 2006 IEEE Tenth International Symposium on*, pages 1–4.

[Lukins et al., 2007] Lukins, T., Ibrahim, Y., Kaka, A., and Trucco, E. (2007). Now you see it: The case for measuring progress with computer vision. In *Proceedings of the 4th International SCRI Research Symposium*, pages 409–422.

[Matsushita et al., 2003] Matsushita, Y., Nishino, K., Ikeuchi, K., and Sakauchi, M. (2003). Illumination normalization with time-dependent intrinsic images for video surveillance. In *In Proc. of IEEE Intl Conf. on Computer Vision and Pattern Recognition, 2004*, pages 3–10.

[Mittal and Paragios, 2004] Mittal, A. and Paragios, N. (2004). Motion-based background subtraction using adaptive kernel density estimation. *Computer Vision and Pattern Recognition. Proceedings of the 2004 IEEE Computer Society Conference on*, 2:II–302–II–309 Vol.2.

[R. Fisher and Wolfart., 2003] R. Fisher, S. Perkins, A. W. and Wolfart., E. (2003). Conservative smoothing. `http://homepages.inf.ed.ac.uk/rbf/HIPR2/csmooth.htm`.

[Rav-Acha et al., 2006] Rav-Acha, A., Pritch, Y., and Peleg, S. (2006). Making a long video short: Dynamic video synopsis. *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, 1:435–441.

[Reyes and Fisher, 2008] Reyes, J. and Fisher, R. (2008). Probabilistic time lapse video. `http://groups.inf.ed.ac.uk/vision/BUILDING/REYES_VIDEOS/`.

[Silverman, 1986] Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC.

[Smith and Kanade, 1997] Smith, M. and Kanade, T. (1997). Video skimming and characterization through the combination of image and language understanding techniques. Technical Report CMU-CS-97-111, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA.

[Sun et al., 2004] Sun, J., Jia, J., Tang, C.-K., and Shum, H.-Y. (2004). Poisson matting. *ACM Transactions on Graphics*, 23(3).

[Sunkavalli et al., 2007] Sunkavalli, K., Matusik, W., Pfister, H., and Rusinkiewicz, S. (2007). Factored time-lapse video. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 26(3).

[Świerczyński and Rokita, 2008] Świerczyński, Z. and Rokita, P. (2008). Increasing resolution of digital images using edge-based approach. *Opto-Electronics Review*, 16:76–84.

[Tappen, 2005] Tappen, M. F. (2005). Recovering intrinsic images from a single image. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(9):1459–1472. Member-William T. Freeman and Member-Edward H. Adelson.

[Taycher et al., 2005] Taycher, L., Fisher, J.W., I., and Darrell, T. (2005). Combining object and feature dynamics in probabilistic tracking. *Computer Vision and Pattern Recognition. IEEE Computer Society Conference on*, 2:106–113 vol. 2.

[Wang et al., 2000] Wang, Y., Liu, Z., and Huang, J.-C. (2000). Multimedia content analysis-using both audio and visual clues. *Signal Processing Magazine, IEEE*, 17(6):12–36.

[Washizawa and Yamashita, 2006] Washizawa, Y. and Yamashita, Y. (2006). Nonlinear wiener filter in reproducing kernel hilbert space. *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, 1:967–970.

[Weiss, 2001] Weiss, Y. (2001). Deriving intrinsic images from image sequences. pages 68–75.

[Wren et al., 1997] Wren, C., Azarbayejani, A., Darrell, T., and Pentl, A. (1997). Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:780–785.

[Yael Pritch and Peleg, 2008] Yael Pritch, A. R.-A. and Peleg, S. (2008). Video synopsis and indexing. `http://www.vision.huji.ac.il/video-synopsis/`.

[Yang et al., 2006] Yang, X., Li, H., and Zhou, X. (2006). Nuclei segmentation using marker-controlled watershed, tracking using mean-shift, and kalman filter in time-lapse microscopy. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 53(11):2405–2414.