Christopher Tsai
May 5, 2007
EE 368 – Digital Image Processing

Problem Set IV

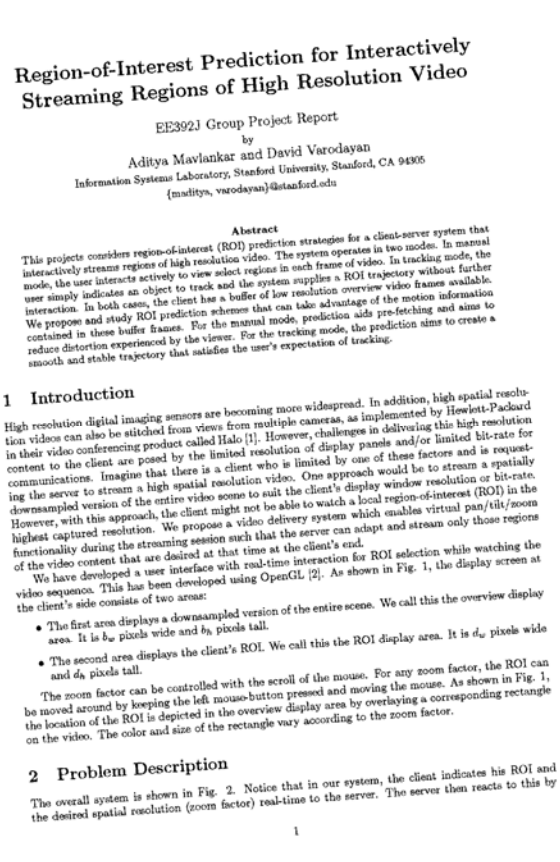
Problem 1 – Document Alignment

Initially, the tilted documents contain handwriting strokes that run not only along the tilted vertical axis but also perpendicular to that main axis; in other words, handwriting on the primary documents display strong preferential direction not only up and down the page (as with the letter *I*) but also left and right, where, for example, hyphens and crossed letters dominate.

Thus, in order to isolate the former vertical strokes, we strive first to eliminate their perpendicular counterparts through erosion. Unfortunately, because erosion reduces the size of all characters, we might thin the strokes beyond detectability in the Hough Transform; thus, instead of merely eroding our handwritten letters, we *open* the document to preserve the size of vertical strokes following elimination of their horizontal counterparts. The left edges of words and paragraphs display the strongest sense of vertical direction, so we set our structuring element to accentuate leftmost points:

```
imopen(~doc1, strel('line', charSize, likelyTilt(1)));
```

Experimenting with various character sizes, we find that a length of 20 pixels resonates most closely with the vertical strokes in the document. Finally, we apply opening with linear structuring elements of several different slopes (tilts) until we find the angle that yields the greatest number of resulting regions; openings that involve structuring elements tilted at incorrect angles (different from the actual document tilt) will not preserve many of the characters in the tilted document, since the 20-pixel line will not fit into many vertical strokes during erosion. When the maximal number of strokes remains following erosion, however, our linear structuring element likely aligns closely parallel to the vertical strokes in the document. We then open our document with this properly tilted structuring element and use this pre-processed (opened) binary image as the basis for our Hough Transform, since the opening no longer contains any perpendicular strokes that could also dominate the transform:



Region-of-Interest Prediction for Interactively Streaming Regions of High Resolution Video

EE392J Group Project Report

by

Aditya Mavlankar and David Varodayan
Information Systems Laboratory, Stanford University, Stanford, CA 94305
{maditya, vsrodayan}@stanford.edu

Abstract

This projects considers region-of-interest (ROI) prediction strategies for a client-server system that interactively streams regions of high resolution video. The system operates in two modes. In manual mode, the user interacts actively to view select regions in each frame of video. In tracking mode, the user simply indicates an object to track and the system supplies a ROI trajectory without further interaction. In both cases, the client has a buffer of low resolution overview video frames available. We propose and study ROI prediction schemes that can take advantage of the motion information contained in these buffer frames. For the manual mode, prediction aids pre-fetching and aims to reduce distortion experienced by the viewer. For the tracking mode, the prediction aims to create a smooth and stable trajectory that satisfies the user's expectation of tracking.

1 Introduction

High resolution digital imaging sensors are becoming more widespread. In addition, high spatial resolution videos can also be stitched from views from multiple cameras, as implemented by Hewlett-Packard in their video conferencing product called Halo [1]. However, challenges in delivering this high resolution content to the client are posed by the limited resolution of display panels and/or limited bit-rate for communications. Imagine that there is a client who is limited by one of these factors and is requesting the server to stream a high spatial resolution video. One approach would be to stream a spatially downsampled version of the entire video scene to suit the client's display window resolution or bit-rate. However, with this approach, the client might not be able to watch a local region-of-interest (ROI) in the highest captured resolution. We propose a video delivery system which enables virtual pan/tilt/zoom functionality during the streaming session such that the server can adapt and stream only those regions of the video content that are desired at that time at the client's end.

We have developed a user interface with real-time interaction for ROI selection while watching the video sequence. This has been developed using OpenGL [2]. As shown in Fig. 1, the display screen at the client's side consists of two areas:

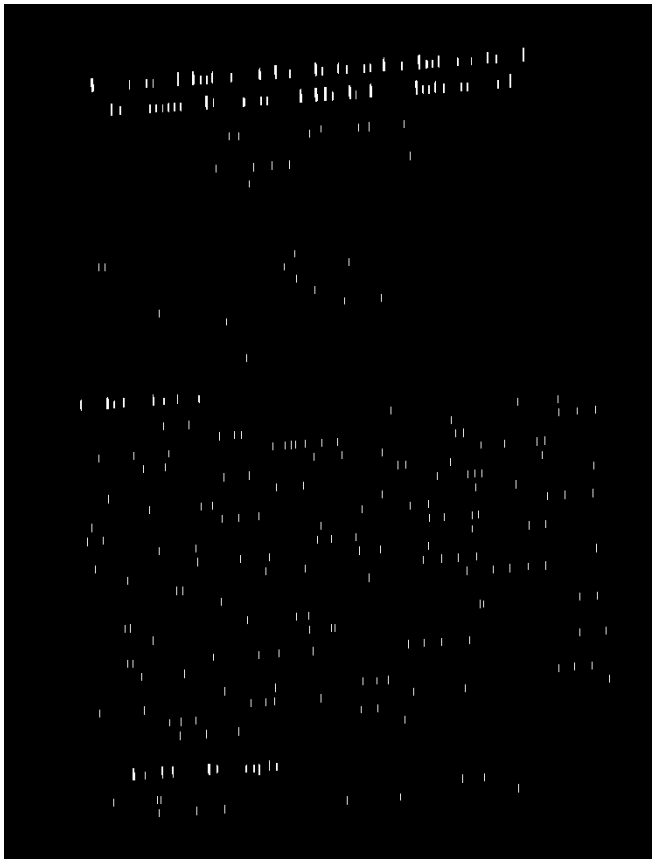
- The first area displays a downsampled version of the entire scene. We call this the overview display area. It is b_w pixels wide and b_h pixels tall.
- The second area displays the client's ROI. We call this the ROI display area. It is d_w pixels wide and d_h pixels tall.

The zoom factor can be controlled with the scroll of the mouse. For any zoom factor, the ROI can be moved around by keeping the left mouse-button pressed and moving the mouse. As shown in Fig. 1, the location of the ROI is depicted in the overview display area by overlaying a corresponding rectangle on the video. The color and size of the rectangle vary according to the zoom factor.

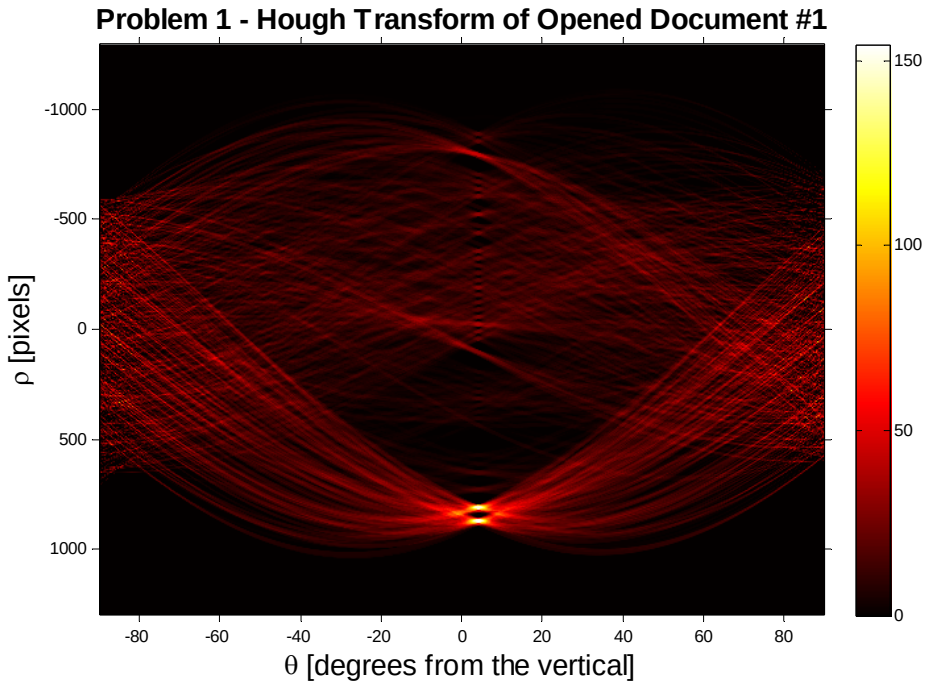
2 Problem Description

The overall system is shown in Fig. 2. Notice that in our system, the client indicates his ROI and the desired spatial resolution (zoom factor) real-time to the server. The server then reacts to this by

1

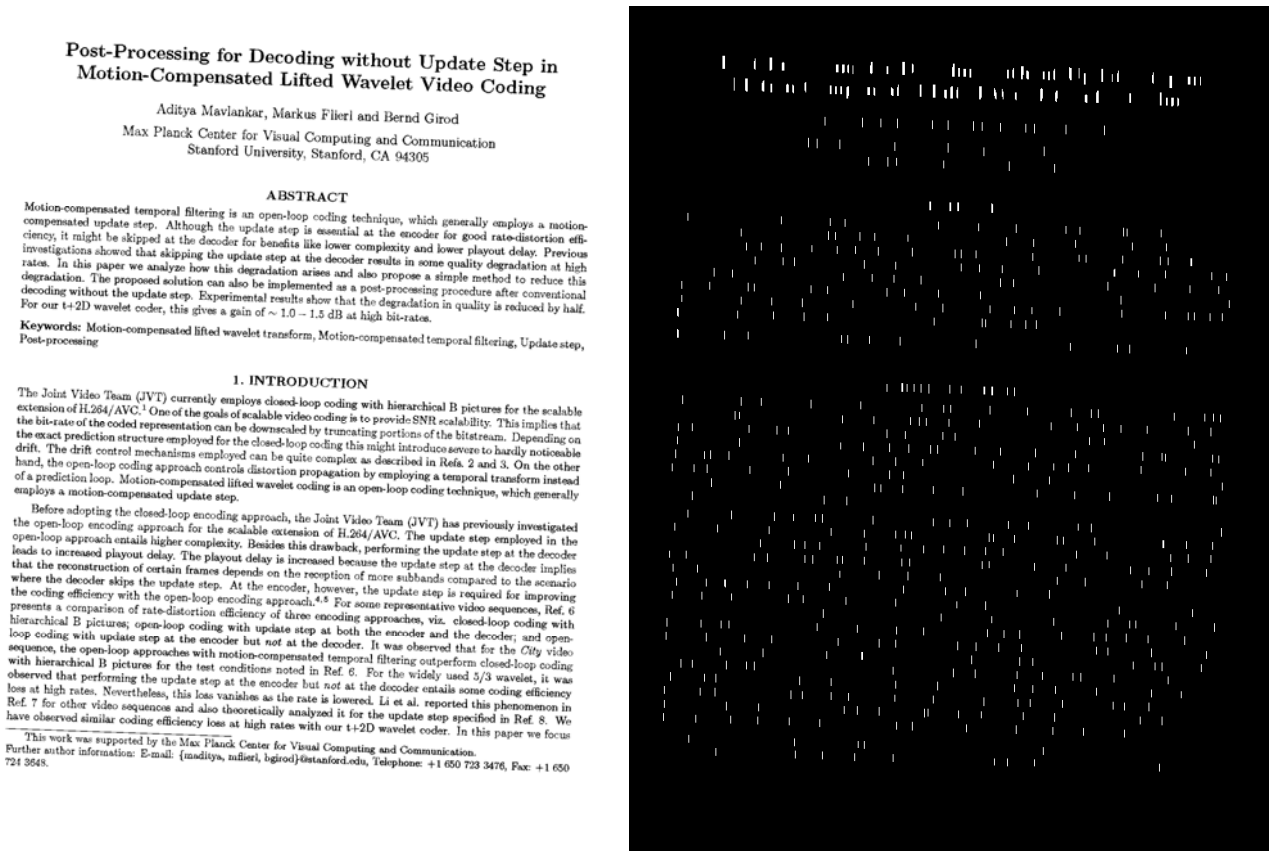


With only tilted vertical strokes prevalent in the opened image, the Hough Transform – or the Radon Transform – now easily converges to the dominant angle of vertical strokes:



From inspection of the bin plot (whose θ-axis we modified to be zero for an untilted document), we see that the bins around 4.15° off-vertical counterclockwise tilt accumulate the most hits in the Hough Transform. After averaging the five tilt angles whose Hough bins boast the most content, we conclude that the vertical strokes dominating the first document lean approximately 4.15° counterclockwise from the vertical of a perfectly aligned document, indicating that the first document is tilted by about 4.15°. Manual measurement of the document rotation ascertains this result.

We repeat the document opening with a differently tilted linear structuring element for the second document, resulting in another opened image accentuating the vertical strokes that run strongly along the direction of the document's rotation:



Post-Processing for Decoding without Update Step in Motion-Compensated Lifted Wavelet Video Coding

Aditya Mavlankar, Markus Flierl and Bernd Girod

Max Planck Center for Visual Computing and Communication
Stanford University, Stanford, CA 94305

ABSTRACT

Motion-compensated temporal filtering is an open-loop coding technique, which generally employs a motion-compensated update step. Although the update step is essential at the encoder for good rate-distortion efficiency, it might be skipped at the decoder for benefits like lower complexity and lower playout delay. Previous investigations showed that skipping the update step at the decoder results in some quality degradation at high rates. In this paper we analyze how this degradation arises and also propose a simple method to reduce this degradation. The proposed solution can also be implemented as a post-processing procedure after conventional decoding without the update step. Experimental results show that the degradation in quality is reduced by half. For our t+2D wavelet coder, this gives a gain of $\sim 1.0 - 1.5$ dB at high bit-rates.

Keywords: Motion-compensated lifted wavelet transform, Motion-compensated temporal filtering, Update step, Post-processing

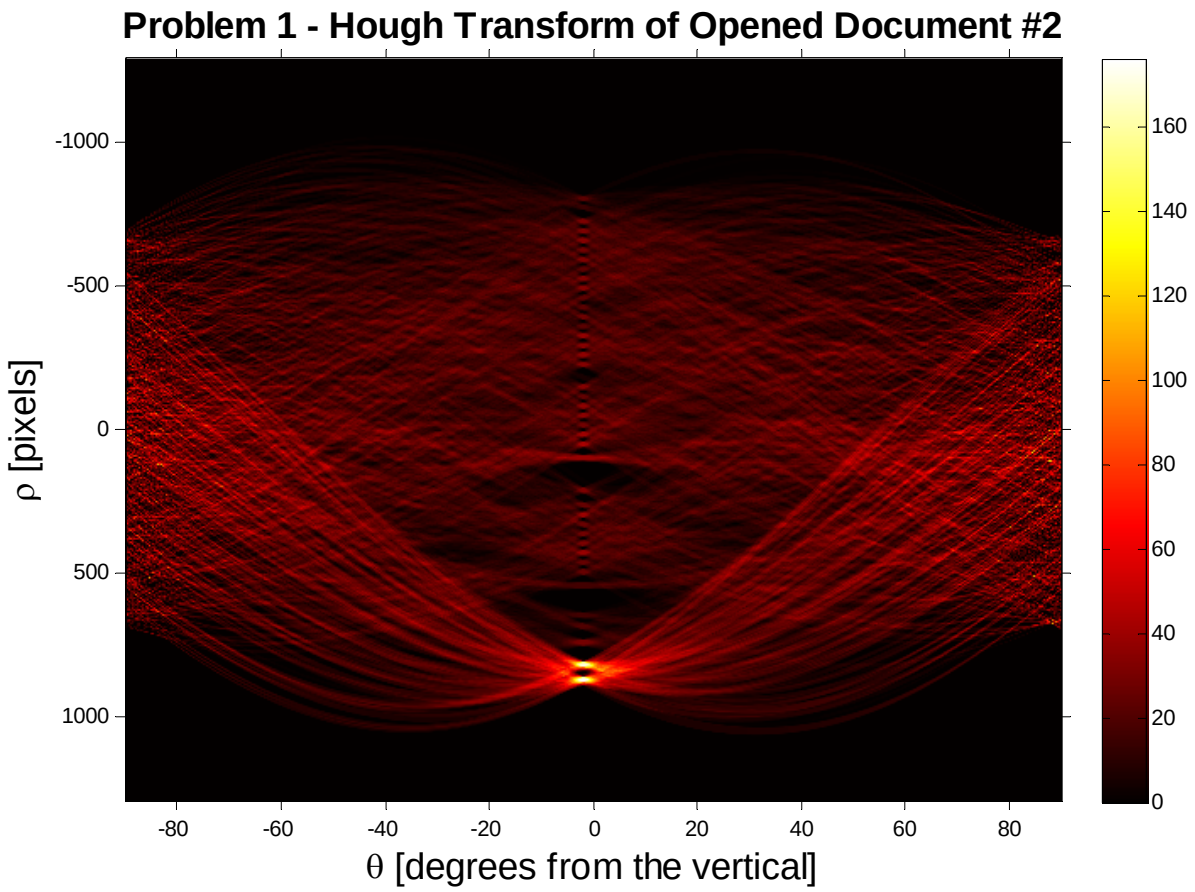
1. INTRODUCTION

The Joint Video Team (JVT) currently employs closed-loop coding with hierarchical B pictures for the scalable extension of H.264/AVC.[1] One of the goals of scalable video coding is to provide SNR scalability. This implies that the bit-rate of the coded representation can be downscaled by truncating portions of the bitstream. Depending on the exact prediction structure employed for the closed-loop coding this might introduce severe to hardly noticeable drift. The drift control mechanisms employed can be quite complex as described in Refs. 2 and 3. On the other hand, the open-loop coding approach controls distortion propagation by employing a temporal transform instead of a prediction loop. Motion-compensated lifted wavelet coding is an open-loop coding technique, which generally employs a motion-compensated update step.

Before adopting the closed-loop encoding approach, the Joint Video Team (JVT) has previously investigated the open-loop encoding approach for the scalable extension of H.264/AVC. The update step employed in the open-loop approach entails higher complexity. Besides this drawback, performing the update step at the decoder leads to increased playout delay. The playout delay is increased because the update step at the decoder implies that the reconstruction of certain frames depends on the reception of more subbands compared to the scenario where the decoder skips the update step. At the encoder, however, the update step is required for improving the coding efficiency with the open-loop encoding approach.[4,5] For some representative video sequences, Ref. 6 presents a comparison of rate-distortion efficiency of three encoding approaches, viz. closed-loop coding with hierarchical B pictures; open-loop coding with update step at both the encoder and the decoder; and open-loop coding with update step at the encoder but *not* at the decoder. It was observed that for the *City* video sequence, the open-loop approaches with motion-compensated temporal filtering outperform closed-loop coding with hierarchical B pictures for the test conditions noted in Ref. 6. For the widely used 5/3 wavelet, it was observed that performing the update step at the encoder but *not* at the decoder entails some coding efficiency loss at high rates. Nevertheless, this loss vanishes as the rate is lowered. Li et al. reported this phenomenon in Ref. 7 for other video sequences and also theoretically analyzed it for the update step specified in Ref. 8. We have observed similar coding efficiency loss at high rates with our t+2D wavelet coder. In this paper we focus

This work was supported by the Max Planck Center for Visual Computing and Communication.
Further author information: E-mail: {maditya, mflierl, bgirod}@stanford.edu, Telephone: +1 650 723 3476, Fax: +1 650 724 3648.

Performing the Hough Transform on this vastly simplified opened image, a small concentration of θ-bins visibly accumulates more objects than bins at other angles, as we observe from the Hough Transform:



Unlike strokes in the first document, near-vertical marks in this second document seems to favor the opposite direction of rotation, with a *negative* tilt angle relative to the proper vertical axis. As the Hough Transform attains its maxima in bins along $\theta \approx -1.95°$ (obtained again through the arithmetic mean of the five angles with the most densely populated bins), we ascertain that the vertical strokes spanning the second document run along an axis tilted approximately 1.95° clockwise from perfect alignment.

Problem 2 – Reverse Engineering

The equation used to generate the Zoneplate pattern in '`zoneplate.tif`' is

$$f(x,y) = \hat{f} \cdot cos\left(a_x(x-x_0)^2 + a_y(y-y_0)^2\right) + f_0$$

To determine the spatial frequency of this two-dimensional sinusoid, we first extract the phase argument:

$$\varphi(x,y) = a_x(x-x_0)^2 + a_y(y-y_0)^2.$$

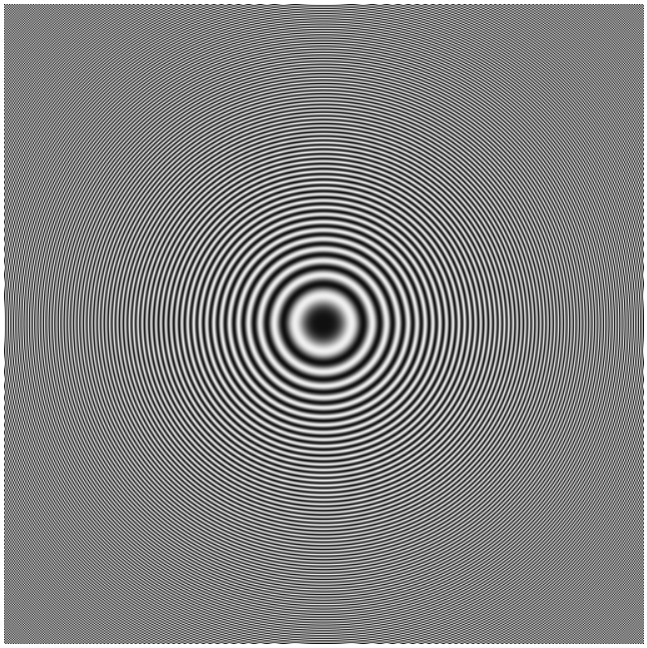
By definition, the angular frequency is the rate of change in the phase along the direction of interest. Applying the Chain Rule from multivariate calculus, we differentiate the phase:

$$\omega_x = \frac{\partial\varphi(x,y)}{\partial x} = 2a_x(x-x_0).$$

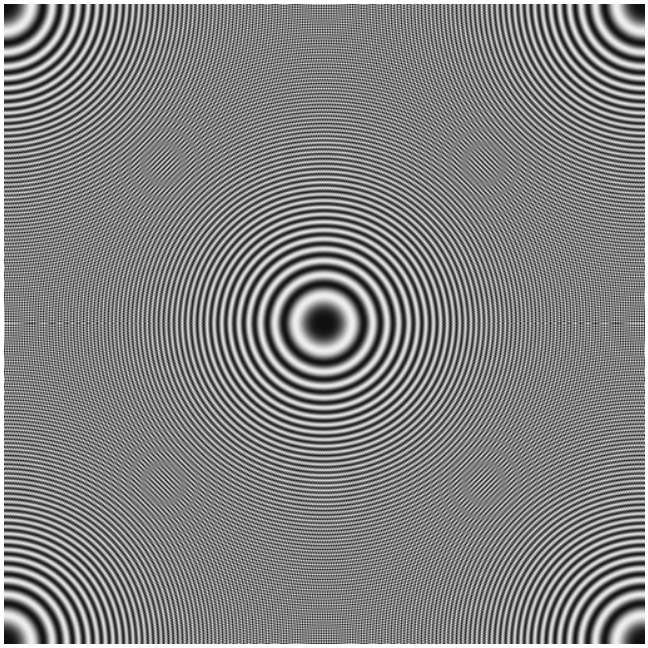
$$\omega_y = \frac{\partial\varphi(x,y)}{\partial y} = 2a_y(y-y_0).$$

These two expressions reveal that the frequency at any point in a zoneplate pattern is directly proportional to the distance of the point from center: (x_0, y_0). In other words, the four replicas that we see in the processed zoneplate image could represent replicas in either frequency or time:

Problem 2 - Original Zoneplate Image



Problem 2 - Processed Zoneplate Image



The center of the processed zoneplate image seems unaltered, so the mystery image processing algorithm must have either produced spatial ringing or frequency aliasing. From the zoneplate image

alone, we cannot determine whether replication has occurred in the spatial domain or the frequency domain, since the two are directly related for the zoneplate pattern.

If the algorithm duplicated the image spatially, then the frequency spectrum likely experienced ideal lowpass filtering, with the sharp frequency cutoff leading to spatial ringing at the corners. On the other hand, frequency replication would result from aliasing, as undersampling the original zoneplate image (at a sampling frequency lower than the Nyquist Rate) would lead high-frequency copies of the spectrum to alias into the central region as low frequencies; especially at the edges of the zoneplate image, where the frequencies are highest, sampling-induced adjacent replica images could easily extend into the central lowpass portion of the spectrum and masquerade as low-frequency signal if we did not space our samples close enough together (and therefore place the replicas far enough apart).

To determine whether replication is spatial or spectral, we examine the Lena image:

Problem 2 - Original Lena Image

Problem 2 - Processed Lena Image

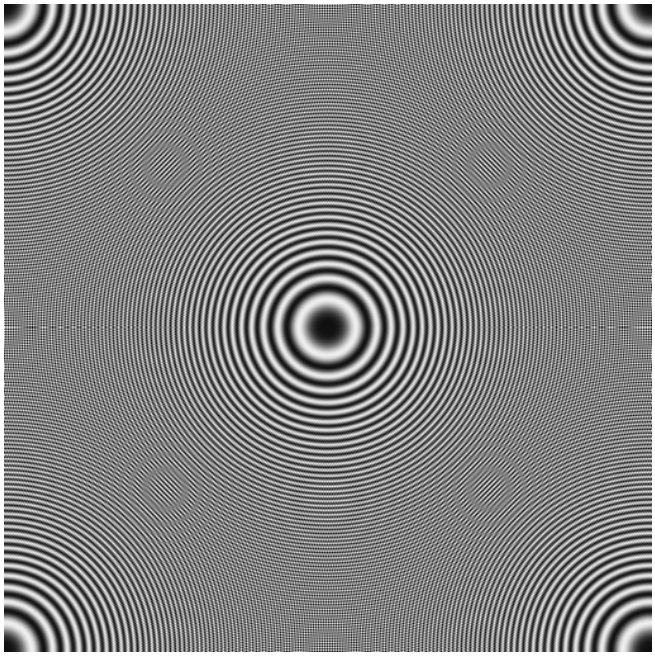
The two images look nearly identical. The difference image virtually fails to register, differing only at the edges of Lena's figure. Since no replication occurs in the processed Lena image, we conclude that the image processing algorithm must replicate the zoneplate pattern's central lobe in the frequency domain. Spectral aliasing occurs in the zoneplate image because its highest frequency is quite high, meaning that

undersampling occurs easily, for even a 2:1 downsampling; the Nyquist criterion for such a high maximum frequency holds that even slight downsampling will lead to some overlap in adjacent replicas in the sampled spectrum, resulting in the aliasing we witness in the corners, where high frequencies should appear. However, the Lena image displays none of these sampling artifacts because the Lena image spectrum contains much fewer high frequencies; most of the Lena image is relatively flat and unchanging, and even the edges do not oscillate as quickly as the outer zoneplate rings. Thus, downsampling the Lena image by a 2:1 factor and then interpolating to reconstruct a (512 × 512) image hardly affects image quality, since the image never contained frequencies high enough to alias into the central spectrum.

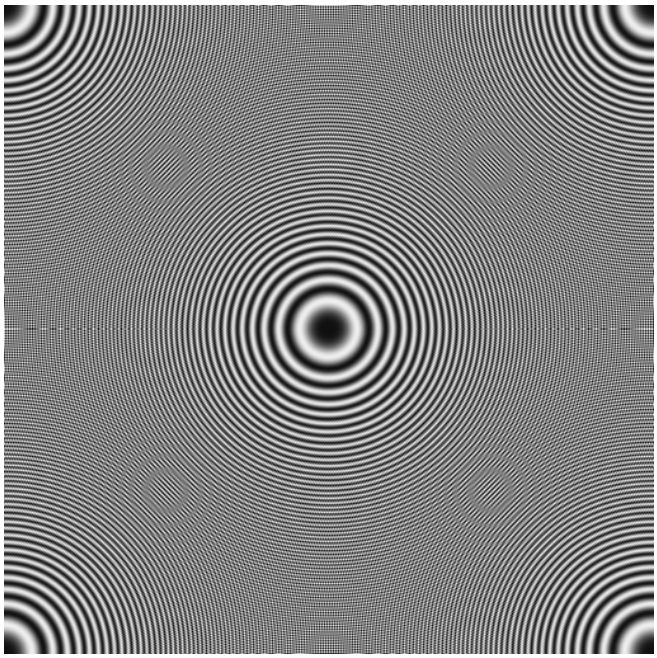
Now that we have determined that undersampling is at fault for the zoneplate image's four corner replicas, we note in particular that no replicas appear at the sides of the image; the four replicas cling to the four corners, suggesting that the algorithm sampled the image *diagonally*, along a line oriented $\pm 45°$ to the axes. Furthermore, the algorithm seems to have applied a 2:1 sampling ratio, since the replicas appear exactly at the edges of the image; the frequency variation seems to cycle through two periods, suggesting that the mystery algorithm compressed the frequency range between (discrete-time Fourier transform) spectral maxima by a factor of two before reconstructing the image. In other words, by taking only *half* the samples in the spatial domain, the algorithm essentially *halves* the spectral distance between adjacent replicas in the frequency domain, allowing two such periods to fit into the (512 × 512) domain frame. This perfect fitting can occur only if the sampling occurs along one of the 45° diagonals; otherwise, replicas would appear along other sides of the image.

In order to ascertain that the algorithm, indeed, performs diagonal downsampling by a factor of two, we implement the diagonal sampling algorithm in Matlab and operate on both images. Juxtaposing our manually processed images beside the given processed images, we ascertain their similarity:

Problem 2 - Processed Zoneplate Image



Problem 2 - Diagonally Sampled Zoneplate



Problem 2 - Processed Lena Image

Problem 2 - Diagonally Sampled Lena

For the samples that do not rest along one of the 2:1 sampling diagonals, our algorithm simply interpolates by computing the arithmetic mean of the four neighbors which do fall on a sampling line. The resultant image closely matches the given processed images; moreover, the mean square difference of the zoneplate images is approximately 1.077, while the mean square difference between the Lena images amounts to only 0.003. All in all, our diagonally downsampled linear interpolation must parallel the mystery algorithm.

Problem 3 – Eigenimages

Let f_i = i^{th} training image in training image set: $\{f_1, f_2, \dots, f_L\}$.

Let c_i = Haar coefficients of the i^{th} training image, organized in corresponding sets: $\{c_1, c_2, \dots, c_L\}$.

If we concatenate vectorized training images in matrix $F = [f_1 \ f_2 \ \cdots \ f_L]$, and likewise combine vectorized Haar coefficients in corresponding order in matrix $C = [c_1 \ c_2 \ \cdots \ c_L]$, then we can write the Haar coefficient matrix as a linear Haar transform of the training image matrix:

$$C = Hr \cdot F$$

Multiplying both sides of this equation by C^T, we obtain:

$$C^TC = (Hr \cdot F)^T \cdot (Hr \cdot F)$$

$$C^TC = F^T \cdot Hr^T \cdot Hr \cdot F$$

$$C^TC = F^T \cdot (Hr^T Hr) \cdot F$$

Because the Haar Transform is an *orthogonal* transformation with real coefficients, its inner product is identity: $Hr^T Hr = I$, allowing us to simplify our expression for the Gram matrix:

$$C^TC = F^T \cdot (I) \cdot F$$

$$\underline{\boldsymbol{C^TC = F^TF}}$$

Thus, the Haar coefficient images share an inner product with the training image set, meaning that the transformation on the training set conserves energy. As a result, because the two matrices are identical, they must also share the same set of L eigenvectors $\{\vec{v}_1, \vec{v}_2, \dots, \vec{v}_L\}$.

The Sirovich and Kirby method begins by considering a seemingly benign eigenvalue problem with regard to the Gram matrices:

$$F^TF \cdot \vec{v}_k = \lambda_k \vec{v}_k$$

Pre-multiplying both sides of this eigenvalue problem by the training set matrix F, we obtain

$$F \cdot F^TF \cdot \vec{v}_k = F \cdot \ \lambda_k \vec{v}_k$$

$$(FF^T) \cdot (F\vec{v}_k) = \lambda_k \ \cdot \ (F\vec{v}_k)$$

From this mere multiplication and re-association, we see that the Sirovich and Kirby eigenvalue problem now resembles the problem of interest:

$$(FF^T) \cdot \Phi_f = \Phi_f \cdot \Lambda$$

where (FF^T) approximates the autocorrelation matrix R_{ff}, the eigenmatrix Φ_f comprises the eigenimages of the training set, and the diagonal matrix Λ contains the eigenvalues along the main diagonal. Thus, we have implicitly solved the eigenvalue problem for the autocorrelation matrix (FF^T), as the eigenmatrix must contain L concatenated eigenimages of the form $(F\vec{v}_k)$, where $\vec{v}_k$ represents the k^{th} eigenvector of the much smaller matrix F^TF. In other words, our eigenimage matrix for training set F comprises the set of concatenated eigenvectors:

$$\Phi_f = [F\vec{v}_1 \;\; F\vec{v}_2 \;\; \cdots \;\; F\vec{v}_L].$$

Likewise, employing similar reasoning, we can reshape the eigenvalue problem for the Haar coefficient matrix initially considering

$$C^TC \cdot \vec{v}_k = \lambda_k \vec{v}_k$$

Here, we note that the eigenvectors $\vec{v}_k$ of the Gram matrix C^TC identically match the eigenvectors of the Gram matrix F^TF, because the two Gram matrices are equal – not only for the Haar Transform but also for *any* unitary transform. Proceeding with the Sirovich and Kirby method, we pre-multiply as before

$$C \cdot C^TC \cdot \vec{v}_k = C \cdot \lambda_k \vec{v}_k$$

$$(CC^T) \cdot (C\vec{v}_k) = \lambda_k \cdot (C\vec{v}_k)$$

We have once again reduced the Gram matrix eigenvalue problem into the problem of interest – the eigenvalue problem for the autocorrelation matrix $R_{cc} = CC^T$:

$$(CC^T) \cdot \Phi_c = \Phi_c \cdot \Lambda$$

which decomposes our set of coefficient images into a set of eigenimages encapsulated in the eigenmatrix:

$$\Phi_c = [C\vec{v}_1 \;\; C\vec{v}_2 \;\; \cdots \;\; C\vec{v}_L].$$

However, recall that we obtain the coefficient sets C by applying the Haar Transform to the training set F:

$$C = Hr \cdot F$$

Substitution into the coefficient eigenmatrix yields

$$\Phi_c = [(Hr \cdot F\vec{v}_1)\ \ (Hr \cdot F\vec{v}_2)\ \cdots\ \ (Hr \cdot F\vec{v}_L)].$$

Thus, the Sirovich and Kirby method reveals that the Eigen-Haar Transform coefficients obtained from solving the eigenvalue problem for the coefficient set $\{(CC^T) \cdot \Phi_c = \Phi_c \cdot \Lambda\}$ are linearly related to the eigenimages flowing forth from the eigenvalue problem on the training set $\{(FF^T) \cdot \Phi_f = \Phi_f \cdot \Lambda\}$. The Eigen-Haar Transform coefficients – the eigenimages resulting from the Haar coefficients set – is simply the Haar Transform of the eigenimages resulting from the training set:

$$\Phi_c = [(Hr \cdot F\vec{v}_1)\ \ (Hr \cdot F\vec{v}_2)\ \cdots\ \ (Hr \cdot F\vec{v}_L)].$$

$$\Phi_c = Hr \cdot [F\vec{v}_1\ \ F\vec{v}_2\ \cdots\ \ F\vec{v}_L].$$

$$\boxed{\boldsymbol{\Phi_c = Hr \cdot \Phi_f}}$$

Eigen-Haar Transform Coefficients = Haar Transform of Eigenimages

Furthermore, because we have made no assumptions on the nature of the Haar Transform past its unitarity, this relation applies to *any unitary transform* – not merely the Haar Transform:

$$\boxed{\boldsymbol{\Phi_c = H \cdot \Phi_f}\ \textbf{for}\ \underline{\textbf{\textit{any}}}\ \textbf{unitary}\ \boldsymbol{H}.}$$

Therein lies the power of the unitary transform: the eigenmatrix of the transform coefficients is simply the transform of the eigenimage matrix.