Cooperative Visual Monitoring in Energy-Constrained Wireless Sensor Networks

THÈSE N^{O} 5608 (2013)

PRÉSENTÉE LE 18 JANVIER 2013

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

LABORATOIRE DE COMMUNICATIONS AUDIOVISUELLES

PROGRAMME DOCTORAL EN INFORMATIQUE, COMMUNICATIONS ET INFORMATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Zichong CHEN

acceptée sur proposition du jury:

Prof. P. Thiran, président du jury
Prof. M. Vetterli, Dr G. Barrenetxea, directeurs de thèse
Prof. Y. Liu, rapporteur
Prof. A. Martinoli, rapporteur
Prof. L. Thiele, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2013

Abstract

Thanks to the decreasing manufacturing cost, size, and energy consumption of CMOS image sensors, wireless cameras are becoming ubiquitous. Visual monitoring applications are numerous, ranging from security to environmental monitoring. Among these, we are particularly interested in natural hazards, where a visual feedback from the monitored area can be used to detect potentially critical situations, such as avalanches, rock slides or fires. One of the key aspects of a wireless camera system design is the reduction of its energy consumption to extend its lifetime without replacing the batteries of nodes. This is particularly challenging for visual monitoring, since cameras generate much higher data rate than traditional sensors. Another challenge of wireless visual monitoring is the vulnerability of the system. Because deployments are often located in remote and uncontrolled environments (e.g., mountain regions) where nodes are exposed to environmental hazards and extreme weather conditions, the reliability of a single camera system would be seriously compromised.

A multi-camera wireless monitoring system provides a solution to these two challenges. First, multiple cameras can collaboratively share the monitoring tasks and hence reduce energy consumption on each camera. Second, physically distributed cameras also share environmental risks. In case of camera failures, the remaining cameras can dynamically reconfigure and continue to monitor the scene of interest. In this thesis we address the question of how cameras can collaborate to efficiently monitor the environment and minimize the energy consumption.

More precisely, we study two fundamental problems, namely, sampling and coding. The first part of the thesis investigates a cooperative sampling scheme that equally distributes the sampling task among all the cameras. We prove that the optimal spreading of cameras' sampling operations is a uniform configuration. To allow the cameras to adaptively establish the correct sampling configuration, we develop a distributed, self-organizing algorithm, that exploits the broadcast nature of wireless communications. We show the effectiveness of this algorithm in fully connected networks, partially connected networks, and under overhearing losses.

The second part of the thesis investigates the cooperative coding methods for static images, videos, and event detection applications. First, for static images captured by distributed cameras, we propose a distributed successive refinement coding scheme. The theoretical study of the Gaussian case proves that this scheme is successively refinable on the rate-distortion limit of distributed coding. Practical experimental results on multi-view images show that it operates within 3dB to the distributed coding bound. Secondly, for videos captured by distributed cameras, we propose and evaluate various cooperative coding schemes for achieving efficient video compression.

Experimental results on a two-camera system show that the per-camera consumption can be reduced by up to 50% with respect to a single-camera system. Finally, we investigate beyond traditional video coding and exploit the computation power of smart cameras. For event detection applications, we propose an energy-efficient coding algorithm that transmits only event-specific information and achieves much lower communication rates than conventional video compression. Experiments on a real-world multi-view dataset show an optimal per-camera energy scaling that is inversely proportional to the number of cameras, without introducing distributed losses.

Keywords: wireless camera network, environmental monitoring, sampling theory, successive refinement, distributed source coding, multi-view images, video compression, energy efficiency, distributed smart cameras.

Résumé

Grâce à la diminution de la taille, des coûts de fabrication et de la consommation des capteurs d'image CMOS, les cameras sans-fil deviennent omniprésentes. Les applications de surveillance vidéo sont nombreuses, allant de la sécurité à la surveillance de l'environnement. Le cas des dangers naturels est particulièrement intéressant, puisqu' un retour visuel de la zone surveillée peut être utilisé afin de détecter de potentielles situations critiques, telles qu'avalanches, glissement de terrain ou feux. Lors du design d'un système de caméras sans-fils, un des aspects prépondérant est la réduction de la consommation d'énergie afin d'augmenter son autonomie lors d'un fonctionnement sur batteries. Cependant, le flux de données important généré par de tels systèmes rend difficile la réduction de leur consommation. Un autre aspect important est la vulnérabilité d'un tel système. Etant souvent déployés dans des zones isolées et non-contrôlées (p.ex. zones montagneuses), ils sont exposés à des conditions météorologiques et environnementales extrêmes. Ainsi, le déploiement d'un système unique n'est clairement pas garant d'une grande fiabilité dans le temps.

Un système de surveillance multi-caméras permet d'apporter une solution à ces deux problématiques. Premièrement, des caméras multiples peuvent se partager les diverses tâches de surveillance, et ainsi réduire leur consommation. Deuxièmement, distribuer des caméras sur une zone à surveiller permet de réduire les risques dûs aux facteurs environnementaux : en cas de défaillance d'une ou de plusieurs caméras, les caméras restantes peuvent dynamiquement se reconfigurer et continuer à surveiller la zone d'intérêt. Dans cette thèse, nous nous intéressons à savoir comment des caméras peuvent collaborer afin de surveiller de manière efficace leur environnement tout en minimisant leur consommation d'énergie.

Plus précisément, nous étudions deux problèmes fondamentaux, à savoir l'échantillonnage et l'encodage des données. La première partie de la thèse étudie un système d'échantillonnage coopératif qui distribue les tâches d'échantillonnage parmi les caméras. Afin de permettre aux caméras d'établir de manière adaptative la configuration d'échantillonnage correcte, nous développons un algorithme distribué et auto-adaptatif qui exploite la diffusion des signaux propre aux communications sans-fil. Nous prouvons que la répartition optimale des opérations d'échantillonnage parmi les caméras suit une configuration uniforme. Nous démontrerons aussi l'efficacité de cet algorithme dans des réseaux totalement interconnectés, partiellement interconnectés, ainsi que lors de pertes de communications.

La seconde partie de cette thèse étudie les méthodologies d'encodage coopératif pour des images statiques, des vidéos, ainsi que dans des applications de détection d'évènements. Dans un premier temps, pour des images statiques capturées par des

caméras distribuées, nous proposons un système d'encodage distribué à affinage successif. L'étude théorique de cas Gaussiens démontre que ce système s'affine de manière successive dans les limites débit-distorsion de l'encodage distribué. Les résultats pratiques obtenus sur des images multi-vues indiquent que le système proposé a une performance débit-distorsion dans la zone des 3dB de l'encodage distribué. Dans un deuxième temps, pour des vidéos capturées par des caméras distribuées, nous proposons et évaluons diverses méthodes d'encodage coopératif permettant une compression vidéo efficace. Les résultats expérimentaux obtenus avec un système bi-caméra ont montré que la consommation par caméra peut être réduite de moitié par rapport à un système composé d'une seule caméra. Finalement, nous nous intéressons au-delà de codage vidéo traditionnels et d'exploiter la puissance de calcul des caméras intelligentes. Pour la détection d'évènements, nous proposons un algorithme d'encodage qui ne transmet que les informations relatives aux évènements spécifiques. Cet algorithme présente une bien meilleure efficacité énergétique puisque son débit de transmission de données est bien plus faible que ceux des systèmes de compression vidéo conventionnels. Les expériences réalisées sur des échantillons d'images multi-vues du monde réel montrent que l'économie énergétique ainsi obtenue est proportionnelle au nombre de caméras, sans pour autant en distribuer les pertes.

Mots clés: réseau de caméras sans-fils, surveillance de l'environnement, théorie d'échantillonnage, affinage successif, encodage distribué, images multi-vues, compression vidéo, rendement énergétique, réseau de caméras intelligentes

Acknowledgements

If one think about a person's life trajectory as a hiking trace across the Swiss Alps, then this thesis should be the highest summit I have reached so far. It is inappropriate to repeat Isaac Newton's famous quote here, but without many people's support it will be impossible to finish this thesis.

As defined by Confucius, the relation between a teacher and a student, is equivalent to the relation between a father and his son, or a king and his citizens. So first of all, I would like to thank my two advisors, Prof. Martin Vetterli and Dr. Guillermo Barrenetxea. Needless to say, Martin is a passionate researcher, which is almost the first impression as soon as one gets to have the "chat" with him. I first met Martin in front of the main building at Tsinghua University while he was visiting Feng's lab (which I discovered very rencently). He is actually the only professor who responded to my annoying email asking for PhD positions, and surprisingly he was right in China to meet me! Without this lucky meeting, I'm probably in another place, so I would like to thank him for offering me this great opportunity. Guillermo is really a nice, patient mentor, and also a great friend. His pragmatic way of research is so impressive and solid, and I really learnt a lot from him – especially the critical thinking and the systematic approach. Plus, I owe a lot to him for proofreading my exotic English writing for dozens of times;) I really learnt a lot each time! Through the four-year PhD study under the influence of Martin and Guillermo, now I know the standard of world-class research, which will be definitely a treasure in future.

I would also take this opportunity to express my gratitude to Professors Yunhao Liu, Alcherio Martinoli, Lothar Thiele, and Patrick Thiran for being my thesis committees. Without their big efforts in distant teleconferencing, insightful suggestions, friendly reminders, and jokes in the after-exam dinner, my thesis will not be as good as the current shape.

I would like to express my gratitude to the financial supports from the Swiss National Science Foundation (SNSF) and EPFL. I owe many thanks to our lab's secretary, Jacqueline, for all the helps and precise organization of the lab. A special thank to Simon, who has left a few years ago but really helped me a lot on the computer issues. I would also like to thank all my LCAV colleagues for their help and support: Mihailo, Pedro, Francisco, Paolo, Andrea, Ivana, Jay, Olivier, François, Yue, Farid, Yann, Ivan, Mitra, Marta, Reza, Andreas, Zhou, and RK. Without them, there will be no joy in the hiking day, ski day, and Christmas dinner. They are also sources of research inspirations, as the weekly TAM is usually a place to learn how great our lab is! Special thanks to Albrecht, Feng, Juri and Runwei, for our pleasant and inspiring research collaborations. Thanks to Prof. Sabine Süsstrunk and Prof.

Michael Gastpar, for their kind comments on my papers. I am also thankful to my office-mate Constantin and Runwei, for all the interesting conversations and research discussions, which really help me to define my thinking of future.

I am very grateful to my mentor and friend Fengjun Lv in California. He is a kind and wise man who is blessed, and really offered me a five-star experience during the whole internship. Special thanks to Ton Kalker, Jiong Zhou and Liang Zhou, for all the happy moments and dinners in the summer. I am also thankful to my college friends who spent precious but really great time with me in the U.S.: Jie Xu, Qiang Chen, Siming Song, Ziyuan Wang, Weiguang Si, Weiyu Zhang, and Jiangping Wang.

I would also like to thank all my friends that I met in Switzerland: Hu Xu, Feng Yang, Runwei Zhang, Jiaqing Du, Li Pu, Xiaolu Sun, Li Chen, Sanhao Ji, Séverine Moret, Sébastien Dey and many more I cannot list here. It is for our great friendship and joyful travels around the world. Last but not least, I am really grateful to my parents Peiju and Weixin, for their unconditional love and support during all these years. A very special thank you to my beloved Weijia, who always gives me the greatest support and accompanies me during all the great and tough time. My life is always colorful with her.

Notations

Cooperative sampling for distributed cameras

n	Number of cameras
f	Sampling frequency of each camera
T	Sampling interval of each camera, equals $1/f$
f_s	Sampling frequency of the system, equals n/T
T_s	Sampling interval of the system, equals $1/f_s = T/n$
δ_i	Offset between the sampling grid of ith and $(i+1)$th cameras
Δ	Duration of a random event
P_d	Event detection probability of the system
$t_i^{(k)}$	Time stamp of the kth transmission of the ith camera
$\mathbf{t}^{(k)}$	Time stamp vector of all cameras' kth transmission

Cooperative coding for distributed image sources

X, Y	Two distributed, correlated sources
C_i	Codeword sent at ith stage
R_i	Rate of the codeword C_i
X_i, Y_i	Reconstruction of X, Y at ith stage
D_{X_i}, D_{Y_i}	Distortion of X_i, Y_i at ith stage
M	Total number of stages of DiSAC2

$p(\mathscr{X})$	Probability density function of a continuous random variable $\mathscr{X}$
$\mathbb{E}(\mathscr{X})$	Expectation of a continuous random variable $\mathscr{X}$
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$
ρ	Correlation between two bivariate Gaussian sources
$\mathcal{G}_{\mu,\sigma^2}(x)$	Gaussian function $\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$
S_1, S_2, S_b	Modulated signals in uncoded transmission
$\alpha_1, \alpha_2, \alpha_b$	Modulation ratios in uncoded transmission
Z_1, Z_2, Z_b	Channel noises in uncoded transmission
$\hat{S}_1, \hat{S}_2, \hat{S}_b$	Received modulated signals in uncoded transmission
β_{ij}	Decoding coefficients in uncoded transmission

Computation versus Communication

m	Block length in a block-based coding algorithm
ΔR	Penalty rate between the actual rate and the theoretical rate
C_E	Complexity of the encoder in source coding
R_p	Entropy per pixel
e_t	Energy per bit for transmission
e_h	Energy per pixel for Huffman coding when $m = 1$
E_p	Global energy per pixel
$R_{\mathcal{X}}(D)$	Rate-distortion function of the source $\mathcal{X}$

Contents

Chapter 1

Introduction

This chapter describes the scope and motivation of this thesis. We first introduce the concept of wireless visual monitoring as a new sensing modality enabled by the development of low-cost low-power image sensors. Then, we discuss the two main challenges in visual monitoring, namely, robustness and energy consumption. Finally, we briefly present our solutions and key contributions, and give a general layout of this thesis.

1.1 To Images and Beyond

The concept of *Smart Dust* was originally proposed by Kris Pister, Joe Kahn, and Bernhard Boser, in a research proposal [1]. The idea is to build wireless sensor nodes of *small size*, *low manufacturing cost* and *low power consumption*. The nodes, once deployed in the field, will autonomously build up an ad-hoc network and report their sensor readings to the end-user in real-time. Fifteen years have passed since this proposal, and wireless sensor networks (WSNs) have now found wide-spread applications in many industrial and consumer applications, such as environmental monitoring, structural monitoring, machine health monitoring, or smart home monitoring.

WSNs are widely used in environmental applications (e.g., Sensorscope [2] developed at EPFL), as they provide significant advantages over the high-cost conventional stations: 1) highly versatile, 2) low-cost, 3) small-size, and 4) dense spatial coverage. In particular, thanks to the development of low-cost and low-power sensors, multimodal wireless monitoring has become a common feature of most of the state-of-the-art WSNs: Multiple sensors are attached to a single node that records and transmits several sources of information.

Most of the traditional sensors used in WSNs are similar to the NO_2 gas sensor illustrated in Figure 1.1. They measure certain physical quantities, such as temperature, humidity, or gas concentration. The measured data is a low-rate one-dimensional time series $I(t)$.

The availability of low-cost low-power image sensors is shifting the paradigm of sensor networks from low-rate data to large amounts of data (Figure 1.1). We can clearly see this from the comparison of various sensors commonly used in current WSNs (Table 1.1). In terms of the signal dimension, an image sensor is very differ-

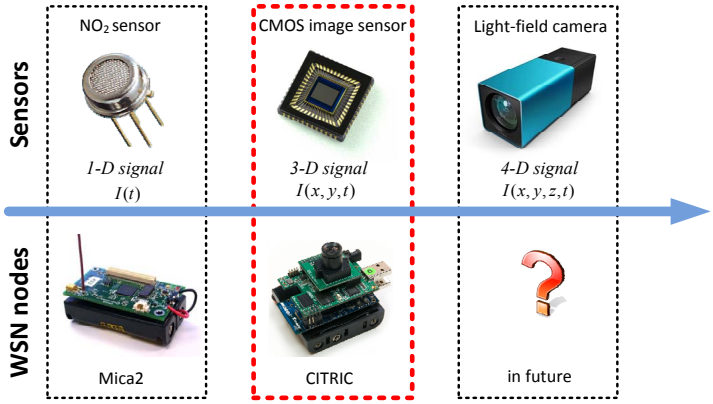


Figure 1.1: *The physical quantity measured by sensors evolves from traditional 1-D signal to 3-D videos and even 4-D light fields. Meanwhile, the WSN node is also evolving.*

Table 1.1: *Comparison of typical sensors*

Sensor type (company)	Temperature (Davis)	NO_2 gas (e2V)	CO gas (e2V)	2-MP CMOS image sensor (OmniVision)
Price [USD]	18	6	9	6
Power [mW]	5	32	80	150
Data rate per sample [byte]	2	2	2	50000

ent from traditional sensors, because it captures a three-dimensional signal $I(x, y, t)$, where x, y are the coordinates in the image plane.

The developments in sensing technologies have been rapidly adopted in wireless sensor nodes. Recently, nodes with image sensor such as Cyclops [3] and CITRIC [4] have became available. We can expect that in the near future, the 4-D light-field sensors [5] will also become ubiquitous in small devices such as wireless sensor nodes.

1.2 Wireless Visual Monitoring

Wireless camera nodes have made wireless visual monitoring applications possible. One or multiple wireless cameras are deployed to jointly (or cooperatively) monitor an area of interest and transmit the captured videos to the end-user through wireless communications. This concept fits many application scenarios, such as *environmental monitoring* [6] and *surveillance* [7]. For instance, Figure 1.2a shows our recent deployment of a wireless camera on the Swiss Alps. A visual feedback from the monitored area can prove indispensable to detect potentially critical environmental situations (avalanches, rock slides, or fires) or to better interpret the telemetry data (snow vs. rain, clouds vs. sun).

The design of a visual monitoring system has two main requirements: 1) robustness and 2) low energy consumption. First, as shown in Figure 1.2a, the surrounding environment of wireless camera imposes great risks on the camera itself. Unpredictable

(a) (b) (c)

Figure 1.2: *Wireless visual monitoring in the wild. (a) The Val Ferret deployment: an autonomous wireless camera is deployed in a valley on the border between Italy and Switzerland. The image with the arrow illustrates the location of the deployment and an image captured by the camera. (b,c) A SwissEx [8] wireless sensor station at Davos was destroyed by an avalanche, before and after snow melt.*

Table 1.2: *Wireless cameras in indoor and outdoor scenarios*

Scenario	Radio range [m]	Energy per frame [J]	Resolution	Node size [cm^2]	Irradiance [J/cm^2/day]
Indoor (Cyclops)	75	0.063	128×128	49	9
Outdoor (Sensorcam)	20000	22	640×480	154	1500

weather conditions such as strong winds, avalanches, or rock slides can damage and destroy a camera. For example, Figure 1.2b and Figure 1.2c show a wireless sensor station (one of our recent deployment) destroyed by an avalanche, before and after snow melt. Similarly, in surveillance applications, the intruders can destroy monitoring cameras on purpose. Therefore, the robustness of the visual monitoring system is crucial in practice.

Second, as the size of a wireless sensor node decreases (recalling *Smart Dust*), the energy constraint on the sensor node is also critical for achieving autonomous and continuous wireless monitoring. Two types of energy sources are common in sensor networks, namely, battery and energy-harvesting photovoltaic (PV) cell. The energy provided by both sources is proportional to their sizes: the energy stored in a battery is the volume of battery times the energy density of battery, and the energy generated by PV cells is proportional to the cell surface. Thus, both batteries and PV cells must shrink in size to enable the minimization of sensor nodes.

To show the energy constraints imposed on the wireless cameras in both indoor and outdoor scenarios, we choose two camera platforms, namely, Cyclops [3] for short-range communication, and Sensorcam [9] for long-range communication. The average irradiance [10] that PV cells can collect in indoor/outdoor scenarios is given in Table 1.2. Assuming the conversion efficiency of PV cell is 20%[1], Figure 1.3 shows the

[1] A typical rate for commercially available crystalline PV cells [11].

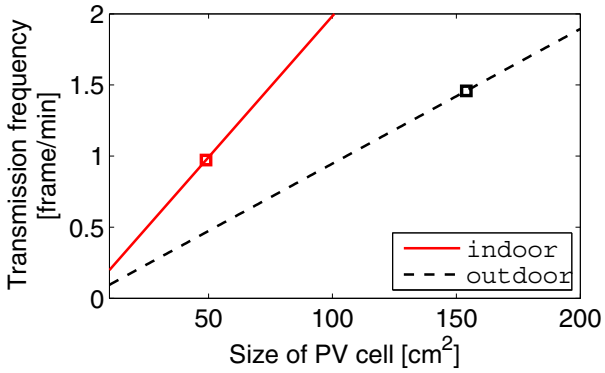


Figure 1.3: *Achievable transmission frequency of wireless cameras in indoor and outdoor scenarios, using traditional image compression technology. The square indicates the point where the size of photovoltaic (PV) cell matches the size of camera node.*

achievable transmission frequency of wireless cameras in indoor and outdoor scenarios, using the Motion JPEG compression method[2]. Clearly, with the energy permitted by the size of camera nodes, the traditional image compression technology only allows for transmitting less than two frames every minute.

Note that the calculation above does not take into account the overheads of sensor nodes, such as energy storage leakages or costs of multiple access protocol (MAC). Hence, in current wireless monitoring cameras, the achievable sampling frequency is usually less than one frame per minute. For instance, in the *Val Ferret* deployment (Figure 1.2a), the camera currently can only transmit one image every half an hour, using the standard JPEG image compression. Since the size of sensor nodes is expected to further decrease in the near future, the severe energy constraints imposed by this size make it necessary to accordingly reduce the *per-camera energy cost*, by using better computation and communication strategies.

A multi-camera wireless monitoring system provides a solution to these two requirements. First, multiple cameras can collaboratively share the monitoring tasks and hence reduce energy consumption on each camera. Second, physically distributed cameras also share environmental risks. In case of camera failures, the remaining cameras can dynamically reconfigure and continue to monitor the scene of interest. Figure 1.4 shows a typical setup where multiple cameras are placed in different locations to observe a common scene of interest, and communicate with a base station (BS). In case of camera failures, as long as one still survives, we will not lose the visual access to the scene of interest. Therefore, the system robustness increases with the number of cameras deployed. However, since cameras observe a common scene of interest, multi-camera system produces image sequences that are highly redundant. This redundancy translates into unnecessary energy consumptions that could potentially decrease the lifetime of the monitoring system.

[2]Cyclops is attached to a Mica2 node for communication, and uses an optimized JPEG compression [12]. Refer to Figure 5.10 in this thesis for the energy cost of Sensorcam.

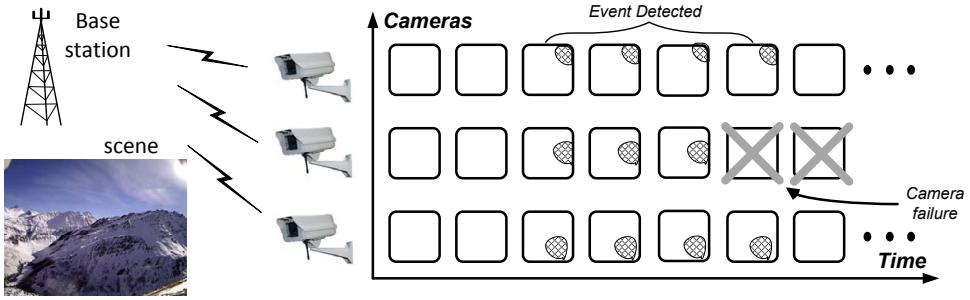


Figure 1.4: *Multiple wireless cameras are deployed to monitor a common scene of interest and communicate with a base station (BS). The captured images span over different cameras and time. The system is still able to detect events of interest even if one or multiple cameras fail.*

1.3 Thesis Outline

This thesis focuses on reducing sampling and communication costs of multi-camera systems, by coordinating all the cameras and distributing the monitoring task. We will investigate two fundamental problems for achieving an efficient multi-camera monitoring system:

1. *Sampling Problem*: how to coordinate multiple cameras for an optimal event-detection probability?

2. *Coding Problem*: how to exploit signal statistics locally and across different cameras for reducing communication costs?

In **Chapter 2**, we propose a cooperative sampling framework to address the *Sampling Problem*. In a n-camera system, each camera samples at frequency f/n, and the sampling time of cameras is arranged in an interleaved manner, thus imitating a single camera sampling at frequency f. In this way, we distribute the risks of physical hazard and the energy burden of wireless transmission into n independent cameras, while the overall event detection capability is kept the same. We further show that a uniform sampling configuration is optimal in terms of event detection probability. To address the unknown clock offsets and dynamically changing network topologies, we develop a distributed desynchronization algorithm to allow the camera network to build up the interleaved sampling configuration autonomously.

To further improve energy efficiency, the *Coding Problem* seeks to compress image sequences captured by multiple cameras over time (Figure 1.4). This requires joint processing across cameras, which is infeasible because there is no direct communication between the cameras. However, as wireless systems often use omnidirectional antennas, the cameras within the transmission range of the emitting camera can overhear the transmitted messages. By using this passive broadcast link, we can potentially encode information of distributed cameras for achieving *cooperative coding*. The second part of the thesis investigates the *cooperative coding* methods for static images, videos, and event detection applications.

In **Chapter 3**, we investigate the *cooperative coding* problem of multi-view images. Through a ping-pong coordination between two cameras, we show that this practical coding method allows the two-encoder successive refinement coding to achieve the theoretical distributed coding limit. Although the image source in this problem is not strictly within the cooperative sampling framework (which captures videos), the proposed successive coding procedure imitates the interleaved sampling configuration. The theoretical study of the Gaussian case proves that this scheme is successively refinable on the rate-distortion limit of distributed coding. Practical experimental results on multi-view images show that it operates within 3dB to the distributed coding bound.

In **Chapter 4**, we investigate the *cooperative coding* problem of multi-view videos. To compress image sequences from n disjoint cameras, we propose three video coding methods, namely, distributed, independent, and joint coding schemes. The experimental results on a two-camera system show that independent coding and joint coding perform substantially better. The comparison between a two-camera system and a single-camera system shows 30%-50% per-camera energy consumption reduction.

In **Chapter 5**, we extend the discussion to event detection applications. We first propose a novel event-driven video coding (EVC) scheme. The motivation is that to deliver an image sequence from a camera, conventional compression methods ignore the "meaning" of video content. In monitoring applications, the information that a user retrieves from the wireless camera is simpler than the raw video itself, e.g., "who enters the monitored area?", or "how is the weather?" By using the computation power of cameras, we can extract more meaningful information from raw images and achieve lower communication rates. The experimental results show that our proposed coding method performs substantially better than conventional video communication schemes such as H.264. Then, we combine EVC into the cooperative sampling framework, and develop a distributed multi-camera EVC scheme. The simulations on a large-scale multi-camera dataset show that the per-camera energy cost scales inversely proportional to the number of cameras.

We conclude in **Chapter 6** with a summary of the thesis and directions for future research.

Chapter 2

Cooperative Sampling Framework for Distributed Cameras

2.1 Introduction

In this chapter, we address the sampling problem for visual monitoring task and propose a general cooperative sampling framework for distributed cameras. It serves as a foundation for the whole thesis. The rest of the chapters of this thesis address the coding and communication problems, which will be based on this sampling framework.

The general idea of exploiting the sampling capability of multiple cameras is inspired by the space-time sound field sampling theory [13], in which the optimal sampling grid is achieved by a tight packing of sound field's spectrum replicas. We translate this idea to the camera network. Our proposed sampling framework seeks to find an optimal sampling strategy that maximizes event detection probability. In a n-camera network, each camera samples at frequency f, and the sampling time of cameras is arranged in an interleaved manner, thus imitating a single camera sampling at a higher frequency $f_s = nf$. In other words, if we fix f_s, the overall sampling operations are essentially distributed into n independent cameras $f = f_s/n$, while the overall event detection capability is kept the same. Section 2.2 discusses the interleaved sampling framework in details, and shows that a uniform interleaved sampling configuration is optimal in terms of event detection probability.

This optimal sampling configuration requires however a perfect synchronization between the cameras. To address unknown clock offsets and dynamically changing network topologies, we present a distributed desynchronization algorithm to allow the camera network to build up the interleaved sampling configuration autonomously. By exploiting the broadcast nature of wireless communication, the algorithm dynamically adapts to new camera arrivals or camera malfunctions. We discuss the communication models of the considered camera network in Section 2.3. In Section 2.4, we study the desynchronization algorithm based on the broadcast graph that models the overhearing connections among cameras. We first introduce the desynchronization

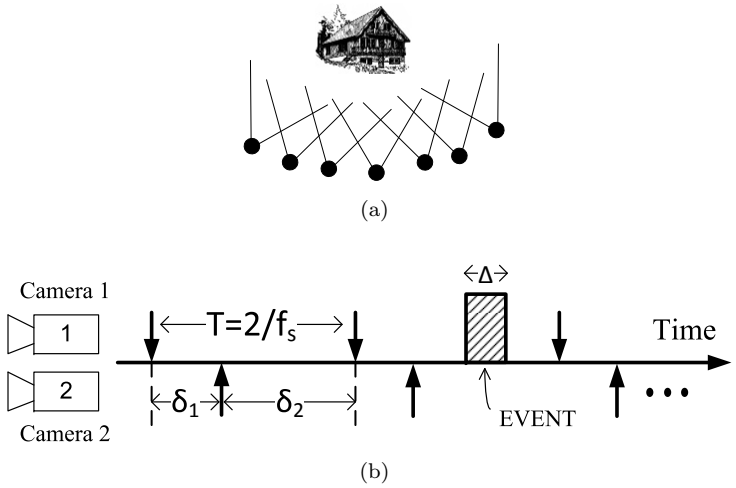


Figure 2.1: *A multi-camera monitoring network: (a) By monitoring a common scene of interest, multiple cameras, represented by the dots, share the physical risk and energy burden of wireless transmissions. (b) Interleaved sampling setup of two cameras that attempts to capture a random event. The arrows represent the sampling time of each camera.*

algorithm and its convergence conditions. Then, we extend it into real-world scenarios where not every pair of nodes can overhear each other. We also address the potential overhearing loss, and discuss about good network deployments for better performance.

2.2 Optimal Sampling Using Multiple Cameras

Figure 2.1a shows the multi-camera wireless monitoring system we consider. To detect a random event, each camera is programmed to periodically capture the scene and transmit the data to an end-user through wireless communications. Given several cameras monitoring a common scene of interest, we investigate the coordination strategy and its potential benefits.

In order to share the energy burden among all cameras, we keep the overall number of images captured by the system fixed. We denote by f_s the sampling frequency of the system; f_s measures the overall number of images captured by all the cameras in a unit time. Similarly, we denote by f the sampling frequency of each camera. For a n-camera monitoring network, the sampling frequency of each camera is

$$f = f_s/n. \tag{2.1}$$

This implies two possible benefits as we increase the number of cameras n:

1. If we fix f_s, the sampling frequency of each camera f is reduced proportionally. Thus, the consumption per camera is reduced.

2. If we fix f, the sampling frequency of the system f_s is increased proportionally. Thus, the probability of catching an event is increased.

We study now the optimal sampling strategy that maximizes the probability of catching an event. To analyze this setup mathematically, we begin with a two-camera setup as shown in Figure 2.1b. The sampling frequency of each camera is fixed at $f_s/2$, and δ_1 is the interleaved offset between the sampling grids of the two cameras. If we assume that a particular event happens randomly (uniform distribution) with a fixed duration Δ, the probability P_d that we capture the event is related to the choice of δ_1. Not surprisingly, a uniform setup, that is, $\delta_1 = 1/f_s$, is shown to be best. In the following, we prove that the optimal sampling strategy is a uniform interleaved setup for any number of cameras.

Theorem 2.1 (interleaved sampling)**.** *In a n-camera monitoring network where each camera samples at a frequency f_s/n, the optimal cooperative strategy that maximizes the probability to capture a uniformly distributed random event, is an interleaved setup with uniform spacing of $1/f_s$.*

Proof. Denote by $\delta_1, \delta_2, \cdots, \delta_n$ the interleaved offsets between the sampling grid of adjacent cameras (the case $n = 2$ is shown in Figure 2.1b). They are constrained by $\sum_{i=1}^{n} \delta_i = n/f_s$.

For $n = 2$, we can assume $\delta_1 < 1/f_s$ because of symmetry. As the event occurs with a uniform distribution and has a duration of Δ, the probability P_d that we catch the event can be expressed as

$$P_d = \begin{cases} f_s\Delta, & \Delta < \delta_1 \\ f_s(\Delta + \delta_1)/2, & \delta_1 < \Delta < 2/f_s - \delta_1 \\ 1, & \Delta > 2/f_s - \delta_1 \end{cases}.$$

Thus, the optimal choice of δ_1 that maximizes the detection probability is

$$\arg\max_{\delta_1} P_d = \begin{cases} \delta_1 > \Delta, & \Delta < 1/f_s \\ \delta_1 < \Delta, & \Delta > 1/f_s \end{cases},$$

from which we know that $\delta_1 = 1/f_s$ is the universal choice to maximize P_d. Therefore, the optimal sampling configuration for two cameras is a uniform interleaved sampling where each camera samples at a frequency of $f_s/2$, and the interleaved offset between the two sampling grids is $1/f_s$.

We can generalize this to the case when $n > 2$: If the random event occurs at a time when the camera k and $k + 1$ can capture it, the optimal strategy is to choose $\delta_k = \delta_{k+1}$, according to the above argument for the two-camera case.

By doing such a partition from δ_1 to δ_{n-1}, we can obtain $\delta_1 = \delta_2$, $\delta_2 = \delta_3$, $\cdots$, and $\delta_{n-1} = \delta_n$. Thus the uniform interleaved setup $\delta_1 = \delta_2 = \cdots = \delta_n$ is the optimal sampling strategy. □

2.3 Communication Models

As we mentioned before, the core motivation of a multi-camera system that monitors a common scene of interest is to increase system robustness. Besides a physically

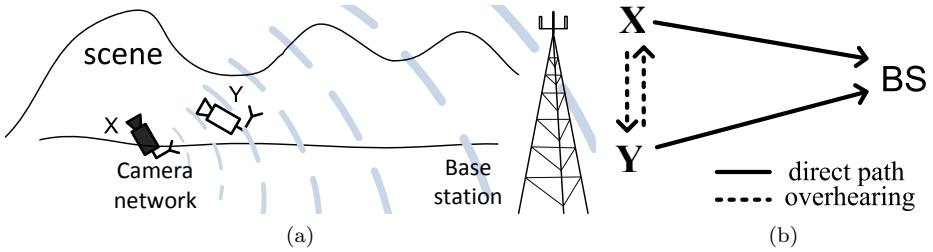


Figure 2.2: *Two cameras X and Y are deployed to monitor a scenery, and communicate independently with the base station (BS). When the active camera X is transmitting an image to the BS, the other camera Y can overhear the same information. (a) Illustration of the setup. (b) Abstract model.*

separated deployment, the communication scheme also plays an important role in the system robustness. We choose to use a single-hop star topology, where n cameras communicate independently with a base station (BS) through long-range radios (see Figure 2.3a). The main reasons for this choice over multi-hop ad-hoc networks are:

1. Every camera is able to communicate to the BS independently. The whole system will not break down when a few nodes fail to work.

2. All cameras share equally the communication cost. There is no particular node becoming the bottleneck of lifetime in the system.

3. The system can be easily setup if we use existing wireless infrastructures such as WiFi or GPRS.

We further assume that inter-camera communication is not implemented as it requires extra hardware, but there are potentially existing *passive* links [14]. As wireless systems often use omnidirectional antennas, the cameras within the transmission range of the emitting camera can actually overhear the transmitted messages. Figure 2.2a shows an example setup of two cameras: Cameras are deployed to monitor an overlapping scenery, and communicate independently with the base station (BS) to deliver the captured images. Direct communication between two cameras is not present, but each camera overhears the information transmitted by the other one. Note that there is no radio interference, because cameras transmit data in an interleaved manner determined by the sampling schedule described in Section 2.2.

When the scale of the camera network increases, we need to consider factors that might impact the broadcast links:

1. In the case of directional antennas, or in the presence of obstacles, overhearing may become infeasible, even if two cameras are close-by.

2. The transmission range of each camera does not necessarily cover all other cameras in the network. Usually, nearby cameras have good overhearing quality, and the quality decreases with distance.

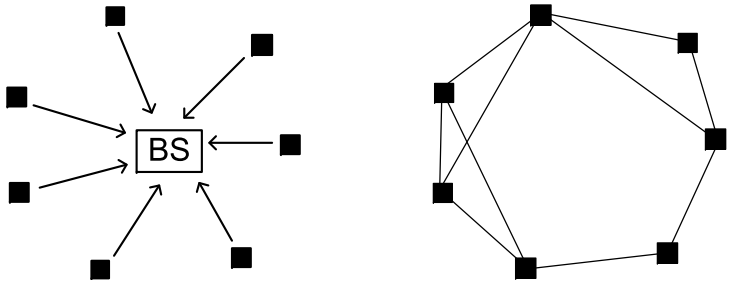


Figure 2.3: *Communication models of a multi-camera system. The solid square denotes a camera. (a) Communication topology: each camera directly transmits data to a base station (BS). (b) Broadcast graph: the edge between vertices indicates that the two cameras can overhear each other.*

We model the relation of overhearing among cameras as a *broadcast graph*. As shown in Figure 2.3b, each vertex represents a camera. An edge between vertices indicates that the two cameras are able to correctly overhear each other, with a probability above a certain threshold (e.g., 95%). We also assume that the radio channel between two cameras is symmetric, which means that if one camera can overhear the other, it is also the case in the opposite direction. Therefore, this broadcast graph is an undirected graph.

2.4 Self-Organized Sampling

As quoted in Strogatz's book on synchronization [15]:

"A great belt of light, some ten feet wide, formed by thousands upon thousands of fireflies whose green phosphorescence bridges the shoulder-high grass... The fluorescent band composed of these tiny organisms lights up and goes out with a precision that is perfectly, and one is left wondering what means of communication they possess which enables them to coordinate their shining as though controlled by a mechanical device."

This impressive phenomenon probably shows the largest scale of synchronization in nature. In electronic communications, thousands of devices in a network often require precise time synchronization. This problem has been studied extensively in sensor networks [16] [17]. In the interleaved sampling framework we proposed in Section 2.2, apparently all cameras also require to be synchronized to align their sampling grids properly. However, in contrast to the traditional synchronization problem where all devices share a common time reference, we seek to construct a network of cameras with evenly scattered time references. In the correct configuration, the clocks of cameras are synchronized with an offset of T/n between adjacent cameras. This is a more challenging problem, because having the correct time reference is not enough; we also need to know the number of cameras for adjusting the interleaved offset.

In this section, we investigate the idea of self-organized sampling, namely, the cameras in the network autonomously setup the optimal sampling configuration without

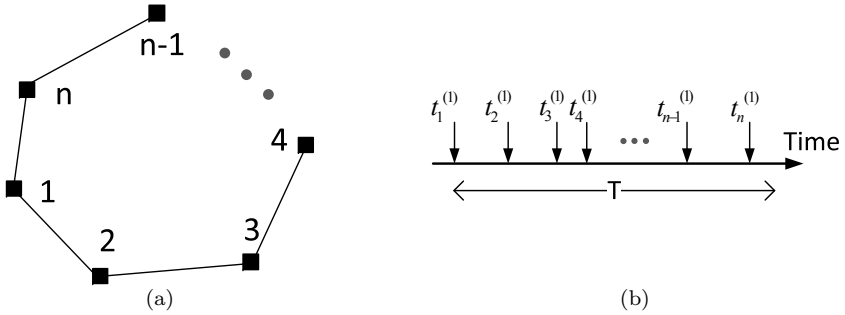


Figure 2.4: *A simple network scenario: (a) The broadcast graph of a n-camera system is a cycle graph. (b) The initial sampling schedule of cameras is in numerical order according to the ID of cameras, although they are not necessarily uniformly spaced.*

using a centralized decision mechanism. Based on the DESYNC algorithm proposed by Degesys et al. [18], we develop a distributed desynchronization algorithm that exploits wireless overhearing. It has to address the following *uncertainties*:

1. Local clocks of cameras are not synchronized and are subject to drift.
2. n decreases: a camera fails to work.
3. n increases: a new camera is installed.

2.4.1 DESYNC algorithm

The DESYNC algorithm was originally proposed by Degesys et al. [18] for coordinating wireless sensor nodes into a so-called *desynchronization status*, where nodes perform their tasks as far away as possible from each other. This is essentially the same configuration as our desired optimal sampling setup (Section 2.2). DESYNC is able to converge quickly when the network topology satisfies certain conditions. In the following, we develop DESYNC in our scenario and present its convergence conditions.

Consider a simple network scenario as shown in Figure 2.4a: n cameras are deployed in a circle around a common area of interest. Overhearing is only possible between adjacent cameras. The broadcast graph is therefore a *cycle graph*. Each camera is given an ID sequentially according to its position in the cycle. The initial sampling schedule of cameras is in the same order, although they are not necessarily uniformly spaced (Figure 2.4b). To simplify the problem, we assume that there is no *overhearing loss*, that is, if two cameras are connected in the broadcast graph, then all the messages transmitted by one camera are overheard at the other one.

We denote the camera 1 as an *Anchor* camera[1]. The *Anchor* camera always fixes its own schedule with a sampling interval T. The other cameras adjust their

[1]The *Anchor* is assigned by the base station. In the case that the *Anchor* fails in the long run, the base station should assign a new one.

Algorithm 2.1 Distributed desynchronization algorithm (DESYNC)

1. **if** I'm *Anchor* **then**
2. Repeat TX every T
3. **else**
4. t, t': current/next TX schedule of "me"
5. t^-: time stamp of the most adjacent beacon before my current TX ($t^- < t$)
6. t^+: time stamp of the most adjacent beacon after my current TX ($t^+ > t$)
7. $t' = T + (t^- + t^+)/2$
8. **end if**

sampling schedule over time to synchronize to the correct configuration, in which all cameras should adapt to the sampling interval T and have an offset T/n between adjacent cameras. The basic idea of the DESYNC algorithm is that each camera sends a beacon together with each data transmission (referred to as TX). Meanwhile, each camera records the time stamps of the overheard beacons from the neighboring cameras, and dynamically adjusts its next sampling time to be equally spaced from the two adjacent cameras. In this way, each camera dynamically adapts to the correct sampling configuration. The detailed algorithm is shown in Algorithm 2.1. The convergence of DESYNC has been proved [18]. We give an alternative proof in the following, which is stronger than the one in [18], because it indicates the convergence speed.

Theorem 2.2 (convergence of distributed desynchronization)**.** *Consider a camera network whose broadcast graph is a cycle graph, and the initial sampling schedule of cameras is in an order that corresponds to cameras' order in the cycle. Without overhearing loss, DESYNC converges to the uniform interleaved sampling configuration in an exponential manner.*

Proof. Denote the number of cameras as n, the scheduled sampling interval of each camera as T, and the time stamp of the kth transmission (TX) of the ith camera as $t_i^{(k)}$, where $i = 1, \cdots, n$, and $k = 1, 2, \cdots$. As shown in Figure 2.4b, the initial sampling schedule of cameras are in numerical order according to the ID of cameras:

$$t_1^{(1)} < t_2^{(1)} < \cdots < t_n^{(1)}. \tag{2.2}$$

Note that the cameras' IDs are labeled sequentially in the cycle, thus (2.2) implies that the initial sampling time of cameras is in an order that corresponds to their order in the cycle[2].

Assuming no overhearing loss, each camera can overhear the neighboring two cameras' beacons. As each camera adjusts its sampling schedule to be in the middle of the schedules of the two neighboring cameras, the order $t_1^{(k)} < t_2^{(k)} < \cdots < t_n^{(k)}$ is maintained for any k. Therefore, we can formulate the distributed desynchronization algorithm as:

[2]This is a strong assumption in practice. We will show that this assumption can be removed for a fully connected network, and provide a startup algorithm to establish a correct initial configuration in a partially connected network (Section 2.4.2).

$$\mathbf{t}^{(k+1)} = \mathbf{A} \cdot \mathbf{t}^{(k)} + \boldsymbol{\eta}, \tag{2.3}$$

where $\mathbf{t}^{(k)} = \left(t_1^{(k)}, t_2^{(k)}, t_3^{(k)}, \cdots, t_{n-1}^{(k)}, t_n^{(k)}\right)^{\mathrm{T}}$, and

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & \ddots & \vdots \\ 0 & \frac{1}{2} & 0 & \ddots & 0 \\ \ddots & \ddots & \ddots & \ddots & \frac{1}{2} \\ \frac{1}{2} & 0 \cdots & 0 & \frac{1}{2} & 0 \end{pmatrix}_{n\times n}, \quad \boldsymbol{\eta} = \begin{pmatrix} T \\ T \\ T \\ \vdots \\ T \\ \frac{3T}{2} \end{pmatrix}_{n\times 1}.$$

For the convergence of the iterative scheme (2.3), we first study the eigenvalues and eigenvectors of $\mathbf{A}$.

$$|\mathbf{A} - \lambda I| = \frac{(1-\lambda)}{\sqrt{\lambda^2-1}} \left(\left(\frac{-\lambda + \sqrt{\lambda^2-1}}{2} \right)^n - \left(\frac{-\lambda - \sqrt{\lambda^2-1}}{2} \right)^n \right), \tag{2.4}$$

thus the eigenvalues

$$\lambda_i = \cos \frac{(i-1)\pi}{n}, \ i = 1, \cdots, n. \tag{2.5}$$

Given the structure of $\mathbf{A}$, the corresponding eigenvector $\mathbf{v}_i$ satisfies $\mathbf{v}_1 = (1, 1, \cdots, 1)^{\mathrm{T}}$, and for $i = 2, \cdots, n$, the first element of $\mathbf{v}_i$ is zero.

Thus, we can expand

$$\mathbf{t}^{(1)} = t_1^{(1)}\mathbf{v}_1 + \sum_{i=2}^{n} a_i \mathbf{v}_i, \ \boldsymbol{\eta} = T\mathbf{v}_1 + \sum_{i=2}^{n} b_i \mathbf{v}_i, \tag{2.6}$$

where a_i, b_i are the expansion coefficients. As the solution of recurrence (2.3) is $\mathbf{t}^{(k)} = \mathbf{A}^k \mathbf{t}^{(1)} + \sum_{j=0}^{k-1} \mathbf{A}^j \boldsymbol{\eta}$, by plugging (2.6), we get

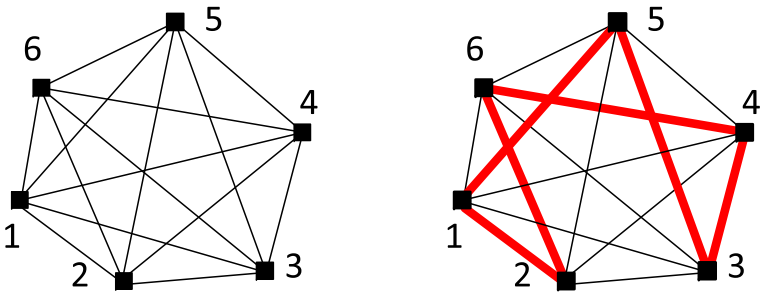
$$\begin{aligned} \mathbf{t}^{(k)} &= t_1^{(1)} \lambda_1^k \mathbf{v}_1 + \sum_{i=2}^{n} a_i \lambda_i^k \mathbf{v}_i + \sum_{j=0}^{k-1} \left(T\lambda_1^j \mathbf{v}_1 + \sum_{i=2}^{n} b_i \lambda_i^j \mathbf{v}_i \right) \\ &= (t_1^{(1)} + kT)\mathbf{v}_1 + \sum_{i=2}^{n} \left(a_i \lambda_i^k + b_i \frac{1-\lambda_i^k}{1-\lambda_i} \right) \mathbf{v}_i. \end{aligned} \tag{2.7}$$

The first equality follows from the fact that $\mathbf{A} \cdot \mathbf{v}_i = \lambda_i \mathbf{v}_i$ for $i = 1, \cdots, n$.

From (2.5), we know $|\lambda_i| < 1$ for $i > 1$. Thus, according to (2.7), $\mathbf{t}^{(k)} - (t_1^{(1)} + kT)\mathbf{v}_1$ converges exponentially to a *unique* solution:

$$\lim_{k\to\infty} (\mathbf{t}^{(k)} - (t_1^{(1)} + kT)\mathbf{v}_1) = \sum_{i=2}^{n} \frac{b_i}{1-\lambda_i} \mathbf{v}_i.$$

It is straightforward to verify that $\mathbf{t}^{(k)} - (t_1^{(1)} + kT)\mathbf{v}_1 = T \cdot \left(0, \frac{1}{n}, \cdots, \frac{n-1}{n}\right)^{\mathrm{T}}$ satisfies the convergence condition. Therefore, the algorithm converges to the uniform interleaved sampling configuration. □



(a) (b)

Figure 2.5: *A fully connected network with six cameras: (a) Broadcast graph, where any pair of cameras can overhear each other. (b) A Hamiltonian cycle of the broadcast graph, where the red bold cycle visits each vertex exactly once. The vertices in the cycle are in an order of* $1 \to 2 \to 6 \to 4 \to 3 \to 5 \to 1$.

DESYNC provides an efficient way to autonomously construct an interleaved sampling schedule for a cycle broadcast graph. In graph theory, a *Hamiltonian cycle* is a cycle that visits each vertex exactly once, except the vertex that is both the start and end. A graph that contains a Hamiltonian cycle is called *Hamiltonian*. Thus, in order to use DESYNC in other network topologies, it suffices to find a Hamiltonian cycle in the broadcast graph, and the vertices in the cycle should have the same order of the initial sampling schedule of cameras.

It is straightforward to show that DESYNC works in a fully connected network. Figure 2.5a shows the broadcast graph of a fully connected network (a complete graph). Note that as opposed to the conventional meaning of "connected", we denote a fully connected network as a network in which any pair of cameras can overhear each other. The red bold lines in Figure 2.5b shows a Hamiltonian cycle that corresponds to the initial sampling schedule $t_1^{(1)} < t_2^{(1)} < t_6^{(1)} < t_4^{(1)} < t_3^{(1)} < t_5^{(1)}$. Actually, in a fully connected network, we can always find a corresponding Hamiltonian cycle for any initial sampling schedule. Therefore, the requirement of a proper initial sampling schedule in Theorem 2.2 can be removed.

Proposition 2.3. *With a fully connected network and no overhearing loss, DESYNC always converges to the uniform interleaved sampling configuration in an exponential manner.*

Proof. Consider an initial sampling schedule of a n-camera network $t_{k_1}^{(1)} < t_{k_2}^{(1)} < \cdots < t_{k_{n-1}}^{(1)} < t_{k_n}^{(1)}$, where $\{k_1, \cdots, k_n\}$ are the n cameras' IDs: $\{1, \cdots, n\}$.

To prove that DESYNC converges to the uniform interleaved sampling configuration, it suffices to find a Hamiltonian cycle $k_1 \to k_2 \to \cdots \to k_{n-1} \to k_n \to k_1$, where k_i is the ID of the vertex (camera) in the broadcast graph. As the broadcast graph is a complete graph, any two vertices are connected. Therefore, the required Hamiltonian cycle can be constructed.

By using DESYNC, since we only use the most adjacent beacons to adjust the next schedule, the beacons overheard from non-neighboring cameras in the Hamiltonian

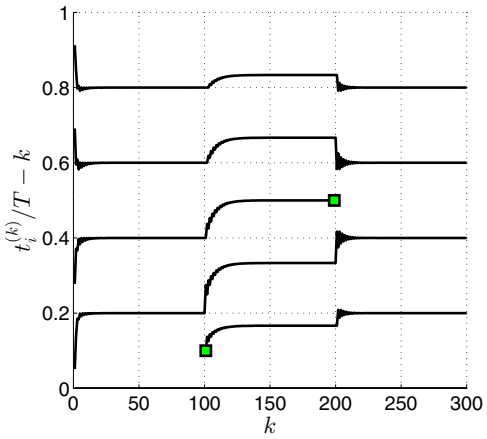


Figure 2.6: *Simulation result of DESYNC in a fully connected network with no overhearing loss. Five unsynchronized cameras start at $k = 0$, one camera pops up at $k = 100$, and one random camera dies at $k = 200$. The green squares labels the camera that shows/disappears. k denotes the number of sampling intervals elapsed. $t_i^{(k)}/T - k$ denotes the normalized sampling time of the ith camera. The trace of the* Anchor *camera always coincides with the x-axis.*

cycle will not affect the algorithm. Thus, according to Theorem 2.2, the network will converge to the uniform interleaved sampling configuration, in an exponential manner. □

Proposition 2.3 states that starting from any sampling configuration, DESYNC will always converges to an interleaved sampling network. This implies that we do not need to manually assign an initial sampling schedule with proper orders. Moreover, as opposed to traditional synchronization schemes, DESYNC is a distributed algorithm that can adaptively track the changes in the network. The total number of cameras n is not required to be known at the cameras. Once the system is alive, each camera maintains its own sampling interval T. When a camera fails or a new camera pops up (n changes), the system automatically adapts to the new sampling configuration, where the sampling frequency of the entire system is always equal to n/T.

Figure 2.6 shows the simulation results of DESYNC in a fully connected network with no overhearing loss. Five unsynchronized cameras start at $k = 0$. One camera appears with a random sampling time at $k = 100$, and one random camera dies at $k = 200$. k denotes the number of sampling intervals elapsed, thus it represents the kth TX of each camera. $t_i^{(k)}/T - k$ denotes the normalized sampling time of the ith camera, which should converge to $(i - 1)/n$ when there are n cameras. Each trace records the evolution of a camera's normalized sampling time. Note that the trace of the *Anchor* camera always coincides with the x-axis because the *Anchor* has a fixed sampling schedule. Under the perfect overhearing condition, Figure 2.6 shows that all cameras quickly converge to the optimal sampling configuration (within 25 steps), withstanding the changes in the network.

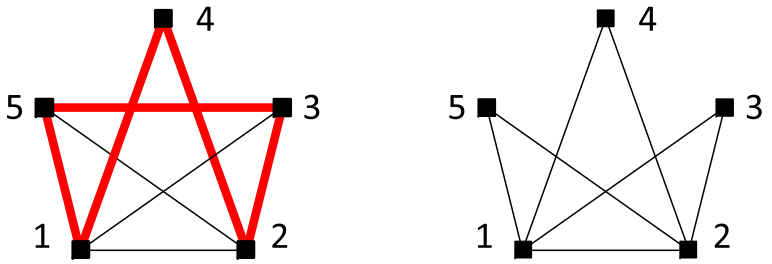


Figure 2.7: *Partially connected networks with five cameras: (a) A broadcast graph that contains Hamiltonian cycles (red bold lines). (b) A broadcast graph without any Hamiltonian cycle.*

2.4.2 DESYNC in Partially Connected Networks

A partially connected network, in which not every two cameras can overhear each other, is a more general case than the fully connected network discussed in the previous section. Degesys and Nagpal [19] have further investigated DESYNC under various network topologies. However, the only presented topologies that converge to the interleaved configuration are the cycle graph and the complete graph. In this section, we investigate partially connected networks and address two important questions:

- What kind of networks can achieve self-organized sampling?
- Does DESYNC work in a partially connected network? If not, what other components do we need to assist DESYNC?

Similar to the proof of Proposition 2.3, if we can find a *Hamiltonian cycle* that connects all the vertices of the broadcast graph, DESYNC converges exponentially to the optimal sampling schedule. This idea can be generalized as following:

Proposition 2.4. *If the broadcast graph of a network has a Hamiltonian cycle and no overhearing loss exists, DESYNC converges to the uniform interleaved sampling configuration in an exponential manner, provided that the initial sampling schedule is in the same order as the positions of cameras in the Hamiltonian cycle.*

Proof. For a n-camera network, assume we find a Hamiltonian cycle in the broadcast graph $k_1 \to k_2 \to \cdots \to k_{n-1} \to k_n \to k_1$, where k_i is the ID of the vertex (camera). If we set the initial sampling schedule as $t_{k_1}^{(1)} < t_{k_2}^{(1)} < \cdots < t_{k_{n-1}}^{(1)} < t_{k_n}^{(1)}$, then the topology and the initial sampling schedule of this setup meets the network model (Figure 2.4) defined in Section 2.4.1.

By using DESYNC, as we only use the most adjacent beacons to adjust the next schedule, the beacons overheard from non-neighboring cameras in the Hamiltonian cycle will not affect the algorithm. Thus, according to Theorem 2.2, the network will converge to the uniform interleaved sampling configuration, in an exponential manner. □

Algorithm 2.2 Startup procedure of DESYNC

1. Find a Hamiltonian cycle in the broadcast graph by using either centralized or distributed algorithms.
2. Starting from the *Anchor* camera, an *initialization* message is relayed in the found Hamiltonian cycle to coordinate the initial sampling schedule.
3. Start DESYNC.

Figure 2.7 shows two sample networks with and without Hamiltonian cycles, respectively. Note that the condition in Proposition 2.3 is sufficient but not necessary to synchronize a camera network to the uniform interleaved sampling configuration. For instance, if we use one specific node to do centralized decisions and synchronize the whole network, then the Hamiltonian condition is not necessary. Such methods are out of scope of this thesis.

Proposition 2.4 suggests that the first step after camera deployment is to check if the broadcast graph contains any Hamiltonian cycles. From graph-theoretic point of view, this is a classical *Hamiltonian cycle problem* that can be solved in a centralized fashion. This problem is well known to be NP-complete [20], and the exact solution has no polynomial-time algorithm. If the graph is modeled as a random graph, then there are polynomial-time heuristic algorithms that are able to find the solution with a certain probability of success [21]. In line with this research direction, Levy [22] proposed a distributed algorithm with linear computation complexity when the network is dense enough. In practice, we can use two approaches to find a Hamiltonian cycle (if it exists) in a partially connected network:

1. One camera collects adjacency tables of all cameras and runs a centralized algorithm such as [21].

2. Each camera communicates with neighbors through broadcast links and use a distributed algorithm [22] to find the solution.

After a Hamiltonian cycle is found in the broadcast graph, we then need to arrange initial sampling schedule as the same order of cameras in the cycle. We summarize in Algorithm 2.2 the overall startup procedures as a preparation step for the DESYNC algorithm.

As shown in Section 2.4.1, for a fully connected network, DESYNC is able to adaptively adjust the sampling configuration when an old camera fails or a new camera arrives. However, this is not the case for a partially connected network, because the Hamiltonian cycle will be broken if we simply remove or add a vertex. Hence, we propose Algorithm 2.3 and Algorithm 2.4 to re-establish a Hamiltonian cycle for i) camera failures and ii) camera arrivals, respectively. Both algorithms first try to fix the Hamiltonian cycle locally, and use Algorithm 2.2 to re-initiate the whole network only if necessary. Note that for both Algorithm 2.3 and Algorithm 2.4, a Hamiltonian cycle can be established only if the new broadcast graph is still Hamiltonian.

2.4.3 Overhearing Loss

In previous sections, we assume that there is no overhearing loss, which is not the case in most practical scenarios. As defined in Section 2.3, if we assume that

Algorithm 2.3 Recovering procedure for camera failures

1. **if** a camera c_1 cannot overhear from the original neighboring camera c_2 for several T **then**
2. The camera c_2 is dead.
3. In the Hamiltonian cycle, as each camera has two neighbors, there will be another camera c_3 also finds the loss of camera.
4. c_1 and c_3 repeatedly broadcasts a *reconnect* message.
5. **if** c_3 or c_1 received the *reconnect* message during a given period **then**
6. It transmits back a *reconnect* message to recover the broken link between c_1 and c_3.
7. **else**
8. Re-initiate the whole network with Algorithm 2.2.
9. **end if**
10. Start DESYNC.
11. **end if**

Algorithm 2.4 Merging procedure for a new camera

1. Keep silent and overhear all the beacons for several T.
2. Find the schedules of two closest beacons in time. If there are multiple choices, select a random one. (If the new broadcast graph is still Hamiltonian, then the two cameras that transmit the found beacons must be neighbors in the original Hamiltonian cycle.)
3. Start sampling with an initial schedule in the middle of the two found beacons.
4. Start DESYNC.
5. **if** the schedule does not converge **then**
6. The new topology is not a Hamiltonian cycle.
7. Re-initiate the whole network with Algorithm 2.2.
8. Start DESYNC.
9. **end if**

the cameras in the broadcast graph are connected when they are able to successfully overhear each other with a probability (e.g., 95%), the algorithm performance can be degraded due to beacon losses.

Figure 2.8a shows the performance of the DESYNC algorithm with 10% of overhearing loss, in a fully connected network. As we can see, DESYNC is not robust to such errors. To overcome this problem, we propose an improved RoDESYNC algorithm that adaptively limits the rate of change of sampling schedules. As described in Algorithm 2.5, each camera has a threshold variable `thre` that is initialized to a certain value at startup (e.g., $0.01 \cdot T$). At each update round, the maximum adjustment is limited to `thre`, and `thre` is adaptively adjusted in the next round, according to the requested adjustment. In this way, the algorithm can learn the optimal threshold and is robust to burst errors caused by overhearing loss. Figure 2.8b shows the performance of RoDESYNC. With the same overhearing loss rate, the improved algorithm converges fast (within 50 steps), and remains steady despite beacon losses. Note that although Figure 2.8 simulates a fully connected network, it is also valid for partially connected networks by replacing DESYNC with RoDESYNC.

Algorithm 2.5 Robust distributed desynchronization algorithm (RoDESYNC)

if I'm *Anchor* **then**
  Repeat TX every T
else
  t, t': current/next TX schedule of "me"
  t^-: time stamp of the most adjacent beacon before my current TX ($t^- < t$)
  t^+: time stamp of the most adjacent beacon after my current TX ($t^+ > t$)
  $t_\Delta = (t^- + t^+)/2 - t$
  if $|t_\Delta| >$ `thre` **then**
    $t' = T + t + \mathrm{sgn}(t_\Delta) \cdot$ `thre`
  else
    $t' = T + t + t_\Delta$
  end if
  if $|t_\Delta| > 2 \cdot$ `thre` **then**
    `thre` $= 2 \cdot$ `thre`
  else if $|t_\Delta| <$ `thre`$/2$ **then**
    `thre` $=$ `thre`$/2$
  end if
end if

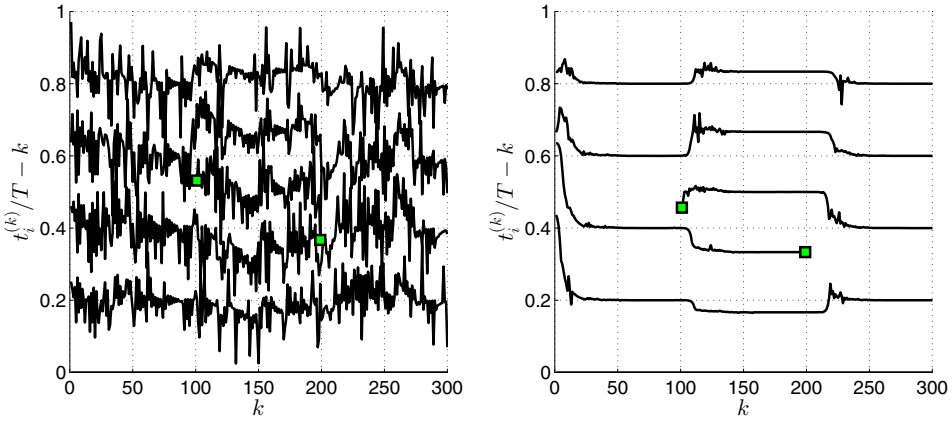


Figure 2.8: *Simulation results of distributed desynchronization algorithm in a fully connected network with 10% overhearing loss. Five unsynchronized cameras start at $k = 0$, one camera pops up at $k = 100$, and one random camera dies at $k = 200$. The green squares labels the camera that shows/disappears. k denotes the number of sampling intervals elapsed. $t_i^{(k)}/T - k$ denotes the normalized sampling time of the ith camera. (a) DESYNC. (b) RoDESYNC.*

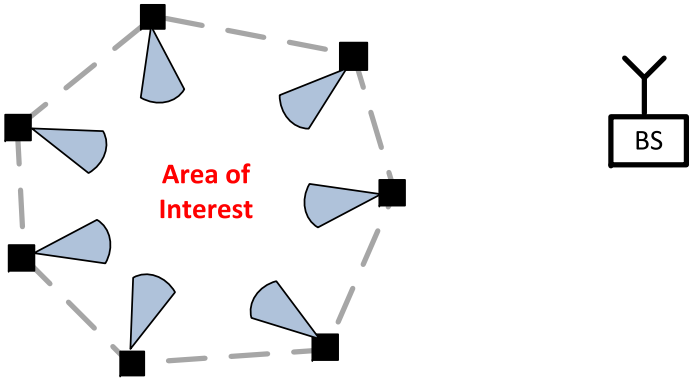


Figure 2.9: *A circular deployment of cameras around a common area of interest. This setup creates a broadcast graph as in Figure 2.4. The dashed lines show the overhearing links between cameras.*

2.4.4 Deployment Guide

As we discussed in Section 2.4.2, DESYNC converges when the broadcast graph is Hamiltonian. Hence, it is essential to keep this constraint in mind during deployment of cameras. As the purpose of the multi-camera network is to monitor a particular area of interest while introducing system robustness, a natural deployment is to place cameras evenly in a circle around the area of interest. Note that the base station (BS) is usually outside the local camera network, thus the radio transmission of each camera can cover the neighboring two cameras. As shown in Figure 2.9, this setup creates a broadcast graph that is same as the cycle graph in Figure 2.4, and it is indeed Hamiltonian.

Besides a circular setup as in Figure 2.9, the following theorem by Ore [23] indicates an alternative rule of deployment for creating a Hamiltonian broadcast graph.

Theorem 2.5 (Ore's theorem [23])**.** *A graph with n vertices ($n \geq 3$) is Hamiltonian if, for each pair of non-adjacent vertices, the sum of their degrees is n or greater.*

This result essentially states that a graph with "sufficiently many edges" will have a Hamiltonian cycle. On average, if each vertex in the broadcast graph has a degree of more than $n/2$, then this broadcast graph must be Hamiltonian according to Theorem 2.5. As a rule of thumb, this is equivalent to say that *the radio transmission range of each camera should cover at least half of the network.*

2.5 Summary

- A multi-camera system that monitors a common area of interest introduces robustness to the system, especially for outdoor and surveillance applications.
- To maximize event-detection probability, multiple cameras must be coordinated to perform interleaved sampling for minimum number of overall sampling operations. A multi-camera system has the same event detection probability as a traditional single camera system, by using the cooperative sampling framework.
- A single-hop star topology of wireless communication ensures the robustness of the communication link.
- Wireless broadcast links can be used for coordinating neighboring cameras, which deliver sparse and short messages. Based on DESYNC and overhearing, we develop a distributed algorithm that synchronizes the camera network to the interleaved sampling configuration.
- For a fully connected network, DESYNC converges exponentially to the optimal sampling configuration. For a partially connected network, DESYNC converges if the broadcast graph contains a Hamiltonian cycle.
- To address overhearing losses, we propose an improved RoDESYNC algorithm that adaptively limits the rate of change of DESYNC.
- A circular deployment of cameras around a common area of interest provides a Hamiltonian broadcast graph. Alternatively, if the radio transmission range of each camera covers at least half of the network, then a Hamiltonian broadcast graph is guaranteed.

Chapter 3

Cooperative Coding for Distributed Image Sources

3.1 Introduction

From this chapter until Chapter 5, based on the cooperative sampling framework described in Chapter 2, we investigate cooperative coding methods for static images, videos, and event detection applications, in three chapters, respectively:

1. For static images captured by distributed cameras, we study a distributed successive refinement coding problem that incorporates interleaved procedures.

2. For videos captured by distributed cameras, we study cooperative coding schemes for achieving efficient data compression.

3. For event detection applications, we study an energy-efficient coding algorithm that transmits only event-specific information captured by distributed smart cameras.

Note that in the first coding scenario, we only consider static multi-view images, while in the later two, the cameras employ the cooperative sampling framework to capture video streams. Nevertheless, for static images, we split the coding procedure into successive phases, imitating the interleaved sampling configuration in cooperative sampling framework.

In this chapter we study coding methods for static images: As shown in Figure 3.1, two cameras need to transmit the stereo-view images X and Y to the base station (BS). We separate the transmission into several phases, so that the BS can recover the image X and Y with progressively increasing quality. This is particularly useful in an energy-limited communication scenario, because we can decide at the receiver whether a high resolution image is really needed after the low resolution version is displayed. Such an idea is called successive refinement and the single source case has been studied extensively in both theory and practical schemes (Section 3.1.1). Several theoretical variants of the multiple source case have been recently investigated (Section 3.1.1), but there is hardly any practical successive refinement schemes for multi-view images.

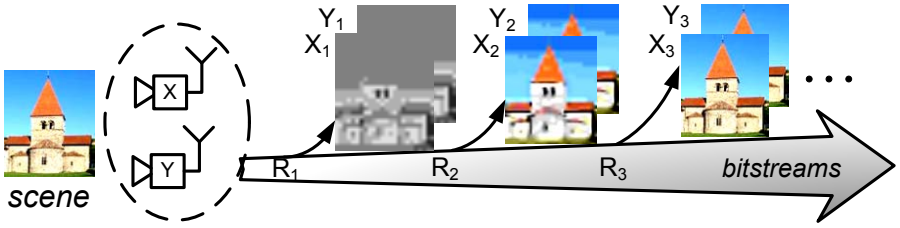


Figure 3.1: *Successive refinement of stereo-view images. Bitstreams are transmitted in several stages, and the base station can recover X and Y with increasing quality.*

To address this problem, we propose a novel two-encoder successive refinement scheme, which we call *Distributed Successive Approximation Coding using Broadcast Advantage (DiSAC2)*. DiSAC2 exploits the broadcast link as a free gossip channel, and two cameras cooperate in a ping-pong fashion to transmits refinements. We study the theoretical property of DiSAC2 assuming X, Y are modeled by a bivariate Gaussian distribution (Section 3.3). It is shown that DiSAC2 has no loss in coding efficiency as compared to the conventional distributed coding limit, for arbitrary successive refinement settings (any number of stages and rate combinations). Then, we apply DiSAC2 to real images and propose a practical stereo-view image coding algorithm (Section 3.4). The simulation results show that DiSAC2 operates close to the distributed coding bound with a gap of less than 3dB. Moreover, as opposed to conventional independent coding where the broadcast link is not used, DiSAC2 achieves up to 5dB of performance gain. Finally, as an alternative to traditional digital communication schemes, we investigate an uncoded transmission scheme that also exploits the broadcast link (Section 3.5). Unlike the single encoder case, we find that the analog communication does not outperform digital schemes in distributed scenarios.

3.1.1 Background

As the idea of distributed successive refinement coding originates from two well studied topics, namely, distributed source coding and successive refinement coding, we briefly present the important background on those topics.

Distributed Source Coding

In a typical multi-camera network, two cameras transmit images independently to the base station (BS), which can be well modeled as a *two-encoder distributed source coding (DSC2)* setup [24]. When broadcast is employed, a passive communication link exists between the two cameras and then partially separated encoders provides a better model (see Figure 3.2).

To deliver images under the severe energy constraints of wireless networks, multi-terminal source coding has an important role as it can push the rate to the theoretical lower limit. The general multi-terminal source coding problem [25] has been posed more than thirty years ago. The rate region for the distributed lossless source coding

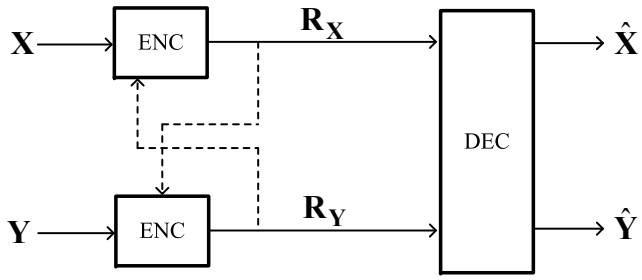


Figure 3.2: *Distributed source coding of two partially separated encoders: overhearing provides a passive communication link (dashed line).*

problem has been solved by Slepian and Wolf [26]. However, the general lossy case is not fully determined yet. Wyner and Ziv [27] solved a special case when one of two sources is entirely known at the decoder. Recently, Wagner et al. [28] gave the rate-distortion region for the two-encoder quadratic Gaussian case[1]. To give an illustration of Wagner's four-dimensional rate-distortion region (R_X, R_Y, D_X, D_Y), we pick the sum-rate $R_X + R_Y$ as a measure of the rate budget and show the coding limits in a three-dimensional space $(R_X + R_Y, D_X, D_Y)$. This *Wagner Surface* represents the rate distortion limits that any distributed coding scheme should operate on or above (see the surface in Figure 3.6).

Practical distributed single-view video coding schemes [29] [30], and distributed multi-view video coding schemes [31] have been developed in recent years. These schemes require no inter-frame communication, but are also very restricted to strong correlation between frames. For instance, if the video frame rate decreases or there are occlusions between different views, the coding efficiency drops very quickly.

Distributed source coding using the broadcast link is a particular form of the source coding problem with partially separated encoders. The rate-distortion region for two-encoder case was first addressed in [32], where coding theorems are determined for two cases: (i) one source is reproduced perfectly at the receiver; (ii) one source is perfectly revealed to the other source. In [33] and [34], this idea is further developed in a lossless Slepian-Wolf setup when an encoder can observe the codeword from the other encoder. It is proved that the admissible rate region is not enlarged. However, the general rate-distortion region for a lossy setup (Figure 3.2) is still unknown today.

Successive Refinement Coding

Comparing with conventional coding methods, successive refinement splits the single codeword into multiple pieces and make it possible to gradually send and reconstruct source(s) with increasing quality. From the rate-distortion perspective, the rate-distortion $R(D)$ of the given source(s) characterizes the performance limit of successive refinement coding: As depicted in Figure 3.3 for a single source, (R_1, D_1) and $(R_1 + R_2, D_2)$ are the R-D pairs achieved at the 1st and 2nd stage respectively. Generally, for most source types, successive refinement operates above the $R(D)$ curve. If (R_1, D_1)

[1] All distortions are defined as $\mathbb{E}d(x, \hat{x})$ where $d(\cdot, \cdot)$ is the quadratic error measure.

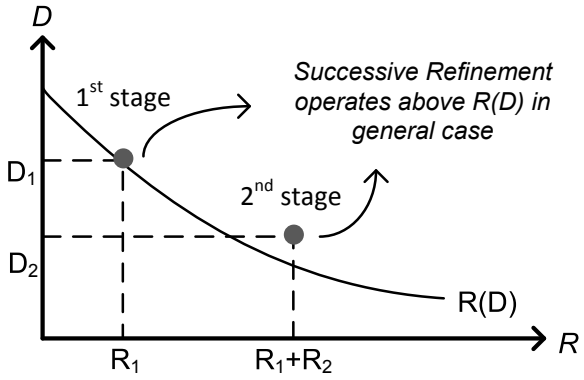


Figure 3.3: *Rate distortion performance: two-stage successive refinement of a single source.*

and $(R_1 + R_2, D_2)$ both operate exactly on the $R(D)$ curve of a source, for any (R_1, R_2), then we say that successive refinement of this source is optimal (*successive refinability*).

Equitz and Cover [35] gave the necessary and sufficient condition for a single source to be successively refinable. To encode source X with rate-distortion function $R(D)$, a coarse description $\hat{X}_1$ with R-D pair (R_1, D_1) is refined to a finer description $\hat{X}_2$ with a rate of R_2. The distortion of $\hat{X}_2$ is denoted as D_2. The optimality, namely, $R_1 = R(D_1)$ and $R_1 + R_2 = R(D_2)$ are both achievable, is obtained if and only if we can write $\hat{X}_1 \to \hat{X}_2 \to X$ as a Markov chain. A Gaussian source with squared-error distortion is one example that satisfies this Markovian condition.

On the question of successive refinement for multiple sources, [36] proposes a sequential coding of correlated sources for video applications, in which the first source is encoded solely while the subsequent source is encoded based on both sources. This scheme is a weak version of centralized coding as it has access to both sources. However, as it does not fully exploit the joint information due to the first step encoding, the minimum sum-rate is sometimes worse than DSC2. Recently, [37] proposed a successive decoding scheme for the distributed source coding problem (no link between encoders). It is proved that successive decoding following a linear fusion achieves the rate-distortion region of DSC2 for the quadratic Gaussian case. However, the final step of fusion actually breaks the successive decoding structure: all results have to be reconstructed after everything is received.

In practical scenarios, images are usually not well modeled by Gaussian sources, thus it is non-trivial to say if a certain image is successively refinable or not. Most of the contemporary single image coders support progressive or scalable coding (synonyms for successive refinement). In particular, the JPEG2000 image coding standard is designed to be scalable in nature [38]: it decomposes the bitstreams into a succession of layers, and each layer contains additional contributions optimized for rate-distortion performance. Therefore, the layered decomposition provides an approximation of successive refinement, as long as the bitstream is truncated at a layer point.

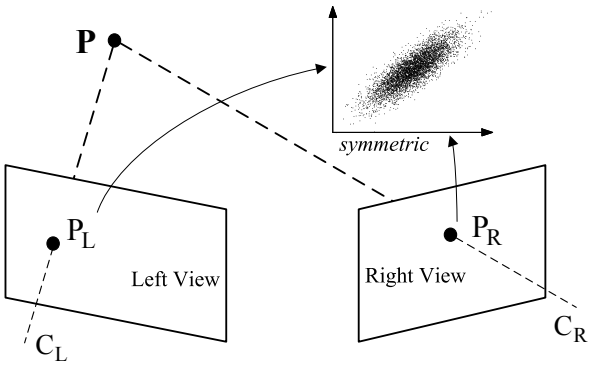


Figure 3.4: *Projections (P_L and P_R) of a given point P in stereo-views are statistically linked by a certain distribution with equal marginal distributions.*

3.2 Broadcast Link and Inter-View Correlation

In this section, we present the communication and image models used in this chapter.

3.2.1 Broadcast Link

As we discuss in Section 2.3, we can exploit the broadcast nature of wireless communications to get a free communication link between cameras. To simplify the theoretical study of the coding method, we further make several assumptions:

1. Overheard messages are error-free: All cameras within the transmission range of the transmitter get the message with no errors, as long as this message is also received at the BS.

2. No message collisions: Both cameras are synchronized and use time-division multiplexing so that only one camera transmits at a time while the other overhears the broadcast message without interference.

Note that the first assumption does not imply there is no channel noise. By Shannon's channel coding theorem [39], the message can be delivered over a wireless channel as long as the source coding rate is smaller than the channel capacity (depending on the channel noise). In our setup (Section 2.3), as the scale of the local camera network is small compared with their distance to the base station (BS), the channel capacity between neighboring cameras is larger than the capacity between the camera and the BS (Figure 2.2). Therefore, as long as the source coding rate meets the channel capacity between the camera and the BS, it also meets the channel capacity between neighboring cameras.

3.2.2 Inter-View Correlation

The correspondence between two stereo-view images can be completely described by epipolar geometry [40] using the pinhole camera model. Illustrated in Figure 3.4, a

point P in real world is projected onto left and right image planes, and the corresponding projection points are P_L and P_R respectively. The intensities of P_L and P_R depend on the intensity of P (if P lies on a lambertian surface), the light path from P to the image plane, and the optical system of the camera. Considering the fact that the light path in air is homogeneous and the imaging system of two cameras are almost identical under proper calibration, the intensities of P_L and P_R are expected to be highly correlated. Therefore, any correspondence point pair determined by epipolar geometry can be statistically modeled by a certain bivariate distribution. Note that this probability distribution should have equal marginal distributions due to the symmetry of the left/right views. If we further assume that such a model is globally *stationary* over the entire image, then the inter-view correlation can be fully described by a single distribution model[2].

3.3 Distributed Successive Refinement of Bivariate Gaussian Sources

In this section, we investigate a theoretical framework of distributed successive refinement. By using the broadcast link, we propose a coding scheme that imitates the ping-pong game, and characterize its rate-distortion performance when the two distributed sources are statistically linked by a jointly bivariate Gaussian model.

3.3.1 Setup and Coding Procedure

The broadcast nature of wireless communication provides a free overhearing mechanism that can be exploited to reduce the transmission rate between the cameras and the BS. For instance, in a two-encoder setup (Figure 3.2), based on the codeword sent by source X, source Y can be encoded at a lower rate by exploiting the correlation between X and Y.

Inspired by the ping-pong game, we extend such an idea to a new distributed source coding scheme, which has a successive refinement structure. As Figure 3.5 shows, the broadcast messages act like a ping-pong ball, which is flipped back and forth between the two encoders. We call such a scheme *two-encoder Distributed Successive Approximation Coding using Broadcast Advantage (DiSAC2)*. We denote the number of encoding stages in DiSAC2 as M (e.g., Figure 3.5 illustrates 4 stages), and ENCx and ENCy as two separate encoders for sources X and Y respectively.

Note that the interleaved sampling configuration of Theorem 2.1 is strongly related to the coding procedure of DiSAC2: As shown in Figure 2.1, two monitoring cameras take turns monitoring the scene just like in a ping-pong game where two players take turns striking the ball.

The sketch of the coding procedure for a M-stage DiSAC2 is as follows:

1. At the first stage, ENCx encodes X without any knowledge of Y. A codeword C_1 is generated and has a rate of R_1. (X_1, Y_1) is reconstructed at the BS after C_1 is received. The corresponding distortion pair is (D_{X_1}, D_{Y_1}).

[2]While this is an approximation, it is a useful model for further mathematical analysis.

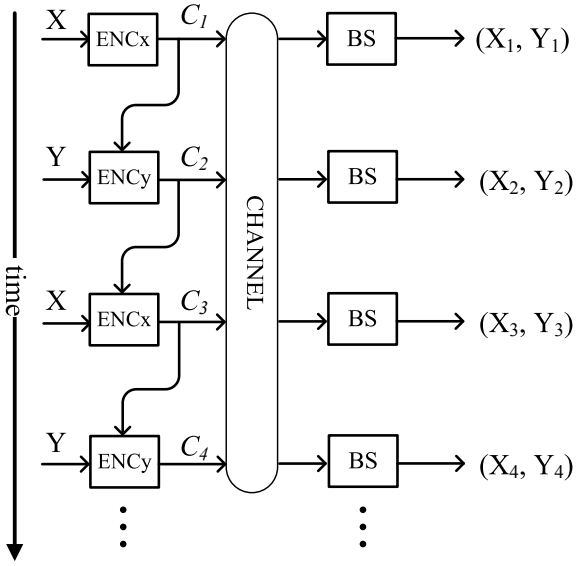


Figure 3.5: *Setup of a two-encoder Distributed Successive Approximation Coding using Broadcast Advantage (DiSAC2). The rate of codeword C_k sent at the kth stage is R_k. The corresponding reconstruction at the BS has a distortion pair (D_{X_k}, D_{Y_k}). Four stages are depicted.*

2. At the second stage, ENCy overhears the codeword C_1 while it is being transmitted to the BS, so it only transmits the refinement which fully exploits the joint information between the source Y and C_1. C_2 is the corresponding codeword sent in the second stage, which has a rate of R_2. (X_2, Y_2) is reconstructed at the BS based on (C_1, C_2). The corresponding distortion pair is (D_{X_2}, D_{Y_2}).

3. Similarly, at the stage k, ENCx or ENCy (depending if k is odd or even) encodes X or Y based on the codewords $(C_1, \cdots, C_{k-1})$. A codeword C_k is generated and has a rate of R_k. (X_k, Y_k) is reconstructed at the BS based on $(C_1, \cdots, C_k)$. The corresponding distortion pair is (D_{X_k}, D_{Y_k}).

If $X = Y$, Figure 3.5 reduces to a successive refinement of a single source. As we know from Section 3.1.1, this is successively refinable on the {*rate, distortion*} curve when the Markovian condition is satisfied (e.g., Gaussian source). Similar results can be investigated for the DiSAC2 scheme. In the following, we specifically discuss the M-stage DiSAC2 with jointly Gaussian sources and quadratic distortion (*DiSAC2-Gaussian*). We assume $(X, Y) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is a bivariate Gaussian distribution with the mean vector $\boldsymbol{\mu} = (0, 0)$, and the covariance matrix

$$\boldsymbol{\Sigma} = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \text{ for } |\rho| < 1.$$

3.3.2 The Gaussian DiSAC2 is Successively Refinable

As we know, a Gaussian source with quadratic distortion is successively refinable on its {*rate, distortion*} *curve*. In this section, we show that a M-stage DiSAC2 in the quadratic Gaussian case is also successively refinable on the {*sum-rate, distortion pair*} *surface* characterized by the rate-distortion region of the DSC2 [28].

Theorem 3.1 (successive refinability)**.** *Given two jointly Gaussian sources* $(X, Y) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$*, the* M*-stage two-encoder Distributed Successive Approximation Coding using Broadcast Advantage (DiSAC2-Gaussian) with rates* $(R_1, R_2, \cdots, R_M)$*, and sum-rate distortion pairs*

$$\{R_1, (D_{X_1}, D_{Y_1})\}, \cdots, \{R_1 + \cdots + R_M, (D_{X_M}, D_{Y_M})\},$$

achieves the {*sum-rate, distortion pair*} *surface characterized by the rate-distortion region of DSC2* $R_{DSC2}(D_X, D_Y)$ *[28], or equivalently*

$$R_{DSC2}(D_{X_k}, D_{Y_k}) = R_1 + \cdots + R_k, \quad \text{for } k = 1, \cdots, M.$$

Sketch of proof The detailed proof of Theorem 3.1 is given in the Appendix (Section 3.A) together with two preparatory lemmas. To provide the reader the methodology we used, we sketch the proof for the simplest two-stage case. In the following, $\mathcal{G}_{\mu,\sigma^2}(x)$ represents a Gaussian function $\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$. For brevity, we do not explicitly write out realizations in a probability density. For instance, $p(Y|X) = \mathcal{G}_{\rho X, 1-\rho^2}(Y)$ represents that the conditional probability of Y given X is a Gaussian distribution with a mean ρX and a variance $1-\rho^2$, where X and Y are treated as the realizations on the right-hand side of the equation. The proof for the general M-stage case is based on induction, and employs similar reasoning.

From [28], the minimum sum-rate for DSC2 is

$$R_{\mathrm{DSC2}}(D_X, D_Y) = \frac{1}{2} \log \frac{\left(1-\rho^2\right)\left(\sqrt{\frac{4D_X D_Y \rho^2}{(1-\rho^2)^2}+1}+1\right)}{2D_X D_Y}. \tag{3.1}$$

For a two-stage DiSAC2 with jointly Gaussian sources and quadratic distortion, we need to prove

$$\begin{cases} R_1 = R_{\mathrm{DSC2}}(D_{X_1}, D_{Y_1}) \\ R_1 + R_2 = R_{\mathrm{DSC2}}(D_{X_2}, D_{Y_2}) \end{cases},$$

so that it achieves the {*sum-rate, distortion pair*} surface characterized by the rate-distortion region of the DSC2, for any rate pair (R_1, R_2).

Following the coding procedure in Section 3.3.1, at the first stage, X is encoded to C_1 using the random codebook argument with R-D pair

$$R_1 = \frac{1}{2} \log \frac{1}{D_{X_1}}, \quad D_{X_1} \leq 1.$$

Then X_1 is the reconstruction of X after the first stage, and it can be decoded as: $X_1 = C_1$, thus $\mathbb{E}d(X, X_1) = D_{X_1}$. As $p(X) = \mathcal{G}_{0,1}(X)$, the *test channel* in the first stage is $p(X|C_1) = \mathcal{G}_{C_1, D_{X_1}}(X)$. Substituting $X_1 = C_1$,

$$p(X|C_1) = \mathcal{G}_{X_1, D_{X_1}}(X). \tag{3.2}$$

To decode Y_1 (reconstruction of Y at the first stage), we calculate the conditional probability

$$\begin{aligned} p(Y|C_1) &= \int_{-\infty}^{+\infty} p(YX|C_1)\ \mathrm{d}X \\ &= \int_{-\infty}^{+\infty} p(Y|X)\cdot p(X|C_1)\ \mathrm{d}X, \end{aligned} \tag{3.3}$$

where the second equality follows from the fact that C_1 is encoded and decoded from X (a definite function of X), thus $p(Y|C_1X) = p(Y|X)$. From the joint distribution of (X, Y),

$$p(Y|X) = \mathcal{G}_{\rho X, 1-\rho^2}(Y). \tag{3.4}$$

Substituting (3.4) and (3.2) into (3.3) leads to

$$p(Y|C_1) = \mathcal{G}_{\mu_1, \sigma_1^2}(Y), \tag{3.5}$$

where $\mu_1 = \rho X_1$, and $\sigma_1^2 = D_{X_1}\rho^2 - \rho^2 + 1$. Y_1 can be decoded as: $Y_1 = \mu_1$, and $\mathbb{E}d(Y, Y_1) = \sigma_1^2$.

To sum up, the {*sum-rate, distortion pair*} in the first stage is:

$$\begin{cases} R_1 = \dfrac{1}{2}\log\dfrac{1}{D_{X_1}} \\ (D_{X_1}, D_{Y_1}) = \left(D_{X_1}, \sigma_1^2\right) \end{cases}. \tag{3.6}$$

Combining (3.1) and (3.6) leads to

$$R_1 = R_{\mathrm{DSC2}}(D_{X_1}, D_{Y_1}). \tag{3.7}$$

At the second stage, C_1 is known due to the broadcast link, thus according to (3.5), $p(Y - \mu_1) = \mathcal{G}_{0,\sigma_1^2}(Y)$. Then, $Y - \mu_1$ is encoded to C_2 using the random codebook argument, with R-D pair

$$R_2 = \frac{1}{2}\log\frac{\sigma_1^2}{D_{Y_2}}, \quad D_{Y_2} \le \sigma_1^2.$$

Following this, Y_2 can be decoded as: $Y_2 = C_2 + \mu_1$, thus

$$\begin{aligned} \mathbb{E}d(Y, Y_2) &= \mathbb{E}d(Y - \mu_1, Y_2 - \mu_1) = \mathbb{E}d(Y - \mu_1, C_2) \\ &= D_{Y_2}. \end{aligned}$$

Similar reasoning as for the first stage calculation gives

$$p(X|C_1C_2) = \mathcal{G}_{\mu_2, \sigma_2^2}(X), \tag{3.8}$$

where

$$\mu_2 = \frac{X_1\left(1-\rho^2\right) + D_{X_1}Y_2\rho}{D_{X_1}\rho^2 - \rho^2 + 1},$$

and

$$\sigma_2^2 = \frac{D_{X_1}\left(\left(\rho^2-1\right)^2 - D_{X_1}\rho^2\left(-D_{Y_2}+\rho^2-1\right)\right)}{\left(D_{X_1}\rho^2-\rho^2+1\right)^2}.$$

X_2 can be decoded as: $X_2 = \mu_2$, and $\mathbb{E}d(X, X_2) = \sigma_2^2$.
Thus, the $\{$*sum-rate, distortion pair*$\}$ in the second stage is:

$$\begin{cases} R_1 + R_2 = \dfrac{1}{2}\log\dfrac{\sigma_1^2}{D_{X_1}D_{Y_2}} \\ (D_{X_2}, D_{Y_2}) = \left(\sigma_2^2, D_{Y_2}\right). \end{cases} \tag{3.9}$$

Combining (3.9) and (3.1), we can verify after some calculations:

$$R_1 + R_2 = R_{\mathrm{DSC2}}(D_{X_2}, D_{Y_2}). \tag{3.10}$$

□

Figure 3.6 gives a visual illustration of Theorem 3.1: We choose an initial operating point $\{R_1, (D_{X_1}, D_{Y_1})\}$ at the first stage of DiSAC2-Gaussian, and then send an additional rate R_2 at the second stage. As the dashed curve on the gray surface suggests, $\{R_1 + R_2, (D_{X_2}, D_{Y_2})\}$ with any R_2 travels along a one-dimensional curve on the *Wagner Surface*. Similarly, after we send an additional rate R_3 at the third stage, $\{R_1 + R_2 + R_3, (D_{X_3}, D_{Y_3})\}$ still travels along a one-dimensional curve on the *Wagner Surface*. This means that we can split the overall rate into any number of pieces, send successively, and they will all operate on the *Wagner Surface* at each stage.

It is worth mentioning that we can essentially consider DiSAC2 as a coding scheme that lays between distributed coding and centralized coding, because we use the passive broadcast link between the two encoders for coding. For this setup with lossless coding, Oohama [34] proved that the broadcast link does not improve the rate region over distributed coding. For lossy coding, distributed coding generally has a rate loss as compared to centralized coding [41]. However, it is still unknown today whether the broadcast link can increase rate-distortion performance in the lossy case. Nevertheless, Wagner [28] did not give an algorithm to achieve the rate-distortion region of the DSC2 for Gaussian case. In contrast, Theorem 3.1 states that DiSAC2, as a practical algorithm, achieves the exact rate-distortion region of the DSC2 for the Gaussian case. Moreover, this result holds in arbitrary successive refinement setups, which has not been investigated in previous literatures on multi-terminal source coding.

3.4 Distributed Successive Refinement of Stereo-View Images

Under the stationary assumption of Section 3.2.2, we extend the DiSAC2 scheme for bivariate Gaussian source (*DiSAC2-Gaussian*) to a practical image coding scheme, namely DiSAC2 for stereo-view images (*DiSAC2-Stereo*). With the help of broadcast, we combine layered decomposition and predictive coding to achieve successive refinement coding of stereo-view images. Then, we show simulation results of the proposed algorithm in two different stereo-view image datasets.

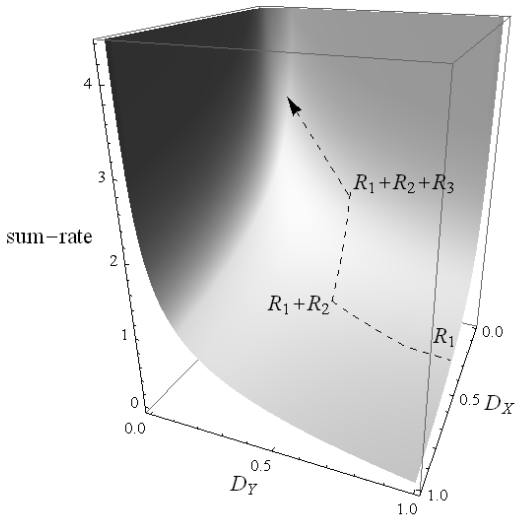


Figure 3.6: *DiSAC2-Gaussian is successively refinable: the {sum-rate, distortion pair} travels on the Wagner Surface. Three stages are illustrated, further stages follow a similar pattern.*

3.4.1 DiSAC2-Stereo

Like in the Gaussian case, we need to find the joint statistics between two sources to perform the distributed successive refinement. As introduced in Section 3.2.2, inter-view correlations are described by the point-wise correspondence between two images. This correspondence can be established via conventional image registration techniques [40]. At the initial stage, one camera first transmits a coarse version of its image to the BS; the other camera overhears the same image and applies view registration to the stereo-view images[3]. The estimated registration parameters are then transmitted to the BS (a few bytes) and therefore are known at both cameras thanks to overhearing. After the raw stereo-view images are properly registered and aligned with each other, we investigate the successive refinement coding of the two aligned images X and Y (same notations as in Section 3.3). In the coding process we consider only the overlapping parts of left and right views. As non-overlapping parts are normally not correlated, they are encoded independently using conventional methods.

Layered decomposition is the conventional method to achieve successive refinement on a single image: Layered bitstreams are transmitted in an incremental manner from low level layers to high level layers (see Figure 3.7a). By extending the idea of layered decomposition to an interleaved setup, we get an intuitive approach for the successive refinement of X and Y: As shown in Figure 3.7b, two images are encoded in ping-pong fashion as in the DiSAC2 scheme, and a new *incremental* layer is transmitted at each new stage. It is worth mentioning that, unlike the single image successive

[3]See Section 4.A for details on the image registration algorithm, that is robust under unequal image qualities between two images.

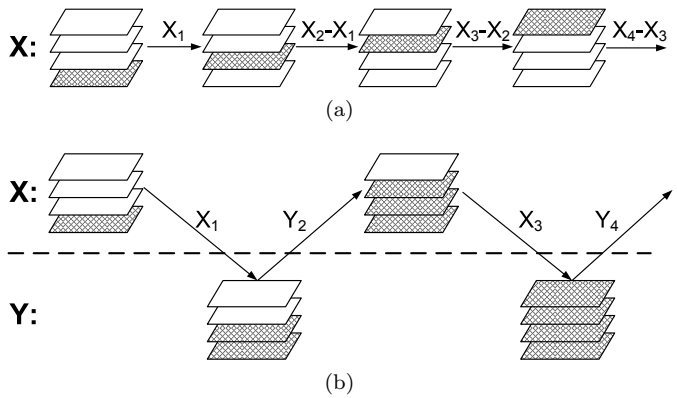


Figure 3.7: *Successive refinement using layered decomposition (4-stage setup). (a) Single image case: an incremental layer is transmitted at each stage. (b) Two correlated images case: incremental layer with lower layers are transmitted together at each stage. Shadowed area represents the transmitted layers.*

refinement (Figure 3.7a), the complete k lowest layers are transmitted at kth stage (see Figure 3.7b). This is because the lower layers between the left view and the right view can be slightly different due to registration error. The images transmitted at each stage are reconstructed at the BS, and we denote them by $X_1, Y_2, X_3, Y_4, \cdots$ (consistent with the notations in Section 3.3).

However, as X and Y are highly correlated, the lower layers of $X_1, Y_2, X_3, Y_4, \cdots$ are also correlated and therefore direct transmission is inefficient. Observe that the coding procedure of DiSAC2-Gaussian actually suggests a simple predictive coding algorithm to exploit correlations between refinement and previously sent codewords: a linear prediction of previous reconstructions can be used to eliminate the redundant information.

For a M-stage DiSAC2-Stereo scheme, at a even stage $i \leq M$ (similar for an odd stage), we need to encode Y and transmit it to the BS. Previous reconstructions $(X_1, Y_2, \cdots, Y_{i-2}, X_{i-1})$ are fully known to ENCy due to the previous broadcasts. Thus we seek to maximize the quality of reconstruction:

$$\max_{\alpha_k,\, k\in[1,i-1]} \text{PSNR}\,(Y_i, Y)\,, \tag{3.11}$$

where Y_i is the reconstruction of Y at the BS, and $\{\alpha_k\}$ are $i-1$ coefficients for predictive coding.

The predictive coding procedure is as follows:

1. Y is decomposed into M layers with increasing quality. By omitting the highest $M-i$ layers, we obtain a coarse version of Y with the lowest i layers, which is denoted as Y_i'.

2. Compute the residual using linear prediction of previous reconstructions:

$$C_i' = Y_i' - (\alpha_1 \cdot X_1 + \alpha_2 \cdot Y_2 + \cdots + \alpha_{i-1} \cdot X_{i-1}).$$

Algorithm 3.1 M-stage *DiSAC2-Stereo*

for stage $i = 1 \to M$ **do**
  if $i == 1$ (the first stage) **then**
    ENCx directly sends the base layer of the left view to the BS.
    Meanwhile, ENCy overhears the left view and applies view registration algorithm.
    ENCy transmits the estimated registration parameters to the BS.
    ENCx overhears the registration parameters.
    Using the registration parameters, two aligned images X and Y for the left camera and the right camera are generated.
  else
    if i is even **then**
      ENCy solve $\max_{\alpha_i,\, i\in[1,i-1]} \text{PSNR}\,(Y_i, Y)$, given a rate R_i.
      Transmit coefficients $\{\alpha_i\}$ and the codeword C_i to the BS.
      ENCx overhears the same message and obtains Y_i.
    end if
    if i is odd **then**
      ENCx solve $\max_{\alpha_i,\, i\in[1,i-1]} \text{PSNR}\,(X_i, X)$, given a rate R_i.
      Transmit coefficients $\{\alpha_i\}$ and the codeword C_i to the BS.
      ENCy overhears the same message and obtains X_i.
    end if
  end if
end for

3. Encode the residual C_i' with a desired rate R_i. C_i is the corresponding decoded reconstruction, and Y_i is obtained by:

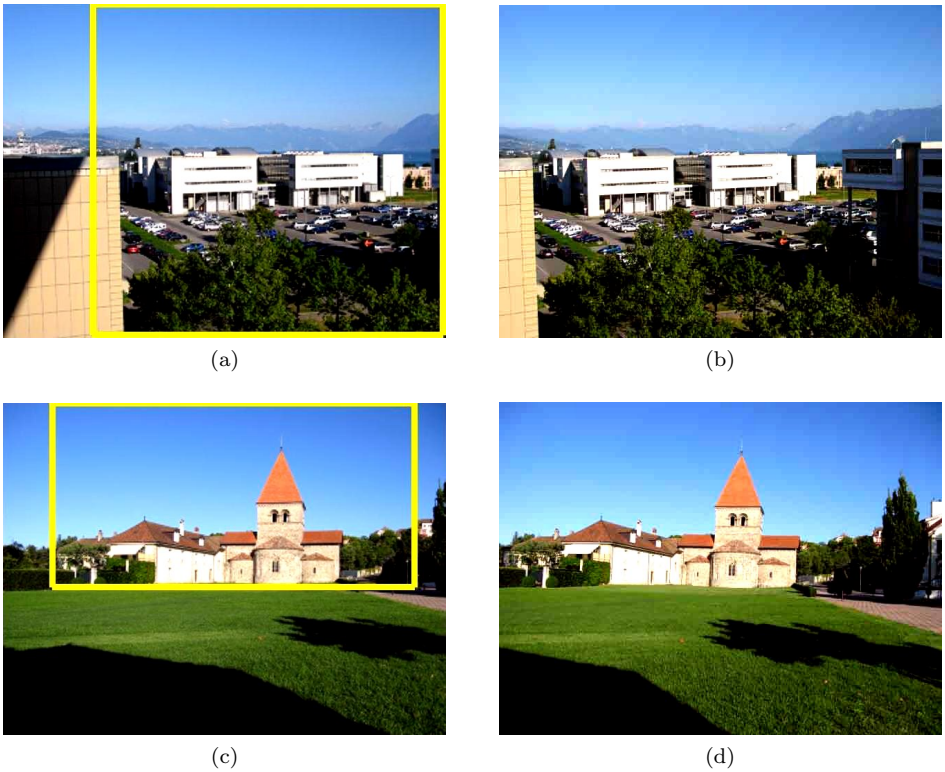
$$Y_i = C_i + (\alpha_1 \cdot X_1 + \alpha_2 \cdot Y_2 + \cdots + \alpha_{i-1} \cdot X_{i-1}).$$

Note that in DiSAC2-Gaussian, the correlation ρ between X and Y has to be sent to the BS for decoding. In more general cases where the joint statistics between two sources cannot be explicitly expressed, we can directly estimate and transmit the linear prediction coefficients of the encoders to the BS for decoding, which just requires a few bytes. Based on the layered decomposition and the predictive coding above, we depict the entire algorithm of a M-stage DiSAC2-Stereo scheme in Algorithm 3.1.

3.4.2 Simulations

We analyze now the performance of the proposed coding method *DiSAC2-Stereo* through simulations. In particular, we study a three-stage *DiSAC2-Stereo* with two stereo-view datasets (Figure 3.8): "Park", captured by a rotating camera, is suitable for image registration, and "Church", captured by two far apart cameras, has complex depth structure and is more prone to registration error.

For all the simulations in this section, *DiSAC2-Stereo* is evaluated with rate triplet (R_1, R_2, R_3) in the range $\{R_1 \in (0.01, 0.05), R_2 \in (0.01, 0.15), R_3 \in (0.01, 0.1)\}$,



(a) (b)

(c) (d)

Figure 3.8: *The stereo-view datasets "Park" and "Church". The yellow box in the left view shows the overlapping part of left and right views detected by the registration algorithm. (a) The left view of "Park". (b) The right view of "Park". (c) The left view of "Church". (d) The right view of "Church".*

among which 245 uniformly distributed grid samples are chosen to reduce the computation burden.

Prediction Coefficient Searching

According to (3.11), for a three-stage *DiSAC2-Stereo*, there are one coefficient α_1, and two coefficients α_1', α_2' that need to be optimized for the 2nd and 3rd stage coding respectively:

$$\begin{cases} \max\limits_{\alpha_1} \ \mathrm{PSNR}\left(Y_2, Y\right) \\ \max\limits_{\alpha_1', \alpha_2'} \ \mathrm{PSNR}\left(X_3, X\right) \end{cases}.$$

Such optimizations can be done with extensive grid searching. To speed up the searching process, we use the following strategies:

1. The second optimization problem is simplified to a 1D searching with the constraint $\alpha_1' + \alpha_2' = 1$.

Table 3.1: *Number of iterations to converge in SQP solver (stop condition: change in the objective PSNR value was less than 0.01).*

Dataset name	Stage	Mean [step]	Standard deviation [step]
`Park`	2nd	2.5	0.7
	3rd	2.5	0.8
`Church`	2nd	3.1	0.8
	3rd	2.6	0.9

2. First use linear searching with a very sparse grid to find a good initial point.

3. Starting from the chosen point, we obtain the optimal coefficient by using a typical sequential quadratic programming (SQP) solver [42].

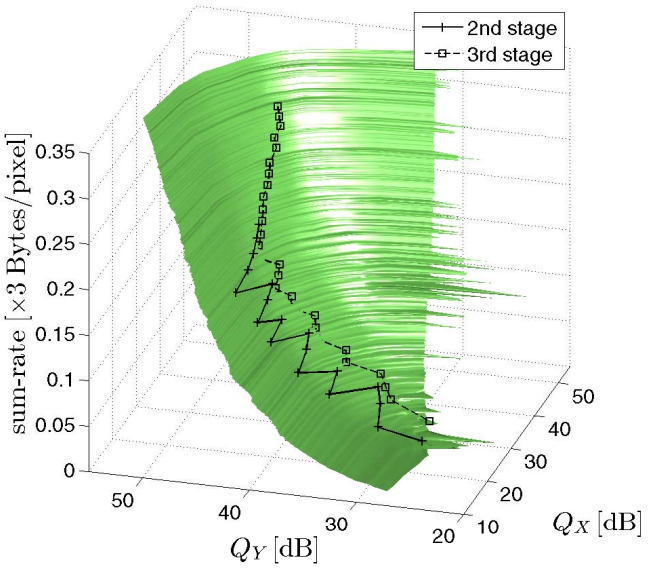
By comparing with the optimal coefficients obtained by grid searching, our method is shown to be efficient and accurate in practice. Table 3.1 shows that the SQP solver usually converges within 3 steps.

Successive Refinement

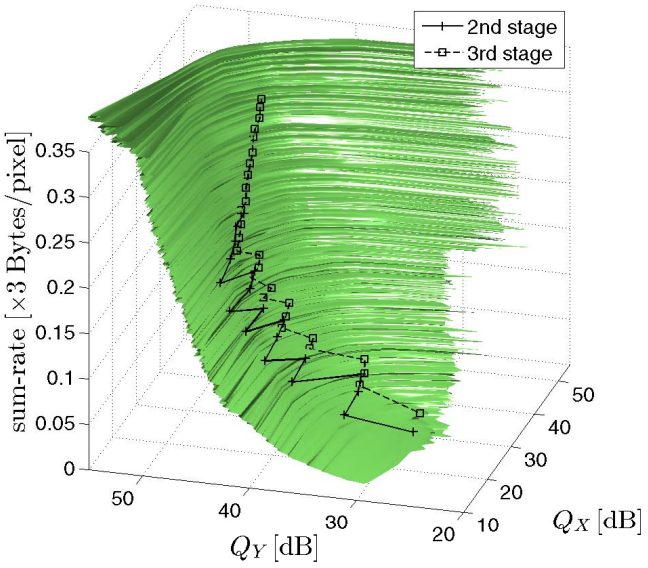
Similar to Figure 3.6 for *DiSAC2-Gaussian*, we verify now how close the {*sum-rate, distortion pair*} achieved by *DiSAC2-Stereo* is to the performance limit of distributed stereo-view coding. To the best of our knowledge, currently there is no distributed stereo-view image coder available. As an alternative, we use conventional prediction-based centralized image coding to approximate the distributed coding limit: By enumerating rate combinations of two encoders, we can obtain the distortion pairs achieved using centralized image coding. The green surfaces in Figure 3.9 show the operating points over all possible combinations, which represent the performance limits of centralized coding.

The two curves in Figure 3.9a (same for Figure 3.9b) represent the operating points of a three-stage *DiSAC2-Stereo* at the 2nd stage and the 3rd stage, respectively. It can be seen that this scheme does perform close to the centralized coding limit for the two datasets. To give a better illustration of coding losses, we measure the PSNR loss of *DiSAC2-Stereo* with respect to the performance limit of centralized coding, both using the same rate triplet (R_1, R_2, R_3). With the simulation results of 245 rate triplets as mentioned in the experimental setup, Figure 3.10 shows the distribution (PDF) of coding losses at the 2nd stage and the 3rd stage, respectively. We can see that the losses are mostly distributed within the range of 3dB: The maximum likelihood values of PSNR losses are 0.6dB/0.8dB at the 2nd/3rd stages for the "Park", and 1.2dB/1.4dB at the 2nd/3rd stages for the "Church". There are several reasons to explain these losses:

1. It is well known that a distributed setup has coding loss with respect to a centralized setup [41]. As a result, the centralized coding that we use provides a lower bound for the performance of distributed coding, and thus the actual coding loss of *DiSAC2-Stereo* will be smaller.



(a) "Park"



(b) "Church"

Figure 3.9: *The surface plotted in* $(Q_X, Q_Y, \text{sum-rate})$ *domain shows the performance limit of distributed stereo-view coding approximated by centralized coding.* Q_X, Q_Y *denote the reconstruction quality of stereo images* X, Y*, in terms of PSNR. The curves with markers show the operating points of a three-stage DiSAC2-Stereo at the 2nd stage and the 3rd stage respectively. For clarity, not all samples are plotted in each curve. The plots for "Park" and "Church" are shown in part (a) and (b), respectively.*

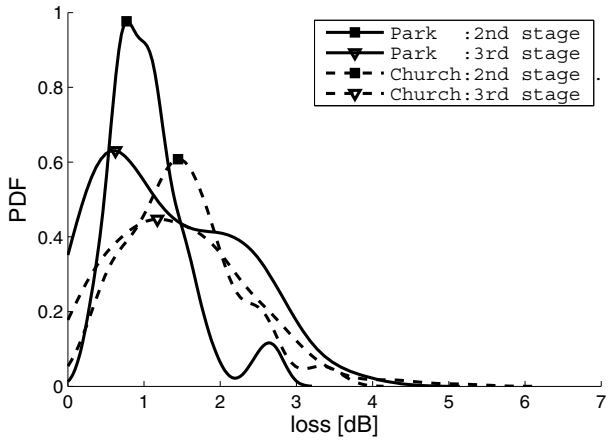


Figure 3.10: *Probability density (PDF) of PSNR losses of DiSAC2-Stereo as compared to the performance limit of centralized coding, at the 2nd stage and the 3rd stage for two datasets respectively. The distribution is obtained by measuring 245 different rate triplet combinations. The density curves are smoothed by Gaussian kernel and normalized. The marker on each curve represents the maximum likelihood value of PSNR loss.*

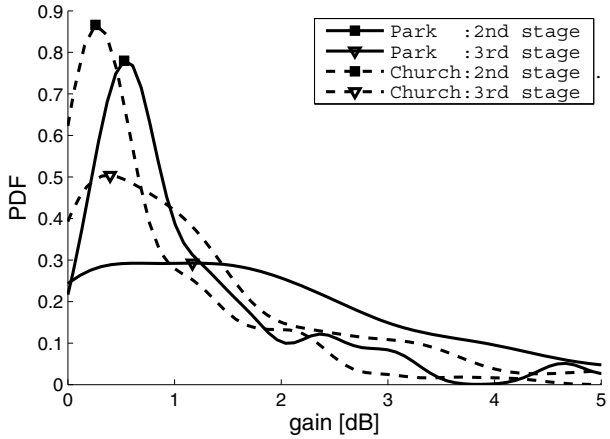


Figure 3.11: *Probability density (PDF) of coding gains of DiSAC2-Stereo as compared to independent intra-coding where the broadcast is not utilized, at the 2nd stage and the 3rd stage for two datasets, respectively. The distribution is obtained by measuring 245 different rate triplet combinations. The density curves are smoothed by Gaussian kernel and normalized. The marker on each curve represents the maximum likelihood value of coding gains.*

2. The successive refinability does not generally hold for any kind of sources. The quadratic Gaussian condition of Theorem 3.1 does not necessarily apply to natural images.

3. The error of stereo-view registration can degrade the correlation model of Section 3.2.2. Particularly, we can see from Figure 3.10 that "Park" performs better than "Church" because it is taken by a fixed rotating camera and has smaller registration errors.

To illustrate the benefits of exploiting inter-view correlations by using the broadcast link, we compare *DiSAC2-Stereo* to an independent intra-coding scheme. If the broadcast is not utilized, each camera just applies conventional single image successive refinement coding to its own image. In contrast, *DiSAC2-Stereo* takes advantage of the broadcast nature to exploit interview correlation. Figure 3.11 shows the distribution (PDF) of coding gains (in terms of PSNR) of *DiSAC2-Stereo* with respect to the independent intra-coding scheme, at the 2nd stage and the 3rd stage, respectively. We can see that the coding gains are distributed over a wide range of 5dB, especially at the 3rd stages. Again, "Park" performs better than "Church" due to its smaller registration error.

3.5 Distributed Uncoded Transmission of Bivariate Gaussian Sources

We showed in Section 3.3.2 that DiSAC2-Gaussian provides a practical coding algorithm to achieve the theoretical distributed coding limit. As an alternative to traditional digital communication schemes, Gastpar [43] discussed a communication strategy called *uncoded transmission*, i.e., the encoder merely transmits a scale version of the source signal without any digital modulations. This analog communication scheme is simple to implement, and is proved to be as efficient as any digital communication schemes for transmitting a single Gaussian source. We investigate in this section whether, for bivariate Gaussian sources with a broadcast link, uncoded transmission can also achieve the theoretical distributed coding limit.

3.5.1 Uncoded Transmission Revisited

Let us revisit the well-known example [43, P.43], where the uncoded transmission is optimal with respect to the traditional digital communication setup.

Figure 3.12 shows the uncoded transmission of a single Gaussian source. X is a discrete-time source with an independent and identically distributed (i.i.d.) Gaussian distribution $\mathcal{N}(0, \sigma_X^2)$. X is mapped to S by a linear coefficient α, and sent across an additive white Gaussian noise (AWGN) channel. $\hat{S}$ is what we receive at the destination, $\hat{S} = S + Z$, where Z is a white Gaussian noise of variance σ_Z^2. We recover $\hat{X}$ using linear estimation $\hat{X} = \beta\hat{S}$. Given the constraint of transmission power $\mathbb{E}S^2 \leq P$, there exists an optimal decoding setup that minimizes the expectation of

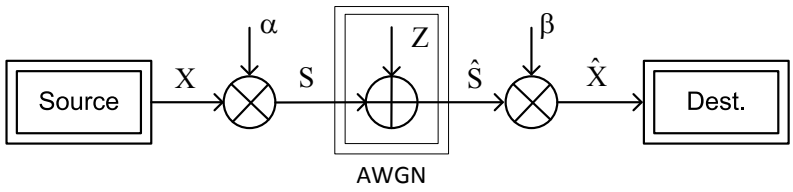


Figure 3.12: *Uncoded transmission of an i.i.d. Gaussian source across an AWGN channel.*

reconstruction error:

$$\begin{aligned} D &= \min \mathbb{E}|X - \hat{X}|^2 \\ &= \min \mathbb{E}|X - \beta \cdot (\alpha X + Z)|^2, \end{aligned}$$

which gives

$$\alpha = \sqrt{\frac{P}{\sigma_X^2}}, \quad \beta = \sqrt{\frac{\sigma_X^2}{P}} \frac{\sqrt{P}}{P + \sigma_Z^2},$$

and

$$D = \frac{\sigma_X^2 \sigma_Z^2}{P + \sigma_Z^2}. \tag{3.12}$$

It can be verified that D is equal to the mean-squared distortion achieved with a separation-based digital coding setup [39]. This means that the uncoded transmission of a single source is optimal in the sense that it performs exactly the same as a traditional digital communication scheme, despite its extremely simple coding procedure.

3.5.2 Distributed Uncoded Transmission Using Broadcast Link

Motivated by DiSAC2, we investigate the uncoded transmission of two distributed sources X, Y, by using the broadcast link between the two encoders. We model X, Y as bivariate Gaussian sources $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where

$$\boldsymbol{\mu} = (0, 0), \quad \boldsymbol{\Sigma} = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \text{ for } |\rho| < 1.$$

As shown in Figure 3.13, two sources are sent by linear mappings in two successive transmissions. In the first time slot, source X is mapped to S_1 with power constraint P_1, and S_1 is sent across an AWGN channel (Z_1 of variance σ_1^2). Meanwhile, S_1 is broadcasted across an AWGN channel (Z_b of variance σ_b^2) towards source Y. $\hat{S}_1, S_b$ are the signals received at the destination and the source Y, respectively.

In the second time slot, we calculate $\Delta_S = Y - \alpha_b S_b$, and map Δ_S to S_2 with power constraint P_2, which is sent across an AWGN channel (Z_2 of variance σ_2^2). $\hat{S}_2$ is the signal received at the destination. The noises Z_1, Z_2, Z_b are assumed to be uncorrelated with each other. Note that $\sigma_b^2 \leq \sigma_1^2$ and $\sigma_b^2 \leq \sigma_2^2$, because the capacity between the neighboring cameras is assumed to be larger than the capacity between the camera and the BS (see Section 3.2.1).

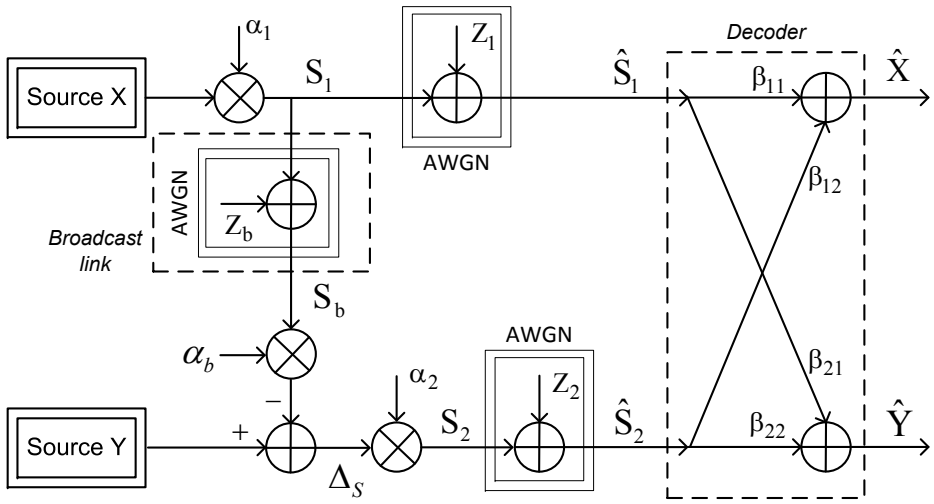


Figure 3.13: *Uncoded transmission of two distributed bivariate Gaussian sources across AWGN channels, with the help of broadcast link. X, Y are sent by two successive transmissions (no interference).*

At the encoder side, we have to setup three parameters $\alpha_1, \alpha_2, \alpha_b$, among which α_1, α_2 are determined by the maximum power constraints P_1, P_2. In the following, we first introduce the optimal decoding procedure, and then discuss the choice of α_b.

Optimal Decoding: At the destination, we recover $\hat{X}, \hat{Y}$ by linear estimation of $\hat{S}_1, \hat{S}_2$:

$$\begin{bmatrix} \hat{X} \\ \hat{Y} \end{bmatrix} = \begin{bmatrix} \beta_{11} & \beta_{12} \\ \beta_{21} & \beta_{22} \end{bmatrix} \cdot \begin{bmatrix} \hat{S}_1 \\ \hat{S}_2 \end{bmatrix}. \tag{3.13}$$

The distortion pair for (X, Y) is defined as

$$\begin{cases} D_X = \mathbb{E}|X - \hat{X}|^2 \\ D_Y = \mathbb{E}|Y - \hat{Y}|^2 \end{cases}.$$

With the power constraint (P_1, P_2), we minimize (D_X, D_Y) to find the optimal $\{\beta_{ij}\}$. From Figure 3.13,

$$\begin{cases} \hat{S}_1 = \alpha_1 X + Z_1 \\ \Delta_S = Y - \alpha_b \alpha_1 X - \alpha_b Z_b \\ \hat{S}_2 = \alpha_2 \Delta_S + Z_2 \end{cases}.$$

Due to the power constraint (P_1, P_2),

$$\begin{cases} \alpha_1 = \sqrt{P_1} \\ \alpha_2 = \sqrt{\dfrac{P_2}{\mathbb{E}|\Delta_S|^2}} \end{cases}, \tag{3.14}$$

where

$$\mathbb{E}|\Delta_S|^2 = \alpha_b^2(P_1 + \sigma_b^2) - 2\alpha_b\rho\sqrt{P_1} + 1. \tag{3.15}$$

By minimizing

$$\begin{cases} D_X = \mathbb{E}|\beta_{11}\hat{S}_1 + \beta_{12}\hat{S}_2 - X|^2 \\ D_Y = \mathbb{E}|\beta_{21}\hat{S}_1 + \beta_{22}\hat{S}_2 - Y|^2 \end{cases},$$

the optimal decoding coefficients $\{\beta_{ij}\}$ can be found by the linear minimum mean-squared error (LMMSE) estimator [44]:

$$\begin{bmatrix} \mathbb{E}|\hat{S}_1|^2 & \mathbb{E}(\hat{S}_1\hat{S}_2) \\ \mathbb{E}(\hat{S}_1\hat{S}_2) & \mathbb{E}|\hat{S}_2|^2 \end{bmatrix} \cdot \begin{bmatrix} \beta_{11} & \beta_{21} \\ \beta_{12} & \beta_{22} \end{bmatrix} = \begin{bmatrix} \mathbb{E}(\hat{S}_1X) & \mathbb{E}(\hat{S}_1Y) \\ \mathbb{E}(\hat{S}_2X) & \mathbb{E}(\hat{S}_2Y) \end{bmatrix}.$$

The corresponding

$$\begin{cases} D_X = \mathbb{E}|X|^2 - \begin{bmatrix} \mathbb{E}(\hat{S}_1X) \\ \mathbb{E}(\hat{S}_2X) \end{bmatrix}^T \cdot \begin{bmatrix} \mathbb{E}|\hat{S}_1|^2 & \mathbb{E}(\hat{S}_1\hat{S}_2) \\ \mathbb{E}(\hat{S}_1\hat{S}_2) & \mathbb{E}|\hat{S}_2|^2 \end{bmatrix}^{-1} \cdot \begin{bmatrix} \mathbb{E}(\hat{S}_1X) \\ \mathbb{E}(\hat{S}_2X) \end{bmatrix} \\ D_Y = \mathbb{E}|Y|^2 - \begin{bmatrix} \mathbb{E}(\hat{S}_1Y) \\ \mathbb{E}(\hat{S}_2Y) \end{bmatrix}^T \cdot \begin{bmatrix} \mathbb{E}|\hat{S}_1|^2 & \mathbb{E}(\hat{S}_1\hat{S}_2) \\ \mathbb{E}(\hat{S}_1\hat{S}_2) & \mathbb{E}|\hat{S}_2|^2 \end{bmatrix}^{-1} \cdot \begin{bmatrix} \mathbb{E}(\hat{S}_1Y) \\ \mathbb{E}(\hat{S}_2Y) \end{bmatrix} \end{cases}, \tag{3.16}$$

where

$$\begin{cases} \mathbb{E}|\hat{S}_1|^2 = P_1 + \sigma_1^2 \\ \mathbb{E}|\hat{S}_2|^2 = P_2 + \sigma_2^2 \\ \mathbb{E}(\hat{S}_1\hat{S}_2) = \alpha_1\alpha_2(\rho - \alpha_b\alpha_1) \\ \mathbb{E}(\hat{S}_1X) = \alpha_1 \\ \mathbb{E}(\hat{S}_2X) = \alpha_2(\rho - \alpha_b\alpha_1) \\ \mathbb{E}(\hat{S}_1Y) = \rho\alpha_1 \\ \mathbb{E}(\hat{S}_2Y) = \alpha_2(1 - \rho\alpha_b\alpha_1) \end{cases}, \tag{3.17}$$

and α_1, α_2 follow from (3.14) (3.15).

Choice of α_b: Δ_S can be thought as the residual of predictive coding, which is modulated to S_2 for transmission. Thus, one strategy is to choose α_b that minimizes the residual energy $\mathbb{E}|\Delta_S|^2$. From (3.15), it is easy to find that

$$\alpha_b = \frac{\rho\sqrt{P_1}}{P_1 + \sigma_b^2}. \tag{3.18}$$

Another choice is to cut the link:

$$\alpha_b = 0, \tag{3.19}$$

which means the broadcast link is not used. We will compare these two choices in the next section.

3.5.3 Comparison with the Digital Scheme

We consider the symmetric channel case where $\sigma_1^2 = \sigma_2^2$, and compare the performance of the uncoded transmission scheme proposed in Section 3.5.2 with the separation-based digital scheme [39].

To transmit two distributed bivariate Gaussian sources, the traditional separation-based digital scheme involves two steps: i) source coding and ii) channel coding. For source coding, the rate-distortion region of this setup is known [28]. For channel coding, as the transmissions of two sources have no interference with each other, the capacity of each AWGN channel is known from [24, P.249]:

$$C = \frac{1}{2}\log_2(1 + \frac{P}{\sigma^2}),$$

where P is the power constraint, and σ^2 is the noise variance. In the extreme case where the source coding rate meets the channel capacity, the overall power for the digital scheme is

$$\begin{aligned} P_{\text{digital}} &= \sigma_1^2\left(2^{2R_1} + 2^{2R_2} - 2\right) \\ &\geq 2\sigma_1^2\left(\sqrt{\frac{(1-\rho^2)\left(\sqrt{\frac{4D'_X D'_Y \rho^2}{(1-\rho^2)^2}+1}+1\right)}{2D'_X D'_Y}} - 1\right), \end{aligned} \tag{3.20}$$

where the second inequality follows from the fact that $2^{2R_1} + 2^{2R_2} \geq 2 \cdot 2^{R_1+R_2}$, and $R_1 + R_2$ is constrained by the minimum sum-rate (3.1). Note that the lower bound of (3.20) is achieved when $D'_X = D'_Y$.

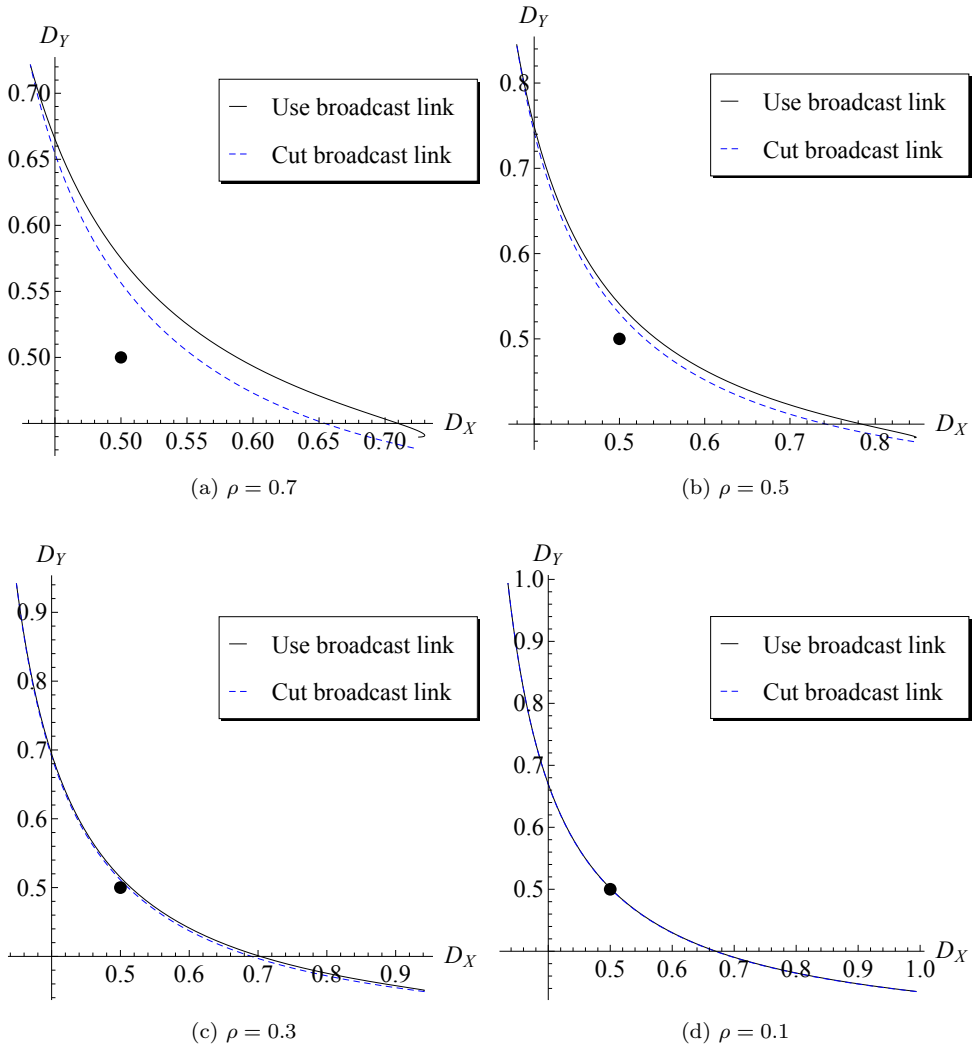
To make the comparison, we fix the distortion pair of the digital scheme as $D'_X = D'_Y = 0.5$. The corresponding power P_{digital} is hence fixed and used as a power constraint for the uncoded transmission scheme: $P_1 + P_2 = P_{\text{digital}}$. Then, from (3.16) and (3.17), the achievable distortion pair (D_X, D_Y) of the uncoded transmission scheme can be calculated.

Figure 3.14 shows the comparison results for different source correlation ρ. There are several interesting observations here:

1. As a practical algorithm, the uncoded transmission scheme (Figure 3.13) cannot achieve the theoretical limit (3.20) that is defined by traditional information theory. In contrast, as shown in Section 3.3.2, its digital counterpart — DiSAC2-Gaussian achieves this limit.

2. When the correlation ρ between two sources decreases, the uncoded transmission scheme operates closer and closer to the digital scheme. In the extreme case, we are essentially transmitting two independent sources. Therefore, according to the optimality in Section 3.5.1, the uncoded transmission scheme performs exactly the same as the digital scheme.

3. Surprisingly, the uncoded transmission scheme suffers from the broadcast link between two encoders. When ρ decreases, the performance loss of using broadcast link also decreases. As a matter of fact, in the analog communication setup, the broadcast link adds a noise to the signal and makes it impossible for the



(a) $\rho = 0.7$ (b) $\rho = 0.5$

(c) $\rho = 0.3$ (d) $\rho = 0.1$

Figure 3.14: *Comparison results between the uncoded transmission scheme and the digital scheme. The noise variances* $\sigma_1 = \sigma_2 = 0.1$, *and* $\sigma_b = 0.03$. *The black dot at (0.5, 0.5) is the distortion pair of the digital scheme. With the same overall power constraint, the two curves are the achievable distortion pairs of the uncoded transmission scheme. The solid curve is the result obtained by minimizing the residual energy (3.18), and the dashed curve is the result without using the broadcast link (3.19).*

overhearing node to have a correct broadcasted message. In contrast, as long as the channel capacity is met, digital communication is resilient to channel noises, and thus is able to exploit the broadcasted message for coding.

3.6 Summary

- Distributed successive refinement coding (DiSAC2) is a novel scheme that combines distributed source coding and successive refinement coding, designed for multi-view images captured by distributed cameras.
- DiSAC2 has a coding procedure similar to a ping-pong game, and shares the same idea as the cooperative sampling framework in Chapter 2.
- By exploiting the broadcast link, DiSAC2 provides a simple distributed coding algorithm that achieves the exact theoretical distributed coding limit in the Gaussian case.
- With a similar coding procedure, DiSAC2 is able to encode distributed multi-view images, based on the layered decomposition and linear prediction methods.
- Unlike the single encoder case, the uncoded transmission scheme with broadcast link does not match its digital counterpart in distributed scenarios.
- Although we restrict the discussions to two-camera case in this chapter, the same idea can be readily extended to more cameras. One straightforward extension is to create a "multiple two-camera" system so that any part of the overall scenery is covered by at least one pair of cameras. In this case, to avoid the interferences among multiple cameras, we can use the adaptive synchronization algorithm proposed in Section 2.4 to properly coordinate the cameras.

3.A Proof of Theorem 3.1

Lemma 3.2. *For a M-stage DiSAC2 with Gaussian sources and quadratic distortion*[4]*, the conditional probability at ith stage is Gaussian distributed: $p(\bar{Z}|C_1\cdots C_i) = \mathcal{G}_{\mu_i,\sigma_i^2}(\bar{Z})$, where*

$$Z = \begin{cases} X & i \text{ is odd} \\ Y & i \text{ is even} \end{cases}, \quad \text{and } \bar{Z} \text{ vice versa.} \tag{3.21}$$

μ_i and σ_i^2 can be calculated in a recursive way from $G_i(\bar{Z})$ in the following:

$$\begin{cases} G_i(\bar{Z}) = \int_{-\infty}^{+\infty} H_i(X,Y)\ dZ \\ H_i(X,Y) = \dfrac{H_{i-1}(X,Y)}{G_{i-1}(Z)} \cdot \mathcal{G}_{Z_i,D_{Z_i}}(Z) \end{cases}, \tag{3.22}$$

where $H_i(X,Y)$ denotes $p(XY|C_1\cdots C_i)$, $G_i(\bar{Z})$ denotes $p(\bar{Z}|C_1\cdots C_i)$, and the initial condition is

$$H_0(X,Y) = \frac{1}{2\pi\sqrt{1-\rho^2}} e^{-\frac{X^2+Y^2-2\rho XY}{2(1-\rho^2)}}. \tag{3.23}$$

Furthermore, $H_i(X,Y)$ can be expressed as $K_i \cdot e^{-h_i(X,Y)}$, where $h_i(X,Y)$ is a polynomial of (X,Y) of degree 2, and K_i is a coefficient that keep $H_i(X,Y)$ normalized. The exponent $h_i(X,Y)$ can be calculated recursively by:

$$h_i(X,Y) = h_{i-1}(X,Y) - \frac{(Z-\mu_{i-1})^2}{2\sigma_{i-1}^2} + \frac{(Z-Z_i)^2}{2D_{Z_i}}. \tag{3.24}$$

Proof. At the initial stage $i = 0$, it is immediate to verify that $H_0(X,Y)$ in (3.23) is consistent with $p(XY)$ and $h_0(X,Y)$ is a polynomial of (X,Y) of degree 2. From (3.22), $G_0(X) = \int_{-\infty}^{+\infty} H_0(X,Y)\ \mathrm{d}Y = \frac{1}{2\pi} e^{-\frac{X^2}{2}}$, which is also consistent with $p(X)$. Thus, Lemma 3.2 holds when $i = 0$.

By induction, we first assume that Lemma 3.2 holds when $i = 2k,\ k \geq 0$, from which we know that

$$\begin{cases} G_{2k}(X) = p(X|C_1\cdots C_{2k}) = \mathcal{G}_{\mu_{2k},\sigma_{2k}^2}(X) \\ H_{2k}(X,Y) = p(XY|C_1\cdots C_{2k}) = K_{2k}\cdot e^{-h_{2k}(X,Y)} \\ h_{2k}(X,Y) \text{ is a polynomial of } (X,Y) \text{ with the degree of 2} \end{cases} \tag{3.25}$$

When $i = 2k+1$, the coding procedure is as follows:

$(C_1,\cdots,C_{2k})$ are already known due to the broadcast advantage. Thus according to (3.25), $p(X-\mu_{2k}) = \mathcal{G}_{0,\sigma_{2k}^2}(X)$. $X-\mu_{2k}$ is then encoded to C_{2k+1} using random codebook argument, with R-D pair

$$R_{2k+1} = \frac{1}{2}\log\frac{\sigma_{2k}^2}{D_{X_{2k+1}}}, \quad D_{X_{2k+1}} \leq \sigma_{2k}^2.$$

[4] Notations inherited from Section 3.3.

X_{2k+1} can be decoded as: $X_{2k+1} = C_{2k+1} + \mu_{2k}$.

As $p(X - \mu_{2k}|C_1 \cdots C_{2k}) = \mathcal{G}_{0,\sigma_{2k}^2}(X)$, the *test channel* in this stage is

$$p(X - \mu_{2k}|C_1 \cdots C_{2k}C_{2k+1}) = \mathcal{G}_{C_{2k+1},D_{X_{2k+1}}}(X).$$

Thus

$$p(X|C_1 \cdots C_{2k+1}) = \mathcal{G}_{C_{2k+1}+\mu_{2k},D_{X_{2k+1}}}(X) = \mathcal{G}_{X_{2k+1},D_{X_{2k+1}}}(X). \tag{3.26}$$

To decode Y_{2k+1}, the conditional probability

$$p(Y|C_1 \cdots C_{2k+1}) = \int_{-\infty}^{+\infty} \frac{p(XY|C_1 \cdots C_{2k})}{p(X|C_1 \cdots C_{2k})} \cdot p(X|C_1 \cdots C_{2k+1})\ \mathrm{d}X,$$

which follows from a similar argument as (3.3) and the fact that C_{2k+1} is encoded and decoded from $(X, C_1, \cdots, C_{2k})$. To calculate the integral, $p(XY|C_1 \cdots C_{2k})$ and $p(X|C_1 \cdots C_{2k})$ are already known from (3.25) and $p(X|C_1 \cdots C_{2k+1}) = \mathcal{G}_{X_{2k+1},D_{X_{2k+1}}}(X)$ due to (3.26). Therefore,

$$\begin{cases} H_{2k+1}(X,Y) = p(XY|C_1 \cdots C_{2k+1}) = \dfrac{H_{2k}(X,Y)}{G_{2k}(X)} \cdot \mathcal{G}_{X_{2k+1},D_{X_{2k+1}}}(X) \\ G_{2k+1}(Y) = p(Y|C_1 \cdots C_{2k+1}) = \displaystyle\int_{-\infty}^{+\infty} H_{2k+1}(X,Y)\ \mathrm{d}X \end{cases}. \tag{3.27}$$

From (3.25) and (3.27),

$$H_{2k+1}(X,Y) = \frac{K_{2k}\sigma_{2k}^2}{D_{X_{2k+1}}} \cdot e^{-\left(h_{2k}(X,Y) - \frac{(X-\mu_{2k})^2}{2\sigma_{2k}^2} + \frac{(X-X_{2k+1})^2}{2D_{X_{2k+1}}}\right)}.$$

Thus, $H_{2k+1}(X,Y)$ can be expressed as $K_{2k+1} \cdot e^{-h_{2k+1}(X,Y)}$, where

$$h_{2k+1}(X,Y) = h_{2k}(X,Y) - \frac{(X-\mu_{2k})^2}{2\sigma_{2k}^2} + \frac{(X-X_{2k+1})^2}{2D_{X_{2k+1}}}. \tag{3.28}$$

Finally, from (3.27) and the fact that $h_{2k+1}(X,Y)$ is a polynomial of (X,Y) with the degree of 2, it is immediate to verify that $p(Y|C_1 \cdots C_{2k+1}) = \int_{-\infty}^{+\infty} K_{2k+1} \cdot e^{-h_{2k+1}(X,Y)}\ \mathrm{d}X$ is Gaussian distributed, and can be denoted as

$$\mathcal{G}_{\mu_{2k+1},\sigma_{2k+1}^2}(Y). \tag{3.29}$$

In conclusion, from (3.27), (3.28), and (3.29), it is proven that Lemma 3.2 holds when $i = 2k+1$. Similar arguments can be used to prove the induction from stage $2k+1$ to stage $2k+2$. Therefore, Lemma 3.2 is proved.

□

Given the recursive calculation process for σ_i^2 as Lemma 3.2, we get the recursive expression of σ_i^2 as follows.

Lemma 3.3. *For a M-stage DiSAC2 with Gaussian sources and quadratic distortion, σ_i^2 at each stage ($i \geq 2$) can be expressed as:*

$$\frac{1}{2\sigma_i^2} = a_{i-1} - \frac{\rho^2}{\frac{\rho^2}{a_{i-1}} + \frac{2(1-\rho^2)^2}{D_{Z_i}}}, \tag{3.30}$$

where $a_i > 0$ and can be computed recursively by

$$a_i = \frac{\rho^2}{4a_{i-1}(1-\rho^2)^2} + \frac{1}{2D_{Z_i}}. \tag{3.31}$$

The notation for Z is the same as (3.21).

The initial condition is given by

$$\begin{cases} a_1 = \dfrac{1}{2(1-\rho^2)} - \dfrac{1}{2} + \dfrac{1}{2D_{X_1}} \\ \dfrac{1}{2\sigma_1^2} = \dfrac{1}{2(1-\rho^2)} - \dfrac{\rho^2}{4a_1(1-\rho^2)^2} \end{cases}. \tag{3.32}$$

Proof. From Lemma 3.2, we know that $h_i(X,Y)$ is a polynomial of (X,Y) of degree 2. Thus, we rewrite

$$h_1(X,Y) = a_1(X - b_1Y - c_1)^2 + p_1(Y - q_1)^2 + s_1, \tag{3.33}$$

where $a_1, b_1, c_1, p_1, q_1, s_1$ are real-valued coefficients.

From (3.22):

$$\begin{aligned} \mathcal{G}_{\mu_1,\sigma_1^2}(Y) = G_1(Y) &= \int_{-\infty}^{+\infty} K_1 \cdot e^{-\left(a_1(X-b_1Y-c_1)^2 + p_1(Y-q_1)^2 + s_1\right)} \, \mathrm{d}X \\ &= K_1' \cdot e^{-p_1(Y-q_1)^2}, \end{aligned} \tag{3.34}$$

By checking the definition of $\mathcal{G}_{\mu_1,\sigma_1^2}(Y)$, $p_1 = \frac{1}{2\sigma_1^2}, q_1 = \mu_1$, K_1' is the normalized coefficient, and $a_1 > 0$ (otherwise (3.34) not integrable).

From (3.23), (3.24) and $\mu_0 = 0, \sigma_0^2 = 1$, we know that

$$h_1(X,Y) = \frac{X^2 + Y^2 - 2\rho XY}{2(1-\rho^2)} - \frac{X^2}{2} + \frac{(X - X_1)^2}{2D_{X_1}}. \tag{3.35}$$

By comparing the coefficients of terms X^2, XY, Y^2 in (3.33) and (3.35),

$$\begin{cases} a_1 = \dfrac{1}{2(1-\rho^2)} - \dfrac{1}{2} + \dfrac{1}{2D_{X_1}} \\ a_1 b_1 = \dfrac{\rho}{2(1-\rho^2)} \\ \dfrac{1}{2\sigma_1^2} = p_1 = \dfrac{1}{2(1-\rho^2)} - \dfrac{\rho^2}{4a_1(1-\rho^2)^2} \end{cases},$$

which proves the initial condition (3.32). Using induction, we first assume that Lemma 3.3 holds when $i = 2k-1,\ k \geq 1$, from which we know that

$$\begin{cases} h_{2k-1}(X,Y) = a_{2k-1}(X - b_{2k-1}Y - c_{2k-1})^2 + p_{2k-1}(Y - q_{2k-1})^2 + s_{2k-1} \\ a_{2k-1}b_{2k-1} = \dfrac{\rho}{2(1-\rho^2)} \\ p_{2k-1} = \dfrac{1}{2\sigma_{2k-1}^2} \end{cases}. \tag{3.36}$$

When $i = 2k$, since $h_{2k}(X,Y)$ is a polynomial of (X,Y) of degree 2, we rewrite

$$h_{2k}(X,Y) = a_{2k}(Y - b_{2k}X - c_{2k})^2 + p_{2k}(X - q_{2k})^2 + s_{2k}, \tag{3.37}$$

where $a_{2k}, b_{2k}, c_{2k}, p_{2k}, q_{2k}, s_{2k}$ are real-valued coefficients. Using similar induction as before, from (3.22) (3.24) (3.36),

$$\begin{cases} a_{2k} = a_{2k-1}b_{2k-1}^2 + p_{2k-1} - \dfrac{1}{2\sigma_{2k-1}^2} + \dfrac{1}{2D_{Y_{2k}}} \\ a_{2k}b_{2k} = \dfrac{\rho}{2(1-\rho^2)} \\ \dfrac{1}{2\sigma_{2k}^2} = p_{2k} = a_{2k-1} - a_{2k}b_{2k}^2 \end{cases},$$

which can be further reduced by comparing with (3.36):

$$\begin{cases} a_{2k} = \dfrac{\rho^2}{4a_{2k-1}(1-\rho^2)^2} + \dfrac{1}{2D_{Y_{2k}}} \\ a_{2k}b_{2k} = \dfrac{\rho}{2(1-\rho^2)} \\ \dfrac{1}{2\sigma_{2k}^2} = a_{2k-1} - \dfrac{\rho^2}{\frac{\rho^2}{a_{2k-1}} + \frac{2(1-\rho^2)^2}{D_{Y_{2k}}}} \end{cases}. \tag{3.38}$$

Thus, (3.38) proves that Lemma 3.3 holds when $i = 2k$. Similar arguments can be used to prove the induction from $i = 2k$ to $2k+1$. Therefore, Lemma 3.3 is proved. □

Proof of Theorem 3.1 Due to the ping-pong structure of the coding procedure, there are some differences between even and odd stages. Without loss of generality, we prove the induction from $2k$ stage DiSAC2 to $2k+1$ stage DiSAC2. Similar arguments can be used to verify the induction from $2k+1$ stage DiSAC2 to $2k+2$ stage DiSAC2 (omitted here for brevity).

By induction, we first assume that a $2k$ stage DiSAC2-Gaussian is successively refinable ($k \geq 1$), that is:

$$R_{\text{DSC2}}(D_{X_{2k}}, D_{Y_{2k}}) = R_1 + \cdots + R_{2k}, \tag{3.39}$$

where $R_{\text{DSC2}}(D_X, D_Y)$ is the rate-distortion region of the DSC2 [28] as defined by (3.1).

According to the coding procedure (see the sketch of proof in Section 3.3.2), the rate and the distortion pairs at stage $2k$ and $2k+1$ are

$$\begin{cases} R_{2k} = \dfrac{\sigma_{2k-1}^2}{D_{Y_{2k}}} \\ (D_{X_{2k}}, D_{Y_{2k}}) = (\sigma_{2k}^2, D_{Y_{2k}}) \end{cases} \text{and} \quad \begin{cases} R_{2k+1} = \dfrac{\sigma_{2k}^2}{D_{X_{2k+1}}} \\ (D_{X_{2k+1}}, D_{Y_{2k+1}}) = (D_{X_{2k+1}}, \sigma_{2k+1}^2) \end{cases}, \tag{3.40}$$

where σ_i^2 is as defined in Lemma 3.2. Plugging (3.40) into (3.39), it holds at stage $2k$ that

$$R_{\text{DSC2}}(\sigma_{2k}^2, D_{Y_{2k}}) = \frac{1}{2}\log\frac{\sigma_1^2\cdots\sigma_{2k-1}^2}{D_{X_1}D_{Y_2}\cdots D_{Y_{2k}}}. \tag{3.41}$$

To prove the successive refinability at stage $2k+1$, we need to verify

$$\begin{aligned} &R_{\text{DSC2}}(D_{X_{2k+1}}, \sigma_{2k+1}^2) = R_1 + \cdots + R_{2k+1} \\ &= \frac{1}{2}\log\frac{\sigma_1^2\cdots\sigma_{2k-1}^2\sigma_{2k}^2}{D_{X_1}D_{Y_2}\cdots D_{Y_{2k}}D_{X_{2k+1}}} \\ &= \frac{1}{2}\log\frac{(1-\rho^2)\left(\sqrt{\frac{4\sigma_{2k}^2 D_{Y_{2k}}\rho^2}{(1-\rho^2)^2}+1}+1\right)\sigma_{2k}^2}{2\sigma_{2k}^2 D_{Y_{2k}}D_{X_{2k+1}}}, \end{aligned} \tag{3.42}$$

where the third equality follows from (3.1) and (3.41). By expanding the left side of (3.42) using (3.1), (3.42) can be further reduced to

$$\frac{\sqrt{\frac{4D_{X_{2k+1}}\sigma_{2k+1}^2\rho^2}{(1-\rho^2)^2}+1}+1}{\sigma_{2k+1}^2} = \frac{\sqrt{\frac{4\sigma_{2k}^2 D_{Y_{2k}}\rho^2}{(1-\rho^2)^2}+1}+1}{D_{Y_{2k}}}. \tag{3.43}$$

From Lemma 3.3, we know that

$$\begin{cases} \sigma_{2k}^2 = 1\Big/2\left(a_{2k-1} - \dfrac{\rho^2}{\frac{\rho^2}{a_{2k-1}} + \frac{2(1-\rho^2)^2}{D_{Y_{2k}}}}\right) \\ \sigma_{2k+1}^2 = 1\Big/2\left(a_{2k} - \dfrac{\rho^2}{\frac{\rho^2}{a_{2k}} + \frac{2(1-\rho^2)^2}{D_{X_{2k+1}}}}\right) \\ a_{2k} = \dfrac{\rho^2}{4a_{2k-1}(1-\rho^2)^2} + \dfrac{1}{2D_{Y_{2k}}} \end{cases}. \tag{3.44}$$

Clearly, it holds that $|\rho| < 1, D_{Y_{2k}} > 0, D_{X_{2k+1}} > 0$, and $a_{2k-1} > 0$ according to Lemma 3.3. Under these conditions, by plugging (3.44) into both sides of (3.43), we can verify the equality after some extensive calculations[5]. Therefore, it proves that a $2k+1$ stage DiSAC2-Gaussian is successively refinable.

In Section 3.3.2, we already proved the initial conditions (3.7) and (3.10). Thus, by using the induction above, we can conclude that any M-stage DiSAC2-Gaussian is successively refinable.

□

[5] One can verify this using computer algebra softwares like Mathematica.

Chapter 4

Cooperative Coding for Distributed Video Sources

4.1 Introduction

We investigated cooperative coding for distributed image sources in the previous chapter. Here, we extend the discussion to distributed video sources.

Traditionally, video coding techniques achieve high compression ratios by exploiting the inter-frame correlations at the encoding process. Multiple frames are buffered, and the encoder applies a computation-intensive motion estimation jointly to all the image frames. Such a method is referred to as joint video coding in this thesis. Recently, distributed video coding has emerged as an alternative coding paradigm. It exploits the inter-frame correlation at the decoding process instead of at the encoding process. In this way, most of the computational complexity is shifted from the encoder to the decoder, which is suitable for cameras with limited computational capability. In Section 4.2, we briefly introduce the joint video coding and the distributed video coding schemes for single-camera scenarios.

To encode video streams from n distributed cameras under the cooperative sampling framework, as the sampling grids of cameras interleave with each other, we can analyze the n video streams as a single video stream (Figure 4.6). However, as consecutive frames of the "merged" video come from disjoint cameras, motion estimation is not directly feasible on any of the cameras. In Section 4.3, we propose three multi-camera video coding methods to address this problem, namely (i) independent, (ii) distributed and (iii) joint coding methods. The independent coding method simply ignores the correlation between the cameras. The distributed coding method employs the single-camera distributed video coding method in the multi-camera scenario. The joint coding method uses the free overhearing for exploiting inter-camera correlations. The coding methods we propose are related to the recent developments in multi-view video coding [45] [31]. Unlike the interleaved setup we consider, in these works, it is assumed the sampling grids of cameras are perfectly aligned with each other.

In Section 4.4, we simulate the proposed multi-camera video coding methods on a practical two-camera system. The experimental results show that independent coding and joint coding perform substantially better than the state-of-the-art distributed

coding. This is due to the fact that inter-frame correlation is not strong enough in monitoring applications. The comparisons between independent coding and joint coding show that joint coding has better performance when the sampling rate f_s is lower than 6 frames per hour. Finally, it is shown that the per-camera consumption is reduced by 30%-50% in a two-camera system, as compared to a single-camera system. This saving is slightly smaller than the ideal saving of 50%, which is primarily due to the overheads of a distributed system. We will investigate the per-camera consumption further with a larger scale experiment in Chapter 5, and give a solution to the problem of distributed losses.

4.2 Video Coding for a Single Camera

Video coding for a single camera is a well-studied topic in the video processing community. It has wide applications in everyday life, as the huge data volume of a raw video sequence is barely possible to store or transmit. Contemporary video coding schemes achieve high compression ratios by exploiting inter-frame correlations. We can classify the video coding schemes into two main categories:

- Joint video coding (JVC), which exploits the inter-frame correlation at the encoder.
- Distributed video coding (DVC), which exploits the inter-frame correlation at the decoder.

Currently, JVC achieves better compression ratio than the state-of-the-art DVC. In the single camera case, the main advantage of DVC is to shift the encoding complexity to the decoder side, and therefore, it is designed for hardware with limited computation power. In Section 4.3, we show that DVC also provides a natural solution for the multi-camera video coding problem. We briefly introduce now both approaches and survey the most widely-used coding schemes.

4.2.1 Joint Video Coding

In the single camera case, the encoder can access all frames of a locally captured video. If the application layer does not enforce a tight constraint on decoding delay, we can compress consecutive frames in a joint fashion. High compression ratios can be achieved by exploiting strong correlations between frames. We start by surveying two commonly used JVC methods, and then propose a simple threshold-based video coder specifically tailored for monitoring applications.

Hybrid Video Coder

The hybrid coder [46] is the foundation of contemporary video coding standards. Figure 4.1 illustrates its basic structure: The raw image frames are usually represented in RGB color space. As human eyes are more sensitive to the edges/shapes than colors, the raw video is first converted into YUV color space, allowing different compression rates for the luma component (Y) and chrominance components (UV), respectively. Then, each frame is quantized in the frequency domain, and the quantized coefficients

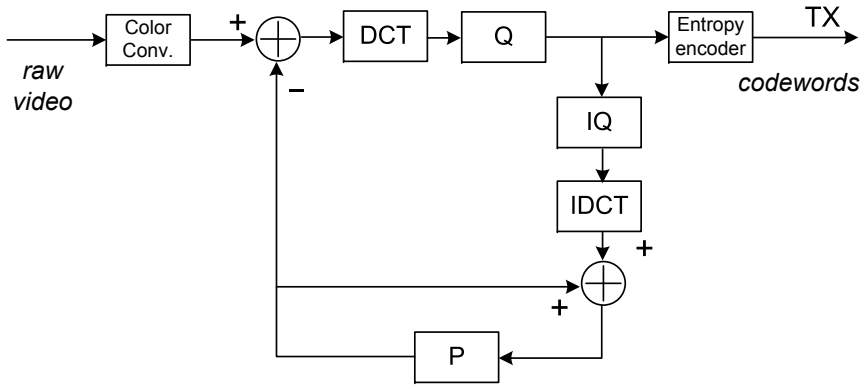


Figure 4.1: *Hybrid encoder: DCT transforms the image into frequency domain. Q is quantization block. P is prediction block. IDCT and IQ are the inverse operations of DCT and Q, respectively.*

are compressed with lossless entropy coding. Meanwhile, the transmitted codewords are locally decoded by inverse operations, and the reconstructed frame is used in prediction coding of the newly arrived frames. Note that it is important to use the reconstructed frame instead of the original frame for prediction coding. Although the encoder has access to the original frames, the decoder has no such information. As shown in the following video coding methods, this design principle is essential.

H.264 Video Coder

H.264 is currently one of the most commonly used video compression standards. It inherits the basic structure of hybrid video coder, and introduces the following major improvements [47]:

1. Adaptive deblocking filter is used in the prediction loop to reduce the block-artifacts.
2. Multiple video frames are stored in memory for inter-frame prediction.
3. Intra-frame prediction is used to exploit correlations in the same frame.
4. The discrete cosine transform (DCT) is replaced by an integer transform for transform coding.

Compared to its predecessors, H.264 is able to achieve higher compression ratio at the cost of significant encoding complexity. It is only until recently that the general-purpose CPUs are able to perform real-time high-definition H.264 encoding. To fit hardware requirements in various application scenarios, H.264 provides several encoding levels for tuning the tradeoff between complexity and compression ratio.

Threshold-based Video Coder

In visual monitoring applications, the camera and the background of the monitored scenery are usually static. For instance, in avalanche monitoring, an autonomous

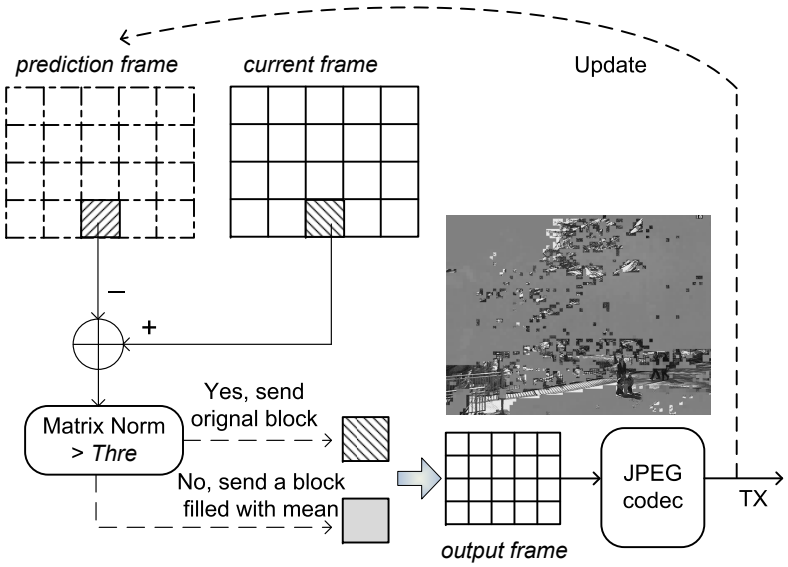


Figure 4.2: *Routine of the threshold-based video coder. The image on the right side is an example of the transmitted codewords.*

camera is fixed to monitor a snow covered mountain, hence the captured videos are static most of the time. To exploit this fact, an intuitive idea is to adopt an event-driven approach, that is, to send pixels that are more "active" than static background pixels. Based on this idea, we propose a simple video coder that has low computational complexity and achieves competitive performance in monitoring applications.

We briefly introduce the idea of the threshold-based video coder in Figure 4.2: Two image buffers are used at the encoder: the prediction frame and the output frame. We compare the incoming new frame with the prediction frame block by block. If the similarity of the two blocks (matrix norm of the difference) exceeds a threshold *Thre*, then the entire new block is sent to the output buffer. Otherwise, only the mean value of the difference is sent. Finally, all the blocks in the output buffer are concatenated into a single image and compressed by a JPEG encoder. The detailed algorithm for this video coder is illustrated in Algorithm 4.1, which includes other important procedures like prediction frame update and adaptive thresholding.

4.2.2 Distributed Video Coding

Distributed source coding [24] addresses the problem of multiple encoders transmitting information to a common decoder, given that the sources are correlated but the encoders can not communicate with each other. Hence, it is only possible to exploit the inter-source correlations at the decoding process. Analogously, single-camera distributed video coding also ignores the inter-frame correlation at the encoder, and the motion estimation process is shifted from the encoder to the decoder. In the single-camera case, the main advantage of distributed video coding is its low encoding

Algorithm 4.1 Threshold-based video coder

1. initialize $Thre, \alpha > 0$, clear prediction frame to all-zero
2. set desired distortion D_0
3. **while** a new image frame is captured **do**
4. **for all** blocks in current frame **do**
5. *cur* is the chosen block for processing in the current frame
6. *mem* is the corresponding block of *cur* in the prediction frame
7. $dif = \|cur - mem\|$, where $\|\cdot\|$ is the matrix norm
8. **if** $dif > Thre$ **then**
9. output *cur* to the buffer
10. **else**
11. output a block filled with *dif* to the buffer
12. **end if**
13. update the prediction frame with the reconstruction of *cur*
14. **end for**
15. concatenate all blocks in buffer into an image, and compress it using JPEG encoder
16. evaluate the distortion D of current frame by locally decoding
17. adaptive threshold: $Thre = Thre - \alpha \cdot (D - D_0)$
18. transmit the codeword to receiver
19. **end while**

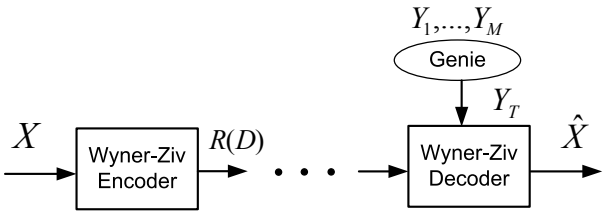


Figure 4.3: *Genie-assisted Wyner-Ziv coding. The Genie reveals T only to the decoder. The encoder does not have access to or is constrained from using $Y_1, ..., Y_M$ [48].*

complexity, which can be useful in mobile platforms. The basic principle of distributed video coding can be explained as follows:

Assume we want to encode a block X of the current image frame, and $Y_1, \cdots, Y_M$ are surrounding blocks that have already been sent and decoded at the receiver. These surrounding blocks are from previous frames or within the same image. Assume that Y_T, $T \in (1, M)$ has the strongest correlation with X. Thus, Y_T can be treated as the side information. This setup is a Wyner-Ziv coding scenario [27] if a Genie could reveal T to the decoder (Figure 4.3). Note that in the encoding process, $Y_1, \cdots, Y_M$ are assumed not to be available to X. Theoretically, if the prediction error $X - Y_T$ is white Gaussian, this distributed coding setup matches the rate-distortion performance of centralized coding, as if $Y_1, \cdots, Y_M$ were completely available to X.

Without the help from the Genie, T becomes an ambiguous state to the decoder. Puri et al. [48] suggested a decoding procedure similar to conventional motion search:

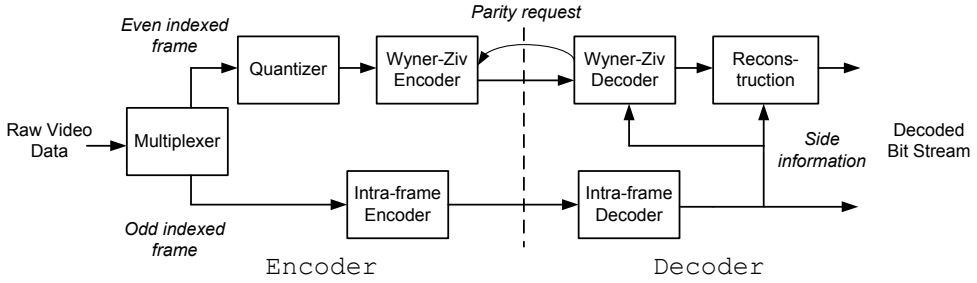


Figure 4.4: *Block diagram of the DISCOVER encoding/decoding architecture.*

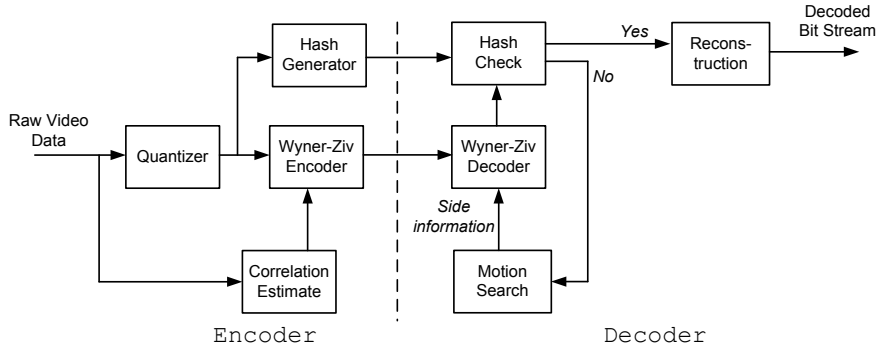


Figure 4.5: *Block diagram of the PRISM encoding/decoding architecture.*

the decoder tries each Y_i until it gets a sufficient correlation between Y_i and the recovered X. Therefore, the intensive motion estimation task is moved from the encoder to the decoder.

In the following, we briefly introduce the two major practical implementations of distributed video coding, namely, Stanford's DISCOVER codec [29] and UC Berkeley's PRISM codec [48].

DISCOVER

The DISCOVER codec [29] is an interpolation-based scheme. As shown in Figure 4.4, the basic idea is to multiplex the input video stream into odd indexed and even indexed frames. It encodes and decodes the odd indexed frame by conventional joint video coding method. For the even indexed frames, the decoder progressively requests parity bits from the encoder and interpolates the unknown even frames from odd indexed frames, until the parity check is passed. Thus, this scheme requires a feedback channel for the parity request procedure.

PRISM

The PRISM codec [48] is a backward prediction-based scheme. As shown in Figure 4.5, it transmits the parity bits and the hash of each frame, parameterized by the

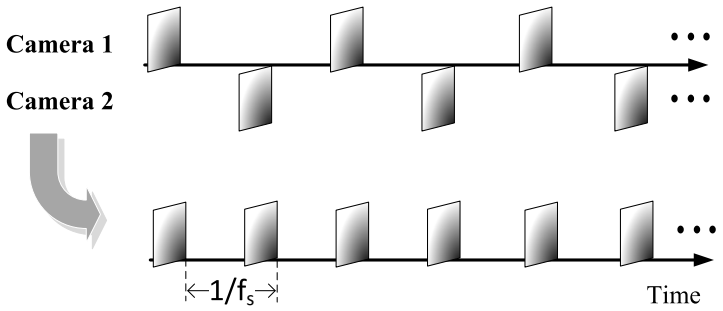


Figure 4.6: *In a two-camera monitoring network, the two raw video streams can be thought as a single video stream with a doubled sampling frequency.*

estimated correlation noise. The receiver performs the Wyner-Ziv decoding by using a motion block as the side information. This procedure is repeated until the hash of the reconstructed block matches the received hash.

4.3 Video Coding in a Multi-Camera System

In a multi-camera monitoring network, multiple cameras cooperate to monitor a common scene of interest, as introduced in the cooperative sampling framework of Chapter 2. If the raw video streams from disjoint cameras are registered properly, so that the shared views are aligned, then it can be considered as a single video stream captured by a single camera that samples at a frequency of f_s (see Figure 4.6 as an illustration of the two-camera case). In this case, the sampling frequency of the "merged" video is n times higher than the sampling frequency of each camera. Intuitively, as the inter-frame correlation increases with the sampling frequency, it is therefore more efficient to jointly encode all the frames than independently coding on each camera.

In this section, we investigate three video coding methods to compress n video streams of a n-camera monitoring system, namely (i) independent, (ii) distributed and (iii) joint coding methods. We assume the view registrations between static cameras have been properly estimated (see Section 4.A for details on the image registration algorithm), and only the overlapping regions are considered in the coding process. As non-overlapping parts are normally not correlated, they are encoded independently using conventional methods.

4.3.1 Independent Video Coding

The most straightforward solution is to apply a conventional joint video coding (JVC) scheme to each camera independently, as shown in Figure 4.7a. Note that we can choose any JVC scheme, such as the ones introduced in Section 4.2.1. This coding scheme does not require any communication between cameras, as the information is directly transmitted from the cameras to the base station (single-hop star topology described in Section 2.3).

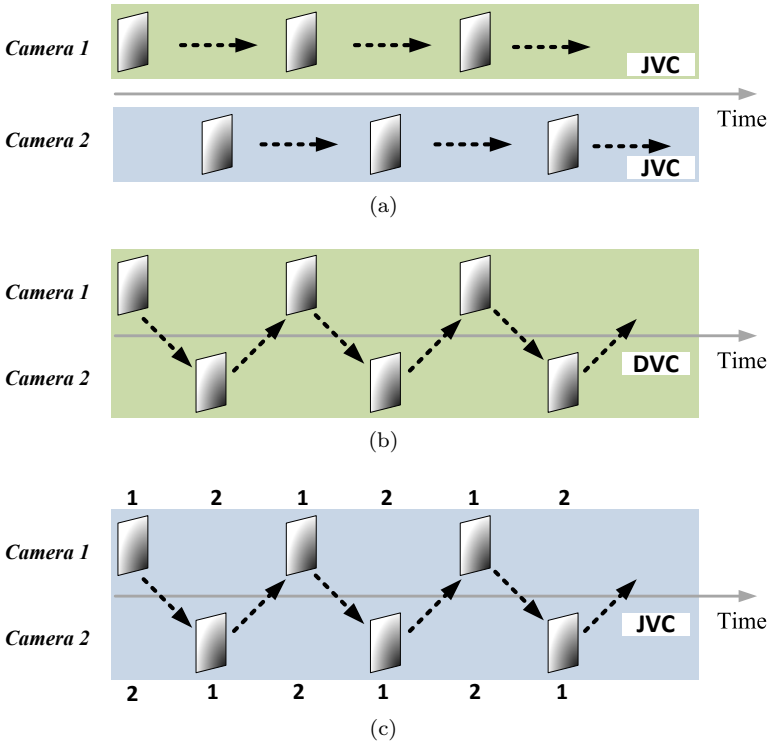


Figure 4.7: *Video coding in a multi-camera system (two-camera case): (a) independent coding, (b) distributed coding, and (c) joint coding. The two color blocks in (a) indicate that the two cameras are encoded/decoded independently, while in (b) and (c) the two cameras are considered together. The dashed arrow indicates the inter-frame dependency in the coding procedure. The number sequences in (c) show the switch position of the encoding unit of Figure 4.8.*

4.3.2 Distributed Video Coding

A different coding technique that also does not require inter-camera communication is distributed video coding (DVC), a specialized tool developed for shifting the significant burden of motion estimation from the encoder to the decoder (Section 4.2.2). In such a scheme, the encoder does not use knowledge of previous and subsequent frames to encode the current video frame. This coding paradigm becomes particularly suitable in a multi-camera system, because consecutive frames are indeed from physically separated cameras.

As illustrated in Figure 4.7b, we consider the n interleaved video streams[1] as a single video stream. In the encoding process, after view registration, DVC encodes each frame independently, which does not require any inter-camera communication. In the decoding process, the base station collects all the codewords and arranges them sequentially according to their time-stamps. Finally, joint decoding of the ordered

[1] For simplicity, we show the case $n = 2$.

codewords can recover the "merged" video stream.

Out of the mainstream distributed video coders introduced in Section 4.2.2, DISCOVER performs closest to the joint video coding method H.264. Hence, we use it as a reference in the experimental part in Section 4.4. Note that another advantage of DVC is that it requires a significantly lower encoding complexity than traditional joint video coding methods, and thus it can be easily implemented on a wireless camera with limited computation capabilities.

4.3.3 Joint Video Coding by Overhearing

In wireless communication scenarios, we can exploit the passive communication link between cameras. Moreover, the interleaved sampling configuration (Chapter 2) guarantees a collision-free communication with no simultaneous transmissions. This condition is satisfied as long as the duration of each transmission is smaller than the sampling time T_s. This is usually verified in practical scenarios because the image sampling frequency is constrained by limited energy supply (see Section 4.4.1 for calculations).

However, JVC requires extra energy consumption for overhearing, as cameras need to turn on the radios in reception mode while other cameras are transmitting. Table 4.1 shows power consumption of short and long range transceivers. We can see that energy consumption in transmission (TX) is significantly higher than in reception (RX) as the communication range increases. This motivates the use of the broadcast link for joint video coding: the energy gain in transmissions will potentially compensate the energy loss in overhearing, especially for long-range communications.

Table 4.1: *Power consumption of short and long range transceivers [49].*

	GSM	802.11b	Bluetooth
RX [mW]	240	60	30
TX [mW]	2860	350	14.5
Range [m]	20000	40	10

In contrast to the distributed video coding approach, we propose a joint video coding method that uses overhearing as a passive link between cameras. By overhearing the codewords generated by other cameras, a camera can perform prediction coding based on video frames of other cameras. Those frames can improve the compression rate as compared to using a camera's own previous frames. Figure 4.8 shows a modified hybrid video coder (see Section 4.2.1) to achieve joint coding of disjoint cameras. Each camera has an encoding unit in which the feedback loop of the hybrid coder is split into two different routes. The first route, when the camera switches to 1, is identical to a conventional hybrid coder in which the source image is fed into the prediction block. The second route, when the camera switches to 2, feeds the overheard message generated by the other camera into the prediction block.

Each camera switches to 1 when in transmission mode, otherwise it switches to 2 for overhearing. The number sequences in Figure 4.7c show the switch positions for two cameras at each time slot. When one camera switches to 1, the other camera

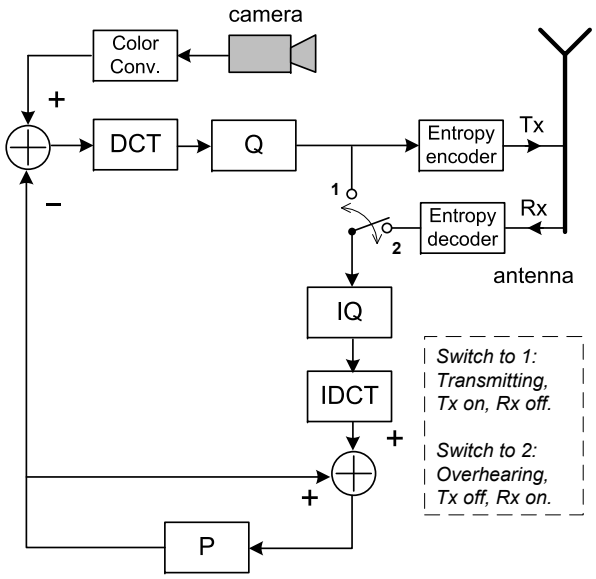


Figure 4.8: *Joint video coding of multiple cameras based on a hybrid video coder (encoder part at each camera). The motion data estimated in the prediction block is also sent but not explicitly drawn in the diagram.*

always switches to 2 for overhearing. By repeating this cooperation, the prediction blocks of all cameras are always synchronized and store the latest frame (among all cameras) for joint coding. Therefore, the encoder of each camera virtually compresses a video sampled at a frequency of f_s, rather than its own sampling frequency f_s/n. In this way, inter-camera correlations are fully exploited and a better compression ratio can be achieved.

4.4 To Overhear or Not to Overhear

Energy consumption is critical in the evaluation of a wireless camera network. In this section, we study the performance of the three proposed multi-camera video coding methods, namely, independent, distributed and joint coding methods, in a practical two-camera system. Unlike the independent/distributed video coding methods, joint video coding utilizes the broadcast link, which is expected to bring energy savings. Nevertheless, extra energy will be consumed for overhearing broadcast messages. Using experimental results, we study whether the broadcast link improves the energy efficiency in practical scenarios.

To obtain an approximate energy profile of a two-camera system, we first collect video datasets for benchmarking purposes. Then we obtain the energy consumption for computation by running the implemented algorithms with these datasets on our embedded image processing platform *Sensorcam*. The encoded video size and the typical long-range radio power (Table 4.1) are used to estimate the energy consumption for communication. Finally, the global energy profile is the sum of computation and

Table 4.2: *Parameters of dataset collecting*

dataset	sample	weather	duration	frames
Scene A	Figure 4.10a	cloudy	14:00-24:00	594
Scene B	Figure 4.10b	sunny	06:30-16:30	600

Table 4.3: *Parameters used in energy profiling*

Colibri PXA270 [50]	
CPU	520MHz
SDRAM	64MB
FLASH	32MB
Consumption (normal)	800mW
Consumption (sleep)	7mW
Long-range radio [49]	
Consumption (Tx@30dBm)	2860mW
Consumption (Rx)	240mW
Consumption (idle)	14mW
Speed	16kbps
camera	
Resolution	640×480
Consumption (active)	120mW
solar power system @ 2.4W	
Battery capacity	135kJ
Average energy supply (winter)	12.6kJ/day

communication consumption. In the following, we first introduce the experimental setup and then present the detailed evaluation results.

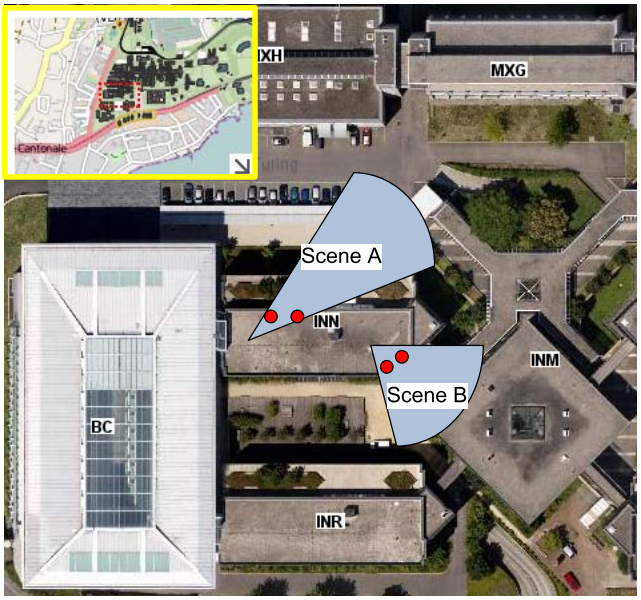
4.4.1 Experimental Setup

Using two conventional cameras (Figure 4.9), we start by collecting two datasets with different depth structure for benchmarking purposes. Scene A (Figure 4.10a) captures the facade of a building, a planar scene well modeled by homography geometry. In contrast, Scene B (Figure 4.10b) has a complex depth structure with buildings that stretch out to the mountains far away. For practical reasons, the two cameras are placed at a distance approximately one meter. This is smaller than in real-world deployments where the separation will be larger for robustness concerns. Nevertheless, Scene B introduces a parallax between two views, mimicking two cameras deployed with a large separation.

The two deployed cameras are synchronized to start simultaneously and sample once a minute. We can get image sequences as in the interleaved sampling setup (Figure 4.6) by selecting appropriate images from the raw dataset. Moreover, by subsampling the video stream, we get datasets with different sampling intervals T_s. The related information of these two datasets is listed in Table 4.2.

Based on an open-source H.264 codec [51], we implement the independent video

(a)



(b)

Figure 4.9: *Experimental setup to capture datasets: (a) Two surveillance cameras with solar power system watching Scene B. (b) The setup of cameras (red dots) on the roof of a building. Two datasets, Scene A and B, are illustrated with the corresponding viewpoints.*

(a)

(b)

Figure 4.10: *Sample images of the two datasets: (a) Scene A. (b) Scene B.*

coding and the joint video coding algorithms on *Sensorcam*, our wireless camera module (see Section 4.B for details). We use DISCOVER as the distributed video encoder and estimate its computation consumption from the H.264 consumption on *Sensorcam*[2].

To analyze the energy requirements in environmental applications, we installed one of the cameras powered by a 12V battery and a 2.4W solar panel in the *Val Ferret* deployment (see Figure 1.2a). Based on the solar radiation data collected, we calculate the average energy supply per day during the winter period (most adverse period) from January 15th 2011 until March 17th 2011. The energy budget and some of the parameters related to the experiment are listed in Table 4.3.

In order to use the joint video coding (Section 4.3.3), we must first ensure that there is enough "idle" time in the communication channel for overhearing (see Sec-

[2]No source code of DISCOVER is available; only an executable codec for Win/Linux workstations. The relative complexity of DISCOVER with respect to H.264 is available here: http://www.img.lx.it.pt/~discover/complexity.html

tion 4.3.3). From Table 4.3, the available energy per day is 12.6kJ $\times$ 0.7 $-$ (7 + 14)mW $\times$ 24h = 7.0kJ, from which we subtract the background consumption during sleep mode. For a 640$\times$480 resolution video, and assuming we use the naive Motion JPEG compression, the typical size of each frame ranges from 20 to 40kB. The energy consumption per frame can be estimated as:

- Wireless transmission: $20 \sim 40$kB/16kbps $\times$ 2860mW $= 28.6 \sim 57.2$J.
- Camera: 1s $\times$ 120mW $= 0.12$J.
- CPU (compress/transmit image): $(20 \sim 40$kB/16kbps + 2s$) \times 800$mW $= 9.6 \sim$ 17.6J.

Therefore, the energy budget allows for a sampling frequency (no transmission at night) at $8 \sim 15$ pic/h. In other words, the sampling interval of each camera is $240 \sim 450$s. To transmit an image, it takes $20 \sim 40$kB/16kbps $= 10 \sim 20$s. As the sampling interval is significantly higher than the image transmission time (240$\sim$450s $\gg$ 10$\sim$20s), it is clear that there is no interference between transmissions of different cameras.

4.4.2 Video Coding Comparisons

We now compare the three video coding methods for a multi-camera system, namely, distributed video coding method (DISCOVER), H.264 based joint coding method, and H.264 based independent coding method (see Section 4.3).

The global energy consumed at each camera can be expressed as:

$$E_{\text{global}} = E_{\text{ENC}} + E_{\text{TX}} + E_{\text{RX}}, \tag{4.1}$$

where E_{ENC} denotes the computation energy spent on video encoding (e.g., H.264), E_{TX} denotes the communication energy for transmitting codewords, and E_{RX} denotes the communication energy for overhearing. Particularly, $E_{\text{RX}} = 0$ for the independent and the distributed coding methods, and $E_{\text{RX}} \neq 0$ for the joint coding method. For error-resilience purposes, we set a group of pictures (GOP) for each video coding method. For instance, if the sampling interval T_s is 1 minute and GOP is 4, then we transmit an independent frame every 4 frames, and thus the receiver can recover from packet loss within 4 minutes.

Figure 4.11 shows the global energy E_{global} versus the video quality of distributed and joint coding methods for both datasets, Scene A and Scene B. We can see that joint video coding outperforms distributed video coding substantially.

The loss for distributed coding suggests that the state-of-the-art DVC schemes rely heavily on strong correlations between successive video frames. In regular video coding scenarios where DVC shows competitive performance, the sampling rate is 30 fps. However, the sampling rate is much lower in monitoring applications, e.g., $0.0003 - 0.02$ fps. Although the scene is relatively static in our scenario, the dramatic light variation in outdoor monitoring creates big variations. As a result, the inter-frame correlation drops quickly as the sampling frequency decreases.

Figure 4.12 shows the energy savings of the joint video coding method compared to the independent video coding method. The savings are plotted in a 2D-parameter

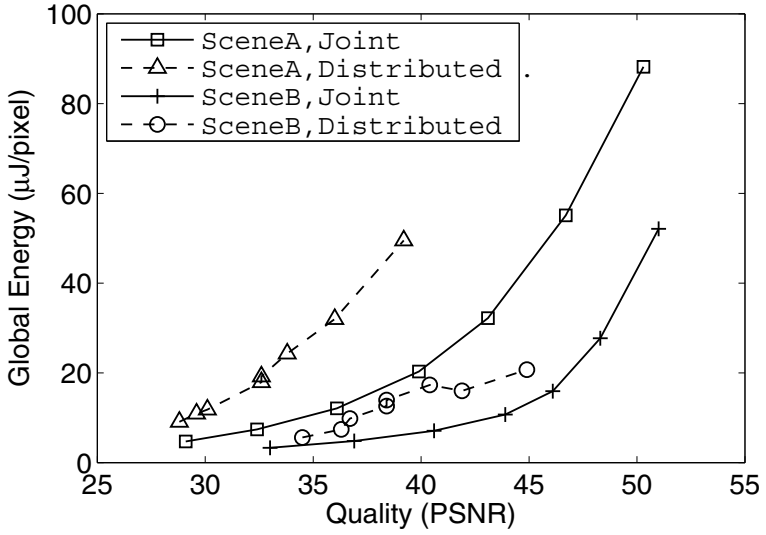


Figure 4.11: *Distributed versus joint video coding: global energy (measured in $\mu J/pixel$) with respect to video quality by using the distributed/joint video coding methods for the two datasets. The GOP values are chosen to maximize video compression ratios. The sampling interval T_s is fixed to 5 minutes.*

space of (T_s, GOP). The video qualities of both schemes are equal and fixed (`QP`=30 in H.264 encoder). Note that as the sampling rate of monitoring applications is much lower than in regular video coding, the GOP value is also lower. The dashed lines show parameter combinations that have a constant error-resilience capability; specifically, the system can recover from an error in a maximum of 60 minutes. Clearly, as the sampling interval T_s increases (follows the dashed line from left to right), the advantage of joint video coding appears. Eventually, when the T_s goes to 20-30 minutes, the energy saving is close to 10%. This trend is mainly due to the decreasing GOP value, which forces more frames refreshing in independent video coding than in joint video coding.

We summarize the comparison results in the following:

1. For monitoring applications, the distributed coding method fails to exploit the temporal correlation between disjoint cameras looking the same scene.

2. In a long-range communication setup, the joint coding method outperforms the independent coding method when the sampling frequency f_s is low.

3. Considering the system complexity, when the sampling rate is not low enough (i.e., sampling interval T_s smaller than 10 minutes), the independent coding method is the most efficient video compression scheme for a multi-camera system.

It is also worth mentioning that in addition to the extra overhearing consumption, the performance of the joint coding method also suffers from the inter-view

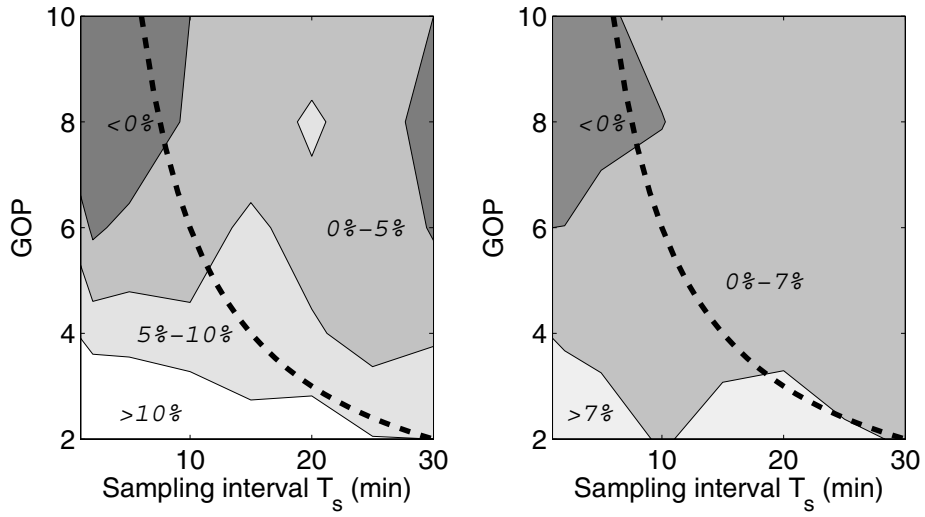


Figure 4.12: *Independent versus joint video coding: global energy saving (measured in %) of the joint video coding method compared with the independent video coding method. The savings are plotted in a 2D-parameter space of (sampling interval T_s, GOP). The video qualities of both schemes are equal and fixed. The dashed lines show parameter combinations that have a constant error-resilience capability (recover in max. $GOP \times T_s = 60$ minutes). (a) Scene A. (b) Scene B.*

registration error. In fact, the image registration algorithm (Section 4.A) uses an approximate stereo-view model, primarily due to the missing 3D depth information of the monitored scene. This algorithm produces more errors when the scene structure becomes complex (e.g., energy saving in Scene B is lower than in Scene A). If the depth information can also be captured by a dedicated device, then we can potentially eliminate most of the registration error by employing an exact stereo-view model. Recently, depth cameras based on active laser projectors are emerging for indoor applications [52] [53]. For outdoor applications, it remains a challenging problem because the range of low-power laser projectors is still quite restricted.

4.4.3 Per-Camera Energy Consumption

The cooperative sampling framework also enables multiple cameras to share energy consumption. Ideally, if we want to achieve a fixed f_s, as the sampling frequency of each camera is inversely proportional to the number of cameras f_s/n, the per-camera energy consumption should also scale as $1/n$. Nevertheless, a distributed system will certainly have performance losses as compared to a centralized system. In this section, we compare the per-camera consumption of a two-camera system with a single-camera system ($n = 2$ versus $n = 1$).

The solid line in Figure 4.13 illustrates the per-camera energy consumption of a

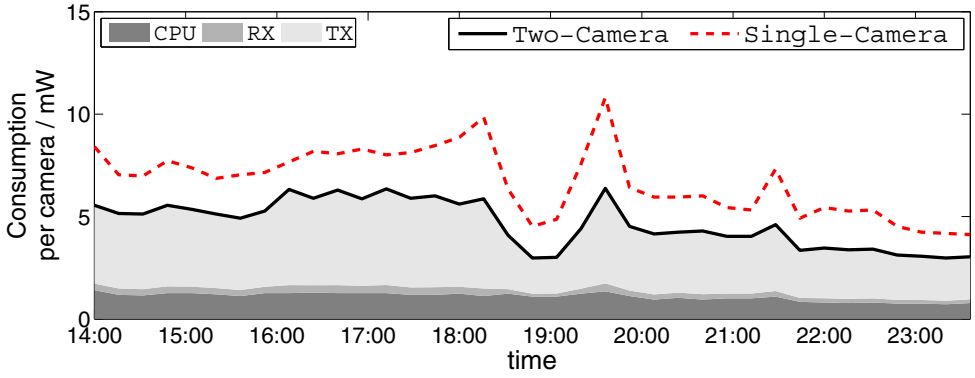


Figure 4.13: *Per-camera energy consumption over time: the solid line for a two-camera system using the joint coding method, and the dashed line for a single-camera system with the same event-detection probability. The dataset used is Scene A (14:00-24:00), the sampling interval T_s is 4 minutes, and GOP is 4. The overall energy consumption of the two-camera system is also illustrated with its three components:* `CPU` *for computation energy spent on video encoding,* `TX` *for energy of wireless transmissions, and* `RX` *for overhearing.*

two-camera system that runs the H.264 based joint coding method. We divide the overall consumption into three categories: `CPU` for computation energy spent on video encoding, `TX` for energy of wireless transmissions, and `RX` for overhearing. As we can see, the energy consumption for communication (TX+RX) and computation (CPU) approximately share 3:1 of the global energy. The computation energy consumption is generally very static, while the communication energy consumption varies as the scene changes. For instance, we can observe significant variations during the sunset period (i.e., 18:30–19:30), which is caused by the camera exposure problem when the light condition changes rapidly. Also, it can be seen that the camera consumes less at night because the scene variation is much smaller.

All previous evaluations focus on a two-camera system. As a comparison, we can also evaluate the per-camera energy consumption of a simple single-camera monitoring system. Despite losing robustness, a single camera does not suffer from registration error and overhearing cost. The dashed line in Figure 4.13 shows the energy consumption of a single-camera system with the same event-detection probability, that is, the same sampling frequency of the system f_s. The comparison shows that the per-camera energy consumption of the two-camera system is reduced by 30%-50%, depending on the scene conditions. These energy savings are approximately consistent with a scaling of $1/n$ for $n = 2$, with an extra loss of less than 20%. As we mentioned before, this loss is due to overheads in a distributed system (e.g., registration error and overhearing cost). In Chapter 5, we will study the per-camera consumption in a larger scale experiment of nine cameras. The result also shows a scaling of $1/n$ with an extra term $\mathcal{O}(\log(n)/n)$. We will propose a coding method that reduces the performance loss of a distributed system.

4.5 Summary

- Joint video coding (JVC) is a traditional approach that generally achieves the best compression ratio. Distributed video coding (DVC) is a new coding paradigm that shifts most of the encoding complexity to the decoder side, which is suitable for cameras with limited computational capability.
- Distributed video coding (DVC) can be used in multi-camera scenarios, because it requires no knowledge of neighboring frames during the encoding process. However, the experimental results show that the state-of-the-art DVC performs worse than independent coding, mainly due to insufficient inter-frame correlations in monitoring applications.
- As the overhearing cost is small in the long-range communication scenario, we can use JVC in a distributed multi-camera system. The experimental results show that JVC outperforms independent coding when the sampling rate is lower than 1/(10min).
- Due to various overheads in JVC, such as registration errors and overhearing costs, independent coding is the best choice when the sampling rate is higher than 1/(10min).
- The experiment on a practical two-camera system shows that the per-camera consumption is reduced by 30%-50%. There is a performance loss of less than 20%, which is primarily due to the overheads of a distributed system.

4.A Inter-Camera Registration

As we mentioned in Section 3.2.2, when the monitored objects and the two cameras are relatively static, we can establish a point to point mapping between the two captured images. As shown in Figure 4.14a, IX and IY are two stereo-view images from the left and the right camera, respectively. After establishing point-wise correspondences between IX and IY, IY is projected onto the image space of IX so that both images are aligned and share a common area. We denote the overlapping area as *common imaging area*. X, Y denote the aligned images for IX and IY, respectively. We apply the joint image/video coding on the *common imaging area*; the rest of the image have very limited correlation and therefore can be encoded independently.

To establish point-wise correspondence between stereo-view images, epipolar geometry is the standard tool for image registration [40]. It requires known depth structure of the monitored scene as a prior. Estimating or capturing depth is one way to retrieve depth structure. However, on one hand, the state-of-the-art depth estimation algorithms are not accurate enough for registration. On the other hand, current depth cameras are restricted to indoor uses due to the limited range of low-power laser projectors. As an alternative, we propose a registration algorithm based on the homography geometry, which is an approximate model of the epipolar geometry. This algorithm is used in Chapter 3 and Chapter 4 for inter-camera registrations.

The homography geometry is described by a single 3×3 matrix $\mathbf{H}$, so that the positions of two correspondence points $\mathbf{x}, \mathbf{y}$ of stereo-view images can be related by a linear transformation:

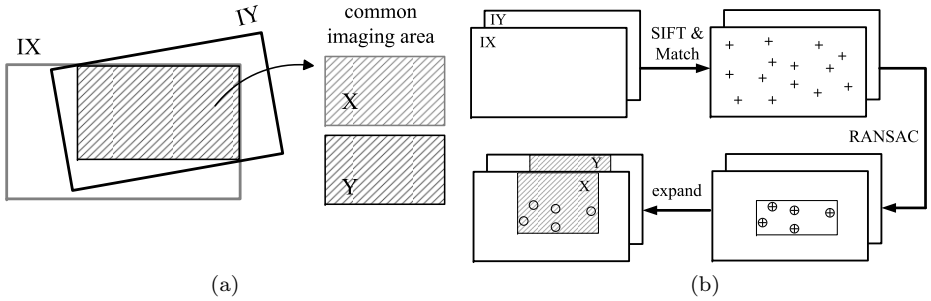
$$\mathbf{x} = \mathbf{H} \cdot \mathbf{y}.$$

Note that $\mathbf{x}, \mathbf{y}$ are homogeneous coordinates [40]. For instance, the coordinate $\mathbf{x} = (x_1, x_2, x_3)$ corresponds to the point at position $(x_1/x_3, x_2/x_3)$.

The simplicity of this model greatly reduces the complexity of model estimation and the transmission overhead. However, such a model can be inaccurate because it is designed for (i) two images taken by a fixed rotating camera or (ii) stereo-view images with only planar scenes. Thus, the registration error in the near-field scenery creates a negative effect for the overall coding efficiency. To overcome this, we propose a robust algorithm to detect a *common imaging area* that can be approximately regarded as the planar scenery. This part of the image obeys the homography geometry, and the other parts of the image are considered to be near-field scenery that does not have enough correlation between the two cameras. This strategy fits our application, since the planar scenes "far" from the cameras dominate in monitoring applications.

To estimate the homography model ($\mathbf{H}$) for two stereo-view images, we need at least 4 non-collinear correspondence point pairs. These correspondence pairs are obtained by extracting feature points using SIFT [54] or SURF [55]. Each feature point is identified by its position in the image and has a feature vector to describe its surrounding textures. With all the generated feature points, we can establish the correspondence using the nearest neighbor search method (*correspondence poll*). To avoid outliers in the correspondence poll, we use RANSAC [56], a robust and efficient model estimation algorithm for finding the errors in the dataset.

To automatically detect the *common imaging area*, we exploit the fact that the feature points selected by RANSAC tends to cluster in space. These points will define the area that is well registered after homography transformation. Still, this simple



(a) (b)

Figure 4.14: *Inter-camera registration: (a) Illustration of common imaging area (slashed zone). (b) Robust registration algorithm of stereo-view images: The features (cross points) extracted from the two images are first matched, and then the inliers (circles) are picked out by RANSAC. The final step is to determine the common imaging area from the positions of those inliers.*

(a) (b)

Figure 4.15: *Inter-camera registration results for the datasets Scene A and Scene B in Section 4.4.1. The registered right view is overlaid on the left view. The yellow frame labels the common imaging area.*

idea may miss some background area (e.g., sky) if the features of the texture are too weak to be selected. However, in monitoring applications, the background usually lays in the upper part of image. Therefore, we can naturally assume that the region above the labeled area belongs to the background (e.g., the sky), which is also a planar scene.

In sum, Figure 4.14b illustrates the entire routine of the robust registration algorithm:

1. Feature points are generated for two stereo-views image IX and IY, and matched with each other.

2. RANSAC selects the inlier point pairs that give a robust estimation of the homography matrix.

3. The minimal rectangular hull containing all inlier point pairs is determined, and is then extended to the top of the image. This is the detected *common imaging area* X, Y.

Figure 4.15 shows sample registration results of the datasets Scene A and Scene B in Section 4.4.1.

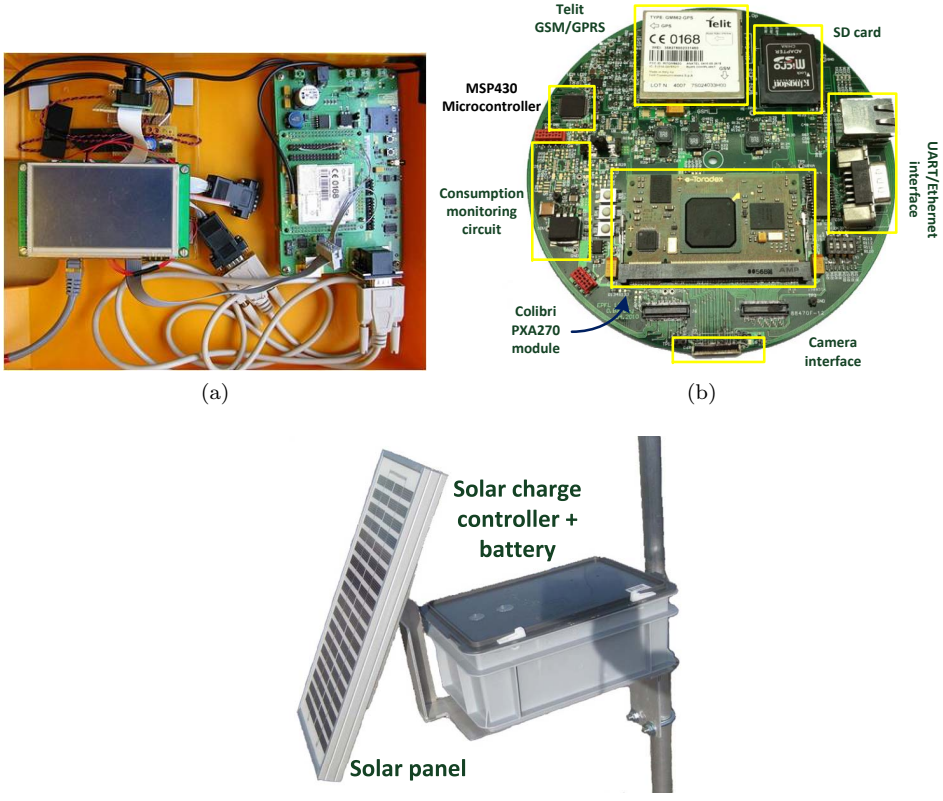
4.B Sensorcam: A Smart Wireless Camera

To get visual information from the environment, we need a wireless camera module that periodically takes pictures of the monitored scene and transmits the image sequence to a base station. Powered by a solar panel and a rechargeable battery, it can be installed for outdoor monitoring. The camera needs to work autonomously for long periods of time, and to achieve the required sampling frequency (1-200 images/hour in our case). In order to minimize power consumption and bandwidth requirements, the camera must be "intelligent" to process large amounts of raw data and transmits as few bits as possible.

Our first approach to environmental visual monitoring was the deployment of a mobile phone with an integrated camera. It was powered by a car battery and installed on the Genepi glacier from October to December, 2007. It kept sending one picture of the glacier every hour during daytime and stopped working after two months due to power depletion. We consider it as a successful feasibility study. However, as the entire system of commercial smart phones is not accessible to common users, it is impractical to make low-level power management in such platforms for energy efficiency.

Based on the experiences gathered in the first deployment, we developed *Sensorcam*, a fully flexible, long-range, smart wireless camera running a Linux-based open system. *Sensorcam* includes a master board capable of embedded computing, GSM radio communication and GPS localization, and is powered by a solar power system (Figure 4.16c). The master board runs an embedded Linux system and is able to control all the hardware peripherals from the software level. The main benefit of the Linux system is that it can directly adopt libraries and programs from the open-source community. For instance, we have extensively used the OpenCV library [57] for image processing programming. It is also worth to mention that the developers of such libraries have started to release optimized codes for embedded platform, thanks to the emergence of mobile computing.

Figure 4.16a shows the first prototype that interconnected several evaluation boards. After testing, we combine all necessary components into a single customized master board (Figure 4.16b). Particularly, it includes: 1) *MSP430 micro-controller for power management*; 2) *Colibri PXA270 module that runs Linux system*; 3) *Telit GM862 GSM module for data communication and GPS localization*; 4) *Camera interface that connects to a camera module (e.g., Omnivision OV7720)*; 5) *UART/Ethernet interface for debugging during software development*; 6) *SD card for data storage.*



(a) (b)

(c)

Figure 4.16: *Sensorcam prototype: (a) Master board v.1. (b) Master board v.2. (c) Solar system.*

Chapter 5

Event-Driven Video Coding Using Smart Cameras

5.1 Introduction

In this chapter, we study coding methods for event detection applications. With the increased computational capabilities of wireless cameras, the camera itself is able to process large amounts of raw data. We will show that when the information (e.g., events) we want to retrieve from the remote camera becomes more specific, a more intelligent coding method is able to achieve significantly lower communication rates than conventional coding methods.

Historically, we have seen that as technology evolves, the energy cost for computation is decreasing rapidly, while the energy cost for communication is lower bounded by Maxwell equations. This has the implication that as the hardware improves, more sophisticated algorithm shall be employed to reduce the bitstreams in communication. Take an extreme example: If we wanted to transmit a message over radio in the 1940s, the best strategy was to directly encode it analogously without any digital processing, simply because the computer at that time (ENIAC) consumed 150kW of power.

Clearly, this tradeoff between the computation and communication energy poses a critical problem on how to determine the most suitable algorithmic complexity, given the current hardware platform. Aiming to minimize the global energy consumption, we formulate this tradeoff in Section 5.2. We point out that three factors can influence the optimal choice of computational complexity, namely, efficiency of computation hardware, efficiency of communication hardware, and efficiency of coding algorithm. Two case studies on Huffman coding and H.264 coding are provided to demonstrate the effect of the hardware efficiency.

On algorithmic efficiency, we argue that a coding system that mimics the human visual system can potentially achieve performance beyond traditional coding methods. This idea leads to the event-driven video coding (EVC) in Section 5.3. To detect unknown events such as natural hazards or unexpected intrusions, monitoring cameras usually directly compress and transfer the image sequences to a base station (BS). A user at the BS can query the recorded images to determine if any

particular event occurred. Conventional video compression schemes such as H.264 are inefficient because they ignore the "meaning" of video content, therefore many bits are wasted in conveying irrelevant information, such as cloud movements, changes in light condition, or shadows. The core of EVC is to use image processing techniques for understanding the monitored scene, and to transmit only critical image fragments that contain events of interest.

The experimental results in Section 5.4 show that EVC achieves a better compression ratio than H.264. Compared with the anomaly-detection based video coding scheme [58], EVC also achieves a better detection rate with the same compression ratio. We implemented EVC on a wireless smart camera deployed for an event-detection application. The results show that while having a similar computational complexity as H.264, EVC reduces the global energy consumption by 40%.

Finally, in Section 5.5, we study a n-camera cooperative monitoring system, by combining EVC and the cooperative sampling framework introduced in Chapter 2. The optimal scaling behavior of per-camera communication cost is $1/n$, which implies that a multi-camera system has no loss as compared to a single-camera system. In practice, we show through experiments on a large scale multi-camera dataset that there is a relative loss of $\mathcal{O}(\log(n)/n)$. Finally, we propose a distributed, multi-camera EVC scheme that significantly reduces this performance loss.

5.2 Computation versus Communication

To illustrate the tradeoff between computation and communication, we start by investigating the rate-distortion theory. The traditional Shannon rate limit can only be achieved with an infinite block length, while the complexity of the coding algorithm increases exponentially with the block length. Hence, in practice, there is a penalty rate in coding with a finite block length. In Section 5.2.1, we characterize this penalty rate with respect to the coding complexity. The result can be regarded as the theoretical support to the insight that computation generally reduces the compression rate.

In Section 5.2.2, we formulate the tradeoff between computation and communication and study the optimal combination that minimizes global energy consumption. We point out that three factors influence the optimal choice of computational complexity, namely, efficiency of computation hardware, efficiency of communication hardware, and efficiency of coding algorithm. In Section 5.2.3 and Section 5.2.4, we demonstrate how to find the optimal computation-communication combination, in two practical coding algorithms (Huffman coding and H.264 coding). The results indicate that for a given coding algorithm and a given hardware platform, the optimal combination is unique.

5.2.1 Rate-Complexity Function

In information theory, Shannon's source coding theorem [39] states that a block-based coding algorithm asymptotically approaches the rate-distortion function[1] of a source when the block length $m \to \infty$. For a finite block length m, there is a penalty rate

[1] In lossless coding, the distortion is always zero.

ΔR between the actual compression rate and the theoretical rate limit. From an algorithmic point of view, a longer block length usually requires more computation. We denote C_E as the coding complexity of the encoder, i.e., average number of operations per symbol to encode a source. We denote by $\Delta R(C_E)$ the rate-complexity function that relates the penalty rate and the coding complexity. By linking ΔR, m and C_E, we characterize now the penalty rate of a lossless and a lossy coding scheme.

Lossless Coding: Let $\mathscr{X}$ be a stationary stochastic process, whose cardinality is 2. According to [24, P.89], if we use a block-based coding with block length m, the expected rate per symbol R satisfies

$$\mathcal{H}(\mathscr{X}) \leq R \leq \mathcal{H}(\mathscr{X}) + \frac{1}{m},$$

where $\mathcal{H}(\mathscr{X})$ is the entropy rate of the process. Thus, the penalty rate $\Delta R = R - \mathcal{H}(\mathscr{X})$ is inversely proportional to the block length m:

$$\Delta R = \mathcal{O}\left(\frac{1}{m}\right). \tag{5.1}$$

Assuming we use Huffman coding for lossless compression, the encoding complexity is:

$$C_E = \frac{1}{m}\mathcal{O}(2^m \cdot \log 2^m) = \mathcal{O}(2^m). \tag{5.2}$$

Combining (5.1) and (5.2) gives:

$$\Delta R(C_E) = \mathcal{O}\left(\frac{1}{\log C_E}\right). \tag{5.3}$$

Lossy Coding: Let $\mathcal{X}$ be a source that produces an i.i.d. distributed sequence $\mathcal{X}^m = \{\mathcal{X}_1, \mathcal{X}_2, \cdots, \mathcal{X}_m\}$. Assume we use random codebook coding for lossy compression of $\mathcal{X}^m$ (see the proof of the achievability of the rate-distortion function in [24, P.351]).

The encoding process is to find the closest codeword of $\mathcal{X}^m$ from the random codebook

$$\left\{ \begin{array}{ccc} \hat{\mathcal{X}}_1^{(1)}, & \cdots, & \hat{\mathcal{X}}_m^{(1)} \\ & \vdots & \\ \hat{\mathcal{X}}_1^{(2^{mR})}, & \cdots, & \hat{\mathcal{X}}_m^{(2^{mR})} \end{array} \right\}, \tag{5.4}$$

where R is the rate used in compression.

Using the brute-force search, it needs $\mathcal{O}(m \cdot 2^{mR})$ comparisons. Therefore, the coding complexity is:

$$C_E = \frac{1}{m}\mathcal{O}(m \cdot 2^{mR}) = \mathcal{O}(2^{mR}). \tag{5.5}$$

For a target distortion D, denote by P_e the probability that there is no codeword in (5.4) jointly typical with the input source $\mathcal{X}^m$. According to [24, P.356], P_e is bounded by

$$P_e \leq e^{-2^{mR} \cdot 2^{-m(R_{\mathcal{X}}(D)+\epsilon)}},$$

where $R_{\mathcal{X}}(D)$ is the rate-distortion function of the source $\mathcal{X}$. To achieve a fixed distortion D when the block length m varies, P_e has to be invariant. This leads to

$$\Delta R = R - R_{\mathcal{X}}(D) = \mathcal{O}\left(\frac{1}{m}\right). \tag{5.6}$$

From (5.5) and (5.6),

$$\Delta R(C_E) = \mathcal{O}\left(\frac{R}{\log C_E}\right) = \mathcal{O}\left(\frac{R_{\mathcal{X}}(D)}{\log C_E}\right). \tag{5.7}$$

These two examples show that the penalty rate ((5.3) and (5.7)) is directly linked to the encoding complexity C_E. In both lossless and lossy coding scenarios, ΔR is inversely proportional to the logarithm of C_E. As C_E increases, ΔR eventually decreases to zero.

Note that the big O notation in both (5.3) and (5.7) implies that different coding algorithms can have a linear difference in the scaling speed. Moreover, when we transform the complexity C_E to the actual computation consumption, the scaling speed can also vary for different hardware platforms. In the following section, we discuss the tradeoff between the computation and communication consumptions, where this scaling speed will lead to different optimal tradeoffs.

5.2.2 Computation and Communication Energy

To retrieve information from a camera over a wireless channel, we typically proceed in two steps: i) compress the raw data by using a source coding method (computation), and ii) transmit the encoded bitstream (communication). When the computational complexity increases, the size of the bitstream after source coding gradually approaches the rate-distortion limit, and hence the communication energy decreases. Meanwhile, an algorithm with a higher complexity requires more computation energy. This computation and communication energy tradeoff is depicted in Figure 5.1. Thus, from the global energy perspective, this suggests that there exists an optimal choice of computation-communication combination that minimizes the global energy consumption.

To achieve a specific task, e.g., event-detection using a wireless camera, the optimal choice of computation-communication combination depends on the following factors:

1. Computation hardware: energy consumed to perform an operation.
2. Communication hardware: energy consumed to transmit one byte.
3. Algorithm efficiency: compression rate achieved with a given number of operations.

In the following, we illustrate the impact of these factors through two case studies.

5.2.3 Case Study I: Huffman Coding

Huffman coding is a lossless data compression method based on entropy encoding. It can be used to compress a raw image that is grouped into blocks of m pixels (each

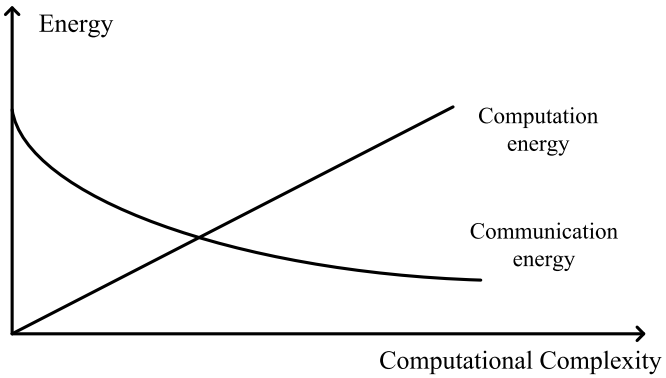


Figure 5.1: *The energy for computation and communication versus computational complexity.*

pixel is represented by a 8-bit integer). Our goal is to find the optimal block length m that minimizes the global energy consumption.

We denote by R_p the entropy per pixel, e_t the energy required to transmit one bit, and e_h the energy to encode one pixel when $m = 1$.

Similar to the analysis in Section 5.2.1, the global energy for compressing and transmitting one pixel is:

$$E_p = e_t(R_p + \frac{8}{m}) + e_h \cdot 256^{m-1} \quad \text{for } m \geq 1. \tag{5.8}$$

By computing $\partial E_p/\partial m = 0$, the optimal block length $m^\star$ that minimizes global energy satisfies

$$(m^\star)^2 \cdot 256^{m^\star - 1} \cdot \ln 2 = \frac{e_t}{e_h}. \tag{5.9}$$

Note that for a given algorithm, e_t/e_h is a hardware-specific parameter. Thus, for each hardware platform, we shall first calculate its parameter e_t/e_h, and then determine the optimal computation-communication combination. Clearly, from (5.9), a larger e_t/e_h corresponds to a larger $m^\star$. In the following, we analyze the optimal coding strategy for two different hardware platforms: *Sensorcam*, an ARM-based smart camera with a long-range radio, and *Mica2*, a low-power sensor node with a short range radio. We use a 256×256 grey image "Lena" as the source input[2].

Sensorcam Platform: To estimate the energy consumption, we implemented Huffman coding on a 2.4GHz Lenovo T400 laptop and measured the time for encoding one pixel – $0.8\text{ms}/(256 \times 256)$. Using the parameters from Table 4.3, e_h can be estimated as follows:

$$e_h = \frac{0.8\text{ms}}{256 \times 256} \times \frac{2.4\text{GHz} \times 800\text{mW}}{520\text{MHz}} = 4.6 \times 10^{-8}\text{J/pixel}.$$

[2]The entropy per pixel R_p of "Lena" is estimated to be 6.57 bits.

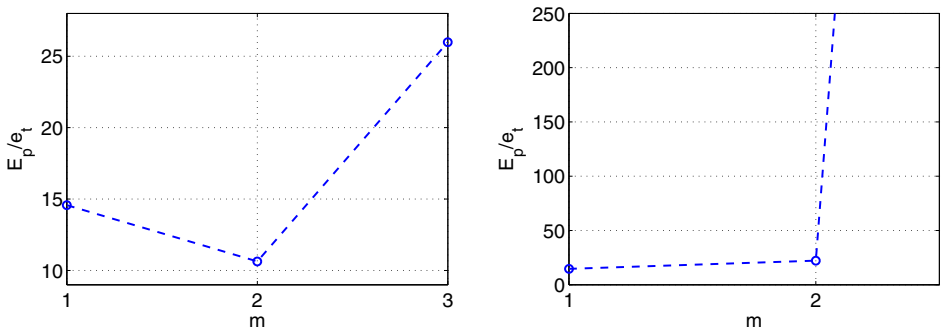


Figure 5.2: *The normalized global energy per pixel E_p/e_t versus the block length m. (a) Sensorcam, $e_t/e_h = 3913$. $m^\star = 2$. (b) Mica2, $e_t/e_h = 22$. $m^\star = 1$.*

The long-range radio transmission power is 2.86W, and the transmission speed is 16kbps. Therefore,

$$e_t = \frac{2.86\text{W}}{16\text{kbps}} = 1.8 \times 10^{-4}\text{J/bit} \quad \therefore e_t/e_h = 3913.$$

By solving (5.9) and rounding to the nearest integer,

$$m^\star = 2.$$

Figure 5.2a shows the normalized global energy per pixel E_p/e_t with respect to the block length m. We can see that the global energy indeed achieves a minimum at $m = 2$.

Mica2 Platform: The *Mica2* platform is composed of an ATmega128 microprocessor and a CC1000 radio. The power consumption of the ATmega128 is 22mW and its CPU frequency is 8MHz. The power consumption of the CC1000 is 69mW in transmission mode and its transmission rate is 38.4bps. Similar calculation as before gives

$$e_h = \frac{0.8\text{ms}}{256 \times 256} \times \frac{2.4\text{GHz} \times 22\text{mW}}{8\text{MHz}} = 8.1 \times 10^{-8}\text{J}.$$

$$e_t = \frac{69\text{mW}}{38.4\text{kbps}} = 1.8 \times 10^{-6}\text{J} \quad \therefore e_t/e_h = 22.$$

By solving (5.9) and rounding to the nearest integer,

$$m^\star = 1.$$

Figure 5.2b shows E_p/e_t versus m for *Mica2*. Comparing to the *Sensorcam* platform, the optimal block length $m^\star$ is smaller because e_t/e_h of *Mica2* is smaller (one can verify this from (5.9)). Therefore, as e_t/e_h decreases, the benefit of computation decreases. Lee et al. [12] studied a similar computation-communication tradeoff for

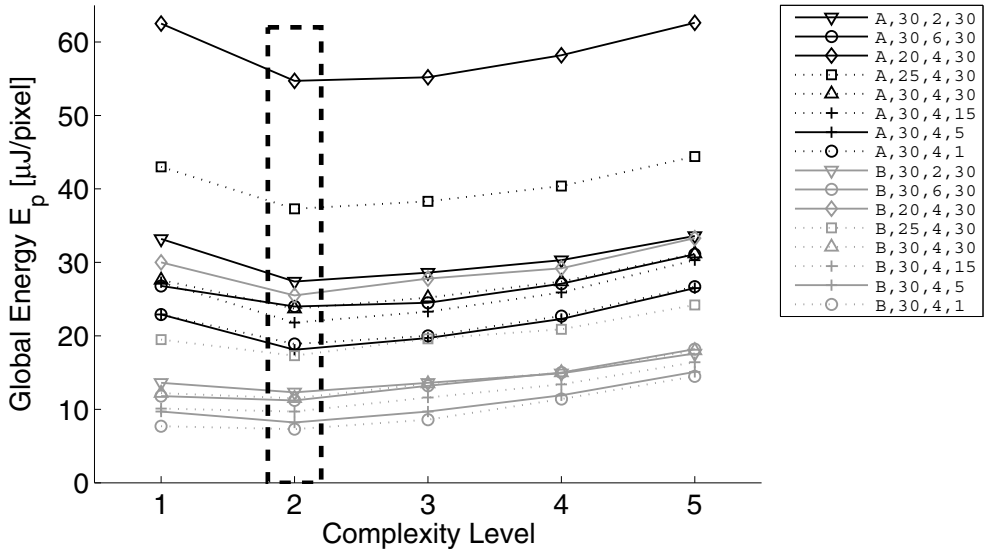


Figure 5.3: *The global energy E_p versus the complexity level of an H.264 video coder. The experimental settings include four tunable parameters, namely, dataset name, video quality, group of pictures (GOP), and sampling interval T_s in minutes.*

JPEG lossy coding, and concluded that JPEG compression brings no benefit on the *Mica2* platform.

In summary, the comparisons in Figure 5.2 show that the optimal choice of the computation-communication combination is effected by the energy efficiencies of both computation and communication hardware. More specifically, by inspecting the ratio between the two energy efficiencies (e.g., e_t/e_h), we can determine whether a higher computational complexity can effectively reduce energy consumption.

5.2.4 Case Study II: H.264 Video Coding

We analyze now the complexity tradeoff of the H.264 encoder. As introduced in Section 4.2.1, the H.264 encoder provides a number of preset options to tradeoff the compression ratio against the encoding speed. For notation purposes, we label these complexity presets as level 1 to 10, where level 1 has the lowest complexity, and the complexity increases with the preset number.

The experimental setup is similar to the one in Section 4.4. We implemented a H.264 video codec on the *Sensorcam* platform, based on an open-source H.264 codec [51]. The two video datasets Scene A and Scene B (Figure 4.10) are used for benchmarking purposes. The global energy is computed as the sum of computation and communication consumptions, estimated from the hardware specifications in Table 4.3. Figure 5.3 shows the global energy versus the complexity for different experimental settings characterized by four tunable parameters, namely, dataset name, video quality, group of pictures (GOP), and sampling interval T_s in minutes (as defined in Section 4.4).

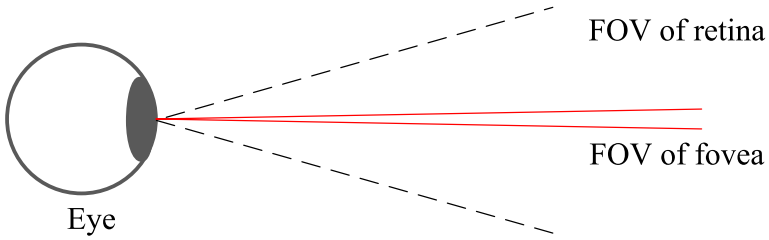


Figure 5.4: *Human visual system. The details are primarily captured on the fovea, which covers approximately* 2° *of FOV. The FOV of the entire retina is much larger, but with relatively low spatial resolution.*

It is interesting to observe that the optimal choice of the H.264 complexity always falls at level 2. This is actually consistent with the analysis in Section 5.2.3: We have shown that for a given algorithm (Huffman coding) and a given hardware platform (*Sensorcam* or *Mica2*), the optimal computation-communication combination is unique and determined by e_t/e_h. Similarly, for H.264 coding and *Sensorcam*, as both the coding method and the hardware platform are fixed, the optimal computation-communication combination is also unique.

5.3 Event-Driven Video Coding (EVC)

As we mentioned in Section 5.2.2, besides the energy efficiency of hardware, the efficiency of the compression algorithm is also critical to reduce global energy consumption. From a traditional rate-distortion point of view, the state-of-the-art H.264 algorithm is hard to surpass. However, as in the human visual system, it is not always necessary to transmit the complete image.

In the human visual system (Figure 5.4), most of the visual detail is captured on the fovea, which covers approximately 2° of field of view (FOV). However, the FOV of the entire retina is much larger than this. The visual system of human works as following:

1. The image captured on the retina (except fovea) has a low spatial resolution, and can be communicated to the brain at a relatively low bandwidth.

2. The image captured on the fovea has a high spatial resolution (but only covers a small region). Most of the communication bandwidth between the brain and the eye is used to transmit this image.

3. The brain searches potential areas of interest in the overall image (low resolution), and the eyes are controlled to focus the line of sight on the point of interest, so that it falls on the fovea.

In monitoring applications, it is often only necessary to transmit the detected events to the user. Therefore, a vision-like video coding can potentially achieve a lower global energy by allowing the camera to analyze the captured video and transmit only the events of interest within each image.

Figure 5.5: *In this example, we consider the non-sky part as the region of interest (ROI). An automated algorithm can find the boundary (the yellow line) between the sky and the foreground buildings/mountains.*

In this section, we propose an event-driven video coding (EVC) algorithm inspired by the threshold-based video coder (Section 4.2.1). To detect events of interest, we analyze each image frame using a saliency detection algorithm that determines whether a pixel belongs to an event and therefore has to be transmitted to the end-user. In this way, we can transmit substantially less bits by ignoring most of the irrelevant changes in image sequences, whereas the image fragments with events are delivered to the receiver. It is designed to achieve the following targets:

- Reduce global energy consumption without reducing the event detection probability.
- Require no supervised learning procedure.
- Simple to implement on an embedded platform.

In Section 5.3.1, we first define the region of interest (ROI) in the EVC framework. Then, in Section 5.3.2, we introduce the "brain" of EVC — saliency detection. We propose a simple yet effective approach based on edge detection, which does not require supervised learning and is robust to light variations. Finally, in Section 5.3.3, we present the overall encoding and decoding procedures of EVC.

5.3.1 Region of Interest

The region of interest (ROI) is a mask indicating which part of the image we actually monitor (e.g., the entrance of a building). A monitoring camera usually captures more information than the ROI, and a simple strategy to reduce energy is to transmit only the ROI of the image. As the monitoring camera is usually static, the ROI of each camera is also fixed. We can manually label the ROI for a camera, or use an automated algorithm to identify it from the captured images. The later approach is application-specific and requires different algorithms depending on how we define the ROI.

We now give an example of an automated algorithm for a common situation in outdoor monitoring applications. Assume we are monitoring an area where the sky does not belong to the ROI. We propose an automatic sky extraction algorithm based on two observations. First, clouds in the sky are dynamic, while the camera and the non-sky part of the image are comparatively static. By calculating the cumulative residuals from successive images, we can discriminate the sky from the foreground through the residual value. Second, the sky is normally at the top of the image, and the non-sky part is at the bottom. By weighting the residuals according to the

Algorithm 5.1 Saliency detection (refer to Figure 5.6 for the italic words)

Require: *cur*: the current image frame.
Require: *ref*: the previous image frame.

1. A *residual* is generated as $cur - ref$.
2. The *residual* is masked by the ROI (Section 5.3.1).
3. The *salient edge* is calculated from the *residual* using a Canny edge detector [59], and then masked so that the over-exposure part is reset to zero.
4. The *salient distance* is calculated as the distance transformation [60] of the *salient edge*.
5. The absolute value of the *residual* is divided by the *salient distance* (per-pixel calculation). Thresholding is applied to the result to get an initial *saliency map*.
6. The *saliency map* is refined with some post-processing, such as removing small fragments, filling holes, etc..

vertical height, image fragments with high variations in the foreground will not be misclassified as sky. Figure 5.5 shows an example of the sky extraction result. It can be seen that most of the foreground buildings and mountains are well separated from the sky.

5.3.2 Saliency Detection

The core concept of the proposed event-driven video coding scheme is to transmit only the image fragments that contains events. We denote by *saliency map* the binary map indicating salient pixels. Naturally, we only consider observable events, that is, events captured by the camera which are visually distinguishable. One option to obtain the saliency map is to apply thresholding to the residual of successive image frames. However, due to the changes in light condition, the correct threshold value can vary within the same image. To avoid this problem, we propose to use the edge of the residual to detect the shape of salient regions. As the edge detection algorithm [59] is based on the local gradient value, it is more robust to intensity changes.

Figure 5.6 gives a block diagram of the proposed saliency detection algorithm. The raw video stream is fed into the algorithm from the left side. The residual image is first calculated by subtracting the preceding image frame. After applying the ROI mask to the residual image, the salient edges are computed by using a Canny edge detector [59]. To eliminate the interference of the sunlight, over-exposed pixels in the salient edges are cleaned up. In order to expand the shape of salient regions into solid salient regions, we first compute the distance transformation [60] of the salient edges, and then combine the distances to salient edges and the residual values to represent the saliency strength of each pixel. By thresholding, the final saliency map is generated. Post-processing techniques such as binary morphology operators can be used to refine the results. The implementation of the algorithm is shown in Algorithm 5.1.

By construction, the edge-detection based saliency detection algorithm we propose does not require supervised learning, is robust for detecting small/medium objects, and is computationally efficient. However, it can be affected by noise in the edge detection procedure, and miss some fragments inside a large continuous area.

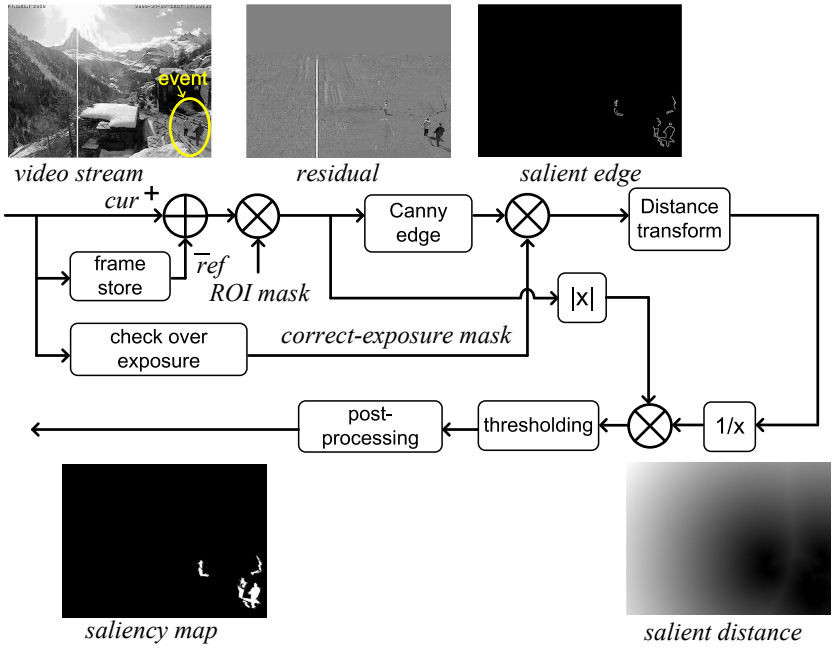


Figure 5.6: *Flow chart of the proposed saliency detection algorithm.*

Existing background subtraction algorithms [61] can also be used as the saliency detection algorithm in the EVC framework. By using a more sophisticated background subtraction algorithm, the EVC can potentially be made more robust. Nevertheless, the dramatically changing light condition and the shadows in outdoor scenarios are common problems to all background subtraction algorithms. This effect is even more critical in monitoring applications, where the sampling frequency is significantly lower than in regular videos (<0.02 fps versus 30 fps).

5.3.3 Coding Procedure

After the saliency map is generated for each image frame, we now explain how to integrate it in the coding procedure of EVC. Similar to other video coding schemes, we define the group of pictures (GOP): A key-frame that is independent of neighboring frames is compressed using a conventional image coding every GOP frames. Without loss of generality, we use JPEG codec for the key frame. The coding procedure is shown in Figure 5.7: For each frame (except key-frame), we divide the generated saliency map into blocks of 8×8 pixels[3], and find the blocks that contain salient pixels (denoted as *event blocks*). If the percentage of the *event blocks* is more than 10%, then we encode the current frame using DPCM [62]. Otherwise, all the *event blocks* are concatenated into an image strip for compression and transmission. In this case, we also need to send the locations of *event blocks* (position array) so that the

[3]This is to meet the block size of the JPEG standard.

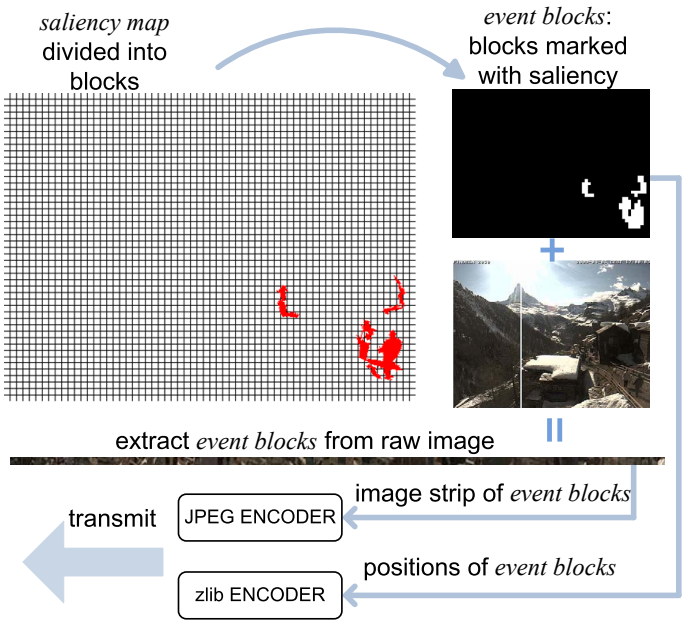


Figure 5.7: *Illustration of the event-driven video encoding procedure. Event blocks are extracted according to the* saliency map*, and are concatenated into an image strip for transmission.*

reconstruction is possible. In practice, a standard data compression scheme such as zlib[4] can be used to reduce the size of the position array before transmission. The detailed algorithm for the encoding procedure is shown in Algorithm 5.2.

At the receiver side, the decoding procedure is straightforward by following the encoding procedure (omitted for brevity). It is worth to mention that when merging the *event blocks* with the background image, it is actually an image blending problem. To reduce the blocking effect, we can first create a blending mask using the position array of *event blocks*, and then apply blending algorithms such as pyramid blending [63] or Poisson image editing [64] to achieve seamless stitching.

5.4 Experimental Results of Single-Camera EVC

In this section, we evaluate the proposed event-driven video coding (EVC) algorithm and compare its compression efficiency with that of Motion JPEG, H.264 and EVC. All algorithms are evaluated with the image sequence "Zermatt" captured by an outdoor webcam. The dataset consists of 300 frames, recorded at a sampling interval of 5 minutes. With the same dataset, we compare EVC's event detection rate with an anomaly-detection based video coding scheme. Finally, we implement EVC on the *Sensorcam* platform and measure the overall energy consumptions in a real deployment.

[4]See http://zlib.net/.

Algorithm 5.2 Event-driven video encoding (EVC)

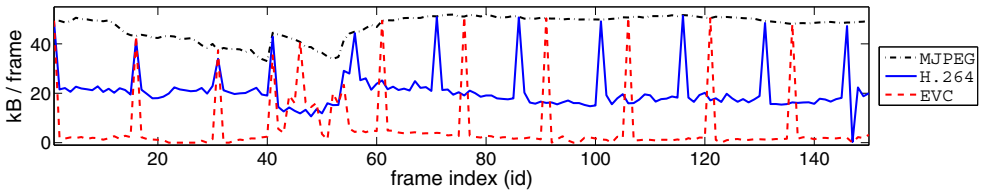
GOP: group of pictures, *mode*: one-byte header, k: counter (initialized to 1)
bh=image height/8, bw=image width/8
$pos[bh][bw]$: position array
mem: decoding buffer (initialized to all-zero)
ref: previous frame (initialized to all-zero)
while a new image frame *cur* is captured **do**
  if (k mod GOP) == 1 **then**
    compress *cur* by using JPEG encoder, *mode*=1
  else
    reset array *pos* to all-zero
    get a *saliency map* from *cur* and *ref* (Algorithm 5.1), and divide it into $bh \times bw$ blocks
    locate the blocks marked with saliency, and set corresponding element in array *pos* to 1
    if sum(pos)/($bh \times bw$) $> 0.1(10\%)$ **then**
      compress $cur - mem$ by using JPEG encoder, *mode*=2
    else
      extract *event blocks* from *cur* according to array *pos*
      concatenate *event blocks* into an image strip, and compress it by using JPEG encoder, *mode*=3
    end if
  end if
  transmit the JPEG codeword to the receiver
  if *mode*==3 **then**
    compress array *pos* by using zlib, and transmit the codewords to the receiver
  else
    transmit one-byte *mode* to the receiver
  end if
  update *mem* by reconstructing the current frame from the transmitted codewords
  $ref = cur$
end while

5.4.1 Compression Rate

We evaluate now the compression rate of three video coding schemes: Motion JPEG, H.264 and EVC. We set the compression qualities of all algorithms to the same value, and the GOP for EVC and H.264 to 15[5]. Note that we only consider the image quality of blocks where the event is detected, since other parts of the image are often irrelevant for monitoring applications. We record the sizes of the encoded frames generated by the three video coding schemes on the first 150 frames of "Zermatt" dataset[6].

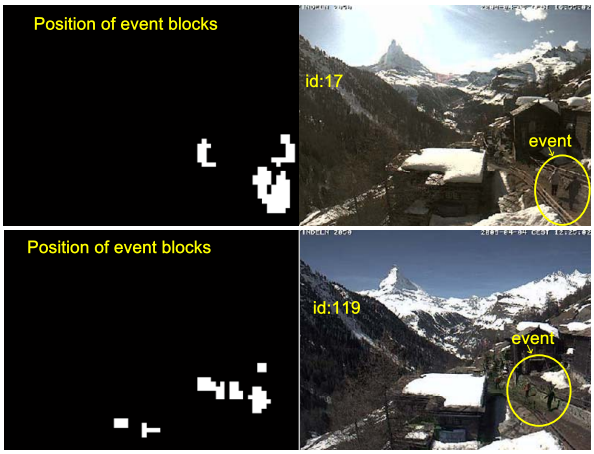
[5]Similarly to the experiments in Section 4.4, the GOP value of interest is lower than in regular videos, because the sampling rate of monitoring applications is lower.

[6]The last 150 frames are used as training data in Section 5.4.2



(a)

(b)



(c)

Figure 5.8: *Simulation results of Motion JPEG (MJPEG), H.264, and EVC. (a) Compression rate, measured in kilobyte per frame, of the three schemes. Each spike in the curve represents a key-frame or DPCM frame. (b) A reference image (id=1) with no event. (c) Two images with events decoded by EVC (id=17, 119), and the corresponding positions of the transmitted event blocks.*

Figure 5.8a shows the compression rate (measured in kilobytes per frame) of the three schemes in time. It can be seen that EVC performs substantially better than the other video coding schemes. The average frame size of EVC is 70% smaller than that of H.264. Figure 5.8c shows also some sample reconstructions of EVC with and without detected events. The positions of the transmitted image fragments are also illustrated, which clearly indicate the detected events. Note that the transmitted key-frames (spikes in Figure 5.8a) provide the background image in reconstruction. The *event blocks* are pasted onto the most recent key-frame.

5.4.2 Detection Rate

Another approach to event-driven video transmission is to determine which frame does contain an event and only to transmit such event-frames. This idea relates to the research on anomaly detection [58], which utilizes pattern recognition techniques to find patterns in data that do not conform to regular behavior. Similar to other pattern recognition algorithms, there are two main drawbacks to this technique:

Algorithm 5.3 Anomaly-detection based video coding (AnVC)

1. Extract PHOW features of residuals of successive images (See [65] for detailed feature extraction algorithm).
2. Select the **last** 150 images of "Zermatt" dataset as a training set, and manually label the images that have events of interest.
3. Train a support vector machine (SVM) [66] by using the training set.
4. Select the **first** 150 images of "Zermatt" dataset as a testing set. Among all images in the testing set, recognize the ones that contain events of interest, by using the trained SVM.
5. Compress all recognized frames with H.264 codec, and transmit the codeword.

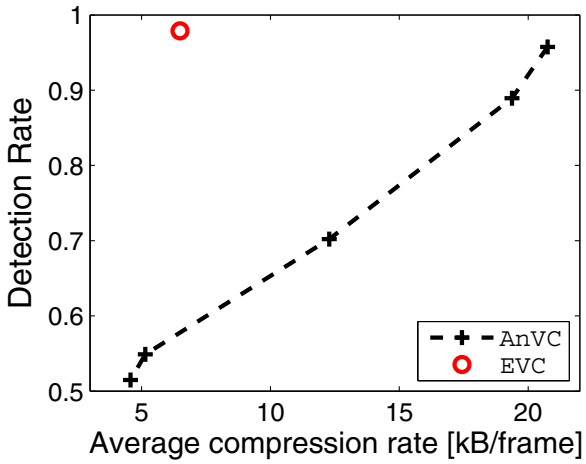


Figure 5.9: *Detection rate versus average encoded frame size for EVC and AnVC respectively.*

1. Some events will be missed, since recognition does not work 100% accurately.

2. Pattern recognition algorithms usually require training data. To collect training data is time consuming, and the collected data can only be used for one specific scene.

In this section, we compare the detection rate of EVC with a video coding scheme that is based on anomaly detection [58]. The implementation of the anomaly-detection based video coding (AnVC) is described in Algorithm 5.3.

We define an event as the appearance of one person in the image, and the *detection rate* as the ratio between the number of people detected at the receiver and the true number. By varying the threshold in SVM decision, AnVC offers a tradeoff between the average compression rate and the detection rate. Figure 5.9 shows the comparison result between EVC and AnVC. With the same compression ratio, EVC almost doubles the detection rate of AnVC.

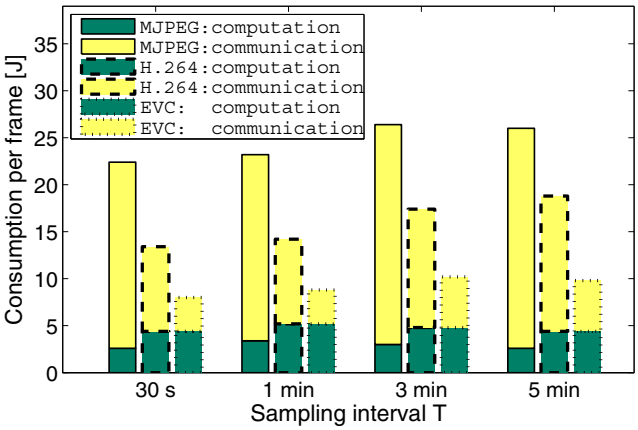


Figure 5.10: *Energy consumption per frame with respect to sampling interval T, for Motion JPEG (MJPEG), H.264 and EVC on Sensorcam. The two parts of each bar represent the computation (green) and the communication (yellow) energy consumptions, respectively.*

5.4.3 Algorithmic Complexity and Global Energy Consumption

We implemented EVC on the *Sensorcam* platform (Section 4.B) using C/C++. The compressed bitstreams are transmitted by GPRS to the end-user. As it takes about 30 seconds to re-establish a GPRS connection (GPRS module is put to sleep between consecutive transmissions), to reduce the radio startup energy, we group the data of 10 frames and transmit them together. For comparison purposes, we also cross-compile for *Sensorcam* the open source H.264 and JPEG algorithms.

The compression quality of Motion JPEG, H.264 and EVC are set to the same value. The GOP of EVC and H.264 are set to 10. We connect an energy meter at the power source to measure the energy consumption[7]. We deploy the system to monitor a courtyard that occasionally has pedestrians and vehicles, and repeat the experiment with different coding schemes and sampling intervals. Figure 5.10 shows the global energy consumption for processing (green) and transmitting (yellow) one frame.

We can see from Figure 5.10 that the computation energy (algorithm complexities) of H.264 and EVC, are approximately the same, whereas the global energy consumption decreases substantially as the "intelligence" of the camera increases. This supports our previous argument: A vision-like video coding achieves lower global energy than traditional rate-distortion-optimal coding algorithms.

In terms of averages, the global energy consumption of EVC is 40% smaller than that of H.264. Note that this saving is smaller than previous simulation results (70%) in Figure 5.8a. This is because Figure 5.8a does not take into account the computation

[7]Due to hardware problems, *Sensorcam* constantly consumes 33mW of background power in sleep mode. We have subtracted this background energy from the global energy, because technically it can be eliminated by suspending *Sensorcam* during the standby period (currently not implemented).

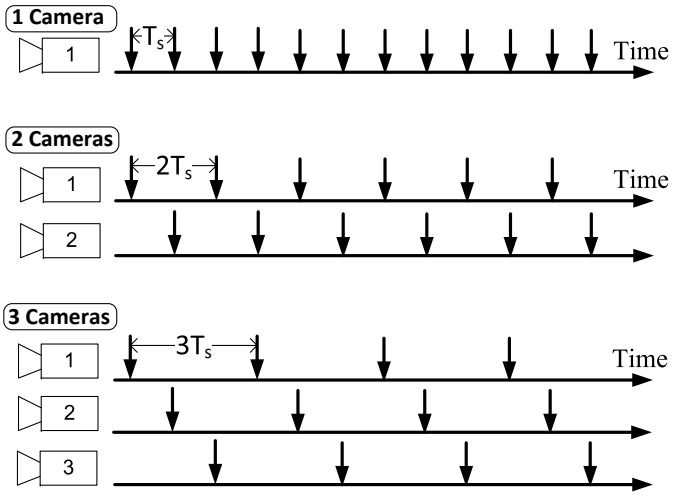


Figure 5.11: *A n-camera system (for n =1,2, and 3) using the cooperative sampling framework with an overall sampling frequency of* $1/T_s$. *Arrows indicate the time when a camera captures an image. Each camera has a sampling interval of* $T = n \cdot T_s$.

energy, which actually has a significant share in the global energy consumption in practical scenarios.

5.5 Cooperative Monitoring with Distributed Smart Cameras

In this section, we further develop the multi-camera cooperative monitoring system by combining the *event-driven video coding (EVC)* (Section 5.3) with the *cooperative sampling framework* (Section 2.2).

We consider an event monitoring system composed by n cameras placed in a distributed manner around the area of interest, and focusing at a common scene. This multi-camera system monitors the scene every T_s minutes and deliver all detected "events" back to an end-user through wireless communications. As shown in Figure 5.11, for a n-camera system that employs the cooperative sampling framework, each camera has a sampling interval of $T = n \cdot T_s$.

We study in this section the scaling behavior of the per-camera energy cost (e.g., joule per day per camera) with respect to the number of cameras n. As the sampling frequency of each camera $f = 1/T$ is inversely proportional to the number of cameras n, the ideal scaling behavior is $1/n$. In practice, we found that the per-frame **computation** cost of each camera is generally static with respect to the sampling interval T (dark green bar in Figure 5.10). Therefore, we can expect that the per-camera computation cost scales exactly as $1/n$. However, as the inter-frame variance increases with T, the per-frame **communication** cost increases with T (light yellow bar in Figure 5.10). Hence, it is to be expected that the per-camera communication cost scales slower than $1/n$.

To verify the realistic performance of EVC on a multi-camera system, we deployed nine time-lapse cameras and used the captured dataset to simulate the video coding algorithm. In Section 5.5.1, we first introduce the experimental setup. In Section 5.5.2, we then apply the single-camera EVC to the dataset, and show that the scaling behavior of per-camera communication cost has an extra term of $\mathcal{O}(\log(n)/n)$. This is further supported by the rate-distortion analysis on a Gaussian model. Finally, in Section 5.5.3, we exploit the unique structure of EVC as opposed to conventional prediction-based video codecs, and propose an improved EVC scheme for a distributed multi-camera system. With the same dataset, the performance loss is eliminated.

5.5.1 A Large Scale Multi-Camera Dataset

We deployed nine time-lapse cameras on the roof of three adjacency buildings (Figure 5.12a). All cameras are focusing at a courtyard with pedestrians and cars entering and leaving. The cameras are deployed as sparse as possible across the three buildings, due to the robustness concern as indicated in Section 1.2. Figure 5.12b shows the cameras we used to capture the dataset. Each camera can be programmed to capture images periodically at a given interval and stores the data locally on a SD card. With four AA batteries, it is able to record around 7000 images, that roughly correspond to 7 days of working life, recording one picture per minute between 6am to 9pm.

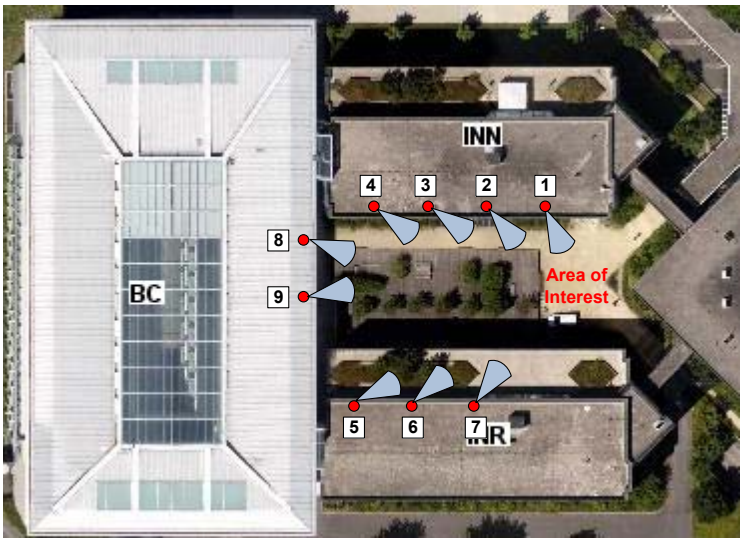
In order to simulate the multi-camera system under the cooperative sampling framework, we manually synchronize all the cameras by calibrating their clocks. All cameras start simultaneously and sample once per minute, generating a raw dataset of nine images sequences from the nine cameras, respectively. We can get datasets at different sampling interval T_s and different number of cameras n, by selecting the appropriate subset of images from the raw dataset.

Figure 5.13 shows sample images from the dataset. We can see that the nine cameras monitor a common area of interest, depicted in Figure 5.12a. Although the captured event — the white car — has various appearances and sizes as seen by different cameras, it is indeed detected by all cameras simultaneously, due to the global synchronization. It is worth mentioning that not all cameras are perfectly static: As the camera are deployed outdoors, wind induces vibrations that affect the recorded image sequence. Such realistic phenomena exist commonly in real scenarios, and will bring extra communication cost for EVC.

As the camera position is fixed throughout the deployment, we manually set the region of interest (ROI) for each camera. In the following simulations, we use the dataset recorded during 10h00-20h00 on June 19, 2012.

5.5.2 Scaling Behavior of Per-Camera Communication Cost

We study scaling behavior of per-camera communication cost by simulating the single-camera EVC on each camera independently. The overall rate of the compressed datastream is recorded. As the communication cost is proportional to the transmitted data rate, we calculate the average per-camera data rate and use it as a measure of per-camera communication cost. Figure 5.14 shows the per-camera communication cost for different number of cameras n, normalized with respect to the cost of the



(a)

(b)

Figure 5.12: *Experimental setup for data acquisition: (a) The setup of nine cameras (red dots) on roof of three buildings, monitoring the courtyard. The ID and the view direction of each camera are marked around the red dot. (b) The time-lapse cameras that are deployed on the roof (ID=2,3,4).*

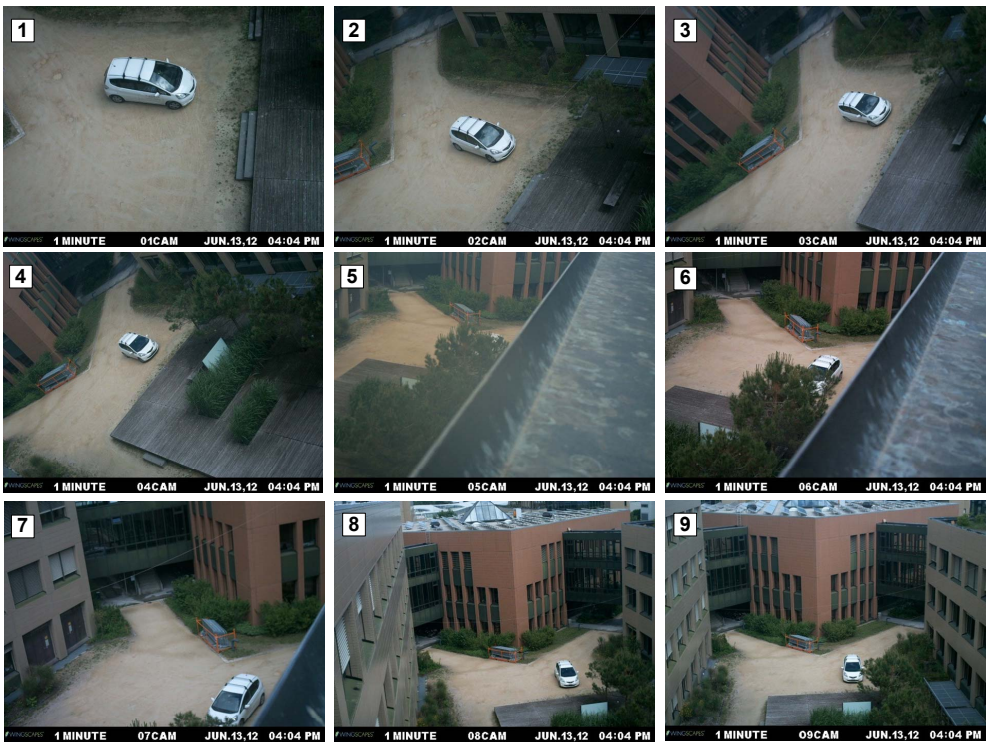


Figure 5.13: *Images captured by the nine cameras at 16h04, June 13, 2012. A white car was recorded by all cameras as an event. The ID of camera is printed on the upper left corner of each image.*

single-camera case $n = 1$. The dashed curve below the solid curve represents the ideal scaling of $1/n$. We can see that there is a significant gap between them.

This phenomenon can be explained by the theoretical rate-distortion analysis. Like most video coding methods, EVC also encodes the innovation between consecutive frames. If we model this innovation as a Gaussian source, then we can calculate the required rate from the classical rate-distortion function of a Gaussian source [24]:

$$R(D) = \frac{1}{2} \log_2 \frac{\sigma^2}{D}, \tag{5.10}$$

where σ^2 is the variance of the Gaussian source.

Theorem 5.1. *Assume the innovation between adjacency frames (interval T_s) is modeled as an independent and identically distributed (i.i.d.) Gaussian source with zero mean and variance σ^2. Then the ratio of the overall communication cost between a n-camera monitoring system and a single-camera monitoring system scales as $1 + \mathcal{O}(\log n)$ asymptotically.*

Proof. As the sum of i.i.d. Gaussian sources is still a Gaussian source, the innovation between a frame at time t and a frame at time $t - l \times T_s$ is a Gaussian source with zero mean and variance $l \times \sigma^2$.

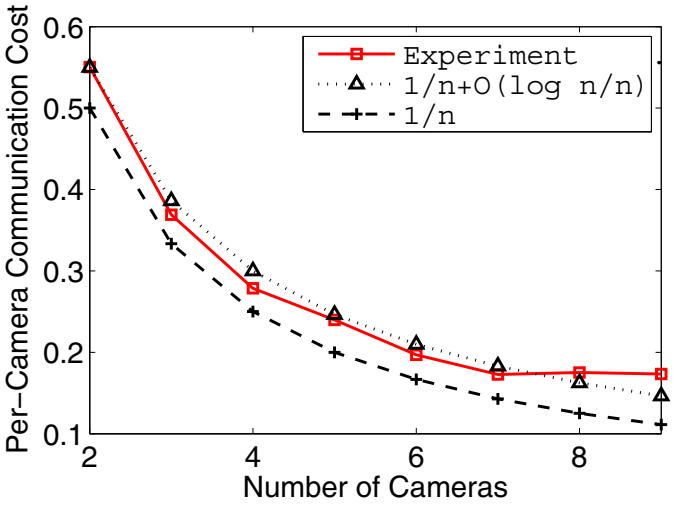


Figure 5.14: *Per-camera communication cost with respect to the number of cameras n. The communication cost is measured by the average data size per camera after EVC coding, and is normalized w.r.t. the cost of the single-camera case $n = 1$. The solid curve represents the experimental result by applying single-camera EVC independently to each camera. The dashed curve represents the ideal scaling $1/n$. The dotted curve represents a scaling of $1/n + 0.05 \cdot \log_2 n/n$.*

We consider a total of n frames to be encoded:

For a n-camera monitoring system under the cooperative sampling framework, each camera encodes its own frame at time t based on the previous frame at time $t - n \times T_s$. Therefore, the innovation can be modeled by a Gaussian source with variance $n\sigma^2$. From the rate distortion function of a Gaussian source (5.10), the overall rate is

$$R^{(n)} = n \cdot \frac{1}{2} \log_2 \frac{n\sigma^2}{D}.$$

For a single-camera monitoring system, a frame at time t is encoded based on the previous frame at time $t - T_s$, and totally n consecutive frames are encoded. As the innovation between two adjacency frames is a Gaussian source with variance σ^2, the overall rate is

$$R^{(1)} = n \cdot \frac{1}{2} \log_2 \frac{\sigma^2}{D}.$$

As the communication cost is proportional to the coding rate, the ratio of the overall communication cost between a n-camera monitoring system and a single-camera monitoring system is

$$\frac{R^{(n)}}{R^{(1)}} = 1 + \frac{\log_2 n}{\log_2(\sigma^2/D)}.$$

Since for a given distortion requirement D, $\log_2 \sigma^2/D$ is a constant, the ratio scales as $1 + \mathcal{O}(\log n)$ asymptotically. □

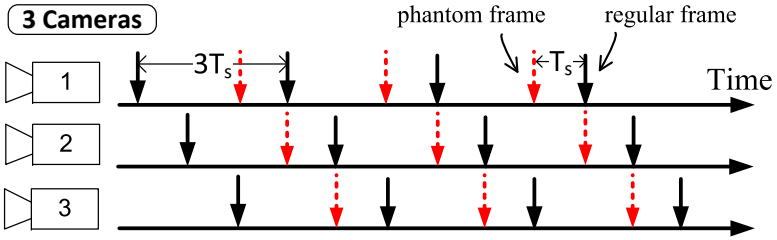


Figure 5.15: *EVC for a distributed multi-camera system: The black solid arrow represents the regular frame that is captured in the original cooperative sampling framework. In the new proposed scheme, each camera captures one extra image in every sampling interval, denoted phantom frame (the red dashed arrow). The time interval between the phantom frame and the regular frame is fixed to T_s.*

By averaging the overall cost with the number of cameras n, Theorem 5.1 states that the per-camera communication cost scales as $1/n + \mathcal{O}(\log n/n)$. To verify this, we plot a scaling of $1/n + 0.05 \cdot \log_2 n/n$ in Figure 5.14 (the dotted curve). We can see that it approximately follows the experimental results. This justifies that if we use EVC for each camera independently, there is an extra cost for a multi-camera system as compared to a single-camera system.

5.5.3 EVC for a Distributed Multi-Camera System

We have already shown in Section 4.4 that joint coding in a multi-camera system suffers from various overheads such as registration errors caused by missing depth information of the monitored scene. In this section, we stick to the distributed approach where cameras do not communicate with each other, and propose an improved multi-camera EVC scheme that exploits inter-camera correlations to reduce the distributed loss (Figure 5.14).

Generally, the inter-frame variance increases with the time interval between two frames. Hence, in a distributed multi-camera system, each camera suffers a loss in compression efficiency as the sampling interval $T = nT_s$ increases with the number of cameras. The method we propose focuses on keeping a constant time interval between encoded frames. As shown in Figure 5.15, on top of the cooperative sampling framework, we let each camera capture one extra image in every sampling interval (red dashed arrow), which we call a *phantom frame*. The time interval between the phantom frame and the regular frame (black solid arrow) is fixed to T_s. This extra image acquisition in practice requires little computation cost, and since it is only used locally, it adds nothing to the communication cost .

Note also that phantom frames are captured at the same time of regular frames of other cameras (Figure 5.15). This implies that any new event found in a phantom frame will be also detected and delivered to the end-user by another camera. Hence, it is of no interest to transmit a phantom frame. However, we use the phantom frame as the reference frame (*ref* in Algorithm 5.2) for computing the saliency map in the EVC algorithm. In this way, we virtually reduce the time interval between encoded

Algorithm 5.4 Encoding of the multi-camera EVC

1. Calculate a saliency map from the *current frame* and the *previous regular frame*, and generate the corresponding position array $\mathcal{S}_A$
2. Calculate a saliency map from the *current frame* and the *previous phantom frame*, and generate the corresponding position array $\mathcal{S}_B$
3. Extract *event blocks* from the *current frame* according to $\mathcal{S}_B$
4. Transmit *event blocks* to the receiver
5. Compress and transmit both $\mathcal{S}_A$ and $\mathcal{S}_B$ to the receiver

Algorithm 5.5 Decoding of the multi-camera EVC

Require: Current $\mathcal{S}_A$, $\mathcal{S}_B$ and *event blocks*

1. decoding buffer: *mem*
2. Find $\mathcal{S}_R$ as the intersection of $\mathcal{S}_A$ and the previous $\mathcal{S}_B$
3. Revert the blocks of *mem* that is marked by $\mathcal{S}_R$ to previous values
4. Paste the *event blocks* to *mem* according to the positions marked by $\mathcal{S}_B$

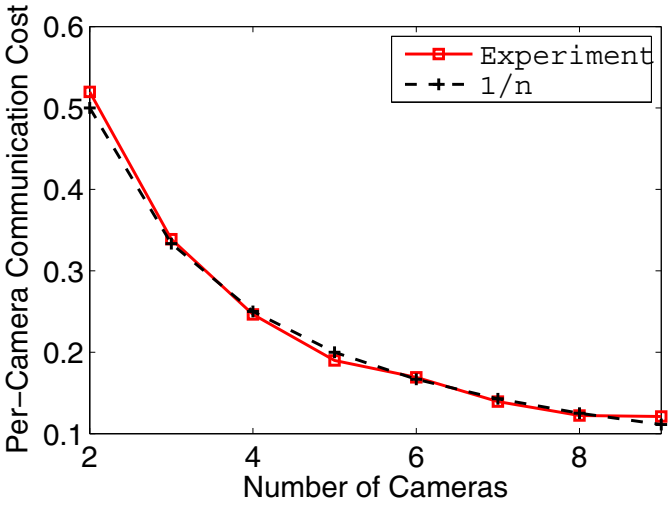


Figure 5.16: *Scaling (solid curve) of per-camera communication cost by using the multi-camera EVC. It scales the same as* $1/n$.

frames from nT_s to T_s.

In the encoding procedure, we calculate two saliency maps for each frame, one from the previous phantom frame, and one from the previous regular frame. We use the first saliency map to detect the new events, and use the second saliency map to detect the old events that are no longer active. To erase the old events at the receiver, we only need to send the positions of the erased blocks (not content), and the receiver can then revert these blocks to previous values. Algorithm 5.4 and Algorithm 5.5 briefly summarize the encoding and decoding procedures of the proposed multi-camera EVC, respectively.

Figure 5.16 shows the per-camera communication cost obtained by using the multi-camera EVC algorithm, with the same experimental setup as in Section 5.5.2. We can see that as opposed to Figure 5.14, the per-camera communication cost scales as $1/n$, which verifies the effectiveness of our proposed algorithm.

As we mentioned before, the per-camera computation cost also scales as $1/n$. Hence, the per-camera global energy consumption will scale as $1/n$, by using the multi-camera EVC algorithm.

5.6 Summary

- Computation can help to push compression rate to the rate-distortion limit, and thus reduce the communication cost. Meanwhile, more computation requires more energy. From a global energy point of view, there is a tradeoff between computation and communication.
- The optimal computation-communication combination depends on the energy efficiency of the hardware platform and the compression efficiency of the coding algorithm. Once these two factors are fixed, the optimal choice of combination is unique.
- Mimicking the human visual system, event-driven video coding (EVC) uses image processing techniques to detect salient regions in an image, and only transmits image fragments marked with saliency.
- The experimental results show that EVC achieves a better compression ratio than H.264. Comparing with the anomaly-detection based video coding scheme, EVC also achieves a better detection rate with the same compression ratio. The implementation of EVC on a wireless smart camera shows that it has a similar computational complexity as H.264, while consuming less energy.
- We study a n-camera cooperative monitoring system by incorporating single-camera EVC into the cooperative sampling framework. Given a fixed sampling target of f_s, the ideal scaling behavior of per-camera communication cost is $1/n$. However, as the inter-camera correlation is not exploited in independent coding, the experimental result on a large-scale multi-camera dataset shows that there is an extra loss term of $\mathcal{O}(\log(n)/n)$.
- We propose an improved multi-camera EVC scheme that exploits the inter-camera correlation with only local information (cameras are completely distributed). The experimental result on the same multi-camera dataset shows a scaling behavior of $1/n$, which is ideal.

Chapter 6

Conclusions and Future Work

We presented a general sampling and coding framework for wireless visual monitoring systems, aiming at distributing the physical risk and energy burden of the monitoring task among many small, cheap, and autonomous camera nodes. We showed that the optimal sampling configuration is uniform, and developed a distributed algorithm that allows the cameras to adaptively form the optimal configuration. This approach requires a specific network topology, namely, the Hamiltonian condition. The extension of this algorithm is to make better use of the collected neighbouring information for relaxing the constraint on network topology. Another property of the current sampling framework is its uniform configuration. The optimality of this setup holds when we do not know any prior on the occurring events. If we have access to such prior information, a non-regular sampling setup will be a better choice. In this line of research, we will consider to use a centralized synchronization approach to coordinate the cameras, which can be more flexible to adjust.

Based on the cooperative sampling framework, we first studied cooperative coding of multi-view images. The proposed successive refinement coding imitates the ping-pong game, and is successively refinable on the theoretical distributed coding bound for the Gaussian case. Based on the layered decomposition and linear prediction method, we also applied the same idea to encode stereo-view images. The extension from the two-encoder case to more encoders remains a challenging task. Currently, the only known rate-distortion region of distributed source coding is also for two encoders.

We investigated cooperative coding of multi-view videos by using three video coding approaches (distributed/independent/joint). The evaluation results on a two-camera system show that in a long-range communication scenario, joint coding outperforms independent coding when the sampling rate is low. The limited energy saving of joint coding pushed us to further exploit the processing power of smart cameras. We demonstrated that the "intelligence" of a camera brings substantial energy savings in event detection applications. The simulation results, as well as real-world deployments, show that the proposed event-driven video coding (EVC) performs substantially better than conventional video coding schemes. We also investigated the combination of EVC with the cooperative sampling framework, and proposed a distributed multi-camera EVC scheme that effectively eliminates the distributed coding

loss. Future work on improving the global energy efficiency will take into account the sensing energy. In some scenarios, the sensing devices can also consume comparable energy with respect to computation or communication.

One of the main assumptions in this thesis is the single-hop network infrastructure. In the next step, we will develop the proposed concepts/algorithms into a more general multi-hop scenario. In such a case, we need to further incorporate routing protocols, transmission-delay handling, and multi-terminal source coding into the system design. Furthermore, in this thesis we studied the scenario where multiple cameras focus at a common scene of interest. A natural extension is to study the case where cameras have limited overlapping views. In this case, the correlation between the cameras is no longer the overlapping pixel values, but the information they infer. For instance, we can infer a city-wide atmospheric visibility map, based on the images collected by many mobile users across the city.

To infer information from images/videos, modern computer vision algorithms have investigated various aspects of estimation problems in real-life, such as recognition, segmentation, tracking, or scene reconstruction. Typically, those algorithms are designed for a single camera node. As we emphasized in this thesis, due to the size of video data and limited communication bandwidths, it will be necessary to push the traditional approach of computer vision from single-camera processing to distributed multi-camera processing.

From an energy minimization point of view, in addition to the sampling and coding schemes presented in this thesis, the *multiple-input and multiple-output* (MIMO) technology [67] can reduce the communication consumption even further. In practice, MIMO has been successfully applied in indoor applications such as IEEE 802.11n WLAN. The basic principle of 802.11n is to use multiple antennas and a rich scattering environment to create spatial diversity. Unlike the indoor scenario, the outdoor monitoring cameras operate in open space where multi-path scattering is significantly lower. In this case, the uplink MIMO channel from the transmitter to the receiver (BS) is a well conditioned fading channel only if the antennas are far apart. Therefore, physically separated wireless cameras provides a potentially good MIMO setup.

For instance, for two cameras (one antenna per camera) located 10km away from the BS and using the 1GHz band for cameras-to-BS communication[1], the separation requirement between the two cameras is 500m [67, P.301]. In such a case, multi-user MIMO (MU-MIMO) can be employed to boost the wireless link speed. In theory, under the same energy constraint, the use of MU-MIMO doubles the maximum wireless link speed. In other words, to transmit the same amount of data, a two-camera system spends half the communication energy than a single-camera system. Therefore, taking into account the energy sharing among cameras, the overall consumption per camera is reduced by a factor of 2×2. Similarly, if the number of cameras is n, MIMO can potentially reduce the communication consumption per camera by a factor of n^2.

This reduction is a significant leap from our current work, as it can reduce the overall system consumption by a factor of n, whereas in our work the system consumption is independent of n. We expect that in the near future practical MIMO communication hardware will emerge in wireless sensor networks. The advance towards this direction will further transform the way how we use many wireless cameras.

[1] Assuming the reception antennas at the BS have a dimension of 6m.

Bibliography

[1] K. Pister, J. Kahn, and B. Boser, "Smart Dust: BAA97-43 Proposal Abstract," 1997.

[2] F. Ingelrest, G. Barrenetxea, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange, "Sensorscope: Application-specific sensor network for environmental monitoring," *ACM Transactions on Sensor Networks (TOSN)*, vol. 6, Mar. 2010.

[3] M. Rahimi, R. Baer, O. Iroezi, J. Garcia, J. Warrior, D. Estrin, and M. Srivastava, "Cyclops: in situ image sensing and interpretation in wireless sensor networks," in *Proc. SenSys '05*. ACM, 2005, pp. 192–204.

[4] P. Chen, P. Ahammad, C. Boyer, S. Huang, L. Lin, E. Lobaton, M. Meingast, S. Oh, S. Wang, P. Yan *et al.*, "Citric: A low-bandwidth wireless camera network platform," in *Proc. ICDSC '08*. IEEE, 2008, pp. 1–10.

[5] R. Ng, M. Levoy, M. Brédif, G. Duval, M. Horowitz, and P. Hanrahan, "Light field photography with a hand-held plenoptic camera," *Computer Science Technical Report CSTR*, vol. 2, 2005.

[6] J. Hicks, J. Paek, S. Coe, R. Govindan, and D. Estrin, "An easily deployable wireless imaging system," in *ImageSense'08: Proceedings of the Workshop on Applications, Systems, and Algorithms for Image Sensing*, 2008.

[7] H. Nguyen, B. Bhanu, A. Patel, and R. Diaz, "VideoWeb: Design of a wireless camera network for real-time monitoring of activities," in *Proc. ICDSC '09*. IEEE, 2009, pp. 1–8.

[8] S. Michel, A. Salehi, L. Luo, N. Dawes, K. Aberer, G. Barrenetxea, M. Bavay, A. Kansal, K. Kumar, S. Nath *et al.*, "Environmental monitoring 2.0," in *Proc. ICDE '09*. IEEE, 2009, pp. 1507–1510.

[9] Z. Chen, P. Prandoni, G. Barrenetxea, and M. Vetterli, "Sensorcam: An energy-efficient smart wireless camera for environmental monitoring," in *Proc. IPSN '12*. ACM, 2012.

[10] M. Gorlatova, A. Wallwater, and G. Zussman, "Networking low-power energy harvesting devices: Measurements and algorithms," in *Proc. INFOCOM '11*. IEEE, 2011, pp. 1602–1610.

[11] M. A. Green, K. Emery, Y. Hishikawa, and W. Warta, "Solar cell efficiency tables (version 37)," *Progress in Photovoltaics: Research and Applications*, vol. 19, no. 1, pp. 84–92, 2011.

[12] D. Lee, H. Kim, M. Rahimi, D. Estrin, and J. Villasenor, "Energy-Efficient Image Compression for Resource-Constrained Platforms," *IEEE Trans. Image Process.*, vol. 18, no. 9, 2009.

[13] T. Ajdler, L. Sbaiz, and M. Vetterli, "The plenacoustic function and its sampling," *IEEE Trans. Signal Process.*, vol. 54, no. 10, pp. 3790–3804, 2006.

[14] T. Cui, L. Chen, and T. Ho, "Distributed optimization in wireless networks using broadcast advantage," in *Proc. CDC '07*. IEEE, 2007, pp. 5839–5844.

[15] S. Strogatz, *Sync: The emerging science of spontaneous order*. Hyperion, 2003.

[16] J. Van Greunen and J. Rabaey, "Lightweight time synchronization for sensor networks," in *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*. ACM, 2003, pp. 11–19.

[17] Q. Li and D. Rus, "Global clock synchronization in sensor networks," *IEEE Trans. Comput.*, vol. 55, no. 2, pp. 214–226, 2006.

[18] J. Degesys, I. Rose, A. Patel, and R. Nagpal, "Desync: self-organizing desynchronization and tdma on wireless sensor networks," in *Proc. IPSN '07*. ACM, 2007, pp. 11–20.

[19] J. Degesys and R. Nagpal, "Towards desynchronization of multi-hop topologies," in *Proc. SASO '08*. IEEE, 2008, pp. 129–138.

[20] D. Johnson and M. Garey, *Computers and Intractability: A Guide to the Theory of NP-completeness*. San Francisco: Freeman&Co, 1979.

[21] Y. Gurevich and S. Shelah, "Expected computation time for hamiltonian path problem," *SIAM J. Comput.*, vol. 16, no. 3, pp. 486–502, Jun. 1987.

[22] E. Levy, G. Louchard, and J. Petit, "A distributed algorithm to find hamiltonian cycles in g(np) random graphs," in *Proceedings of the First international conference on Combinatorial and Algorithmic Aspects of Networking*. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 63–74.

[23] O. Ore, "Note on hamilton circuits," *The American Mathematical Monthly*, vol. 67, no. 1, pp. 55–55, 1960.

[24] T. Cover and J. Thomas, *Elements of information theory*. Wiley-Interscience, 1991.

[25] T. Berger, "Multiterminal source coding," in *The Information Theory Approach to Communications (CISM Courses and Lectures, no. 229)*, G. LONGO, Ed. Vienna/New York: Springer-Verlag, 1978, pp. 171–231.

[26] D. Slepian and J. Wolf, "Noiseless coding of correlated information sources," *IEEE Trans. Inf. Theory*, vol. 19, no. 4, pp. 471–480, 1973.

[27] A. Wyner and J. Ziv, "The rate-distortion function for source coding with side information at the decoder," *IEEE Trans. Inf. Theory*, vol. 22, no. 1, pp. 1–10, 1976.

[28] A. B. Wagner, S. Tavildar, and P. Viswanath, "Rate region of the quadratic gaussian two-encoder source-coding problem," *IEEE Trans. Inf. Theory*, vol. 54, no. 5, pp. 1938–1961, 2008.

[29] B. Girod, A. M. Aaron, S. Rane, and D. Rebollo-Monedero, "Distributed video coding," *Proc. IEEE*, vol. 93, no. 1, pp. 71–83, Jan. 2005.

[30] R. Puri, A. Majumdar, and K. Ramchandran, "Prism: A video coding paradigm with motion estimation at the decoder," *IEEE Trans. Image Process.*, vol. 16, no. 10, pp. 2436–2448, 2007.

[31] C. Yeo and K. Ramchandran, "Robust distributed multiview video compression for wireless camera networks," *IEEE Trans. Image Process.*, vol. 19, no. 4, pp. 995–1008, 2010.

[32] A. Kaspi and T. Berger, "Rate-distortion for correlated sources with partially separated encoders," *IEEE Trans. Inf. Theory*, vol. 28, no. 6, pp. 828–840, 1982.

[33] J. Korner, "Successive encoding of correlated sources," *IEEE Trans. Inf. Theory*, vol. 29, no. 3, pp. 390–395, 1983.

[34] Y. Oohama, "Universal coding for correlated sources with linked encoders," *IEEE Trans. Inf. Theory*, vol. 42, no. 3, pp. 837–847, May 1996.

[35] W. Equitz and T. Cover, "Successive refinement of information," *IEEE Trans. Inf. Theory*, vol. 37, no. 2, pp. 269–275, 1991.

[36] H. Viswanathan and T. Berger, "Sequential coding of correlated sources," *IEEE Trans. Inf. Theory*, vol. 46, no. 1, pp. 236–246, 2000.

[37] H. Behroozi and M. Reza Soleymani, "Successively structured gaussian two-terminal source coding," *Wireless Personal Communications*, vol. 48, no. 4, pp. 485 – 510, 2009.

[38] C. Christopoulos, A. Skodras, and T. Ebrahimi, "The JPEG2000 still image coding system: an overview," *IEEE Trans. Consum. Electron.*, vol. 46, no. 4, pp. 1103–1127, 2000.

[39] C. E. Shannon, "A mathematical theory of communication," *Bell system technical journal*, vol. 27, 1948.

[40] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge Univ Pr, 2003.

[41] R. Zamir, "The rate loss in the wyner-ziv problem," *IEEE Trans. Inf. Theory*, vol. 42, no. 6, pp. 2073–2084, 1996.

[42] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge Univ Pr, 2004.

[43] M. Gastpar, "To code or not to code," Ph.D. dissertation, EPFL, 2002.

[44] M. Vetterli, J. Kovacevic, and V. K. Goyal, *Signal Processing: Foundations*. Cambridge Univ Pr, 2012.

[45] B. Song, O. Bursalioglu, A. Roy-Chowdhury, and E. Tuncel, "Towards a multi-terminal video compression algorithm using epipolar geometry," in *Proc. ICASSP '06*. IEEE, 2006.

[46] A. Habibi, "Hybrid coding of pictorial data," *IEEE Trans. Commun.*, vol. 22, no. 5, pp. 614–624, 1974.

[47] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, "Video coding with H. 264/AVC: tools, performance, and complexity," *IEEE Circuits Syst. Mag.*, vol. 4, no. 1, pp. 7–28, 2005.

[48] R. Puri, A. Majumdar, P. Ishwar, and K. Ramchandran, "Distributed video coding in wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 23, no. 4, pp. 94–106, 2006.

[49] A. Wang and C. Sodini, "On the energy efficiency of wireless transceivers," in *Proc. ICC '06*. IEEE, 2006, pp. 3783–3788.

[50] *Colibri XScale PXA270 Datasheet* , Toradex AG.

[51] x264. Library and application for encoding video streams into the H.264/MPEG-4 AVC format. `http://www.videolan.org/developers/x264.html`.

[52] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison *et al.*, "Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera," in *Proc. UIST '11*. ACM, 2011, pp. 559–568.

[53] S. Liu, Y. Wang, L. yuan, J. Bu, P. Tan, and J. Sun, "Video stabilization with a depth camera," in *Proc. CVPR '12*. IEEE, 2012.

[54] D. Lowe, "Object recognition from local scale-invariant features," in *Proc. ICCV '99*, vol. 2. IEEE, 1999, pp. 1150–1157.

[55] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[56] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[57] OpenCV. A library of programming functions for real time computer vision. `http://opencv.willowgarage.com/wiki/`.

[58] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[59] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679 –698, 1986.

[60] G. Borgefors, "Distance transformations in arbitrary dimensions," *Computer vision, graphics, and image processing*, vol. 27, no. 3, pp. 321–345, 1984.

[61] M. Piccardi, "Background subtraction techniques: a review," in *IEEE International Conference on Systems, Man and Cybernetics*, vol. 4. IEEE, 2004, pp. 3099–3104.

[62] A. Netravali and B. Haskell, *Digital pictures: representation, compression, and standards*. Plenum Publishing Corporation, 1995.

[63] P. Burt and E. Adelson, "A multiresolution spline with application to image mosaics," *ACM Transactions on Graphics (TOG)*, vol. 2, no. 4, pp. 217–236, 1983.

[64] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," *ACM Transactions on Graphics (TOG)*, vol. 22, no. 3, pp. 313–318, 2003.

[65] Z. Chen, F. Yang, A. Lindner, G. Barrenetxea, and M. Vetterli, "How is the weather: automatic inference from images," in *Proc. ICIP '12*. IEEE, 2012.

[66] C. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[67] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge Univ Pr, 2005.

[68] Z. Chen, G. Barrenetxea, and M. Vetterli, "Distributed successive approximation coding using broadcast advantage: the two-encoder case," in *Proceedings of the 48th Annual Allerton Conference on Communication, Control and Computing*, 2010, pp. 1110–1116.

[69] ——, "Distributed successive refinement of multiview images using broadcast advantage," *IEEE Trans. Image Process.*, vol. 21, no. 11, pp. 4581–4592, 2012.

[70] ——, "Share risk and energy: Sampling and communication strategies for multi-camera wireless monitoring networks," in *Proc. INFOCOM '12*. IEEE, 2012.

[71] ——, "Event-driven video coding for outdoor wireless monitoring cameras," in *Proc. ICIP '12*. IEEE, 2012.

Curiculum Vitæ

Chen, Zichong （陈子冲）

Rue du Centre 44A
1025 St-Sulpice
Switzerland

http://lcav.epfl.ch/people/zichong.chen
e-mail: chenzc04@gmail.com
Date of Birth: February 13, 1986
Born in Ningbo, Chinese citizen

Education

2008–2012 **Ph.D.** in Computer, Communication and Information Sciences, Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland

2004–2008 **B.Sc.** in Electronic Engineering, Tsinghua University, Beijing, China

Professional experience

09/2009–12/2012 **Research and Teaching Assistant**, Audiovisual Communications Laboratory (LCAV), Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland. Advisor: Prof. Martin Vetterli and Dr. Guillermo Barrenetxea.

08/2011–11/2011 **Research Intern**, Media Technology Lab, Huawei North America Headquarters, Santa Clara, CA, United States. Advisor: Dr. Fengjun Lv and Dr. Ton Kalker.

07/2007–07/2008 **Research Assistant**, Microwave and Digital Communications Laboratory, Department of Electronic Engineering, Tsinghua University, Beijing, China. Advisor: Prof. Yuantao Gu. Thesis topic: *Design and Implementation of Satellite Beacon Receiver* (**Best thesis award**).

Publications

Journal papers

1. Z. Chen, G. Barrenetxea, and M. Vetterli. Distributed Successive Refinement of Multiview Images Using Broadcast Advantage. *IEEE Transactions on Image Processing*, vol. 21, no. 11, 2012.

Conference papers

1. Z. Chen, G. Barrenetxea, and M. Vetterli. Event-Driven Video Coding for Outdoor Wireless Monitoring Cameras. *Proceedings of International Conference on Image Processing*. Oct. 2012.
2. Z. Chen, F. Yang, A. Lindner, G. Barrenetxea, and M. Vetterli. How is the Weather: Automatic Inference from Images. *Proceedings of International Conference on Image Processing*. Oct. 2012.
3. Z. Chen, P. Prandoni, G. Barrenetxea, and M. Vetterli. Sensorcam: An Energy-Efficient Smart Wireless Camera for Environmental Monitoring. *Proceedings of 11th ACM/IEEE Conference on Information Processing in Sensor Networks*. Apr. 2012. Poster abstract.
4. Z. Chen, G. Barrenetxea, and M. Vetterli. Share Risk and Energy: Sampling and Communication Strategies for Multi-Camera Wireless Monitoring Networks. *Proceedings of 31st Annual IEEE International Conference on Computer Communications*. Mar. 2012.
5. Z. Chen, G. Barrenetxea, and M. Vetterli. Distributed Successive Approximation Coding using Broadcast Advantage: the Two-Encoder Case. *Proceedings of 48th Annual Allerton Conference on Communication, Control and Computing*. Sep. 2010.

Awards and honors

2008	Fellowship, School of Computer and Communication Sciences, EPFL
2008	Honor of Undergraduate Student, Tsinghua University
2007	National Scholarship, China
2004	Freshman Scholarship (recommended for admission without exams), Tsinghua University
2003	First Prize of the 20th Chinese Physics Olympiad

Academic service

Reviewer	IEEE/ACM Transactions on Networking, ACM Transaction on Sensor Networks, IEEE Transactions on Image Processing, European Signal Processing Conference (EUSIPCO 2012)