Foveation Techniques and Scheduling Issues in Thinwire Visualization

by

Ee-Chien Chang

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

May, 1998

Approved: ______________________________

Professor Chee Yap

Research Advisor

To my family

Acknowledgments

Firstly I wish to express my gratitude to my advisor, Professor Chee Yap. Without his encouragement, support, guidance and the occasional prodding, none of this would have been possible.

In addition, I would like to acknowledge with gratitude Professor Stéphane Mallat for his invaluable advice and help in all matters involving wavelets.

I would also like to thank Professor Eero Simoncelli and other members of my thesis committee for their suggestions and patience.

Special thanks to all my friends who make my stay in New York City a very memorable one.

Lastly, I am grateful to the Department of Computational Science, National University of Singapore for providing financial assistance for my studies here.

Contents

Chapter 1

Introduction

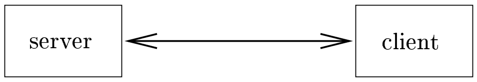


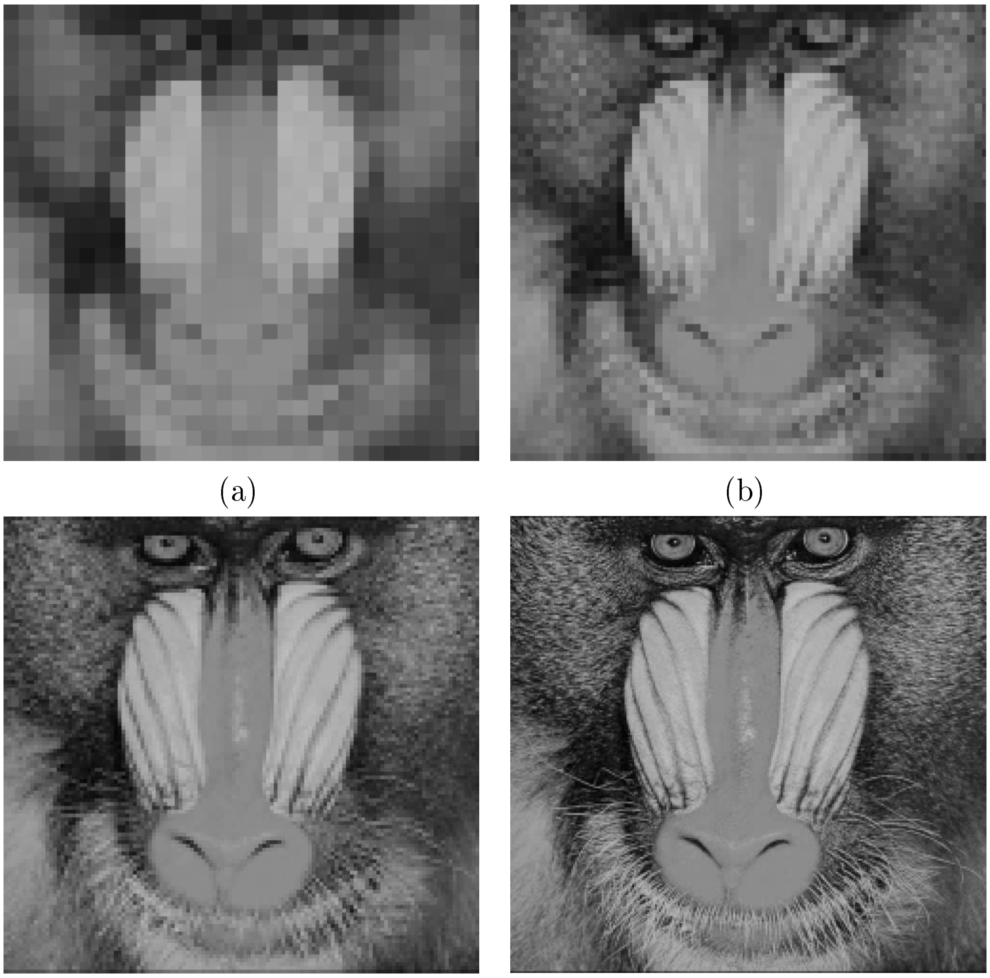
Figure 1.1: Thinwire model.

We are interested in the visualization of large images across a network. Upon request, the *server* sends an image across the network to the *client*, who in turn, presents this image to the *viewer*. As the network is of relatively low bandwidth, we call it a *thin wire* (Figure 1.1). A usual practice under this setting is progressive transmission where a low resolution image is first sent across and it is then progressively and uniformly refined (Figure 1.2). A key observation is that, at any moment, the viewer is only interested in a region around a point (the *gaze point*) in the image. To exploit this, we give a scheme where the viewer interactively indicates this region and the selected region has higher priority in the refinement process (Figure 1.3). As a result, the displayed image in the client-side does not has uniform resolution. We call such image a *space-variant* image and call the scheme *interactive progressive transmission*.

A fundamental difference between this scheme and the usual progressive transmission scheme is that we bring what the viewer wants to see into consideration, in other words, we place more emphasis on the visualization process. This shift in emphasis opens new perspectives of the problem. In this report, we focus on this difference. There are three issues that arise, and the next three chapters are organized accordingly.

First, we need to understand what we mean by an image having varied resolution, and need to know how to distribute the resolution on an image and how to progressively refine such a space-variant image. In chapter two, we look at a psychophysical finding of human vision and study space-variant image in two approaches. In the first approach, we give a formulation that expresses the process of obtaining a space-variant image from a uniform image as a functional operator and study its operator matrix. In the second approach, we give a scheme where, loosely speaking, the space-variant image is the one that minimizes a weighted distance from the original image, given a fixed number of bits.

The second issue arises from the interactivity between the viewer and the system. If we treat what the viewer wants to view as a request for data, then, throughout the visualization process, more requests will be generated



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

Figure 1.2: Progressive Transmission: A low resolution image (a) is first sent across. The subsequent image (b), (c) and (d) are refined progressively.

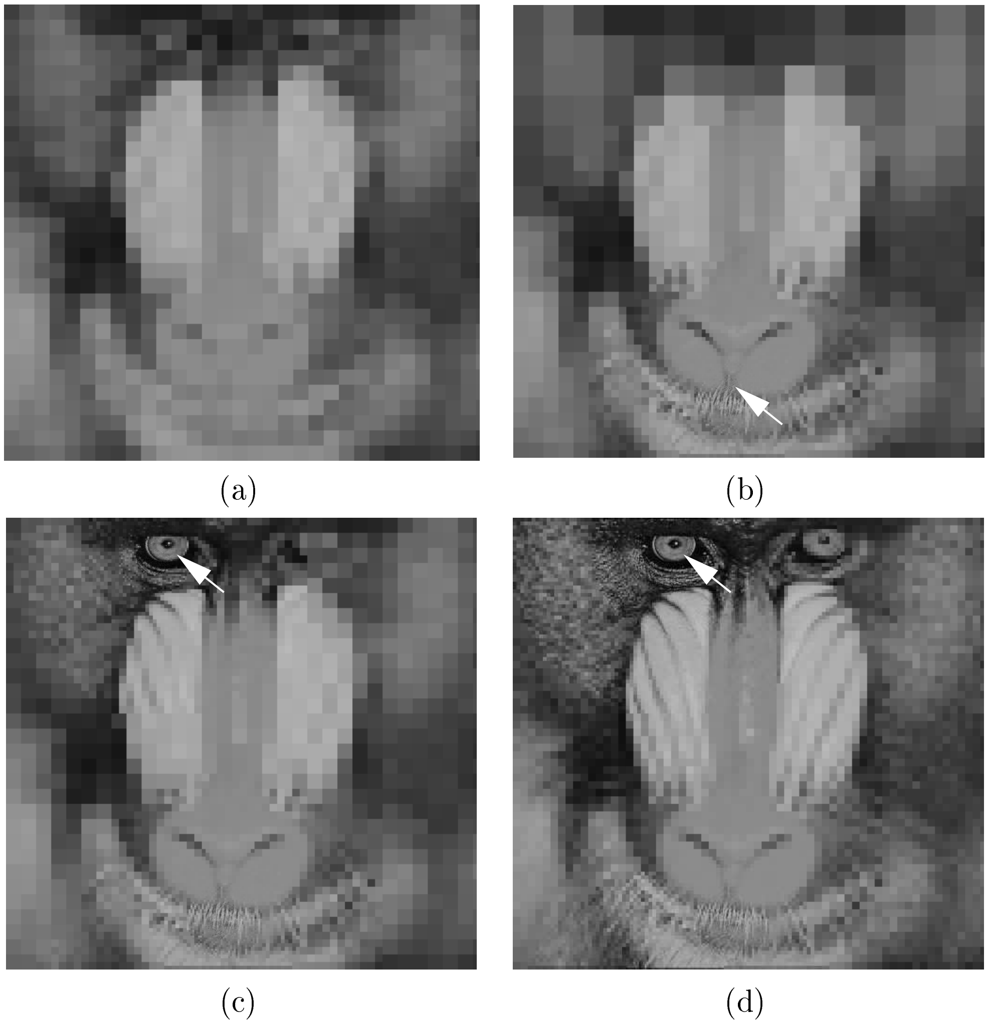


Figure 1.3: Interactive progressive transmission: the white arrow points to the region selected by the viewer. Note that in (c), the details around the "mouth" retain. The position of the arrow remains unchanged from (c) to (d), and thus more refinements are performed on the "left eye".

than the server could serve. Thus, it is necessary to select the importance requests which are to be sent to the server. This motivates the formulation of an on-line scheduling problem in chapter three. An interesting property of this problem which we exploit is that: a request could be scaled down and served at a lower level of service. We give a few scheduling strategies and prove their competitive ratio.

The actual implementation of the system is not straightforward. Since the images are large, a more careful memory management strategy is required. Furthermore, as visualization is a real-time process, this induces additional constraints in the system design. In chapter four, we study these constraints and describe an implementation of interactive progressive transmission.

Chapter 2

Foveation

We study two aspects of foveation. First, we define foveation as a "space-variant" low-pass filtering process, through which we describe how resolution is "distributed" on the image. Second, we formulate the problem of approximating a foveated/space-variant image using a fixed number of bits, while trying to minimize some weighted error. Based on these two studies, we propose two progressive transmission schemes.

2.1 Introduction

Figure 2.1(a) is a uniform resolution image whereas Figure 2.1(b) is a *foveated* image. We call the point of highest resolution the *fovea* and the process of going from a uniform image to a foveated image *foveation*. In general, we call an image with non-uniform resolution a *space-variant* image. These terms are borrowed from biological vision systems. Note that in Figure

(a) Uniform resolution image.

(b) Space variant resolution image

Figure 2.1: Foveation.

2.1(b), resolution decreases gradually as the distance from the fovea increases. One of our aims here is to formulate this space-variant structure of an image.

2.1.1 Physiological Aspect of Foveation

It is well-known that our visual system has a space-variant nature where the resolution is high in the center of the visual field but gradually falling off towards the peripheral. Studies of this space-variant structure in the visual cortex could be traced back to [26, 14, 30], who suggest a well-defined map-like representation of visual field in the cortex. In the early 1940's, Talbot and Marshall [41] demonstrate and confirm this hypothesis. Subsequent studies by Schwartz [32] show that the complex logmap is a good model for the mapping of the visual field in the visual cortex. Schwartz gives a recent survey on this topic in [33]. The form of logmap proposed by Schwartz is characterized by two real constants k and a and can be conveniently expressed as a complex function $LOG : \mathbb{C} \to \mathbb{C}$ of the form:

$$LOG(z) := \begin{cases} k \ln(z + a) & \text{if } Re(z) > 0, \\ k \ln(z - a) & \text{otherwise.} \end{cases}$$

The logmap maps the point (x, y) in the *retinal plane* to a point (ρ, θ) in the *visual cortex plane* where

$$\begin{aligned} \rho &:= k \ln(\sqrt{(|x| + a)^2 + y^2}), \quad \text{and} \\ \theta &:= \begin{cases} \tan^{-1}\left(\frac{y}{x+a}\right) & \text{if } x > 0, \\ \tan^{-1}\left(\frac{y}{x-a}\right) + \pi & \text{otherwise.} \end{cases} \end{aligned} \tag{2.1}$$

This mapping is illustrated in Figure 2.2.

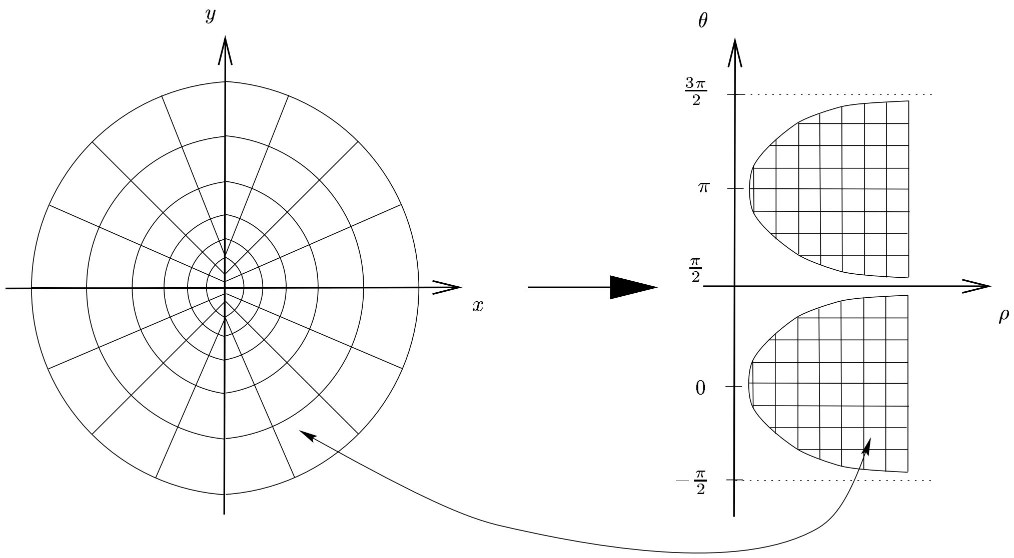


Figure 2.2: Transformation from the visual field to the visual cortex.

By sampling the visual cortex plane uniformly, the value at each sample point (ρ_i, θ_j) could be viewed as the result of applying an "averaging" operation (or low-pass filtering) on the neighborhood of $LOG^{-1}(\rho_i, \theta_j)$ in the retinal plane. Call the sample points $\{LOG^{-1}(\rho_i, \theta_j)\}$ the *logmap* pixels. Note that the logmap determines the arrangement of logmap pixels in the retinal plane. An advantage of using the logmap pixels to represent the visual field (rather than the uniform grid) is that it permits a reduction of the total number of pixels while retaining high density of pixels in the "interesting" region, and this has motivated the design of many computer vision and visualization systems.

2.1.2 Applications of Foveation in Computer System

From an engineering point of view, it is not necessary to adhere to the exact form of space-variant mapping obtained from physiological findings. For example, Panerai et al. [29] use a logmap of the form,

$$\begin{aligned} \rho &:= k\ln(\sqrt{x^2+y^2}+a), \quad \text{and} \\ \theta &:= \begin{cases} \tan^{-1}\left(\frac{y}{x}\right) & \text{if } x > 0, \\ \tan^{-1}\left(\frac{y}{x}\right) + \pi & \text{otherwise.} \end{cases} \end{aligned} \tag{2.2}$$

They also study various arrangements of logmap pixels (in the retinal plane), namely *log-Cartesian* and *horizontal log-Cartesian* as illustrated in Figure 2.3. Another interesting arrangement is the one obtained by blending two arrangements: the usual uniform arrangement within the fovea and the space-variant arrangement on the peripheral. Examples of this configuration are described in [28]. A disadvantage of this configuration is the difficulty in merging the two arrangements smoothly.

The values at the logmap pixels can be obtained from a scene using a foveated vision chip (or seeing silicon) where the photoreceptors are organized in a way similar to the biological retina. Each photoreceptor carries out a sampling process based on some sampling function. A typical sampling function is the Mexican hat, whose width depends on the location of the photoreceptor. The width of a sampling function increases linearly as the distance of the photoreceptor from the center of the chip increases. Therefore, the process is a non-uniform sampling process. A survey on some of the past and current technology of vision chip could be found in [28], which is also available in the web [27].

Another method for obtaining the value at the logmap pixels is from a two dimensional uniform resolution raster image, for example Figure 2.1 (a). One approach to extract the values is by directly "emulating" the non-uniform sampling process, that is, the retinal plane is first partitioned into blocks where each block corresponds to a logmap pixel, and then the value of each logmap pixel is obtained by averaging all the original (uniform) pixels in the same block. The real-time system designed by Kortum [16] and the real-time system CORTEX-I developed by Wallace et al. [46] use this approach. The partition is usually precomputed and stored in a lookup table, and thus, the

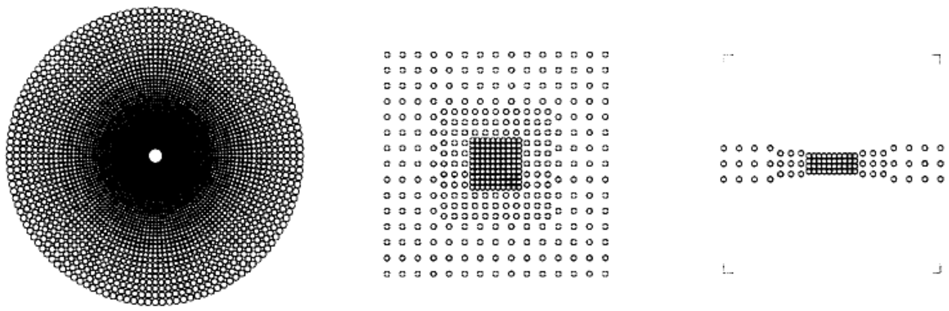


Figure 2.3: The logmap arrangement (given by (2.2)), log-Cartesian and horizontal log-Cartesian arrangements. Image reproduced from [29].

"emulating" process is fast but lack flexibility in the sense that the arrangement of the logmap pixels is fixed. A more flexible approach first computes a hierarchical representation of the image, and the foveated image is then obtained by cut-and-paste from different scales of the image. The method proposed by Burt [6, 1, 5] which uses Gaussian pyramid as the hierarchical representation of the image belongs to this approach.

Machine Vision A uniform image contains too much information and this is always a bottleneck in real-time vision systems. Two straightforward ways to reduce information are by lowering the resolution uniformly or by reducing the size of the visual field. Foveation provides a simple and fast blending of both means of reduction. Yeshurun and Schwartz [48] generalized the use of foveation in machine vision to *space-variant vision.* The performance of a space-variant vision system could be further enhanced by dynamically moving the fovea so as to gather relevant information. This methodology is also known as *smart-sensing* introduced by Burt [5]. Again, this dynamic nature is motivated by the human vision. An implementation of such a system is CORTEX-I[46, 4].

Another interesting property of foveation lies in the geometry of the logmap. Note that rotation and scaling in the retinal plane amounts to translation along the θ and ρ axis in the visual cortex plane respectively. Apparently, this is a desirable property in some vision tasks, for example, in corner detection. In addition, based on experimental results, Panerai et al. [29] report that computation in the visual cortex plane is more robust in finding correlation of two images I_{left} and I_{right} where both images are obtained from a binocular camera system.

A lot of basic image processing operations depend on the neighborhood relationship of pixels. In the usual uniform image, this neighborhood is usually taken as the 4-neighbor or 8-neighbor depending on whether the four corners are to be included. It is not clear how the neighborhood relationship should be defined for the logmap pixels. Wallace et al. [46] study various configurations of connectivity graph for the logmap pixels, where the connectivity graph represents the neighborhood relationship of the logmap pixels: two pixels are adjacent if and only if there is an edge between them.

Visualization The existence of a space-variant nature in our visual system suggests that a foveated image and the full resolution image is visually indistinguishable if the viewer's gaze point coincides with the center of the foveated image. This observation could be confirmed by psychological experiment and has been exploited by some visualization systems.

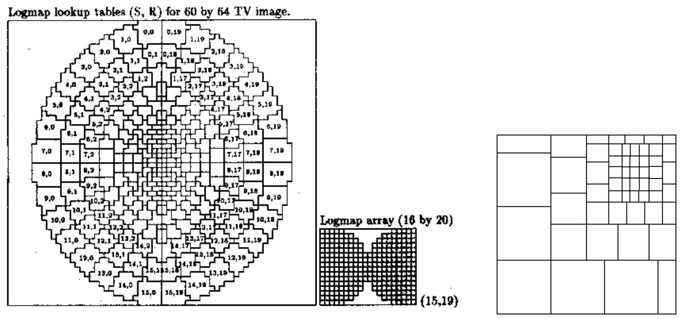
Since the landmark NSF Report in 1987 [25], visualization is an active research area. Visualization is a process whereby the viewer gains insight into a large data set. An interesting issue in visualization concerns how to visualize a large geometric data set. A possible approach is by allowing the viewer walks through these objects interactively. Of course, this method is computationally intensive due to the size of the data and the real-time requirement. This is where foveation could play an active role. Examples are flight simulation systems designed by [42, 11].

Volume rendering, which could be viewed as a projection from a three dimensional data to a two dimensional image, is inherently computationally intensive due to the large size of volumetric data set. In three dimensions, foveation results in even more significant data reduction, which in turn, leads to a reduction of rendering time. Since volume data is a generalization of image data, techniques on image foveation could be extended to volume naturally, except that it poses more of a computational challenge. Levoy and Whitaker build a system that achieve real-time rending of volume data [17] using foveation. They perform volume rendering at different resolution and the foveated image is obtained by pasting images of different resolutions using interpolation.

Recently, there is growing interest in visualizing large data sets across the Internet, and in these applications, the low transmission rate of the network is a bottleneck. Foveation, in this case, helps to reduce the network workload. As opposes to the previous applications, the data here could be the simple array data, for example image or video.

Using foveation or region of interests (ROI) to reduce network workload is an active area in video conferencing [31, 13, 2, 3]. Attention is also focused on the means to select ROI, for example in [31, 10]. Note that video and still images are very different in nature and have different requirements. Video transmission stresses even more on real-time performance, but there is less requirement of progressive refinement since successive frames or images displayed are not the same. For still images, it is well known that by allowing the user specifies the area of interest could greatly improve compression rate. For example, the commercial Summus' Wavelet Image (WI) has the feature of multiple "region of interests focusing" [39]. Here, the role of the user is to select the region of interests before the image is compressed. The selected regions are then compressed with less distortion. However, the role of the viewer is passive in the sense that he is not allowed to choose the region of interests.

A point to note here is that the viewer's gaze point is not fixed, in fact,



(a) (b)

Figure 2.4: Foveation as a non-uniform sampling process. Figure (a) is reproduced from [46] and (b) is reproduced from [16]

it is actively moving on the image. Thus, to provide a foveated image that is indistinguishable to the viewer throughout the visualization process, it is necessary to track the viewer's gaze point and produce the corresponding foveated image. Example of systems using eye tracking are [17, 16].

We would like to clarify that in our thinwire application, we do not intend to track the viewer's gaze point. Instead, the viewer is to "cooperate" with the system by selecting his gaze point using the conventional pointing devices, for example mouse, joystick or even keyboard. We also do not aim to use the logmap to produce foveated images that are indistinguishable from their corresponding full resolution images. Instead, we use the logmap as a guide to the "distribution" of resolution on the images.

2.2 Space-Variant Resolution

2.2.1 Foveation Operator

Our goal here is to formally describe the process of foveation and understand the role of the logmap in this process. We formulate this process as an operator and analyze its operator matrix with respect to some wavelet basis. The analysis suggests some approximation methods. Based on these methods, we propose an interactive progressive transmission scheme.

Definition in one dimension. We begin with one dimensional functions. A *foveation* [8] of a function $f : \mathbb{R} \to \mathbb{R}$ is determined by a *weight function* $w : \mathbb{R} \to \mathbb{R}_{\geq 0}$ and a *scaling function* $g : \mathbb{R} \to \mathbb{R}$. The weight function is non-negative and $w(x) = 0$ only for finitely many x. The scaling function g is integrable and normalized so that $\int_{-\infty}^{\infty} |g(x)|\, dx = 1$.

Define the *foveation* operator T^{fov} as

$$\left(T^{\mathrm{fov}} f\right)(x) := \int_{-\infty}^{\infty} f(t) \frac{1}{w(x)} g\left(\frac{t-x}{w(x)}\right) dt. \tag{2.3}$$

By writing

$$g_x(t) := \frac{1}{w(x)} g\left(\frac{t-x}{w(x)}\right), \tag{2.4}$$

T^{fov} can be rewritten as

$$\left(T^{\mathrm{fov}} f\right)(x) = \langle f, g_x \rangle.$$

Call the result of the above operation a *space-variant image*.

Treating T^{fov} as an integral operator

$$(Tf)(x) := \int K(x,t) f(t) dt, \tag{2.5}$$

then in our case, the kernel $K(x,t)$ is the scaling function $g_x(t)$.

We are interested in a special class of weight functions. A *standard weight function* is one of the form

$$w_{\text{std}}(t) := \alpha|t - \gamma| + \beta, \tag{2.6}$$

where α, β and γ are fixed constants and $\alpha, \beta \geq 0$. We call α the *rate*, β the *foveal resolution* and γ the *gaze point*, and call the resulting space-variant image a *foveated image*. Figure 2.1 (b) is the result of applying T^{fov} on Figure 2.1(a) with a standard weight function.

We could obtain new weight function by combining several weight functions. Given two weight functions, their *blended weight function* w is defined as

$$w(x,y) = \min\{w_1(x,y), w_2(x,y)\}.$$

We call a weight function blended from finitely many weight functions a *multi-fovea* weight function and the corresponding space-variant image a *multi-foveated image*.

In our thin-wire application, we only consider standard weight function and multi-fovea function. Beside the physiological motivation, a multi-fovea weight function is easy to manipulate since it is composed by finitely many standard weight functions where each in turn can be easily represented by their rate α, foveal resolution β and gaze point γ. This simple description is desirable in the thinwire applications.

Tabernero et al. study a closely related *foveatization* operator [40]. We will address the differences between their operator and ours.

Definition in two dimensions. In two dimensions, the scaling function $g : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is integrable and normalized as follow:

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} |g(s,t)| \, ds \, dt = 1.$$

The two dimensional weight function $w : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^{+} \cup \{0\}$ is non-negative and vanishes only at finitely many (x, y). Let

$$g_{x,y}(s,t) := \frac{1}{w(x,y)^2} g\left(\frac{s-x}{w(x,y)}, \frac{t-y}{w(x,y)}\right).$$

Define the two dimensional foveation as

$$\left(T^{\text{fov}} I\right)(x,y) := \langle I, g_{x,y} \rangle. \tag{2.7}$$

One way to obtain a two dimensional weight function is by extending a one dimensional weight function $w(\cdot)$ in the following way:

$$w(x,y) := w(\|(x,y)\|_2),$$

where $\|.\|_2$ is the usual 2-norm.

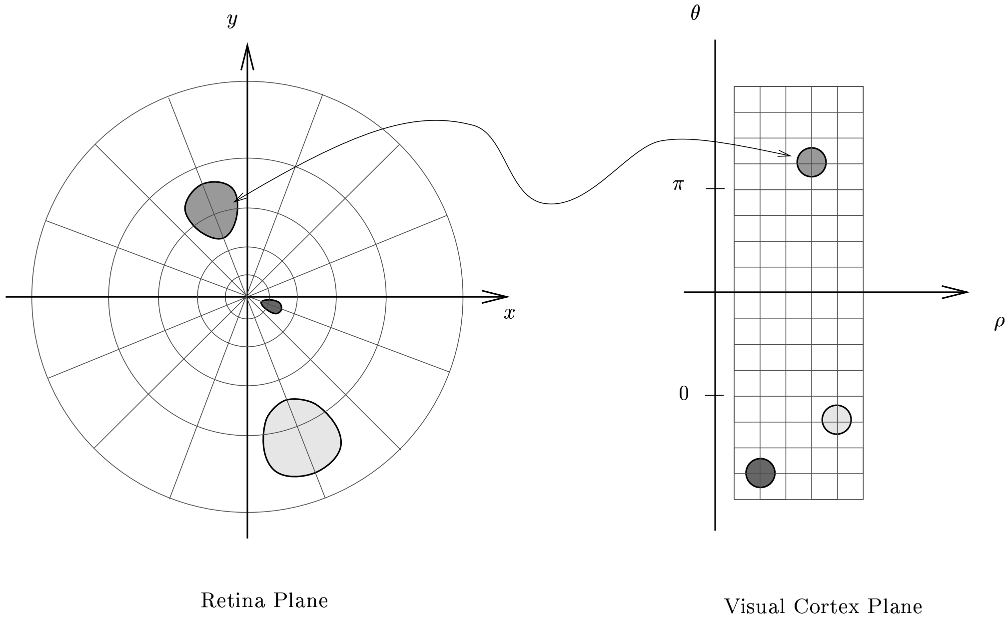


Figure 2.5: The corresponding neighborhood in the visual cortex plane and the retinal plane. Note that the neighborhood in retinal plane has different orientations and shapes.

Motivations by logmap. The definition of T^{fov} and standard weight function is motivated by the logmap. It intends to capture the situation where convolution is performed in the visual cortex plane. Let us break the integral in the definition of T^{fov} into two halves and assume $x > 0$,

$$T^{\text{fov}} f(x) = \int_0^{\infty} f(t) \frac{1}{x} g\left(\frac{t-x}{x}\right)\, dt + \int_{-\infty}^{0} f(t) \frac{1}{x} g\left(\frac{t-x}{x}\right)\, dt.$$

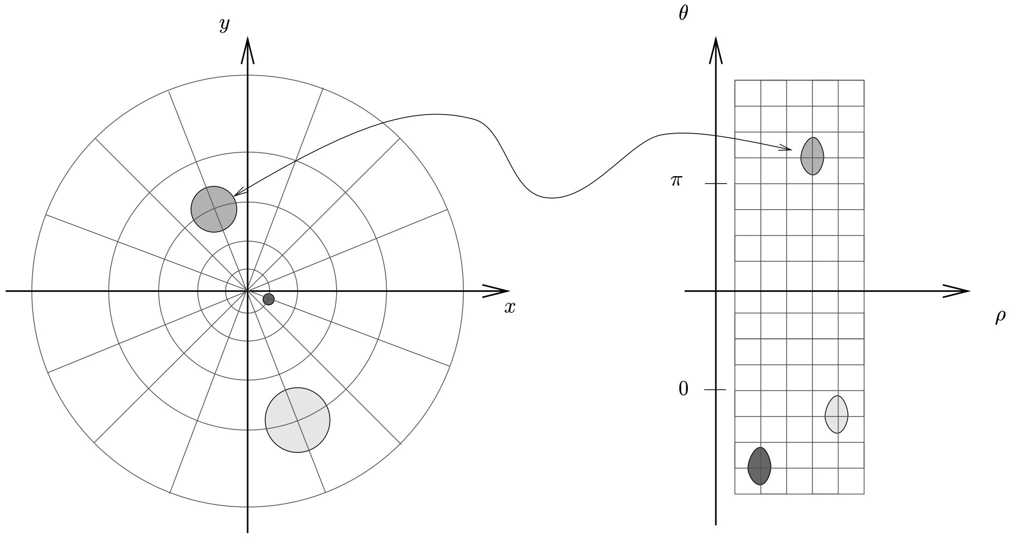


Figure 2.6: As opposed to Figure 2.5, the neighborhoods in the retinal plane have common orientation and shape.

We could rewrite the first half as a convolution.

$$\begin{aligned}
&\int_0^\infty f(t)\frac{1}{x}g\left(\frac{t-x}{x}\right)\,dt \\
&\quad = \int_0^\infty f(e^u)e^{-y}g\left(\frac{e^u-e^y}{e^y}\right)\,de^u \\
&\qquad \text{(where } e^y = x \text{ and } e^u = t\text{)} \\
&\quad = \int_{-\infty}^\infty f(e^u)e^{-(y-u)}g\left(e^{-(y-u)}-1\right)\,du \\
&\quad = \int_{-\infty}^\infty f^*(u)g^*(y-u)\,du,
\end{aligned}$$

where

$$\begin{aligned}
f^*(u) &:= f(e^u), \text{ and} \\
g^*(v) &:= e^{-v}g\left(e^{-v}-1\right).
\end{aligned}$$

Similarly, the second half can be rewritten as a convolution of f_* with g_* where

$$\begin{aligned} f_*(u) &:= f(-e^u), \text{ and} \\ g_*(v) &:= -e^{-v} g\left(-e^{-v} - 1\right). \end{aligned}$$

This simplicity suggests that perhaps we should reformulate the notion of foveation by defining it as a convolution in the visual cortex plane (as illustrated in Figure 2.5). However, in our application, we are interested in multi-foveated image. A disadvantage of considering convolution in the visual cortex plane is that the corresponding space-variant filters in the retinal plane have "orientation" and "shape" that depend on their locations (Figure 2.5), a fact which would cause difficulties in constructing multi-foveated image. Instead, we keep the orientation and shape of the filters but vary their width according to their locations (Figure 2.6). Note that the difference between these two formulations amounts to whether a modulo $|\cdot|$ is applied to $w(x)$ (the $w(x)$ within the parameter of g) in the definition (2.3).

The foveatization operator [40], on the contrary, is equivalent to convolution in the visual cortex plane. Given a prototype filter P, the foveatization of a function is defined as

$$\widetilde{T}^{\text{fov}} f(x) := \int \frac{1}{|x|} f(t) P\left(\frac{t}{x}\right) dt, \tag{2.8}$$

By letting $g(t) := p(t+1)$, we have

$$\widetilde{T}^{\text{fov}} f(x) := \int \frac{1}{|x|} f(t) g\left(\frac{t-x}{x}\right) dt.$$

Note the difference between this formulation and (2.3). If g is even, that is $g(x) = g(-x)$, then (2.8) can be viewed as a foveation operator with $w(x) = |x|$.

2.2.2 Operator Matrix of Foveation

One approach to analyze an operator $T : L^2(\mathbb{R}) \to L^2(\mathbb{R})$ is by studying its operator matrix A whose entries

$$a_{i,j} := \langle T(g_i), g_j \rangle,$$

where $\mathcal{B} = \{g_m\}_{m \in \mathbb{Z}}$ is an orthonormal base. We are interested in the case where $\mathcal{B}$ is an orthonormal compactly supported wavelet base $\{\psi_{j,n}\}_{j,n \in \mathbb{Z}}$.

The operator matrix is also useful in designing an approximation algorithm. If the function f is uniformly sampled as a N-vector with respect to the father wavelet $\phi_{j_0,0}$ at some scale j_0, and the kernel (refer to (2.5)) is also sampled with respect to $\phi_{j_0,0}$ as a $N \times N$ matrix K, then a direct method to compute $T(f)$ takes $\theta(N^2)$ arithmetic operations. However, in the cases where the operator matrix A is sparse, we may able to do better. For example, if A is diagonal and there is a linear time algorithm to decompose and reconstruct f with respect to $\mathcal{B}$, then, we have a trivial linear time algorithm.

Note that there is a linear time fast wavelet transform for the orthonormal compactly supported wavelet $\mathcal{B}$,

For a non-sparse operator matrix, we can compress it by suppressing entries that are below a threshold value. For many operators, by choosing an appropriate wavelet, the compressed matrix $\widetilde{A}$ is sparse with a very thin diagonal band, that is, $|i-j| > k \Rightarrow \widetilde{a}_{i,j} = 0$, where k is some small constant. This leads to a $O(kn)$ time approximation algorithm.

Now, consider the foveation operator. Let

$$\theta_{j,m,k,n} := \langle T^{\mathrm{fov}} \psi_{j,m}, \psi_{k,n} \rangle. \tag{2.9}$$

Let us compute, numerically, the foveation operator matrix. Figure 2.7 illustrates such a matrix computed using DAUB8 wavelet. The scaling function is a Gaussian function cutting off at $|x| > 5$, that is,

$$g(x) := \begin{cases} \frac{1}{\sqrt{2\pi}} e^{-x^2/2}, & \text{if } |x| < 5, \\ 0, & \text{otherwise.} \end{cases}$$

The weight function is the standard weight function $w(x) := (\alpha|x|)^{-1}$ with rate $\alpha := 1/30$.

A quick visual inspection suggests that the matrix is dominated by the diagonal entries $\theta_{j,m,j,m}$ where $j, m \in \mathbb{Z}$. On the other hand, Figure 2.8 suggests that the diagonal entries decay, relatively slower, away from the fovea. In other words, there are two directions of decay away from each $\theta_{j,0,j,0}$: a very fast decay off the diagonal and a slower decay away from the

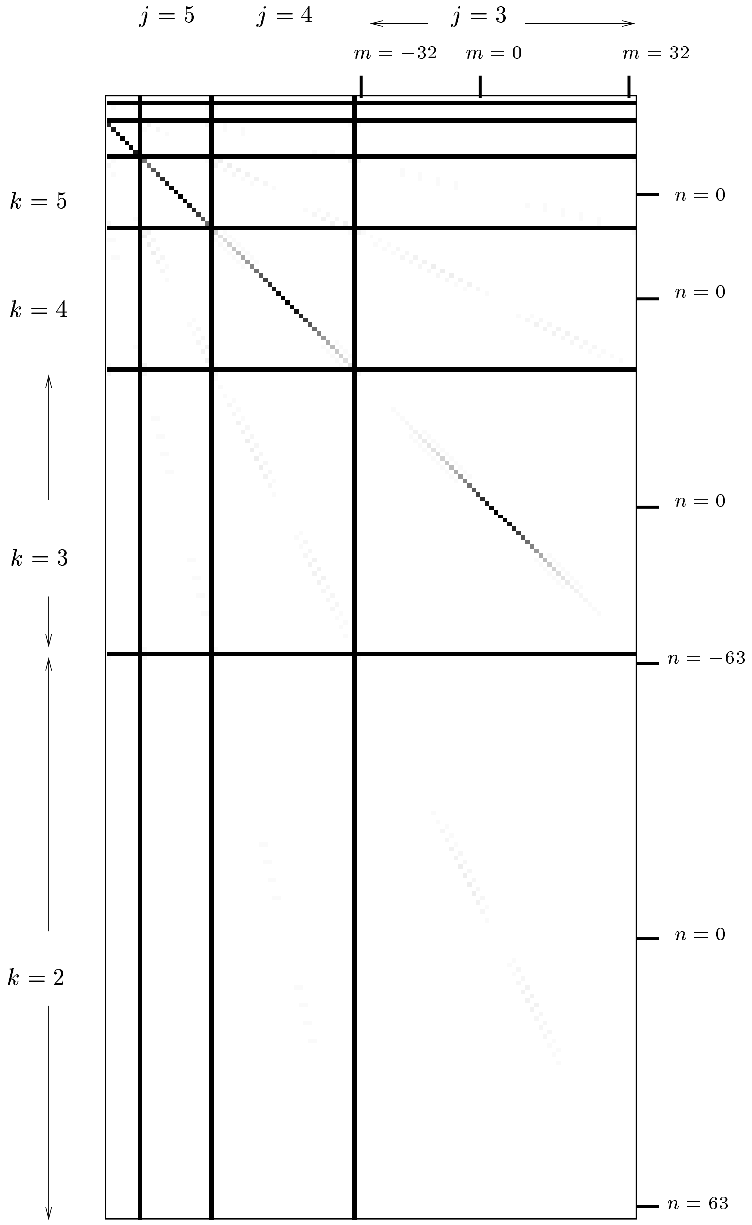


Figure 2.7: Computed numerical value of the operator matrix $\{\theta_{j,m,k,n}\}$. The entries are grouped into blocks where each block consists of entries with the same first and third index. For example, the block in the bottom-left corner consists of entries of the form $\theta_{6,m,2,n}$ where $-3 \le m \le 3$ and $-63 \le n \le 63$. The intensity of each pixel corresponds to the value of the corresponding entry; a darker pixel has a larger value.

fovea.

Another interesting observation from Figure 2.8 is the self similarity across the scales, that is, $\theta_{j,n,j,n} = \theta_{k,n,k,n}$ for any n, j, and k.

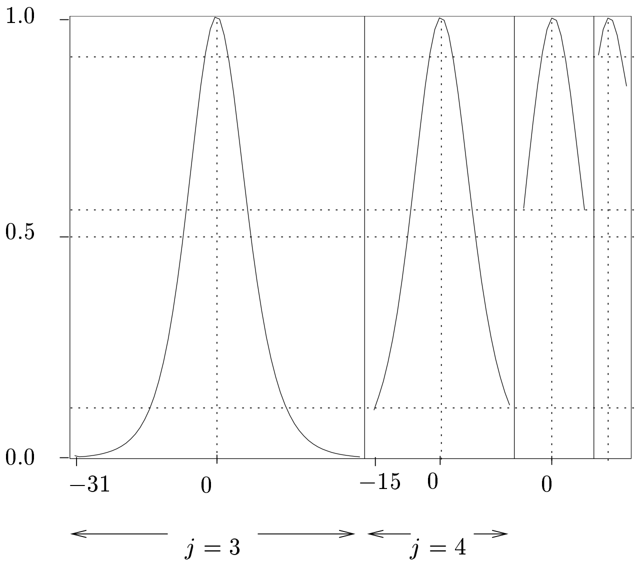


Figure 2.8: The cross section of Figure 2.7 along the diagonal entries $\theta_{j,m,j,m}$.

We justify these two observations in the next two sections. We first give an informal approximation of the diagonal entries $\theta_{0,n,0,n}$ which provides some insights on the roles of ψ and g. Next we give a few general bounds on the entries.

We restrict our discussion to compactly supported wavelet and scaling function. Furthermore, we only consider standard weight function with zero foveal resolution. Specifically,

Condition 2.1

1. *$\texttt{supp}(g) \subseteq [-A, A]$,*

2. *$w(x) = \alpha|x|$, where $\alpha A < 1$, and $\alpha > 0$,*

3. *$\{\psi_{j,m}\}_{j,m\in\mathbb{Z}}$ is an orthonormal compactly support wavelet, $\texttt{supp}(\psi) \subseteq [-A, A]$, and ψ is uniformly Lipschitz $q \geq 0$.*

2.2.3 Approximating the diagonal entries

We assume Condition 2.1. For simplicity, we assume $\alpha = 1$. By the compact support of ψ,

$$\theta_{0,n,0,n} = \int_{n-A}^{n+A} \int_{-\infty}^{\infty} \psi_{0,n}(x)\psi_{0,n}(t) g_x(t)\, dt\, dx,$$

where $g_x(t) := |x|^{-1} g\left(|x|^{-1}(t-x)\right)$. Note that we only have to consider g_x for $x \in [n-A, n+A]$. For large n, $h_n(\cdot - x)$ is a good approximation of g_x, where

$$h_n(t) := \frac{1}{|n|} g\left(\frac{t}{n}\right).$$

Using this approximation, we have a much simpler form:

$$c_n := \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \psi_{0,n}(x) h_n(t-x) \psi_{0,n}(t)\, dt\, dx.$$

By interchanging the integrals (which is possible by Fubini's theorem) and treating the first two terms as a convolution, the above can be rewritten as:

$$c_n = \int_{-\infty}^{\infty} \left(\psi_{0,n} \star h_n\right)(t) \psi_{0,n}(t)\, dt.$$

By applying the Parseval's formula and convolution theorem,

$$
\begin{aligned}
c_n &= \int_{-\infty}^{\infty} \widehat{\psi}_{0,n}(w)\overline{\widehat{h}_n(w)\widehat{\psi}_{0,n}(w)}\,dw \\
&= \int_{-\infty}^{\infty} \overline{\widehat{h}_n(w)}\left|\widehat{\psi}_{0,n}(w)\right|^2\,dw, \qquad (2.10)
\end{aligned}
$$

where we use the following convention for the Fourier transform:

$$\hat{f}(w) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} e^{-ixw} f(x)\,dx.$$

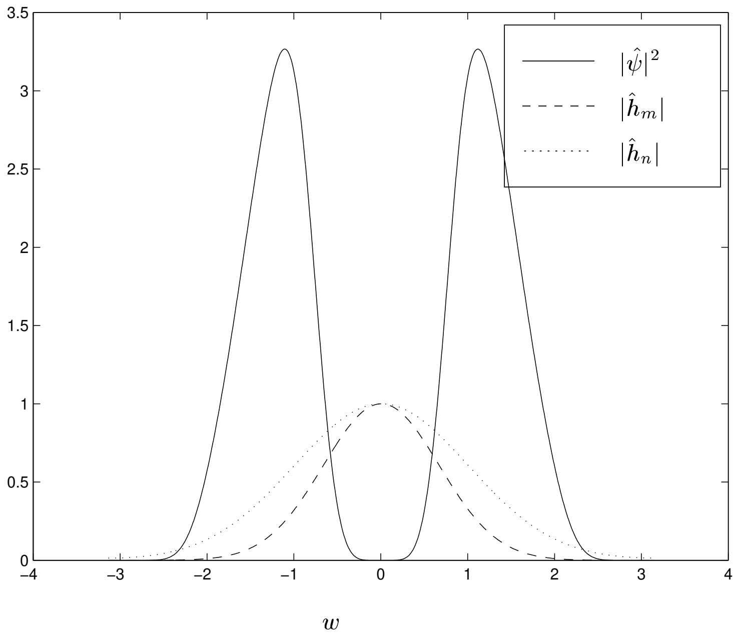


Figure 2.9: $\left|\hat{\psi}\right|$, $\left|\hat{h}_m\right|$ and $\left|\hat{h}_n\right|$ where m and n are some integers such that $m > n$.

Figure 2.9 illustrates (2.10). Since $\hat{h}_n(\cdot) = \hat{g}(|n|\,\cdot)$,

$$|c_n| = \int_{-\infty}^{\infty} \overline{\widehat{g}(|n|w)}\left|\widehat{\psi}(w)\right|^2\,dw.$$

Since ψ has compact support, $\hat{\psi}$ is at least p times continuously differentiable at 0. Furthermore, since ψ has p vanishing moments, we have $\hat{\psi}^{(k)}(0) = 0$ for $k < p$. Hence $|\hat{\psi}(w)| < Cw^p$ for some constant C. Thus

$$
\begin{aligned}
|c_n| &\leq C^2 \int_{-\infty}^{\infty} |\hat{g}(|n|w)| \, w^{2p} \, dw \\
&= C^2 |n|^{-2p-1} \int_{-\infty}^{\infty} |\hat{g}(w)| \, w^{2p} \, dw.
\end{aligned}
$$

Furthermore, if g is at least $2p$ times continuously differentiable, then

$$
\int_{-\infty}^{\infty} |\hat{g}(w)| \, |w|^{2p} < \infty,
$$

and thus leads to $|c_n| = O(|n|^{-2p-1})$.

What about $\langle T^{\text{fov}}\phi, \phi\rangle$? Similarly, e_n is a good approximation where

$$
e_n := \int_{-\infty}^{\infty} \overline{\hat{h}_n(w)} \left|\hat{\phi}(w)\right|^2 \, dw.
$$

Since $|\hat{\phi}(0)| = 1$, we expect a slower decay:

$$
\begin{aligned}
e_n \leq E \int_{-\infty}^{\infty} |\hat{h}_n(w)| \, dw \\
= \frac{E}{n} \int_{-\infty}^{\infty} |\hat{g}(w)| \, dw = E_1 n^{-1},
\end{aligned}
$$

where E and E_1 are some constants. The effect of this slower decay appears in two dimensional foveation, and we will revisit this in section 2.2.6.

2.2.4 Bounds on the entries

In this section, we study the operator matrix more rigorously. Again, we assume the wavelet ψ, scaling function g and weight function w satisfy Condition 2.1.

Self-similarity. This simple lemma tells us that the entries $\theta_{j,m,k,n}$ are the same across the scales. It implies that to compute the operator matrix we can first compute $\theta_{0,m,k,n}$ for all m, k and n and then extend them to all the other scales.

Lemma 2.1

$$\theta_{j,m,k,n} = \theta_{j-\ell,m,k-\ell,n} \quad \textit{for any} \quad \ell \in \mathbb{Z}.$$

Proof. By definition,

$$\begin{aligned}
\theta_{j,m,k,n} &= \int_{-\infty}^{\infty} \psi_{k,n}(x) \int_{-\infty}^{\infty} \psi_{j,m}(t)(w(x))^{-1} g\left(\frac{t-x}{w(x)}\right) dt\, dx \\
&= \int_{-\infty}^{\infty} 2^{-k/2}\psi(2^{-k}x - n) \int_{-\infty}^{\infty} 2^{-j/2}\psi(2^{-j}t - m)(w(x))^{-1} g\left(\frac{t-x}{w(x)}\right) dt\, dx.
\end{aligned}$$

Substituting $t' = 2^{-\ell}t$ and $x' = 2^{-\ell}x$,

$$\begin{aligned}
\theta_{j,m,k,n} &= \int_{-\infty}^{\infty} 2^{-(k-\ell)/2}\psi(2^{-(k-\ell)}x' - n) \\
&\qquad \int_{-\infty}^{\infty} 2^{-(j-\ell)/2}\psi(2^{-(j-\ell)}t' - m)(w(x'))^{-1} g\left(\frac{t'-x'}{w(x')}\right) dt'\, dx' \\
&= \theta_{j-\ell,m,k-\ell,n}.
\end{aligned}$$

Q.E.D.

Fast Decay. The next lemma (Lemma 2.2) tells us that for $\theta_{j,m,k,n}$ to be non-zero, m is of the order of $2^{k-j}n$. The following lemma (Lemma 2.3) tells us that $\theta_{j,m,k,n}$ is small if $|k-j|$ is large. These two lemmas together imply a decay off the diagonal. To illustrate, if $j=k$, then using the first lemma, n is closes to m. If $j \neq k$, then the second lemma suggests that it is small. Lemma 2.4 gives a sharper bound than Lemma 2.3 and it depends on the regularity of the wavelet and $|j-k|$. However, it is only applicable for large $|m|$ or large $|n|$.

On the other hand, Lemma 2.5 suggests the relatively slower decay away from the fovea. It gives a bound that decays as $|m|$ or $|n|$ increase and depends on the regularity of the scaling function g and the number of vanishing moments ψ has. In particular, the diagonal entries (where $(j,m)=(k,n)$) decay accordingly.

By just considering the support of g and ψ, we have the following:

Lemma 2.2 *There are constants C and D such that, if $\theta_{j,m,k,n} \neq 0$, then we have the followings.*

(a)

$$|n| < 2A \Rightarrow |m| < 2^{k-j}C + D.$$

(b)

$$|m| < 2A \Rightarrow |n| < 2^{j-k}C + D.$$

(c)

$$|n| \geq 2A \quad \textit{and} \quad |m| \geq 2A \Rightarrow C|m| > 2^{k-j}|n| > C^{-1}|m|.$$

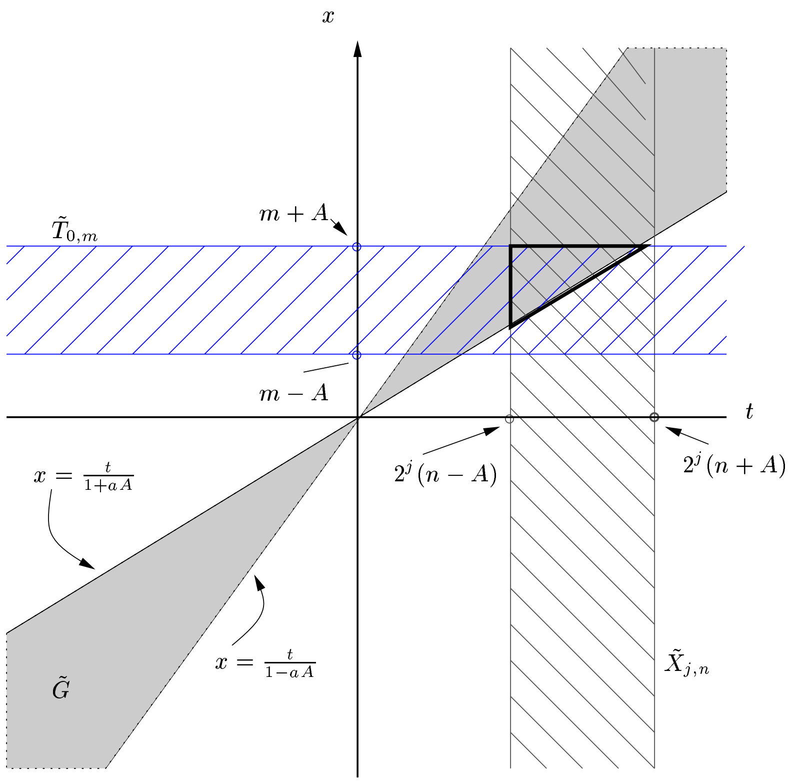


Figure 2.10: The set $\widetilde{G}$, $\widetilde{X}_{j,m}$ and $\widetilde{T}_{0,n}$ when $n > A$.

Proof. Figures 2.10 and 2.11 are useful in visualizing the proof. Let

$$\begin{aligned}
G &:= \{(t,x) : t \in \mathtt{supp}(g_x)\}, \\
T_{k,n} &:= \{(t,x) : t \in \mathbb{R}, x \in \mathtt{supp}(\psi_{k,n})\}, \quad \text{and} \\
X_{j,m} &:= \{(t,x) : x \in \mathbb{R}, t \in \mathtt{supp}(\psi_{j,m})\}.
\end{aligned}$$

Note that $G \subseteq \widetilde{G}, T_{k,n} \subseteq \widetilde{T}_{k,n}$ and $X_{j,m} \subseteq \widetilde{X}_{j,m}$, where

$$\widetilde{G} := \{(t,x) : t \in [x - |x|\alpha A, x + |x|\alpha A],\}\,,$$

$$\begin{aligned}
\widetilde{T}_{k,n} &:= \left\{(t,x) : t \in \mathbb{R}, x \in [2^{-k}(-A-n), 2^{-k}(A-n)]\right\}, \quad \text{and} \\
\widetilde{X}_{j,m} &:= \left\{(t,x) : x \in \mathbb{R}, t \in [2^{-j}(-A-n), 2^{-j}(A-n)]\right\}.
\end{aligned}$$

Figures 2.10 and 2.11 illustrate these three sets. Pictorially, the two lines enclosing $\widetilde{G}$ in Figure 2.10 are the lines $\{(t,x) : t = x(1 - A\alpha)\}$ and $\{(t,x) : t = x(1 + \alpha A)\}$ respectively.

From the definition of $\theta_{j,m,k,n}$, we can deduce that $\theta_{j,m,k,n} \neq 0$ implies

$$\widetilde{G} \cap \widetilde{T}_{k,n} \cap \widetilde{X}_{j,m} \neq \emptyset. \tag{2.11}$$

Now we analyze conditions on j, m, k and n implied by (2.11).

(a) If $|n| < 2A$, then we have $(\widetilde{T}_{0,n} \cap \widetilde{G}) \subseteq \{(t,x) : |x| \leq 3A \text{ and } |t| \leq 3A(1 + \alpha A)\}$ which implies that $\widetilde{T}_{0,n} \cap \widetilde{G}$ is a subset of

$$\{(t,x) : x \in \mathbb{R}, |t| \leq C\}, \tag{2.12}$$

where C is a constant. For (2.11) to hold, it is necessary that $\widetilde{T}_{j,m}$ has non-empty intersection with (2.12). There are two cases to consider: Suppose $(0,0) \in \widetilde{X}_{j,m}$, then $|m|$ must be less than or equal to A; Otherwise, that is $(0,0) \notin \widetilde{X}_{j,m}$, we must have $2^j(|m| - A) \leq C$. By combining both cases, we have

$$|m| < 2^j C + D,$$

where D is a constant.

By self-similarity (Lemma 2.1), we generalize the result to all k, that is, if $\beta_{j,m,k,n} \neq 0$, then

$$|m| < 2^{j-k}C + D. \tag{2.13}$$

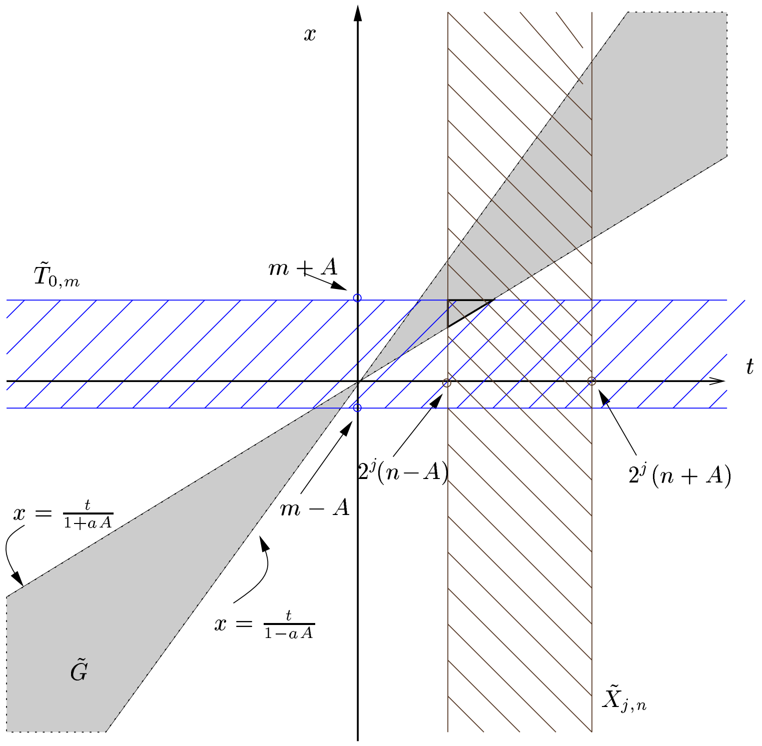


Figure 2.11: The sets $\widetilde{G}$, $\widetilde{T}_{j,m}$ and $\widetilde{X}_{0,n}$ when $0 < n < A$.

(b) Note that in the proof of part (a), the role of $\widetilde{T}_{k,n}$ and $S_{j,m}$ are interchangeable, and thus implying part(b).

(c) Suppose $|n| \geq 2A$, then we have the situation depicts in Figure 2.10.

If $n > 0$, then

$$(\widetilde{T}_{0,n} \cap \widetilde{G}) \subseteq$$
$$\{(t,x) : |x-n| \leq A, (1-\alpha A)(n-A) \leq t \leq (n+A)(1+\alpha A)\}.$$

Otherwise ($n \leq 0$), we have

$$(\widetilde{T}_{0,n} \cap \widetilde{G}) \subseteq$$
$$\{(t,x) : |x-n| \leq A, (1+\alpha A)(n+A) \leq t \leq (n-A)(1-\alpha A)\}.$$

Combining both sets, and together with the assumption that $|n| \geq 2A$, we have the weaker statement:

$$(\widetilde{T}_{0,n} \cap \widetilde{G}) \subseteq \{(t,x) : x \in \mathbb{R}, C^{-1}|n| < |t| < C|n|\}, \tag{2.14}$$

for some positive constant C. Note that $\widetilde{X}_{j,m}$ has non-empty intersection with (2.14) only if $2^j(|m|-A) \leq C|n|$. Since $|m| \geq 2A$, we have

$$|m| \leq C_1 2^{-j}|n| + A,$$

where C_1 is a positive constant. It is also necessary that $2^j(|m|+A) \geq C^{-1}|n|$ which can be written as $|m| \geq C_2^{-1}2^{-j}|n|$ since $|m| \geq 2A$.

In general, by self-similarity (Lemma 2.1), if $\beta_{j,m,k,n} \neq 0$, $|m| \geq 2A$ and $|n| \geq 2A$, then

$$C^{-1}2^{k-j}|n| \leq |m| \geq C2^{k-j}|n|, \tag{2.15}$$

for some constant C.

To simplify notations, (a), (b) and (c) are stated with common constants.

Q.E.D.

Lemma 2.3 *There is a constant J such that for any $j, k, |n| < 2A$ and $|m| < 2A$,*

$$|\theta_{j,m,k,n}| \;\leq\; J2^{-|k-j|/2}.$$

Proof.

(a) First we show $|\theta_{j,m,k,n}| \leq J2^{(k-j)/2}$.

$$\begin{aligned}
\theta_{0,m,k,n} &= \int_{-\infty}^{\infty} \psi_{k,n}(x) \int_{-\infty}^{\infty} \psi_{0,m}(t)(\alpha|x|)^{-1} g\left(\frac{t-x}{\alpha|x|}\right) dt\, dx \\
|\theta_{0,m,k,n}| &\leq \sup_{-\infty<x<\infty} \left\{ \int_{-\infty}^{\infty} \psi_{0,m}(t)(\alpha|x|)^{-1} g\left(\frac{t-x}{\alpha|x|}\right) dt \right\} \cdot \\
&\qquad\qquad \int_{-\infty}^{\infty} |\psi_{k,n}(x)|\, dx \\
&= J_1 \int_{-\infty}^{\infty} |\psi_{k,n}(x)|\, dx \\
&\leq J_2 2^{k/2}.
\end{aligned}$$

Thus,

$$|\theta_{j,m,k,n}| \leq J_2 2^{(k-j)/2}.$$

(b) Next we show $|\theta_{j,m,k,n}| \leq J2^{(j-k)/2}$. By substituting $t' = tx^{-1}$ into the definition of $\theta_{0,m,k,n}$, we have

$$\begin{aligned}
\theta_{0,m,k,n} &= \int_{0}^{\infty} \psi_{k,n}(x) \int_{-\infty}^{\infty} \psi_{0,m}(t'x)\alpha^{-1} g\left(\frac{t'-1}{\alpha}\right) dt'\, dx \\
&\quad + \int_{-\infty}^{0} \psi_{k,n}(x) \int_{\infty}^{-\infty} \psi_{0,m}(t'x)(-\alpha^{-1}) g\left(-\frac{t'-1}{\alpha}\right) dt'\, dx.
\end{aligned}$$

Consider the first term in the right hand side (the second term is similar).

$$\begin{aligned}
&\left|\int_0^\infty \psi_{k,n}(x) \int_{-\infty}^\infty \psi_{0,m}(tx)\alpha^{-1} g\left(\frac{t-1}{\alpha}\right) \, dt \, dx\right| \\
&= \left|\int_c^{2-c} \alpha^{-1} g\left(\frac{t-1}{\alpha}\right) \int_0^\infty \psi_{k,n}(x)\psi_{0,m}(tx) \, dx \, dt\right| \\
&\qquad (\text{since } \mathtt{supp}(g(\alpha^{-1}(\cdot - 1)) \subseteq [c, 2-c] \text{ where } \ c := 1 - \alpha A) \\
&\le \max_{c \le t \le 2-c} \left\{\int_0^\infty \psi_{k,n}(x)\psi_{0,m}(tx) \, dx\right\} \int_c^{2-c} \left|\alpha^{-1} g\left(\frac{t-1}{\alpha}\right)\right| \, dt \\
&\le F_3 \max_{c \le t \le 2-c} \left\{\int_0^\infty \psi_{k,n}(x)\psi_{0,m}(tx) \, dx\right\} \\
&\le F_3 \max_{c \le t \le 2-c} \left\{2^{-k/2} \int_0^\infty |\psi_{0,m}(tx)| \, dx\right\} \\
&= F_3 2^{-k/2} \int_0^\infty |\psi_{0,m}(cx)| \, dx \\
&\le F_4 2^{-k/2}.
\end{aligned}$$

Q.E.D.

Note that the above two lemmas exploit the compact support of ψ and g, but do not consider other useful properties like the smoothness of g. The next two lemmas make use of the regularity of ψ and g to obtain sharper result. (Recall that we have assumed Condition 2.1.)

Lemma 2.4 *Suppose ψ is uniformly Lipschitz q and has at least q vanishing moments, then there is a constant E such that for any j, m, k, n, where not both $|n| < 2A$ and $|m| < 2A$, we have*

$$|\theta_{j,m,k,n}| \ \le \ E \ 2^{-|k-j|(q+1/2)}.$$

Proof.

(a) First we show $|\theta_{j,m,k,n}| \leq E2^{(k-j)(q+1/2)}$. Let $\widetilde{W} := \widetilde{G} \cap \widetilde{T}_{k,n} \cap \widetilde{X}_{j,m}$, where $\widetilde{G}$, $\widetilde{T}_{k,n}$ and $\widetilde{X}_{j,m}$ are as defined in Lemma 2.2. Since either $|n| \geq 2A$ or $|m| \geq 2A$, it is necessary that $\widetilde{W} \subseteq \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0}$ or $\widetilde{W} \subseteq \mathbb{R}_{\leq 0} \times \mathbb{R}_{\leq 0}$. Therefore, we have two cases: either $n > 0$, $m > 0$ and

$$\theta_{0,m,k,n} = \int_0^\infty \int_0^\infty \psi_{k,n}(x)\psi_{0,m}(t)(\alpha x)^{-1} g\left(\frac{t-x}{\alpha x}\right) dt\, dx, \quad (2.16)$$

or $n < 0$, $m < 0$ and

$$\theta_{0,m,k,n} = \int_{-\infty}^0 \int_{-\infty}^0 \psi_{k,n}(x)\psi_{0,m}(t)(-\alpha x)^{-1} g\left(\frac{t-x}{-\alpha x}\right) dt\, dx.$$

We only consider the first case since the second case is similar.

$$\begin{aligned}\theta_{0,m,k,n} &= \int_0^\infty \int_0^\infty \psi_{k,n}(x)\psi_{0,m}(t)(\alpha x)^{-1} g\left(\frac{t-x}{\alpha x}\right) dt\, dx, \\ &= \int_0^\infty \psi_{k,n}(x) \\ &\qquad \int_0^\infty \psi_{0,m}(t'x)\alpha^{-1} g\left(\frac{t'-1}{\alpha}\right) dt'\, dx. \qquad (2.17) \\ &\qquad \text{(by substituting } t' = tx^{-1}\text{)}\end{aligned}$$

If $n > 2A$, then $\psi_{k,n}(x) = 0$ for $x < 0$. Using this fact, (2.17) can be rewritten as

$$\int_{-\infty}^\infty \psi_{k,n}(x) \int_0^\infty \psi_{0,m}(t'x)\alpha^{-1} g\left(\frac{t'-1}{\alpha}\right) dt'\, dx.$$

Otherwise, we have $m > 2A$. Since ψ has compact support, $\psi_{0,m}(t'x) = 0$ whenever t' and x have different sign. Therefore, (2.17) can again be rewritten as the above.

Since $\mathtt{supp}(g) \subseteq [-A, A]$, and $\alpha A < 1$, we have

$$\mathtt{supp}(g(\alpha^{-1}(\cdot - 1))) \subseteq [c, 2-c],$$

where $c := 1 - \alpha A$ is a positive constant.

$$\begin{aligned} \theta_{0,m,k,n} &= \int_c^{2-c} \alpha^{-1} g(\frac{t-1}{\alpha}) \int_{-\infty}^{\infty} \psi_{k,n}(x)\psi_{0,m}(tx)\, dx\, dt \qquad (2.18) \\ &= \int_c^{2-c} \alpha^{-1} t^q g\left(\frac{t-1}{\alpha}\right) \int_{-\infty}^{\infty} \psi_{k,n}(x) \left(t^{-q}\psi_{0,m}(tx)\right)\, dx\, dt. \end{aligned}$$

By Lemma A.2 (in Appendix A), since $\psi_{0,m}$ is uniformly Lipschitz q, then so is the function $t^{-q}\psi_{0,m}(xt)$ on x. Together with Theorem A.3,

$$|\int_{-\infty}^{\infty} \psi_{k,n}(x) t^{-q}\psi_{0,m}(tx))\, dx| \leq E_1 2^{k(q+1/2)},$$

for some constant E_1. Therefore

$$\begin{aligned} |\theta_{0,m,k,n}| &\leq \max_{c \leq t \leq 2-c} \left\{ \left| \alpha^{-1} t^q g\left(\frac{t-1}{a}\right) \right| \right\} \cdot \\ &\qquad \int_c^{2-c} E_1 2^{k(q+1/2)}\, dt = E_2 2^{k(q+1/2)}. \end{aligned}$$

(b) We now show $|\theta_{j,m,k,n}| \leq E2^{(j-k)(q+1/2)}$. By self-similarity (Lemma 2.1), (2.18) can be rewritten as:

$$\theta_{j,m,0,n} = \int_c^{2-c} \frac{1}{\alpha} g\left(\frac{t-1}{\alpha}\right) t^{-1/2} \int_{-\infty}^{\infty} \psi_{0,n}(x) \left(t^{1/2}\psi_{j,m}(tx)\right)\, dx\, dt$$

$$|\theta_{j,m,0,n}| \le \max_{c\le t\le 2-c}\left\{\left|\frac{1}{\alpha}g\left(\frac{t-1}{\alpha}\right)t^{-1/2}\right|\right\}\cdot$$
$$\int_c^{2-c}\left|\int_{-\infty}^{\infty}\psi_{0,n}(x)\left(t^{1/2}\psi_{j,m}(tx)\right)\,dx\right|\,dt.$$

Since $\psi_{0,n}$ is uniformly Lipschitz q,

$$\begin{aligned}|\theta_{j,m,0,n}| &\le E_3\int_c^{2-c} t^{-(q+1/2)}2^{j(q+1/2)}\,dt\\ &\le E_4 2^{j(q+1/2)}.\end{aligned}$$

Q.E.D.

Lemma 2.5 *Suppose g is uniformly Lipschitz ρ and ψ has at least ρ vanishing moment, then, for any $|n|\ge 2A$, $|m|\ge 2A$ and any j,k,*

$$\begin{aligned}|\theta_{j,m,k,n}| &\le F\ 2^{(j-k)(\rho+1/2)}|n|^{-(\rho+1)}, \quad \textit{and}\\ |\theta_{j,m,k,n}| &\le F\ 2^{(k-j)/2}|m|^{-(\rho+1)}.\end{aligned}$$

Proof.

(a) First we show $|\theta_{j,m,k,n}| \le F\ 2^{(j-k)(\rho+1/2)}|n|^{-(\rho+1)}$.

Since g is uniformly Lipschitz ρ, then so is $(\alpha x)^\rho g(\cdot/(\alpha x)-1)$.

$$\theta_{j,m,0,n} = \int_{-\infty}^{\infty}\psi_{0,n}(x)|\alpha x|^{-(\rho+1)}\int_{-\infty}^{\infty}\psi_{j,m}(t)\left(|\alpha x|^\rho g\left(\frac{t}{\alpha x}-1\right)\right)\,dt\,dx.$$

Therefore,

$$\begin{aligned}|\theta_{j,m,0,n}| &\leq \int_{-\infty}^{\infty} \left|\psi_{0,n}(x)(\alpha x)^{-(\rho+1)} F_1 2^{j(\rho+1/2)}\right| \, dx \\ &= F_1 2^{j(\rho+1/2)} \int_{n-A}^{n+A} \psi_{0,n} |\alpha x|^{-(\rho+1)} \, dx.\end{aligned}$$

Since $|n| \geq 2A$,

$$\begin{aligned}|\theta_{j,m,0,n}| &\leq F_1 2^{j(\rho+1/2)} (\alpha|n-A|)^{-(\rho+1)} \int_{-\infty}^{\infty} |\psi_{0,n}(x)| \, dx \\ &\leq F_2 2^{j(\rho+1/2)} |n|^{-(\rho+1)}.\end{aligned}$$

Generalizing to all k, we have

$$|\theta_{j,m,k,n}| \leq F_2^{(j-k)(\rho+1/2)} |n|^{-(\rho+1)}. \tag{2.19}$$

(b) Now we show $|\theta_{j,m,k,n}| \leq F \; 2^{(k-j)/2} |m|^{-(\rho+1)}$.

If $|n| \geq 2A$ and $|m| \geq 2A$, then combining (2.19) and Lemma 2.2(c), we obtain

$$\begin{aligned}|\theta_{j,m,k,n}| &< F_2 2^{(j-k)(\rho+1/2)} (C^{-1} 2^{j-k} |m|)^{-(\rho+1)} \\ &< F_3 2^{(k-j)/2} (|m|)^{-(\rho+1)}.\end{aligned}$$

Q.E.D.

2.2.5 Boundedness of T^{fov}

It is interesting to know whether the foveation operator is bounded. An operator T is bounded if there is a constant B such that for any function $f \in L^2(\mathbb{R})$,

$$\|T(f)\|_2 \leq B\|f\|_2.$$

We use the fast decays of the operator matrix to show the boundedness.

Theorem 2.6 *For a scaling function $g : \mathbb{R} \to \mathbb{R}$ and weight function $w : \mathbb{R} \to \mathbb{R}_{\geq 0}$, if*

(a) $\operatorname{supp}(g) \subseteq [-A, A]$*, for some constant A,*

(b) *g is uniformly Lipschitz $p > 0$, and*

(c) $w(x) = \alpha|x|$*, where $\alpha A < 1$, and $\alpha > 0$,*

then the operator T^{fov} is bounded.

To prove the theorem, we first find an appropriate wavelet ψ and study the matrix with entries $\langle T^{\text{fov}}\psi_{j,m}, \psi_{k,n}\rangle$. Next, by using the self-similarity and fast decay property, we apply Schur's lemma (Lemma A.1) to show that T^{fov} is bounded. This approach of proving the boundedness of an operator could also be found in [12] and [20].

We first show a few lemmas and present the proof of this theorem at the end of this section. In the rest of this section, we assume that the scaling function g and weight function w satisfy the conditions stated in the theorem. Furthermore, we assume ψ is a wavelet with support contained in $[-A, A]$, and ψ is uniformly Lipschitz q with $q \geq \frac{5}{4}$ and has at least 2 vanishing moments. In the proof of the theorem, we will show that it is also possible to "scale" up the support of g so that such a wavelet exist.

Let

$$\rho := \min\{p, 2\}. \tag{2.20}$$

Define

$$w_\ell := 2^{\ell\lambda},$$

where λ is a constant given by

$$\lambda := \max\{\frac{1}{2} - \rho, \frac{1}{4}\}. \tag{2.21}$$

The next lemma takes care of cases when $|n|$ and $|m|$ are small.

Lemma 2.7 *There is a constant B such that the followings hold.*

(a) *For any $|\widetilde{n}| < 2A$ and $\widetilde{k}$,*

$$\sum_j \sum_m w_j |\theta_{j,m,\widetilde{k},\widetilde{n}}| \leq B w_{\widetilde{k}}.$$

(b) *For any $|\widetilde{m}| < 2A$ and $\widetilde{j}$,*

$$\sum_k \sum_n w_k |\theta_{\widetilde{j},\widetilde{m},k,n}| \leq B w_{\widetilde{j}}.$$

(c) *For any $\widetilde{m}$ and $\widetilde{j}$,*

$$\sum_k \sum_{|n|<2A} w_k |\theta_{\widetilde{j},\widetilde{m},k,n}| \leq B w_{\widetilde{j}}.$$

(d) *For any $\widetilde{n}$ and $\widetilde{k}$,*

$$\sum_j \sum_{|m|<2A} w_j |\theta_{j,m,\widetilde{k},\widetilde{n}}| \leq B w_{\widetilde{k}}.$$

Proof.

(a)

$$\begin{aligned}
&\sum_j \sum_m w_j |\theta_{j,m,\widetilde{k},\widetilde{n}}| \\
&\quad = \sum_j \sum_{|m|>2A} w_j |\theta_{j,m,\widetilde{k},\widetilde{n}}| + \sum_j \sum_{|m|\leq 2A} w_j |\theta_{j,m,\widetilde{k},\widetilde{n}}| \\
&\quad \leq \underset{\text{(by Lemma 2.4)}}{\sum_{j\leq\widetilde{k}} \sum_{|m|>2A} w_j E 2^{(j-\widetilde{k})(q+1/2)}} + \underset{\text{(by Lemma 2.4)}}{\sum_{j>\widetilde{k}} \sum_{|m|>2A} w_j E 2^{(\widetilde{k}-j)(q+1/2)}} \\
&\qquad + \underset{\text{(by Lemma 2.3)}}{\sum_{j\leq\widetilde{k}} \sum_{|m|\leq 2A} w_j J 2^{(j-\widetilde{k})/2}} + \underset{\text{(by Lemma 2.3)}}{\sum_{j>\widetilde{k}} \sum_{|m|\leq 2A} w_j J 2^{(\widetilde{k}-j)/2}} \\
&\quad = \quad w_{\widetilde{k}} \sum_{j\leq\widetilde{k}} \sum_{|m|>2A} E 2^{(j-\widetilde{k})(q+1/2+\lambda)} \\
&\qquad + w_{\widetilde{k}} \sum_{j>\widetilde{k}} \sum_{|m|>2A} E 2^{(\widetilde{k}-j)(q+1/2-\lambda)} \\
&\qquad + w_{\widetilde{k}} \sum_{j\leq\widetilde{k}} \sum_{|m|\leq 2A} J 2^{(j-\widetilde{k})(1/2+\lambda)} \\
&\qquad + w_{\widetilde{k}} \sum_{j>\widetilde{k}} \sum_{|m|\leq 2A} J 2^{(\widetilde{k}-j)(1/2-\lambda)}.
\end{aligned} \tag{2.22}$$

Consider the last two terms in the right hand side. For some constant B_1,

$$\begin{aligned}
& w_{\widetilde{k}} \sum_{j \le \widetilde{k}} \sum_{|m| \le 2A} J 2^{(j-\widetilde{k})(1/2+\lambda)} + w_{\widetilde{k}} \sum_{j > \widetilde{k}} \sum_{|m| \le 2A} J 2^{(\widetilde{k}-j)(1/2-\lambda)} \\
\le \;& 4AJ w_{\widetilde{k}} \left(\sum_{j \le \widetilde{k}} 2^{(j-\widetilde{k})(1/2+\lambda)} + \sum_{j > \widetilde{k}} 2^{(\widetilde{k}-j)(1/2-\lambda)} \right) \\
\le \;& B_1 w_{\widetilde{k}}. \\
& (\text{since } \lambda < \frac{1}{2})
\end{aligned}$$

We now consider the first two terms in (2.22). By Lemma 2.2, the number of m's such that $\theta_{j,m,\widetilde{k},\widetilde{n}} \neq 0$ is less than $2^{\widetilde{k}-j} C + D$. Thus,

$$\begin{aligned}
& w_{\widetilde{k}} \sum_{j \le \widetilde{k}} \sum_{|m| > 2A} E 2^{(j-\widetilde{k})(q+1/2+\lambda)} + w_{\widetilde{k}} \sum_{j > \widetilde{k}} \sum_{|m| > 2A} E 2^{(\widetilde{k}-j)(q+1/2-\lambda)} \\
\le \;& C w_{\widetilde{k}} \left(\sum_{j > \widetilde{k}} 2^{(\widetilde{k}-j)} E 2^{(\widetilde{k}-j)(q+1/2-\lambda)} \right. \\
& \qquad \left. + \sum_{j \le \widetilde{k}} 2^{(\widetilde{k}-j)} E 2^{(j-\widetilde{k})(q+1/2+\lambda)} \right) \\
& + D w_{\widetilde{k}} \left(\sum_{j > \widetilde{k}} 2^{(\widetilde{k}-j)(q+1/2-\lambda)} + \sum_{j \le \widetilde{k}} 2^{(j-\widetilde{k})(q+1/2+\lambda)} \right) \\
\le \;& w_{\widetilde{k}} B_2. \\
& (\text{since } q + \frac{1}{2} - \lambda > 1).
\end{aligned}$$

This shows part (a). We can similarly show part (b) by switching j with k and m with n.

(c)

$$\begin{aligned}
&\sum_{k}\sum_{|n|<2A} w_k|\theta_{\widetilde{j},\widetilde{m},k,n}| \\
&\quad \le \; 4A\left(\sum_{k<\widetilde{j}} 2^{k\lambda} J 2^{(k-\widetilde{j})(1/2)} + \sum_{k\ge\widetilde{j}} 2^{k\lambda} J 2^{(\widetilde{j}-k)(1/2)}\right) \\
&\qquad\qquad \text{(by Lemma 2.3)} \\
&\quad = \; 4Aw_{\widetilde{j}}\left(\sum_{k<\widetilde{j}} J 2^{(k-\widetilde{j})(1/2+\lambda))} + \sum_{k\ge\widetilde{j}} J 2^{(\widetilde{j}-k)(1/2-\lambda)}\right) \\
&\quad \le \; w_{\widetilde{j}} B_3.
\end{aligned}$$

This shows part (c). Similarly, we have part (d) by symmetry.

Q.E.D.

Lemma 2.8 *There is a constant G such that the following bounds hold.*

(a) *For any $|\widetilde{m}| > 2A$ and $\widetilde{j}$,*

$$\sum_{k>\widetilde{j}} w_k \sum_{|n|\ge 2A} |\theta_{\widetilde{j},\widetilde{m},k,n}| \le w_{\widetilde{j}} G.$$

(b) *For any $|\widetilde{m}| > 2A$ and $\widetilde{j}$,*

$$\sum_{k\le\widetilde{j}} w_k \sum_{|n|\ge 2A} |\theta_{\widetilde{j},\widetilde{m},k,n}| \le w_{\widetilde{j}} G.$$

(c) *For any $|\widetilde{n}| > 2A$ and $\widetilde{k}$,*

$$\sum_{j<\widetilde{k}} w_j \sum_{|m|\ge 2A} |\theta_{j,m,\widetilde{k},\widetilde{n}}| \le w_{\widetilde{k}} G.$$

(d) *For any $|\widetilde{n}| > 2A$ and $\widetilde{k}$,*

$$\sum_{j \geq \widetilde{k}} w_j \sum_{|m| \geq 2A} |\theta_{j,m,\widetilde{k},\widetilde{n}}| \leq w_{\widetilde{k}} G.$$

Proof.

(a) By Lemma 2.5,

$$\begin{aligned} \sum_{|n| \geq 2A} |\theta_{\widetilde{j},\widetilde{m},k,n}| &\leq \sum_{|n| \geq 2A} F 2^{(\widetilde{j}-k)(\rho+1/2)} |n|^{-(\rho+1)} \\ &\leq G_1 2^{(\widetilde{j}-k)(\rho+1/2)}. \end{aligned}$$

Summing over all k larger than $\widetilde{j}$,

$$\begin{aligned} \sum_{k > \widetilde{j}} w_k \left(\sum_{|n| \geq 2A} |\theta_{\widetilde{j},\widetilde{m},k,n}| \right) &\leq \sum_{k > \widetilde{j}} 2^{k\lambda} G_1 2^{(\widetilde{j}-k)(\rho+1/2)} \\ &= 2^{\widetilde{j}\lambda} G_1 \sum_{k > \widetilde{j}} 2^{(\widetilde{j}-k)(\rho+1/2-\lambda)} \\ &\leq w_{\widetilde{j}} G_2. \qquad (2.23) \\ &\quad (\text{since } \lambda < \frac{1}{2}) \end{aligned}$$

(b) For any $\widetilde{m} \geq 2A$, and k, by Lemma 2.2, the number of n's such that $\theta_{\widetilde{j},\widetilde{m},k,n} \neq 0$ is less than $C|m|2^{\widetilde{j}-k}$. Combined with Lemma 2.5,

$$\begin{aligned} \sum_{|n| \geq 2A} w_k |\theta_{\widetilde{j},\widetilde{m},k,n}| &\leq 2^{k\lambda} \left(C|\widetilde{m}| 2^{\widetilde{j}-k} \right) \left(F 2^{(k-\widetilde{j})/2} |\widetilde{m}|^{-(\rho+1)} \right) \\ &\leq w_{\widetilde{j}} G_3 |\widetilde{m}|^{-\rho} 2^{(\widetilde{j}-k)(1/2-\lambda)}. \qquad (2.24) \end{aligned}$$

On the other hand, by applying Lemma 2.4, we have

$$\begin{aligned}\sum_{|n|\geq 2A} w_k|\theta_{\widetilde{j},\widetilde{m},k,n}| &\leq 2^{k\lambda}\left(C|\widetilde{m}|2^{\widetilde{j}-k}\right)F2^{(k-\widetilde{j})(q+1/2)} \\ &\leq w_{\widetilde{j}}G_4|\widetilde{m}|2^{(k-\widetilde{j})(q-1/2+\lambda)}. \qquad (2.25)\end{aligned}$$

Now, we have two inequalities: (2.24) decays as $|\widetilde{m}|$ increases but grows as k decreases; whereas it is in the other way in (2.25). Let ξ be a positive constant $\xi := \rho(\frac{1}{2}-\lambda)^{-1}$ (so that $2^{\xi(1/2-\lambda)\log_2|\widetilde{m}|} = |\widetilde{m}|^{\rho}$). Summing up all "small" k using (2.24), we have

$$\begin{aligned}\sum_{\widetilde{j}-\xi\log|\widetilde{m}|\leq k\leq\widetilde{j}} w_k|\theta_{\widetilde{j},\widetilde{m},k,n}| &\leq G_3 w_{\widetilde{j}}|\widetilde{m}|^{-\rho}2^{\xi(1/2-\lambda)\log\widetilde{m}} \\ &\leq G_5 w_{\widetilde{j}}. \\ &\quad \text{(by choice of } \xi)\end{aligned}$$

By definition of λ (2.21), if $\rho \leq \frac{1}{4}$, then $\lambda = \frac{1}{2}-\rho$, which implies $\xi = 1$. Thus

$$\xi\left(q-\frac{1}{2}+\lambda\right) = q-\rho > 1.$$

On the other hand, if $\rho > \frac{1}{4}$, then $\lambda = \frac{1}{4}$ which implies $\xi = 4\rho$. Together with the assumption that $q \geq \frac{5}{4}$,

$$\xi\left(q-\frac{1}{2}+\lambda\right) > q-\frac{1}{4} \geq 1.$$

Therefore $2^{-\xi(q-1/2+\lambda)\log_2(\widetilde{m})} \leq |\widetilde{m}|^{-1}$.

Summing up the "large" k using (2.25), and applying this inequality, we have

$$\begin{aligned}
\sum_{k<\widetilde{j}-\xi\log|\widetilde{m}|} w_k|D_3\theta_{\widetilde{j},\widetilde{m},k,n}| &\le G_4 w_{\widetilde{j}} \sum_{h<\widetilde{j}} F2^{(h-\xi\log|\widetilde{m}|-\widetilde{j})(q-1/2+\lambda)}|\widetilde{m}| \\
&\le G_6 \sum_{h<j} w_{\widetilde{j}}|\widetilde{m}|(|\widetilde{m}|^{-1})2^{(h-\widetilde{j})(q-1/2+\lambda)} \\
&\le G_7 w_{\widetilde{j}}.
\end{aligned}$$

(c) Using Lemma 2.5, it is easy to verify that, for any $|\widetilde{n}| \ge 2A$,

$$\begin{aligned}
\sum_{|m|\ge 2A} |\theta_{j,m,\widetilde{k},\widetilde{n}}| &\le \sum_{|m|\ge 2A} 2^{(\widetilde{k}-j)/2}|m|^{-(\rho+1)} \\
&\le H_1 2^{(\widetilde{k}-j)/2}.
\end{aligned}$$

Summing over all j smaller than $\widetilde{k}$,

$$\begin{aligned}
\sum_{j\le\widetilde{k}} w_j \sum_{|m|\ge 2A} |\theta_{j,m,k,\widetilde{n}}| &\le \sum_{j>\widetilde{k}} 2^{j\lambda} H_1 2^{(\widetilde{k}-j)/2} \\
&= \sum_{j>\widetilde{k}} 2^{\widetilde{k}\lambda} H_1 2^{(\widetilde{k}-j)(1/2+\lambda)} \le w_{\widetilde{k}} H_2.
\end{aligned}$$

(d) By Lemma 2.2, the number of m's such that $\theta_{j,m,\widetilde{k},\widetilde{n}} \ne 0$ is less than $C|\widetilde{n}|2^{\widetilde{k}-j}$. By applying Lemma 2.5,

$$\begin{aligned}
w_j \sum_{|m|\ge 2A} |\theta_{j,m,\widetilde{k},\widetilde{n}}| &\le 2^{j\lambda} C\left(|\widetilde{n}|2^{\widetilde{k}-j}\right) F2^{(j-\widetilde{k})(\rho+1/2)}|\widetilde{n}|^{-(\rho+1)} \\
&= 2^{\widetilde{k}\lambda} H_3 |\widetilde{n}|^{-\rho} 2^{(j-\widetilde{k})(\rho+1/2-\lambda)}. \qquad (2.26)
\end{aligned}$$

Since $\rho > 0$ and $|\tilde{n}| \geq 2A$, therefore $|\tilde{n}|^{-\rho}$ is bounded above. Thus we have,

$$\begin{aligned}
\sum_{j<\widetilde{k}} \sum_{|m|\geq 2A} w_j |\theta_{j,m,\widetilde{k},\widetilde{n}}| &\leq w_{\widetilde{k}} H_3 2^{(j-\widetilde{k})(\rho+1/2-\lambda)} \\
&\leq w_{\widetilde{k}} H_4.
\end{aligned}$$

Q.E.D.

Proof of Theorem 2.6.

Recall that to apply Lemma 2.7 and Lemma 2.8, we have to find a wavelet ψ that satisfies these assumptions:

1. $\texttt{supp}(\psi) \subseteq [-A, A]$, and

2. ψ is uniformly Lipschitz $q \geq \frac{5}{4}$ and has at least two vanishing moments.

Note that we can arbitrary scale up the support of the scaling function g as long as the rate α of the weight function scales up proportionally. For example, for any $d > 0$, let

$$\begin{aligned}
\widetilde{g}(x) &:= g(d^{-1}x), \quad \text{and} \\
\widetilde{w}(x) &:= d^{-1}\alpha|x|,
\end{aligned}$$

then the foveation operator under this new scaling function $\widetilde{g}$ and weight function $\widetilde{w}$ are the same as the original T^{fov}, although $\texttt{supp}(g) \subseteq [-dA, dA]$. Therefore, the restriction on ψ can be relaxed to the followings:

1. ψ has compact support, and

2. ψ is uniformly Lipschitz $q \geq \frac{5}{4}$ and has at least two vanishing moments.

Certainly, such wavelet exists. An example is $DAUB6$ [9].

Now, by straight forward application of Schur's lemma using Lemma 2.8 and Lemma 2.7, we have the boundedness of T^{fov}.

Q.E.D.

2.2.6 Approximation of Foveation

Given a function $f : \mathbb{R} \to \mathbb{R}$ represented by a sequence $\{a[n]\}_{n\in\mathbb{Z}}$, where

$$f = \sum_{n=-\infty}^{\infty} a[n]\phi_{0,n}, \tag{2.27}$$

and ϕ is a father wavelet. We want to compute its foveation with respect to a standard weight function and some scaling function g. Let ψ be the mother wavelet corresponds to ϕ, as noted before (section 2.2.4), if g is regular and ψ has enough vanishing moments, then the operator matrix (with respect to ψ) is dominated by the diagonal entries. This intuitively justifies the following approximation of the foveation operator.

$$T^{\mathrm{fov}} f \approx \sum_{j=1}^{\infty} \sum_{n=-\infty}^{\infty} \theta_{j,n,j,n} d_j[n] \psi_{j,n},$$

where each $d_j[n] := \langle f, \psi_{j,n} \rangle$ can be computed from $\{a[n]\}_{n=-\infty}^{\infty}$ by the fast wavelet transform. We further approximate each $\theta_{j,n,j,n}$ by a precomputed value $c_j[n]$, and summing j and n over finite range:

$$T^{\text{fov}} f \approx \langle f, \phi_{\ell_0,0} \rangle \phi_{\ell_0,0} + \sum_{j=1}^{\ell_0} \sum_{n=0}^{N} c_j[n] d_j[n] \psi_{j,n}. \tag{2.28}$$

Call the matrix/array $\{c_j[n]\}_{j,n}$ the *mask*.

We could precompute (numerically) $c_j[n]$ directly from the definition of $\theta_{j,n,j,n}$ or from the approximation (2.10) described in section 2.2.3. Care must be taken in computing $\theta_{j,n,j,n}$. If it is computed directly from the definition, then we need to decompose the kernel $g_x(t)$ with respect to $\phi_{0,0}$ and this is not easy. In practice, we sample the kernel uniformly at a unit spacing and thus inducing error, especially when j and x are small. Figure 2.12 and Figure 2.15(b) are computed in this way. Note that in Figure 2.12, due to the error, self-similarity does not hold when j is small. A remedy is to compute $\theta_{j,n,j,n}$ for some large j (typically, $j > 3$) and then extending it to other scales using self-similarity.

The given function may not represented as in (2.27). For example, it may be represented by the uniform sample $\{\tilde{a}[n]\}_{n\in\mathbb{Z}}$ where each $\tilde{a}[n] := f(n)$. In this case, the fast wavelet transform is no more exact. Consequently, some properties, in particular, self-similarity, does not hold.

Since we usually do not know the precise method by which the function is sampled, and in any case, this sampling error is small when the scale j is large, we could still use the self-similar mask which is computed in high resolution (large j).

The precomputed mask is only applicable for a specific weight function. In the thinwire application, a faster method is required. We precompute a "lookup" table L of size L_{size}, with respect to the standard weight function

$$w_0(x) := \alpha_0|x|,$$

where α_0 is some small constant. Typically, a good choice of α_0 is $(L_{\text{size}})^{-1}$, however, for convenience in discussion, we assume that $\alpha_0 = 1$. Hence, each $L[k]$ stores the precomputed value of $\theta_{1,k}^{1,0}$. For entries in other scales or entries correspond to different weight functions, their values are "looked-up" from this table.

Let us define $\theta_{j,n}^{\alpha,\beta}$ to be the diagonal entries $\theta_{j,n,j,n}$ correspond to the foveation where the standard weight is $w(x) = \alpha|x| + \beta$.

To approximate $\theta_{j,n}^{\alpha,\beta}$, the lookup procedure returns $L[k]$, where

$$k := \lfloor \alpha(|n| + 2^{-j}\beta) \rfloor.$$

If ($k > L_{\text{size}}$), then the value zero is returned. Recall that $L[k] = c_1^{1,0}[k]$.

In particular, if $\beta = 0$ but $\alpha \neq 1$, then the lookup amounts to a dilation by a factor of α. If $\alpha = 1$ but $\beta \neq 0$, then the lookup amounts to a translation. This lookup procedure can be justified by using the similar trick in deriving (2.10). The details are shown in the next few paragraphs.

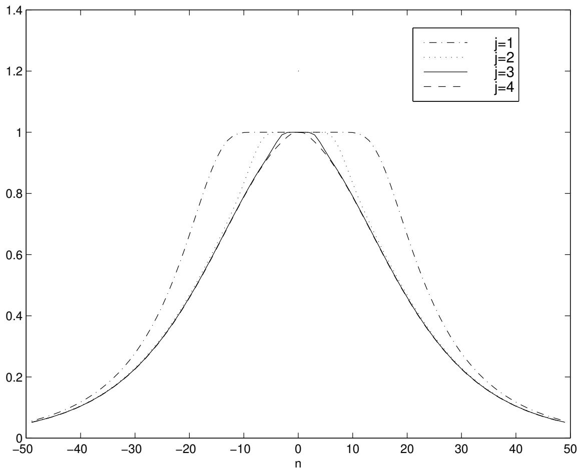


Figure 2.12: Numerically computed $\theta_{j,n,j,n}$ plotted as a function of n. The samples are taken uniformly at a unit spacing.

Different rate α. Assuming n is large and $\frac{n}{\alpha}$ is an integer, let us compare $\theta_{1,n}^{1,0}$ and $\theta_{1,\frac{n}{\alpha}}^{\alpha,0}$ for some $\alpha > 0$.

$$\begin{aligned}\theta_{1,n}^{1,0} &= \int_{-\infty}^{\infty} \psi_{1,n}(x) \int_{-\infty}^{\infty} \frac{1}{|x|} g\left(\frac{t-x}{|x|}\right) \psi_{1,n}(t)\, dt\, dx \\ &= \int_{-A}^{A} \psi_{1,0}(x) \int_{-\infty}^{\infty} \frac{1}{|x+n|} g\left(\frac{t-x}{|x+n|}\right) \psi_{1,0}(t)\, dt\, dx.\end{aligned}$$

On the other hand,

$$\begin{aligned}\theta_{1,\frac{n}{\alpha}}^{\alpha,0} &= \int_{\frac{n}{\alpha}-A}^{\frac{n}{\alpha}+A} \psi_{0,\frac{n}{\alpha}} \int_{-\infty}^{\infty} \frac{1}{\alpha|x|} g\left(\frac{t-x}{\alpha|x|}\right) \psi_{0,\frac{n}{\alpha}}(t)\, dt\, dx \\ &= \int_{-A}^{A} \psi_{0,0} \int_{-\infty}^{\infty} \frac{1}{|\alpha x+n|} g\left(\frac{t-x}{\alpha|x|+n}\right) \psi_{0,0}(t)\, dt\, dx.\end{aligned}$$

The outer integral integrates x from $-A$ to A. Therefore, for large n, $|x|+n$ is a good approximation of $\alpha|x|+n$, and we could approximate $\theta_{1,n}^{\alpha,0}$ by $\theta_{1,\frac{n}{\alpha}}^{1,0}$.

Figure 2.13 confirms this approximation numerically. Recall that $\theta^{1,0}_{1,\frac{n}{\alpha}}$ is precomputed and stored in $L[\frac{n}{\alpha}]$.

Different foveal resolution β. With non-zero β, the self-similarity property no long holds. This is intuitively clear since the foveal no longer has "infinite" precision, and thus $\theta^{1,\beta}_{j,0}$ vanishes for small j.

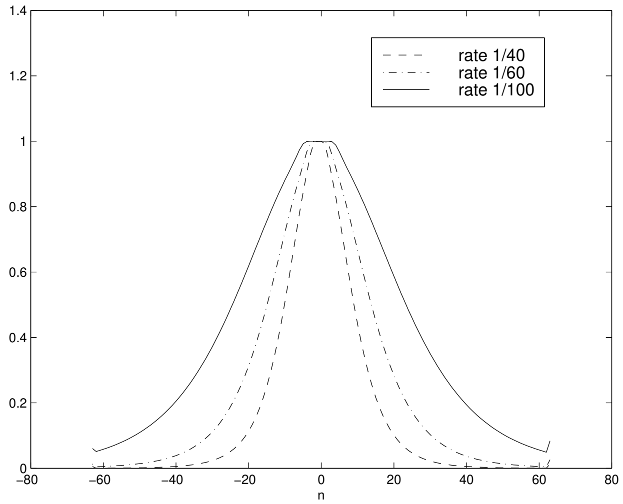
$$\begin{aligned}
\theta^{1,\beta}_{j,0} &= \int_{-\infty}^{\infty} \psi_{j,n}(x) \int_{-\infty}^{\infty} \psi_{j,n}(t) \frac{1}{|x|+\beta} g\left(\frac{t-x}{|x|+\beta}\right) dt\, dx \\
&= \int_{-\infty}^{\infty} \psi_{0,n}(x) \int_{-\infty}^{\infty} \psi_{0,n}(t) \frac{1}{|x|+2^{-j}\beta} g\left(\frac{t-x}{|x|+2^{-j}\beta}\right) dt\, dx \\
&= \int_{n+2^{-j}\beta-A}^{n+2^{-j}\beta+A} \psi_{0,n+2^{-j}\beta}(x) \\
&\qquad \int_{-\infty}^{\infty} \psi_{0,n+2^{-j}\beta}(t) \frac{1}{|x-2^{-j}\beta|+2^{-j}\beta} g\left(\frac{t-x}{|x-2^{-j}\beta|+2^{-j}\beta}\right) dt\, dx.
\end{aligned}$$

If $n + 2^{-j}\beta - A$ is large, we have $|x - 2^{-j}\beta| + 2^{-j}\beta = |x|$. Therefore, we approximate $\theta^{1,\beta}_{j,0}$ by

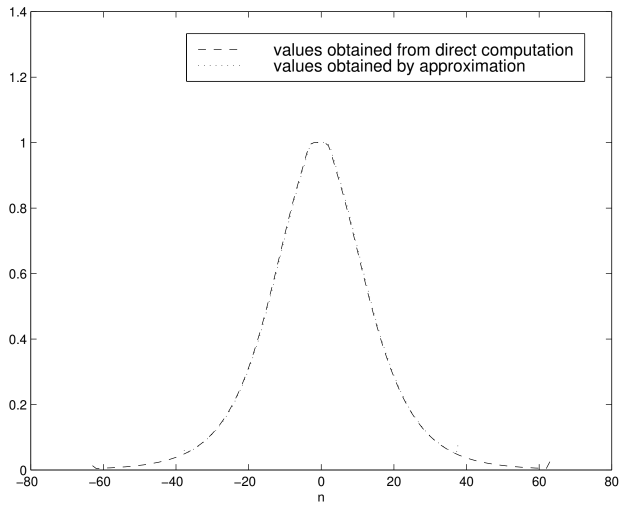
$$\begin{aligned}
\theta^{1,\beta}_{j,n} &\approx \int \psi_{0,n+2^{-j}\beta}(x) \int \psi_{0,n+2^{-j}\beta}(t) \frac{1}{|x|} g\left(\frac{t-x}{|x|}\right) dt\, dx \\
&= \theta^{1,0}_{1,k}.
\end{aligned}$$

where $k := n + 2^{-j}\beta$.

Generalization to two dimensions. We now consider raster images. Given an image represented by $N \times N$ pixels $\{p[i,j]\}_{0 \le i,j < N}$. We consider



(a)



(b)

Figure 2.13: (a) Entries of $\theta_{1,n}^{\alpha,0}$ with different rate α plotted as a function of n. (b) Approximating the entries with rate $\alpha = 1/60$ by dilating the entries with rate $\alpha = 1/100$.

the "non-standard" two dimensional wavelet. Let

$$
\begin{aligned}
\Phi_{j,m,n}(x,y) &:= \phi_{j,m}(x)\phi_{j,n}(y), \\
\Psi^{\mathrm{d}}_{j,m,n}(x,y) &:= \psi_{j,m}(x)\psi_{j,n}(y), \\
\Psi^{\mathrm{v}}_{j,m,n}(x,y) &:= \psi_{j,m}(x)\phi_{j,n}(y), \quad \text{and} \\
\Psi^{\mathrm{h}}_{j,m,n}(x,y) &:= \phi_{j,m}(x)\psi_{j,n}(y).
\end{aligned}
$$

As is in the one dimensional case, we assume that

$$I = \sum_{n=0}^{N-1}\sum_{m=0}^{N-1} p[n,m]\Phi_{0,n,m}.$$

The foveation of I, with respect to a weight function w and a separable scaling g, is approximated by

$$\langle I, \Phi_{\ell_0,0,0}\rangle \Phi_{\ell_0,0,0} + \sum_{k,m,n,j} c^k_j[m,n] d^k_j[m,n] \Psi^k_{j,m,n},$$

where

$$d^k_j[m,n] := \langle I, \Psi^k_{j,m,n}\rangle,$$

and each $c^k_j[m,n]$ is an approximation of the diagonal entries,

$$\int_{-\infty}^{\infty} dy \int_{-\infty}^{\infty} dx\ \Psi^k_{0,m,n}(x,y) \int_{-\infty}^{\infty} dt \int_{-\infty}^{\infty} ds\ \Psi^k_{0,m,n}(s,t) g_{w(x,y)}(s,t).$$

Since $w(x,y)$ is not separable, we can not separate the scaling function in the above integration. However, by assuming that $w(x,y)$ remains constant in the domain $[m-A, m+A] \times [n-A, n+A]$, $g_{w(x,n)}(s)g_{w(m,y)}(t)$ approximates $g_{w(x,y)}(s,t)$. Using this approximation, we have a simpler form, through which we derive a lookup procedure similar to that in one dimension.

To precompute the two dimensional lookup table directly from the definition is computational intensive, especially when the support of ψ or g is large. By using similar argument in the previous paragraph, we have

$$c_j^k[m,n] = \begin{cases} c_j[r] \cdot b_j[r], & \text{if } k = \text{v}, \\ b_j[r] \cdot c_j[r], & \text{if } k = \text{h}, \\ c_j[r] \cdot c_j[r], & \text{otherwise}, \end{cases} \tag{2.29}$$

where $r := \sqrt{m^2 + n^2}$, each $c_j[m]$ is the mask in one dimension and each $b_j[n]$ is an approximation of

$$\langle \phi_{j,m}, T^{\text{fov}} (\phi_{j,m}) \rangle.$$

The approximation (2.29) not only suggests a way to compute $c_{j,m,n}^k$ it also suggests a way to reduce the size of the lookup table: instead of keeping a two dimensional mask, we could keep two masks $\{c_j[m]\}, \{d_j[m]\}$, and compute $c_j^k[m,n]$ as required.

As observed in section 2.2.3, $\langle \phi_{j,m}, T(\phi_{j,m}) \rangle$ decays much slower than $\langle \psi_{j,m}, T(\psi_{j,m}) \rangle$. Thus the vertical component has a slower decays compares to the diagonal component. Figure 2.14(a) shows the contour plot of a mask and Figure 2.16(a) is the foveation computed using this mask.

Another speedup in precomputation is by assuming the entries are symmetric around the fovea. Thus, we only have to compute $c_1^k[0,m]$ and then rotate it to fill out the lookup table.

A simplified approximation. An interesting simplification is by further approximating $c^k_{j,m,n}$ by it's thresholded value $TH(\theta_{j,n,j,n})$, where

$$\mathrm{TH}(x) = \begin{cases} 1 & \text{if } x > D, \text{ and} \\ 0 & \text{otherwise,} \end{cases}$$

for some constant D. Call this simplified mask the *0-1 mask*.

Unlike the previous method, most coefficients here could be completely excluded from the reconstruction process. Another way of viewing this simplified approximation is by treating $\{\mathrm{TH}(\theta_{j,n,j,n})\}$ as a indicator, where each entry indicates whether the corresponding coefficient is to be included. Figure 2.14(b) illustrates a self-similar 0-1 mask.

This method of using a 0-1 mask to produce "foveated" image is essentially the well-known technique of Burt [5]. In a certain sense, we give a justification of this method by arguing that it is indeed an approximation of the foveation operator with standard weight function.

The advantages of using the 0-1 mask, beside its simplicity, is the computational speedup in the reconstruction process. As observed in the introduction of this chapter, foveation is also applied to speedup computation, which is achieved by ignoring most of the wavelet coefficients. For example, in [7], the projection of the volume data onto a plane is computed by performing a texture mapping operation for each coefficient. Note that the original approximation scheme is not suitable.

Figure 2.16 and Figure 2.17 shows the approximation of Figure 2.15(a) using DAUB6 and Haar wavelet respectively.

Summary. Let us summarize the methods described in this section.

1. We approximate the operator matrix of T^{fov} by a diagonal matrix.
2. Each diagonal entry is precomputed directly from the definition or from the approximation (2.10).
3. As it is impossible to store the diagonal entries for all standard weight functions, we precompute and store the diagonal entries of one standard weight in a lookup table. For other weight functions, the values are derived from this table.
4. In two dimensions, by using (2.29), we could speedup the computation of the lookup table, and reduce the size of the table.
5. The algorithm could be further simplified by thresholding each entries.

2.2.7 Transmission Scheme

We now propose a general interactive progressive scheme motivated by the above approximation. Let

$$d_j^k[m,n] := \langle I, \Psi_{j,m,n}^k \rangle.$$

For simplicity, assume that the interactions between the client and the server occur at discrete times $t = 1, 2, \ldots$. At time t, the server keeps a multi-fovea

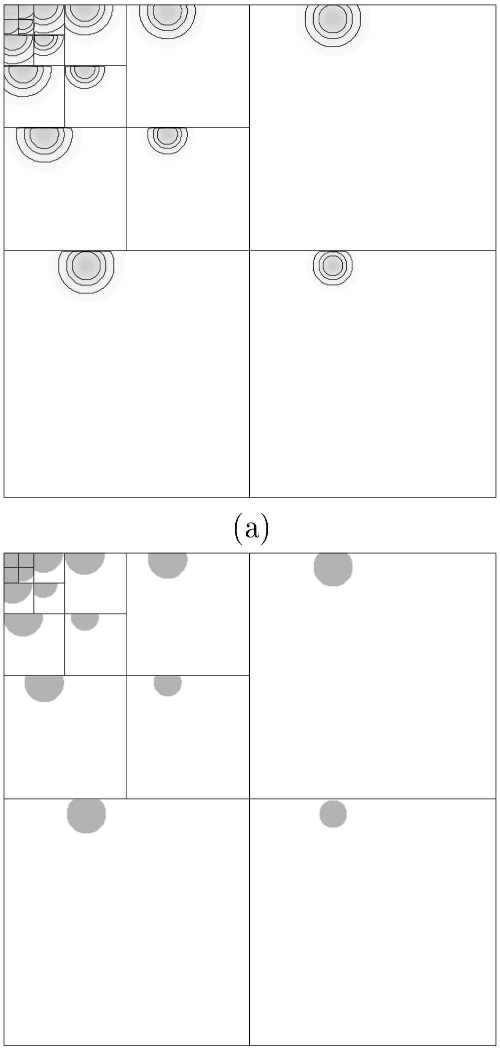
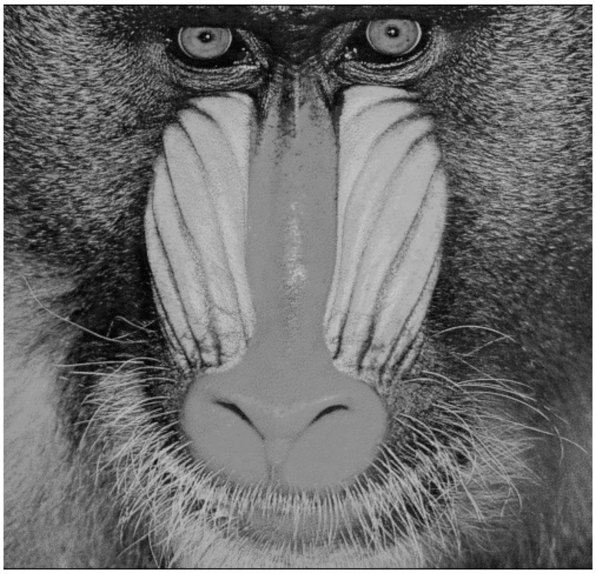


Figure 2.14: (a)The contour plot of a mask. Note that the vertical component and diagonal component are different. (b) The simplified 0-1 Mask.



(a)

(b)

Figure 2.15: Foveation with gaze point at the 'left eye' and with rate $\alpha = 1/(80 \text{ pixels})$. (a) Original image I. (b) Computed directly from the definition. The pixels in the original image take integer value in $\{0, \ldots, 255\}$.

(a)

(b)

Figure 2.16: Approximation of Figure 2.15(b). (a) Computed using the mask as shown in Figure 2.14(a). (b) Computed using the 0-1 mask as shown in Figure 2.14(b). The coefficients are not quantized; they are represented up to the precision of their floating point representation.

(a)

(b)

Figure 2.17: Foveation using Haar wavelet. The weight function is same as that in Figure 2.16. (a) Computed using the diagonal entries. (b) Computed using the 0-1 mask.

weight function w_t and a time varying mask $C = \{c_\ell^k[m,n]\}_{0<\ell\le\ell_0, 0\le m,n<N}$. Here, the mask is used as follows: for simplicity, if the mask is 0-1, then it tells whether a particular coefficient has been received or not. In general, it is a value in [0,1]. Further, each mask value $c_\ell^k[m,n]$ is non-decreasing over time.

Initially, $w_0(x) := 2^{\ell_0}$ and the corresponding mask C are initialized accordingly using the lookup table. In the first interaction, the server sends the coefficient $a := \langle \Phi_{\ell_0,0,0}, I\rangle$ and a constant $\triangle$ to the client. For each subsequent interaction, say the interaction t, the followings are performed.

1. The client sends a standard weight function $\hat{w}$ (represented by its foveal resolution, rate and gaze point) to the server.

2. Upon receipt of $\hat{w}$, the server performs the following:

    (a) From the lookup table, the server computes $\{\hat{c}_j^k[m,n]\}_{k,j,m,n}$ which is the mask for $\hat{w}$.

    (b) For each k, j, m and n such that

    $$\hat{c}_j^k[m,n] > c_j^k[m,n],$$

    the server sends

    $$\frac{\lfloor d_j^k[m,n]\hat{c}_j^k[m,n]\triangle\rfloor}{\hat{c}_j^k[m,n]\triangle} - \frac{\lfloor d_j^k[m,n]c_j^k[m,n]\triangle\rfloor}{c_j^k[m,n]\triangle}$$

    to the client, and resets

    $$c_j^k[m,n] \leftarrow \hat{c}_j^k[m,n].$$

3. The client reconstructs the foveated image.

It is easy to check that the reconstructed image in step 3 is

$$a\Phi_{\ell_0,0,0} + \sum_{k,j,m,n} \frac{\lfloor c_j^k[m,n] d_j^k[m,n] \triangle \rfloor}{\triangle} \Psi_{j,m,n}^k.$$

We may simplify the mask by rounding each $c_j^k[m,n]$ to its nearest power of 2. In this case, each $\log_2 c_j^k[m,n]$ indicates the number of bits (of $d_j^k[m,n]$) that has been send to the client. To illustrate, suppose the value of a coefficient $d_1^{\mathrm{v}}[0,0]$ is $(b_1b_2b_3b_4)_2$ in binary representation, and $\hat{c}_1^{\mathrm{v}}[0,0] = 2^{-3}$ in the first round of interactions, then the server will send b_1 to the client in step 2(b). During the next round, if $\hat{c}_1^{\mathrm{v}}[0,0] = 2^{-1}$, then the server just has to send two extra bits b_2b_3 to the client.

If the mask is further simplified to a 0-1 mask, then step 2(c) can be rewritten as:

2(c) For each k, j, m and n such that

$$\hat{c}_j^k[m,n] - c_j^k[m,n] = 0$$

the server sends $\lfloor d_j^k[m,n] \triangle \rfloor$ to the client, and resets

$$c_j^k[m,n] \leftarrow 1.$$

Here, each entry in the mask indicates whether the corresponding coefficient has been send. In Chapter 4, we describe an implementation using this simplified transmission scheme.

2.3 Weighted Distance

2.3.1 Introduction

In the context of progressive transmission, the performance of a scheme on an image I is usually measured by the distortion of I_t from I, where I_t is the image reconstructed by the client at time t. The distortion is usually the mean square error $\|I - I_t\|_2^2$. How should we measure the "goodness" of a scheme when the client requests for a foveated image? To take $\|T^{\text{fov}}(I) - I_t\|_2^2$ as the distortion is not adequate. Imagine a situation in our thin wire application, where a client requests a foveated image but the server managed to sent the original image I. By using $\|T^{\text{fov}}(I) - I\|_2^2$ as a measure, the performance of the server is "poor". On the contrary, the original image may be preferred by the client.

Instead of taking $\|T^{\text{fov}}(I) - I_t\|_2^2$ as a measure, we use a weighted norm between I and I_t as the underlying measure in the transmission scheme. In the usual setting, mean square error is measured on the retinal plane. Following the same spirit as is in the definition of foveation operator, we apply the mean square error in the visual cortex plane, which corresponds to a weighted mean square error in the retinal plane. A optimal approximate foveated image in this setting, loosely speaking, is the image that minimizes its weighted distance from the original image but represented by a fixed (predefined) number of bits.

Image compression and progressive transmission. One of the early papers on progressive transmission was by Sloan and Tanimoto [38]. Since then, it is an active research area and most current image compression schemes include features of progressive transmission. The popular graphic format GIF and JPEG [45] already have progressive transmission mode. A survey on progressive transmission could be found in [44].

Under a progressive transmission scheme, an image I is first transformed and stored as a linear bit stream $b_0, b_1, \ldots, b_n$. For any t, it is possible to reconstruct an approximation I_t of the original I based on $b_0, b_1, \ldots, b_t$. It is desirable that the distortion is bounded by ϵ_t, that is

$$||I_t - I||_2 < \epsilon_t, \tag{2.30}$$

and ϵ_t has fast decay as t increases.

One would argue that the mean square error in (2.30) may not be the right measure. It is well known that the sensitivity of the human vision varies across different frequencies [23]. However, mean square error is still of much interests since it is easy to manipulate mathematically. Many compression schemes, for example JPEG, use a weighted mean square error as a guide for compression.

Any lossy image compression scheme naturally suggests a possible progressive transmission scheme, which could be obtained by progressively improving the quality of compression. Most lossy image compression schemes could be described in the framework of transform code. Under this framework, an image is encoded in three steps: transformation, quantization and lossless compression.

Transformation. This first step applies an invertible linear transformation to the image, in other words, it decomposes the image with respect to a base $\mathcal{B}$. There are many types of such transformation. To name a few, the Karhunen-Loeve transformation (also known as principal orthogonal decomposition), which is optimal in the sense that it "completely" decorrelates the image by diagonalizing the covariance matrix; the DCT (Discrete Cosine Transform) which is used by JPEG; and subband coding. A subband coding is determined by a pair of filters, a low-pass filter H and a high-pass filter G. Its transform code is obtained by repetitive application of H and G follows by decimation on the image. Thus, the transform code is a hierarchical representation of the image. The fast wavelet transform, Gaussian and Laplacian pyramids could be viewed as a form of subband coding.

It is desirable that the base $\mathcal{B}$ is orthonormal with respect to the 2-norm. If this is the case, then the transformation has the energy invariant properties which implies that an error ϵ occurs in a coefficient induces a same error ϵ in the represented image. This ability of predicting error is very useful in analyzing and design compression scheme. Furthermore, orthonormal base tends to decorrelate the image better.

Quantization. After transformation, the coefficients $\{a_m\}_{m=1}^M$ are then *quantized*, that is, each coefficient a_m is approximated by $\widetilde{a}_m$ taken from a finite set of values. The quantization error is the error $a_m - \widetilde{a}_m$. This steps contributes to the lossy feature of the transform code.

A quantizer Q is a staircase function characterized by $x_i, i = 0, \ldots, N-1$ and $b_i, i = 0, \ldots, N$:

$$Q(x) = x_i \quad \text{if} \quad x \in [b_i, b_{i+1}).$$

Call each $[b_i, b_{i+1})$ a bin. In all quantizers considered in this report, $x_i := \frac{1}{2}(b_i + b_{i+1})$. In the case where all bins have equal size, we call it the *uniform quantizer*. We are only interested in scalar quantization where each $\widetilde{a}_m$ is determined solely by the value of a_m, as opposed to the vector quantization where $(\widetilde{a}_1, \widetilde{a}_2, \ldots, \widetilde{a}_M)$ are jointly determined by the vector $(a_1, a_2, \ldots, a_M)$.

For a random source A and a quantizer Q, let $\widetilde{A} := Q(A)$. If the number of bins is fixed, say N, to minimize the expected mean square error $E\{(A - Q(A))^2\}$, Lloyd-Max algorithm [24, 15] gives the optimal solution where smaller bins are "allocated" to more popular values. However, if we fix the expected mean square error, say D, and would like to minimize the entropy of $\widetilde{A}$, surprisingly, the uniform quantizer outperforms the quantizer obtained by Lloyd-Max algorithm; in fact it is near optimal for small D (or large number of bins) assuming A is well-behaved (for example, when A is Laplacian [47] or the quantizer and A satisfy the high bit rate assumption described in [21]). Note that the entropy of $\widetilde{A}$ is a lower bound for the average bit size needed to code $\widetilde{A}$ [34].

In the context of progressive transmission, there are two types of scalar quantization: multiscale and embedded quantization. In multiscale quantization, the data is first quantized by a coarse quantizer and the quantization error is then quantized by a finer quantizer. In the embedded quantization [43], the output of the coarse quantizer coincide with the output of the finer quantizer applied in the next step. As a result, for each refinement, the refined quantized values can be obtained by concatenation of the previous values with the additional bits. On the other hand, in the case of multiscale quantization, an arithmetic addition is required for each coefficient. Therefore, embedded quantization is computationally more efficient. Furthermore it is also more efficient in achieving low distortion [43].

Lossless Compression. This step is also known as *channel compression*. After quantization, the image is represented as a sequence of bytes. Lossless compression applies an invertible transformation to remove redundancy. A major framework for lossless compression scheme is entropy encoding. Let A to be a random source which takes value among a finite set of symbols $\{a_i\}_{1\leq i\leq k}$. An entropy coding scheme determines a binary coding for the symbols so as to minimize the expected bit size. Let

$$p_i := \mathrm{Prob}(A = a_i).$$

The entropy of A is defined as

$$-\sum_{i=1}^{k} p_i \log p_i.$$

The Shannon theorem shows that the entropy is a lower bound for the expected bit size to encode A. Popular entropy encoding schemes are Huffman coding and arithmetic coding.

Many effective schemes in fact are not based on entropy coding. Perhaps this is because entropy coding assumes that the A_i's are independent, which is not the case for most transform code. For example, small valued coefficients tends to cluster. A simple compression that exploit this observation is run-length code, which encodes a consecutive sequence of n zeros by the integer n. This simple compression turns out to be very effective and JPEG employs run-length coding followed by an entropy coding.

Consider the wavelet coefficients of an image. By visual inspection, one could conclude that there is coherence across the scales. Methods that exploit this observation is the zero-tree developed by Shapiro [35] and the predictive method by Simoncelli and Buccigrossi [36].

Using transform code to generate foveated images. Most current image compression schemes are concerned with uniform progressive transmission. However, we could easily generalize them to to space-variant progressive transmission by using different bin size across the spatial domain. This is especially so in the cases of wavelet transformed code, since wavelet is space localized. For JPEG, since an image is first subdivided into non-overlapped blocks of 8×8 sub-image, and each block is treated independently, they can be easily modified to support the space variant progressive transmission. So, the question now is, given a standard weight function and scaling function, how should the quantization be done in order to approximate the corresponding foveated image. We will focus on this question.

Bits Allocation of Transform Code. Let us treat the three steps of transform code more formally but restrict the discussion to the cases where the base $\mathcal{B}$ is orthonormal, and where scalar quantization and entropy encoding are applied in the last two steps.

We follow the approach in [21]. Let $\mathcal{B} = \{g_m\}_{0 \le m < N}$ be an orthonormal basis. A function/image f is represented by its coefficients $\{a_m\}$ with respect to $\mathcal{B}$ where each $a_m = \langle f, g_m \rangle$.

Let Y, $\{A_m\}_{0 \le m < N}$ and $\{\widetilde{A}_m\}_{0 \le m < N}$ to be the random variables of the signals/images f to be coded, the coefficient $\{a_m\}_{0 \le m < N}$ and the quantized $\{\tilde{a}_m\}_{0 \le m < N}$ respectively. Thus

$$Y = \sum_{m=0}^{N-1} A_m g_m,$$

and each

$$A_m = \langle Y, g_m \rangle.$$

Let $\widetilde{Y}$ to be the random variable of the represented function (after quantization),

$$\widetilde{Y} = \sum_{m=0}^{N-1} \widetilde{A}_m g_m.$$

Since g_m is orthonormal, the expected mean square error of approximating Y is

$$E\{\|Y - \widetilde{Y}\|_2^2\} = \sum_{m=0}^{N-1} E\{|A_m - \widetilde{A}_m|^2\}\|g_m\|_2^2 = \sum_{m=0}^{N-1} E\{|A_m - \widetilde{A}_m|^2\}. \quad (2.31)$$

Let r_m be the entropy of $\widetilde{A}_m$. Consider the following problem.

(a) Given a random source A_{m_0} and a fixed constant D, what is the quantizer such that the expected mean square error is D and the entropy of $\widetilde{A}_{m_0}$ is minimized.

Let

$$r := \sum_{m=0}^{N-1} r_m.$$

The question (a) can be further extended to the whole function/image:

(b). Given a random image Y and a base $\mathcal{B}$, a fixed constant D, what is the quantizer such that the expected mean square error is D and r is minimized.

Note that r is the entropy of the joint distribution of $\{\widetilde{A}_m\}$ only if all $\widetilde{A}_m$ are independent; in general, r is smaller. As observed before, we are only interested in scalar quantization, thus we still use r to measure the average bit size.

It can be shown that if the base induces equal distortion in each direction of g_m, then the uniform quantizer is optimal. This result can be extended to a weighted case where the expected mean square error (2.31) is replaced by a weighted mean square error,

$$\sum_{m=0}^{N_1} \frac{1}{w_m^2} E\{|A_m - \widetilde{A}_m|^2\}. \tag{2.32}$$

In this case, the bin size for A_m is $\triangle w_m$, where $\triangle$ is a constant depends on the expected weighted error D.

In this section, we consider a problem similar to (b) but instead of using the 2-norm (2.31) as a measure of error, we use a weighted norm determined by a standard weight function. The expected weighted norm is later approximated by (2.32).

2.3.2 Weighted Norm in One Dimension

Given a weight function $w : \mathbb{R} \to \mathbb{R}_{>0}$ where $w(x) \geq 0$ for all x and $w(x) = 0$ only for finitely many x. Define, for two one-dimensional functions $f : \mathbb{R} \to \mathbb{R}$, $g : \mathbb{R} \to \mathbb{R}$, and a weight function w,

$$\langle f, g \rangle_w := \int \frac{f(x)g(x)}{w(x)}\, dx.$$

Define the weighted norm of f as

$$\|f\|_w := \sqrt{\langle f, f \rangle_w}. \tag{2.33}$$

It is easy to verify that $\|.\|_w$ is a norm.

Instead of using the mean square error to measure distance between images, we take the weighted norm as a measure and consider the following expected weighted error.

$$E\{\|Y - \widetilde{Y}\|_w^2\}.$$

We are interested in the case where $\mathcal{B}$ is a wavelet base. Unfortunately, (2.31) is no longer valid since $\mathcal{B}$ is not orthonormal under $\|.\|_w$, since $\langle \psi_{j,m}, \psi_{k,m} \rangle_w \neq 0$ for $(j,m) \neq (k,m)$.

Note that for a function f,

$$\|f\|_w^2 = \sum_{j,m} \langle f, \psi_{j,m} \rangle^2 \|\psi_{j,m}\|_w^2 + 2 \sum_{j,m} \langle f, \psi_{j,m} \rangle \sum_{(k,n) \neq (j,m)} \langle f, \psi_{k,n} \rangle \langle \psi_{j,m}, \psi_{k,n} \rangle_w.$$

Lemma 2.9 shows that for any weight function $w(t) := \alpha|t - \gamma| + \beta$, and for each j, m, $\sum_{k,n} \langle \psi_{j,m}, \psi_{k,n} \rangle_w$ is small. Thus, we take the following as an approximation of $E\{\|Y - \widetilde{Y}\|_w^2\}$:

$$\sum_{j,m} E\{|A_{j,m} - \widetilde{A}_{j,m}|^2\} \|\psi_{j,m}\|_w^2.$$

We further approximate each $\|\psi_{j,m}\|_w$ by $(w_{j,m})^{-1}$ where

$$w_{j,m} := \sqrt{\alpha} 2^{j/2} \sqrt{|m - 2^{-j}\gamma| + 2^{-j}\frac{\beta}{\alpha}}. \tag{2.34}$$

Here, $\{w_{j,m}\}$ play the role of weight in (2.32). We could temporarily ignore the first term $\sqrt{\alpha}$ since it is a common factor for all $w_{j,m}$. For convenience, we fix the gaze point at the origin (since generalization to other locations is easy). Thus, we only consider weight function of the following form:

$$w(x) = |x| + \beta_0,$$

where $\beta_0 := \frac{\beta}{\alpha}$.

Lemma 2.9 *Suppose ψ has compact support $\mathtt{supp}(\psi) \subseteq [-A, A]$, then there is a constant G, such that for any weight function $w(x) = |x| + \beta_0$, $|\widetilde{m}| > 2A$ and $\widetilde{j}$,*

(a)

$$\sum_{(k,n)\neq(\widetilde{j},\widetilde{m})} \langle \psi_{\widetilde{j},\widetilde{m}}, \psi_{k,n} \rangle_w < G 2^{-\widetilde{j}} \frac{1}{\widetilde{m}^2 + 2^{-\widetilde{j}}\beta_0},$$

(b)

$$\|\psi_{\widetilde{j},\widetilde{m}}\|_w^2 = 2^{-\widetilde{j}} \frac{1}{|\widetilde{m}| + 2^{-\widetilde{j}}\beta_0} + O\left(\widetilde{m}^{-2}\right).$$

Proof.

(a) First, consider the case where $k > \widetilde{j}$.

$$\begin{aligned}
&\langle \psi_{\widetilde{j},\widetilde{m}}, \psi_{k,n} \rangle_w \\
&\quad = \int_{-\infty}^{\infty} \frac{1}{|x| + \beta_0} \psi_{\widetilde{j},\widetilde{m}}(x) \psi_{k,n}(x)\, dx \\
&\quad = 2^{-\widetilde{j}} \int_{\widetilde{m}-A}^{\widetilde{m}+A} \frac{1}{|x| + \beta_0 2^{-\widetilde{j}}} \psi_{0,\widetilde{m}}(x) \psi_{k-\widetilde{j},n}(x)\, dx \qquad (2.35)\\
&\quad = 0 + \int_{\widetilde{m}-A}^{\widetilde{m}+A} \left(\frac{1}{|x| + \beta_0 2^{-\widetilde{j}}} - \frac{1}{|\widetilde{m}| + A + \beta_0 2^{-\widetilde{j}}} \right) \psi_{0,\widetilde{m}}(x) \psi_{k-\widetilde{j},n}(x)\, dx
\end{aligned}$$

$$\leq 2^{-\tilde{j}} \max_{x\in[|\widetilde{m}|+A,|\widetilde{m}|-A]} \left(\frac{1}{|x|+\beta_0 2^{-\tilde{j}}} - \frac{1}{|\widetilde{m}|+A+\beta_0 2^{-\tilde{j}}} \right) \left(2^{(\tilde{j}-k)/2} G_1 \right)$$
$$\leq 2^{-\tilde{j}} G_2 \left(\frac{1}{(|\widetilde{m}|+\beta_0 2^{-\tilde{j}})^2 - A^2} \right) \left(2^{(\tilde{j}-k)/2} G_1 \right)$$
$$\leq G_3 2^{-\tilde{j}} 2^{(\tilde{j}-k)/2} \frac{1}{(|\widetilde{m}|+\beta_0 2^{-\tilde{j}})^2}. \tag{2.36}$$
(since $|\widetilde{m}| > A$)

Since the number of n such that $\langle \psi_{\tilde{j},\widetilde{m}}, \psi_{k,n} \rangle_w \neq 0$ is bounded by a constant (recall that $k > \tilde{j}$), thus, for some constant G_4,

$$\sum_n \langle \psi_{\tilde{j},\widetilde{m}}, \psi_{k,n} \rangle_w \leq 2^{-\tilde{j}} G_4 2^{(\tilde{j}-k)/2} \frac{1}{(|\widetilde{m}|+\beta_0 2^{-\tilde{j}})^2}. \tag{2.37}$$

We now consider the case where $k < \tilde{j}$. By combining the fact that $k < \tilde{j}$ and $|\widetilde{m}| > 2A$, it follows that $|n| > A$ if $\langle \psi_{\tilde{j},\widetilde{m}}, \psi_{k,n} \rangle_w \neq 0$. By switching $(\tilde{j}, \widetilde{m})$ and (k, n) in (2.36),

$$\begin{aligned}
\langle \psi_{\tilde{j},\widetilde{m}}, \psi_{k,n} \rangle_w &\leq 2^{-k} G_3 2^{(k-\tilde{j})/2} \frac{1}{(|n|+\beta_0 2^{-k})^2} \\
&\leq 2^{-k} G_3 2^{(k-\tilde{j})/2} \frac{1}{(C_1 2^{\tilde{j}-k}|\widetilde{m}|+\beta_0 2^{-k})^2} \\
&\quad \text{(since } |n| > C_1 2^{\tilde{j}-k}|m| \text{ for some constant } C_1 < 1) \\
&\leq 2^{-\tilde{j}} G_4 2^{(k-\tilde{j})3/2} \frac{1}{(|\widetilde{m}|+\beta_0 2^{-\tilde{j}})^2}
\end{aligned}$$

Since the number of n such that $\langle \psi_{\tilde{j},\widetilde{m}}, \psi_{k,n} \rangle_w \neq 0$ is less than $C_2 2^{\tilde{j}-k}$ for some constant C_2, we have,

$$\sum_n \langle \psi_{\tilde{j},\widetilde{m}}, \psi_{k,n} \rangle_w \leq \left(C_2 2^{\tilde{j}-k} \right) 2^{-\tilde{j}} G_5 2^{(k-\tilde{j})3/2} \frac{1}{(|\widetilde{m}|+\beta_0 2^{-\tilde{j}})^2}$$

$$\leq \quad 2^{-\widetilde{j}} G_5 2^{(k-\widetilde{j})/2} \frac{1}{(|\widetilde{m}| + \beta_0 2^{-\widetilde{j}})^2}. \tag{2.38}$$

Lastly, consider the case where $k = \widetilde{j}$. Using the same approach in the last two cases, we could obtain

$$\sum_{n \neq \widetilde{m}} \langle \psi_{\widetilde{j},\widetilde{m}}, \psi_{k,n} \rangle_w \quad \leq \quad 2^{-\widetilde{j}} G_3 \frac{1}{(|\widetilde{m}| + \beta_0 2^{-\widetilde{j}})^2}. \tag{2.39}$$

Now, combining (2.38), (2.37) and (2.39):

$$\sum_{(k,n) \neq (\widetilde{j},n)} \langle \psi_{\widetilde{j},n}, \psi_{k,n} \rangle_w \quad \leq \quad G 2^{-\widetilde{j}} \frac{1}{(|\widetilde{m}| + \beta_0 2^{-\widetilde{j}})^2}.$$

(b) From (2.35),

$$\begin{aligned}
&\langle \psi_{\widetilde{j},\widetilde{m}}, \psi_{\widetilde{j},\widetilde{m}} \rangle_w \\
&\quad = 2^{-\widetilde{j}} \int_{\widetilde{m}-A}^{\widetilde{m}+A} \frac{1}{|x| + \beta_0 2^{-\widetilde{j}}} \psi_{0,\widetilde{m}}(x) \psi_{0,\widetilde{m}}(x)\, dx \\
&\quad = \frac{2^{-\widetilde{j}}}{m + \beta_0 2^{-j}} + \\
&\qquad 2^{-\widetilde{j}} \int_{\widetilde{m}-A}^{\widetilde{m}+A} \left(\frac{1}{|x| + \beta_0 2^{-\widetilde{j}}} - \frac{1}{m + \beta_0 2^{-j}} \right) \psi_{0,\widetilde{m}}(x) \psi_{0,\widetilde{m}}(x)\, dx \\
&\quad = \frac{2^{-\widetilde{j}}}{m + \beta_0 2^{-j}} + O(m^{-2}).
\end{aligned}$$

Q.E.D.

2.3.3 Weighted Norm in Two Dimensions

In two dimensions, for $I : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ and $J : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, with respect to a weight function, define

$$\langle I, J \rangle_w := \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \frac{I(x,y)J(x,y)}{(w(x,y))^2}\, dx\, dy,$$

and the weighted norm as

$$\|I\|_w := \sqrt{\langle I, I \rangle_w}.$$

Similar to the cases in one dimension, in two dimensions, if

$$w(x,y) := \sqrt{\alpha(x^2+y^2)+\beta},$$

we approximate $\|\Psi_{j,m,n}\|_w$ by $w_{j,n,m}^{-1}$ where

$$w_{j,m,n} := 2^j \sqrt{\alpha} \sqrt{\alpha(m^2+n^2) + 2^{-2j}\frac{\beta}{\alpha}}. \tag{2.40}$$

Note the differences on the exponent between the above and (2.34).

It is easy to see that

Lemma 2.10 *If* $\mathtt{supp}(\psi) \subseteq [-A, A]$, *then for any weight function of the form* $w(x,y) = \sqrt{x^2+y^2+\beta_0}$, *for each* $m > 2A$, $n > 2A$, j, *and* $k \in \{v, h, d\}$,

$$\|\Psi_{j,m,n}^k\|_w^2 = \frac{2^{-2j}}{m^2+n^2+2^{-2j}\beta_0} + O(m^{-4}+n^{-4}).$$

2.3.4 Bit Allocation

Given a standard weight function $w(x, y) := \sqrt{\alpha(x^2 + y^2) + \beta}$, recall that we want to substitute $w_{j,n,m}$ (2.40) into (2.32). Let us examine the value of $w_{j,n,m}$ for different j, n, m and β. Assume that we are working on a N by N pixels image where each pixel take value from $\{0, 1, \ldots, 2^b - 1\}$, and $N = 2^{\ell_0}$.

First, ignore the effect of β and α. For fixed n and m, when j increases, the bin size decreases by a factor of 2. However, as j increases, the maximum possible value of the coefficients increases by a factor of 2, and thus the number of bins remains unchanged. This implies that the number of bits reserved for each coefficient across the scales is constant. For a fixed j, as m and n increase, the bin size increase at a rate of $\sqrt{m^2 + n^2}$. Thus, the number of bits reserved for a coefficient decrease at a rate of $\frac{1}{2}\log_2(m^2 + n^2)$. Now we consider non-zero foveal resolution. The foveal resolution specifies the highest possible resolution near the fovea. Consider the case where $\alpha = 1$ and $\beta \neq 0$. For all scale j, $w_{j,0,0} = \beta$. This induces a uniform resolution near the fovea.

Figure 2.18 shows the values of $j + \log_2 w_{j,0,n}$ as a function of n for different values of j. Given a fix constant $\triangle$, the value $(\log_2 \triangle) + j + \log_2 w_{j,0,n}$ can be interpreted as the number of bits required to encode the coefficient (before the lossless compression) for $\Phi_{j,0,n}$.

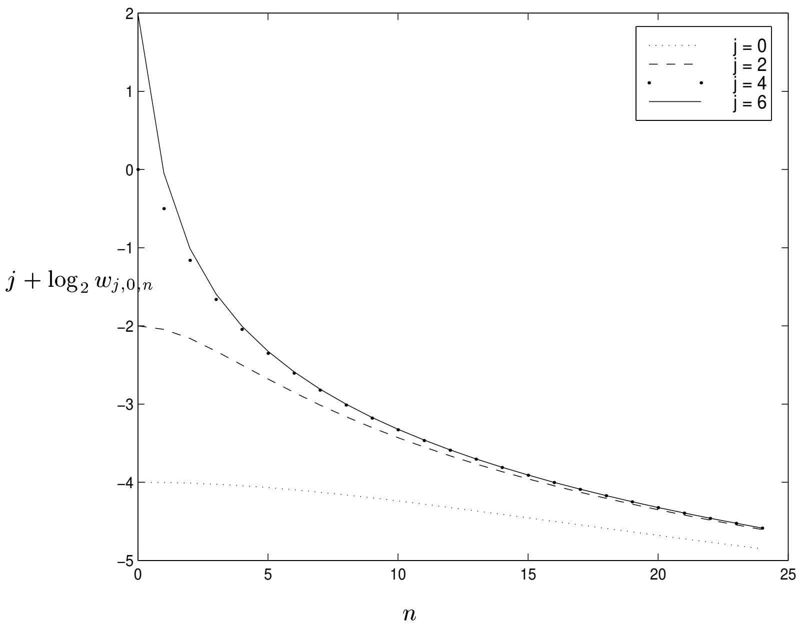


Figure 2.18: The values of $j + \log_2 w_{j,0.n}$ as a function of n. Here, the ratio $\frac{\beta}{\alpha} = 2^4$.

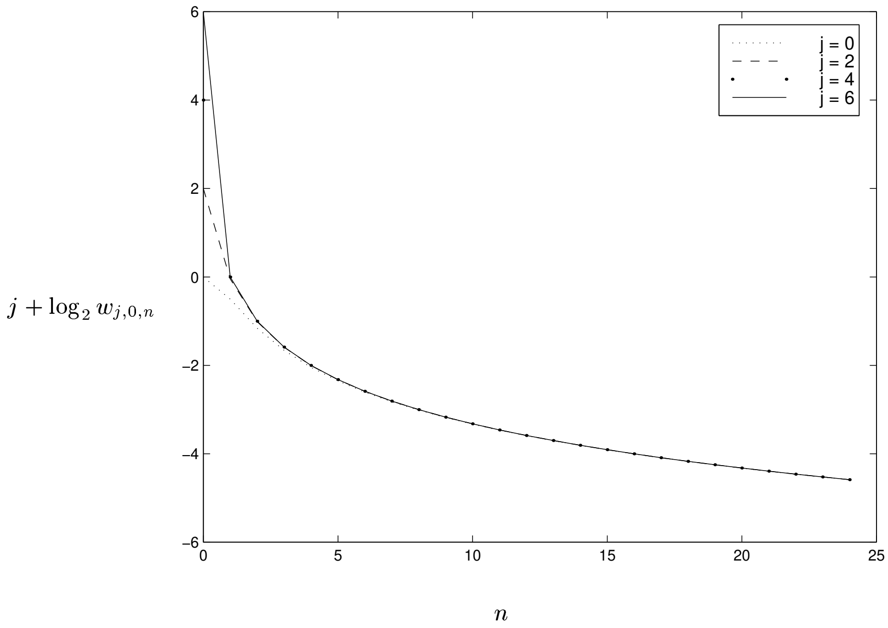


Figure 2.19: The values of $j + \log_2 w_{j,0.n}$ as a function of n. Here, the ratio $\frac{\beta}{\alpha} = 2^0$.

(a)

(b)

Figure 2.20: Two foveated images generated using same number of bits (about 0.21 bits per pixel). (a) Generate by quantizer with $\{w_m\}$ described in this section. The weighted distance from the original is about 76.4. (b) Computed using the 0-1 Mask The weighted distance $\|I - T^{\text{fov}}(I)\|_{w_0}$ is about 86.8. Note that for fairness in comparison, we use a common weighted norm with $w_0(x, y) = 1 + \sqrt{x^2 + y^2}$ for both images. Furthermore, both images are not further compressed with lossless compression.

2.3.5 Transmission Scheme

The transmission scheme is similar to that in section 2.2.7 except that the mask $\{c^k_{j,m,n}\}_{k,j,m,n}$ are replaced by a set of weights $\{w_{j,m,n}\}_{j,m,n}$.

Let

$$d^k_{j,m,n} := \langle I, \Psi^k_{j,m,n} \rangle.$$

For simplicity, assume that the interactions between the client and the server occur at discrete time $t = 1, 2, \ldots$. At time t, the server keeps a multi-fovea weight function w_t and a *weight table* $W = \{w^k_\ell[m,n]\}_{0 \le \ell \le \ell_0, 0 \le m,n < N}$. Initially, $w_0(x) := 2^{\ell_0}$ and each entries of weight table is initialize to zero. In the first interaction, the server sends the coefficient $a := \langle \Phi_{\ell_0,0,0}, I \rangle$ and a constant $\triangle$ to the client. For each subsequent interaction, say the t-interaction, the followings are performed.

1. The client sends a standard weight function $\hat{w}$ (represented by its foveal resolution, rate and gaze point) to the server.

2. Upon receipt of $\hat{w}$, the server performs the following:

   (a) From the lookup table, the server computes $\hat{W} = \{\hat{w}_{j,m,n}]\}$ which is the mask for $\hat{w}$.

   (b) For each k, j, m and n such that

$$\hat{w}_{j,m,n} > w_{j,m,n},$$

the server sends

$$\frac{\lfloor d_j^k[m,n]\hat{w}_{j,m,n}\triangle\rfloor}{\hat{w}_{j,m,n}\triangle} - \frac{\lfloor d_j^k[m,n]\hat{w}_j[m,n]\triangle\rfloor}{\hat{w}_{j,m,n}\triangle}$$

to the client, and resets

$$w_{j,m,n} \leftarrow \hat{w}_{j,m,n}.$$

3. The client reconstructs the foveated image.

The foveated image reconstructed in step 3 is

$$a\ \Phi_{\ell_0,0,0} + \sum_{k,j,m,n} \frac{\lfloor d_j^k[m,n]w_{j,m,n}\triangle\rfloor}{w_{j,m,n}\triangle}\Psi_{j,m,n}^k.$$

If we further simplify the above by thresholding the weight, and fix each $\triangle = 1$, then step 2(b) becomes:

2(b) For each k, j, m and n such that

$$\hat{w}_{j,m,n} - w_{j,m,n} = 1,$$

the server sends

$$\frac{\lfloor d_j^k[m,n]\hat{w}_{j,m,n}\rfloor}{\hat{w}_{j,m,n}}$$

to the client and resets

$$w_{j,m,n} \leftarrow 1.$$

2.4 Conclusion

In this chapter, we describe two approaches to producing a foveated image from a uniform resolution image. In the first approach we give a method that, given an image I, constructs an image I_0 which approximates $T^{\text{fov}}(I)$, using $\|I_0 - T^{\text{fov}}(I)\|_2^2$ as a measure. The construction is achieved by applying a mask $\{c_j^k[m,n]\}_{k,j,mn}$ on the wavelet coefficients $\{d_j^k[m,n]\}_{k,j,m,n}$ of I,

$$I_0 = \sum_{k,j,m,n} \frac{\lfloor c_j^k[m,n] d_j^k[m,n] \triangle \rfloor}{\triangle} \Psi_{j,m,n}^k, \tag{2.41}$$

where $\triangle$ is some constant. The formulation of the foveation operator T^{fov} is motivated by the logmap transformation, and it intends to capture the situation where convolution is performed on the visual cortex plane (the domain after logmap transformation).

In the second approach, we tries to find a $Q(\cdot)$ that minimizes the expected (by treating I as a random variable) weighted distance $\|Q(I) - I\|_w^2$, given a fixed average number of bits for $Q(I)$. Since we usually do not know the distribution of I, it is impossible to design such a method $Q(\cdot)$. Assuming that wavelet transform decorrelates I, we design an approximation using a set of weights $\{w_j[m,n]\}_{j,m,n}$,

$$I_0' = \sum_{k,j,m,n} \frac{\lfloor w_j[m,n] d_j^k[m,n] \triangle \rfloor}{w_j[m,n] \triangle} \Psi_{j,m,n}^k, \tag{2.42}$$

where $\triangle$ is some constant. Again, the definition of weighted distance is motivated by the logmap transformation. The weighted distance amounts to the usual mean square error in the visual cortex plane.

Both (2.41) and (2.42) could be simplified by thresholding the weights and the mask respectively. As mentioned in the introduction, a well-known method of producing a foveated image is by a cut-and-paste from an hierarchical representation of the image. This method coincides with the extreme form of the simplification just mentioned, where a mask/weights is thresholded or rounded to $\{0, 1\}$.

Chapter 3

On-line Scheduling with Level of Service

In this chapter, we study an abstract on-line scheduling problem motivated by the thinwire model. If we treat what the viewer wants to view as a request for data, then, throughout the visualization process, a continuous stream of requests will be generated. Because of the thinwire assumption, more requests will be generated than the bandwidth could support. Thus, it is necessary to select the more important requests which are to be sent to the server. An interesting property of this problem which we exploit is that: a request could be scaled down and served at a lower level of service.

3.1 Introduction

Scheduling is an important issue in resource management. Suppose I is an instance of a scheduling problem, usually a sequence of jobs or requests. The goal is to find an optimal *schedule* which achieves minimum cost or, equivalently, maximum merit.

We are interested in on-line scheduling. In this setting, the problem instance is released gradually, a single job or request at a time, while the scheduler upon receipt of each new job/request, is required to make decision and output a part of the schedule. Performance of an on-line scheduler could be measured by its competitiveness, introduced by Sleator and Tarjan [37]. Suppose S is a scheduler and $S(I)$ is the schedule produced by S on I, then S is c-competitive if for any instance I,

$$merit(\mathtt{opt}(I)) \leq c \ . \ merit(S(I)) + b,$$

where b is some constant and $\mathtt{opt}(I)$ is the optimal off-line schedule. The *competitive ratio* of a scheduler S is defined by

$$C := \sup_{I} \frac{merit(\mathtt{opt}(I))}{merit(S(I))}.$$

We say that a problem is *no better than* c-competitive if there does not exist a scheduler that is c_1-competitive, for any $c_1 < c$. A classic example is the paging problem where the main memory can only holds at most k pages and the goal is to minimize the number of page faults. Sleator and Tarjan

[37] show that the LIFO (last in first out) and LRU (least recently used) strategies are k-competitive and the problem is no better than k-competitive. The paging problem is a special case of the k-server problem introduced by Manasse, MacGeoch, and Sleator [22].

There are a few variants on the computational models. One may consider randomized scheduler. A randomized scheduler S, as opposed to the deterministic scheduler, is allowed to toss coins randomly. Given an instance I, let $S(I)$ to be the random variable of the schedule output by S. We say that S is c-competitive if for any I,

$$merit(\texttt{opt}(I)) \leq c \ . \ E[merit(S(I))] + b,$$

where b is a constant. Similarly, we say that S has competitive ratio C if

$$C := \sup_I \frac{merit(\texttt{opt}(I))}{E[merit(S(I))]}.$$

Average case analysis is another major branch in the analysis of on-line problems. Here, the instance is drawn from a known distribution, and a scheduler S is measured by its expected performance. If we write I to be the random variable for the instance, then the expected performance is

$$E[merit(S(I))].$$

While the k-server and the paging problem focus on the fundamental issue of how to "distribute" jobs among k servers, the case where there is only one server is also interesting. An example is the on-line intervals packing problem where the instance consists of open intervals and a schedule is a subset of non-overlapping intervals. Lipton and Tomkins [18] study a variant where the input intervals are sorted by their left endpoints, and the intervals are restricted to two types: intervals with unit size and intervals with size k. The goal is to cover the real line as much as possible. They give a randomized scheduler that is 2-competitive.

The on-line scheduling problem studied here could be treated as a variant of on-line interval packing problem. We describe this scheduling problem in a user-manager setting where the user issues requests and the manager schedules the requests[1]. We give a class `ArbFit` of managers. Two members in this class `Greedy` and `EndFit` have competitive ratio 2. `Greedy` and `EndFit` have interesting interpretation and in a certain sense, `Greedy` is always better than `EndFit`. We show that every manager in `ArbFit` is 3-competitive; on the other hand, there is a manager that has competitive ratio 3. A natural extension of `Greedy` and `EndFit` is a randomized algorithm `gdyXend` which randomly changes mode between `Greedy` and `EndFit`. We do not know the expect performance of `gdyXend`, however, since `gdyXend` adheres to the rule of `ArbFit`, it is always 3-competitive.

We show that this problem is no better than $2(2-\sqrt{2}) \approx 1.1716$-competitive by describing an adversary.

[1] We avoid the term "client-server" since they have specific meaning in the thinwire model.

3.2 Problem formulation and definition

Basic formulation In this user-manager setting, the user issues requests and the manager tries to serve the requests within its limited resource. An *instance* is a sequence of *requests*. Each request q has four parameters, (s, t, u, v), where s, the *start time*, is the time the request is issued; t, the *deadline*, is the time the request has to be completed; u is the size of the request and v is the *weight* of the request. Write $st(q), dl(q), sz(q), wt(q)$ for its start time, deadline, size and weight respectively. The requests are issued one at a time and if a request p is issued before q, then $st(p) \leq st(q)$. The manager could serve any issued request but at any time, only a request could be served. A request q is completely served if it has been served for a total of $sz(q)$ time. A request can not be served if it's deadline has already passed or it has been completely served. The service is *preemptive without penalty*, that is, the manager could switch from one request to another request at anytime and he could also switch back to an uncompleted service. However, the number of such switchings must be bounded by some polynomial $p(n)$ where n is the total number of requests. For each completely served request, the manager gains $sz(q)wt(q)$ merit points. Unlike usual scheduling problems, a partially served request contributes a proportional amount of merit points: If a request q is served for a total of x time, then the manager gains $xwt(q)$ merit points. The goal of the manager is to maximize the total merit points gained.

Call a request q *alive* at time t_0 if $st(q) \le t_0 \le dl(q)$. At time t_0, requests could have been partially or completely served within $(-\infty, t_0]$. Those alive requests that are not yet completely served are said to be *pending* at time t_0.

Schedule Given an instance $I = \{q_1, q_2, \ldots, q_n\}$, define the *schedule* H to be a piecewise constant function from $\mathbb{R}$ to $\{q_1, q_2, \ldots, q_n\} \cup \{\emptyset\}$, satisfying $|H^{-1}(q_k)| \le sz(q_k)$ and $H^{-1}(q_k) \subseteq (st(q_k), dl(q_k)]$ for all $k = 1, 2, \ldots, n$. The schedule describes how requests are served through the time. If a request q_k is served at t_0, then $H(t_0) := q_k$. If no request is served, then $H(t_0)$ is defined to be $\emptyset$. Call an half-opened half-closed interval $(t_1, t_2] \subset \mathbb{R}$ a *time-slot.* We further add a restriction on the definition of a schedule: for any request q_k, $H^{-1}(q_k)$ must be either the empty set or the union of a finite number of time-slots. Call a point of discontinuity in the schedule a *switch-point.*

The *merit* of a schedule $merit(H)$ is defined as follows:

$$\sum_{j=1}^{n} wt(q_j)|H^{-1}(q_j)|.$$

If S is a manager and I an instance, then we write $S(I)$ for the schedule obtained by applying S on I. If S is random, then $S(I)$ is a random schedule.

An *off-line optimal* schedule for an instance I is a schedule that maximizes *merit.* Denote by $\texttt{opt}(I)$ an off-line optimal schedule for I.

Pending request Call a request q *alive* at time t_0 if $st(q) \leq t_0 \leq dl(q)$. With respect to a manager, at time t_0, some requests could have been partially or completely served within $(-\infty, t_0]$. Call those alive requests that are not yet completely served the *pending* requests at time t_0.

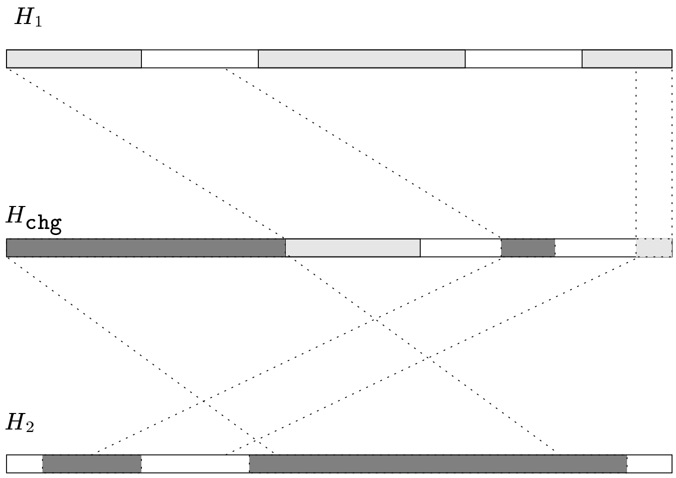


Figure 3.1: The charging scheme.

Ordering of requests. Note that start times of requests are non-unique. However, since the requests are issued one at a time, there is a natural ordering of the requests.

The managers given in this chapter make decisions by giving priority to heavier weighted requests. In the case where $wt(p) = wt(q)$, we resolve the tie by treating p "heavier" than q if and only if p arrives before q. In the rest of this chapter, when we use the phrase "p is heavier than q", we are referring to this total ordering.

Charging Scheme. We often need to argue that the merit of a schedule H is smaller than the total merit of other schedules, say H_1 and H_2. Our approach is to *charge* a portion of H to H_1 and the remaining to H_2. Intuitively, the charging process can be viewed as first cutting H_1 and H_2 into pieces and then joining some of the pieces to form another piecewise-constant function $H_{\texttt{chg}}$. Each piece, after cutting, could be translated before joining. As it may be the case that $|H_{\texttt{chg}}^{-1}(q)| > sz(q)$ for some request q, thus $H_{\texttt{chg}}$ is not a schedule. The cut-and-paste must be done in a way that for all t, $wt(H_{\texttt{chg}}(t)) \geq wt(H(t))$. Therefore,

$$merit(H) \leq merit(H_{\texttt{chg}}) \leq merit(H_1) + merit(H_2).$$

In particular, if $H_1 = H_2$, then we have

$$merit(H) \leq 2 \cdot merit(H_1).$$

Figure 3.1 illustrates an example of $H_{\texttt{chg}}$, H_1 and H_2. Formally, a charging scheme of H to H_1 and H_2 can be described by two functions: a piecewise constant function $C_{\mathrm{x}} : \mathbb{R} \to \{1, 2\}$ and another piecewise-linear function $C_{\mathrm{y}} : \mathbb{R} \to \mathbb{R}$. Each linear piece $(a_1, a_2]$ of the function C_{y} can be written as

$$C_{\mathrm{y}}(t) = t + \alpha \quad \text{for } t \in (a_1, a_2],$$

where α is a constant. We further require that for each $i \in 1, 2$, C_{y} is one-one restricted to the domain $C_{\mathrm{x}}^{-1}(i)$. From C_{x} and C_{y}, define $H_{\mathtt{chg}}$:

$$H_{\mathtt{chg}}(t) = \begin{cases} H_1(C_{\mathrm{y}}(t)), & \text{if } C_{\mathrm{x}}(t) = 1, \\ H_2(C_{\mathrm{y}}(t)), & \text{otherwise.} \end{cases}$$

We often use the following phrase: charge $(s', e']$ from H to H_i at $(s, e]$. This means that $C_{\mathrm{x}}(t) := i$ and $C_{\mathrm{y}}(t) := t - s' + s$ for $t \in H^{-1}(q)$. Note that a precondition is $e - s = e' - s'$. (Figure 3.2). Equivalently, if $H^{-1}(q)$ is the interval $(s', e']$, we may also say that we charge a request q from H to H_i at $(s, e]$.

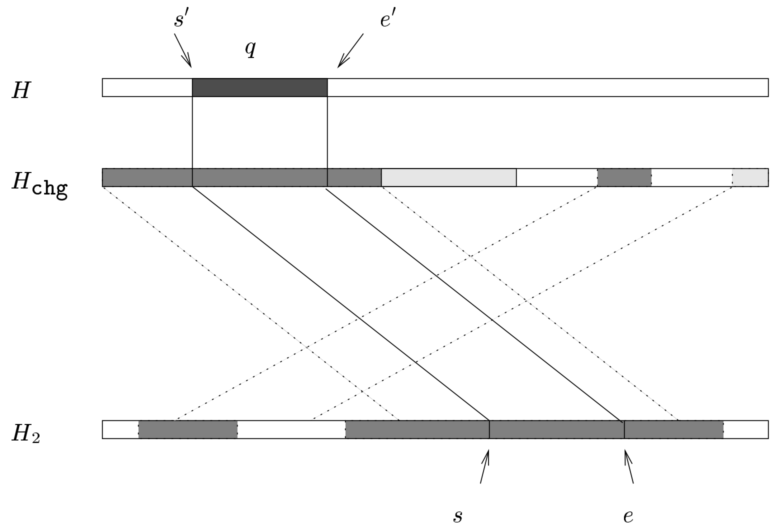


Figure 3.2: Charging the request q in H to H_2 at $(s, e]$.

The purpose of this formal definition is to clarify what a charging scheme is. In the rest of this chapter, we will not describe $H_{\mathtt{chg}}$, C_{x} and C_{y} explicitly.

3.3 Examples of Managers

In this section, we describe two managers `Greedy` and `EndFit`. At any moment, `Greedy` always serves the current heaviest pending request, whereas `EndFit` always serves according to the off-line optimal schedule of the current pending requests. In a certain sense, the two managers are at two extreme ends of a spectrum. Both managers fall into a general class of managers, `ArbFit`, which we are going to describe now.

A manager in this class performs the following upon arrival of a request q.

1. Stops the current service (that is, preempt).

2. Computes a 'plan', which is a schedule for the pending requests. We call it a 'plan' because the manager may not carry out the schedule as planned due to the arrival of new requests later. The plan is computed by considering the pending requests one by one, starting from the heaviest request down to the lightest request. (Here, we are referring to the total ordering described in section 3.2). Let p be the request being considered. Its allocation is subjected to the following restriction:

   (∗) The allocation to p must be maximized. (For example, if it is possible to completely allocate p, the whole of p must be allocated). However, there is no restriction on where to allocate p. Allocated time-slots are not available for requests which are subsequently considered.

(We will give a formal description of $2(*)$ later in this section).

3. Carries out the plan until a new request arrives.

Let the plan computed after step 2 be $Plan^{+}(S, I, q)$. Let $Plan^{-}(S, I, q)$ be the original plan just before step 2 is executed. Note that $Plan^{-}(S, I, q)$ is actually $Plan^{+}(S, I, q')$, where q' is the request which arrives just before q.

A formal way to describe the restriction $2(*)$ on $Plan^{+}(S, I, q)$ is as follow: for any request p, let I_p be the union of time-slots that are contained in the life span of p, and are allocated with requests not lighter than p, that is,

$$I_p := (st(p), dl(p)] - \left(\bigcup_{p' \text{ is heavier than } p} (Plan^{+}(S, I, q))^{-1}(p') \right).$$

Then, we must have

$$\left|(Plan^{+}(S, I, q))^{-1}(p)\right| = \min\left\{sz(p), |I_p|\right\}.$$

Different managers in `ArbFit` differ in how they allocate the request being considered in step $2(*)$. `Greedy` is the manager who allocates the request being considered in the earliest possible time-slot. This agrees with the previous description of `Greedy`.

Another manager may allocate the request being considered in the latest possible time-slot. Interestingly, at any moment, say at time t_0, its plan is an off-line optimal schedule of the pending requests at t_0. This description coincides with the manager `EndFit` described in the beginning of this section. This seem to be counter-intuitive as the original description of `EndFit` seems to suggest that it is not a good strategy whereas the alternative description seems to suggest the other way.

We now show that these two descriptions agrees with each other. Let P be the set of pending requests upon arrival of q_0. For each $q \in P$, let $\hat{q} := (t_0, dl(q), s, wt(q))$ where s is the remaining size of q yet to be served, and let $\hat{P}$ be the set of these newly defined requests. Then we have the following lemma:

Lemma 3.1 *The* $Plan^+(\texttt{EndFit}, I, q_0)$ *is an optimal schedule for* $\hat{P}$.

Proof. Let H be an off-line optimal schedule of $\hat{P}$. Consider the heaviest request $\hat{p}$. Let the time-slots where $\hat{p}$ is allocated be $(s_1, t_1], (s_2, t_2], \ldots, (s_k, t_k]$. Let $s := |H^{-1}(\hat{p})| = \sum_{i=1}^{k}(t_i - s_i)$. Since $\hat{p}$ is the heaviest request, it is easy to verify that $s = \min\{sz(\hat{p}), dl(\hat{p}) - st(\hat{p})\}$, which implies that $\hat{p}$ will be completely served as long as its life span not smaller than its size. We want to perform a swap so that $\hat{p}$ is allocated in the time-slot $(dl(\hat{p}) - s, dl(\hat{p})]$. This can be achieved by swapping the portion of $\hat{p}$ allocated in $(s_i, t_i]$ with requests allocated in $(u_i, u_i + (t_i - s_i)]$ where $u_i = dl(\hat{p})) - s + \sum_{j=1}^{i-1}(t_i - s_i)$, for $i = 2, \ldots, k$, and $u_1 := dl(\hat{p})) - s$. These swaps are possible because all requests have a common start-time $st(\hat{p})$. The new time-slot allocated for $\hat{p}$

is exactly same as that for p in $Plan^{+}(\texttt{EndFit}, I, q_0)$.

We can now repeat this process for the next heaviest request. The only difference is that this allocation may be spread out over more than one interval. The final plan is clearly $Plan^{+}(\texttt{EndFit}, I, q_0)$. **Q.E.D.**

To understand how the manager `EndFit` works, it is helpful to visualize the relationship between $Plan^{-}(\texttt{EndFit}, I, q)$ and $Plan^{+}(\texttt{EndFit}, I, q)$ upon arrival of a new request q. To obtain $Plan^{+}(\texttt{EndFit}, I, q)$ from $Plan^{-}(\texttt{EndFit}, I, q)$, first find the location to "insert" the newly arrives request q; then "squeeze" q in by "pushing" the original allocated requests leftward (to an earlier time-slot). In so doing, some requests may be "pushed-out" from the plan. The plans produced by `Greedy` can be viewed in the similar way except that the "push" is in the opposite direction.

3.3.1 Competitive ratio of `Greedy`

First, we state a lemma whose purpose is to tackle a technical complication. Consider a schedules H_1. At time t_0, where t_0 is a switch point in H_1, it is possible that a request q has been partially, but not completely served, that is, $\left|H_1^{-1}(q) \cap (-\infty, t_0]\right| \notin \{0, sz(q)\}$. This causes a technical complication, which is not difficult but tedious to handle. It is even more tedious in the case where we want to compare two schedules H_1, H_2, which have different sets of switch points.

We say that an instance I is *intact* in a schedule H, if each request q, $H^{-1}(q)$ is connected, and either $|H^{-1}(q)| = 0$ or $sz(q)$. (By definition, an empty set is connected). We say that a request q is *refined* to $q_1, \ldots, q_k$ for some $1 \leq k$ (or $q_1, \ldots, q_k$ forms a refinement of q), if $q_i = (st(q), dl(q), s_i, wt(q))$, for each $i = 1, \ldots, k$, where

$$\sum_{i=1}^{k} s_i = sz(q),$$

and $s_i \geq 0$ for each $i = 1, \ldots, k$. (Figure 3.3). An instance I_0 is a refinement of I if the requests in I_0 are refinements of requests in I. Since our scheduling problem is preemptive, q is essentially the same as the combined of q_1 and q_2.

It is easy to see that:

Lemma 3.2 *Given an instance I, if I_0 is a refinement of I, then*

$$\begin{aligned} merit(\mathtt{opt}(I)) &= merit(\mathtt{opt}(I_0)), \\ merit(\mathtt{Greedy}(I)) &= merit(\mathtt{Greedy}(I_0)), \quad \textit{and} \\ merit(\mathtt{EndFit}(I)) &= merit(\mathtt{EndFit}(I_0)). \end{aligned}$$

Furthermore, there is an refinement I_1 which is intact in $\mathtt{opt}(I_1)$, $\mathtt{EndFit}(I_1)$, *and* $\mathtt{Greedy}(I_1)$.

Q.E.D.

Lemma 3.3 *The competitive ratio of* `Greedy` *is at least 2.*

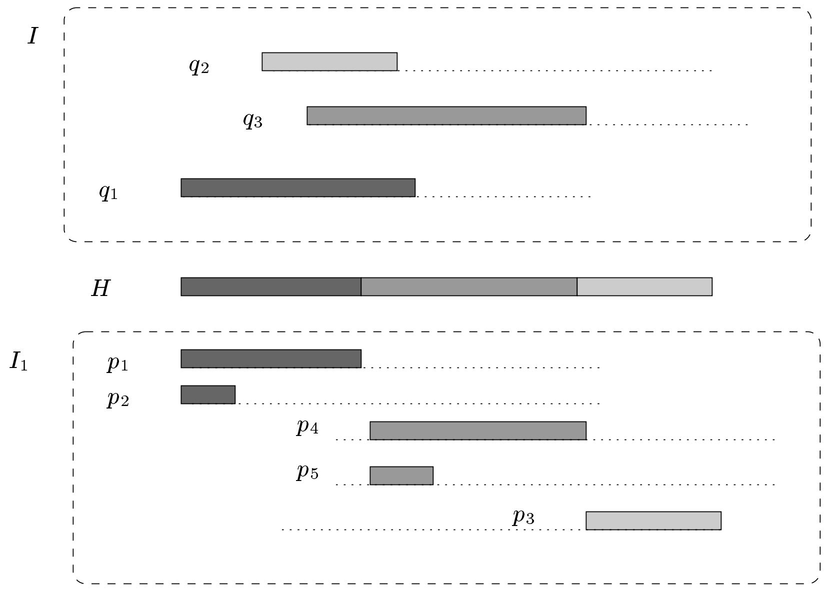


Figure 3.3: Instance I_1 is a refinement of I that is intact in H.

Proof. Take an instance of two requests $q_1 := (0, 2, 1, 1)$ and $q_2 := (0, 1, 1, 1)$ where q_1 arrives before q_2 (Figure 3.4). **Q.E.D.**

Theorem 3.4 *For any instance I,*

$$2 \cdot \ merit(\texttt{Greedy}(I)) \geq \ merit(\texttt{opt}(I)).$$

Proof.

Given an instance I, let $H_{\texttt{gdy}}$ be the schedule $\texttt{Greedy}(I)$ and $H_{\texttt{opt}}$ be the off-line optimal schedule $\texttt{opt}(I)$. By Lemma 3.2, we can assume that I is intact in both $H_{\texttt{gdy}}$ and $H_{\texttt{opt}}$.

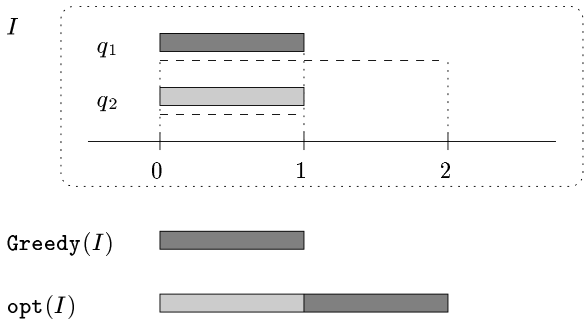


Figure 3.4: This example demonstrates that the competitive ratio of `Greedy` is at least 2.

Let H_0 be an identical copy of $H_{\texttt{gdy}}$. We want to charge requests served in $H_{\texttt{opt}}$ to H_0 and $H_{\texttt{gdy}}$. Let $\{t_1, t_2, \ldots, t_m\}$ be the distinct switch-points in $H_{\texttt{opt}}$, where $t_i < t_j$ if and only if $i < j$.

We describe the charging scheme $\widetilde{H}$ by construction.

For each t_i, starting from $i = 1$ to $m - 1$, consider the time-slot $(t_i, t_{i+1}]$. Let $q_{\texttt{opt}} := H_{\texttt{opt}}(t_{i+1})$, and let $q_{\texttt{gdy}}$ be the lightest request served during $(t_i, t_{t+1}]$ by `Greedy`.

There are two cases:

1. If the request $q_{\texttt{gdy}}$ is not lighter than $q_{\texttt{opt}}$, charge $q_{\texttt{opt}}$ from $H_{\texttt{opt}}$ to $H_{\texttt{gdy}}$ at $(t_i, t_{i+1}]$.

2. Otherwise, Charge $q_{\texttt{opt}}$ from $H_{\texttt{opt}}$ to H_0 at $H_0^{-1}(q_{\texttt{opt}})$.

We have to show that in the second case, $|H_0^{-1}(q_{\texttt{opt}})| \geq |(t_i, t_{i+1}]|$. In the first place, why is the weight of $q_{\texttt{gdy}}$ lighter? The request $q_{\texttt{gdy}}$ is chosen by

Greedy because it is the heaviest request among the pending requests. This implies that $q_{\mathtt{opt}}$ is not a pending request, although it is alive at time t_i. So $q_{\mathtt{opt}}$ must has been completely served by Greedy.

Q.E.D.

3.3.2 Competitive Ratio of EndFit

In this section, we show that the competitive ratio of EndFit is 2.

Lemma 3.5 *The competitive ratio of* EndFit *is at least 2*

Proof. Take two requests $q_1 := (0, 2, 1, 1)$ and $q_2 := (1, 2, 1, 1)$ (Figure 3.5).

Q.E.D.

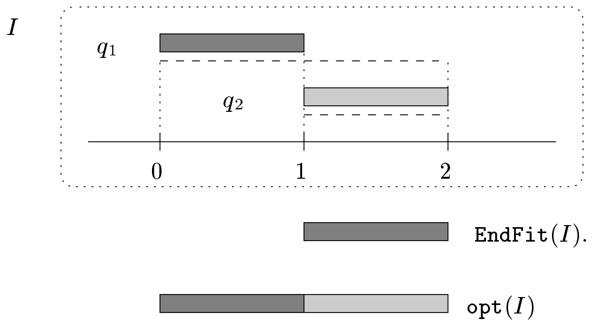


Figure 3.5: This example demonstrates that the competitive ratio of EndFit is at least 2.

To prove the upper bound, given an instance I, we modify I to obtain an instance $\widetilde{I}$ such that $merit(\mathtt{opt}(\widetilde{I})) = merit(\mathtt{opt}(I))$ and yet $merit(\mathtt{EndFit}(\widetilde{I})) \leq$

$merit(\texttt{EndFit}(I))$. The new instance $\widetilde{I}$ is of a restricted form which we exploit in proving the claimed bound.

A request $\widetilde{q}$ is a *trimmed* request of q if $st(\widetilde{q}) \geq st(q)$, $dl(\widetilde{q}) = dl(q)$, $wt(\widetilde{q}) = wt(\widetilde{q})$ and $sz(\widetilde{q}) \leq sz(q)$. An instance $\widetilde{I}$ is a trimmed instance of I if there is a one-one (not necessary onto) mapping from $\widetilde{I}$ to I such that any $\widetilde{q}$ in $\widetilde{I}$ is a trimmed request of its corresponding request in I. Clearly, $merit(\texttt{opt}(I)) \geq merit(\texttt{opt}(\widetilde{I}))$, but what about the relationship between $\texttt{EndFit}(I)$ and $\texttt{EndFit}(\widetilde{I})$?

Lemma 3.6 *(Key lemma). If $\widetilde{I}$ is a trimmed instance of I, then*

$$merit(\texttt{EndFit}(\widetilde{I})) \leq merit(\texttt{EndFit}(I)).$$

Proof.

Before proving the lemma, let us view the plans of `EndFit` from another direction and make two remarks. Although these two remarks seem "technical", they are important steps in this proof because they establish a connection between the off-line and on-line setting of `EndFit`.

On the arrival of a request q, let P be the set of pending requests. For each $p \in P$, let $\hat{p}$ be the trimmed request $\hat{p} := (st(q), dl(p), s, wt(p))$ where s is the remaining size of p yet to be served. Let $\hat{P}$ be the collection of such $\hat{p}$. Let H to be a schedule of $\hat{P}$ defined as follow:

For each $\hat{p} \in \hat{P}$, $H(t)$ is not lighter than $\hat{p}$ for any $t \in (t_0, dl(\hat{p})]$, where

$$t_0 := \begin{cases} st(\hat{p}), & \text{if } |H^{-1}(\hat{p})| < sz(\hat{p}), \\ \inf\ H^{-1}(\hat{p}), & \text{otherwise.} \end{cases}$$

Note that there is a unique schedule that satisfies the above, thus the definition is valid. It is easy to see that the plan $Plan^{+}(\texttt{EndFit}, I, q) = H$.

The above definition is made on the set of pending requests $\hat{P}$. We want to relax this by replace it by the original requests. Let I_0 be set of requests that contains q and all requests that arrive before q. Let H_0 be the schedule that satisfies the following:

For any $p \in I_0$, $H_0(t)$ is not lighter than p for any $t \in (t_0, dl(p)]$, where

$$t_0 := \begin{cases} st(p), & \text{if } \left|{H_0}^{-1}(p)\right| < sz(p), \\ \inf\ {H_0}^{-1}(p), & \text{otherwise.} \end{cases}$$

We can view H_0 as the schedule obtained by applying `EndFit` on I_0 in an off-line setting. Let `EndFitOff` be such a manager, and thus $\texttt{EndFitOff}(I_0) = H$. Clearly, H_0 is not the same as the original H, since all $\hat{p} \in \hat{P}$ have common start time while this may not true for requests in I_0. However, we have:

Remark (a). *For any $t \in (t_0, \infty)$, $H(t)$ is equivalent to $H_0(t)$ in the sense that $H(t) = \hat{p}$ if and only if $H_0(t) = p$.*

Proof of Remark(a)

This can be shown by contradiction. We may assume $\hat{P}$ and I_0 are intact in H and H_0 respectively. Suppose on the contrary, then there is a maximum $t_{\max}$ such that $H_0(t_{\max})$ is not equivalence to $H(t_{\max})$. Let $\hat{p}_{\max} := H(t_{\max})$ and consider the corresponding $p_{\max}$ in H_0. There are two cases: either $p_{\max}$ is heavier or lighter than $H_0(t_{\max})$. Let us consider the first case. It is not possible that H_0 allocates $p_{\max}$ at a time later than $t_{\max}$ since this will contradict either the definition of $t_{\max}$ or the intactness of P. It is also not possible that H_0 allocates $p_{\max}$ at an earlier time or does not even serve it, since this will contradict the definition of H_0. We can similarly show that the second case leads to contradiction. **Q.E.D. (of Remark (a))**

Remark (a) is useful because it establishes an equivalent between the plan H_0 with a schedule H which is computed by an off line manager.

It is easy to see that:

Remark (b). *Suppose I is any set of requests, then, for any $p \in I$, and for all t, $\texttt{EndFitOff}(I)(t)$ is not lighter than $\texttt{EndFitOff}(I \backslash \{p\})(t)$.*

Given an instance I, suppose $G_0 := \texttt{EndFitOff}(I)$ and $G_1 := \texttt{EndFitOff}(I \backslash \{p\})$, where p is any request, write

$$G_0 \stackrel{\text{del } p}{\Longrightarrow} G_1$$

to express the relationship that G_1 is obtained from G_0 by deleting p from the instance. In the other direction, we use the notation:

$$G_1 \overset{\text{ins } p}{\Longrightarrow} G_0.$$

By definition, $G_0 \overset{\text{del } p}{\Longrightarrow} G_1 \overset{\text{ins } p}{\Longrightarrow} G_0$.

Now, back to the proof of the lemma. Given an instance I and a trimmed instance $\widetilde{I}$, it is sufficient to consider the case where $\widetilde{I}$ differs from I by only one request: a request $\widetilde{q}$ in $\widetilde{I}$ which is the trimmed request of q in I, and $sz(\widetilde{q}) \in \{0, sz(q)\}$.

Let the requests in I, listed in their order of arrival, be

$$p_0, \ldots, p_k, q, q_1, \ldots q_m,$$

and the corresponding trimmed instance $\widetilde{I}$ be

$$\widetilde{p}_0, \ldots, \widetilde{p}_k, \widetilde{q}_1, \ldots, \widetilde{q}_\ell, \widetilde{q}, \widetilde{q}_{\ell+1}, \ldots \widetilde{q}_m.$$

Let $H_0 := \texttt{EndFitOff}(\{p_0, \ldots, p_k\})$ and $\widetilde{H}_0 := \texttt{EndFitOff}(\{\widetilde{p}_0, \ldots, \widetilde{p}_k\})$. Define $H_1, H_2, \ldots$ and $\widetilde{H}_0, \widetilde{H}_1, \ldots$ as follows:

$$H_0 \overset{\text{ins } q}{\Longrightarrow} H_1 \overset{\text{ins } q_1}{\Longrightarrow} H_2 \ \ldots \overset{\text{ins } q_\ell}{\Longrightarrow} H_\ell \overset{\text{ins } q_{\ell+1}}{\Longrightarrow} \ldots$$

$$\widetilde{H}_0 \overset{\text{ins } \widetilde{q}_1}{\Longrightarrow} \widetilde{H}_1 \ \ldots \overset{\text{ins } \widetilde{q}_\ell}{\Longrightarrow} \widetilde{H}_{\ell-1} \overset{\text{ins } \widetilde{q}}{\Longrightarrow} \widetilde{H}_\ell \overset{\text{ins } \widetilde{q}_{\ell+1}}{\Longrightarrow} \ldots$$

Note that H_0 and $\widetilde{H}_0$ are essentially the same, since each $\widetilde{p}_i$ is a exact copy of p_i, for $i = 0, 1, \ldots, k$. We write $H_0 \longleftrightarrow \widetilde{H}_0$ to refer to the equivalence just mentioned. If $H_1 \overset{\text{del } q}{\Longrightarrow} H_0$ and $H_0 \longleftrightarrow \widetilde{H}_0$, we write

$$H_1 \overset{\text{del } q}{\longrightarrow} \widetilde{H}_0.$$

Inductively, we have the following:

$$
\begin{array}{ccccccccccccc}
H_0 & \stackrel{\text{ins } q}{\Longrightarrow} & H_1 & \stackrel{\text{ins } q_1}{\Longrightarrow} & H_2 & \ldots & \stackrel{\text{ins } q_\ell}{\Longrightarrow} & H_\ell & \stackrel{\text{ins } q_{\ell+1}}{\Longrightarrow} & \ldots & & & \\
& \nwarrow\!\!\searrow & \downarrow \text{del } q & & \downarrow \text{del } q & & & \downarrow \text{del } q & & & & & \\
& & \widetilde{H}_0 & \stackrel{\text{ins } \widetilde{q}_1}{\Longrightarrow} & \widetilde{H}_1 & \ldots & \stackrel{\text{ins } \widetilde{q}_\ell}{\Longrightarrow} & \widetilde{H}_{\ell-1} & \stackrel{\text{ins } \widetilde{q}}{\Longrightarrow} & \widetilde{H}_\ell & \stackrel{\text{ins } \widetilde{q}_{\ell+1}}{\Longrightarrow} & \ldots &
\end{array}
$$

Note that, by Remark (b), we have

$$wt(H_1(I)(t)) \geq wt(\widetilde{H}_0(\widetilde{I})(t)), \quad \text{for } t \in (st(q), st(q_1)]. \tag{3.1}$$

So far we have only concerned about `EndFitOff`. We now apply Remark(a) to bridge the above diagram to the on-line setting. For any $t \in (st(q), st(q_1)]$, by Remark (a), since no request arrives between q and q_1,

$$
\begin{aligned}
\texttt{Plan}^+(\texttt{EndFit}, I, q)(t) &= H_1(t), \text{ and} \\
\texttt{EndFit}(\widetilde{I})(t) &= \widetilde{H}_0(t).
\end{aligned}
$$

Combining with (3.1), we have

$$wt(\texttt{EndFit}(I)(t)) \geq wt(\texttt{EndFit}(\widetilde{I})(t)), \quad \text{for } t \in (st(q), st(q_1)].$$

This inequality can be extended to the following:

$$wt(\texttt{EndFit}(I)(t)) \geq wt(\texttt{EndFit}(\widetilde{I})(t)), \quad \text{for } t \in (st(q), st(\widetilde{q})]. \tag{3.2}$$

Now, there are two cases. If $sz(\widetilde{q}) = 0$, then $\widetilde{H}_\ell = \widetilde{H}_{\ell-1}$ and by extending the previous argument,

$$wt(\texttt{EndFit}(I)(t)) \geq wt(\texttt{EndFit}(\widetilde{I})(t)), \quad \text{for } t \in (st(\widetilde{q}), \infty). \tag{3.3}$$

On the other hand, if $sz(\widetilde{q}) = sz(q)$, then by inserting $\widetilde{q}$, we have $\widetilde{H}_\ell \longleftrightarrow H_\ell$. Therefore,

$$wt(\texttt{EndFit}(I)(t)) = wt(\texttt{EndFit}(\widetilde{I})(t)), \quad \text{for } t \in (st(\widetilde{q}), \infty). \tag{3.4}$$

Since the instance I and $\widetilde{I}$ differ only on q,

$$wt(\texttt{EndFit}(I)(t)) = wt(\texttt{EndFit}(\widetilde{I})(t)), \quad \text{for } t \in (-\infty, st(q)]. \tag{3.5}$$

By combining (3.2), (3.3), (3.4) and (3.5), we have

$$merit(\texttt{EndFit}(I)) \geq merit(\texttt{EndFit}(\widetilde{I})).$$

Q.E.D.

Theorem 3.7 *For any instance I,*

$$\text{merit}(\texttt{opt}(I)) \leq 2 \cdot \text{merit}(\texttt{EndFit}(I)).$$

Proof.

By Lemma 3.2, we can assume that I is intact in $\texttt{opt}(I)$. Let $\widetilde{I}$ be the trimmed instance of I such that for any request $q \in I$, if $(\texttt{opt}(I))^{-1}(q) = (t_1, t_2]$, then $st(\widetilde{q}) := t_1$ and $sz(\widetilde{q}) := t_2 - t_1$; otherwise if $\texttt{opt}(I)^{-1}(q) = \emptyset$, then $sz(\widetilde{q}) := 0$. We further assume that $\widetilde{I}$ is intact in all the plans.

$\widetilde{I}$ has the desirable property that requests arrive in a "constant" rate, that is, if a request q arrives at time t, then no other request arrives during $(t, t + sz(q))$.

Let H_1 and H_2 be two identical copies of $\texttt{EndFit}(\widetilde{I})$. We want to charge requests in $\texttt{opt}(\widetilde{I})$ to H_1 and H_2.

Consider a request q in $\widetilde{I}$. Upon arrival of q, there are two cases.

1. If q is allocated in the plan $Plan^+(\texttt{EndFit}, \widetilde{I}, q)$, it is possible that there are some requests which are originally allocated in $Plan^-(\texttt{EndFit}, \widetilde{I}, q)$, but not in the new plan. Call these requests the *ousted requests.*

2. Otherwise, call q the ousted request.

Let s be the total size of the ousted requests. Note that $s \leq sz(q)$ and in $Plan^+(\texttt{EndFit}, \widetilde{I}, q)$, the total merit of the ousted requests is not more than the total merit of the requests allocated in $(st(q), st(q) + s]$. Further note that the plan will be carried out without interruption at least until $st(q) + sz(q)$ since there is no request arrives during $(st(q), st(q) + sz(q))$. Charge the ousted requests to H_2 at $(st(q), st(q) + sz(q)]$ and the served requests in $(st(q), st(q) + sz(q)]$ to H_1 at $(st(q), st(q) + sz(q)]$. Now, we have

$$2 \cdot merit(\texttt{EndFit}(\widetilde{I})) \geq merit(\texttt{opt}(\widetilde{I})).$$

By Lemma 3.6, we have

$$2 \cdot merit(\texttt{EndFit}(I)) \geq merit(\texttt{opt}(I)).$$

Q.E.D.

3.3.3 Competitive ratio of `ArbFit`

Lemma 3.8 *There is a manager in* `ArbFit` *with competitive ratio at least 3.*

Proof.

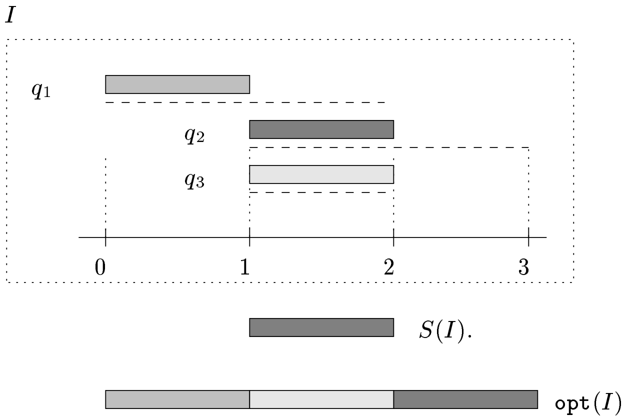


Figure 3.6: Counter example for Lemma 3.8.

Consider the instance comprising the requests $q_1 := (0, 2, 1, 1)$, $q_2 := (1, 3, 1, 1 + \epsilon)$ and $q_3 := (1, 2, 1, 1)$, where ϵ is any positive constant. The optimal merit is $3 + \epsilon$ as illustrated in Figure 3.6(b). Consider the manager S who, on the arrival of the first request, applies the rule of `EndFit` but changes to the rule of `Greedy` upon arrival of subsequent requests. It is easy to verify that S will only serve the request q_2 and thus the ratio $\texttt{opt}(I)/S(I)$ is $(3 + \epsilon)/(1 + \epsilon) = 3 - \delta$, for some positive δ. Since $\delta \to 0$ as $\epsilon \to 0$, the competitive ratio of S is at least 3.

Q.E.D.

Note that the above counter example is constructed by combining the counter examples for `Greedy` and `EndFit`. Similarly the proof of the upper bound is based on the ideas in the proof for `Greedy` and `EndFit`. Note that the definition of `ArbFit` gives its managers too much freedom in deciding their plans, and this makes analysis difficult. To restrict this freedom, given a manager S and I, we find a well-behaved manager $\widetilde{S}$ and an instance $\widetilde{I}$ such that $\widetilde{S}(\widetilde{I})$ and $\texttt{opt}(\widetilde{I})$ are "same" as $S(I)$ and $\texttt{opt}(I)$ respectively. Then, we exploit the property of $\widetilde{S}$ and $\widetilde{I}$ to prove the claimed bound.

Theorem 3.9 *For any $S \in$ `ArbFit`. S has competitive ratio of at most 3.*

Proof.

Given an instance I and manager S, we first define a modification $\widetilde{I}$ which is of a simple form, and a well-behaved Manager $\widetilde{S}$. Next, we show that

$$3\ merit(\widetilde{S}(\widetilde{I})) \geq merit(\texttt{opt}(\widetilde{I})).$$

The instance $\widetilde{I}$ is defined in a way that $merit(\texttt{opt}(I)) = merit(\texttt{opt}(\widetilde{I}))$ and $merit(S(I)) = merit(S(\widetilde{I}))$.

Part I. The following is a lemma similar to Lemma 3.2, which we state without proof.

Lemma 3.10 *Given a manager $S \in$ `ArbFit` and an instance I, suppose I_1 is a refinement of I, then there is another $S_1 \in$ `ArbFit` such that for any $t \in \mathbb{R}$, p' is in the refinement of p, where $p' := S_1(I_1)(t)$ and $p := S(I)(t)$.*

The given instance I may not be intact in $S(I)$. We want to achieve the intactness by refining I. This can be done by incrementally break the requests in I to match the schedule $S(I)$. However, since the refined instance I' is no longer same as I and it is possible that $S(I')$ is very much different from $S(I)$. By Lemma 3.10, we can find a S' such that $S'(I')$ is essentially same as $S(I)$. Thus, we can find an refinement I' of I that is intact in the schedule $S'(I')$. The intactness can be similarly extended to other schedules, that is, given a schedule H, we could find an refinement I^* and schedule S^* such that I^* is intact in both H and $S^*(I^*)$.

By the above argument, we may assume that the given instance I is intact in $\mathtt{opt}(I)$, $S(I)$ and all plans $\mathit{Plan}^+(S, I, q)$.

Part II. Given an instance I, let $J_{\mathtt{opt}}$ be the set of requests that are served in $\mathtt{opt}(I)$ and let $J_{\mathtt{arb}}$ be the set of requests that are served in $S(I)$. Let $\widetilde{I}$ be a trimmed instance of I defined as follow:

1. For each $q \in J_{\mathtt{arb}}$, the corresponding trimmed request $\widetilde{q}$ is the one whose $st(\widetilde{q}) := s'$ and $sz(\widetilde{q}) := sz(q)$, where $(s', t'] := K^{-1}(q)$ and $K := S(I)$;

2. For each $q \in I - (J_{\mathtt{arb}} \cup J_{\mathtt{opt}})$, $sz(\widetilde{q}) := 0$;

3. For each $q \in J_{\texttt{opt}} - J_{\texttt{arb}}$, the corresponding trimmed request has start-time $st(\widetilde{q}) := u'$ and $sz(\widetilde{q}) := v' - u'$, where $(u', v'] := H^{-1}(q)$ and $H := \texttt{opt}(I)$.

Let $\widetilde{J}_{\texttt{opt}}$ and $\widetilde{J}_{\texttt{arb}} \subseteq \widetilde{I}$ be the corresponding set of $J_{\texttt{opt}}$ and $J_{\texttt{arb}}$ respectively.

It is easy to see that there exist a manager $\widetilde{S} \in \texttt{ArbFit}$ such that $\widetilde{S}(\widetilde{I}) = S(I)$. However, we can show more. In Part III, we describe a "well-behaved" $\widetilde{S}$.

Part III. Let $\widetilde{S}$ to be the server that always allocates a request in $\widetilde{J}_{\texttt{arb}}$ in the earliest time-slot $[\mathrm{st}(\widetilde{q}), \mathrm{st}(\widetilde{q}) + \mathrm{sz}(\widetilde{q}))$. Note that there are only one pending request in $\widetilde{J}_{\texttt{arb}}$. For the pending requests in $\widetilde{J}_{\texttt{opt}} - \widetilde{J}_{\texttt{arb}}$, it allocates them in a way same as `EndFit`. We can ignore the remaining pending requests since all of them have size 0. Figure 3.7 illustrates an example of a plan.

By definition, $merit(\widetilde{S}(\widetilde{I})) = merit(S(I))$.

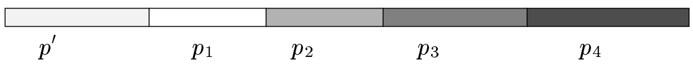


Figure 3.7: In this example, $wt(p_i) > wt(p_j)$ if and only if $i > j$, and $wt(p_1) < wt(p') < wt(p_2)$.

We need to show that this is a valid manager in `ArbFit`. That is, all the plans satisfy the restriction 2(∗) in the definition of `ArbFit`. We show this by contradiction.

Suppose $\widetilde{q}$ is the request such that $Plan^+(\widetilde{S}, \widetilde{I}, \widetilde{q})$ does not satisfy the restriction $2(*)$, Violation of restriction $2(*)$ implies that there is a request, say $\widetilde{q}'$, that is not allocated completely, and yet there is at least a request lighter than $\widetilde{q}'$ that is allocated in $[st(\widetilde{q}), dl(\widetilde{q}'))$. By definition of $\widetilde{S}$, this lighter request is in $\widetilde{J}_{\mathtt{arb}}$ and note that there is only one request in $\widetilde{p}$. Call this lighter request $\widetilde{p}$. Let $\widetilde{P}$ be the set of pending requests heavier than $\widetilde{p}$. Thus, by definition, $\widetilde{q}' \in \widetilde{P}$.

Now, we consider the original manager S. Let q',q, p and P be the counterpart of $\widetilde{q}'$, $\widetilde{q}$, $\widetilde{p}$ and $\widetilde{P}$ respectively. Since p is served by S, it is allocated by S in $Plan^+(S, I, q)$ at the earliest time-slot. Therefore all requests in P, in particular q', must be allocated in $Plan^+(S, I, q)$, and they are all allocated at a time later than $st(\widetilde{q}) + sz(p)$.

If we swap the requests in $Plan^+(S, I, q)$ as in the proof of Lemma 3.1 (that is, by swapping a heavier request with a lighter request that is allocated in a later time-slot), then what we have is the result of applying `EndFit` on P. This implies that $\widetilde{q}'$ could be allocated in $Plan^+(\widetilde{S}, \widetilde{I}, \widetilde{q})$ and thus contradicts our assumption.

Part IV. Now we exploit the restricted form imposed on $\widetilde{S}$. Consider a request $\widetilde{q}$. Due to the arrival of $\widetilde{q}$, there are some requests in $Plan^-(\widetilde{S}, \widetilde{I}, \widetilde{q})$ that are not in $Plan^+(\widetilde{S}, \widetilde{I}, \widetilde{q})$. Note that the total size of these ousted requests is less than or equal to $sz(q)$, and furthermore, they are all lighter than $\widetilde{q}$.

We call these requests the *pushed-out* requests and say that they are being pushed-out by q.

Part V. Let H_1, H_2 and H_3 be three copies of $\widetilde{S}(I)$. We want to charge $\texttt{opt}(I)$ to them.

Now, consider the server $\widetilde{S}$ and its plan on the arrival of a request $\widetilde{q}$.

1. If $\widetilde{q} \in \widetilde{J}_{\texttt{opt}} \cap \widetilde{J}_{\texttt{arb}}$, by definition of $\widetilde{J}_{\texttt{arb}}$ and $\widetilde{S}$, $\widetilde{q}$ will be served. Charge the requests pushed by $\widetilde{q}$ to H_1. In addition, charge $\widetilde{q}$ to H_3.

2. Otherwise, if $\widetilde{q} \in \widetilde{J}_{\texttt{opt}} - \widetilde{J}_{\texttt{arb}}$ there are two sub-cases:

   (a) If $\widetilde{q}$ is allocated in $\mathit{Plan}(\widetilde{S}, \widetilde{I}, \widetilde{q})$, charge the requests pushed by $\widetilde{q}$ to H_2.

   (b) If $\widetilde{q}$ is not allocated, charge $\widetilde{q}$ to H_2.

Part VI. We have to check that all requests in $\widetilde{J}_{\texttt{opt}}$ are being charged. If a request is in $\widetilde{J}_{\text{opt}} \cap \widetilde{J}_{\text{opt}}$, then it is being charged to H_3. Otherwise, if it belongs to the sub-case (b), then it is charged to H_2. If it belongs to sub-case (a), then it will be allocated in $H^{+}_{\widetilde{q}}$. Since it is not in $\widetilde{J}_{\texttt{arb}}$, it will eventually be pushed and charged to H_1 or H_2. Therefore, we have

$$3\mathit{merit}(\widetilde{S}(\widetilde{I})) \geq \mathit{merit}(\texttt{opt}(\widetilde{I})),$$

which implies

$$3\mathit{merit}(S(I)) \geq \mathit{merit}(\texttt{opt}(I)).$$

Q.E.D.

It is instructive to see how the instance described in Lemma 3.8 is charged by the proof of Theorem 3.9. In this example, the request q_1, q_2 and q_3 are charged to H_1, H_2 and H_3 respectively.

3.4 Lower Bound

We want to show that no deterministic online manager achieves a competitive ratio better than $2(2-\sqrt{2})$. To prove this, we describe an adversary, who given any manager, after observing the behavior of this manager, gives an instance that this manager fails to outperform the claimed ratio.

The Adversary We describe a very simple adversary.

At time zero, the adversary issues two requests, $q_0 = (0, 2, 1, 1)$ and $q_1 = (0, 1, 1, \sqrt{2}-1)$ (Figure 3.8). At time $t = 1$, based on the portion of q_0 and q_1 served, the adversary decides whether he needs to issue another request $q_2 = (1, 2, 1, 1)$. If less than $\frac{1}{2}$ of request q_0 is served, q_2 is issued, otherwise, it is not issued.

If q_2 is issued, then the merit of the off-line optimal schedule is 2, but the merit gained by the manager is at most $\frac{1}{2} + \frac{\sqrt{2}-1}{2} + 1$; if q_2 is not issued, then the merit of the off-line optimal schedule is $1 + (\sqrt{2}-1)$ but the merit gained by the manager is at most $1+\frac{\sqrt{2}-1}{2}$. In either case, the off-line optimal schedule is always better by a factor of $2(2-\sqrt{2})$, which rounds to 1.1716.

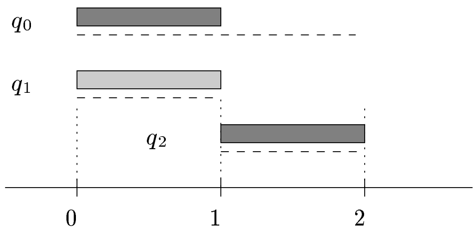


Figure 3.8: The adversary.

3.5 Comparing `Greedy` with `EndFit`

Both `Greedy` and `EndFit` have competitive ratio 2 and it is easy to find two instances I_0 and I_1 such that

$$\begin{aligned} merit(\texttt{Greedy}(I_0)) &> merit(\texttt{EndFit}(I_0)), \quad \text{and} \\ merit(\texttt{Greedy}(I_1)) &< merit(\texttt{EndFit}(I_1)). \end{aligned}$$

On the other hand, by working with a few examples, it seems that `Greedy` tends to perform better than `EndFit`. In this section, we show, in some aggregate sense, `Greedy` is better than `EndFit`.

For a request $q = (s, t, u, v)$, define its reflection $ref(q) := (-s, -t, u, v)$. For an instance $I = \{q_1, q_2, \ldots, q_n\}$, its reflection $ref(I)$ is

$$\{ref(q_1), ref(q_2), \ldots, ref(q_n)\}.$$

A technical complication is that if $st(q_i) = st(q_j)$, and q_i arrives before q_j, then $ref(q_i)$ is defined to have arrived before q_j. The motivation of these definition would be clear in the proof.

Note that the $\texttt{EndFit}(I)$ is very similar to $\texttt{Greedy}(\mathit{ref}(I))$. Figure 3.9 illustrates such an example.

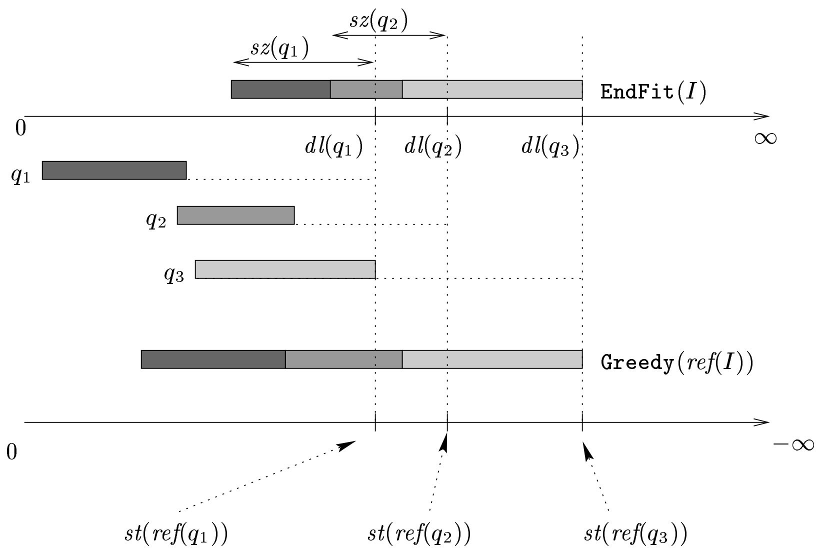


Figure 3.9: $\texttt{EndFit}(I)$ and $\texttt{Greedy}(\mathit{ref}(I))$

Theorem 3.11 *For any instance I,*

$$\begin{aligned} &\mathit{merit}(\texttt{Greedy}(I)) + \mathit{merit}(\texttt{Greedy}(\mathit{ref}(I))) \\ &\quad \geq \mathit{merit}(\texttt{EndFit}(I)) + \mathit{merit}(\texttt{EndFit}(\mathit{ref}(I)))\,. \end{aligned}$$

Furthermore, there is an I_0 such that

$$\begin{aligned} &\mathit{merit}(\texttt{Greedy}(I_0)) + \mathit{merit}(\texttt{Greedy}(\mathit{ref}(I_0))) \\ &\quad > \mathit{merit}(\texttt{EndFit}(I_0)) + \mathit{merit}(\texttt{EndFit}(\mathit{ref}(I_0)))\,. \end{aligned}$$

Proof.

Since $ref(ref(I)) = I$, it is sufficient to show that $merit(\texttt{Greedy}(ref(I))) \geq merit(\texttt{EndFit}(I))$. Let $H_{\texttt{end}} := \texttt{EndFit}(I)$ and $H_{\texttt{gdy}} := \texttt{Greedy}(ref(I))$. Note that, t is a switch point in $H_{\texttt{gdy}}$ if and only if $-t$ is a switch point in $H_{\texttt{end}}$.

Index the request in I as $\{q_1, q_2, \ldots, q_n\}$ where q_i is heavier than q_j if and only if $i < j$. Recall that if $wt(q_i) = wt(q_j)$, we resolved the tie by their order of arrival. By definition of *ref*, $ref(q_i)$ is heavier than $ref(q_j)$ if and only if q_i is heavier than q_j.

Let W be the set of all switch points in $H_{\texttt{end}}$. We show by induction on the index that for any request q_i, we have $wt(H_{\texttt{gdy}}(-t)) \geq wt(q_i)$ where $t \in H_{\texttt{end}}^{-1}(q_i) - W$. (We need to consider W due to the technical complication that the reflection of a time-slot, by definition, is not a time-slot). The base case, that is, the case where the request is the heaviest, is easy to check. Consider q_k. Let $(s_1, s_2] := H_{\texttt{end}}^{-1}(q_k)$. By the definition of $\texttt{EndFit}$, all the requests served in $(s_2, dl(q_k)] - W$ are heavier than q_k. Now consider $H_{\texttt{gdy}}$. By induction hypothesis, all the requests served in $(-dl(q_k), -s_2] - W$ are heavier than $ref(q_k)$, thus q_k must be served in a time later than $-s_2$. This implies that all the requests served in $(-dl(q_k), -s_2] - W$ are not lighter than q_k.

The second part of the theorem is easy to verify. **Q.E.D.**

3.6 Remarks

Randomized manager `gdyXend`. Consider the counter examples given in Lemma 3.3 and Lemma 3.5 (or Figure 3.4 and Figure 3.5). Although `Greedy` performs badly in the first example, it give the off-line optimal schedule for the second example. Similarly, `EndFit` performs badly on the second but not on the first example.

This observation suggests a randomized manager `gdyXend` who randomly changes mode between `Greedy` and `EndFit`. That is, upon arrival of a request, he tosses a coin. If the outcome is `Head` (which occurs with probability 1/2), he applies the rule of `Greedy` to compute the plan, otherwise, he applies the rule of `EndFit`.

Since the plans still follow the rule of `ArbFit`, `gdyXend` is 3-competitive. It is easy to verify that by giving the instance I as in the proof of Lemma 3.8, the expected merit points gains by `gdyXend` is $2 + \epsilon$. Therefore the competitive ratio of `gdyXend` is at least $1\frac{1}{2}$.

Number of switch points. By definition, the manager is given the freedom of switching requests with no penalty. This rises a few questions. Would a manager, achieves a better performance if he is allowed to increase the number of switches? If the number of switches does matter, then what is the "limit"? That is, what is a manager who is allowed to make infinite number of switches? We now give an alternative formulation of the scheduling problem which could be treated as one that allow infinite number of switches.

In this *extreme* scheduling problem, the request takes the same form (s, t, u, v) as is in the original problem. In this case, however, a request is served at a rate. How a request q is served throughout its life span is defined by its *rate function* $f_q : \mathbb{R} \rightarrow [0, 1]$. The total size of q served is

$$\int_{st(q)}^{dl(q)} f_q(t)\, dt,$$

and the number of merit gains from the serving of this request is

$$wt(q) \int_{st(q)}^{dl(q)} f_q(t)\, dt.$$

Given an instance I, a schedule is a collection of rate functions that satisfies the following:

1. for any $q \in I$,
$$\int_{st(q)}^{dl(q)} f_q(t)\, dt \leq sz(q),$$
and

2. for all $t \in \mathbb{R}$,
$$\sum_{q \in I} f_q(t) \leq 1.$$

Similar to the original problem, at time t_0, call the requests not yet completely served the *pending requests*, and the schedule the manager intends to carry out, which may be interrupted by the arrival of new request, the *plan*.

With respect to a schedule, the *remaining space* at time t_0 is

$$1 - \sum_{q \in I} f_q(t_0),$$

and we say that the schedule is *full* at time t_0 if the remaining space at t_0 is zero.

Now, we describe a manager in this extreme setting. A simple greedy manager `FairShare` computes its plan by considering the pending requests one by one, starting from the heaviest down to the lightest. Suppose q is the pending request being considered and s is the size of q not yet served. `FairShare` distributes q equally over the time slot $(st(q), dl(q)]$. Specifically, for each time slot which has less than $s/(dl(q) - st(q))$ remaining space, it allocates q to this time slots so that it is full; for the remaining time slots, it allocates $s/(dl(q) - st(q))$ of q over these slots. Then it recursively allocates the remaining of q until either q is completely allocated or $(st(q), dl(q)]$ is full.

Using the "charging" approach, it is easy to show that `FairShare` is 2-competitive. What about the lower bound? Using the counter example in Lemma 3.8, one can see that the competitive ratio of `FairShare` is at least 1.5. A better example is the instance $I_0 = \{q_1, \ldots, q_n\}$, where $q_i = (0, i, 1, 1)$ and q_i arrives before q_j if and only if $i > j$ (so that q_i is heavier than q_j for $i > j$). In this case, $merit(\texttt{opt}(I_0)) = n$. On the other hand,

$merit(\texttt{FairShare})(I_0) \leq b$ where b is the largest integer such that

$$\sum_{i=b}^{n} \frac{1}{i} \geq 1.$$

Therefore the optimal schedule is better by a factor of $e/(e-1) - \epsilon'_n$, where e is the base of the natural logarithms and ϵ_n is a constant depending on n which approaches 0 as n increases. Thus, the competitive ratio of `FairShare` is at least $e/(e-1)$ which rounds to 1.582.

Chapter 4

System Design and Implementation

4.1 Introduction

We have implemented an interactive progressive transmission system based on the transmission scheme described in section 2.2.7. In this chapter, we do not describe the implementation in detail. Instead, we focus on the general framework of the system design and possible improvements.

As is in the usual client-server system, there are three components in our system design: transmission scheme, server-side component and client-side component. We concentrate the discussion on the client-side component because it is the "front-end" of the visualization process.

We divide the discussion of the *client-side* component into two parts. We first describe the interactions between the user and the system, this includes issues on how to interpret the user's feedback, and what image is to be rendered in response to the user's feedback. Next, we describe the system architecture and data structure that support these user-system interactions. Note that there are a few concurrent activities carried out in the client-side and also note that the size of our image and the real-time requirement impose major constraints. As such, the coordination among various processes, subjected to the limited computing resource to achieve "real-time" performance is a key element in a successful implementation.

4.2 Transmission scheme

The design of this component is based on the 0-1 mask described in Section 2.2.7 with Haar wavelet.

Let $e_\ell[i,j]$ be the coefficient of the father wavelet $\Phi_{\ell,i,j}$ and let $d_\ell^k[i,j]$ be the coefficient of the mother wavelet $\Psi_{\ell,i,j,\cdot}^k$. Let D_ℓ^k be the matrix (array) whose (i,j)-th entry is $d_\ell^k[i,j]$. Similarly for E_ℓ. Call the set of matrices $D_1^k, \ldots, D_{\ell_0}$, $k \in \{\mathrm{v}, \mathrm{h}, \mathrm{d}\}$ the *coefficient pyramid* and the set of matrices $E_0, \ldots, E_{\ell_0}$ the *image pyramid*. Let C_ℓ^k be the mask. Recall that each entry $c_\ell^k[n,m]$ in the mask indicates whether the client has received the mother wavelet coefficient $d_\ell^k[n,m]$. We further simplified the scheme by using a common mask for the vertical, diagonal and horizontal components ($C_\ell = C_\ell^k$ for $k \in \{\mathrm{v}, \mathrm{d}, \mathrm{h}\}$). Again, call the set of matrices $C_0, \ldots, C_{\ell_0 - 1}$ the *mask pyramid*.

Note that the discussion in section 2.2.7 is not restricted to any particular wavelet. However, Haar has the non-overlap property which is very important in the implementation of the client-side component. The difficulties in using other smoother wavelets is the major fall back of the current design. We will address this issue again.

We assume lossless transmission. In fact, this scheme can only operate under lossless transmission, because both the client and server keep a common mask. In particular, we use the TCP/IP protocol for transmission.

We omit the detail specifications like the hand-shaking procedure and the byte arrangement.

4.3 Client-side component: Part one

4.3.1 Features

We first describe the features provided by the system from the viewer's point of view.

The viewer wants to view a large image (for example, a $3000 \times 5000 \times 24$ bits image), which is stored in a server connected by a thin-wire. Based on the partial data received from the server, the client reconstructs a multi-foveated image. As the image is large, only a portion of the whole multi-foveated image could be displayed within the displaying window. Call the displayed portion the *viewing image*. Call the boundary of the viewing image with respect to the whole image the *viewing boundary*. (Figure 4.1). The viewing image is multi-foveated. We use "pixel" as an unit to measure the width of the viewing image and viewing boundary. If the width of the viewing image is w_1 pixels, and the width of the viewing boundary is w_2 pixels, then we say that the *zoom levels* of the viewing image is $\log_2(w_2/w_1)$. In current implementation, we only allow integer zoom level.

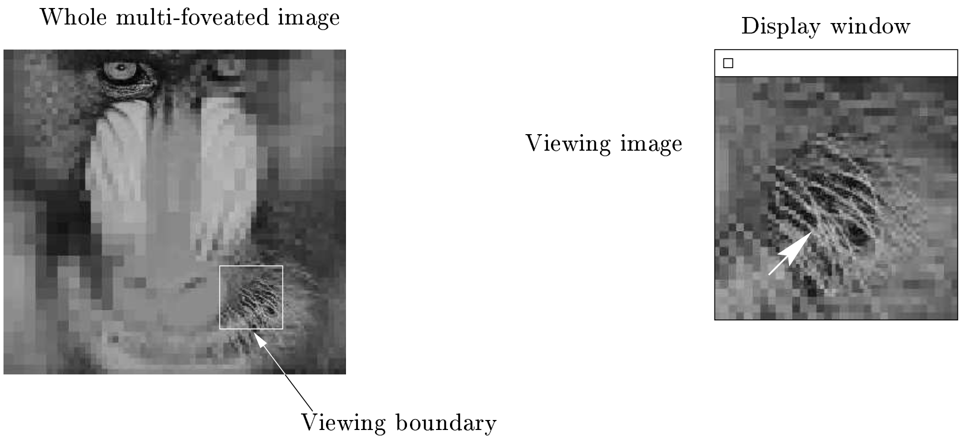


Figure 4.1: An example of display window, viewing boundary and viewing image.

After a network connection to the server is established, a low resolution image is first sent over and displayed on the display window. Now, the viewer indicates his *intention* by performing a sequence of the following "requests":

Moving	The viewer could freely move the mouse/cursor pointer within the image.
Zooming in/out	The viewer could zoom the image. That is, the viewing

boundary could be shrunk or enlarged while keeping the size of the display window unchanged.

Panning	The viewer could smoothly move the viewing boundary to a new location while keeping its size unchanged.
Jumping	The viewer could set the the top-right corner of the viewing boundary to a new location, regardless of its previous location.
Do-nothing	The viewer could choose to do nothing.

Details around the moving cursor pointer are sent across the network continuously and upon receipt, these are filled in continuously into the viewing image.

4.3.2 Translation of mouse movement

We now establish the link between the user's request and the data request to be sent to the server. Recall that each request is represented by a weight function which in turn is represented by the rate α, foveal resolution β and the fovea γ.

Assume that we are working on a discrete time frame $t = 0, 1, 2, \ldots$. At time t, let (x_t, y_t) be the position of the mouse pointer (with respect to the whole image). Let z_t be the current zoom level which is an integer. Let w_t to be the weight function to be sent to the server at time t. The weight function

w_t is represented by its rate α_t, fovea resolution β_t and gaze point γ_t. At all times, the fovea resolution is determined by z_i:

$$\beta_t \leftarrow 2^{z_t},$$

and the gaze point is always the mouse position:

$$\gamma_t \leftarrow (x_t, y_t).$$

The rate α_t is always set to a predefined value say, R_0, except when the user chooses to do nothing. In this case, a possible scheme is

$$\alpha_t \leftarrow \frac{\alpha_{t-1}}{1 + \alpha_{t-1} S},$$

where S is another predefined constant. This relationship of α_t and α_{t-1} can be rewritten as,

$$\frac{1}{\alpha_t} = \frac{1}{\alpha_{t-1}} + S.$$

Hence, if the user chooses to do nothing, the inverse rate $\frac{1}{\alpha}$ increases by S. In our implementation, the rate is not increased by a constant rate. We employ a heuristic which, taking the sparseness of the mask pyramid into consideration, chooses a rate such that the number of bytes requested for remains approximately fixed at a predefined constant. Thus, if the user moves the mouse to a region which already has data sent, then the increment S will be larger.

4.4 Client-side component: Part two

A few activities occur concurrently in the client-side: the system needs to keep track of the user's input, sending and receiving data to and from the server, reconstructing the image and rendering the reconstructed foveated image. We assign these activities to three modules: `Network`, `Display`, and `User`. We further introduce one more module `Manager` whose responsibility is to coordinate these modules.

The main shared data among these modules consists of the mask pyramid and the image pyramid. Call the shared data `Shared`.

4.4.1 Data Structure

Here are two observations.

1. Since the image is large, if the reconstructed image is stored in a full two dimensional array, locality will not be preserved in the sense that two neighboring pixels may be stored in two memory locations far apart, and thus leads to frequent page swaps.

2. On the other hand, since the image is large, the whole image can not be displayed at full resolution on the display window. The viewing image is a part of the image and/or a zoom-out image. Thus, it is not necessary to store a full two dimensional array of the reconstructed foveated image.

To handle the memory space problem (1), we exploit (2).

The reconstruction process is separated into two stages and `Shared` acts as an intermediate storage between the two stages. When data arrives, it is *updated* into `Shared`; and whenever required, the viewing image is then *rendered* from `Shared`.

`Shared` consists of the mask and image pyramid. Recall that the coefficient $d_\ell[i,j] = 0$ if the mask $c_\ell[i,j] = 0$ for any ℓ, i and j. For most of the time, the mask is very sparse. There are a number of ways to represent a sparse matrix. To choose a good representation, we consider the operations it supports. We delay the description of the data structure until appendix B. Instead, in the next three paragraphs, we describe the three basic operations it supports.

Updating. On receipt of a set of m (mother wavelet) coefficients from the server, suppose N is the width and height of the image, what is the number of operations required to update the image pyramid? By the fast wavelet transform, this updating can be done using $O(N^2)$ arithmetic operations, which is linear in term of the total number of image pixels. However, this is still not feasible since N is large. We need an incremental reconstruction algorithm that depends only on m.

For the Haar wavelet, due to its non-overlap property, there is a straight forward incremental reconstruction algorithm (assuming the pyramid is stored as a full two dimensional array), which takes $O(m)$ arithmetic operations. The sparse data-structure should achieve a similar performance.

Rendering. This is the most frequently called operation. Most system graphics routines that display an image onto the display unit takes as an argument a full two dimensional array, whose entries have one-one correspondence to the pixels of the displaying image. Thus, there is a gap between the image pyramid and the representation of the viewing image (Figure 4.2). Though the connecting step is straightforward, due to its frequent uses, it has to be brought into consideration in the design of the sparse image pyramid.

The use of Haar basis simplifies the operation, since we just have to duplicate a coefficient and move them into the viewing image, as shown in Figure 4.2.

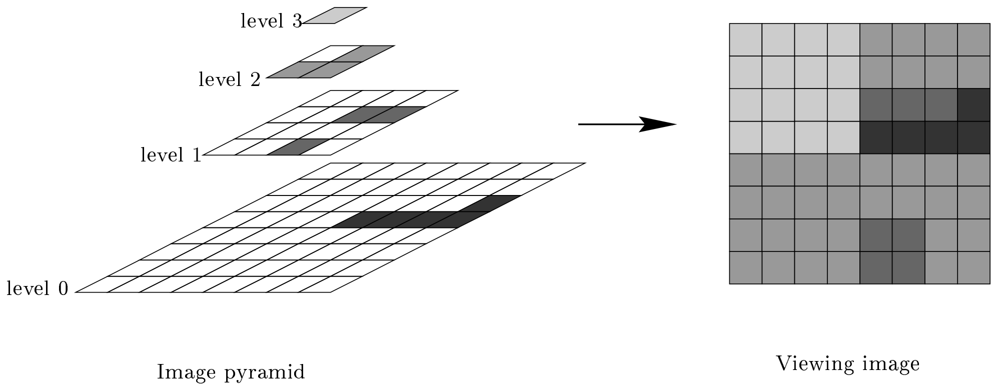


Figure 4.2: Each cell on the left represents a father wavelet coefficient. An empty cell indicates that the coefficient is not yet available. A cell on the right represents a pixel in the viewing image.

Consider a situation where the user wants to pan. A method to carry out such panning is by first moving the current pixels in the viewing image to their respective new location and the new exposed area is then rendered from the image pyramid. Since the viewing image is "moving", the user is insensitive to the details. Thus, it is more cost-effective to perform an approximate but fast rendering operation for the newly exposed area. This motivates rendering in different modes. The mode to be called is decided by the `Manager` module.

Query. Beside the updating and rendering operations, `Shared` is also required to support queries on the sparseness of the mask pyramid. Specifically, when given a query of the form $(\ell, r_s, r_e, c_s, c_e)$, `Shared` should report an approximation of the number of non-zero elements in this range, that is, it should give an approximation of the number

$$\sum_{r_s \le r \le r_e} \sum_{c_s \le c \le c_e} c_\ell^k[r, c].$$

This information is useful in, for example, deciding whether a request is to be scaled up or scaled down.

4.4.2 Coordinating various activities

A naive round-robin method to handle different modules might be a loop consisting the following steps:

1. Checks whether the user wants to zoom, pan, or jump, and immediately performs this operation.

2. Checks the current mouse position and translates it into a request.

3. Sends the request to the server and waits for the reply. Once the data arrive, updates `Shared`.

This is clearly not a good solution in general. For example, while waiting for the reply in step 3, the user may want to pan the viewing image.

In our solution, all four modules (`User`, `Network`, `Display` and `Manager`) run concurrently. `Manager` holds responsibility for coordinating the other modules. All modules report their status to `Manager` and `Manager` issues instructions, based on the current status and the sparseness of `Shared`.

A simple manager is as follows:

1. If `Display` is idling, constructs and issues an instruction.

2. If `Network` is idling, constructs and issues an instruction.

3. Checks the current status.

Figure 4.3 depicts the relationship among various modules and data-structure. To understand this figure, we describe the client-side activities after the viewer issues his requests.

1. `User` reports the status of the input devices, for example mouse position, to `Manager`.

2. Upon notification by `User`, `Manager` translates the status of the input devices to a list of requests. Next, based on some strategy, `Manager` decides whether this request is to be dropped, scaled down or delayed. If the decision is to serve a scaled down request $\tilde{q}$, `Manager` instructs `Network` to send $\tilde{q}$ to the server.

3. `Network` sends the request to the server and waits for the requested data. After all data arrived, `Network` updates `Shared` and notifies `Manager`.

4. Upon notification, based on the current situation, the `Manager` decides whether a sub-image is to be rendered. If this is the case, it instructs the `Display` to do so.

5. `Display`, instructed by `Manager`, renders the sub-image and displays it on the display window.

4.5 Server-side component

A very important consideration regarding the design of server is the memory usage. The current design breaks each matrix M_i into sub-matrices (typically 128 by 128 coefficients), where each sub-matrix is stored as a single file. Whenever the client requests coefficients which lie in some sub-matrix, the whole sub-matrix is loaded into the main memory (unless it is already in

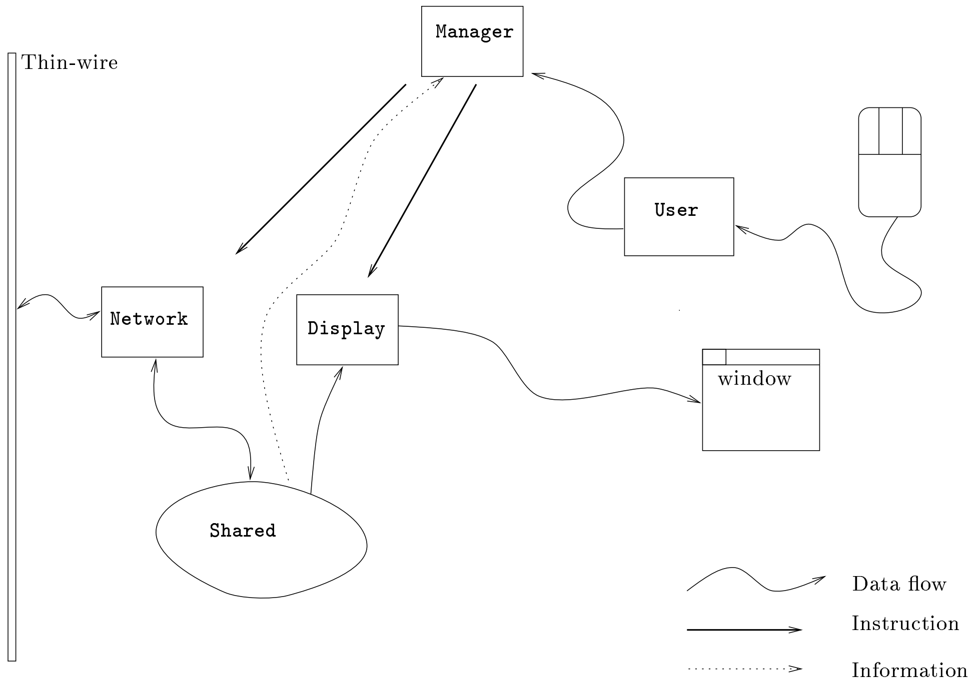


Figure 4.3: Overall System Design.

memory). The requested coefficients are then extracted and sent across. When all coefficients in the sub-matrix have been sent, it is released from the main memory. Currently, we rely on the operating system to deal with the caching between secondary memory and the main memory. It is not difficult to incorporate a more sophisticated cache management algorithm which could be "tuned" for server of different computing capacity and which could anticipate the client's requests.

As for the mask pyramid, in the current implementation, its data-structure is exactly same as that in the client-side (Section 2.3). Since a server may serve many clients, maintaining a pyramid for each client can become a burden. Currently, we do not have a straight forward way to improve it. One possible way is to group coefficients into blocks (for example 8 by 8 coefficients per block), and force the client to receive a whole block whenever the client request for any coefficient in that block. In this way, the size of the mask pyramid could be reduced by a factor of 64.

4.6 Implementation

We implemented our system in a Unix platform and the program is written in C++. Concurrency among the modules is achieved by using thread programming. Thus, the time sharing among the modules is being taken care of by the operating system. The graphics interface is done by directly calling the X-windows library. The system is tested on SPARCstation 5. No special graphics hardware is assumed.

4.7 Remarks and Future works

In this section, we describe some possible improvements, most of which centered around the trade-offs among these requirements:

1. Reconstruction time: Suppose the client receives an additional m bytes, this is the time which the client takes to update the data structure (in our implementation, the image and mask pyramid).

2. Distortion rate: This reflects the effectiveness of the transmission scheme. Assume that the weight function w_0 is fixed throughout the transmission, and let I_b be the reconstructed foveated image after b bytes have been received by the client, then we want a fast decay (as a function of b) of the weighted distortion,

$$||I_b - I||_w^2.$$

3. Response time: Note that there is a delay between the time the user issues his request, say a request to pan, and the time the image is actually displayed. We call this delay the *response time*. In our implementation, the rendering time is the biggest factor in the response time.

The followings are possible trade-offs among these three requirements.

Using other wavelet. Using Haar wavelet to approximate the foveated image causes undesirable visual artifacts. In addition, the distortion rate is poorer than for other smoother wavelets. However, Haar has the non-overlap property which we have exploited in this implementation to achieve a fast incremental reconstruction.

It is unlikely that a fast incremental reconstruction algorithm exists for overlapping wavelet. However, we could tackle this problem in another direction. Since the viewing image is small compare to the whole image, on arrival of new coefficients, the father wavelet coefficients corresponding to the viewing image could be reconstructed first; others could be delayed until an appropriate time.

Using other transmission schemes. Chapter two describes a few transmission schemes. The method implemented here in fact is the simplified version. A more sophisticated transmission scheme has better distortion rate but slower reconstruction.

Reducing the pyramid. We have mentioned in section 4.5 that a possible method to reduce the size of the pyramid is by grouping coefficients into blocks. In so doing, we will have lower performance in distortion rate but faster reconstruction.

Measurement of real-time performance. A way to combine the measurement of reconstruction time, distortion rate and user response time could be as follow: let I_t be the reconstructed multi-foveated image at time t. Let I_t' be the image displayed. Note that I_t' is not necessarily same as I_t. Let w_t be the weight function corresponding to the user's request at time t and let the *window* $W_t : \mathbb{R}^2 \to \{0, 1\}$ be the function where $W_t(x, y) = 1$ if and only if (x, y) is a point/pixel within the viewing boundary (Figure 4.1). The effectiveness of the whole integrated system at time t could be measured by

$$\|(I - I_t')W_t\|_{w_t}.$$

Appendix A

A.1 Schur's Lemma

Lemma A.1 *(Schur). Let T be an operator whose matrix elements in an orthonormal base $\{g_n\}_{n\in N}$ are $\theta_{n,m} = \langle Tg_n, g_m\rangle$. If there are two sequences of positive numbers $\{w_m\}$ and $\{\hat{w}_m\}$ and a constant B such that*

$$\sum_{m=0}^{\infty} |\theta_{n,m} w_m| < B\hat{w}_n \tag{A.1}$$

and

$$\sum_{n=0}^{\infty} |\theta_{n,m} \hat{w}_n| < B w_m, \tag{A.2}$$

then

$$\|T\|_2 \leq B.$$

Proof. By the Cauchy-Schwarz inequality and (A.1), for any $x :=(x_0, x_1, \ldots)$, and any n, we have

$$\begin{aligned}\sum_{m=0}^{\infty} |\theta_{n,m} x_m| &\leq \sum_{m=0}^{\infty} (|\theta_{n,m}| w_m)^{1/2} \left(|\theta_{n,m}| x_m^2 \frac{1}{w_m}\right)^{1/2} \\ &\leq (B\hat{w}_m)^{1/2} \left(\sum_{m=0}^{\infty} |\theta_{n,m}| x_m^2 \frac{1}{w_m}\right)^{1/2}.\end{aligned}$$

Combine with (A.2), we have

$$\begin{aligned}
\|Tx\|_2^2 &= \sum_{n=0}^{\infty}\left(\sum_{m=0}^{\infty}\theta_{n,m}x_m\right)^2 \\
&\leq \sum_{n=0}^{\infty}\left((B\hat{w}_n)\sum_{m=0}^{\infty}|\theta_{n,m}|x_m^2\frac{1}{w_m}\right) \\
&= \sum_{m=0}^{\infty}B\frac{x_m^2}{w_m}\sum_{n=0}^{\infty}|\theta_{n,m}\hat{w_m}| \\
&\leq B^2\sum_{n=0}^{\infty}x_n^2.
\end{aligned}$$

Q.E.D.

A.2 Multiresolution analysis

A multiresolution analysis [19] is given by a sequence $\{V_j\}_{j\in\mathbb{Z}}$ of closed subspaces of $L^2(\mathbb{R})$ satisfying the followings:

1. $V_j \subset V_{j-1}$ for all $j \in \mathbb{Z}$.

2. $\bigcup_{j\in\mathbb{Z}} V_j$ is dense in $L^2(\mathbb{R})$ and $\bigcap_{j\in\mathbb{Z}} V_j = \{0\}$.

3. $f \in V_j \Leftrightarrow f(2^j\cdot) \in V_0$.

4. $f \in V_0 \Rightarrow f(\cdot - n) \in V_0$ for all $n \in \mathbb{Z}$.

5. There exist a $\phi \in V_0$, such that $\{\phi(\cdot - n) : n \in \mathbb{Z}\}$ is an orthonormal basis in V_0.

Let $\{V_j\}_{j\in\mathbb{Z}}$ be a multiresolution analysis, and let W_n be the orthogonal complement of V_n in V_{n-1}, that is,

$$V_n \oplus W_n = V_{n-1},$$

then, there is a function, known as the *mother wavelet* ψ, such that $\{\psi(\cdot - k)\}_{k\in\mathbb{Z}}$ forms an orthonormal basis of W_0.

Let

$$\begin{aligned}\phi_{j,m} &:= 2^{-j/2}\phi(2^{-j}(\cdot - m)), \quad \text{and}\\ \psi_{j,m} &:= 2^{-j/2}\psi(2^{-j}(\cdot - m)),\end{aligned}$$

then $\{\psi_{j,m}\}_{m\in\mathbb{N}}$ and $\{\phi_{j,m}\}_{m\in\mathbb{N}}$ are basis of V_j and W_j respectively.

A.3 Regularity

Let f be a function which is $n+1$ times continuously differentiable. For a $v \in \mathbb{R}$, such that $f^{(n+1)}$ is bounded in a neighborhood $[v-h_0, v+h_0]$ of v, let p_v be the Taylor polynomial in this neighborhood:

$$p_v(t) := \sum_{k=0}^{n} \frac{f^{(k)}(v)}{k!}(t-v)^k.$$

We say that the function f is uniformly Lipschitz α over $[a,b]$ if there exist a $K > 0$ such that

$$\text{for all } (t,v) \in [a,b]^2, |f(t) - p_v(t)| \leq K|t-v|^\alpha.$$

The Lipschitz regularity of f over $[a,b]$ is the sup of the α such that f is Lipschitz α.

Lemma A.2 *If f is uniformly Lipschitz α over $[a, b]$, then the function g*

$$g(x) := c^{-\alpha} f\left(cx\right),$$

is uniformly Lipschitz α over $[c^{-1}a, c^{-1}b]$, where c is a positive constant.

Proof. Let p_{cw} and q_w be the Taylor formula of f and g at cw and w respectively. By definition,

$$g^{(k)}(x) = c^{-\alpha+k} f^{(k)}\left(cx\right).$$

Thus,

$$q_w(x) = \sum_{k=0}^{n} c^{-\alpha+k} \frac{f^{(k)}(cw)}{k!}(x-w)^k = c^{-\alpha} p_{cw}(cx).$$

Therefore, for all $(x, w) \in [c^{-1}a, c^{-1}b]^2$,

$$\begin{aligned} |g(x) - q_w(x)| &= |c^{-\alpha} f(cx) - c^{-\alpha} p_{cw}(cx)| \\ &< Kc^{-\alpha}|cx - cw|^{\alpha} = K|x - w|^{\alpha}. \end{aligned}$$

Q.E.D.

Theorem A.3 *Let ψ be a wavelet with n vanishing moments, $\psi \in C^n$ and has a compact support. Let $0 < \alpha < n$ be a non-integer real number. If $f \in L^2(\mathbb{R})$ is uniformly Lipschitz α over $[a, b]$, then for any $\epsilon > 0$, there exist $A > 0$ such that, for all $u \in [a, b]$ and $s \in \mathbb{R}^+$, we have:*

$$|\langle \psi^{u,s}, f \rangle| \le A s^{\alpha+1/2},$$

where $\psi^{u,s}(t) := |s|^{-1/2} \psi(\frac{t-u}{s})$.

A proof of the above theorem and a discussion on regularity could be found in [21].

Appendix B

B.1 Sparse representation for 0-1 Mask Pyramid

The 0-1 mask pyramid stores the mask $\{c_j[m,n]\}_{j,m,n}$. We use a straightforward representation. The pyramid is represented as a list of sparse matrices. Each sparse matrix is represented as an array of ordered linked list, where each list corresponds to a row. Each item in the linked list stores two integers: the starting column and ending column of non-zero entries. The items in a list are ordered in term of the starting column.

Conceptually, each linked list is a sequence of non-overlapping intervals, and the union of this intervals is the indices of non-zero entries.

Updating this pyramid is straightforward. However, care has to be taken in implementing the rendering operation so as to achieve computational efficiency.

B.2 Sparse representation for image pyramid

The image pyramid stores the coefficient $\{d_j^k[m,n]\}_{k,j,m,n}$. Note that an entry $d_j^k[m,n] = 0$ if the corresponding entry in the 0-1 mask $c_j[m,n] = 0$. Let us call an entry $d_j^k[m,n]$ in the image pyramid non-available if and only if $c_j[m,n] = 0$.

The image pyramid is represented as a list of sparse matrices and each matrix is represented in two levels. The matrix is subdivided into smaller sub-matrices of size, say 64 by 64. A sub-matrix is allocated only when one of its entry is available. The first level is a matrix (two-dimensional array) of pointer where each points to a sub-matrix (Figure B.1).

Recall that one of the supported operations is query, that is, given a query of the form $(\ell, r_{\mathrm{s}}, r_{\mathrm{e}}, c_{\mathrm{s}}, c_{\mathrm{e}})$, an approximation of the total number of non-available entries in this range is returned. To facilitate this operation, for each sub-matrix, we store a sub-total, which is the number of non-available entries in the sub-matrix. To answer a query, we just have to compute a weighted sum of the sub-totals.

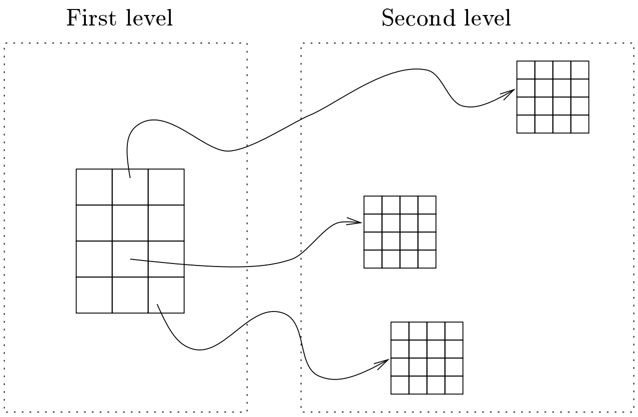


Figure B.1: Two-level representation.

Bibliography

[1] C.H. Anderson, P.J. Burt, and G.S. van der Wal. Change detection and tracking using pyramid transform techniques. *Proceedings of the DARPA Image Understanding Workshop*, pages 72–78, 1985.

[2] A. Basu, A. Sullivan, and K.J. Wiebe. Variable resolution teleconferencing. In *IEEE Systems, Man, and Cybernetics Conference*, pages 170–175, 1993.

[3] A. Basu and K.J. Wiebe. Videoconferencing using spatially varying sensing with multiple and moving fovea. In *Proceedings of the IEEE International Conference on Pattern Recognition*, volume 3, pages 30–34, 1994.

[4] B. Bederson, R.S. Wallace, and E.L. Schwartz. A miniaturized active vision system. In *11th IAPR International Conference on Pattern Recognition*, pages 58–62, The Hague, Netherlands, 1992. Specialty Conference on Pattern Recognition Hardware Architecture.

[5] P.J. Burt. Smart sensing within a pyramid vision machine. *Proceedings of the IEEE*, 76(8):1006–1015, 1988.

[6] P.J. Burt and Theodore Adelson. A laplacian pyramid for data com-

pression. *IEEE Trans. Commun.*, C-8:1230–1245, 1981.

[7] B. Cabral, N. Cam, and J. Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. *1994 Symp. on Volume Visualization*, pages 91–97, 1994.

[8] E.C. Chang and C. Yap. A wavelet approach to foveating images. *13th ACM Symposium on Computational Geometry*, pages 397–399, 1997.

[9] Ingrid Daubechies. *Ten Lectures on Wavelets.* SIAM, 1992.

[10] A. Eleftheriadis and A. Jacquin. Automatic face location detection and tracking for model-assisted coding of video teleconferencing sequences at low bitrates. *Signal Processing: Image Communication*, 7(3):231–248, 1995.

[11] R.A. Fisher and H.M. Tong. A full-field-of-view dome visual display for tactical combat training. In *Proc. Image Conference IV*, Phoenix, Arizona, June, 1987.

[12] M. Frazier and B. Jawerth. Applications of the ϕ and wavelet transforms to the theory of function spaces. In *Wavelets and Their Applications*, pages 377–418. Jones and Bartlett, 1992.

[13] B. Girod. Eye movements and coding of video sequences. *SPIE Visual Communications and Image Processing*, 1001:398–405, 1988.

[14] G.G. Holmes. The cortical localization of vision. *Br. Med. J*, ii:193–199, 1919.

[15] Nuggehally S. Jayant and Peter Noll. *Digital coding of waveforms: principles and applications to speech and video.* Englewood Cliffs, New Jersey, 1984.

[16] Philip Kortum and Wilson S. Geisler. Implementation of a foveated image coding system for image bandwidth reduction. In *Human Vision and Electronic Imaging, SPIE Proceedings Vol. 2657*, pages 350–360, 1996.

[17] M. Levoy and R. Whitaker. Gaze-directed volume rendering. *Computer Graphics*, 24(2):217–223, March 1990.

[18] Richard J. Lipton and Andrew Tomkins. Online interval scheduling. *Proceedings of the 5th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 304–311, 1994.

[19] S. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Trans. Pattern Anal. Machine Intell.*, 11:674–692, 1989.

[20] S. Mallat, Z. Zhang, and G. Papanicolaou. Adaptive covariance estimation of locally stationary processes. *Annals of Stat.*, 1997 (To appear).

[21] Stephane Mallat. *A Wavelet Tour of Signal Processing.* Academic Press, 1998.

[22] M. Manasse, L.A. McGeoch, and D. Sleator. Competitive algorithms for server problems. In *Proc. 20th Annual ACM Symposium on Theory of Computing*, pages 322–333, 1988.

[23] David Marr. *Vision: a computational Investigation into the Human Representation and Processing of Visual Information.* W.H. Freeman, 1982.

[24] J. Max. Quantizing for minimum distortion. *IRE Transactions on Information Theory*, IT-6(1):7–12, 1960.

[25] B.H. McCormick, T.A. DeFanti, and M.D. Brown. Visualization in scientific computing. *ACM Computer Graphics (special issue)*, 21, 1987.

[26] M. Minkowski. Experimentelle untersuchungen uber die beziehungen der grosshirninde. *Arb. Hirnanat. Inst. Zurich*, 7:259, 1913.

[27] Alireza Moini. Vision chips or seeing silicon. www page at url:⟨http://www.eleceng.adelaide.edu.au/groups/gaas/bugeye/visionchips/⟩, Last update April 1997.

[28] Alireza Moini. Vision chips or seeing silicon. Technical Report, Center for High Performance Integrated Technologies and Systems, The University of Adelaide, March 1997.

[29] F. Panerai, C. Capurro, and G. Sandini. Space variant vision for an active camera mount. In *Proc. SPIE AeroSense95*, 1995.

[30] S. Poliak. The main afferent fibre systems of the cerebral cortex in primates. *Univ. Calif. Publ. Anat.*, 2:107–207, 1932.

[31] T.H. Reeves and J.A. Robinson. Adaptive foveation of mpeg video. *Proceedings, 4th ACM International Multimedia Conference*, 1996.

[32] E.L. Schwartz. Spatial mapping in primate sensory projection: Analytic structure and relevance to perception. *Biological Cybernetics*, 25:181–194, 1977.

[33] Eric L. Schwartz. Topographic mapping in primate visual cortex: History, anatomy, and computation. In D. H. Kelly, editor, *Visual Science and Engineering: Models and Applications*, pages 293–360. Marcel Dekker, 1994.

[34] C.E. Shannon. Communications in the presence of noise. In *Proceedings of the IRE*, volume 37, pages 10–21, January 1949.

[35] J.M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41(12):3445–3462, December 1993.

[36] E.P. Simoncelli and R.W. Buccigrossi. Embedded wavelet image compression based on a joint probability model. In *4th IEEE International Conference on Image Processing*, Santa Barbara, 1997.

[37] D. Sleator and R. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

[38] K.R. Sloan and S.L. Tanimoto. Progressive refinement of raster images. *IEEE Trans. Comput.*, C-28(11):871–874, 1979.

[39] Wavelet technology and information access. www page at url:⟨http://www.summus.com/⟩, Summus, Ltd, Last update November 21,1997.

[40] A. Tabernero, J. Portilla, and R. Navarro. Duality between the local spectrum of a signal and its inverse fourier transform, the local signal. Technical Report 53, Instituto de Optica (CSIC), Spain, 1997.

[41] S.A. Talbot and W.H. Marshall. Physiological studies on neural mechanisms of visual localization and discrimination. *Am. J. Ophthalmol*, 24:1255–1263, 1941.

[42] H.M. Tong and R.A. Fisher. Progress report on an eye-slaved area-of-interest visual display. In *Proc. Image Conference III*, Phoenix, Arizona, May, 1984.

[43] K.H. Tzou. Embedded max quantization. In *Proc. IEEE Int. Conf. Acoust. Speech. Sig. Proc.*, pages 505–508. ICASSP'86, 1986.

[44] Kou-Hu Tzou. Progressive image transmission: a review and comparison of techniques. *Optical Engineering*, 26(7):581–589, 1987.

[45] G.K. Wallace. The jpeg still picture compression standard. *Communications of the ACM*, 34(4):30–44, April 1991.

[46] R.S. Wallace, P.-W. Ong, B. Bederson, and E.L. Schwartz. Space variant image processing. *Intl. J. of Computer Vision*, 13(1):71–90, 1994.

[47] R.C. Wood. On optimum quantization. *IEEE Transactions on Information Theory*, IT-15(2):248–252, 1969.

[48] Y. Yeshurun and E.L. Schwartz. Shape description with a space-variant sensor: Algorithms for scan-path, fusion and convergence over multiple scans. *IEEE Trans. Pattern Anal. Machine Intell.*, PAPMI-11:1217–1222, 1989.