Philosophy Doctor Thesis

Video Quality for Video Analysis

By

Pavel Korshunov

Department of Computer Science

School of Computing

National University of Singapore

2011

Philosophy Doctor Thesis

Video Quality for Video Analysis

By

Pavel Korshunov

Department of Computer Science

School of Computing

National University of Singapore

2011

Advisor: Dr. Wei Tsang Ooi
Deliverables:
Thesis: 1 Volume

Abstract

Video analysis algorithms are commonly used in a wide range of applications, including video surveillance systems, video conferencing, autonomous vehicles, and social web-based applications. It is typical in such systems to transmit video or images over an IP-network from video sensors or storage facilities to the remote processing servers for subsequent automated analysis. As video analysis algorithms advance to become more complex and robust, they start replacing human observers in these systems. The situation when algorithms are receivers of video data creates an opportunity for more efficient bandwidth utilization in video streaming systems. One way to do so is to reduce the quality of the video that is intended for the algorithms. The question is, however, can algorithms accurately perform on the video with lower quality than a typical video intended for human visual system? And if so, what is the minimum quality that is suitable for algorithms?

Video quality is considered to have spatial, SNR, and temporal components and normally a human observer is the main judge of whether the quality is high or low. Therefore, quality measurements, methods of video encoding and representation, and ultimately the size of the resulted video are determined by the requirements of human visual system. However, we can argue that computer vision is different from human vision and therefore has its own specific requirements to video quality and quality assessment.

Addressing this issue, we first conducted experiments with several commonly used video analysis algorithms to understand their requirements on video quality. We chose freely available and complex algorithms including two face detection algorithms, face recognition, and two object tracking algorithms. We used JPEG compression, nearest neighbor scaling, bicubic scaling, frame dropping, and other algorithms to degrade video quality, calling such degradations *video adaptations*. Experiments demonstrated that video analysis algorithms maintain high level of accuracy until video quality is reduced to a certain minimal threshold. We term such threshold the *critical video quality*. Video with this quality has much lower bitrate compared to the video compressed for human visual system.

Although this result is promising, given a video analysis algorithm, finding its crirtical video quality is not a trivial task. In this thesis, we apply an analytical approach to estimate the critical video quality. We develop a rate-accuracy framework based on the notion of rate-accuracy function, formalizing the tradeoff between algorithm's accuracy and video quality. This framework addresses the dependency between video adaptation used, video data, and accuracy of video analysis algorithms.

The principal part of the framework is to use reasoning about key elements of the video analysis algorithm (how it operates), essential effects of video adaptations on video (how it reduces quality), and if available, the semantic information about video (what is the video's content). We show that, based on such reasoning and a number of heuristic measures, we can also reduce the amount of experiments for finding critical video quality.

We also argue that in practice, an approximation of the critical video quality can be sufficient. We propose using video quality metrics to estimate its value. Since today's metrics are developed for human visual system, new metrics needs to be developed for video analysis. We propose two types of metrics. One type is based on the measurement of visual artifacts that video encoders introduce to video such as blockiness and blurriness metrics. Another type is a general measurement of information loss, for which we propose to use measure of mutual information. We demonstrate that visual artifacts based metrics give more accurate video assessments but work only for certain video adaptations; while mutual information is more conservative but

can be used for larger variety of video adaptations and is easier to compute. For temporal video quality, we study the effect of frame dropping on tracking algorithms. We demonstrated that by reasoning about tracking algorithms, as well as additional knowledge about tracked objects (measurements of its speed and size), we can estimate the value of critical frame rate analytically, or even approximate the tradeoff between tracking accuracy and video bitrate.

To summarize the contribution of the thesis: (i) we demonstrate on the few video analysis algorithms their tolerance to low critical video quality, which can lead to significant bitrate reductions when such an algorithm is the only "observer" of the video; (ii) we argue that finding such video quality is a hard task and suggest estimating it using algorithm-tailored metrics; and (iii) we demonstrate benefits in designing algorithms tolerant to reduced video quality and video encoders customized for video analysis.

Subject Descriptors:
I.2.10 Vision and Scene Understanding
C.2.4 Distributed Systems

Keywords:
Video Analysis Algorithm, Video Quality, Blockiness, Blurriness, Mutual Information, Video Surveillance

Acknowledgement

First of all, I would like to thank my advisor Wei Tsang Ooi for guiding me relentlessly and patiently through the Research Valley, which while being exciting and utterly rewarding in many ways, is still a very hard journey. I also want to thank my parents, my three younger brothers, and my little sister for being always there for me, even though we were separated by 10000 miles. Without family, I would not be able to push this work through to the finish line.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

We can describe the basic tasks of video analysis as automated extraction, processing, and structuring of essential information from images and image sequences obtained in the real world. These tasks are performed by video analysis algorithms, which define the way computers can "see" the world. The collection of such algorithms forms the field of computer vision, which is defined by Haralick and Shapiro as *"science that develops the theoretical and algorithmic basis by which useful information about the world can be automatically extracted and analyzed from an observed image, image set, or image sequence from computations made by special-purpose or general-purpose computers"* (Haralick & Shapiro, 1993).

In the last decade, computer vision research produced complex, fast, and accurate video analysis algorithms. Such characteristics as "complex", "fast", and "accurate" are relative to the specific tasks, previous approaches, or our expectations. Today's algorithms are complex in a sense that they are useful in many practical operations of detection, identification, and tracking of objects and events. Algorithms' speed is acceptable and is often real-time for conventional video sizes due to the latest advances in computing speed and optimizations in algorithms computations. The improvement in accuracy was influenced by many openly available datasets containing large collections of practical video and image data for testing of video analysis algorithms. Regularly organized competitions and challenges also motivate further growth of algorithms' performance. Therefore, with the latest increases in efficiency and reliability of video analysis algorithms, it is reasonable to say that computer vision is not only enhancing

but is replacing human vision in many practical applications.

The number of applications that rely or incorporate video analysis as a part of their core functionality is constantly growing. Traditionally, such applications include security-based applications such as video surveillance, visual biometric, and personal identification. In recent years, other types of video analysis-based application have emerged. Autonomous vehicles and unmanned aircrafts are good examples of such systems. Social applications such as social networks and photo sharing services started integrating face detection and recognition into their services. Many brands of hand held-photo and video cameras, as well as camera phones also include, at least, a face detection algorithm. Video analysis algorithms are also becoming an important integral part of video conferencing systems, systems for intelligent homes, care and nursery systems that watch over elderly and disabled people, and so on. Let us demonstrate how these intelligent and automated systems benefit from the video analysis algorithms.

New generation of video surveillance is one of the most prominent applications relying on video analysis algorithms. The research goal for such systems is to relieve human guard from the constant monitoring task of the surveillance site. Specifically, the aim is to alert the human guard only in situations when an action or a human intervention is required, for instance if video analysis algorithms detect and identify a suspicious person, object or event. According to Wu *et al.*, suspicious events are rare in typical surveillance environment (Wu, Jiao, Wu, Chang, & Wang, 2003b), which makes the goal feasible and achievable, subject to acceptable accuracy and efficiency of video analysis. The recent availability of fast computers, cheap video sensors, and advances in network technology brought the research in surveillance systems closer to this goal. But not only that, it also greatly expanded the range of surveillance applications from being used mainly for conventional monitoring of government or military facilities to an essential part of traffic control systems and integral part of intelligent homes. These advances made video surveillance a commodity easily available to general public. For example, in 2006, it was reported in the news[1] that in England there were more than 4 million surveillance cameras installed in the public places. London alone had about 500 thousands cameras in place. These figures indicate

[1] http://news.bbc.co.uk/2/hi/uk_news/6108496.stm

the increasing demand for efficient video analysis algorithms that can be easily deployed over existing infrastructure and can relieve human guards from the unnecessary surveillance tasks.

While the surge in hardware and network availability has given surveillance a new life, it has also sparked new types of video analysis-based applications, for example, in unmanned autonomous vehicles and aircrafts. The DARPA Grand Challenge[2] for autonomous vehicles is a prominent demonstration of the latest advances in computer vision and machine learning. The latest 2007 Urban Challenge required vehicles to navigate in suburban-like environment with heavy traffic, while obeying traffic laws, being able to drive around a parking lot as well as automatically react to road blocks and unexpected obstacles. The participating vehicles used GPS systems, radars, laser sensors, and video cameras to navigate themselves in a desert and urban environments. Although lasers and GPS were the most popular means of orientation among participating teams, video analysis was used for scene visualization, road detection, object detection and image filtering. The reason for relying more on GPS and laser systems in automatic navigation is that video analysis algorithms have not yet reached the level of maturity required by demanding practical applications. But because video is such a natural type of data for people, video analysis tools, though limited, are still implemented whenever possible.

Video analysis algorithms are also being integrated into various social services and collaborative systems. In video conferencing and presentation capturing systems algorithms are used for automated control and position of the cameras, for switching between the cameras and projectors, for zooming on faces of the speakers, etc. Photo sharing web services like Picasa[3] provide face detection algorithms for identifying people on the uploaded pictures. In Japan, many mobile phones are equipped with automated recognition of 2D barcodes, which are becoming very popular for encoding advertisements or additional information about the items in shops, information signs, etc. With an increasing accuracy of video analysis, many more applications can be developed. For instance, one could use mobile camera phone for automated tagging of friends on a photo or identification of the current location based on analysis of a landmark captured in the picture. It is evident that video analysis algorithms have grown to

[2]http://www.darpa.mil/grandchallenge/index.asp

[3]http://picasa.google.com/

become essential components in diverse range of conventional and newly emerging applications and technologies.

Many intelligent automated applications, examples of which we described above, rely on IP-based network for video or image transmission. It can be attractive for these applications to capture a video first and transmit it to a remote location for an automated processing later. There can be several reasons to perform video analysis remotely. For instance, the protection of intellectual property such as video analysis algorithms or increase in system's efficiency (it is easy to increase/decrease computational power of remote proxies when necessary) can be such reasons. Running video analysis algorithms remotely also allows using cheap video sensors with no computational power, which decreases the cost of the system. However, video streaming, whether it is necessary or attractive, comes at the tradeoff of the system's scalability. Video and images are conventionally bandwidth demanding data while the network bandwidth is a constrained resource. Therefore, there is a tradeoff between the number of videos that can be streamed through network and the quality (bitrate) of each video.

In this study, we address the problem of video streaming, considering systems that significantly rely on video analysis, with algorithms being main observers of the video. We also focus on a subset of such systems that use IP-network to transmit captured video to the remote location for its subsequent automated processing. Aside from some of the examples described above, we assume a typical representative system to be a video surveillance system with the following architecture. IP-based video cameras transmit video to remote processing proxies running video analysis algorithms. Each proxy relays the video to a monitor station (human guard) only in cases where something suspicious happens at the surveillance site. As it was stated above, suspicious events are typically rare (Wu et al., 2003b), therefore, most of the times, the video is transmitted only between cameras and proxies.

In such automated networked systems, solving a scalability problem comes down to understanding the requirements that video analysis algorithms pose to video quality. To our knowledge, however, there has been no systematic research in understanding of these requirements. For a given video analysis algorithm, such questions like how much video quality can

be sacrificed while not changing algorithm's accuracy or how to determine a sufficient video bitrate do not have clear answers. Typically, a newly proposed video analysis algorithm is tested on a set of video or images encoded with a quality conventional to human vision system. For instance, the survey of face detection algorithms by Hjelmas and Low (Hjelmas & Low, 2001) discusses and compares the performance of about ten different algorithms. Their accuracy is tested using the subsets of image dataset collected by MIT and CMU, well known as MIT/CMU dataset. However, such datasets consist of photos or video frames encoded for viewing by a human. In this thesis, we argue that computer vision is different from human vision and therefore its requirements to video quality should be studied differently.

There are few studies, particularly in the compression domain, that notice the effects of decrease in video quality on the accuracy of video analysis algorithms. Eickeler *et al.* (Eickeler, Muller, & Rigoll, 2000) propose a face recognition algorithm comparing its performance with several other recognition algorithms. In one of the experiments, the authors record the accuracy of their algorithm by running it on Olivetti Research Laboratory face database compressed to different JPEG compression ratios. The results demonstrate that the algorithm has no significant decline in accuracy until the compression ratio 7.5 : 1. Funk *et al.* (Funk, Arnold, Busch, & Munde, 2005) degrade test images with various compression qualities of JPEG and JPEG2000 to find how differently these compression algorithms affect performance of several fingerprint and face recognition algorithms. While concluding that JPEG compression has a higher impact than JPEG2000, the important result is that the tested algorithms show a decrease in accuracy only when images are highly compressed (based on the figure shown in the paper, to 10 times in terms of file sizes). Another study by Delac *et al.* (Delac, Grgic, & Grgic, 2005) also compares the effect of JPEG and JPEG2000 on several modifications of recognition algorithms. The conclusion the authors make is that "not only that compression does not deteriorate performance but it, in some cases, even improves it slightly" (Delac et al., 2005). The above results can be summarized and represented by illustration in Figure 1.1. It depicts the trend that accuracy of the described video analysis algorithms show when video quality is decreased. The figure essentially demonstrates a tradeoff between accuracy of an algorithm

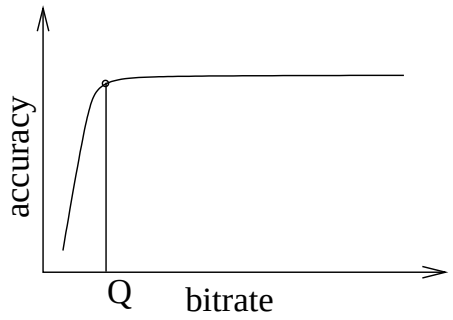


Figure 1.1: An example of rate-accuracy tradeoff for a video analysis algorithm.

and video bitrate, suggesting a certain sweet spot, the value of video quality, until which the accuracy remains the same as for original video. From the figure, it is evident that algorithms perceive video quality differently compared to humans.

However, noticing and stating the difference between computer vision and human vision perceptions of video quality is not sufficient for practical applications. It is important to understand if the rate-accuracy tradeoff given in Figure 1.1 is common for different kinds of video analysis algorithms. If so, the presence of such sweet spot can have important implications, as it suggests a limit on video quality and bitrate. Therefore, the thesis aims to study and answer the following questions:

- Determine if the rate-accuracy tradeoff in Figure 1.1 is common for various types of algorithms. Verify if it has a sweet spot.
- Understand how to find such tradeoff (or sweet spot) in practice for a given video analysis algorithm.
- Analyze the practical usefulness for the rate-accuracy tradeoff in automated network based systems. Study how knowing the tradeoff can improve scalability of such systems.

To answer these questions, we first, picked several commonly used and freely available video analysis algorithms. We use face detection, face recognition, face tracking, and blob tracking, which represent various types of video analysis algorithms. The algorithms are (i) Viola-Jones (Viola & Jones, 2004) and Rowley (Rowley, Baluja, & Kanade, 1998) face detection algorithms (ii) QDA-based recognition algorithm (Lu, Plataniotis, & Venetsanopoulos, 2003) (iii) CAMSHIFT (Bradski, 1998) face tracking algorithm, and (iv) a blob tracking, which uses

frame-differencing foreground object detection (Li, Huang, Gu, & Tan, 2003). There are other reasons for choosing these types of algorithms. Face detection and recognition are popular algorithms in large variety of applications from security systems to photo cameras. The availability of standard test data with ground truth is an extra reason to experiment with face detection and face recognition. Since these algorithms require only still images to work, we also consider face and blob tracking algorithms to study the impact of temporal component of the video on video analysis. Blob tracking is also commonly used in outdoor video surveillance systems.

To determine the tradeoff between video bitrate and accuracy of the algorithms, we measure the changes in accuracy for each algorithm with input video of different quality. To change video quality we use such *video adaptations* as JPEG compression, frame dropping, as well as bicubic, nearest neighbor, and pixel area relation spatial scaling. In agreement with Figure 1.1, we find that video analysis algorithms show almost no degradation in accuracy until a certain threshold, the corresponding quality for which we term *critical video quality*. We demonstrate that encoding video with critical video quality can amount to significant video bitrate reductions, e.g., 23 times for Viola-Jones face detection algorithm.

However, given a video analysis algorithm, how do we find its critical video quality? The naive approach is to empirically search for critical quality by running the algorithm with differently degraded videos, and find the video of the lowest quality, with which the algorithm still works well. These are the types of experiments that we first perform with our selected video analysis algorithms. In such empirical experiments, the components that participate in forming the rate-accuracy tradeoff are treated as black boxes. This process is illustrated by Figure 1.2. The figure shows how a video adaptation is used to degrade video quality; then, based on the performance of video analysis algorithm (in our case, it is accuracy), the process either loops back to continuing degrading video, or stops since value of critical video quality is found. In such scenario, neither information about video analysis algorithm, nor semantics of the video, nor the specific properties of video adaptations are considered. However, such information can help in avoiding unnecessary experiments. For instance, increasing frame rate does not help to improve the accuracy of a typical object detection algorithm, since object detection does not

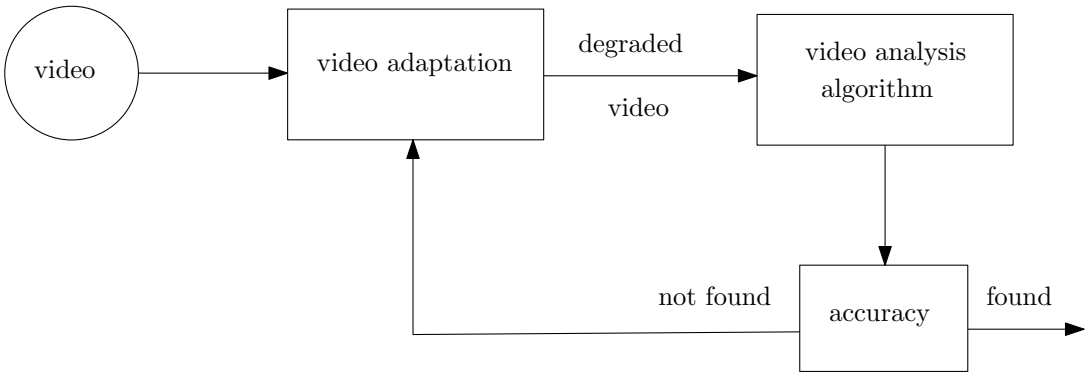


Figure 1.2: A process of finding critical video quality for a video analysis algorithm when video is degraded with a video adaptation.

rely on the temporal video component. This is a simple and intuitive example, but it illustrates that by knowing an algorithm's requirements, we can limit the scope of the experiments needed for find the critical quality value. Therefore, instead of using a black box blind approach, by analyzing each component of Figure 1.2, we can develop framework that can be used in practice.

In general, however, determining the requirements of a video analysis algorithm to video quality is a hard problem. Because the number of algorithms is very large and they are highly heterogeneous, we cannot generalize results on finding critical quality using just a few algorithms. Also, many non-trivial algorithms are based on neural networks or alike and are trained on rich empirical data (natural video or images). Such design of the algorithms prevents any justifiable formal analysis of their performance and generalization of the experiments. But since performing blind experiments for every different video analysis algorithm is undesirable, we propose to use a combination of reasoning/analysis (whenever possible) and experimental heuristics. We use a notion of rate-accuracy function as the centerpiece of the rate-accuracy framework, and a set of guidelines on how to use reasoning and heuristics for estimating critical video quality value. We identify a set of video properties that are crucial for a given video analysis algorithm. By studying the effect of a video adaptation on these video properties, we estimate how adaptation affects accuracy of the algorithm.

One important step in estimating critical video quality is to have metrics of video quality that are (i) suitable for video analysis algorithms and (ii) adequately measure degradation by video adaptations. For SNR video quality, the available metrics, such as PSNR[4], SIMM (Wang,

[4]Peak signal-to-noise ratio is commonly used as the quality metric of lossy compression codecs.

Bovik, Sheikh, & Simoncelli, 2004), PEVQ[5], are not suitable, because they were developed to measure the quality from the human perspective. To accommodate video analysis algorithms, we propose using metrics of visual artifacts (which manifest a strong video alteration), such as blockiness and blurriness, as well as adaptation independent measure of mutual information. We show that these metrics satisfy both above criteria. We find that the use of common metrics in a system that implements several video analysis algorithm and video adaptations can help in estimation of critical video quality, which reduces the number of experiments needed to find it. Therefore, the practical implementation of the critical video quality concept becomes more attractive. Based on our experiments, artifact metrics show higher precision in estimation of the critical video quality. Mutual information, however, is independent of the choice of video adaptation, though it is less precise compared to artifact metrics. As for temporal video quality, we show that by analyzing tracking algorithms and the effect of frame dropping on the speed of the tracked object, the critical frame rate can be found analytically without running experiments.

Use of reasoning/analysis opens possibilities for tuning video analysis algorithms to be more robust against the degradation of video quality. We show that by modifying face and blob tracking algorithms (adjusting video analysis algorithm's component of Figure 1.2) we can make them more tolerant to a lower frame rate. The algorithms are adjusting to the drops in frame rate using the measurements of speed and size of the tracked object and the estimation of these values after the next frame drop. Predicting where the object is likely to move after the frame drop reduces the chance of losing the object by tracking algorithm.

On the other hand, a JPEG compression algorithm can be modified (adjusting video adaptation's component of Figure 1.2) without affecting the accuracy of face detection algorithms, which we demonstrate by simplifying JPEG quantization table. Originally JPEG is designed to suit human visual system, perceptional requirements of which are incorporated into quantization table. Different implementations of JPEG have different quantization tables but all of them are obtained experimentally and usually are hard-coded into the algorithms. For video

[5]Perceptual Evaluation of Video Quality, more details here: http://www.pevq.org/

analysis algorithms, we replaced such table with a table constructed with a simple formula, which reflects the principles of quantization but does not contain information related to human perception. Such manipulation simplifies JPEG algorithms, since it removes the requirement of hard-coding YUV quantization tables.

1.1 Contributions

This work is the first to extensively study the tradeoff between accuracy of the video analysis algorithms and video quality and bitrate. Our results demonstrate that computer vision is different from human vision. Requirements of particular video analysis algorithm to video quality are easier to find, compare to human visual system, because with algorithms we can just run experiments and measure the resulted accuracy. However, video analysis algorithms are not so uniformed in the way they perceive video compared to humans. Such heterogeneity makes it hard to develop metrics of video quality adequate for all algorithms. It is also hard to design a uniform approach to finding critical video quality for various algorithms. Addressing this problem, we propose using metrics of video quality of two types: specific metrics selected based on the type of video encoding used and video analysis algorithm and metrics that measure general loss of information such as mutual information measure.

Armed with algorithm-specific video quality metrics, we focus our attention on the relationship between video analysis algorithms and video encoders. We show that new video analysis algorithms can be designed to accept low or purposely reduced quality of the video. Also, we believe that developing video encoders tuned to computer vision instead of human vision can improve the general robustness and stability of the video analysis algorithms as well as their tolerance to lower video quality.

The following summarizes the contributions of this thesis:

- We introduce the notion of critical video quality. A video analysis algorithm does not show significant loss of accuracy when ran on video with critical video quality or higher. Furthermore, video with this quality has much lower bitrate compared to the video conventionally encoded for human visual system. Therefore, we can save bandwidth when

video is streamed for computer vision.

- To avoid searching exhaustively for the value of critical video quality, we propose estimating it using video quality metrics that are selected specifically for a given video analysis algorithm.

- Using blob tracking and CAMSHIFT algorithms as examples, we demonstrate that video analysis algorithms can be designed to tolerate low video quality.

- By using simpler quantization tables for JPEG compression, we demonstrate the possibility of developing new compression algorithms designed for computer vision rather than human vision.

Now we describe background work for this thesis. We first discuss several video analysis algorithms that were used in our experiments and video adaptations that we employ to degrade video quality. We also present the overview of several video surveillance systems, as these systems are the main examples for application of our work. Finally, we describe the architecture of distributed video surveillance system that is assumed in this thesis.

1.2 Background

In this section, we describe the context of our work in the relevant research literature. As the major direction of the thesis is to study the requirements to video quality and bitrate when video analysis algorithms set to be observers, we give the background overview of several interconnected research areas. We give a review of the algorithms used in our experiments and analysis, following it with describing how we change and measure video quality in experiments. We end the background chapter with a general overview of the video surveillance systems, which are the main applications of our research findings.

1.3 Video Analysis Algorithms

Video or image analysis emerged with the ability to digitize the photography and video of the world around us. At first video analysis served mainly the purpose to help in image tuning and image effects for correcting imperfections in photos or making them look better. Then, after development of digital video surveillance cameras, the use of video analysis has increased in security applications.

As computational resources have grown and became more available, research in video analysis has expanded dramatically, with complex and meaningful video analysis algorithms coming close to be used in practical applications. Numerous algorithms have being proposed every year: object detection, tracking, recognition, event analysis, and fusion algorithms that work on combination of results from basic algorithms. As object detection, recognition, and tracking are some of the most common and important basic types of today's video analysis algorithms, we consider them in this thesis.

For the basic understanding of the background, we give a brief overview of various video analysis algorithms with emphasis only on several of them that are used in our experiments. We use Viola-Jones and Rowley face detection, QDA-based face recognition, and CAMSHIFT as face tracking and blob tracking algorithms. The main reason for choosing these particular algorithms was their availability to us as well as their complexity adequate to the practical reality (as opposed to simple motion detection).

1.3.1 Face Detection

Face detection can be considered as typical fairly complex type of object detection. Other common object detections are pedestrian, car, and carried item (suitcase) detections. It can be argued that out of the typical detectable objects, face is one with finer and harder to define features, which makes it more susceptible to large reductions in the quality of corresponding image. This reasoning has influenced our choice to use face detection algorithms in our experiments, since we wanted to determine critical video quality for useful and non-trivial video analysis algorithms.

Hjelmas and Low (Hjelmas & Low, 2001) give an overview of the evolution of face detection from first algorithms until year 2001. Algorithms have evolved from feature-based approaches, which rely on description of a face shape and its content, color, and edges, to image-based approaches that use a learning algorithm such as neural network or a weak classifier and train it using a set of simple image features or image statistics to determine a rare face among large amount of visual noise. The latter algorithms show better accuracy for detection of faces in realistic complex background. Typically, such algorithms have the following several common stages and components in detection of a face. The size of a face that can be detected is fixed to some minimum, usually close to 20×20 pixels. In an image a faces are searched by moving such window at different scales with small steps across the image. At each step classifier or filter, which is the core of the algorithms, matches the window to the "generic" face that can be generally described as a set of signature features and is obtained via offline training. In the final stage, all positive overlapping matches are combined together and a face is considered to be found for each such location. The main differences between various face detection algorithms lie in the implementation of a classifier and in the choice of signature features used to represent the face.

Rowley's Face Detection One of the first successful face detection algorithms based on neural networks was proposed by Rowley *et al.* (Rowley et al., 1998). The authors search for a face in every 20×20 pixel region first adopting preprocessing step proposed by Sung and Poggio (Sung & Poggio, 1998). Preprocessing includes lighting correction, subtracting bilinear lighting approximation in the region, and histogram equalization. Preprocessed image is passed through the neural network that looks for specific features, the hidden units, in a shape of smaller and larger squares and parallel stripes. These features are meant to detect such subregions of a face as mouth, lips, and eyes. Since neural network guesses many regions in the image as potential faces, a filtering stage is applied. Only regions with a number of overlapping detections above a certain empirical threshold are marked as a face. Such threshold determines the tradeoff between the accuracy rate and the rate of false positive detections. In our experiments, we use

the version of the algorithm available online[6] freely.

Viola-Jones Face Detection One of the most popular face detection algorithms available for the public use is the algorithms proposed by Viola and Jones (Viola & Jones, 2004) and implemented in Intel's OpenCV library[7]. The authors use some ideas from work by Papageorgiou *et al.* (Papageorgiou, Oren, & Poggio, 1998), which proposed to use Haar-like features as basic elements for face representation (see Figure 4.1). Similarly to Rowley, the authors also used the preprocessing stage suggested by Sung and Poggio (Sung & Poggio, 1998). The major contribution of Viola and Jones however, is the drastic improvement in the algorithm's detection speed, making it nearly real-time. They propose to use a hierarchy of classifiers constructed using AdaBoost (Freund & Schapire, 1995) method for selection of only important features. Each classifier in such hierarchy makes a decision (present or not present) on a single feature only, which serves as an input to the classifier at the higher level of the hierarchy. The authors pre-compute a special image representation, *integral image*, which requires small number of operations per pixel. This preprocessing step can be computed in a constant time, hence greatly speeding up the detection algorithm. The speed of Viola-Jones algorithms demonstrates 15 times faster detection rate compared to the Rowley's algorithm.

1.3.2 Recognition

Face recognition is an important task for wide range of applications including search engines, biometric and human-computer interaction applications, and video surveillance. We use face recognition algorithm based on QDA method, which is proposed by Lu *et al.* (Lu et al., 2003). The authors focus on solving a common problem of linear discriminant analysis (LDA) and quadratic discriminant analysis (QDA) based algorithms. For LDA and especially QDA based type of algorithms, the problem is the small number of available training samples compared to the dimension of the sample space. To overcome the problem, the authors proposed a modification of the QDA-based algorithm introducing additional weights to the recognition classifiers,

[6] http://vasc.ri.cmu.edu/NNFaceDetector/

[7] http://sourceforge.net/projects/opencvlibrary

which reduces the variance in the sample space making it more biased to certain type of samples. Experimental results presented in (Lu et al., 2003) confirm that the proposed solution outperforms several other face recognition algorithms including PCA-based, LDA-based, and traditional QDA based. We thank Terence Sim for providing the implementation of this algorithm. It is the only video analysis algorithm used in our experiments that is not available for public use.

1.3.3 Tracking

Object tracking is another important category of video analysis algorithms that we believe must be addressed in this study. One reason is that tracking is the central operation for many automated video surveillance systems, as well as for many emerging applications such as autonomous vehicles and robots. Another reason to study this type of algorithms separately is because of its dependency on the continuity of the video. Therefore, unlike for detection and recognition, video frame rate is a significant video quality for tracking operation.

There are two major approaches to tracking an object: feature-based and based on foreground object. First approach is to search the current frame for the set of specific features and relate to their position in the previous frame. Establishment of such relation identifies the tracked object, otherwise it is considered to be lost. Another approach is to identify moving regions in the current frame, which are called foreground objects. This step is usually based on the frame differencing operation. Foreground regions are obtained by subtracting one frame from another and applying on a set of obtained pixels a connected components algorithm or similar. Identified foreground regions are then relayed to the currently tracked objects based on their recorded trajectories, a set of features, or through other means. There is also a third type of tracking algorithms, which is the hybrid combination of the first two approaches. We chose two object tracking algorithms, which are described in brief below. CAMSHIFT face tracking is the representative of the feature-base tracking approach, while blob tracking algorithm relies on frame differencing approach and tracks detected foreground objects.

CAMSHIFT Tracking For our experiments on tracking faces we use CAMSHIFT algorithm proposed by Bradski in (Bradski, 1998) and implemented in OpenCV library. The algorithm tracks dynamically changing probability distributions. CAMSHIFT is essentially an adaptation of mean shift algorithm (finds the peak of a histogram in a single image) to a sequence of frames. We use a color histogram as the mean to track a face. CAMSHIFT employs a running average to keep histogram values adjusted with every next frame. Algorithm searches for the peak of the histogram inside a region of the previously known location of a face. The search window is 150% of the last found face size. Algorithm is simple and very fast but tracks only a single object and is easily affected by changes in environment such as lighting or occlusions. It also does not detect a tracked object automatically, so the initial location of a face must be set either manually or by using face detection. In our experiments with pre-recorded test videos, to avoid ambiguities, we set face location manually, while in practical lab tests, we rely on Viola-Jones face detection.

Blob Tracking We also use blob tracking algorithm implemented in OpenCV library, which is based on foreground detection proposed by Li *et al.* (Li et al., 2003). Foreground detection is done using background substraction from the current frame. Background mask is constantly updated and maintained with every new frame. Major contribution of the authors is the robust and fast algorithms for background maintenance. After foreground object is detected, the connected component analysis is performed to find the connected parts of the same object. Then, trajectory for each foreground object is constructed and updated accordingly. The algorithm can track objects in real time.

1.4 Video Adaptations and Video Assessment

Video adaptation is basically a term with which we describe a general way to alter the video. In this work, we consider mostly video adaptations resulting in reduced video bitrate by means of compressing it, scaling it down, or decreasing its frame rate. The purpose of typical video encoders is to reduce the size of the video while preserving its visual quality. Typically, the

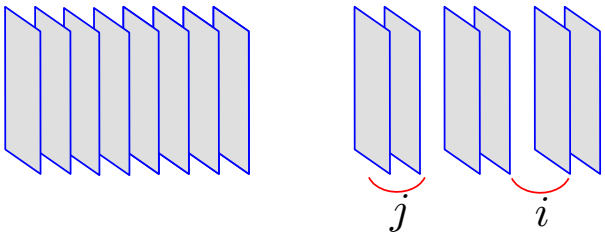


Figure 1.3: Dropping i out of $i + j$ frames. i is the drop gap.

judge of the quality is a human, and today's video encoders are developed for human vision. Since we do not take into account preserving video quality in terms of human vision, we consider video adaptation, as a more general way to modify video than encoding.

We have chosen JPEG[8] and MJPEG (which is many JPEG frames put together to form a video sequence) compression algorithms for images and videos we used in our experiments because of their relative simplicity, open availability, and a wide use. MJPEG is relevant in surveillance systems, since many network cameras, such as produced by Axis, primarily stream MJPEG. Axis cameras in particularly support MPEG-2 as well, but this codec requires a license, which reduces its popularity. In some of our experiments, besides MJPEG, we also used H.261 codec for video compression.

For face and blob tracking algorithms, we change the video frame rate of a test video by dropping frames from the original video using drop pattern: "drop i out of $i + j$ frames" (see Figure 1.3 for illustration). We vary i and j from 1 to 14. The value i represents the gap between frames. And j represents how many consecutive frames remain. For example, if we drop every third frame, i equals to 1 and j to 2; when three consecutive frames out of nine frames are dropped, i is 3 and j is 6. Note that while these two patterns give the same average frame rate, the accuracy of the tracking algorithm can be different.

In varying video quality with different video adaptations, it is important to compare the resulted qualities. Since in our experiments observers of the video are video analysis algorithms, we propose using metrics of video quality that are specific to these algorithms. A standardized metric can be used to compare videos degraded by video adaptations with different types of distortion. It also can be used for finding critical video quality for an analysis algorithm,

[8]We use the popular free implementation by IJG http://www.ijg.org/

provided the metric is a "perceptual" metric for the algorithm, i.e., it fits the way the algorithm analyses the video. Although, several quality metrics exist, such as objective PSNR metric or perceptive VQM and SIMM, they were designed for human visual system and, therefore, cannot be applied directly to video analysis. Algorithms, unlike humans, have different requirements on the video quality, and hence, the challenge is to design a metric that can accurately measure video quality for as many algorithms as possible.

We consider three different metrics that can be used to measure SNR quality of the video: blockiness, blurriness, and mutual information. Blockiness and blurriness are common distortion types, often called *video artifacts*. Other artifacts also include color bleeding, loss of colorfulness, and others. A non-reference blockiness by Muijs and Kirenko (Muijs & Kirenko, 2005) and bluriness by Chung *et al.* (Chung, Wang, Bailey, Chen, & Chang, 2004) metrics are adopted in our experiments. We demonstrate that proposed metrics can be used to estimate critical SNR quality for Viola-Jones face detection, Rowely face detection, and QDA-based face recognition algorithms. By "estimate" we mean that a single value of a given metric can be used to determine critical video qualities (sweet-spot of the given rate-accuracy curve) corresponding to different video adaptations. We use JPEG compression and various scaling algorithms as examples of different adaptations. We use blockiness metric with blocky video adaptations such as JPEG, nearest neighbor, and area relation scaling; and blurriness with bicubic and bilinear scaling.

Blockiness, blurriness, and potentially other visual artifact metrics can be used only with certain video adaptations. Such restriction causes inconvenience in using artifact metrics when wide range of video adaptations is implemented in a system. Therefore, it is desirable to have metric of video quality for a video analysis algorithm that is independent from the choice of video adaptation. We propose mutual information as such metric and show that it suites face detection and face recognition algorithms well. Mutual information was first introduced in information theory (Shannon, 1948) and has proven itself as a good similarity metric in image registration. It measures the amount of statistical information two different images share about each other, and it is easy to compute. This is more general measure of distortion compared to a

visual artifact metric, which focuses on a specific type of distortion. Also, mutual information is a better measure of video quality for video analysis algorithms than a commonly used distortion metric PSNR. This is because, for instance, mirroring an image to itself, while not affecting the performance of face detection or face recognition, changes the value of PSNR. Also, PSNR metric was developed to approximate value of MSE to human visual system. Mutual information value, on the other hand, is not affected by such operations like mirroring. It is also more general and simple way to measure the distortion and is not focused on human perception.

We demonstrate advantages of mutual information by measuring the quality of video degraded with different types of video adaptations. In addition to previously used blocky adaptations (JPEG, nearest neighbor and pixel area relation scaling), we also consider bicubic scaling algorithm, which adds a strong blurriness to the degraded image. We conduct experiments for both face detection algorithms, Viola-Jones (Viola & Jones, 2004) and Rowley (Rowley et al., 1998), and QDA-based face recognition (Lu et al., 2003). Similar to blockiness and blurriness, we show that mutual information can be used as a metric of video quality for the selected algorithms. It means that a single threshold value of mutual information can be used to estimate the critical quality for a particular algorithm across various video adaptations.

1.5 Video Surveillance Systems

In this section, we present a review of several automated video surveillance systems as such systems are popular applications that can benefit from the solution proposed in this thesis. In the literature review Chapter 2, we discuss video surveillance emphasizing on how the bandwidth problem is dealt with in such systems. In this section, however, we give a background observation of video surveillance systems in general. Major trends in the surveillance research focus on implementation of efficient system architecture, practically useful event and tracking algorithms, effective collaborations between multiple video cameras, data fusion and trajectory building. We start with VSAM, one of the pioneering automated systems for outdoor surveillance, and end with DOTs, a sophisticated indoor surveillance system developed by FXPAL.

Since the focus of the research was primarily on the development of video analysis algorithms,

the problem of the bandwidth limitation was getting a little attention. Among researchers there are two predominant ways to handle this problem. First one is taken by system developed for practical use and they follow VSAM's approach in providing as much visual data as a particular network conditions allow. In such approach, conventional means for reducing bitrate of video data are implemented; such reduced video resolution and compression. Another approach is to assume a sufficient availability of the bandwidth, which allows focusing on other research problems. There is no research on automated surveillance that would address specifically the problem of the bandwidth in such systems.

VSAM Surveillance System One of the first full scale automated video surveillance systems is VSAM, which was developed under the leadership Kanade as part of DARPA project (Collins, Lipton, Kanade, Fujiyoshi, Duggins, Tsin, Tolliver, Enomoto, & Hasegawa, 2000). The purpose of the system was an outdoor surveillance using a distributed network of calibrated video cameras and other sensors (thermo-cameras, omnicamera). The authors advocated paradigm of the "smart" sensors, which objectives were to perform independent surveillance tasks in real-time. It employed several state of the art algorithms of object detection and tracking, object classification into categories of human or a vehicle, and a simple gate analysis of moving people. Since VSAM was a full scale surveillance system, the authors also addressed such issues as data fusion from multiple sensors, cooperation of several sensors on a given surveillance task, user interface issues like visualization and sensor control, as well as practical issues such as computational and bandwidth efficiency. To accommodate limited computational power, the authors made some hard choices on efficiency of detection and tracking algorithms. The problem of limited bandwidth was addressed by allowing only a low quality video to be streamed in real time from one selected video sensor. The rest of the sensors (they also had a workstation attached to them) were only sending data describing tracked objects: coordinates, size, speed, etc. Limited bandwidth was one of the major limitations of the system making choices of a workstation to each camera and 3-D visualization of the surveillance site to be necessities rather than aids in the surveillance.

The results obtained from VSAM system have encouraged more research efforts in develop-

ing full-scale automated surveillance systems. Since major novelty in such systems was their automated component, developing robust and accurate video analysis algorithms became one of the main directions in the surveillance research. The main attention was on improving accuracy of detection and tracking, incorporating them into event analysis, dealing with occlusions, tracking through multiple cameras, attempting to discard the calibration of the video sensors. We consider several of such systems here.

KNIGHT Surveillance System KNIGHT system focuses on outdoor surveillance (Javed & Shah, 2002) (Javed, Rasheed, Alatas, & Shah, 2003). The proposed approach assumes the use of multiple un-calibrated cameras with overlapping and not overlapping fields of views (FOVs). Authors also present algorithm for movement detection and tracking of moving objects for single camera. The system also is able to differentiate a walking, running or falling person as well as cars, groups of objects. Each camera in the system has attached workstation for processing video data to provide graphic vision operations such as object detection and tracking. The results of operations are sent to a central server, which combines them for analysis and further movement predictions. To provide multi camera tracking mechanism, since the system does not require calibration of cameras and has no knowledge of paths topology, the system training phase is introduced. During the training phase the system learns the relationships between cameras and probable paths of movements using Parzen windows. Efficient tracking across overlapping cameras is relatively easy since the relationships between fields of views of different cameras are known from the training phase. While the main challenge is to correctly predict an object trajectory across multiple non-overlapping cameras. During the active phase of the system central server collects from all cameras the information about their viewed objects movements such as current objects velocities, directions speeds, etc. Using such local trajectories of all cameras and possible paths information obtained during the training phase the server predicts global trajectory of each object using linear velocity model (Javed et al., 2003). This allows tracking of objects across cameras with overlapping FOVs as well as prediction of objects trajectories through multiple non-overlapping cameras.

The single camera tracking detection and classification algorithms were tested on the set of

general video sequences performing well when people were not occluded. Authors claim correctness and high performance of tracking algorithm across multiple cameras but the evaluation was performed using a small testbed.

SfinX Surveillance System Concurrently the research on the system SfinX which has objectives similar to the KNIGHT system is carrying out by group from University of California. The overview of the system and the main results can be found in (Wu et al., 2003b) (Niu, Jiao, Han, & Wang, 2003) (Wu, Wu, Jiao, Wang, & Chang, 2003a) (Rangaswami, Dimitrijevi, Kakligian, Chang, & Wang, 2004). The focus of their research is mainly on development of algorithms for intelligent classification of moving objects such as people and cars and event recognition to distinguish suspicious events, i.e. one person is passing an object to another person. The problem of tracking across multiple cameras was not addressed. Interestingly, authors of system KNIGHT argue that maintenance of calibration of a large network of sensors is a significant maintenance task (Javed & Shah, 2002). However the approach proposed in (Wu et al., 2003a) requires camera calibration and authors claim that it needed to be done once and off-line. The problem of pose registration of moving camera was suggested to be solved using Church's algorithm, which was originally developed for aerial photogrammetry. This technique requires knowledge of only three observed landmarks' coordinates (compare to usual six point correspondences) for each camera.

Authors proposed more sophisticated algorithm of classification events and objects than one used by VSAM system. Considering one camera, the movement trajectories are recognized first. Then, the algorithm recognizes motion patterns such as hands and head motions. Performing sequence alignment learning and imbalanced kernel boundary alignment techniques, the authors are able to extract suspicious events and movements of people. Analysis of merging and splitting of objects is also performed for more semantic classification (Niu et al., 2003). The evaluation shows that the algorithm tracks very well even the occluding and splitting objects, but the tests were not comprehensive and, hence, not convincing enough.

DOTs Surveillance System There are also systems that are specific to indoor surveillance. Indoor surveillance provides a special subset of condition that makes some tasks of video analysis easier. The major difference of indoor conditions from outdoor is the absence of interferences from weather, presence of persistent and controlled lighting, and a more structured terrain with movement trajectories that are easier to predict.

A good example of the sophisticated indoor surveillance system is the system recently developed by FXPAL called DOTS (Girgensohn, Kimber, Vaughan, Yang, Shipman, Turner, Rieffel, Wilcox, Chen, & Dunnigan, 2007). The system operates with multiple calibrated cameras installed in hallways and public places in a typical office environment. It is designed for automatic real-time tracking of multiple people with the emphasis on convenient and reliable user interface. The tracking algorithm is based on foreground segmentation that is robust to shadows and illumination changes; occlusions are also handled by the algorithm. Use of calibrated cameras allows efficient tracking through cameras with overlapped and non-overlapped views as well as location estimation of the tracked object. Face detection algorithm is also implemented to identify faces at entrance and exit locations of the building. Detected faces are bound with tracked objects and used as visual identifiers by the system's user interface. The system provides an elaborate and flexible interface, which includes map of the surveillance site, timeline that includes stored and current surveillance information (trajectories, videos, faces) and 3D virtual model of the building. System's implementation uses 23 Axis 210 IP video cameras and Motion JPEG video is streamed and recorded at the rate of 15 frames per second.

1.6 Our Architecture of Video Surveillance System

We assume the following architecture of a video surveillance system. It consists of a number of video sources, processing proxies, and monitoring stations, connected via a wide area network. Video sources can be either networked cameras or video sensors. These sources capture, encode and transmit video streams to processing proxies. Processing proxies are computers dedicated to the processing and filtering of incoming video streams, and if needed, relaying them to monitoring stations. The need to relay depends on the queries specified by users. For instance,

a user may request to see a certain video if suspicious events are detected. A sample query is "Show me the video of secured room X if someone is detected in the room." A video source sends surveillance video from room X to a remote proxy. The proxy then runs a motion detection algorithm on the surveillance video. The proxy relays the video to the monitor only if motion is detected in the room. Figure 1.4 shows the architecture of such distributed surveillance system.

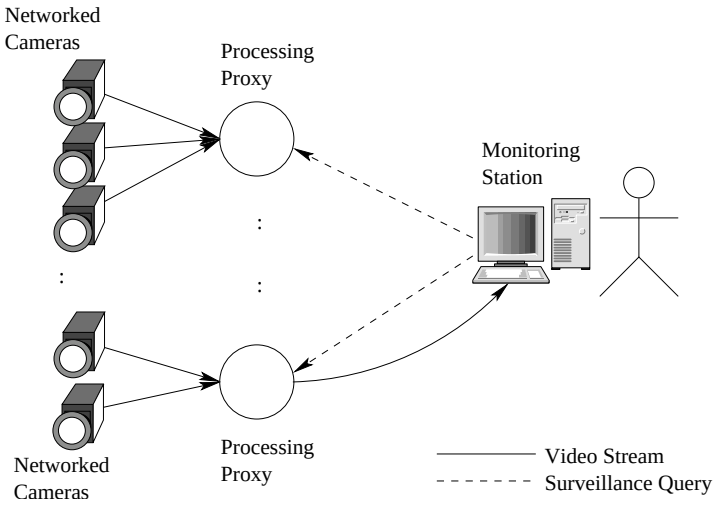


Figure 1.4: Architecture of Distributed Video Surveillance System.

Using such distributed architecture for video surveillance has several advantages. First, it allows flexibility in adding and removing cameras. Second, since video processing is done at the proxies, cheap networked cameras or video sensors can be used as video sources. Finally, by filtering uninteresting video at the proxies, the number of streams to be sent to the monitoring station is kept small, thus increasing the scalability of the video surveillance system. Due to these advantages, this type of architecture is becoming common in commercial video surveillance systems (e.g., ObjectVideo[9], MOXA[10]).

Note that our architecture does not consider archiving full quality video from video sources. While such archives would be useful for forensic video analysis, performing continuous archiving of full quality video from large number of video sources does not scale. Our architecture, however, does not preclude archiving videos at monitors.

[9] http://www.objectvideo.com

[10] http://www.moxa.com

Chapter 2

Literature Review

In this chapter, we review some of the work that is most relevant to our study. We discuss the rate-distortion framework and its application to video and image compression. We also describe framework based on utility function, which is an extension of rate-distortion framework that consider video quality in a broader sense, consisting of with SNR, spatial, and temporal components. Also, video compression is generalized into notion of video adaptation, which can include frame dropping, scaling, and other video degradations. In this thesis, we take a similar approach to the utility-based framework, but, instead of human, we assume video analysis algorithms to be main observers of the video. We also briefly discuss the ways to reduce video bitrate based on the information about video content, including techniques using region of interests and approaches based on viewer attention. Issues of scalability in video surveillance and sensor networks are also discussed emphasizing on how scalability problem is addressed in practical systems.

2.1 Rate-Distortion Theory and Utility Function

Originally proposed by Shanon, rate-distortion theory focuses on a unit of information transmitted over a noisy channel and studies the relation between distortions caused by the transmission and the number of bits necessary to encode the information. An application of this theory to image and video compression was developed into a framework, commonly known as rate-distortion

framework (Ortega & Ramchandran, 1998). In this framework, rate is the number of bits per second of the compressed video. Distortion is interpreted as an amount of degradation in quality of compressed video compared to original. Since human is assumed to be the main video observer, the resulted distortion of the compressed video should satisfy requirement of human visual system (HVS). Therefore, the distortion in rate-distortion framework corresponds to the perceptual video quality. Since perceptual quality is hard to measure, HVS-oriented distortion metrics are normally used. One of the most common and simple metric of distortion, while also the most criticized, is mean square error (MSE).

Equipped with video quality metrics, rate-distortion framework deals with the tradeoff between distortion and bitrate of compressed video. Low video bitrate is desirable for faster transmissions or smaller storage while low distortion entails higher perceived video quality. Higher compression yields lower bitrate but also higher distortion manifesting the rate-distortion tradeoff. It was discovered however that absence of high visual frequencies is less noticeable by human visual system than absence of low frequencies; therefore, when performing lossy compression, higher frequencies can be discarded first. Such approach allows achieving significant reductions in bitrate with minimal impact on the perceptual quality. All commonly used lossy compression algorithm such as JPEG, MPEG, and JPEG2000, are based on this approach.

The complexity of rate-distortion framework lies, however, in the fact that video and image data are not homogeneous. Different regions of an image or a video frame can have different intensity variations, different color and other image statistics, as well as can have be different semantically. For instance, a face in the image can be more important than the background. Therefore, to achieve an overall desired distortion, different compression parameters should be used for different image regions. For example, in JPEG, each 8×8 pixels block of an image is compressed independently (Ortega & Ramchandran, 1998). Such approach results in different combinations of compression parameters leading to the same overall value of distortion (measured with some metric). Each such combination, in turn, corresponds to different bitrate. Since the goal of compression is to minimize the bitrate, the problem of finding the combination of compression parameters resulting in smallest bitrate needs to be solved.

Therefore, the rate-distortion framework addresses two related optimization problems. First problem is to minimize the resulted video bitrate while preserving overall distortion above a certain threshold. Second problem is: for given bitrate value, determine compression parameters that result in compressed video with the least distortion. Both of these problems are inter-dependent in a sense that solving one leads to the solution of another. There are several methods for finding optimal solution. One of the most popular is method of Lagrange optimization (Ortega & Ramchandran, 1998). It exploits the fact that rate-distortion function is generally a convex function and finding the optimal solution at a certain point is equivalent to finding the angle of the tangent line to the function. Another popular method exploits dynamic programming when possible solutions are gradually built up from one another, and the optimal can be found by a taking a simple minimum (or maximum) value in the table of solutions (Ortega & Ramchandran, 1998).

Rate-distortion framework is generally applied to image compression, more specifically to reducing image (or video) signal-to-noise ratio (SNR). However, video quality can be considered to have three components, SNR, spatial, and temporal. Therefore, the idea of the distortion (typically tied with SNR quality), and subsequently rate-distortion framework, can be viewed in more general sense. Generalizing rate-distortion framework, the group of researchers under Chang S.F. (Kim, Wang, & Chang, 2003; Wang, Kim, & Chang, 2003; Chang & Anthony, 2005) proposed the notion of utility function, which formalizes the combined quality of the video with SNR, spatial and temporal components. Video compression is also extended to a general notion of video adaptation, a general way of degrading video. Video adaptations can include frame dropping, spatial scaling, and other video altering such as de-noising, or more sophisticated content-aware filtering. Therefore, the problem of optimizing video bitrate in the rate-distortion framework evolves in finding a set of optimal parameters for compression, frame rate, and scaling that would satisfy the general constraint on video utility. Likewise utility can be maximized given the minimal required bitrate.

The authors of utility function based framework (Wang et al., 2003) consider two different video adaptations as examples, dropping of DCT components in MPEG compression and frame

dropping, as well as their combinations. The utility function is considered to be specific to a particular application, for instance, it can be objective or subjective video quality, user satisfaction, etc. The authors argue however utility function is, it can be represented as a set of several video characteristics or features, such as motion variance, average quantization step, average motion intensity, average PSNR, etc. (see (Wang et al., 2003) for more details). Therefore, the problem of finding optimal utility for the given bitrate constraint transforms into the problem of finding optimal values for the features set. The authors developed a prototype of the adaptation system for MPEG-4 video codec. The system uses combination of video adaptations, dropping of DCT components and frame dropping, and it achieves higher utility (PSNR and subjective tests used as an example of utility functions) for the given limit on video bitrate. The value of utility was higher compared to scenarios when only one of the above video adaptations was used. These results demonstrate that a clever combination of different video adaptations can lead to higher gains in video quality, or otherwise is also true: lower video bitrate can be achieved with the same video quality.

The latest research on how video quality affects perception of human visual system is conducted by group under S. Hemami (Rouse & Hemami, 2008a) (Rouse & Hemami, 2008b) (Rouse, Pepion, Hemami, & Callet, 2009). The authors of the work differentiate three types of video assessments, namely, fidelity assessment (visibility of distortions), quality assessment (tolerance to visible distortions) and utility assessment (usefulness of the distorted image with reference to original), which is the primary focus of their study. The authors argue that there exists a recognition threshold, the value of distortion, degrading beyond which the content of the image cannot be recognized by the human visual system. Two ways of degrading video quality was presented, signal-based (dropping subbands of discrete wavelet transform) and preserving visual structure (smoothing based on total variation). Using the subjective studies through of questionnaire 25 people and natural images from A57 database, based on the subjective scores, the recognition thresholds were found to be different for each image. Authors use the information about these recognition thresholds to develop a new video quality assessment algorithm, called NICE, because commonly used algorithms, including PSNR, SIMM, VSNR, VIF, were found

to be not satisfactory, especially, for high distortions.

In this thesis, we adopt the utility function to the scenario when a video analysis algorithm is the observer of the video instead of human. In our work the problem also grows in dimension since, instead of the few user-oriented utility functions, every video analysis algorithm is impacted differently by every different video adaptations. From the other hand, contrary to utility functions accuracy of an algorithm can be obtained experimentally. In Chapter 5 we discuss in more details the dependency between video adaptations and accuracy of video analysis algorithms.

2.2 Semantic Video Reduction

Many techniques were proposed for adapting video transmission rate to meet the bandwidth constraints of wide area networks. One of the first suggested methods, presented by Eleftheriadis and Anastassiou (Eleftheriadis & Anastassiou, 1995), uses a rate-distortion function to find minimal distortion. Based on the bandwidth capacity predicted via monitoring the current state of the network, the video is dynamically reshaped by being encoded with different quantization values. Extending this idea, Kim and Altunbasak (Kim & Altunbasak, 2001) suggested a technique to reshape video by scaling its spatial, temporal and SNR properties. This technique was later generalized into a utility-based framework by Kim *et al.* (Kim et al., 2003). These approaches aimed on reducing the time and complexity of re-encoding the video for the network with limited bandwidth. In this thesis, we adapted some of these ideas, though we focus on the case where the video observers are video analysis algorithms rather than human.

Region of interest (ROI) is another technique to reduce video transmission rate. This technique transmits only important regions in video frames at high quality (Schumeyer, Heredia, & Barner, 1997) (Sanchez, Basu, & Mandal, 2004). This approach can be adapted for video analysis algorithms, for instance, video sources can stream only regions with faces for the later recognition. Implementation of ROI in a practical system, however, requires a significant level of intelligence and more computing power at video sources. Video sources would have to execute detection algorithms for extracting such regions of interest from the video before transmitting.

Our work is different because we consider an analysis algorithm to be a "viewer" of the video but is similar in a sense that we want to understand what is the minimal important information we need to keep in the video for the analysis algorithm to perform accurately still.

2.3 Scalability of Video Surveillance

Some of the researches in video surveillance proposed several solutions for reducing the amount of information streaming through network. Yuan *et al.* (Yuan, Sun, Varol, & Bebis, 2003) and Nair *et al.* (Nair & Clark, 2002) presented systems that avoid using excessive network bandwidth by periodically sending still images from a video source to the end user. VSAM (Collins et al., 2000) deals with bandwidth constraint by sending only one low quality video at a time, and relies on workstations attached directly to video sources for the detection, tracking and classification of events. Such solutions limit the amount of visual information that is available to the viewer and are not scalable. The authors of many recent surveillance systems, for example SfinX (Rangaswami et al., 2004) and KNIGHT (Javed et al., 2003), prefer not to address the problem of video streaming and, instead, focus on developing more accurate video analysis algorithms. However, the authors of DOTS surveillance system (Girgensohn et al., 2007) acknowledge that even for their indoor surveillance system, the scalability was a problem, allowing them to have only 15 video cameras streaming simultaneously. Addressing the problem of video streaming in surveillance systems, in this thesis, we suggest taking the advantage of video analysis algorithms' tolerance to low video quality. Since in typical surveillance scenario, suspicious events are rare (Wu et al., 2003b), it is not necessary for human to constantly observe all video streams but only those that require his/her attention. Therefore, most of the time, the video is transmitted for video analysis algorithms only, allowing us to significantly reduce its quality and, hence, increase the scalability of the surveillance system.

A recent work that is most related to ours that of Boyle on the effects of capture conditions on the CAMSHIFT face tracker (Boyle, 2001). That study aims to recommend how to set up a low-end web-cam for face tracking on desktop computers. Similar to our experiments, the study examines the effects of frame size, frame rate and compression quality on the CAMSHIFT face

tracker. Experiments presented by the author, however, only focus on a few values for each of these qualities, and do not explicitly address the issue of critical video quality.

In the thesis, we address the scalability problem of automated distributed video systems. We suggest fixing the bitrate of streaming videos to the limit, for which video analysis algorithms still perform accurately. To this date, there is no research has been done that would address this issue. Intuitively, video analysis algorithm should have a threshold on video quality. Our goal is to determine how low it is compared with quality conventional for human vision. We also want to understand how different video analysis algorithms perceive video quality. We want to formulate approach for determining the required quality for an algorithm in a practical scenario.

2.3.1 Sensor Networks

Sensor networks belong to a type of distributed system that required pass large amount of real-time data from one remote node to another. Another characteristic of sensor networks is often limited computational and electrical power of each node. Therefore, the problem of sending as little data as possible is important for such systems.

A good representative of sensors networks is IrisNet. The general architecture proposed by Gibbons *et al.* (Gibbons, Karp, Ke, Nath, & Seshan, 2003) aims on organizing real-time changing data in the distributed storage with ability to perform real-time queries by the user. It is assumed that the raw data from sensor can be preprocessed near the source and can be significantly reduced in size for ease of transfer. The system has an agent-based architecture consisting of sensing and organizing agents. Clearly, the sensing agents collect and process data and transfer result to the organizing agents, which maintain up-to-date data in storage and perform queries. The distributed data are organized in XML hierarchy based on their geographical location. The system insures that each list of the hierarchy has a unique path from the root and allocates the data on physical machines in branch-based manner. The system maintains for entire network one XML document representing this logical structure of data. The geographical-based approach for organizing data into hierarchy is straight forward since

the authors of the system assume each sensor has its own uniquely defined geographical location. During query processing organizing agent identifies what parts of answer for the query can be found locally and locations of others parts of answer using knowledge of the described hierarchy. As fresh data is sent from the sensor to organizing agent it updates the corresponding storage and cashes are updated as well. For verification of freshness of requested data timestamps of updates are stored along with data. The obvious drawback of this approach is that it is not scalable. If system manages significantly large number of sensors it has to maintain as a one piece the XML document, which will be very large in size because contains the path information about each sensor. And XML way of representation itself is not very scalable since XML syntax is over saturated.

Chapter 3

Video Quality and Video Analysis: Motivation and Overview

In this chapter, we discuss the motivation for this study and present overview of the experiments demonstrating the dependency between video bitrate and accuracy of several video analysis algorithms. We conducted preliminary experiments to understand what kind of video quality is necessary for video analysis algorithms and how changes in quality affect their accuracy. In these experiments, face detection and face recognition algorithms were used. We focus on two video adaptations, JPEG compression and nearest neighbor scaling; and we study how these adaptations and their combination affects the accuracy of the algorithms. The process of determining critical video quality can be illustrated by Figure 1.2. Obtained results motivated us to study the dependency between accuracy of the video analysis algorithms and video bitrate (we call it rate-accuracy tradeoff) in more details and to consider other examples of algorithms and video adaptations. The following chapters contain extensive experiments of several detection, recognition, and tracking algorithms. First, we treat these algorithms and video adaptations as black boxes and determine critical video quality via experimental search. Then, we propose a framework based on combination of reasoning and heuristic measurements, allowing us to formalize the tradeoff between video bitrate and accuracy of video analysis algorithms, as well as to estimate a critical video quality in practical scenario.

3.1 Rate-Accuracy Tradeoff

To understand what rate-accuracy tradeoff is we take an algorithm, video/image, and video adaptation and, gradually degrading its input video, we note the changes in algorithm's accuracy. The rate-accuracy tradeoff can be visualized by plotting the bitrate of the degraded video vs. the resulted accuracy value of the algorithm.

As a first example of video analysis algorithm, we picked face detection by Viola and Jones (Viola & Jones, 2001) implemented in OpenCV [1] library. The advantages of using this algorithm include its popularity in research community, availability of standard test data with ground truth, and that it is non-trivial (as opposed to motion detection for example). Its main disadvantage however is that it works on static images and the changes in video frame rate have no meaningful impact on its accuracy. Figure 3.1 demonstrates how changes in compression quality affect algorithm's accuracy. We compressed an image from a standard MIT/CMU dataset (see Section 3.2 for more details) using different JPEG [2] values and ran the algorithm on it. A square around the face indicates that the algorithm detected the face. We can see that face is not detected only for the last face in the figure. The compression value 9 that was used in the last case leads to the 90% reduction of image file size. If most of the images with detectable faces can be reduced to a similarly small size, this finding can be useful for many practical automated applications.

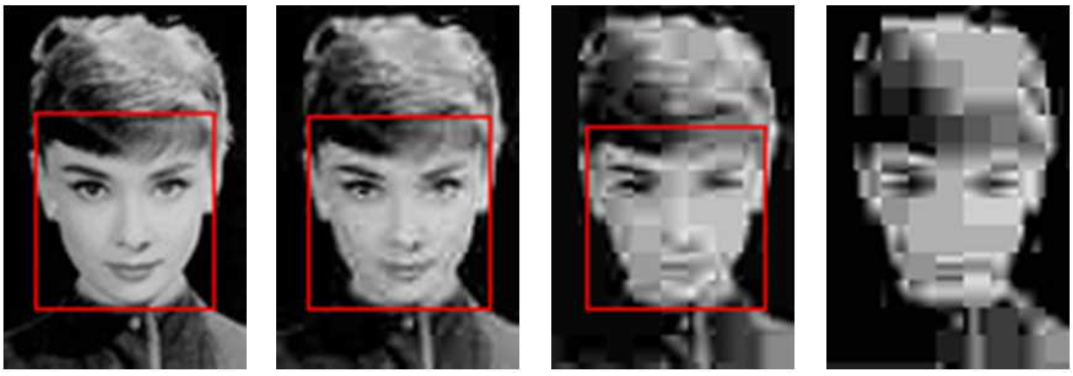


Figure 3.1: Example of how video degradation (JPEG compression) can affect video analysis algorithm (Viola-Jones face detection). Displayed image is degraded using JPEG quantizer values 100, 50, 25, and 9.

[1] Open Source Computer Vision library. More details can be found at http://opencv.willowgarage.com/wiki/.

[2] We use library by Independent JPEG Group (IJG)

We test this hypothesis on practical video that was shot in out lab using Axis 207 network camera (see a snapshot example in Figure 3.2). To reduce quality of the video, we use two different video adaptations, JPEG compression and nearest neighbor scaling. A combination of these two degradations is used as well: video is first scaled down to some value, then compressed, and scaled back up to its original size. Without going into the details about how we degrade video and compute accuracy of face detection (details are given in Section 3.2), we plot the rate-accuracy tradeoff in Figure 3.3. Different curves on the figure represent different ways the video is degraded.

Figure 3.2: Frame of the video used in experiments demonstrated in Figure 3.3. Network camera Axis 207 was used.

By looking at Figure 3.3, we can make one important observation. The accuracy of the face detection (index value in the figure) can be at the high level (same as for original video) even when the video is reduced to almost 10% of original size (0.1 ratio in the figure). Also, there are several ways to degrade a video while obtaining the same ratio between detection accuracy and the size reduction. Hence, the problem is to find the points on the curve that would correspond to the best possible algorithm's accuracy given the bitrate. These points actually form a pareto set of all the points in the figure. In Figure 3.3, pareto set points are marked with a triangle. The points from the pareto set is what comprises rate-accuracy function, since points corresponding

to low accuracy for the high bitrate can be discarded. The optimal point, corresponding to the highest accuracy given the smallest bitrate, can be approximated using lagrangian function or other convex optimization (Boyd & Vandenberghe, 2004) methods.

The problem is that such rate-accuracy tradeoff found for a given triplet: algorithm, video, and video adaptation cannot be simply generalized to other similar combination. There are many different video analysis algorithms and many ways to degrade video, which can affect the accuracy of algorithms differently. Also, the quality of the original video and some of its characteristics, such as lighting, background conditions, etc., can also affect the tradeoff between bitrate and accuracy. Therefore, in this thesis, we study the problem from different approaches. First, several video analysis algorithms that are not trivial and commonly used in practical applications are selected. Second, we employ several different video adaptations representing three components of video quality, SNR, spatial, and temporal. We also use standard test data as well as practical videos shot on our campus or from security network cameras installed in our lab. We obtain different rate-accuracy tradeoffs for various combinations of algorithm, video, and video adaptation. By studying these tradeoffs, we suggest a formalization of the tradeoff and several approaches on how to estimate it.

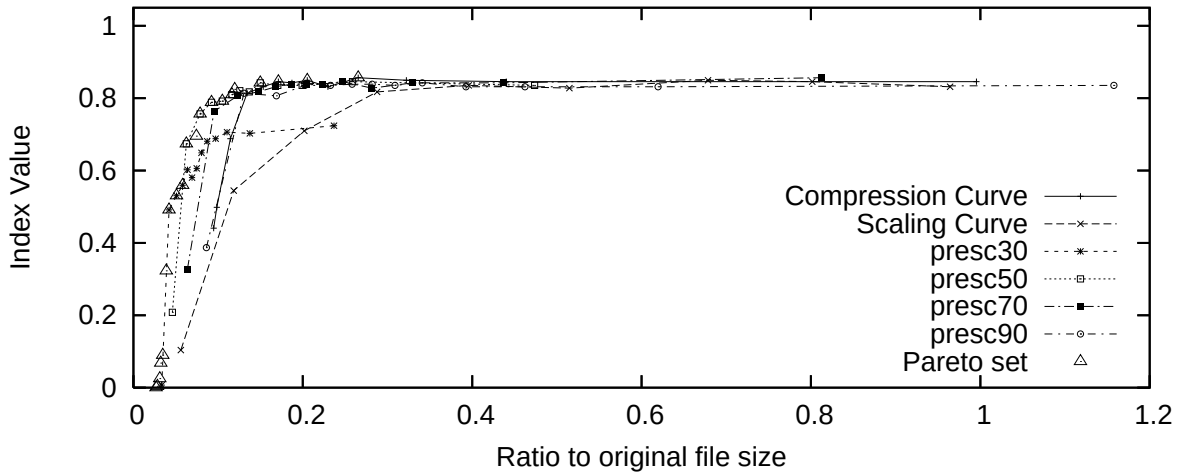


Figure 3.3: Accuracy of Viola-Jones face detection algorithm vs. compression and scaling adaptations, as well as their combination.

3.2 Overview of Experiments

Before proceeding to describe our results, we first explain in more details how our experiments were conducted. To determine the video quality requirements for a particular video analysis algorithm, we degrade the original video to a point when the accuracy of the algorithm drops significantly. We call such point a *critical video quality*, indicating the quality threshold above which the algorithm performs with its original accuracy. The video is degraded in small steps with a video adaptation, such as JPEG compression or frame dropping. It should be noted that, in this thesis, we understand the accuracy of an algorithm as a *relative* measurement. It is understood in terms of how much the accuracy changes when the video is degraded from its original quality to low quality.

We use the following video analysis algorithms in our experiments: OpenCV[3] implementation of Viola-Jones (Viola & Jones, 2001) face detection and Rowley face detection (Rowley et al., 1998) algorithms; QDA-based face recognition (Lu et al., 2003) algorithm; CAMSHIFT (OpenCV) face tracking (Bradski, 1998) algorithm, and blob tracking algorithm that relies on frame differencing and foreground object detection by Li *et al.* (Li et al., 2003). We picked these algorithms because they are freely available, fairly complex, and commonly used in various applications. Also, for face detection and face recognition algorithms, there are standard datasets with ground truth available.

3.2.1 Test Data

Dataset	**Characteristics**	**Short description**	**Algorithm**
MIT/CMU (subset A), snapshot Figure 3.4(a)	images with 168 faces of different sizes	various background/lighting conditions	face detection
Yale set, snapshot Figure 3.4(b)	images with 165 faces of 15 people, 320×240	various lighting conditions, different facial expressions	face recognition

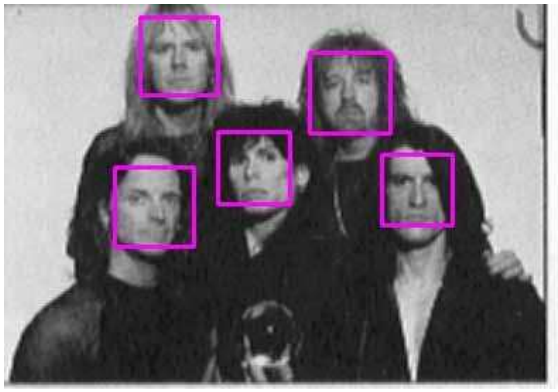
Continued on next page

[3] http://www.intel.com/research/mrl/research/opencv/

Dataset	Characteristics	Short description	Algorithm
Video of the lab door, snapshot Figure 6.9(a)	video, 22000 frames, 320×240, 5 fps, Canon VCC4 camera	surveillance of the door in a research lab	face detection, face recognition
Video in the lab, snapshot Figure 3.2	video, 543 frames, 320×240, 15 fps, Axis 207 camera	two people move in the frame facing the camera	face detection, face recognition
A person in front of the camera, snapshot Figure 4.7	300 frames, size 160×120, 5 fps, Axis 207 camera, 208 faces in total, 147 of which are frontal	a person moving in and out of the camera, making a face appear smaller or larger	face detection
Videos with moving faces, snapshot Figure 4.15(a)	video, 600 frames, 352×288, 30 fps, webcam	office settings, web-cam, face moves close-far from the camera	face detection, face tracking
News clips (SEQAM laboratory), snapshot Figure 3.4(d)	video, 200-350 frames, 320×240, 25 fps	talking heads, news anchors and interviewed stars	face tracking
Videos from PETS2001 dataset, snapshot Figure 4.18(a)	video, 150-250 frames, 384×288, 30 fps	outdoor car park, car/bicycle/people move across the frame	blob tracking
Video from VISOR dataset, snapshot Figure 3.4(f)	video, 150 frames, 320×256, 30 fps	outdoor car park, car moves across the frame	blob tracking
Campus videos, snapshot Figure 3.4(e)	video, 200-500 frames, 360×288, 25 fps, SONY HD camera	outdoor, cars on the road, a person walking in the field	blob tracking
Videos from RESCUE project, snapshot Figure 3.4(c)	video, 150-250 frames, 320×240, 8 fps, AXIS cameras	indoor, corridors, stairs, and halls, people walking	blob tracking

Table 3.1: Summary of datasets used in the experiments with different video analysis algorithms.

Datasets used in our experiments are summarized in Table 3.1. We use standard MIT/CMU and Yale datasets with provided ground truth for testing the accuracy of face detection and face recognition algorithms respectively. For face recognition, typically, the set of images is divided into gallery and probe subsets. Images in gallery have faces that are assumed to be known at the moment of recognition and images in probe set contain faces that are being recognized by the algorithm. To avoid bias in our recognition results, we divide the Yale dataset into



(a) Image from MIT/CMU dataset (Viola-Jones and Rowley face detection).

(b) Image from Yale dataset (face recognition).

(c) Snapshot from RESCUE videos (blob tracking).

(d) From database by SEQAM laboratory (CAMSHIFT face tracking).

(e) Shot on campus with hand-held camera (blob tracking).

(f) From VISOR video database (blob tracking).

Figure 3.4: Snapshot examples of videos used in our experiments.

four randomly generated pairs of gallery (36% images) and probe (64% images) subsets; our experimental results are obtained as average values corresponding to four subset pairs.

To test face detection and face recognition algorithms in practical scenario, we used two videos that we recorded to simulate an indoor video surveillance system. First video is a one hour long video of the door in our research lab that was recorded using Cannon VCC4 camera with the default video quality settings (320×240 resolution and JPEG compression 90). Among the recorded 22,000 frames, we manually marked 237 faces as ground truth, including 138 frontal and 99 profile faces. Second video is recorded in our lab and included two people walking in and out of the camera view. We manually marked 279 faces as the ground truth for face detection and face recognition algorithms. These two videos are used to verify critical video qualities estimated for face detection and face recognition algorithms using mutual information metric. Since face recognition algorithm is often used in combination with face detection, the frames with detected faces, including false positives, are used as the input probe faces. Verification metric (Grother, Micheals, & Phillips, 2003) was used to test accuracy of the face recognition. For each person in the test videos, one representative face-shot is selected and is used to verify the probe faces in the video. Basically, for each input face (which is the output of face detection), recognition algorithm compares it with the set of gallery faces (one face for each distinct person) to determine the best match (the one with the highest similarity score).

For face tracking algorithm, due to the lack of standard test videos, we use our own videos of a face captured with a web-cam (see a snapshot example in Figure 4.15). We also test face tracking on some movies and news clips, as in Figure 3.4(d).

Blob tracking algorithm is tested on several video fragments from standard video dataset PETS2001 and Visor database. Video sequences include moving cars, bicycle, and people. To make experiments easier and clearer, only single object tracking is performed. We also shot several videos of moving cars and people on the grounds of our university campus. Although Sony HD handheld camera was used, to make running time of the blob tracking practically feasible, the original HD video was resized to 360×288 resolution (more than 5 times reduction in resolution).

Video adaptation	Degradation pattern	Algorithm	Dataset
JPEG compression	quantizer from 1 to 99 with step 2	face detection, face recognition	MIT/CMU, Yale
	quantizer from 10 to 100 with step 10	face tracking, blob tracking	Videos with moving faces, PETS2001, VISOR, Campus, and RESCUE videos
Scaling (nearest neighbor, bicubic, pixel area relation)	2 to 100 percent of original size with step 2	face detection, face recognition	MIT/CMU, Yale, video in the lab
	from 25 to 100 with step 25	face tracking, blob tracking	Videos with moving faces, PETS2001, VISOR, Campus, and RESCUE videos
Combination of JPEG compression and scaling (nearest neighbor and bicubic)	Dcaling from 10 to 100 percent with step 10, for each step, compressing from 1 to 99 with step 2	face detection, face recognition	MIT/CMU, Yale, video in the lab
Frame dropping	drop i out of $i + j$ frames $(i, j = 15)$	face tracking, blob tracking	Videos with moving faces, news clips, PETS2001, VISOR, Campus, and RESCUE videos

Continued on next page

Video adaptation	Degradation pattern	Algorithm	Dataset
Combination of frame dropping and nearest neighbor scaling	Scaling from 25 to 100 percent with step 25, for each step, drop i out of $i+j$ frames $(i, j = 15)$	blob tracking	PETS2001, VISOR, campus, and RESCUE videos
Combination of frame dropping and JPEG compression	Compression from 10 to 100 percent with step 10, for each step, drop i out of $i+j$ frames $(i, j = 15)$	face tracking, blob tracking	Videos with moving faces, PETS2001, campus, and RESCUE videos

Table 3.2: Summary of video adaptations used in the experiments with different video analysis algorithms.

3.2.2 Video Adaptations

Table 3.2 summarizes video adaptations that we use to change different video quality for our video analysis algorithms.

SNR quality of images is degraded with IJG[4] implementation of JPEG compression algorithm. In this implementation, compression quality 1 corresponds to image with the highest compression ratio (the most distorted image) and 99 to image with lowest compression ratio (the least distorted image). In most of the experiments, images are degraded from quality 1 to quality 99 with step 2, leading to 50 different SNR qualities. In some experiments (such cases are indicated), to make them run faster, the step size is 4 instead of 2.

For videos, we degrade SNR quality using VirtualDub[5] open source video editor. Microsoft Video 1 codec is used to degrade videos for face tracking algorithm and Indeo video 5.10 codec is used for blob tracking algorithm. Originally, we used Microsoft Video 1, which adds heavy blocky artifacts when video quality is significantly reduced. It is not a problem for face tracking because the videos have large moving objects and blockiness has less affect on accuracy. However, in the most videos used to test blob algorithms, moving objects are small in size and such blockiness is not acceptable. Therefore, we chose Indeo video 5.10 codec instead because

[4] http://www.ijg.org/

[5] http://www.virtualdub.org/

it produces clearer videos with a similar bitrate. Video quality for face tracking ranges from 10 (higher distortion, low quality) to 100 (best quality) with step 10. Video quality for the blob tracker ranges from 10 to 100 with step 10.

Spatial quality of images is reduced using OpenCV implementations of nearest neighbor, bilinear, bicubic, and pixel area relation scaling algorithms. With a given scaling algorithm, we reduce spatial sizes of images from 100 to 2 percent of the originals with step 2 or, for some experiments, step 4. Then, we scale them back to the original sizes. For example, downscaling 320×240 pixels image to 60% results in 192×144 image, which, in turn, is scaled back to the original 320×240 size. Such downscaling-upscaling transformation can be used in a practical scenario of distributed video surveillance system, as demonstrated in Figure 3.5. The video with reduced spatial size is sent by a camera to a proxy through network. Upon receiving a video frame, the proxy upscales it to its original size and runs a video analysis algorithm. The downscaling-upscaling of the video stream allows us to reduce amount of data transmitted across the network link between camera and proxy/server, thus reducing the required bandwidth.

To scale videos, we used bicubic scaling algorithm that is built into VirtualDub video editor. The spatial quality ranges from 25 to 100 with step 25. The approach is similar to scaling images. he video is scaled down to a given percentage of its frame size and scaled back to the original.

The scaling adaptations can also be combined with SNR adaptation. We combine nearest neighbor scaling with JPEG compression in the following way. Images are first prescaled to several spatial sizes (20%, 30%, 40%, etc.) after which they are compressed with JPEG quantizer varying between 1 and 99 with step 2. Then, images are decompressed and scaled back to their original spatial sizes. Compressing downscaled video frame allows achieving even higher reduction in bitrate at the expense of receiving frame with higher distortion at the proxy (in the example shown in Figure 3.5).

To degrade temporal video quality, we reduce the video frame rate by dropping frames using drop pattern: "drop i out of $i+j$ frames" (see Figure 1.3 for illustration). We vary i and j from 1 to 14. The value i represents the gap between frames, and j represents how many consecutive frames remain. For example, if we drop every third frame, i equals to 1 and j to 2; when

three consecutive frames out of nine frames are dropped, i is 3 and j is 6. Note that while these two patterns give the same average frame rate, the accuracy of the tracking algorithm can be different. We also combine SNR and scaling video adaptations with frame dropping to determine how such combination affects accuracy of face tracking.

3.2.3 Algorithms Accuracy

To evaluate accuracy of face detection, we compute the detection index as follows. For each JPEG quality, the number of detected faces is recorded. Using available ground truth, we obtain the number of correctly detected faces and divide it by the recorded total number of faces to get the detection index. We also note the number of faces that are wrongly detected by the algorithm. Dividing this number by the total number of faces, we obtain the false positive index of face detection.

In experiments with Yale dataset, the identification task of face recognition algorithm is evaluated using the standard performance metric, rank one of cumulative match characteristic (CMC) (Grother et al., 2003). CMC rank one value is computed for images from the probe set only. In experiments with practical surveillance video, we evaluate the recognition algorithm by using the standard verification performance metric instead.

We compute the accuracy of tracking algorithms as follows. The mean distance between the center of the tracked object (face, blob, or human) in degraded video (with applied drop pattern) and the center of the face in the original video is recorded. We use this mean distance (in pixels) as a metric of accuracy for the tracking algorithm. Essentially it measures the error of tracking across all frames in the video, therefore, we call the metric *average error*. For a given dropping pattern, smaller average error means better relative accuracy of the algorithm.

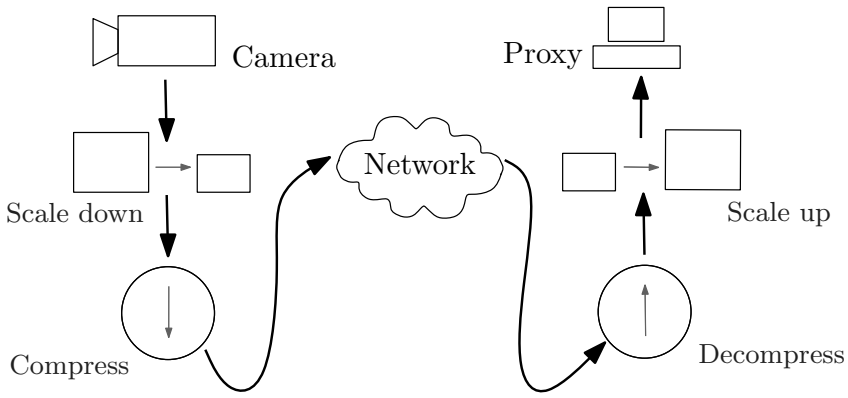


Figure 3.5: Video surveillance scenario of combining scaling and compression adaptations to further reduce bitrate.

Chapter 4

Finding Critical Video Quality

In this chapter, we present experimental results of finding critical video quality for several examples of video analysis algorithms. The experiments aim to demonstrate how different types of video degradation affect accuracy of different video analysis algorithms. The emphasis is made on the following pairs: SNR adaptation (JPEG compression) and face detection; spatial adaptations (various scaling algorithms) and face recognition; and temporal adaptation (frame dropping) and blob tracking.

4.1 Face Detection

We investigate how the accuracy of Viola-Jones and Rowley face detection algorithms change when SNR quality of the video is reduced. Viola-Jones algorithm is an object detection algorithm that uses a cascade of classifiers based on Haar-like features (see Figure 4.1). Intuitively, it should perform accurately as long as images contain such features. Rowley algorithm is based on the statistical changes in intensity values across a given image. Those regions that reflect the patterns collected through algorithm's training are marked as a face. We present experimental findings showing changes in accuracy of these two algorithms for degraded SNR quality (see Section 3.2 for more detailed description of experiments).

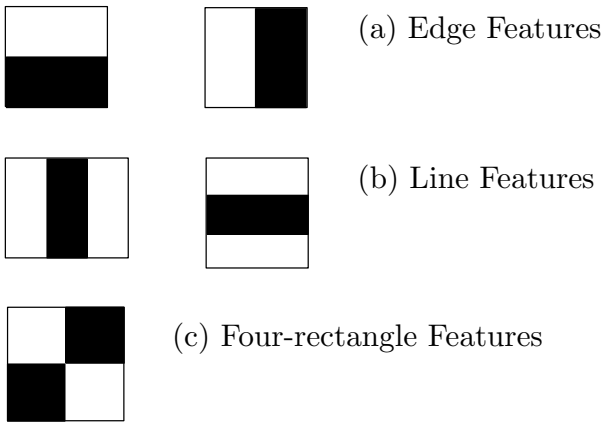
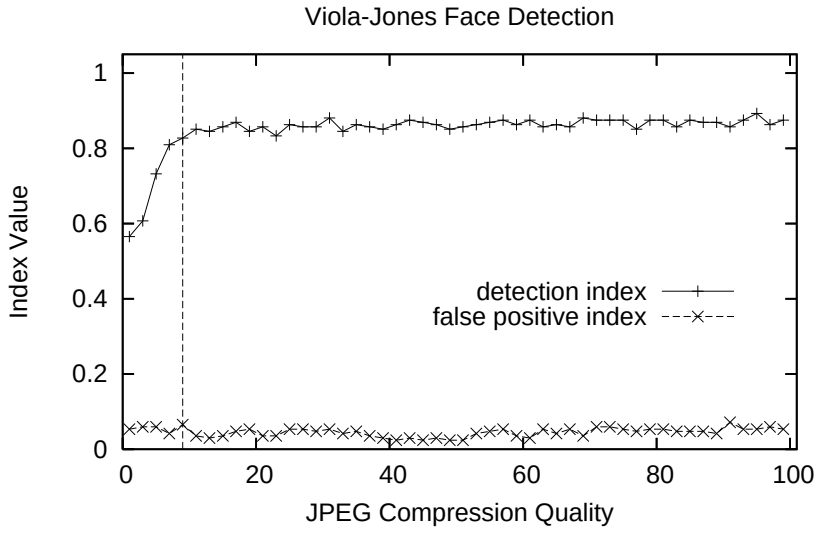


Figure 4.1: Haar-like features used by Viola-Jones face detection algorithm.

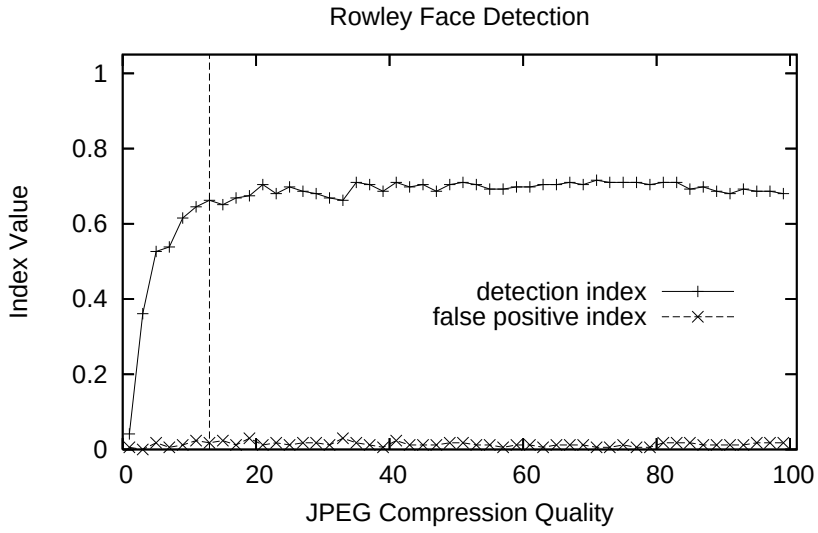
4.1.1 SNR quality

The experimental results for Viola-Jones algorithm and MIT/CMU dataset are presented in Figure 4.2(a). The figure shows both detection and false positive indexes of the face detection algorithm against different compression qualities. It can be noted that the average accuracy of the face detection algorithm does not change significantly when JPEG compression quality is decreased from 99 to 9 (indicated with the dashed vertical line on the figure). For quality less than 9, the detection index demonstrates a sharp decrease. Since 90-95 is the default JPEG compression quality used in typical video surveillance cameras (e.g., Axis 207, Canon VCC4), compressing images to quality 9 can lead to significant reduction in size. Also note that the false positive index does not increase in response to reduced compression quality, which means that only the detection index is affected. Therefore, we can transmit video frames compressed with quality 9 and achieve similar detection results as with uncompressed video. If we conservatively choose 20 as the critical compression quality, we find that the average file size of JPEG images in the MIT/CMU data set is 15.8 KB compared to 135.6 KB for original images (a nine times reduction in size). This reduction, however, does not directly apply to a normal video, since video encoders typically use motion estimation between frames to achieve higher compression.

The effect of JPEG compression on accuracy was also tested for Rowley face detection algorithm. Results, presented in Figure 4.2(b), demonstrate that this algorithm is generally less accurate compared to Viola-Jones algorithm (see Figure 4.2(a)). Nevertheless, the detection



(a)



(b)

Figure 4.2: Accuracy of face detection algorithms vs. JPEG compression quality.

index of Rowley algorithm shows the same pattern of being at its original level until JPEG compression quality is reduced to value 13 (indicated with the dashed vertical line in the figure). Conservatively, the critical compression quality can also be chosen as 20. False positive of Rowley algorithm is lower than Viola-Jones algorithm and it is also not affected by the decrease in compression quality.

Figure 4.2(a) and Figure 4.2(b) demonstrate that both face detection algorithms have noticeable fluctuations in the detection index. The main reason for such fluctuations in the

detection lies in the reliance of the Viola-Jones and Rowley algorithms on different threshold values, which are empirically obtained through offline training of their classifiers. These values affect the detection sensitivity of algorithms to the faces in the input images. Slight changes in the pixel values of an image due to compression can unpredictably affect the decision of the algorithm on faces that are near the threshold. Also, these algorithms are sensitive to factors such as face size, lighting, background conditions, etc. These factors can cause the algorithm to oscillate between detecting and not detecting faces as we vary the compression quality. To analyze this type of behavior, we compute the cumulative distribution function (CDF) on the maximum JPEG quality which causes the face detection algorithm to fail. The resulting CDF for Viola-Jones algorithm is shown in Figure 4.3 (the results for Rowley algorithm are similar).

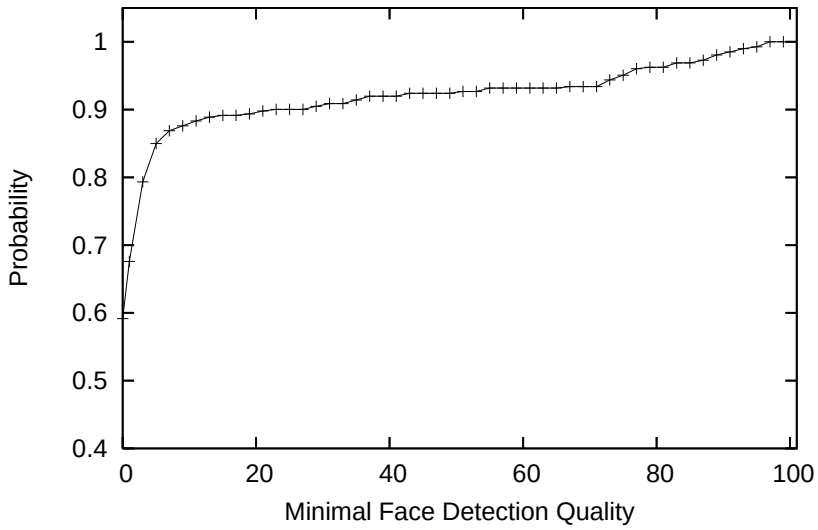


Figure 4.3: CDF for minimal face detection quality. Viola-Jones face detection.

The CDF curve in Figure 4.3 is computed as follows: For each face, we find a compression quality q such that the face is not detected when compressed with quality q, but is detected when compressed with any quality larger than q. We call this compression quality the *minimal face detection quality*, and regard it as a random variable for Figure 4.3.

The CDF function shows relatively large decline (about 9% or 45 faces) when compression quality decreases from 100 to 20. This figure demonstrates inconsistent detection results for 9% of the faces. Combining this result with observation from Figure 4.2, we can deduce that in this subset of faces, the faces are constantly changing from being detected to not being detected as

we vary the quality. For instance, one particular face is detected for compression quality 77, 76, 74 and 73, but is not detected for value 75. These faces are found to be of a smaller size. We re-plot Figure 4.3 for different face size, measured as the maximum of distances between left eye and right eye, and between eye-line and mouth. The new plot, Figure 4.4, shows that the algorithm tends to maintain more consistent detection behavior for faces that are larger than 30 pixels, as we can see from the noticeable drops of the probability value for the faces with size less than 40 pixels.

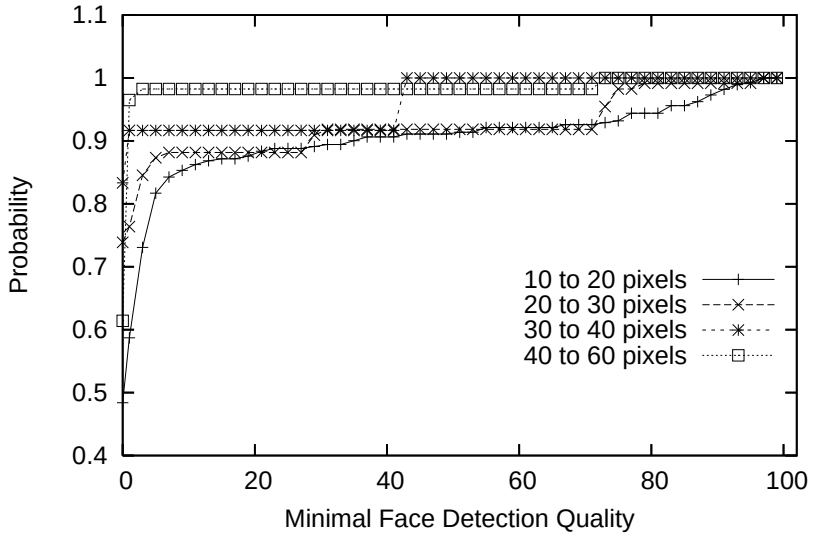


Figure 4.4: CDF for Minimal Face Detection Quality for Different Face Size. P=3, T=-0.0001. Viola-Jones face detection.

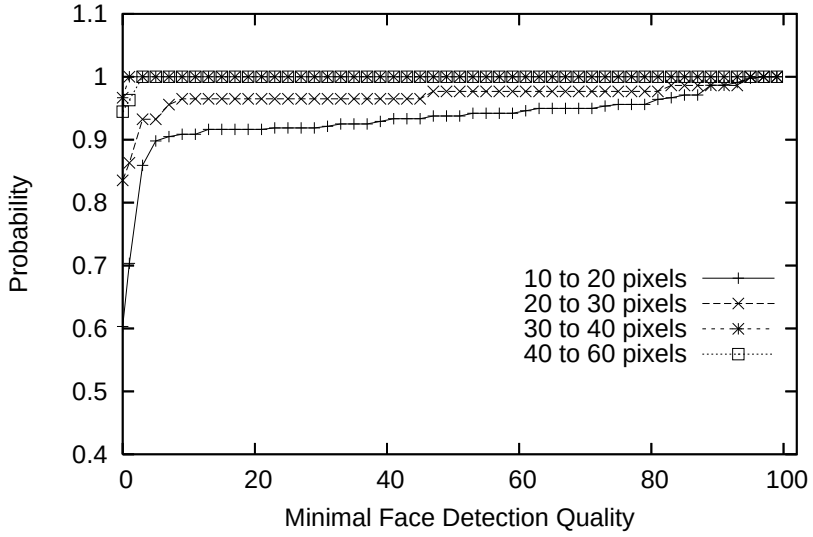


Figure 4.5: CDF for Minimal Face Detection Quality for Different Face Size. P=4, T=-1.0.

The reason for the fluctuations in detection lies in the reliance of the algorithm on different threshold values. These values affect the detection sensitivity of the algorithm for the faces in the input images. Slight changes in the pixel values of an image due to compression can unpredictably affect the decision of the algorithm on faces that are near the threshold. Further experiments support this explanation: We increase absolute values of two main threshold parameters: the pruning value P for face candidates and a threshold T that is used inside the cascade classifier. Value P is used as follows: an image area, candidate to be a face, is discarded if the number of times it is detected by the cascade of classifiers is less than pruning value P, which means that increasing P leads to classifying lesser image areas as faces. On the other hand, increasing absolute value of threshold T, which is used inside the decision tree of each classifier, leads to many more areas of the image to be marked as possible faces by the cascade. We change these values from the default values of P=3 and T=-0.0001 to P=4 and T=-1.0. Figure 4.5 shows the new results. The consistency of detection across different compression quality values improves. The parameters, however, increase the sensitivity of the algorithm, causing the number of false positives to increase by five times. Interestingly, using the new parameters, the number of false positives is not significantly affected by compression quality.

To strengthen our results, we also run Viola-Jones algorithm on the surveillance video of our lab door (see Section 3.2 for more details). Video is compressed to two different quality values, 90 and 20, using two sets of threshold parameters $P = 3, T = -0.0001$ and $P = 4, T = -1.0$.

	P=3,T=-.0001	**P=4,T=-1.0**
detection index		
quality 90	0.63	0.77
quality 20	0.61	0.76
false positive index		
quality 90	0.004	0.09
quality 20	0.01	0.11

Table 4.1: Experiments with Face Detection Algorithm and Actual Surveillance Image Set of 237 Faces.

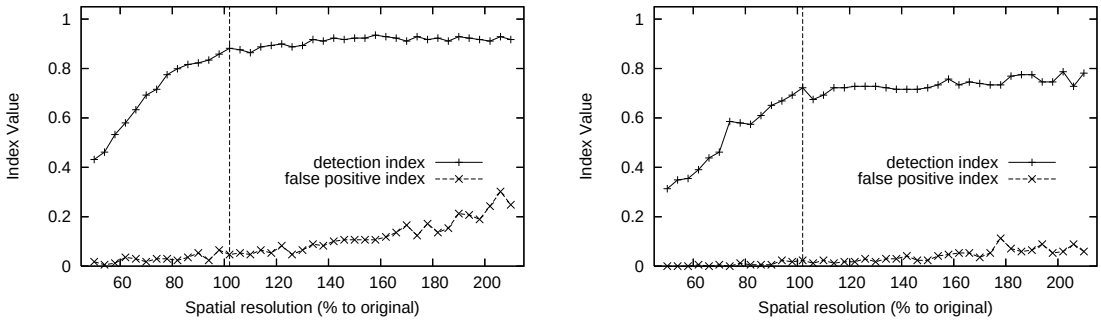
From the experiments, we find that the Viola-Jones algorithm exhibits unpredictable behavior temporally, returning false positive results periodically. We exclude such false positives by considering a face as detected only if it is detected consecutively for three frames. This

method is reasonable because faces are usually present in a video in a consecutive sequence of frames (assuming sufficiently high frame rate). Table 4.1 presents the detection and false positive indexes for the algorithm with the two sets of parameters. This result verifies that both the detection and false positive indexes do not change significantly when compression quality is reduced to 20. The results obtained for real surveillance data are consistent with our findings on the MIT/CMU data set.

4.1.2 Scaling quality

Spatial adaptations are another set of adaptations affecting the accuracy of face detection algorithms. The details on how we scale the test images for Viola-Jones and Rowley algorithms are given in Section 3.2. It is important to note that a typical learning-based face detection algorithm has a limit on a minimal detectable size of a face. For example, the limit for Viola-Jones algorithm is 20×20 pixels, which is set during the training stage of the face classifier. Therefore, if the image containing face is down-sampled by scaling algorithm to such extend that the size of the face falls below this limit, the detection algorithm will not be able to find this face. That is why it is often not reasonable to perform a simple down-sampling of the images, as we did with SNR quality by simply reducing it.

There are two practical cases of using scaling adaptation with face detection. First case is when face sizes in a video fall under the minimal detectable limit of the face detection. To detect the faces, we can up-scale video frames until faces become large than the detectable limit. This approach would be useful if videos sensors are cheap with low spatial resolution and no optical zoom. To be practical, however, we need to know by how much to increase the size of the frames. We can compute it via offline profiling of the typical face sizes at the given video sensor. Another practical case of using scaling adaptation is reducing the bandwidth of the streaming video from video sensor to proxy that runs face detection. We proposed to down-scale images first and up-scale them back to original size, with understanding that in practical situation, downscaled small size images would be transmitted through network, while scaled-back images would be fed to a detection algorithm (see Figure 3.5 for the illustration).



(a) Viola-Jones face detection

(b) Rowley face detection

Figure 4.6: Accuracy of Viola-Jones and Rowley algorithms when MIT/CMU images are scaled with nearest neighbor to various spatial resolutions.

To see the effect of up-scaling and down-scaling on face detection, we plot in Figure 4.6 accuracy results of Viol-Jones and Rowley algorithm vs. different spatial sizes of MIT/CMU images. We used nearest neighbor scaling because it is one of the simplest scaling algorithms. Spatial size of the images varies from 50% (in both width and height) of their original size to 200% of their original size. From the figure, we can note that accuracy of the algorithms increases slightly when images are larger than the original size. This observation indicates that some of the faces in the dataset were smaller than the face size minimal detectable limit. The important observation is, however, that with increase in the image sizes, the false positive index increases significantly. This means that using spatial scaling can give increase in accuracy but also results in the increase of false positive.

To understand how up-scaling of the video frames can improve detection accuracy in practical applications, we conducted experiments with a surveillance video of a person moving in front of the camera. It was shot with Axis 207 in a typical lab-like environment (see Table 3.1 for more details). The person moves his head with a lot of motion, moving in and out of the camera view, moving hands in front of the face, turning his head in all directions. Due to a small frame size, which implies a low cost, low power video sensor, for example a cheap mobile phone, many facial shots are below the minimal detectable limit (20×20 pixels) of Viola-Jones algorithm. From the original video (160×120 pixels resolution) we also derived two videos with 224×168 and 256×192 frames sizes by up-scaling it to 140% and 160% respectively. We ran Viola-Jones detection algorithm on these three sets of frames and detection results are presented

Figure 4.7: Examples of Viola-Jones detection for different resolutions of the practical video.

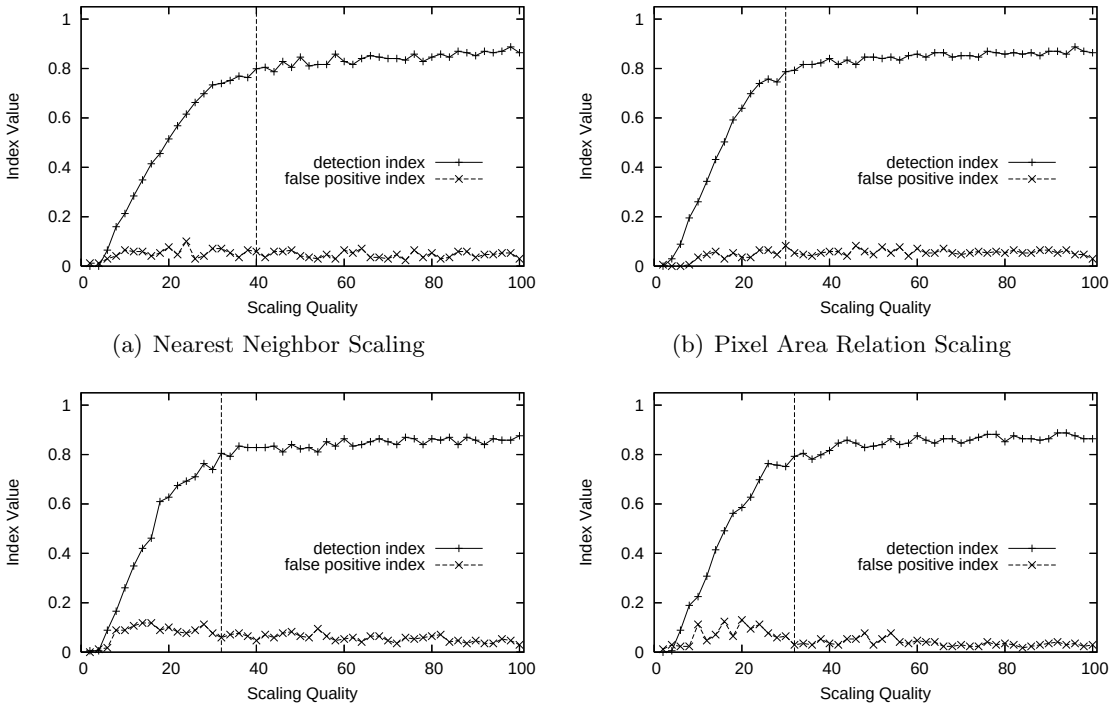
in Table 4.2. Snapshots of videos with all different frame sizes are shown in Figure 4.7.

From the Table 4.2 we can notice a significant improvement in the detection index caused by up-scaling the original low resolution video. Since many of the faces in the original video were too small for the detection algorithm to recognize, the increase in spatial resolution allowed to detect more faces. The detection accuracy for video scaled to 160% is three times higher than for original video. Such results provide practical benefits for using spatial scaling for low resolution video sensors. As with MIT/CMU results, Table 4.2 also shows an increase in false positive for up-scaled videos.

When the resolution of video sensors is high and video is transmitted over network before it is analyzed, the problem of reducing video bitrate is important. Therefore, we can propose to first down-scale video frames, transmit them over network, and then up-scale them back to the original size. Firgure 4.8 and Firgure 4.9 demonstrate how down-sampling-up-sampling of MIT/CMU images with different scaling algorithms affects accuracy of Viola-Jones and Rowley face detection respectively. We can notice that detection index and false positive show similar behavior across all scaling algorithms. Arguably, nearest neighbor scaling gives the worst accuracy results for both face detection algorithms, since the corresponding detection index inhibits a more rapid decline compared to other scaling algorithms. In these experiments, scaling qual-

	Original	**140% scaled**	**160% scaled**
detection index	48 faces	119 faces	146 faces
frontal faces	0.33	0.80	0.99
all faces	0.23	0.57	0.70
false positive index	0 faces	1 face	4 faces

Table 4.2: Up-scaling 160×120 video to higher spatial size for Viola-Jones face detection to notice small faces.



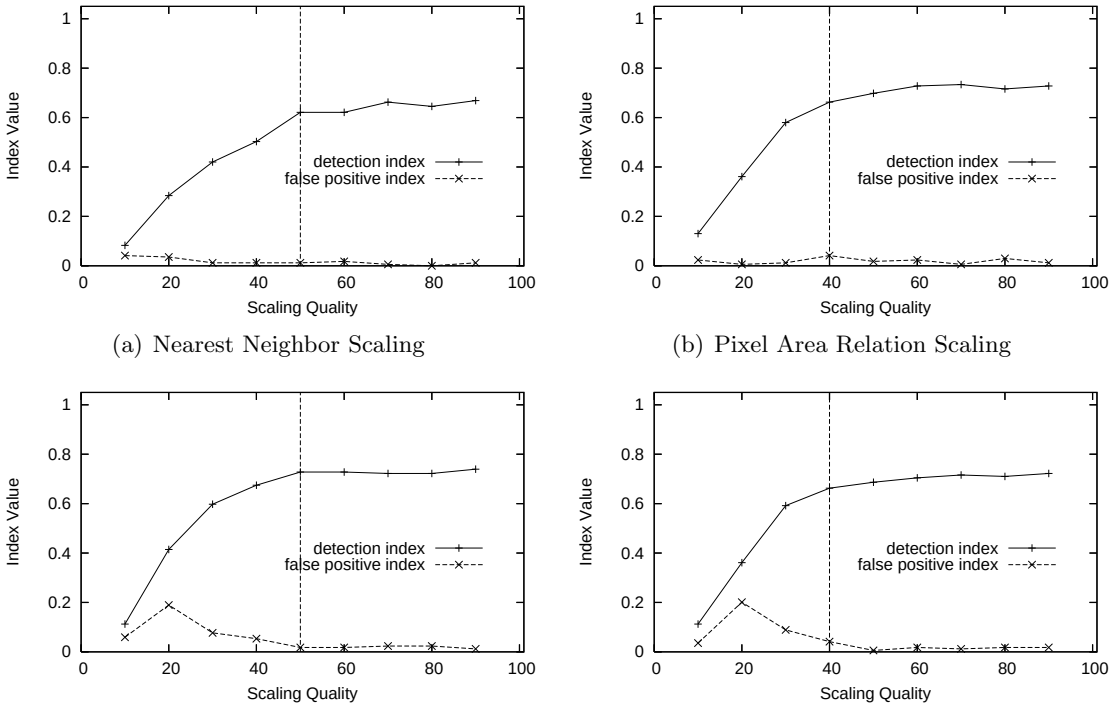
(a) Nearest Neighbor Scaling

(b) Pixel Area Relation Scaling

(c) Bilinear Scaling

(d) Bicubic Scaling

Figure 4.8: Degrading scaling quality for Viola-Jones face detection, MIT/CMU dataset.



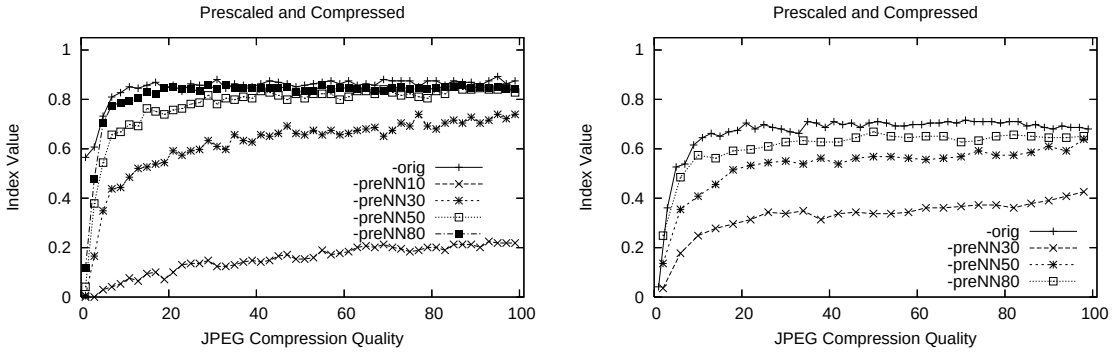
(a) Nearest Neighbor Scaling

(b) Pixel Area Relation Scaling

(c) Bilinear Scaling

(d) Bicubic Scaling

Figure 4.9: Degrading scaling quality for Rowley face detection, MIT/CMU dataset.



(a) Viola-Jones face detection (b) Rowley face detection

Figure 4.10: MIT/CMU images are prescaled with nearest neighbor and compressed with JPEG for Viola-Jones and Rowley algorithms.

(a) (b)

Figure 4.11: The effect of image down-scaling (to 30%) followed up by its up-scaling to original size. The image is from MIT/CMU dataset. Nearest neighbor scaling is used.

ity is the percentage in size (width×height) to which images were down-scaled first and then up-scaled back from. Such process distorts images in a similar fashion as JPEG compression does. The example of the image degraded in such way is shown in Figure 4.11. Looking at the figures 4.8 and 4.9 we can chose the critical spatial quality for each scaling and face detection algorithm as the beginning of more rapid decrease in detection index. We indicated the such points with vertical dashed lines on the figures and also summarized the them into a Table 4.3 together with the corresponding reduction in bitrate (average size of degraded images compared to file size of original un-scaled images). The bitrate reduction data was computed for the surveillance video in the lab shot with Axis 207 camera (see Table 3.1).

To reduce the size of the images even further, we degrade them using the combination of

Scaling Algorithms	Viola-Jones		Rowley	
	Critical Quality	Bitrate Reduction(times)	Critical Quality	Bitrate Reduction(times)
Nearest neighbor	40	3.5	50	2.5
Bilinear	38	3.8	50	2.8
Bicubic	35	3.87	40	3.5
Pixel area relation	30	5.7	45	3.17

Table 4.3: Critical spatial qualities and corresponding reduction in bitrate for several scaling algorithms and Viola-Jones and Rowley face detection.

scaling and SNR adaptations. A suitable scenario could be as following: images are prescaled to a certain spatial quality, compressed with certain SNR quality, transmitted over network, decompressed, scaled back to their original spatial quality, and used as input to video analysis algorithm. We prescaled MIT/CMU images with nearest neighbor scaling to several quality values. We chose nearest neighbor scaling since it demonstrated the worst critical spatial quality among other scaling algorithms. The detection results for Viola-Jones and Rowley algorithms for the combination of scaling and compression are presented in Figure 4.10. Curve marked as "-orig" corresponds to the images with no prescaling and is identical to results in Figure 4.2. Curve "-preNN50" corresponds to mages that were prescaled to 50% of their original size with nearest neighbor scaling. Other curves are marked likewise. We can notice that shapes of the curves in this figure are similar to the one from experiments with JPEG compression. The prescaling however, has a negative effect on overall accuracy. Even when images are prescaled to 80%, the accuracy of both face detection algorithms is noticeably lower than for images without scaling adaptation applied.

It is hard to find a definitive and quantitative answer to why our tested face detection algorithms remain accurate for highly compressed or scaled images. Intuitively, algorithm's accuracy depends on what type of features it searches for in an image and how it performs the search. The type of distortions, caused to video/image by reduction in quality, affects algorithms' accuracy as well. For instance, Viola-Jones algorithm is based on Haar-like features, which are affected relativelly insignificantly by the strongest artifact of JPEG compression, as compared to an edge detection algorithm. The design of the algorithm, however, plays very important role as well. Many modern algorithms (including Viola-Jones and Rowley algorithms)

are based on empirical training using a large pool of real-life images with faces of various qualities, shapes, and scales. Therefore, thresholds and pruning values obtained in the training stage have a strong affect on algorithms' accuracy as well as their robustness to reduction in video quality. In Chapter 5 and Chapter 6 we discuss this issue in more details.

4.2 Face Recognition

The accuracy of QDA-based face recognition algorithm (Lu et al., 2003) is evaluated for the following spatial video adaptations: nearest neighbor and pixel area relation scaling algorithms (see Table 3.2). The results are presented in Figure 4.12(a) and Figure 4.12(b) respectively. Similarly to the accuracy pattern of face detection algorithms, the accuracy of face recognition does not change until video quality is reduced to a critical spatial quality. As indicated with dashed vertical lines in the figures, for nearest neighbor scaling algorithm the critical quality is 20% of the original images sizes and for pixel area relation it is 11%. On average, for images from Yale set, these qualities reduce file sizes to 9.9% of their original sizes (10 times reduction) for the nearest neighbor scaling algorithm and to 4.2% (24 times reduction) for the pixel area relation scaling.

Similarly to face detection algorithms, face recognition remains accurate for images with significantly reduced quality. We use QDA-based recognition, which relies on a quadric offline-trained classifier to determine whether two input faces belong to the same person or not (a categorization task). Detection and recognition algorithms are designed to perform their task accurately on data (images or video) with large amount of noise. Visual artifacts of compression and scaling can be regarded as noise added to an image. Therefore, since detection and recognition algorithms are designed to be prone to noise, they are robust on video that is highly compressed or scaled.

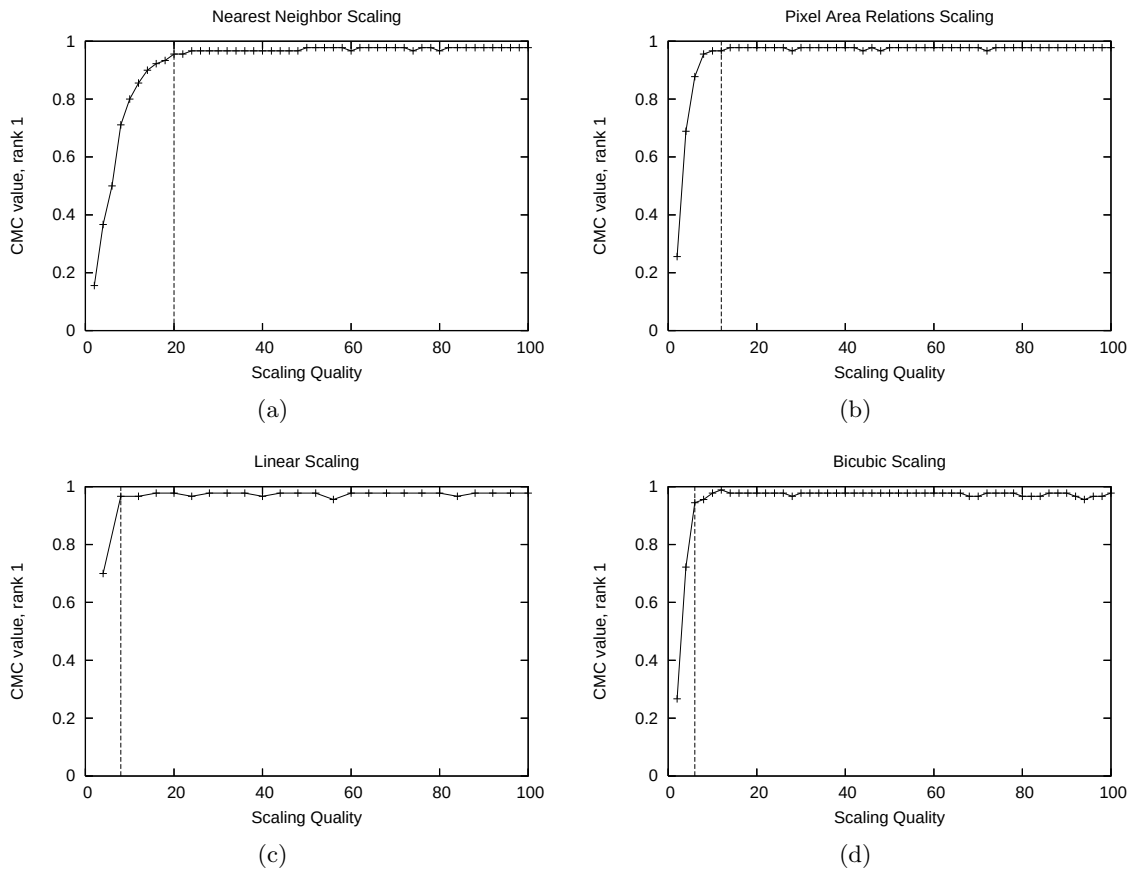


Figure 4.12: Identification CMC value of face recognition vs. scaling quality of scaling and JPEG compression algorithms.

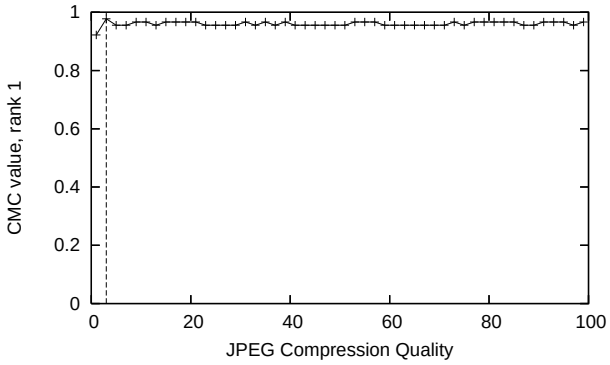


Figure 4.13: Identification CMC value of face recognition vs. JPEG compression algorithms.

4.3 Face Tracking

In this section, we study the trade-off between accuracy of implemented in OpenCV library CAMSHIFT (Bradski, 1998) face tracking algorithm and two qualities of the video, temporal and SNR. We run the tracking algorithm on video with different frame dropping patterns and

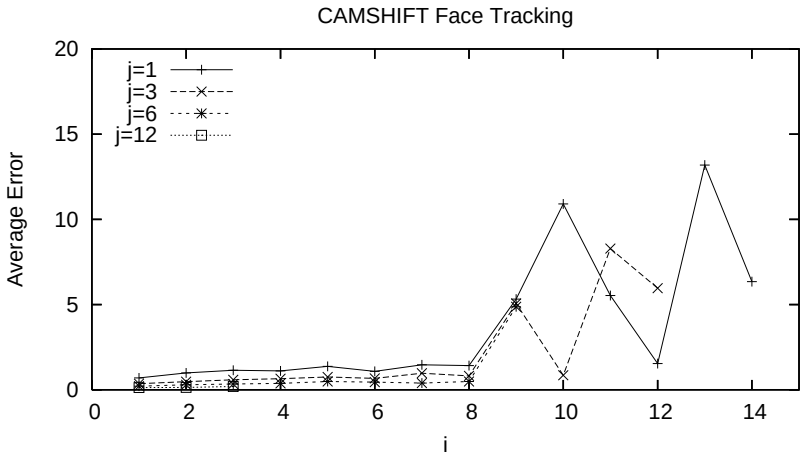
compute tracking average error as described in Section 3.2. We also test face tracking for different compression qualities.

Figure 4.14(a) shows the average error for one of the test videos for patterns with i varying from 1 to 14 and j equal to 1, 3, 6 and 12. The figure shows that drop gap i plays a more important role in the accuracy of the tracking algorithm compared to j. We can see from the figure that accuracy is consistent with increase of i and decrease of value j. Only when gap i is more than 8, the algorithm shows unpredictable behavior; we call this drop gap a *critical drop gap*. The reason for unpredictable behavior is that CAMSHIFT algorithm searches for a given object's histogram inside a subwindow of the current frame of the video, which is computed as 150% of the object size detected in the previous frame. Therefore, if the object, moves between two frames from its original location for a distance larger than half of its size, the algorithm will lose the track of the object. With another drop gap, the face may be able to move out and move back into the search subwindow. Hence, the oscillations in the algorithm's accuracy occur.

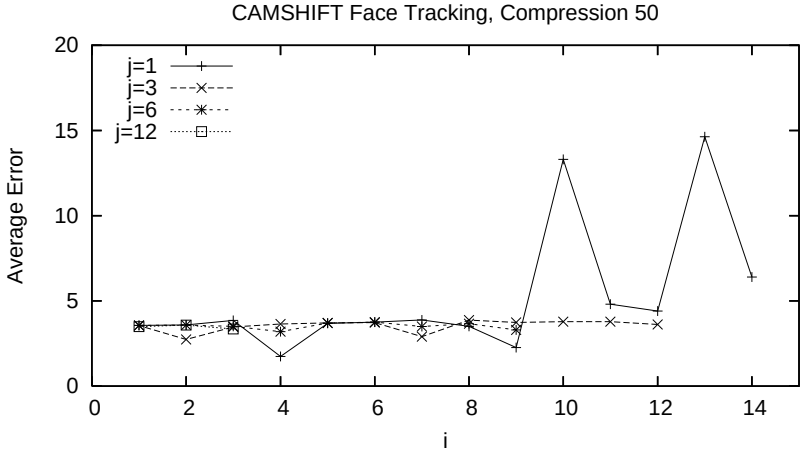
Such observations demonstrate the significance of gap i for the accuracy of the face tracking algorithm. In the video used for Figure 4.14(a), i should be bounded by 8 for the tracking to be consistently accurate. Therefore, the algorithm can achieve reasonable accuracy (within two pixels) using the pattern: "Drop 8 frames out of 9 frames." In other words, the video source only needs to send at 1/9 of the original frame rate.

Next, we study the effect of SNR quality on the accuracy of face tracking. We compress the video with different compression qualities and repeat the experiments with frame dropping pattern. The results for video with compression quality 50 are shown in Figure 4.14(b). We can see that accuracy is lower on average for video of higher compression ratio. An increase in compression ratio leads to an increase in average face distance ratio since highly compressed video has fewer details, making the border of a tracked face less distinct. Figure 4.15(b) shows the effect of the compression with quality 50 using a frame sample from the test video.

The results reported above come from experiments on a single video, captured using a web-cam in a normal office environment. We repeat the experiments for different videos with different



(a)



(b)

Figure 4.14: Average error vs. drop gap for CAMSHIFT algorithm. Video was compressed to quality 100 in 4.14(a)and quality 50 in 4.14(b).

content and notice that for a movie clip with talking person, moving his hands occasionally, the critical drop gap is 14, even when the video was compressed with quality 10. On the other hand, for a movie clip showing a character moving his head constantly in a fast and jumpy motion, the critical drop gap is found to be 4 ("drop 4 out of 5 frames" pattern). We also run experiment with web-cam video captured in different lighting conditions. The critical drop gap found for various videos and different compression qualities is plotted in Figure 4.16. The figure shows that compression quality does not significantly affect accuracy of the face tracking algorithm, hence, the type of face motions is a major video constraint for the accurate tracking. Face tracking algorithm is resistant to video compression because it is based on histogram

(a) (b)

Figure 4.15: A snapshot frame from a test video for CAMSHIFT face tracking. In (a) it is compressed with quality 100 and in (b) with quality 50.

matching. Since DCT-based compression removes high frequencies from a video, it does not have a significant effect on the histogram of a face.

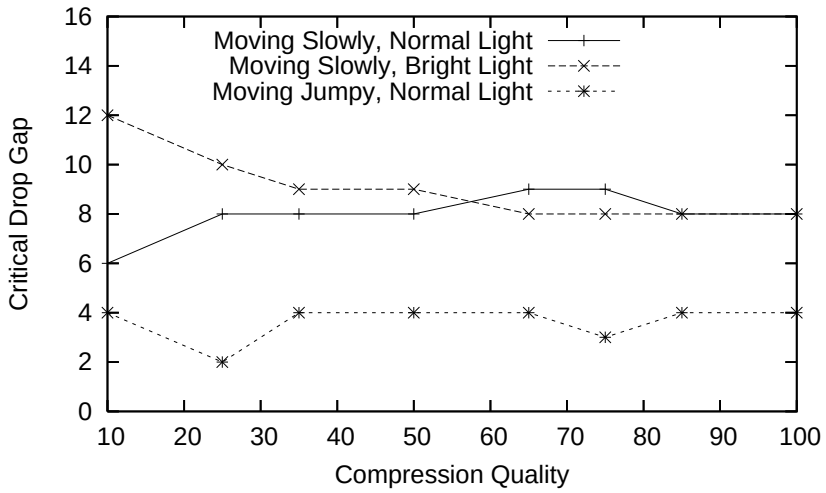


Figure 4.16: Critical drop gap vs. compression quality.

4.4 Blob Tracking

Similarly to face tracking in Section 4.3, we performed experiments with the blob tracking algorithm that is based its tracking on the frame differencing. We use a fragment of PETS2001 video and VISOR video. See snapshots in Figure 4.18(a) and respectively Figure 3.4(f). Unlike face tracking, failure of the blob tracking algorithm is harder to define. If too many consecutive

frames are dropped, the object in the current frame appear so far away from its location in the previous frame that the frame differencing operation results in detecting two separate blobs (see Figure 4.18(b)). Therefore, strictly speaking, the algorithm would not loose tracked object because of the frame dropping. Instead, it gets wrongly confused that a new object entered a scene. For typical original video with 30 fps, dropping even 15 consecutive frames may not result in splitting the tracked object into two blobs (at least for our test videos). Instead, the more frames are dropped, the larger becomes the tracked object. This observation is demonstrated by Figure 4.19(a) and Figure 4.19(b), which show the drop gap vs. average error of the blob tracking. As we can see, as oppose to CAMSHIFT tracking, blob tracking does not lose object even for large drop gap, but the error grows rapidly. Although, technically blob tracking algorithm does not lose an object, its accuracy shows a significant drop for low frame rate video. It appears that we cannot guarantee algorithm to perform with original accuracy when we reduce frame rate. However, we realize that with simple reasoning based on tracked object's and algorithm's features it is possible to adapt algorithm to low frame rate without significant loss of the accuracy. We discuss the analysis and the approach in Section 7.1.

We also test the effect of SNR quality on accuracy of blob tracking using PETS2001 video. The results of blob tracking for videos compressed with qualities 10 and 20 (Indeo video 5.10 codec) are presented in Figure 4.20(a) and Figure 4.20(b). As we can see, the frame dropping is still applied. This was done to understand how lower SNR quality changes algorithm's accuracy for different frame rates. We can notice that the accuracy for compressed videos shows a similar pattern to the original video, which means the accuracy is affected insignificantly by the compression. Such results are expected since blob tracking algorithm only detects and tracks foreground object represented as blobs of pixels. Since compression removes fine details from each frame, the blobs can still be tracked.

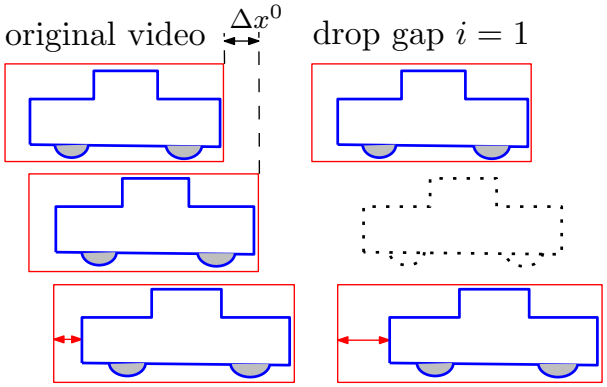
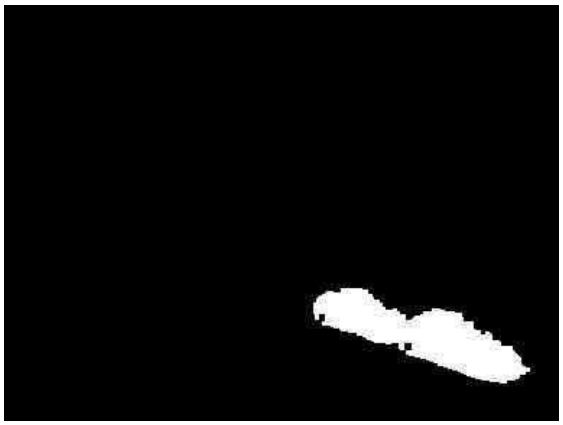


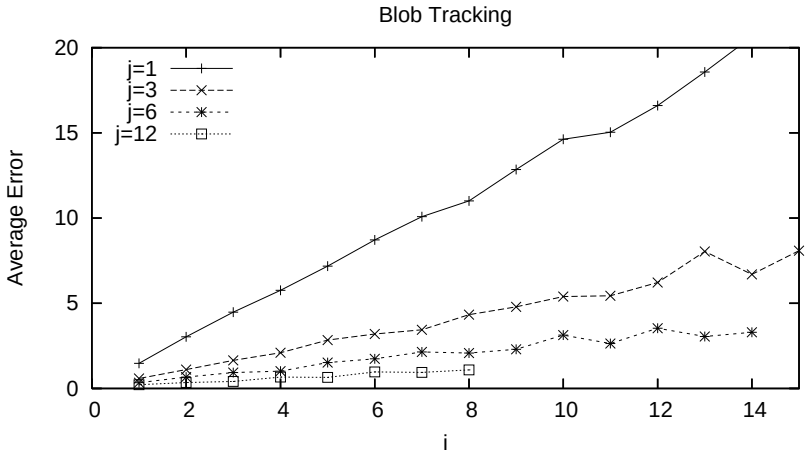
Figure 4.17: The schema of the difference between object foreground detection for original video and for video with dropped frames.

(a) Detected foreground object with drop gap 14 frames. PETS2001 video.

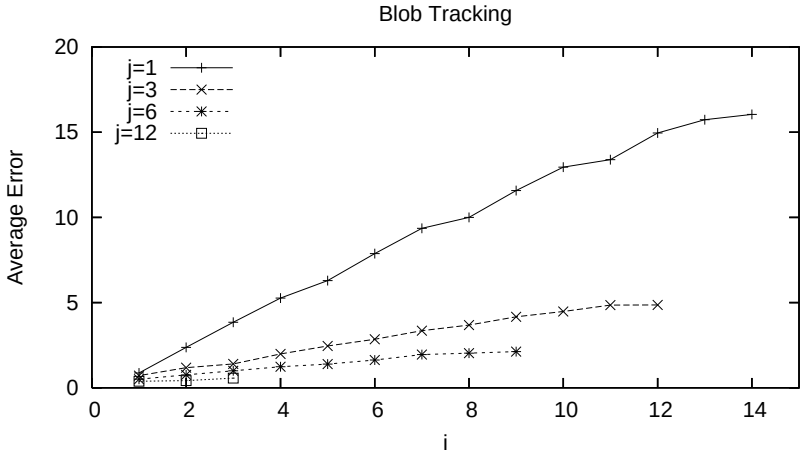


(b) Binary mask of the frame in 4.18(a). Effect of drop gap on frame differencing.

Figure 4.18: The foreground object detection based on frame differencing.

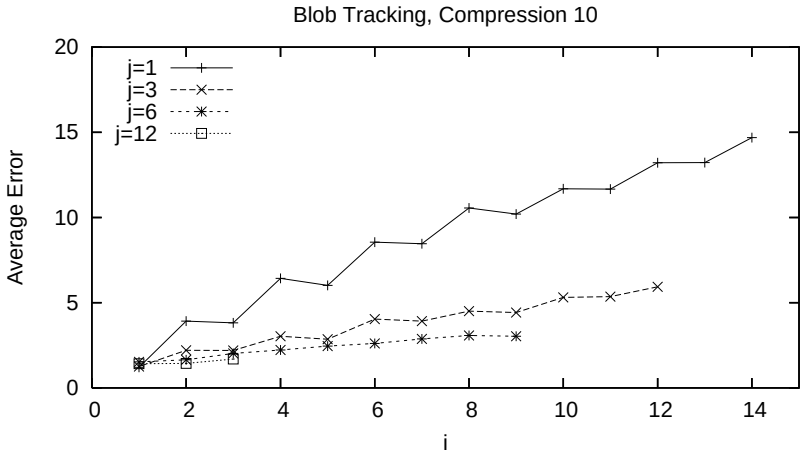


(a) Original VISOR video.

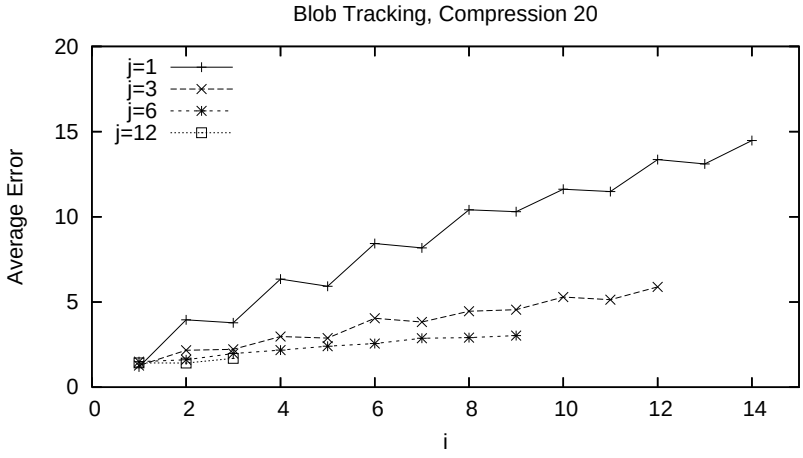


(b) Original PETS2001 video.

Figure 4.19: Accuracy of blob tracking algorithm for VISOR (snapshot in Figure 3.4(f)) and PETS2001 (snapshot in Figure 4.18(a)) videos.



(a)



(b)

Figure 4.20: Accuracy of blob tracking algorithm for PETS2001 video compressed with quality 10 and 20.

Chapter 5

Rate-Accuracy Framework

Our experiments on the selected video analysis algorithms demonstrate the dependency between accuracy of the selected video analysis algorithms and reduced video quality. We show that critical video quality exists for face detection, recognition, and tracking algorithms when SNR, spatial, and temporal video qualities are reduced. We considered the effect of SNR and spatial quality (and their combination) on face tracking and face detection; and the effect of SNR and temporal quality on face tracking. The blob tracking, due to the specifics of the frame differencing of foreground object detection, does not show a clear critical temporal quality but demonstrates a strong correlation with frame dropping pattern.

Running experiments with different combinations of video analysis algorithms and video adaptations is not enough to understand the relationship between an algorithm and a video adaptation. Video analysis algorithms, video data, and video adaptations (e.g., compression) are often complex in their nature. Also, given large growing number of video analysis algorithms, which are constantly evolving, there are many different combinations of algorithms and adaptations. Testing how every adaptation affects accuracy of every algorithm is not feasible in a practical system. Therefore, we need to do have some way to formalize and analyze the tradeoff between video bitrate (result of applying a video adaptation) and accuracy of a video analysis algorithm (a response to reduction in video quality).

In this chapter, we generalize our experimental findings by introducing the rate-accuracy function for a given video analysis algorithm. We first estimate the complexity of the experi-

ments needed for finding critical video quality for a given video analysis algorithm and several available video adaptations. Then, we suggest an approach to reduce a number of experiments by analyzing the features that an algorithm is searching for in video, features of the video itself, and the way adaptations affect these features.

By advancing in our reasoning about video features, we diverge from the idea of simply reducing the number of experiments for finding critical video quality. We show that video features metrics (measures how much of certain video feature is present in the video) can be used to evaluate quite accurately the performance of video analysis algorithms and are good in estimation of critical video quality across the corresponding video adaptations. Thus, video features metrics can be considered as an instrument for measuring the quality of computer vision. Such reasoning can also lead to developing improved versions of video analysis algorithms that are more tolerant to lower video bitrate.

5.1 Rate-Accuracy Function

The rate-accuracy function of a video analysis algorithm, under an environmental condition, gives the minimum rate of a video stream that satisfies the required accuracy of the algorithm. Deriving this function is non-trivial, as the rate of the stream depends on three different quality dimensions: SNR, temporal and spatial, each having different effects on different video analysis algorithms. For instance, the accuracy of a face detection algorithm mainly depends on the SNR quality of video while the accuracy of an object tracker is affected by the temporal quality as well.

Let the accuracy a of a computer video analysis algorithm $\mathcal{A}$ under an environmental condition e be a function g of the quality of the video, which consists of temporal quality τ, compression quality γ, and spatial quality σ. The environmental conditions can be some discrete values that would categorize video into a certain way. For example, a set of such conditions can include "good lighting", "snow falling", "outdoor", "road traffic", etc. Then, we can express the accuracy function as:

$$a = g_{\mathcal{A},e}(\tau, \gamma, \sigma) \tag{5.1}$$

We can obtain the function g through off-line experiments with the algorithm that are similar to presented in Chapter 4.

The rate of the streaming video, R, depends directly on the video quality based on some function r, i.e.:

$$R = r(\tau, \gamma, \sigma) \tag{5.2}$$

There are two different problems that may be of interest in the context algorithm's practical application. The first is to minimize the transmitting video bit rate for the given accuracy a' of an algorithm $\mathcal{A}$. Let $S(a')$ be the set of solution triples $(\tau', \gamma', \sigma')$ that satisfies the equation:

$$a' = g_{\mathcal{A},e}(\tau, \gamma, \sigma) \tag{5.3}$$

Each triple, or combination of encoding parameters, leads to different possible video bit rates, among which we should find one that minimizes the bit rate. Denoting $R_{\mathcal{A}}(a')$ as the minimal bit rate of a video satisfying the accuracy a' of algorithm $\mathcal{A}$, we can express:

$$R_{\mathcal{A}}(a') = \min_{(\tau',\gamma',\sigma') \in S(a')} (r(\tau', \gamma', \sigma')) \tag{5.4}$$

As an example, let algorithm $\mathcal{A}$ be the CAMSHIFT face tracker, and accuracy a (measured as average face distance ratio) be 0.3. Based on the experimental findings presented in Figure 4.14(a) and Figure 4.14(b), the accuracy, subject to environmental conditions, can be achieved with videos encoded using the parameters shown in Table 5.1. For example, the first row of Table 5.1 shows that we can achieve the accuracy of 0.3 with SNR quality of 100, and frame rates between 3.3 fps and 30 fps. This set of encoding parameters yield a resulting bit rate ranging from 0.2 Mbps to 2 Mbps. Note that videos of compression quality below 50 cannot satisfy the required accuracy of 0.3. Hence, we can find the minimum bit rate according to Equation 5.4, which is of 12 kbps and achieved using video of compression quality 50 and temporal quality 3.75 fps.

Besides minimizing the bit rate, the set of equations above also gives us a framework for dynamic video rate adaptation, with the goal of maximizing the accuracy of a given algorithm. Such formulation is similar to the utility-based adaptation framework presented by Kim *et al.* (Kim et al., 2003). In their framework, the rate is constrained, and the goal is to find

Compression Quality	Min FPS	Max FPS	Min Bit Rate	Max Bit Rate
100	3.3	30	0.2 Mbps	2 Mbps
75	3.3	30	70 kbps	0.6 Mbps
50	3.75	30	12 kbps	100 kbps

Table 5.1: Profiles of Video Matching Required for Face Tracking Accuracy of 0.3.

the maximum quality of the video based on human perceptions. In our context, instead of maximizing the video quality, we need to maximize accuracy of the algorithm. The problem becomes the following: find a combination of encoding parameters to maximize the accuracy of the algorithm when the available bandwidth is less than B:

$$a_{max} = \max_{\tau,\gamma,\sigma}(g_{\mathcal{A},e}(\tau,\gamma,\sigma)) \tag{5.5}$$

$$\text{subject to } r(\tau,\gamma,\sigma) \leq B \tag{5.6}$$

In the case where several video analysis algorithms need to be performed on the same video source, the resulting video should have the quality to satisfy the most quality-sensitive algorithm among them. The maximum of temporal, spatial and SNR qualities among all values that meet the accuracy of all these algorithms should be used. If there is a constraint on network bandwidth, priority can be assigned to each operation. Taking the priorities into account, the resulting video rate can be adjusted by solving the max-min problem with varying SNR, temporal and spatial qualities.

5.2 Estimation of the Rate-Accuracy Function

We have demonstrated that our tested video analysis algorithms can sustain high reductions in video quality, and we introduced rate-accuracy function, a formal representation of the phenomena. An important question remains however, of how to determine the critical video quality in practice. One approach is to perform an exhaustive search by running a video analysis algorithm on degraded video with every video adaptation used in the system. Such search is inefficient and therefore undesirable. We demonstrated, however, that by using reasoning, critical video quality can be approximated. Sometimes even a simple intuition may suffice. For example, we

can reason that in typical office environment, 0.5 seconds is enough to capture face of a person passing in front of a video camera. Hence, without resorting to experiments, we can estimate that 2 fps should be the minimal frame rate of the video that is intended for face detection algorithms. But such simple reasoning would not work in all cases. Compression algorithms (JPEG, MPEG4, etc.), for instance, are very complex and, therefore, it is hard to estimate their effect on non-trivial face detection or recognition algorithms. To obtain a fairly accuracy estimation of critical video quality, we need to take a closer look at video adaptations, video analysis algorithms, and the video itself.

But first, we want to understand the complexity of running all necessary experiments for determining critical video quality for a given video analysis algorithm and several video adaptations. Let's assume that there are n video adaptations that modify a video, changing its quality and bit rate. By running algorithm $\mathcal{A}$ on each possible modification of the video, we obtain a corresponding accuracy value. All pairs of the video bit rate and the algorithm's accuracy form the rate-accuracy function. The sweet spot in the curve of the function determines the value of the critical video quality. The number of experiments, necessary for finding the critical video quality, we call *experimental complexity*. Using such quantitative metric allows us to measure the benefits of estimating critical quality instead of using the blind experimental search.

5.2.1 Straightforward Approach

First, we assume the simplest situation when we have no specific knowledge about algorithm $\mathcal{A}$ and consider it as a black box. We can only determine the rate-accuracy function through experiments. Therefore, for each variation of adaptations, we obtain the modified video, note the video bit rate, execute the algorithm $\mathcal{A}$, and record its accuracy.

Notations and Assumptions Let D_i denote a video adaptation, where $i = 1...n$. We assume that each video adaptation D_i has a parameter, which we call *modification degree*, measuring how much the adaptation modifies the video. If the modification degree of the adaptation is j_i, we write the adaptation in form of $D_i^{j_i}$. The word quality is often used in the sense of the modification degree creating ambiguity with the video quality describing the level of

video distortion. For example, JPEG compression quality is the modification degree for JPEG adaptation, which should not to be confused with compression video quality.

Although in reality each adaptation can have different number of modification degrees, for simplicity of computing the experimental complexity, we assume that each adaptation D_i has m modification degrees.

We define $\mathcal{D}(J)$ to be a set of video adaptations, where each D_i is stated in its modification degree j_i. Hence, $\mathcal{D}(J)$ describes the state of all D_i and can be written as following:

$$\mathcal{D}(J) = \{D_1^{j_1}, D_2^{j_2}, ..., D_n^{j_n}\},\ where\ J = (j_1, j_2, ..., j_n) \tag{5.7}$$

Vector J represents one of the variations obtained by changing j_i from 0 to m for each adaptation D_i. We denote the total number of variations as $T(n, m)$.

We denote the video in its original unmodified condition as v_0. The expression $D_i^{j_i} \cdot v_0 = v_{j_i}$ means that applying adaptation D_i with degree j_i to the original video v_0 results in modified video v_{j_i}. Therefore,

$$\mathcal{D}(J) \cdot v_0 = D_1^{j_1} \cdot D_2^{j_2} \cdot ... \cdot D_n^{j_n} \cdot v_0 \equiv v_J \tag{5.8}$$

Thus, video v_J is the video that is obtained from original video v_0 by applying video adaptations with modification degrees defined by J.

We assume that the order in which adaptations are applied to the video is not important, i.e., adaptations have a commutative property: $\forall i, k, D_i \cdot D_k \cdot v_0 = v_{j_i k} = D_k \cdot D_i \cdot v_0$. The assumption is valid in many cases; consider: for MJPEG format, dropping frames followed by compression results in the same video as compression followed by dropping frames (since MJPEG is just a series of independent frames compressed with JPEG), cropping (in percentage to the spatial size) followed by spatial scaling gives the same result as scaling followed by cropping, etc. While adaptations of similar type, i.e., two different lossy compressions JPEG and JPEG2000, may not be commutative, it is rare that they would be used together in the same set $\mathcal{D}(J)$.

We also define r_J to be the bit rate of the video v_J and a_J to be the accuracy of the algorithm $\mathcal{A}$, obtained by executing the algorithm with the input video v_J.

Complexity Estimation $T(n, m)$ denotes the number of all possible states $\mathcal{D}(J)$ of n adaptations, each changing the modification degree from 0 to m. Computing $T(n, m)$ is similar to computing the number of variations with repetitions in combinatorics. It is analogous to placing $m + 1$ objects ($j_i = 0...m$) into n slots, since vector $J = (j_1, j_2, ..., j_n)$. The total number of variations with repetitions equals to $(m + 1)^n$. Therefore,

$$T(n, m) = (m + 1)^n \tag{5.9}$$

In practice, m can be large, for instance, JPEG compression quality varies from 1 to 100. Number of adaptations n is, usually, at least 2. For example, the commonly used adaptations are: frame-dropping, compression, cropping, and spatial scaling. Since we treat the video analysis algorithm $\mathcal{A}$ as a black box, it is necessary to conduct $(m+1)^n$ experiments. For each state $\mathcal{D}(J)$, we obtain video v_J, compute bit rate r_J of the resulted video, and run algorithm $\mathcal{A}$ on v_J to find accuracy a_J. Functions r_J and a_J form the rate-accuracy function that can be defined as

$$a_J = g_{\mathcal{A}}(r_J) \tag{5.10}$$

Both a_J and r_J are functions depending on video, which is obtained by applying adaptations determined by $\mathcal{D}(J)$. The sweet spot of the rate-accuracy curve, i.e., the critical video quality, is the value of J that gives the minimal video bit rate for a given algorithm's accuracy.

Figure 4.2(a) shows the example of accuracy a_J for the face detection algorithm vs. JPEG compression quality, when vector J consisting of just one element. As the vertical dashed line in the figure indicates, we can consider the compression quality 9 to be the critical quality for the Viola-Jones face detection.

5.2.2 Video Features

Running blind-search experiments to find critical video quality is an exponential function of the number of video adaptations as we have derived in equation 5.9. Such complexity makes it infeasible to use several different video adaptations in practical systems. We, therefore, propose reducing the number of experiments by considering *video features* that are certain characteristics

of the video, changing which (with video adaptations) would affect accuracy of video analysis algorithm $\mathcal{A}$. The reasoning is that since treating an algorithm, video, and adaptations as black boxes gives large experimental complexity, analysis of additional information would help in reducing the number of necessary experiments.

Different algorithms are sensitive to different video features. For example, video features of an object tracking algorithm can include speed and size of the tracked object, color of the background, histogram, etc. If the speed of the tracked object increases significantly, the tracker can lose the object. Also, if the color of the background becomes similar to the color of the tracked object, the tracker will lose the object, confusing it with the background.

But before going into details, let's compute the new experimental complexity by considering the set of video features just as a "bridge" connecting n video adaptations and algorithm $\mathcal{A}$. In Section 5.2.3, we continue to reason about video features and how to use them.

Notations and Assumptions Assume that we have identified the set of p video features that are critical to the accuracy of the video analysis algorithm $\mathcal{A}$. Identifying such features is a difficult task and requires a deep knowledge of $\mathcal{A}$. We discuss different categories and examples of video features in Section 5.2.4.

We denoted the set of features for algorithm $\mathcal{A}$ as

$$\mathcal{F}_{\mathcal{A}} = \{f_1, f_2, ..., f_p\} \tag{5.11}$$

When video adaptations modify the video, they also affect video features in the set $\mathcal{F}_{\mathcal{A}}$. Note, that a video adaptation affects only some of the video features. For example, scaling or compression adaptations do not change such video feature as the speed of the tracked object. Therefore, for each adaptation D_i, there exists a corresponding set $\mathcal{F}_{D_i} \subseteq \mathcal{F}_{\mathcal{A}}$. The modification degree of D_i affects video features from $\mathcal{F}_{D_i}$ only.

Figure 5.1 shows the graphical representation of relationships between video adaptations, video features, and video algorithm. The figure draws attention to the fact that video adaptations modify video features $\mathcal{F}_{D_i}$ in the video, which in turn, affect the accuracy of the algorithm $\mathcal{A}$. The bit rate of the modified video and the corresponding accuracy of the algorithm form

the rate-accuracy function as discussed in Section 5.2.1.

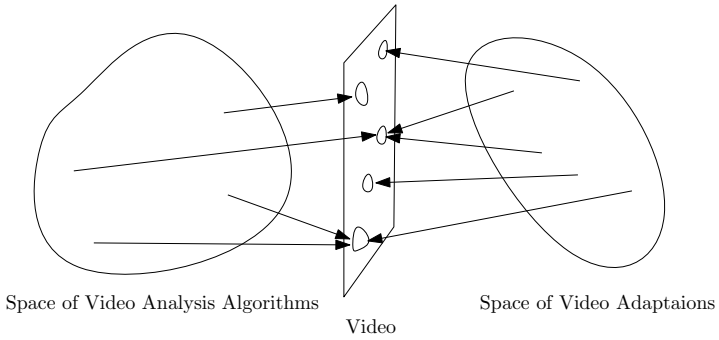


Figure 5.1: The relationship between video analysis algorithms and video adaptations.

Assume that we can measure the current condition or *state* of the video feature. Video adaptations, by modifying the video, change the states of the video features from $\mathcal{F}_{\mathcal{A}}$. For instance, the feature such as speed (per frame) of the object of interest is affected by the frame-dropping adaptation. More details on how to analyze and measure video features can be found in Section 5.2.4.

Each change in the modification degree of D_i corresponds to different *states* that features from $\mathcal{F}_{D_i}$ have. We denote the state of the feature f_k as $f_k^{s_k}$, where s_k is called the *state degree* and is the value measuring the state of feature f_k. Thus, f_k^0 indicates the original state of a video feature, before any adaptation is applied. Since we want to estimate the experimental complexity, we can assume, for the worst case, that $\mathcal{F}_{D_i} \equiv \mathcal{F}_{\mathcal{A}}$.

We can describe the effect of applying a video adaptation as follows. Video adaptation D_i changes its modification degree from j_i to $j_i + 1$, i.e., $D_i^{j_i}$ becomes $D_i^{j_i+1}$, modifying video from v_{j_i} to v_{j_i+1}, affecting some feature $f_k \in \mathcal{F}_{D_i}$ to change its state degree from s_k to s'_k, what results in change of the accuracy from a to a'.

Complexity Estimation In this section, we extend the complexity estimation given in Section 5.2.1 by introducing p video features, in addition to n video adaptations and one video analysis algorithm $\mathcal{A}$. We consider video features, without semantically analyzing them, as an extra set of entities serving as a "bridge" between $\mathcal{A}$ and adaptations. We estimate the experimental complexity by analyzing, first, how video adaptations change video features, and,

second, how video features affect the accuracy of algorithm $\mathcal{A}$. In Section 5.2.5, we aim to reduce the resulted experimental complexity through analysis of the video features.

There are n video adaptations changing the states of p video features. As we found earlier, the total number of states of adaptations $\mathcal{D}(J)$ is $(m+1)^n$. Therefore, the experimental complexity of finding all possible state degrees of p video features equals to

$$O(p \times (m+1)^n) \tag{5.12}$$

Video features, in turn, affect the accuracy of the video analysis algorithm $\mathcal{A}$. We have two assumptions. First, we assume that each value s_k, the state degree of the feature f_k, has a lower, s_k^l and upper, s_k^u, limits. For example, the speed of object of interest is bounded between 0 and the original speed. Theoretically, some features may have no fixed limit on the state degree, for instance, the size of object can be spatially up-scaled to any number. But in practice, we can always choose some fixed value for such limit. Second, we assume that each video feature is changing in discrete steps and we can identify the minimal step of change, for example, for spatial size it would be a pixel, and for speed it would be a pixel per frame. We denote the minimal step of change for the feature f_k as Δs_k. Every time $f_k^{s_k}$ is changed by a video adaptation D_i to $f_k^{s'_k}$ we can write that $s'_k - s_k = x \times \Delta s_k$, where x is some integer constant. Since a feature has lower and upper limits on its state degree, we can write for every feature f_k,

$$s_k^u - s_k^l = m_f \times \Delta s_k \tag{5.13}$$

where m_f is the maximum times the feature f_k can be changed if every time the step of change is Δs_k. For the simplicity of the experimental complexity estimation, we assume that m_f is the same for every video feature. Since there are total p video features that determine the accuracy of the algorithm $\mathcal{A}$, the experimental complexity to find accuracy of $\mathcal{A}$ via video features can be estimated as

$$O(m_f + 1)^p \tag{5.14}$$

Combing two complexity estimations above, the total number of experiments for obtaining

the rate-accuracy function by using video features is estimated as

$$O(p \times (m+1)^n + (m_f+1)^p) \tag{5.15}$$

This estimation is the worst case estimation and is not surprising that it appears to be large, since we introduced an additional set of video features between the algorithm and video adaptations. However, by considering the semantics of video features, in many cases, we can significantly reduce the experimental complexity. In the best case, when all experiments are substituted by reasoning, the experimental complexity in Equation 5.15 becomes just a computational complexity. In the next section, we study and analyze video features aiming to maximize the use of analytical expressions instead of experiments.

5.2.3 Analysis of Video Features

As we mentioned in Section 5.2.2, we consider video features as video characteristics that are modified by a given video adaptation and affect the performance of video analysis algorithm $\mathcal{A}$. Our hypothesis is that we can apply measure or analytically describe most of the video features affecting algorithm $\mathcal{A}$, which allows us to run less experiments than is estimated by Equation (5.15).

5.2.4 Identifying and Measuring Video Features

We categorized all video features that the accuracy of a general algorithm depends on into three distinctive groups:

- Video features related to algorithm's object of interest. Object of interest can be, for example, silhouette, face, car, etc., i.e., an object that the algorithm searches for in a video. We denote such set of features as $\mathcal{F}_O$.

- Video features resulted from using video adaptations to degrade the video. These features are those that were not part of the original quality video but introduced to it due low bitrate encoding and compression. Such features include blockiness, blurriness and other common visual artifacts. We denote such set of features as $\mathcal{F}_V$.

- Video features related to algorithm's internal structures; often also called features but to avoid ambiguity we call them internal structures. Such structures can be Haar-like features, moving foreground blobs, histograms, etc., i.e., what algorithm uses to search for the object in a video. These features also include the probabilistic features, which are defined by the training stage of the learning type of algorithms. We denote such set of features as $\mathcal{F}_I$.

Therefore, video features used by the algorithm $\mathcal{A}$ can fall into three groups:

$$\mathcal{F}_{\mathcal{A}} = \mathcal{F}_O \cup \mathcal{F}_V \cup \mathcal{F}_I, \tag{5.16}$$

We mainly focus on the first two categories of features as essential for finding the critical video quality. As for the internal structures of the algorithm, $\mathcal{F}_I$, we try to avoid using these features in our reasoning; but only consider them in relation to $\mathcal{F}_O$ and $\mathcal{F}_V$ features. The reason is that $\mathcal{F}_I$ features over-complicate the analysis: (i) they are very different for each individual video analysis algorithm; (ii) algorithms are often updated and modified to include various new features to improve their accuracy; (iii) many new algorithms are developed every day. Therefore, it is not feasible to analyze the internal features of the algorithms and develop any practically working approach for the estimation of critical video quality. Also, many video analysis algorithms are the learning algorithms, i.e., they are based on neural networks, AdaBoost, etc. Therefore, the training stage of such algorithms is crucial in shaping their performance accuracy. The method of training and the used training data (video) justify the probability with which an algorithm tracks, detects, or recognizes objects or events. Once the algorithm is trained however, the probabilistic decision tree is fixed, so applying video adaptations to degrade video quality would not affect the decision making process of the algorithm. Therefore, such probabilistic features of the algorithm do not affect how relative accuracy of the algorithm, which is our focus, changes when video adaptations are applied.

We can identify and measure video features in the set $\mathcal{F}_O$ analytically. The features of this type include such properties of object of interest as height, width, shape, speed, etc. These features do not depend on the video analysis algorithm itself, and therefore, we can analytically

compute the effect on them caused by a video adaptation. For example, we can compute the relative size of the object with respect to the spatial scaling adaptation or relative change in object's speed with respect to the number of the dropped frames.

Identifying the second type of features, $\mathcal{F}_V$, is less straightforward and requires a specific knowledge about video, video adaptations, and even a little about video analysis algorithms themselves. For example, considering Haar-like features used by Viola-Jones face detection as an internal structure, we can make a general deduction that sharpness of edges and intensity contrast near the edges are important for face detection to be accurate. Therefore, it is reasonable to assume that video adaptations affecting such properties of the image, would have a critical effect on face detection. Such reasoning lead us to focusing on *visual artifacts*, the most prominent artifacts that video adaptations, such as JPEG compression, introduce to images/video at low bitrates. Visual artifacts can be viewed as the structural damage of a video adaptation to the original image/video quality. The examples of visual artifacts include:

- Blockiness. This artifact gives a visible blocking effect to the image or video. It can be the result of JPEG compression (see Figure 3.1), nearest neighbor scaling (see Figure 4.11 and Figure 6.4(a)), pixel area relation scaling (Figure 6.4(b)), video encoders (see for example Figure 4.15), and others.

- Blurriness. This artifact makes edges in the image wider and less sharp and can be the result of some lossy compression algorithms like JPEG2000, bicubic scaling (see Figure 6.5(a)), bilinear scaling (see Figure 6.5(b)), and others.

- Color bleeding or loss of colorfulness. It is caused by the suppression of high-frequency coefficients of the chroma component of the image. The artifact can be a result of compression adaptation as well as other encoders. We do not consider the effect of this artifact on critical video quality because it does not appear to be prominent for the used JPEG compression and scaling algorithms. However, for more advanced algorithms (such as JPEG200 or wavelet-based compression), this artifact (and others such as ringing) can play a more important role and should be considered.

One of the main reasons why visual artifacts are great video features to use when estimating critical video quality for video analysis algorithms is that there are many metrics measuring how much of the artifact is present in the image/video. Therefore, the values of visual artifact metrics can be treated as the state degrees (described in Section 5.2.2) of the $\mathcal{F}_V$ features. In this study, we focus on two prominent visual artifacts, blockiness and blurriness and use corresponding non-reference metrics. More details on the analysis and experiments demonstrating the advantages of using artifact metrics can be found in Chapter 6.

Computing Critical Quality Assuming that the prominent video features of the algorithm $\mathcal{A}$ are identified as described in Section 5.2.4, and the relevant metrics are chosen, in this section, we describe how to compute the critical video quality value using video features.

Analogous to a human observer, who sets requirements on a perceived video quality, a video analysis algorithm imposes specific constraints on the video features. Typically, a human prefers frame rate higher than 24 fps, spatial resolution to be maximized, video frames neither to be blur nor rigid, etc.; likewise, the algorithm has its preferences to certain video features. Each algorithm has a different set of thresholds on how degraded a video feature can be; and based on our various practical experiments in Chapter 4, we assume that algorithms are more tolerant to low quality video than a human observer. The deterministic nature of video analysis algorithms and their relative simplicity compare to the human visual system encourages us to use reasoning approach for identifying thresholds for state degrees of the video features (see Section 5.2.2 for the definition).

We assume that algorithm $\mathcal{A}$ has lower, denoted as $\tilde{s}^l_{\mathcal{A},k}$, and upper, denoted as $\tilde{s}^u_{\mathcal{A},k}$, thresholds on state degree of each feature f_k from $\mathcal{F}_\mathcal{A}$. Recalling the lower and upper limits on state degrees of the video features defined in Section 5.2.2, the following holds for each feature f_k:

$$s^l_k \leq \tilde{s}^l_{\mathcal{A},k} < \tilde{s}^u_{\mathcal{A},k} \leq s^u_k \tag{5.17}$$

There are several ways to identify such thresholds. Some are set in the code of the algorithm, for example, face detection algorithms set a limit on the minimal detectable size, usually, 20×20 pixels. Some can be determined from the indirect factors such as maximal traceable speed of

the object in the video. Others can be only obtained through experiments, for example, the maximal thickness of the edge for a face detection algorithm.

A threshold of the algorithm can be termed as the critical state degree of the feature f_k if decreasing or increasing the state s_k below or above the threshold degrades the original accuracy a_0 of the algorithm $\mathcal{A}$. Once we find the critical state degrees for all video features from $\mathcal{F}_\mathcal{A}$, we can derive the critical video quality. We base this claim on the assumption that the accuracy of $\mathcal{A}$ is determined by the critical state degrees. We demonstrate this approach and show that the claim is valid in Chapter 6 using examples of blockiness and blurriness visual artifacts.

Consider the example when adaptation D is JPEG compression, and Viola-Jones face detection is the algorithm $\mathcal{A}$. Let video features or the artifacts of the adaptation D be $\mathcal{F}_\mathcal{A} = \{f_u = blurriness,\ f_o = blockiness,\ f_e = jerkiness,\ f_c = color\ blindness\}$. Changing JPEG compression quality from 1 to 100, since it is modification degree of D, leads the state $\mathcal{D}(J)$ to be $\{D^1\}, \{D^2\}, ..., \{D^{100}\}$, with vector J changing from (1) to (100) respectively. To each D^j, there are corresponding values of metric functions, i.e., state degrees $\{s_u, s_o, s_e, s_c\}$, for each features from $\mathcal{F}_\mathcal{A}$. Our hypothesis suggests that we can identify critical state degrees for each feature, i.e., $\{\tilde{s_u}, \tilde{s_o}, \tilde{s_e}, \tilde{s_c}\}$, such that they determine a given accuracy of the algorithm $\mathcal{A}$. As follows from the experimental results presented on Figure 4.2(a), when JPEG compression quantizer is 9, one of the feature suppose to match its critical state degree, brining the accuracy of $\mathcal{A}$ to the sweet spot. This example of this situation is studied in Section 6.1 with only blockiness as the video feature in the set $\mathcal{F}_\mathcal{A}$; it is shown that the same value of blockiness corresponds to sweet spots in different video adaptations.

5.2.5 Reducing Experimental Complexity Using Video Features

The experimental complexity, stated in Equation 5.15, consists of two main parts. Relationships between video adaptation and video features contribute to the first part and relationships between video features and the algorithm $\mathcal{A}$ to the second. In this section, we, first, present the computational algorithm that combines changing video adaptations, corresponding video features, and the accuracy of $\mathcal{A}$. The experimental complexity given in Equation 5.15 matches

the computational complexity of this algorithm, when each step of the algorithm is an experiment. Second, we consider two groups of video features, denoted in Section 5.2.3 by $\mathcal{F}_O$ and $\mathcal{F}_V$, and, using analysis presented in Section 5.2.4, replace in computational algorithm as many experiments as possible with analytical formula.

Computational Algorithm To find the critical video quality, we need to increase modification degrees of video adaptations until video features match the corresponding critical state degrees. The problem is how to increase the modification degrees of n adaptations that the video gradually changes from original to the video that is most degraded by adaptations. We propose an algorithm that imposes a partial order on the way each video adaptation D_i changes its modification degree. Recall that $\mathcal{D}(J)$ is the set of adaptations with modifications degrees, where J is the state vector. If we start from $J = (0, 0, ..., 0)$, by increasing modification degree for each D_i, we should eventually reach $J = (m, m, ..., m)$. Let's choose the current adaptation $\tilde{D}$ to be consequently D_1, D_2, D_3, and so on. To go through all variations of J, at one step, we either increase or decrease by 1 the modification degree of $\tilde{D}$ and perform the recursive call making the previous adaptation as $\tilde{D}$. In such manner, we visit all states from $(0, 0, ..., 0)$ to $(m, m, ..., m)$, when $\tilde{D}$ becomes D_n in m's modification degree. Therefore, any time the state vector $J = (j_1, ..., j_i, ..., j_n)$ changes to the next vector denoted as J', the following rule applies,

$$
\begin{aligned}
&J = (j_1, ..., j_i, ..., j_n) \mapsto J' \Rightarrow \\
&\exists i : j_i \mapsto j_i \pm 1 \Rightarrow \\
&\mathcal{D}(J) = \{D_1^{j_1}, ..., D_i^{j_i}, ..., D_n^{j_n}\} \mapsto \mathcal{D}(J') = \left[\begin{array}{l} \{D_1^{j_1}, ..., D_i^{j_i+1}, ..., D_n^{j_n}\} \\ \{D_1^{j_1}, ..., D_i^{j_i-1}, ..., D_n^{j_n}\} \end{array} \right.
\end{aligned} \tag{5.18}
$$

The algorithm ensures that every step only one video adaptation increases or decreases its modification degree by 1, thus, advancing p video features in $\mathcal{F}_\mathcal{A}$ to the previous state degrees or the next state degrees. For every change in state degrees of features in $\mathcal{F}_\mathcal{A}$ there is a corresponding accuracy of the algorithm $\mathcal{A}$. The computational algorithm finding accuracy of $\mathcal{A}$, for every step out of $(m + 1)^n$ steps, has the same computational complexity as stated

in Equation 5.15. In case we need experiments to find state degrees of all features and the accuracy of $\mathcal{A}$, the complexity, then, becomes experimental.

Substitute Experiments with Analysis In Section 5.2.5, we proposed the computational algorithm that can be used to find the critical video quality for a given accuracy of the video analysis algorithm $\mathcal{A}$. Through analysis of video features, we aim to use analytical formulas instead of experiments in the computational algorithm. We consider each of the two groups of features, $\mathcal{F}_O$ and $\mathcal{F}_V$, individually.

Video features in $\mathcal{F}_O$ are related to the video object of interest, and therefore, we can analytically measure their state degrees. For example, we can measure speed and size of the object of interest. The thresholds on such features imposed by the algorithm $\mathcal{A}$ are either stated in its code, i.e., minimal detectable size for face detection, or can be derived indirectly, i.e., minimal traceable speed of the object for object tracking. Therefore, for every modification degree of the corresponding adaptations, the state degrees of features in $\mathcal{F}_O$ can be computed analytically, as well as the accuracy of the algorithm $\mathcal{A}$ can be analytically derived from their state degrees.

Thus, defining $p_o = \|\mathcal{F}_O\|$, the experimental complexity can be reduced to

$$O((p - p_o) \times (m + 1)^n + (m_f + 1)^{(p-p_o)}) \tag{5.19}$$

Video features in $\mathcal{F}_V$ are related to the video adaptations degrading video quality for the algorithm $\mathcal{A}$. These video features typically are visual artifacts which we suggest to measure with available metrics. Such features include blockiness, blurriness, edge sharpness, contrast, brightness, etc. By measuring the effect of modification degrees for each video adaptation, we can further reduce the experimental complexity to

$$O((m_f + 1)^{p_i}), \tag{5.20}$$

where $p_i = p - p_o$.

The analytical estimation of the critical state degrees for the video features in $\mathcal{F}_V$ appears to be complicated. If such estimation is not possible, we can run experiments to find these critical

state degrees, which, recall, are thresholds of the algorithm $\mathcal{A}$. However, since for each feature we need to determine only one value from $m_f + 1$ values, we can use experimental search, for example, binary search. The use of the binary search reduces the experimental complexity to

$$O((\log(m_f + 1))^{p_i}) \tag{5.21}$$

Chapter 6

SNR Quality Estimation

In Chapter 5, we have proposed to use reasoning and rely on video features when approximating critical video quality for video analysis algorithms in practical systems. We also estimated the experimental complexity of finding such critical quality. In this chapter, we are going to show how the proposed video features and their metrics can be used in practice. We first focus on the features from the set $\mathcal{F}_V$, which are the result of video adaptation degrading video quality. Video features related to object of interest, $\mathcal{F}_O$, and their analysis is considered in Chapter 7.

As discussed, for effective estimation of critical SNR quality, we propose to use quality metrics specific to video adaptations and video analysis algorithms. We propose to measure *visual artifacts*, including blockiness, blurriness, color bleeding or loss of colorfulness, and ringing. For instance, blockiness is the most prominent artifact of JPEG compression. In the same time, Viola-Jones face detection relies on Haar-like features (see Figure 4.1), which means that its accuracy is significantly affected by blockiness. Therefore, we can find a value of blockiness for face detection that corresponds to critical JPEG quality and use this value to estimate a critical quality for other video adaptations with strong blockiness. In such practical scenarios, when there are fewer visual artifacts than the number of possible adaptations, measuring video quality with artifact metrics is more efficient than performing exhaustive experiments for every possible pair of video analysis algorithm and video adaptation.

While quality metrics exist for human visual system, for example, objective PSNR metric or perceptive VQM and SIMM, these metrics cannot be applied to video analysis because, unlike

humans, every algorithm has different requirements to the quality.

In this chapter, we consider estimation of critical SNR quality for Viola-Jones, Rowely face detection and QDA-based face recognition algorithms, using JPEG compression and several scaling algorithms as video adaptations. We first demonstrate that blockiness and bluriness metrics (blurriness is shown for face recognition only) can be accuracte in estimating critical SNR quality. Therefore, in terms of our rate-accuracy framework, algorithm $\mathcal{A}$ is consequently considered as face detection and recognition algorithms. Video adaptation D is either JPEG compression or a scaling algorithm. And we simplify the set video features to one video feature at a time, which we pick to be the most prominent video feature of a particular video adaptation. We pair blockiness with JPEG compression, nearest neighbor, and pixel area relation scaling; and blurriness with bicubic and bilinear scaling algorithms. We also propose to use more general metric, mutual information, for measuring the degree of video adaptation's degradation of the image quality. We show that mutual information can also be used as an adaptation independent, though less accurate compared to blockiness or bluriness, metric of SNR video quality.

6.1 Blockiness Metric

In this section, we demonstrate that blockiness can be used as a video quality metric for various blocky video adaptations. To avoid inconsistencies with definition of critical SNR quality, we call the corresponding value of blockiness metric as *threshold* on blockiness. We first find such threshold for a face detection or recognition algorithm and a single video adaptation with blockiness artifact, e.g., JPEG compression. To demonstrate that it can be used as metric, we show that the same threshold value can be used to determine critical SNR quality for other blocky video adaptation as well.

A non-reference blockiness metric by Muijs and Kirenko (Muijs & Kirenko, 2005) is adopted in our experiments. We chose this metric, because it is easy to implement and easy to adjust for blocks of different size. In a given blocky image, the metric measures the contrast between local gradient of the block's edge and the average gradient of the adjacent pixels. Essentially, the metric's value is the ratio of these gradients. It considers horizontal and vertical block edges

separately and takes the average of these values across all the blocks in the image.

We use images from MIT/CMY dataset for face detection algorithms (see Section 3.2 for more details) with JPEG compression as video adaptation. For recognition algorithm, we use Yale dataset and different scaling algorithms.

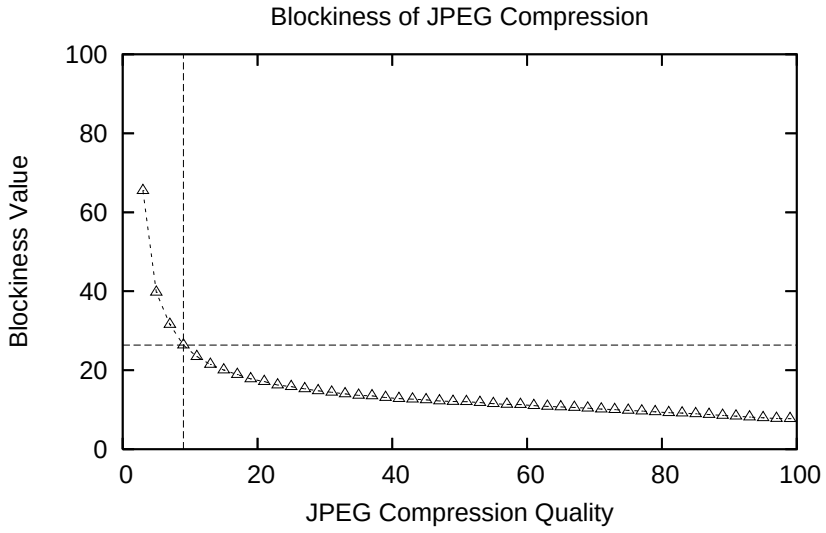
6.1.1 Face Detection

Since blockiness is the most prominent visual artifact of JPEG compression, it is reasonable to suggest that this artifact would affect accuracy of face detection. We compute blockiness for each compressed image assuming that block artifacts of JPEG have a size of 8×8 pixels. Since we later use blockiness for other video adaptations that have blocks of different sizes, we normalize its original value by multiplying it with the block's size. Using the MIT/CMU dataset, we measure the blockiness for different JPEG compression qualities and plot the results in Figure 6.1(a).

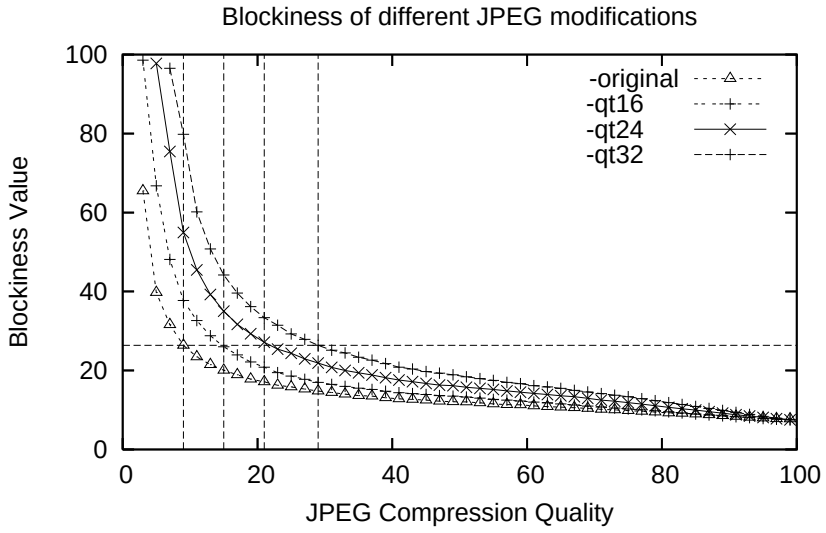
For Viola-Jones face detection algorithm, taking JPEG compression quality 9 (the sweet spot in Figure 4.2(a)), we can suggest 26.4 to be the threshold on blockiness (indicated by the dashed line in Figure 6.1(a)). In order to verify that blockiness is a suitable quality metric for Viola-Jones algorithm, we need other video adaptations with blockiness artifact, and, for these adaptations, the same threshold value should fit the corresponding sweet spots.

We artificially created other blocky video adaptations by modifying JPEG compression. We created three simple quantization tables of JPEG that lead to blockier images than the original JPEG compression. Tables are constructed without any specific reason in mind, except they should be simple and emulate the pattern of the original quantization table. We used formula $a_{ij} = (4 + i)(4 + j)$, to obtain seven rows of the first table with last row and column repeated twice. Multiplying values of this table by 3/2, we obtain the second table and multiplying them by 2, we obtain the third table. We term the corresponding JPEG compressions according to their tables' most top-left values: "qt16", "qt24", and "qt32". JPEG with the original quantization table is marked as "original".

Blockiness values for our JPEG modifications are compared in Figure 6.1(b). Taking 26.4



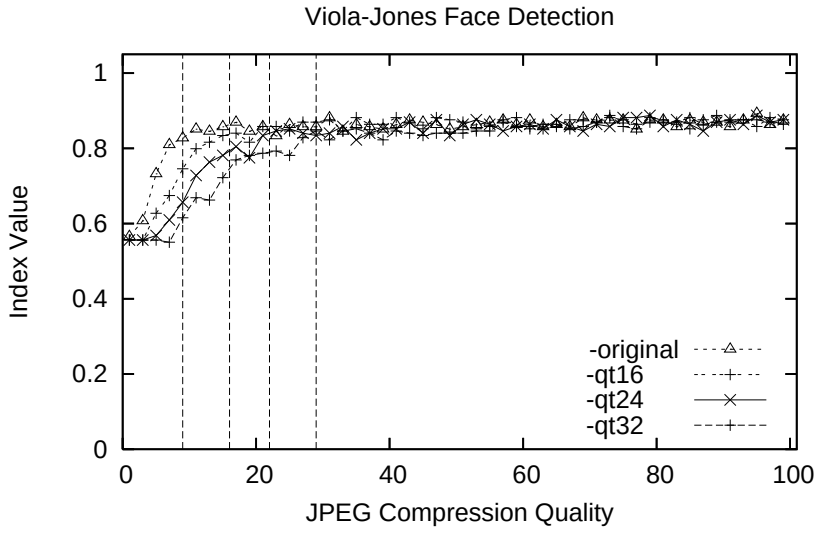
(a)



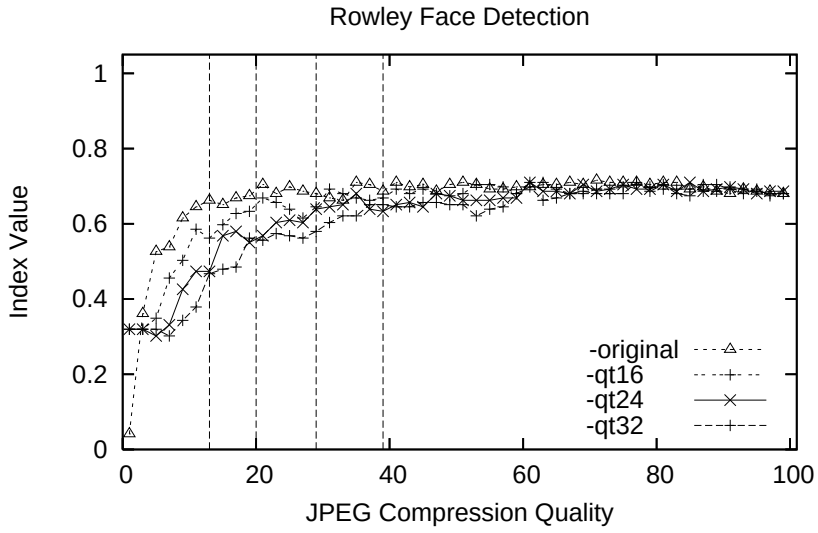
(b)

Figure 6.1: Value of blockiness metric vs. JPEG compression quality for different modifications of JPEG algorithm.

as threshold on blockiness determined above, we can estimate that the critical SNR quality for "qt16" should be 15, for "qt24" should be 21, and for "qt32" should be 29 (all values are indicated in the figure with dashed vertical lines). Plotting accuracy of Viola-Jones algorithm against compression qualities of these JPEG modifications in Figure 6.2(a) demonstrates that the estimated critical SNR qualities match the sweet spots of the corresponding curves very well. Therefore, the same threshold on blockiness determines the critical SNR quality value for Viola-Jones face detection algorithm and several different versions of JPEG compression.



(a)



(b)

Figure 6.2: Accuracy of Viola-Jones and Rowley face detection algorithms vs. JPEG compression quality for different modifications of JPEG algorithm.

To verify that blockiness as the quality metric is not specific to Viola-Jones algorithm only, we conducted the above experiments for Rowley algorithm. The threshold on blockiness is determined as 21.5 based on the sweet spot value 13 from Figure 4.2(b) and the blockiness measurements of JPEG in Figure 6.1(a). Therefore, critical SNR qualities for different modifications of JPEG can be estimated as 20 for "qt16", 29 "qt24", and 39 for "qt32" (from Figure 6.1(b)). Plotting accuracy of Rowley algorithm against our versions of JPEG compression in Figure 6.2(b) confirms the estimated values as they fit the corresponding sweet spots.

Rowley face detection algorithm is based on variations in pixel intensities, which are not blocky type of features as Haar-features of Viola-Jones algorithm. Nevertheless, blockiness metric estimates the critical SNR quality for Rowley algorithm well, because we use JPEG compression, for which blockiness is a main visual artifact. This observation indicates that the accuracy of face detection is mostly affected by the type of video adaptation's distortion rather than features that algorithm relies on in its detection.

(a)

	Critical quality	Image size (bytes)
original	9	6955
qt16	15	6861
qt24	21	6604
qt32	29	6739

(b)

	Critical quality	Image size (bytes)
original	13	8547
qt16	20	8171
qt24	29	7980
qt32	39	8035

Table 6.1: Critical video qualities and corresponding average images sizes estimated with blockiness metric for Viola-Jones (a) and Rowley (b) algorithms with original and modified JPEG compressions.

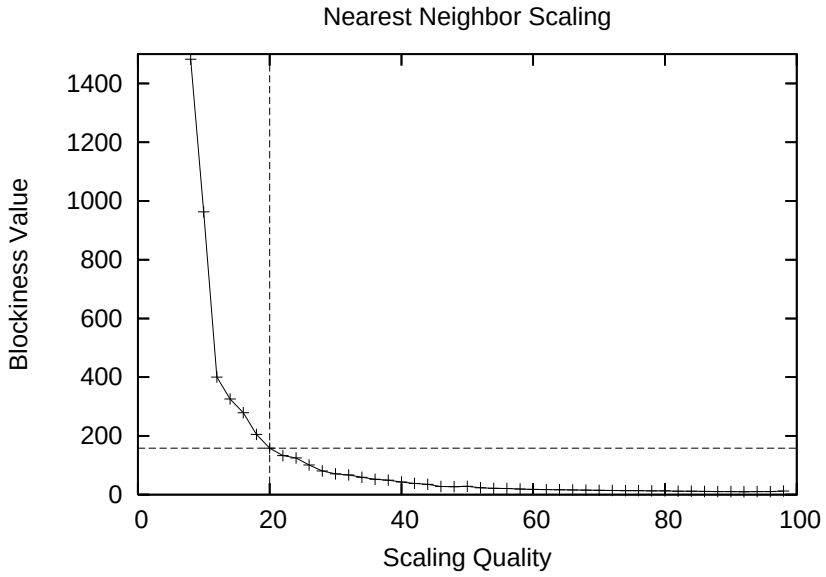
Note that the proposed simple modifications of JPEG are more preferable compared to original JPEG compression. First, the original quantization table is empirically determined to fit human visual system, which is not well suited for video analysis algorithms. Second, modified quantization tables can be expressed using formula and hence easier to use in practice compared to storing tables in memory of every device that uses JPEG compression (the current situation). The only concern with simpler modifications of JPEG would be that their critical SNR qualities amount to bigger file size compared to original JPEG. To address this concern, we measured the average size of tested images compressed the critical qualities for Viola-Jones algorithm in Table 6.1(a). From the table, we notice that each critical quality corresponds to images with average size 8% of the images compressed with conventional JPEG quality 90. Hence, our simplified versions of JPEG lead to similar or arguably better bitrate reductions than the original JPEG. Similarly, Table 6.1(b) shows that for Rowley algorithm, critical SNR qualities of original and modified JPEG compressions result in images with comparable average sizes. These findings suggest that simpler and more efficient encoders can be developed for these face detection algorithms.

6.1.2 Face Recognition

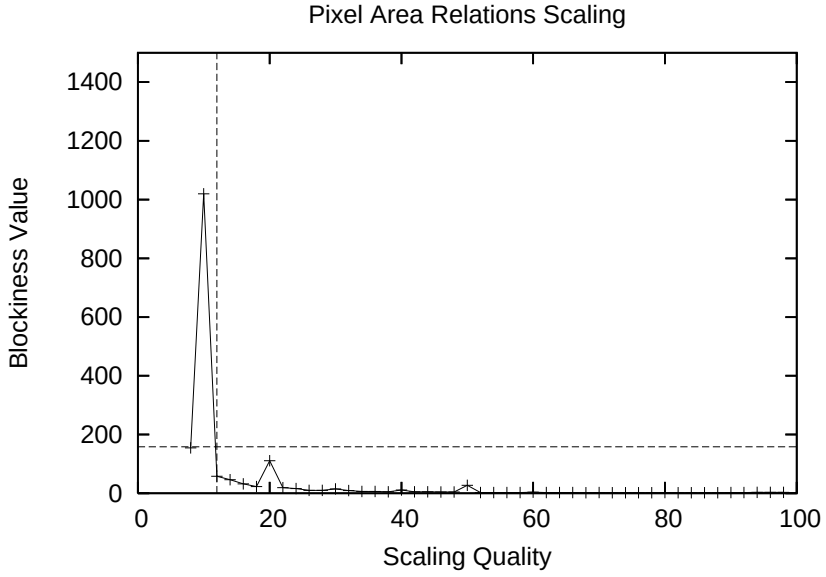
For QDA-based face recognition algorithm, we estimate the critical video quality using blockiness metric for nearest neighbor and pixel area relation scaling algorithms. These scaling algorithms exhibit strong blockiness visual artifacts, as shown in Figures 6.4(a) and Figures 6.4(b). Unlike JPEG compression, however, sizes of resulted blocks depends on the value of scaling quality (the percentage to which images are pre-scaled to). For example, consider downscaling an original image to 50% using nearest neighbor. After scaling back, each pixel in the resulted image is repeated, resulting in the blocks of 2×2 pixels. Therefore, we adopted the blockiness metric used in Section 6.1 to blocks of different size. The blockiness value for nearest neighbor and pixel area relation scaling algorithms are presented in Figure 6.3(a) and Figure 6.3(b) respectively. Combining these measurements with results on accuracy of the face recognition algorithm given in Figure 4.12(a) and Figure 4.12(b), we can find value 158.5 to be a threshold on blockiness. Note that the same threshold value is obtained for different scaling adaptations. This fact indicates that blockiness can be used as SNR quality metric for QDA face recognition as well.

6.1.3 Blurriness Metric

Similarly to blockiness, blurriness is another common artifact that video adaptations introduce to the images. We demonstrate that a metric measuring blurriness, naturally the blurriness metric, can also be used to determine a critical SNR quality for certain video adaptations. We use a non-reference blurriness metric proposed by Chung *et al.* (Chung et al., 2004), which measures the spread of the edges in a blurred image. As we can see in Figure 6.5(a) and Figure 6.5(b), bicubic and bilinear scaling algorithms introduce a strong blurriness artifact into an image, to which a down-scaling-up-scaling was applied. Blurriness metric computed for Yale dataset is presented in Figure 6.6(a) for Bicubic scaling and Figure 6.6(b) for bilinear scaling. Considering QDA-based recognition algorithm, we marked with vertical dashed lines the critical qualities of the algorithm for bicubic and bilinear adaptations respectively. For details on corresponding critical video qualities of the recognition algorithm, refer to Figure 4.12(d)



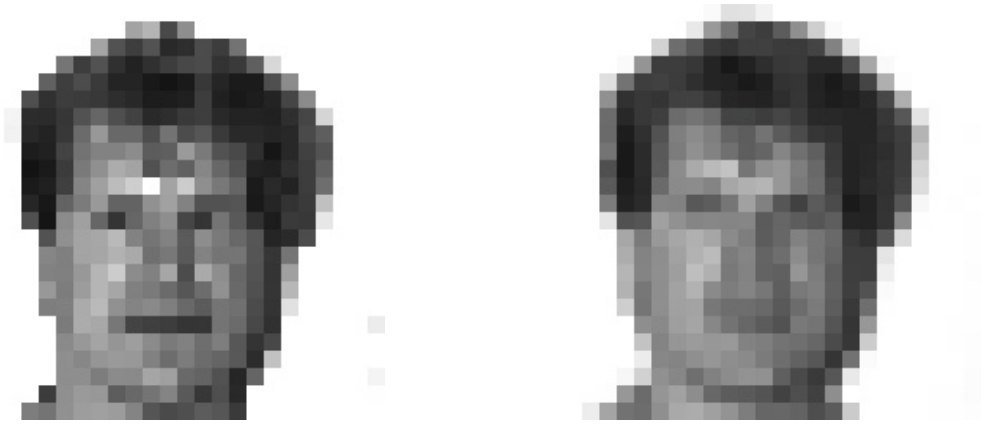
(a)



(b)

Figure 6.3: Blockiness metric vs. scaling quality for nearest neighbor 6.3(a) and pixel area relation 6.3(b) scaling algorithms.

and Figure 4.12(c) in Section 4.2. We can notice that these critical qualities correspond to the same value 42.6 of blurriness, as shown with a horizontal dashed line in Figures 6.6(a) and 6.6(b). Such strong correlation of blurriness and accuracy of recognition algorithm shows that blurriness can be a good metric for estimation of critical SNR quality for video adaptations that make images or video blurry.



(a) Nearest neighbor algorithm, scaling quality 10. (b) Pixel area relation algorithm, scaling quality 10.

Figure 6.4: Nearest neighbor and pixel area relation scaling algorithms demonstrate a strong blockiness artifact. An example image is from Yale dataset.

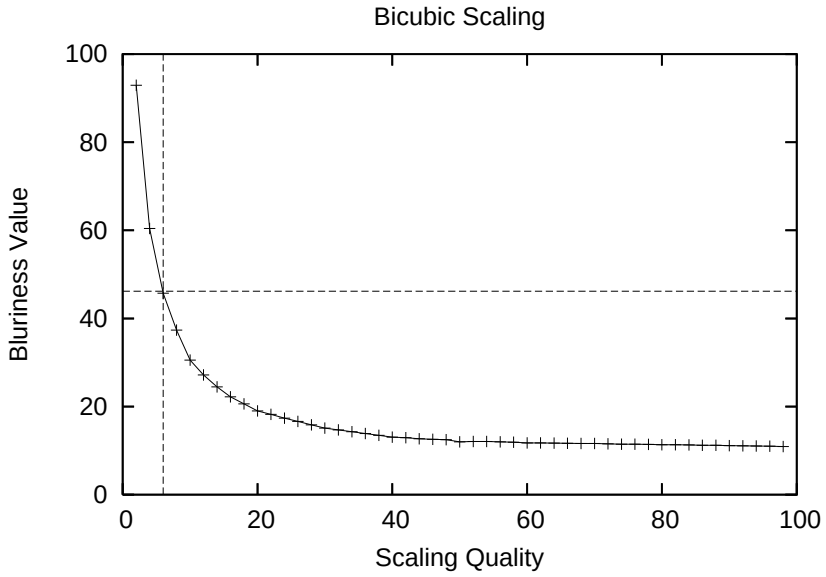
(a) Bicubic algorithm, scaling quality 10. (b) Bilinear algorithm, scaling quality 10.

Figure 6.5: Bicubic and bilinear scaling algorithms demonstrate a strong blurriness artifact. An example image is from Yale dataset.

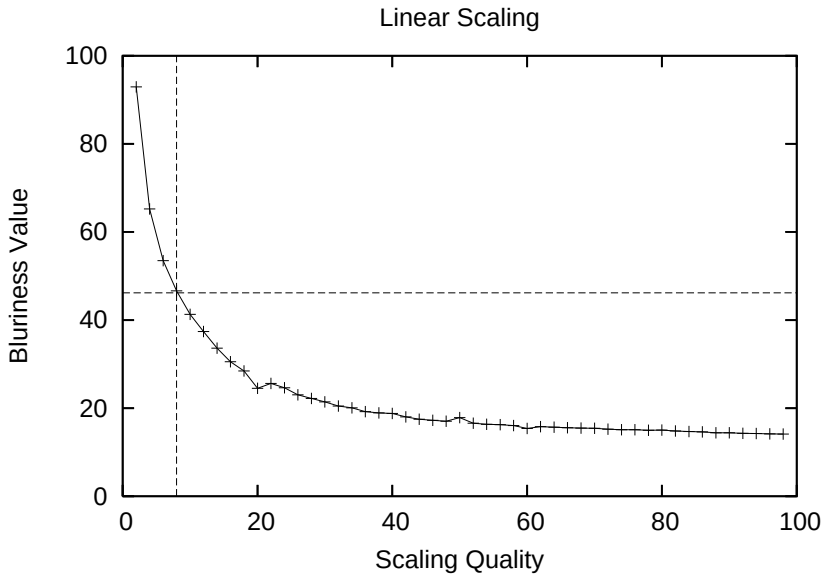
6.1.4 Mutual Information Metric

Visual artifact metrics can be used only with video adaptations that produce the measured artifacts. Such restriction causes inconvenience in using artifact metrics in practice. Therefore, it is desirable to have a video quality metric that is more independent of the way the video is degraded. In this section, we propose mutual information as such a metric and show that it suits face detection and face recognition algorithms.

Mutual information was first introduced in information theory (Shannon, 1948) and has proven itself as a good similarity metric in image registration. It measures the amount of statistical information two different images share about each other. It is easy to compute and it is



(a)



(b)

Figure 6.6: Blurriness metric vs. scaling quality for bicubic 6.6(a) and bilinear 6.6(b) scaling algorithms.

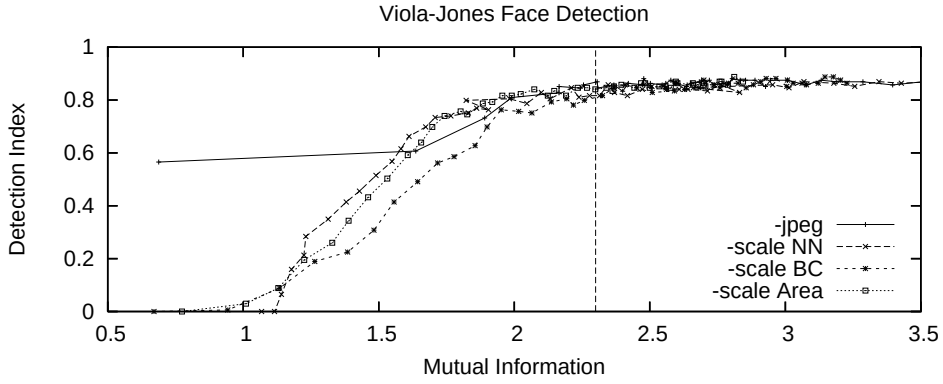
a more general measure of distortion compared to a visual artifact metric (such as blockiness), which focuses on a specific type of distortion. Also, mutual information is a better measure of video quality for video analysis algorithms than PSNR. This is because, for instance, mirroring an image to itself, while not affecting the performance of face detection or face recognition, changes its PSNR. Mutual information value, on the other hand, is not affected by such opera-

tions.

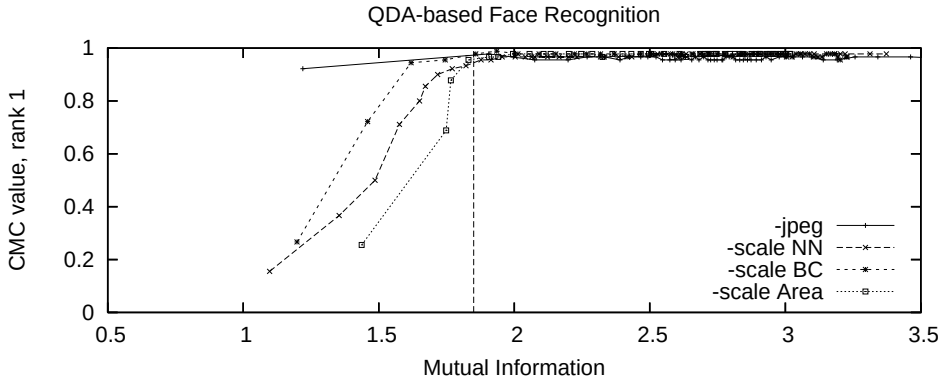
We demonstrate the advantages of mutual information by measuring the quality of video degraded with different types of video adaptations. In addition to previously used blocky adaptations (JPEG, nearest neighbor, and pixel area relation scaling), we also consider bicubic scaling algorithm, which adds a strong blurriness artifact to the degraded image. We conduct experiments for Viola-Jones face detection and QDA-based face recognition algorithms. Similar to experiments with blockiness and bluriness, we show that mutual information can be used as a metric of video quality for the selected algorithms. It means that a single threshold value of mutual information can be used to estimate the critical quality for a particular algorithm across various video adaptations.

To compare experimental results on mutual information for different adaptations, we plot the value of mutual information vs. the accuracy of a given video analysis algorithm. The results are presented in Figure 6.7(a), for face detection and in Figure 6.7(b), for face recognition. We explain how a single curve on the graph is obtained, using example of JPEG compression (marked as "jpeg") and face detection algorithm. Images from the MIT/CMU test dataset are compressed with JPEG compression qualities varying from 1 to 99. For each JPEG quality, we compute detection index of the face detection algorithm and the average mutual information, using original uncompressed images as references. Note that mutual information has lower value for more distorted images and higher value for less distorted. The resulted pair of detection index and mutual information represent one point on "jpeg" curve. Curves for scaling algorithms are obtained similarly. Curves marked as "scale NN", "scale BC", and "scale Area" correspond to nearest neighbor, bicubic, and area-based scaling respectively. For face recognition algorithm, Yale dataset is used (partitioned to probe and gallery subsets as described in Section 3.2), and cumulative match characteristic (CMC) rank one value (Grother et al., 2003) is computed.

Figure 6.7(a) demonstrates that a mutual information value between 2 to 2.3 can be considered as a threshold corresponding to the critical video quality for the face detection algorithm for the given set of images. The threshold is actually an interval, because the face detection algorithm is not very robust to high noises in images showing frequent fluctuations in accuracy.



(a)

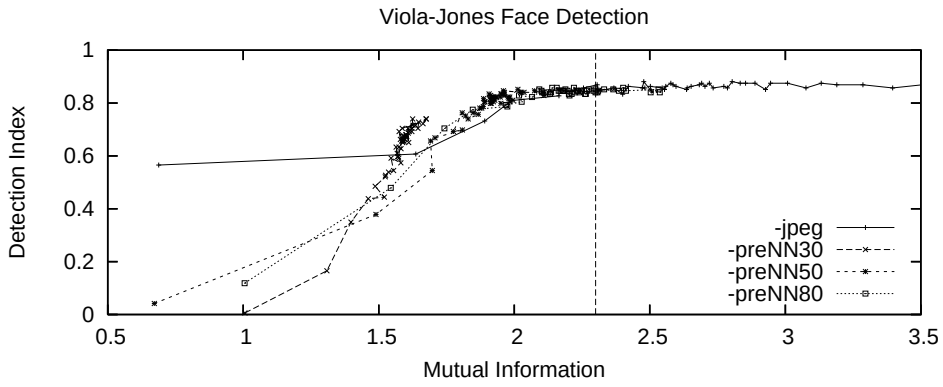


(b)

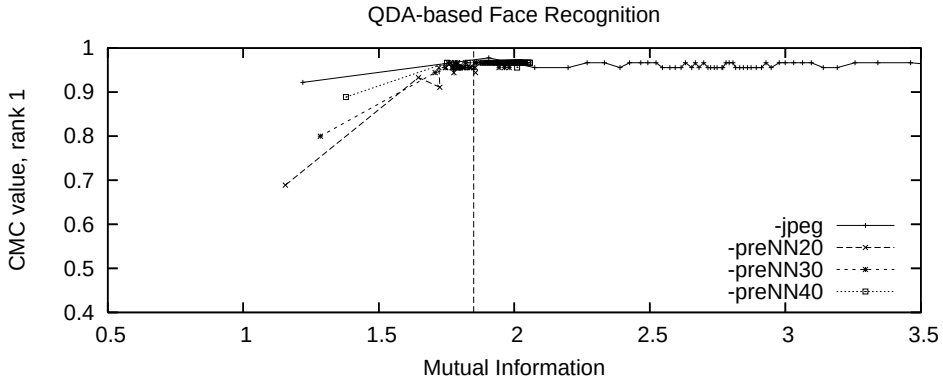
Figure 6.7: Mutual information vs. accuracy of face detection and face recognition algorithms. Different curves correspond to different types of video adaptations.

In practice, we can conservatively use 2.3 to be the threshold for mutual information metric, as indicated with the dashed vertical line in the figure. This value reflects the quality 17 for JPEG compression (which is between sweet spot value 9 and our conservatively selected critical quality 20), 54 for nearest neighbor, 48 for bicubic, and 52 for area scaling algorithms. Degrading images in MIT/CMU dataset to these qualities corresponds to approximately 12, 4, 5, and 6 times reductions in average image sizes.

For face recognition, the CMC rank one value is plotted against the value of mutual information in Figure 6.7(b). From the figure, the face recognition threshold value on mutual information can be conservatively set to 1.8. This value gives approximately 10, 11, 21, and 29 times reduction in Yale image sizes for JPEG compression, nearest neighbor, bicubic, and area scaling algorithms respectively.



(a)



(b)

Figure 6.8: Mutual information vs. accuracy of face detection and face recognition algorithms. Different curves correspond to different combinations of nearest neighbor scaling and JPEG compression.

Since blocky and blurry types of video adaptations were used in these experiments, it demonstrates that, compared to artifact metrics, mutual information is adaptation independent. Therefore, we can use mutual information to measure SNR quality for a combination of different video adaptations. For example, video frames can be scaled down first and then compressed with JPEG to achieve a higher bitrate reduction. We only need to make sure that for the resulted frames, the value of mutual information is above the threshold.

6.1.5 Combining Several Video Adaptations

Figure 3.5 shows a practical video surveillance scenario, where the surveillance video is reduced by scaling followed by compression. Combination of two adaptations allows even higher reductions in video size compared to using single adaptation (compression or scaling). We use nearest

neighbor scaling for its speed. It also shows the worst reduction results compared with other scaling algorithms. As described in Section 3.2, images from MIT/CMU and Yale datasets are degraded following this sequence: prescaled, compressed with JPEG, decompressed, and scaled back to their original resolution. The accuracy of Viola-Jones face detection and QDA-based face recognition algorithms are compared on the degraded and original images. Accuracy vs. mutual information are plotted in Figure 6.8(a) for face detection and in Figure 6.8(b) for face recognition. The vertical dashed lines indicate the mutual information's threshold values of 2.3 for face detection and 1.8 for face recognition that were found in Section 6.1.4. For face detection algorithm, the images were prescaled to 30%, 50%, and 80% of their original resolution, which correspond to curves "preNN30", "preNN50", and "preNN80". Images for face recognition algorithm were prescaled to 20%, 30%, and 40%.

By looking at Figure 6.8(a) and measuring the reduction in files sizes for the corresponding transformations, we notice that there is no apparent benefit in combining neighbor scaling and JPEG compression video adaptations for face detection algorithm. Only images prescaled to 80% and compressed with JPEG compression quality higher than 75 have mutual information larger than the threshold. Evidently, the best choice for face detection, because of simplicity and amount of bitrate reduction, is to apply a single JPEG compression with quality 17.

With face recognition, the situation is different (Figure 6.8(b)), since the accuracy of face recognition is not affected by pre-scaling as much as face detection algorithm. Hence, when using face recognition, we can achieve higher bitrate reduction by combining scaling and JPEG compression. By measuring resulted files sizes, we found that the best reduction in size is achieved by prescaling images to 30 with nearest neighbor and then compressing them with JPEG quality 20.

6.1.6 Lab Experiments

To verify the critical video qualities determined in the previous section for Viola-Jones face detection and face recognition algorithm in a practical scenario, we installed a video camera in our research lab and pointed it at the door (see the description of the test video given in

(a) (b)

Figure 6.9: An example of original video frame (JPEG compression value 90) used in practical tests (a) and an example of test frame scaled with nearest neighbor to 30% followed by JPEG compression with quality 20 (b).

Section 3.2). We degrade the original video frames to JPEG quality 20, as the critical SNR quality for face detection. For face recognition, we prescale the video with nearest neighbor algorithm to 30 percent first, then compress it with 20 JPEG quality. An example of the original camera frame shown in Figure 6.9(a) can be visually compared with the degraded frame in Figure 6.9(b). The resulted reductions in bandwidth are presented in Table 6.2. The reduction in bandwidth amounts to 3.9 times for face detection and 12.5 times for face recognition. If we also reduce original video frame rate from conventional 30 fps to 5 fps, which is a reasonable frame rate for detection and recognition, the reduction amounts to 23 times for face detection algorithm and 75 times for face recognition.

Video	Mutual Information	Bitrate (kbps)	Reduction
Original	-	4403.2	-
Video for FD	2.7158	1138.8	3.9
Video for FR	1.798	352.2	12.5

Table 6.2: The reduction of video bitrate: original video, degraded video for face detection (FD), and for face recognition (FR) algorithms.

We evaluated both video analysis algorithms with video degraded in the above manner, considering each frame as a separate image. Coordinates of faces detected by face detection algorithm were given as an input to the recognition algorithm. We evaluate the recognition

algorithm by using the verification, instead of identification, performance metric (Grother et al., 2003). The choice of evaluation metric is not essential to us, since we only concern with the consistency in algorithm's performance when the video is changed from the original high quality to the degraded low quality.

Practical evaluation showed that face detection algorithm correctly detected 144 out of 237 faces in images compressed with both JPEG quality 20 and 90. The algorithm, however, had falsely detected four faces for quality 20 and one face for 90. To avoid random false positives occurring due to algorithm's fluctuations, only faces that are present in three consecutive frames were counted as a real face. The detected faces from the degraded video, including false positives, were used as inputs to recognition algorithm. Recognition showed two false positives for degraded video (i.e., false positives from face detection were not recognized) and surprisingly five false positives for the original video. Since we used only one face per person in the gallery for verification, adding more faces to the gallery, may improve the recognition performance.

From our experiments, we can notice that the same type of degradation results in different mutual information values depending on the image types. This is because computation of mutual information requires the reference image. Therefore, in practice, two situations need to be considered: (i) finding the threshold on mutual information for the given video and (ii) checking if mutual information for current live frames exceeds the threshold. Since the original and degraded video frames are required for computing mutual information, during the normal operation of the system, its value should be computed at the video source for each frame. The threshold value on mutual information can be found interactively during the calibration stage of the system, by incrementally decreasing the video quality and evaluating the performance of video analysis algorithms. Another way is to build a table of typical thresholds values for different categories of images offline and use corresponding values in particular live scenarios.

Experiments with artifact and mutual information metrics demonstrate that once the corresponding threshold is found for a face detection or recognition algorithm, it can be used to determine critical SNR qualities for different video adaptations, e.g., JPEG compression or nearest neighbor scaling. To understand which metric to use and what metric's threshold is,

we reason about a video analysis algorithm (understand what video features it relies upon) and a video adaptation (determine how it degrades the video). Limited empirical experiments, however, are still required for finding metric's threshold for SNR quality.

Chapter 7

Temporal Quality Estimation

In the Chapter 5, we demonstrated how features from set $\mathcal{F}_V$ such as blockiness and blurriness help in estimating critical video quality when such non-trivial video adaptations like JPEG compression, and various scaling algorithms are used. In this chapter, we focus on object oriented features, $\mathcal{F}_O$, specifically on speed and size the moving object. As algorithm $\mathcal{A}$ we consider both blob tracking and CAMSHIFT object tracking. We demonstrate that unlike the $\mathcal{F}_V$ video features that required metrics and some number of experiments to estimate critical SNR quality, we can use a simple reasoning about features from $\mathcal{F}_O$ and frame dropping adaptation D to approximate temporal critical quality.

As shown in Section 3.2, we degrade temporal video quality by applying the dropping pattern "drop i frames out of $i + j$ frames", where i is *drop gap*, and j is the number of consecutive remaining frames (see Figure 1.3).

First, we present an estimation of the critical drop gap for an object tracking algorithm without taking into account the specific method of detection and tracking. For simplicity, consider a video containing a single moving object, which can be accurately tracked by the algorithm. We can notice that dropping frames affects the speed of object. Since video is a sequence of discrete frames, the speed of object can be understood as a distance between the centers of object positions in two consecutive frames, which we call inter-frame speed denoted as Δd. Without loss of generality, we can say that for every object tracking algorithm there exists a $\Delta\tilde{d}$ such that, if object moves for a larger distance than $\Delta\tilde{d}$, the algorithm loses it.

Let Δd_0 be the maximal inter-frame speed of the object in the original video, when no frame dropping is applied yet. If we drop frames with drop gap $i = 1$, the new maximum inter-frame speed can be approximated as $\Delta d_1 = 2\Delta d_0$. Then, for general frame dropping pattern, $\Delta d_i = (i + 1)\Delta d_0$. Assume we know the original speed of the object and the algorithm's threshold $\Delta\tilde{d}$. Then, we can compute the maximum number of consecutive frames that can be dropped, i.e., critical drop gap $\tilde{i}$, as

$$\tilde{i} = \frac{\Delta\tilde{d}}{\Delta d_0} - 1. \tag{7.1}$$

7.1 Blob Tracking Algorithm

For blob tracking algorithm, due to frame differencing detection, the value $\tilde{i}$ depends on the size and the speed of tracked object. If too many consecutive frames are dropped, the object in the current frame appear so far away from its location in the previous frame that the frame differencing operation results in detecting two separate blobs (see Figure 4.18(b)). Such tracking failure occurs when the distance between blob detected in the previous frame and blob in the current frame is larger than the size of the object itself. Therefore, this distance is the threshold distance $\Delta\tilde{d}$. To determine its value, we need to estimate the coordinates of the blob center in the current frame, which depend on its location and size in the previous frame.

In this analysis, we assume a single object monotonously moving in one direction. Although this assumption considers only a simplified scenario, many practical surveillance videos include objects moving in a single direction towards or away from the camera view. Also, such movements of the object in camera's view as rotating or only changing in size (when object goes away/towards camera view but does not move sideways) do not have a significant effect on frame differencing object detection. We also assume, without loss of generality, that the object moves from left to right with its size increasing linearly. The assumption allows us to consider only changes in coordinate x, and width w. Increase/decrease in size is important because when tracked objects approach or move away from the camera, their size changes. In practice, when object moves in both x and y coordinates, the overall critical drop gap would be the minimum of the two values estimated for corresponding coordinates.

Consider the original video when no frames are dropped. We assume the average distance between fronts of the blob when it shifts from the previous frame to the current frame is Δx^0. We consider the front of the object because it is more accurately detected by frame differencing. When frame differencing is used, the resulted detected blob is the union of the object presented in the previous and current frames (see Figure 4.18(b)). Therefore, when we drop frames, the width of the blob in the frame following after the drop gap will be larger than that in the original video sequence (see Figure 4.17 for illustration). However, the front of the blob would be detected in the same way as in the original video.

Since frame dropping affects size of the detected object, we consider average change in size as Δw^0. The superscript indicates the size of the drop gap, which is 0 when frames are not dropped. Assume that x_k^0 is x-coordinate of blob's center in k-*th* frame, then, we can estimate its coordinate in the frame $k+i+1$ as following,

$$x_{k+i+1}^0 = x_k^0 + (i+1)\Delta x^0 - (i+1)\frac{\Delta w^0}{2}. \tag{7.2}$$

If i frames are dropped after frame k, the detected blob in the $k+i+1$ frame is the union of actual object appearing in frames k and $k+i+1$ (as Figure 4.17 illustrates). Then, the width difference $(w_{k+i+1}^i/2 - w_k^0/2)$ can be approximated as $(i+1)\Delta x^0/2$. Therefore, the blob's center in the $k+i+1$ frame can be estimated as,

$$x_{k+i+1}^i = x_k^i + (i+1)\Delta x^0 - (i+1)\frac{\Delta x^0}{2} = x_k^0 + (i+1)\frac{\Delta x^0}{2}, \tag{7.3}$$

since $x_k^i = x_k^0$.

As was mentioned, $\Delta\tilde{d} = |x_{k+\tilde{i}+1}^{\tilde{i}} - x_k^{\tilde{i}}|$, where $\tilde{i}$ indicates the critical drop gap. The failure of the blob tracking implies that $\Delta\tilde{d} = w_k^0$, where value w_k^0 is the width of the blob detected in frame k. Therefore, from equation (7.3), we obtain $w_k^0 = \Delta\tilde{d} = (\tilde{i}+1)\frac{\Delta x^0}{2}$, from which we can find the critical drop gap to be

$$\tilde{i} = \frac{2w_k^0}{\Delta x^0} - 1. \tag{7.4}$$

In practice, values w_k^0 and Δx^0 can be determined by either keeping the history of speed and size of tracked object or by estimating their average values for a particular surveillance site.

In addition to the estimation of the critical drop gap for blob tracking, we can estimate the dependency function between accuracy of the algorithm and video frame rate. Such estimation is possible because of the way drop gap affects the accuracy of the frame differencing object detection algorithm used in blob tracking. We can define blob detection error for a particular frame as the distance between blob centers detected in this frame for the degraded video (with dropped frames) and the original video. Then, the average error, denoted as ϵ_{ij}, is the average blob tracking error for all frames in the video. This ϵ_{ij} function can be used as accuracy metric for the blob tracking depicting the tradeoff between tracking accuracy and video frame rate.

Using equations (7.2) and (7.3) we can estimate the blob tracking error for $k+i+1$ frame as following,

$$\left|x^i_{k+i+1} - x^0_{k+i+1}\right| = (i+1)\left|\frac{\left(\Delta x^0 - \Delta w^0\right)}{2}\right| = (i+1)C, \tag{7.5}$$

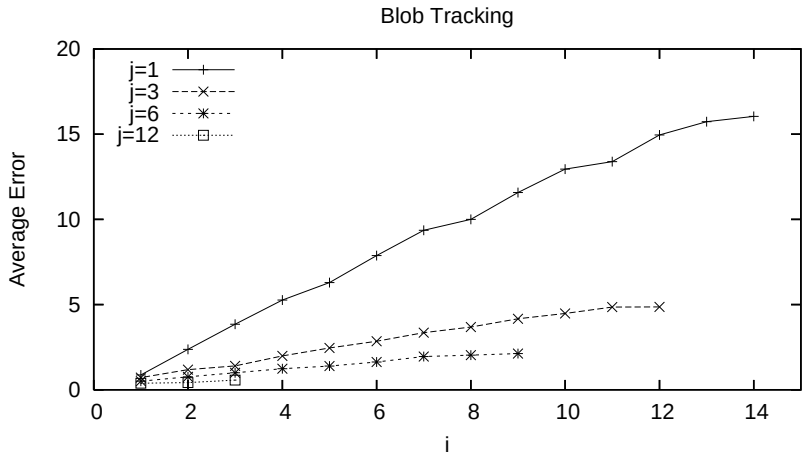
where constant $C \geq 0$ depends on the size and the speed of object in the original video.

Since we apply the dropping pattern "drop i frames out of $i+j$ frames", we need to estimate the blob tracking error for each of the remaining j frames in the video. There is no error in detecting blob for $j-1$ frames that do not have drop gap in front of them, i.e., for these frames, the result of the frame differencing would be the same as in original video with no dropping. Therefore, the average error for all j frames is the error estimated for the frame, which follows the drop gap (equation (7.5)) divided by j:

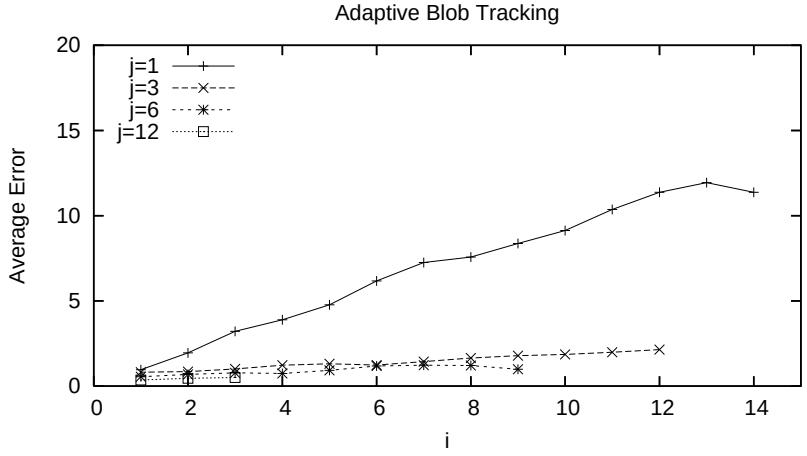
$$\epsilon_{ij} = \frac{i+1}{j}\left|\frac{\left(\Delta x^0 - \Delta w^0\right)}{2}\right| = \frac{i+1}{j}C. \tag{7.6}$$

Note the important property of this function that the average error is proportional to i and inversely proportional to j.

We performed experiments to validate the estimation of the average blob tracking error ϵ_{ij}. We use several videos from ViSOR video database, PETS2001 datasets, as well as videos we shot on campus with a hand-held camera (example screenshots in Figure 4.18(a), Figure 3.4(e), and Figure 3.4(f)). Videos include moving cars, person on a bicycle and people walking in a distance. We ran blob tracking algorithm on these videos and applied different dropping patterns. We plot the resulted average error against drop gap i when value j is 1, 3, 6, and 12.



(a)

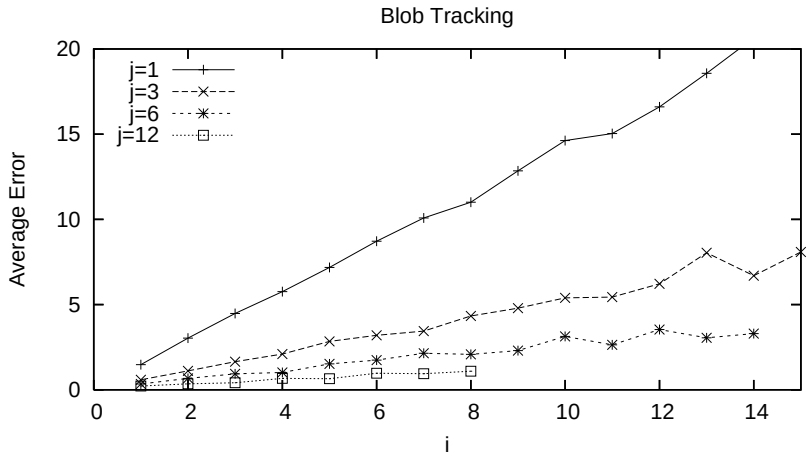


(b)

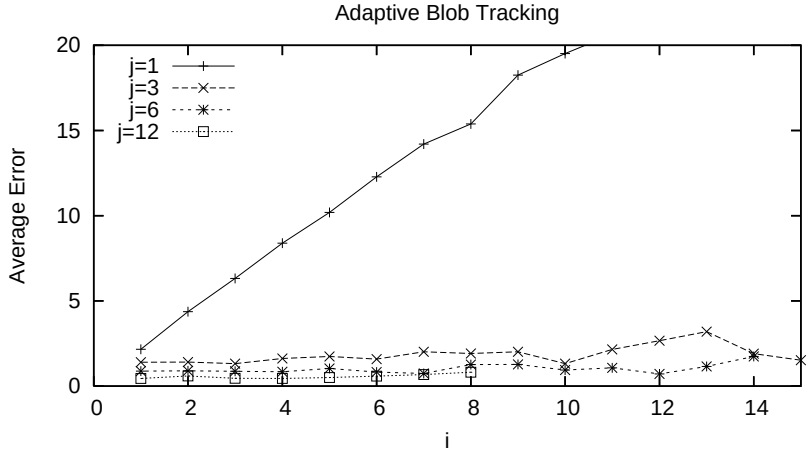
Figure 7.1: Accuracy of original and adaptive blob tracking algorithm for PETS2001 video (snapshot in Figure 4.18(a)).

The results are shown in Figure 7.1(a) (original video is 158 frames of 384 ×288, 30 fps) and Figure 7.2(a) (original video is 148 frames of 320 ×256, 30 fps).

Figure 7.1(a) shows the resulted average tracking error plotted against the drop gap i when value j is 1, 3, 6, and 12. It can be noted from the Figure 7.1(a) that for each fixed value j the average error is proportional to i. Also, average error is inversely proportional to j, as indicated by the angles of each line in the graph (for instance, angle of the line marked as "j=1" is three times larger than the angle of the line "j=3"). Figure 7.2(a) demonstrates similar results. These experimental results strongly support our analytical estimation of the average error given in the equation (7.6). The figures do not reflect the critical drop gap value because even for large drop



(a)



(b)

Figure 7.2: Accuracy of original and adaptive blob tracking algorithm for VISOR video (snapshot in Figure 3.4(f)).

gaps the blob tracking did not lose the track of the car in this test video sequence.

7.2 CAMSHIFT Algorithm

CAMSHIFT object tracking (Boyle, 2001) relies on color histogram detection and mean shift algorithm for tracking. The algorithm searches for a given object's histogram inside a subwindow of the current frame of the video, which is computed as 150% of the object size detected in the previous frame. Therefore, if the object, moves between two frames from its original location for a distance larger than half of its size, the algorithm will lose the track of the object. Hence, assuming we drop i frames before frame $k+i+1$, the threshold distance $\Delta\tilde{d} = \frac{w_k^0}{2}$, where w_k^0 is

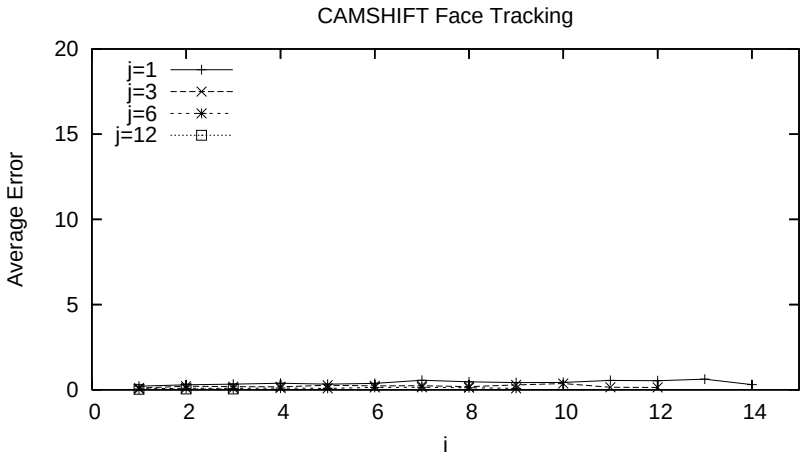


Figure 7.3: Accuracy of original and adaptive CAMSHIFT tracking algorithm for video with slow moving face (snapshot in Figure 4.15(a)).

the width of the blob detected in frame k. Since CAMSHIFT does not use frame differencing, drop gap does not have an additional effect on object's size. Therefore, we can estimate the center of the blob after drop gap i using the equation (7.2) instead of equation (7.3). Hence, the critical drop gap can be derrived as

$$\tilde{i} = \frac{w_k^0}{2\Delta x^0 - \Delta w^0} - 1. \tag{7.7}$$

Estimating the average tracking error loses its meaning for CAMSHIFT tracking because it uses a simple threshold for detection of the object in the current frame. If the drop gap of the given frame dropping pattern is less than critical drop gap in equation (7.7), the algorithm continue tracking the object, otherwise it loses it. And the critical drop gap depends on the changes in speed and size of the object.

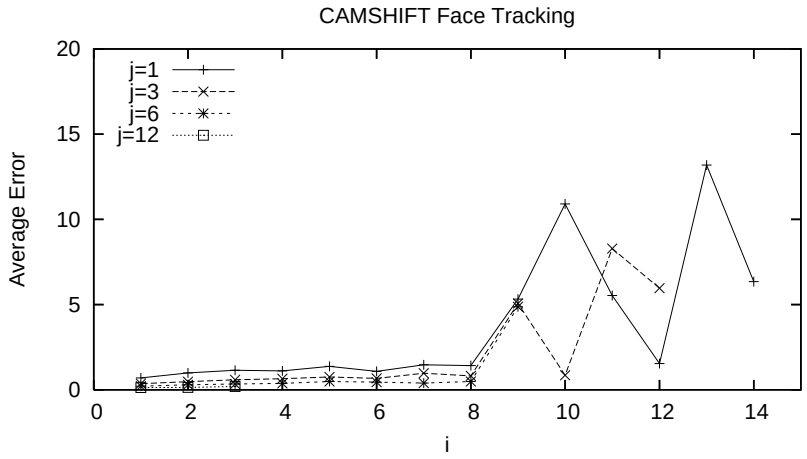
We performed experiments with CAMSHIFT tracking algorithm to verify our analytical estimation of the critical drop gap (equation (7.7)). We used several videos of a moving face shot with a simple web-cam, videos of talking heads by SEQAM laboratory and some movie clips (example screenshots in Figure 4.15(a) and Figure 3.4(d)). Figure 7.4(a) (original video is 600 frames of 352 $\times$288, 30 fps) and Figure 7.3 (original video is 303 frames of 320 $\times$230, 30 fps) show average tracking error vs. drop gap for CAMSHIFT tracking and various frame dropping patterns. Figure 7.3, corresponding to the video of a talking head (see snapshot in Figure 3.4(d)), demonstrates that tracking algorithm does not lose the face even when drop gap

is 14 frames. The reason is because the face in the video does not move around and is always present in the search subwindow of CAMSHIFT tracker. However, for the experiments shown in Figure 7.4(a), the video with fast moving head was used (see snapshot in Figure 4.15(a)). It can be noted that the algorithm does not lose the face until value of drop gap is 8, because for the smaller drop gaps, the face is still within a search subwindow and can be detected by the histogram matching. The fluctuations in the average error for the larger drop gaps appear because the face is either lost by the tracker or, for some large enough gaps, it would move out of the subwindow and move back in, hence the tracker does not lose it. We conducted experiments with more videos and observed that the critical drop gap value is smaller for videos with faster moving faces and larger for videos with slower moving faces. These observations agree with equation (7.7).

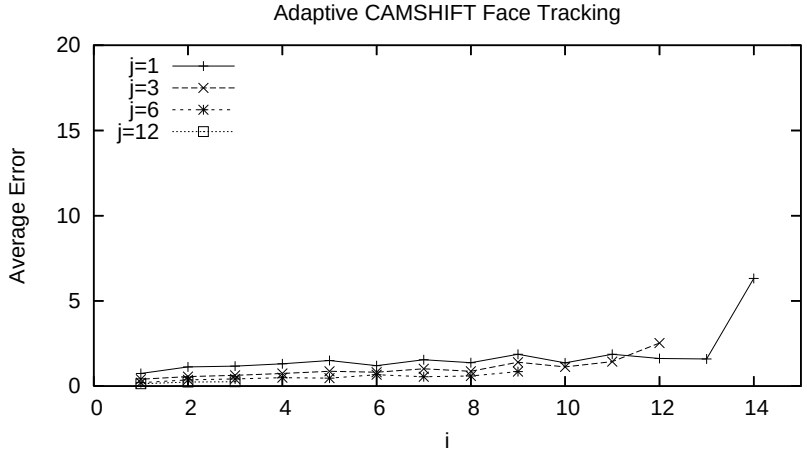
7.3 Adaptive Tracking

We propose to modify blob tracking and CAMSHIFT algorithms and make them more tolerant to video with low frame rate. We have shown that average error and the critical frame rate of tracking algorithms depend on speed and size of the object in the original video. Therefore, if we record these characteristics for previous frames, the location and the size of object in the frame that follows a drop gap can be approximated. Adjusting to frame dropping in such way allows us to reduce the average error for blob tracking algorithm and increase the critical drop gap for the CAMSHIFT algorithm.

Blob tracking algorithm tracks the detected foreground object using the simplified version of Kalman filter: $x_k = (1 - \alpha)x_{k-1} + \alpha z_k$, where x_k and x_{k-1} represent estimated coordinates of the object in the current and previous frames, z_k is the output of the object detector, and $\alpha \leq 1$ is some constant. When $\alpha = 1$, then the tracker trusts the measurement z_k fully and its average error can be estimated by equation (7.6). In cases when $\alpha < 1$, the accuracy of the tracking against the frame dropping worsens, due to the larger shifts in blobs' centers for videos with high drop gap. We propose using adaptive Kalman filter (Welsh & Bishop, 2001) to make blob tracking more tolerant to the frame dropping. We apply the filter only to the width of



(a)



(b)

Figure 7.4: Accuracy of original and adaptive CAMSHIFT tracking algorithm for video with fast moving face (snapshot in Figure 4.15(a)).

the object, because the front is detected correctly by frame differencing (see Figure 4.17). The filter can be defined as following,

$$\tilde{w_k} = w_{k-1} + K_k\,(w_{k-1} + u_k) \qquad \tilde{P_k} = P_k + Q_k \tag{7.8}$$
$$P_k = (1 - K_k)\tilde{P_k} \qquad K_k = \frac{\tilde{P_k}}{(\tilde{P_k} + R_k)},$$

where Q_k and R_k are the process and measurement noise covariances; $\tilde{w_k}$ is the new estimate of the blob's width in the current frame; w_{k-1} is blob's width in the last not dropped frame; u_k is the width measurement provided by the frame-differencing based detector.

Kalman filter depends on correct estimation of the error parameters, Q_k and R_k. By looking at Figure 4.17, we can set $Q_k = (i\Delta w^0)^2$, which estimates how big the tracked object should

be at frame $k+i+1$ compare to its width before the drop gap at frame k. R_k is essentially the error of the measurement, i.e., the output of the foreground object detector, therefore, $R_k = (w^i_{k+i+1} - w^0_{k+i+1})^2$.

Since w^i_{k+i+1} can be estimated as $w^0_k + (i+1)\Delta x^0$ and w^0_{k+i+1} as $w^0_k + (i+1)\Delta w^0$, we can approximate $R_k = (i+1)^2(\Delta x^0 - \Delta w^0)^2$. We obtain the values of Δw^0 and Δx^0 by recording the speed of the object and how fast it grows in size using last two available frames.

To compare how adaptive Kalman filter improves the accuracy of blob tracking, we performed the same experiments varying frame dropping pattern. The average error for blob tracking with adaptive Kalman filter is plotted in Figure 7.1(b) and Figure 7.2(b), which can be compared to results with original algorithm in Figure 7.1(a) and Figure 7.2(a) respectively. We can note that the accuracy of the adaptive blob tracking algorithm is improved for larger drop gaps (larger frame rate reduction). In both figures, Figure 7.1(b) and Figure 7.2(b), the angles of the lines in the graph are not inversely proportional to j anymore, giving fundamentally different bound on the average error. All lines with $j > 1$ are almost parallel to x-axis. It means that Kalman filter adapts very well to the drastic changes in speed and size of the object that occur due to the frame dropping. The constant increase in the average error for $j = 1$, is because, for such dropping pattern, all remaining frames are separated by drop gaps. In this scenario, adaptive Kalman filter accumulates the approximation error of object's size and speed. Therefore, the critical frame rate can be achieved with j that is at least equal to 2. If we take $i = 12$, the original frame rate is reduced by 7 times.

We also modified the CAMSHIFT tracking algorithm, adjusting the size of its search subwindow to the frame dropping. We simply increased the subwindow size in the current frame by $i\Delta x^0$, where i is the drop gap. The average error of this adaptive CAMSHIFT algorithm for the video with fast moving face is shown in Figure 7.4(b). Comparing with the results of original algorithm in Figure 7.4(a), we can notice that the adaptive tracker performs significantly better for the larger drop gaps. The experiments show that we can drop 13 frames out of 14 with a tradeoff in small average error. It means that CAMSHIFT algorithm, for this particular video sequence, can accurately track the face with frame rate reduced by 13 times from the original.

For the news videos of talking heads, where face does not move significantly around, adaptive algorithm performs with exactly the same accuracy results as the original algorithm. Therefore, Figure 7.3 illustrates essentially both versions of the algorithm, original and adaptive. These experiments demonstrate that by using analysis to modify CAMSHIFT algorithm, we can improve its performance on videos with fast moving faces, while retaining the original accuracy on videos with slow moving faces.

Chapter 8

Conclusion

In this thesis, we evaluated the effect of video quality degradation on several typical examples of video analysis algorithms. The surprising finding of this study is that tested algorithms show very high tolerance towards large reductions in video quality. Demonstrated consistency in accuracy for low video quality amounts to at least 10 times lesser video bitrate than a conventional requirement of human visual system.

We argued that an algorithm-oriented video quality metrics need to be developed. Metrics based on visual artifacts, blockiness and blurriness as examples, and mutual information were suggested. Artifact metrics show more precision when used to estimate critical video quality for a given video analysis algorithm and video adaptation. Mutual information, however, is not only easier to compute, it is also less dependent on the type of video adaptation, making it more practical.

Our analysis of tracking algorithms have shown that better algorithms can be designed with high tolerance towards low video quality. We demonstrated that by using extra information about tracked object, blob and face tracking algorithms can be modified so, their performance on low quality video improves by a magnitude.

The main limitation of the thesis is the fact that video analysis algorithms are heterogeneous in their nature. Therefore, the results of the study cannot be generalized to other algorithms except those, for which experimental results are presented. However, we believe that non-trivial and useful video analysis algorithms can be classified in a limited number of groups that show

similar responses in terms of accuracy to various reductions in video quality.

Video analysis algorithms in their core often use empirical data or are training-based. Such lack of the determinism makes it impossible to fully formalize the behavior of the algorithms. Therefore, the idea that common video analysis algorithms require lesser video quality than humans needs to be supported with more experiments on typical examples of algorithms. Changes in algorithms' accuracies need to be studied for major video adaptations used in practical systems, i.e., commonly used video encoders.

Another important limitation is the "academic" setup of our experiments with standard datasets and lab-shot videos used for testing. Performing experiments in the controlled environment unarguably have a positive effect on the obtained results. Some of the conditions that can weaken the performance of video analysis algorithms with low quality video can include poor lighting, object occlusions, a tracked object moving with a variable speed or in a circle. Poor performance of the algorithm under such conditions, however, would be mostly due to its imperfection. Based on our own experience and observation, the degradation of the video quality would not have a significant effect on the performance on average, but the results would not show a convincing pattern. The logical notion "falsity implies anything" could be used to describe the situation. Nevertheless, we strongly believe that our findings, to a high degree, would still remain true in practical systems and environments. However, a deeper study of the relationships between analysis algorithms and video quality would greatly benefit building more robust and efficient automated intelligent systems with video analysis.

Overall, the results of the study strongly suggest that it is impractical and inefficient to treat video analysis algorithms in the same manner as a human video observer. The resource economical video analysis algorithms can and should be designed. The encoding algorithms better matching the computer vision need to be developed as well. This study shows that, in terms of video quality and video encoding, computer vision is very different from human vision.

8.1 Related Publications

Korshunov, P., & Ooi, W. T. (2005). Critical video quality for distributed automated video surveillance. *Proceedings of the 13th ACM International Conference on Multimedia, ACMMM'05* (pp. 151–160), Singapore, November, 2005.

Korshunov, P. (2006). Rate-accuracy tradeoff in automated, distributed video surveillance systems. *Proceedings of the the 14th ACM International Conference on Multimedia, ACMMM'06* (pp. 887–889), Santa-Barbara, USA, October, 2006.

Korshunov, P., & Ooi, W. T. (2010). Reducing frame rate for object tracking. *In proceedings of the 16th International MultiMedia Modeling Conference, MMM'10* (pp. 454–464), Chongqing, China, January, 2010.

Korshunov, P., & Ooi, W. T. (2012). Video quality for face detection, recognition and tracking. *To appear in ACM Transactions on Multimedia Computing, Communications and Applications journal, ACM TOMCCAP (the paper is accepted)*, 2012.

Appendix A

Prototype of the Video Surveillance System

To test our experimental findings in a practical environment, we have built a prototype of the video surveillance system. Although the system is fairly simple with only one camera, one proxy, and one monitor station, its importance is the presence of real devices and the IP-network, which allow us to demonstrate the practical application of the critical video quality. The prototype uses a Canon VCC4 camera connected to an LML33 capture card, one computer as a processing proxy, and another computer serving as a monitoring station. To transmit and display video, we use the OpenMash[1] framework. Together with OpenMash we adopted its extension called Indiva (Ooi, Pletcher, & Rowe, 2004), which allows us remotely control the compression quality, frame rate of the video captured from the camera, and gather necessary statistics. We use Viola-Jones face detection and CAMSHIFT tracking as the examples of video analysis algorithms, which runs on the proxy processing the incoming video from the camera. Also, only SNR video quality was degraded using MJPEG and H.261 encoders.

In this experimental setup, we assume that the critical video quality for a given video analysis algorithm and video adaptation is known (through off-line profiling or estimation). In the case of Viola-Jones face detection and compression, we take conservative value of 20,

[1] www.openmash.org

assuming the JPEG compression value (see experiments presented in Section 4.1). Our video surveillance system can dynamically adjust the rate of streaming video depending on the result of the face detection. When there is no face detected in the video, the camera can stream low quality video to the processing proxy. In this case, the proxy would be in "observe" mode, continuously running video analysis algorithms on low quality video without relaying it to the monitor. In this scenario, we are saving the bandwidth on the link between the camera and proxy by streaming low bitrate video, and we do not use any bandwidth on the link between proxy and monitor. Once the algorithm detects something in the video, the proxy requests the video source to raise the quality of the video to the quality suitable for human visual system and relay it to the monitor, thus alerting the end user. In this scenario, the proxy would be in "alert" mode. Hence, in the observe mode, usage of network bandwidth is minimized, and in alert mode, full quality video is transmitted from video source to monitor.

The experiments on the prototype system are carried out in an office-like environment. We use video of size 352×288. Faces appearing in a video generally have eyes, nose, and mouth within a 20×20 pixels square. We run our system in several scenarios for both the MJPEG and H.261 video encoders, the two main encoders available in OpenMash.

To verify our experimental findings presented in Section 4.1, we run our system with changing compression quality every three seconds, ranging from 90 to 1 and decreasing by 2 every time. We use scenarios where one person is sitting in front of the camera, moving her head and talking. The sample shots are shown in Figure A.1(c) and Figure A.1(d). We run the system in such scenarios eight times each, using the MJPEG and H.261 encoders. For faces that have eyes, nose and mouth within a square of 10x10 pixels size (e.g., Figure A.1(d)), the detection index demonstrates unpredictable fluctuations. Faces that are bigger in size (e.g., Figure A.1(c)) are correctly detected at least until compression quality is reduced to 15. These observations are consistent with our experimental results on images from both the MIT/CMU data set and our own lab surveillance.

Our prototype system can dynamically adapt the bit rate for surveillance video according to the current result of the face detection algorithm. When no face is detected, the system

runs in observe mode, using only a small amount of bandwidth. Video is compressed with quality equal to 20, and the proxy does not relay it to the monitor. Once a face is detected, the system automatically switches to alert mode by changing compression quality to 90, and relays the video to the monitor to alert the user. The system switches back to observe mode when no face is detected. We run the prototype on a video scene with a person walking in and out of the camera's view. The sample shots of the video used are shown in Figure A.1(a) and Figure A.1(b). The system successfully detects faces and changes to alert mode in accordance with our experimental findings.

We collect the bit rate for the MJPEG and H.261 encoders during a period of 100 seconds. The collected data is shown in Figure A.2(a) and Figure A.2(b). The figures show that when there are no faces detected, i.e., the compression quality is reduced to 20, the bandwidth on average is reduced up to 94% for the H.261 encoder and up to 72% for the MJPEG encoder. The H.261 encoder demonstrates higher reduction in bandwidth for videos with static background due to its conditional replenishment algorithm (McCanne & Jacobson, 1995). The important thing to note is that the frame rate remains at 30 fps throughout the experiment. Since the frame rate of the video is less important for face detection, we can further reduce the frame rate to 5 fps in observe mode. By doing so, we obtain bandwidth reduction of up to 35 times for the H.261 encoder and up to 29 times for the MJPEG encoder.

The above experiments are conducted on a video scene with static background. In our ex-

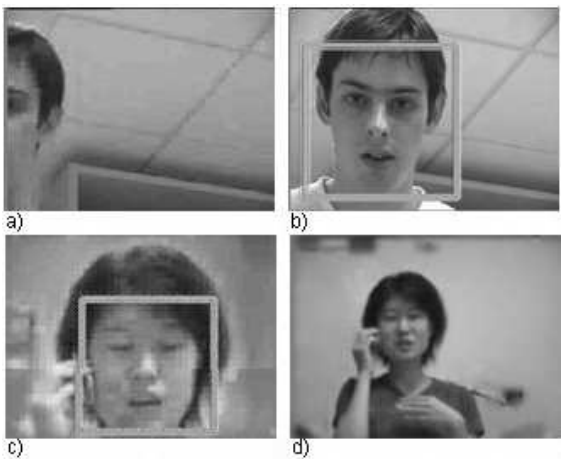
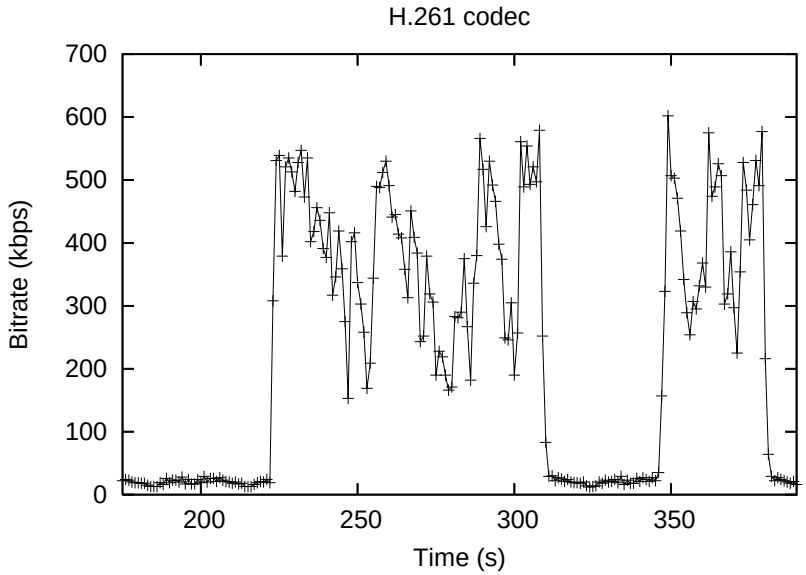
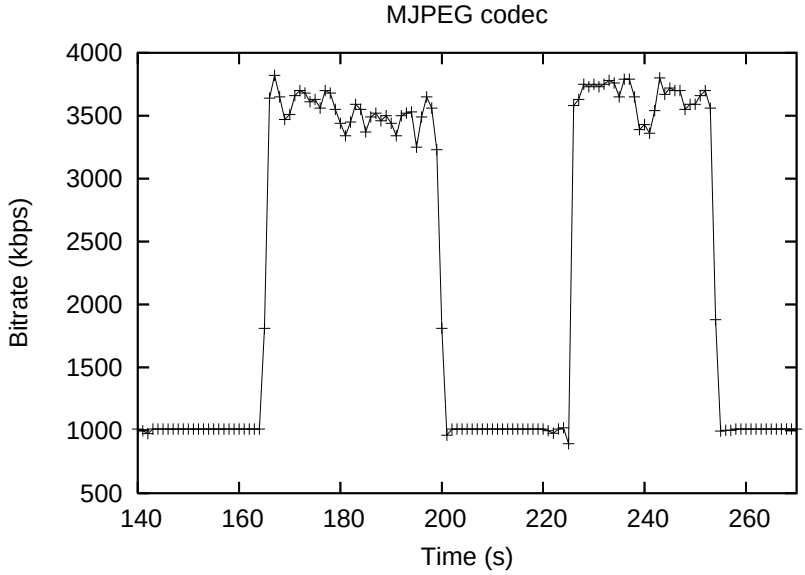


Figure A.1: Sample video shots used in experiments on the prototype video surveillance system.



(a)



(b)

Figure A.2: Video bitrate when a face comes in and out of the camera's view for H.261 and MJPEG video codecs.

periments on video scene with intensive background motions, the effect of motion on bandwidth reduction is significantly reduced, showing mainly the effect caused by a decrease in compression quality. With these conditions, we can still obtain up to six times bandwidth reduction for the H.261 encoder. For MJPEG, there is no significant differences in the bandwidth measurement since the MJPEG format is not motion compensated.

In similar experiments on the CAMSHIFT face tracker, the way the tracking algorithm was

used in our prototype is different. Usually, the tracking algorithm is used to support higher level tasks such as detecting suspicious behavior, identifying a running or falling person, group tracking, etc. Therefore, the decision whether to stream video to the user or not would be made by those algorithms. We do not implement such high level algorithms. Therefore, instead of switching between observe mode and alert mode, we simply run the tracking algorithm on the video with the suggested critical video quality of compression 50 and frame rate of 6 fps. Such settings lead to an MJPEG bit rate of 175 kbps on average, giving us 16 times reduction in the bandwidth.

A possible concern is the latency caused by switching from observe to alert mode. Such latencies might cause high quality video frames of suspicious events to be lost. To address this concern, we measure the latency between when a face is detected, and when high quality video is received at the monitor in our prototype. This delay is found to be at most 100 ms. A caveat is that our prototype system runs over a local area network. This latency might increase if the system is deployed over a wide-area network.

References

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Boyle, M. (2001). *The effects of capture conditions on the CAMSHIFT face tracker* (Technical Report 2001-691-14). Department of Computer Science, University of Calgary, Alberta, Canada.

Bradski, G. R. (1998). Computer vision face tracking as a component of a perceptual user interface. *Proceedings of the Forth IEEE Workshop on Applications of Computer Vision, WACV'98* (pp. 214–219), Princeton, NJ, January, 1998.

Chang, S.-F., & Anthony, V. (2005). Video adaptation: Concepts, technologies, and open issues. *Special Issue on Advances in Video Coding and Delivery, Proceedings of IEEE*, *93*(1), January, 2005, 148–158.

Chung, Y.-C., Wang, J.-M., Bailey, R., Chen, S.-W., & Chang, S.-L. (2004). A non-parametric blur measure based on edge analysis for image processing applications. *Proceedings of the IEEE international conference on Cybernetics and Intelligent Systems, CIS'04*, Vol. 1 (pp. 356–360), Singapore, December, 2004.

Collins, R., Lipton, A., Kanade, T., Fujiyoshi, H., Duggins, D., Tsin, Y., Tolliver, D., Enomoto, N., & Hasegawa, O. (2000). *A system for video surveillance and monitoring* (Technical Report CMU-RI-TR-00-12). Carnegie Mellon University: Robotics Institute.

Delac, K., Grgic, M., & Grgic, S. (2005). Effects of JPEG and JPEG2000 compression on face recognition. *Lecture Notes in Computer Science, Pattern Recognition and Image Analysis*, *3687*, August, 2005, 136–145.

Eickeler, S., Muller, S., & Rigoll, G. (2000). Recognition of JPEG compressed face images based on statistical methods. *Image and Vision Computing Journal, Special Issue on Facial Image Analysis*, *18*, March, 2000, 279–287.

Eleftheriadis, A., & Anastassiou, D. (1995). Constrained and general dynamic rate shaping of compressed digital video. *Proceedings of the IEEE International Conference on Image Processing, ICIP'95* (pp. 396–399), Washington, DC, USA, October, 1995.

Freund, Y., & Schapire, R. E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. *Proceedings of the Computational Learning Theory, Second European Conference, EuroCOLT'95* (pp. 23–37), Barcelona, Spain, March, 1995.

Funk, W., Arnold, M., Busch, C., & Munde, A. (2005). Evaluation of image compression algorithms for fingerprint and face recognition systems. *Proceedings of 6th Annual IEEE*

SMC Information Assurance Workshop, IAW'05 (pp. 72–78), Darmstadt, Germany, June, 2005.

Gibbons, P. B., Karp, B., Ke, Y., Nath, S., & Seshan, S. (2003). Irisnet: An architecture for internet-scale sensing. *Proceedings of the 29th international conference on Very large data bases, VLDB'03*, Vol. 29 (pp. 1137–1140), Berlin, Germany, September, 2003.

Girgensohn, A., Kimber, D., Vaughan, J., Yang, T., Shipman, F., Turner, T., Rieffel, E., Wilcox, L., Chen, F., & Dunnigan, T. (2007). DOTS: Support for effective video surveillance. *Proceedings of the 15th ACM International Conference on Multimedia, ACMMM'07* (pp. 423–432), Augsburg, Germany, September, 2007.

Grother, P. J., Micheals, R. J., & Phillips, P. (2003). Face recognition vendor test 2002 performance metrics. *Proceedings of the 4th International Conference on Audio Visual Based Person Authentication, AVBPA'03* (pp. 937–945), Guildford, UK, June, 2003.

Haralick, R. M., & Shapiro, L. G. (1993). *Computer and robot vision*, Vol. 1. Addison-Wesley.

Hjelmas, E., & Low, B. K. (2001). Face detection: A survey. *Computer Vision and Image Understanding*, *83*(3), July, 2001, 236–274.

Javed, O., Rasheed, Z., Alatas, O., & Shah, M. (2003). KNIGHT^M: A real-time surveillance system for multiple overlapping and non-overlapping cameras. *Proceedings of the IEEE International Conference on Multimedia and Expo, ICME'03*, Vol. 1 (pp. 649–652), Baltimore, Maryland, July, 2003.

Javed, O., & Shah, M. (2002). Tracking and object classification for automated surveillance. *Proceedings of the 7th European Conference on Computer Vision, ECCV'02* (pp. 343–357), Copenhagen, Denmark, May, 2002.

Kim, J., Wang, Y., & Chang, S.-F. (2003). Content-adaptive utility-based video adaptation. *Proceedings of the IEEE International Conference on Multimedia and Expo, ICME'03*, Vol. 3 (pp. 281–284), Baltimore, Maryland, July, 2003.

Kim, M., & Altunbasak, Y. (2001). Optimal dynamic rate shaping for compressed video streaming. *Proceedings of the International Conference on Networking, ICN'01* (pp. 786–794), Colmar, France, July, 2001.

Li, L., Huang, W., Gu, I. Y., & Tan, Q. (2003). Foreground object detection from videos containing complex background. *Proceedings of the 11th ACM International Conference on Multimedia, ACMMM'03* (pp. 2–10), Berkeley, CA, USA, November, 2003.

Lu, J., Plataniotis, K. N., & Venetsanopoulos, A. N. (2003). Regularized discriminant analysis for the small sample size problem in face recognition. *Pattern Recognition Letters*, *24*, December, 2003, 3079–3087.

McCanne, S., & Jacobson, V. (1995). vic: A flexible framework for packet video. *Proceedings of the Third ACM International Conference on Multimedia, ACMMM'95* (pp. 511–522), San Francisco, CA, November, 1995.

Muijs, R., & Kirenko, I. (2005). A no-reference blocking artifact. measure for adaptive video processing. *Proceedings of the 13th European Singal Processing Conference, EUSIPCO'05*, Antalya, Turkey, September, 2005.

Nair, V., & Clark, J. J. (2002). Automated visual surveillance using hidden markov models. *Proceedings of the 15th International Conference on Vision Interface, VI'02* (pp. 88–92), Calgary, May, 2002.

Niu, W., Jiao, L., Han, D., & Wang, Y. (2003). Real-time multiperson tracking in video surveillance. *Proceedings of the Fourth International Conference on Information, Communications and Signal Processing and Fourth IEEE Pacific-Rim Conference On Multimedia, ICICS-PCM'03*, Vol. 2 (pp. 1144–1148), Singapore, December, 2003.

Ooi, W. T., Pletcher, P., & Rowe, L. (2004). Indiva: A middleware for managing distributed media environment. *Proceedings of the SPIE Conference on Multimedia Computing and Networking, MMCN'04* (pp. 211–224), Santa Clara, CA, jan, 2004.

Ortega, A., & Ramchandran, K. (1998). Rate-distortion techniques in image and video compression. *IEEE Signal Processing Magazine*, *15*(6), November, 1998, 23–50.

Papageorgiou, C., Oren, M., & Poggio, T. (1998). A general framework for object detection. *Proceedings of the Sixth International Conference on Computer Vision, ICCV'98* (pp. 555–562), Bombay, India, January, 1998.

Rangaswami, R., Dimitrijevi, Z., Kakligian, K., Chang, E., & Wang, Y. (2004). The SfinX video surveillance system. *Proceedings of the IEEE International Conference on Multimedia and Expo, ICME'04*, Taipei, Taiwan, June, 2004.

Rouse, D., & Hemami, S. S. (2008a). Analyzing the role of visual structure in the recognition of natural image content with multi-scale ssim. *Proceedings of SPIE Conference on Human Vision and Electronic Imaging, SPIE'08*, Vol. 6806, San Jose, CA, USA, January, 2008.

Rouse, D., & Hemami, S. S. (2008b). How to use and misuse image assessment algorithms. *Proceedings of Western New York Image Processing Workshop, WNYIP'08*, Rochester, NY, USA, September, 2008.

Rouse, D., Pepion, R., Hemami, S. S., & Callet, P. L. (2009). Image utility assessment and a relationship with image quality assessment. *Proceedings of SPIE Conference on Human Vision and Electronic Imaging, SPIE'09*, Vol. 7240, San Jose, CA, USA, January, 2009.

Rowley, H., Baluja, S., & Kanade, T. (1998). Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*, January, 1998, 23–38.

Sanchez, V., Basu, A., & Mandal, M. (2004). Prioritized region of interest coding in JPEG2000. *Proceedings of the 17th International Conference on Pattern Recognition, ICPR'04*, Vol. 2 (pp. 799–802), Melbourne, Australia, August, 2004.

Schumeyer, R., Heredia, E. A., & Barner, K. E. (1997). Region of interest priority coding for sign language videoconferencing. *Proceedings of the First IEEE Workshop on Multimedia Signal Processing, MMSP'05* (pp. 531–536), Princeton, NJ, June, 1997.

Shannon, C. (1948). A mathematical theory of communication. *Bell System Technical Journal*, *27*, July, 1948, 379–423.

Sung, K.-K., & Poggio, T. (1998). Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(1), January, 1998, 39–51.

Viola, P., & Jones, M. (2001). Robust real-time face detection. *Proceedings of the ICCV 2001 Workshop on Statistical and Computation Theories of Vision, ICCV'01*, Vol. 2 (p. 747), Vancouver, Canada, July, 2001.

Viola, P., & Jones, M. (2004). Robust real-time face detection. *International Journal of Computer Vision*, *57*(2), April, 2004, 137–154.

Wang, Y., Kim, J., & Chang, S.-F. (2003). Content-adaptive utility-based video adaptation. *Proceedings of the IEEE International Conference on Image Processing, ICIP'03*, Vol. 1 (pp. 189–192), Barcelona, Catalonia, Spain, September, 2003.

Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, *13*(4), January, 2004, 600–612.

Welsh, G., & Bishop, G. (2001). An introduction to the kalman filter. *Proceedings of SIGGRAPH 2001*, Vol. Course 8, Los Angeles, CA, USA, August, 2001.

Wu, G., Wu, Y., Jiao, L., Wang, Y., & Chang, E. (2003a). Multi-camera spatio-temporal fusion and biased sequence-data learning for security surveillance. *Proceedings of the 11th ACM International Conference on Multimedia, ACMMM'03* (pp. 528–538), Berkeley, CA, USA, November, 2003.

Wu, Y., Jiao, L., Wu, G., Chang, E., & Wang, Y. (2003b). Invariant feature extraction and biased statistical inference for video surveillance. *Proceedings of the IEEE International Conference on Advanced Video and Signal Based Surveillance, AVSS'03* (pp. 284–289), Miami, FL, July, 2003.

Yuan, X., Sun, Z., Varol, Y., & Bebis, G. (2003). A distributed visual surveillance system. *Proceedings of the IEEE International Conference on Advanced Video and Signal Based Surveillance, AVSS'03* (pp. 199–204), Miami, FL, July, 2003.