Abstract
This paper aims to develop a highly novel framework to systematically trade-off computational complexity with output distortion, in linear multimedia transforms, in an optimal manner. The problem is important since in practical real-time systems, the computational resources available to the algorithm changes with time. The problem is difficult since the output distortion is signal dependent making dynamic transform adaptation in a systematic manner, hard. We solve the problem by developing an approximate transform framework. There are three key contributions of this paper – (a) a polynomial approximation framework that decouples the system from the particular signal, (b) existence results and estimates for a convex complexity-distortion curve and (c) dynamic adaptation of the algorithm to changes in computational resources. We have applied this approach on the FFT, since it is widely used in multimedia and fundamental to many algorithms. We have obtained a 3~5 times speed-up on the FFT with excellent results.

1. Introduction
In this paper we present a highly novel framework for developing linear transforms that adapt to changing computational resources. The problem is important since in multimedia systems, the computational resources available to content analysis algorithms are not fixed, and a generic computationally scalable framework for content analysis algorithms is needed.

For example in Figure 1, we show an example of a system shows computational resources are changing over time. However, for a fixed analysis transform (e.g. FFT / DCT) there will be a time between \( t_1 \) and \( t_2 \) when the transform cannot operate at all. We seek an adaptive transform that is able to gracefully adapt to the resources available, but with greater error.

There has been prior work on adaptation in multimedia. There has been work on content adaptive transcoding [4,9] that focuses on adapting the media stream. Other complexity-scalable work includes [2,5] that looks at graceful degradation of video decoders under resource constraints. In more theoretical work [6] the authors look at properties of approximate transform formalisms and in [8] relationship between Kolmogorov complexity and distortion. However several issues remain – (a) while there has been some success in complexity scalable decoders, there are no formal generic adaptation strategies to guide us for other content analysis applications, (b) given a specific transform (say DCT) approximation and distortion, there is no framework that enables us to estimate the gap between resources for our approximation and an optimal solution (in terms of minimizing resources).

This paper attempts to create a framework for resource-adaptive multimedia content analysis. The problem is made difficult since the relationship between computational resources and distortion depends on the specific content.

We attempt to solve the problem in the following way. We focus our attention on the FFT, since this is one the most widely used transforms in multimedia content analysis. We effectively decouple the signal from the transform, by computing efficient polynomial basis approximations to the input signal. Then, we use pre-computed Fourier transforms on the polynomial basis to speed the computation. Then, we derive a theoretical relationship between computational complexity and distortion using the idea of a basis approximation. Using the basis complexity distortion function, we derive an algorithms that shows how we can adapt to changing computational resources. Our experimental results indicate a convex, complexity-distortion relationship and our preliminary results indicate that our adaptation technique performs well.

The rest this paper is organized as follows. In the next section we analyze the FFT complexity. In section 3 we show how to compute an approximate FFT, and section 4, we present the complexity distortion framework. In section 5 we show our experimental results and we present our conclusions in section 6.

2. Complexity Analysis of FFT
In this section, we analyze the complexity of 2D FFT for an 8*8 image block in which the components are real numbers. In this paper we shall assume that a single real addition, subtraction, or multiplication use equivalent resources.

We first process the columns of the input block, and then process the rows of the result. Since the components in the input block are real numbers, it requires 38*8 operations to process...
FFT for 8 columns. The result block has two rows of real numbers and 6 rows of complex numbers. Thus, the operation number of FFT for 8 rows of the results is 64*6+38*2. Combining the operations of FFT for columns and rows, the 2D FFT for an 8*8 real block requires 764 operations.

It can be easily shown that the traditional DFT takes 1524 operations. Compared with DFT, FFT significantly reduces the amount of computation. However, the computational complexity of the FFT is fixed. This becomes an issue in real-time systems where the availability of computing resources can change unpredictably.

### 3. FFT Approximation

In this section, we introduce an FFT approximation algorithm based on polynomial basis function and analyze the complexity.

#### 3.1 Complexity-Distortion signal dependence

It is very difficult to determine the relationship between complexity and distortion of FFT approximation because it depends on the input signal \( X \). Different input signals will have different distortions for the same approximation transform complexity. In order to break signal dependence, we use a set of polynomial basis functions to approximate input signals. Figure 2 shows the architecture of our FFT approximation algorithm. Firstly input signal is approximated with basis functions. Next, FFT is carried out on these basis functions.

**Figure 2. Diagram of FFT Approximation**

#### 3.2 Polynomial Basis for signal independence

We break signal dependence by computing polynomial basis functions to approximate the input signal. The 2D separable polynomial basis approximation for an 8*8 block can be represented as:

\[
X_k(u,v) = a_0 + \sum_{i=1}^{k} a_i u^i + \sum_{j=1}^{k} a_{n+j} v^j \quad 0 \leq u, v < 8, \quad <1>
\]

where \( X_k(u,v) \) is the approximation of pixel \((u,v)\) using polynomial degree \( k \) function, \( a_i \) are coefficients. In order to use the polynomial approximation, we have to estimate the coefficients of the approximating polynomial. Clearly the computational complexity of using the polynomial approximation increases with increase in the polynomial degree. Therefore, we limit polynomial basis functions to degree 0, 1, 2 which correspond to DC, linear and quadratic approximation model respectively (Table 1).

**Table 1. Polynomial basis functions**

<table>
<thead>
<tr>
<th>Polynomial Degree</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (DC)</td>
<td>( a_0 )</td>
</tr>
<tr>
<td>1 (Linear)</td>
<td>( a_0+a_1u+a_2v )</td>
</tr>
<tr>
<td>2 (Quadratic)</td>
<td>( a_0+a_1u+a_1u^2+a_2v+a_3v^2 )</td>
</tr>
</tbody>
</table>

#### 3.2.1 Basis Parameter Estimation

We apply least square error (LSE) estimation algorithm[7] to estimate parameters that minimize the LSE between the original \( x(u,v) \) and approximation \( x'(u,v) \). The solution (8*8 block) is:

\[
a = A \beta, \quad <2>
\]

where \( a \) is vector expression of \( a_0 \), \( A \) is a \((2k+1)*(2k+1)\) coefficient matrix for polynomial degree \( k \), \( \beta \) is vector expression \( \beta_i \):

\[
\beta_i = \begin{cases}
\sum_{u=0}^{7} \sum_{v=0}^{7} x(u,v) & i \leq k \\
\sum_{v=0}^{7} (v-k) \sum_{u=0}^{7} x(u,v) & i > k
\end{cases}, \quad <3>
\]

where \( k \) is the degree of polynomial basis function using for approximation. Figure 3 shows the coefficient matrix \( A \) for DC, linear and quadratic basis parameter estimations.

**Figure 3: Matrix A for polynomial basis approximation**

For DC approximation, computing \( \beta_0 \) and \( a_0 \) requires 63 and 64 operations respectively. For linear and quadratic approximations, the parameter estimation includes two steps. Firstly, we compute the summations of each row and last 7 columns and save these 15 summations. The complexity is 105 additions. Next, we use \(<3>\) and \(<2>\) to compute \( \beta \) and \( a \). Here we use complexity vector \( C(\beta) \) to denote the complexity of vector \( \beta \). Each component \( C(\beta_i) \) of complexity vector \( C(\beta) \) is the operation number of computing \( \beta_i \). The complexity vectors \( C(\beta) \) for linear and quadratic models are \([7,12,12]^T \) and \([7,12,12,12]^T \). Thus, the complexity of basis parameter estimation for linear and quadratic models is:

\[
C_k = 105 + \sum_{i=0}^{3} C(\beta_i) \quad k = 1, 2, \quad <4>
\]

where \( k \) is the polynomial degree number. Table 2 shows the complexity of parameter estimation for an 8*8 block using polynomial basis function with degree 0, 1, 2.

**Table 2. Complexity of basis parameter estimation**

<table>
<thead>
<tr>
<th>Complexity</th>
<th>DC</th>
<th>Linear</th>
<th>Quadratic</th>
</tr>
</thead>
<tbody>
<tr>
<td>64 operations</td>
<td>145 operations</td>
<td>184 operations</td>
<td></td>
</tr>
</tbody>
</table>
where \([x'(u,v)]\) is the matrix expression of \(x'(u,v)\) defined in <1>, \(F_{8 \times 8}(\theta_{1}, \theta_{2})\) is the FFT matrix for input \([x'(u,v)]\), [1] is an \(8 \times 8\) constant matrix in which all components equal 1, \([u']\), the transpose of \([v']\), is an \(8 \times 8\) matrix in which all components in the \(n\)th row equal \(n\).

Figure 4. FFT matrix form for \([1], [u'], [v'], [x'(u,v)]\)

The forms of FFT matrices \(F_{8 \times 8}(\theta_{1}, \theta_{2})\), \(F_{8 \times 8}([u'])\), \(F_{8 \times 8}([v'])\), \(F_{8 \times 8}([x'(u,v)])\) are shown in Figure 4. Because the polynomial basis approximation is separable, the non-zero components of result FFT matrix converge into the first row and the first column. The complexity of \(a_{0}F_{8 \times 8}(\theta_{1}, \theta_{2})\) is zero since the value of the only non-zero component equals \(b_{0}\) \((b_{0}=64a_{0})\) which is obtained. The complexity of \(a_{0}F_{8 \times 8}([u'])\) \((a_{0}F_{8 \times 8}([v']))\) is 4 operations because there are only 4 different coefficients in the first column of \(F_{8 \times 8}([u])\). In the similar way, the complexity of \(a_{0}F_{8 \times 8}([v'])\) \((a_{0}F_{8 \times 8}([v']))\) is 8 operations. Therefore, we obtain the complexities of FFT of basis functions shown in Table 3.

Table 3. Complexity of FFT of basis function

<table>
<thead>
<tr>
<th>DC</th>
<th>Linear</th>
<th>Quadratic</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 operations</td>
<td>10 operations</td>
<td>42 operations</td>
</tr>
</tbody>
</table>

3.4 Complexity of FFT Approximation

The computational complexity of FFT approximation using polynomial basis includes two parts: (a) Complexity of basis parameter estimation (shown in Table 2), (b) FFT computation for basis functions (shown in Table 3). Combining these two parts, we obtained the complexity of FFT approximation.

Table 4. Complexity of FFT approximation

<table>
<thead>
<tr>
<th>DC</th>
<th>Linear</th>
<th>Quadratic</th>
<th>Exact FFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>63</td>
<td>155 operations</td>
<td>226 operations</td>
<td>764 operations</td>
</tr>
</tbody>
</table>

As we discussed in section 3.3, since the value of the only non-zero component in \(a_{0}F_{8 \times 8}(\theta_{1}, \theta_{2})\) equals \(b_{0}\) \((b_{0}=64a_{0})\), we can use \(b_{0}\) to construct FFT matrix for DC approximation without the need to estimate \(a_{0}\). Hence, the complexity of FFT approximation using DC function is one operation less than the complexity of parameter estimation. It is observed that FFT approximation based on polynomial basis functions greatly reduces computation complexity compared with the exact FFT.

4. Complexity-Distortion Curve

In this section, we will do two things (a) establish a theoretical framework for the C-D curve, and (b) obtaining a useful estimate of the C-D curve using basis set approximations.

4.1 Establishing a theoretical framework

We shall use the well established definitions from rate distortion theory [3], to define the relationship between complexity and distortion. Then we shall obtain the C-D curve for the of FFT approximation for images. It is easy to see that these results will generalize to arbitrary linear transforms.

Let \(X\) be an random variable defined on a class \(F\) (e.g. faces) of size \(N\times N\) images. The complexity \(C\) of the FFT approximation of an image is defined as the average number of operations per \(N\times N\) block. We then define the distortion \(D\) due to the transform approximation as follows:

\[
D = E\left(d(X(k_1,k_2),X_{a}(k_1,k_2))\right),
\]

where \(X(k_1,k_2)\) is exact FFT matrix of block \((k_1,k_2)\), \(X_{a}(k_1,k_2)\) is approximation FFT matrix, \(d\) is a distortion measure, and where \(E\) is the expectation operator. The complexity distortion region \(R\) for an image for the FFT transform is defined as the closure of the set of achievable complexity distortion pairs \((C,D)\). The definition for an image class \(F\) is then the infimum of complexities \(C\) such that \((C,D)\) is in the achievable complexity distortion region for some image belonging to the class, for a given distortion \(D\).

The complexity distortion function \(C(D)\) for each image, for the FFT is defined as the infimum of complexities \(C\) such that \((C,D)\) is in the achievable complexity distortion region of the image for a given distortion \(D\). Since \(C(D)\) is the minimum complexity over increasingly larger sets as \(D\) increases, \(C(D)\) is non-increasing in \(D\). We can similarly define a \(C(D)\) function for each image class as well.

4.2 Basis set approximations

The \(C(D)\) lower bounds described above are theoretical and in practice, it will be very difficult to reach the lower bound. However, we can use a basis set approximation to come close to the theoretical lower bound – Basis Set Complexity Distortion Function (BSCDF). A basis set is the approximating basis function set used for the FFT approximation. The complexity distortion function for basis set \(b_i\) is defined as:

\[
C_{a}(D) = \inf_{b_i < D} (C_{b_i}),
\]

\[
D_{b_i} = E\left(d(X(k_1,k_2),X_{b_i}(k_1,k_2))\right),
\]

where \(X_{b_i}(k_1,k_2)\) is approximation FFT matrix using basis \(b_i\). We define the Basis Complexity Distortion Function as the infimum of complexity distortion functions of all possible basis sets:

\[
C_{a}^{\ast}(D) = \inf_{b_i < D} (C_{b_i}(D)),
\]

where \(B\) is the set of basis sets i.e. it contains all possible basis sets within it e.g. polynomial, Gaussian, Haar etc. We conjecture that basis complexity distortion function is equal to the theoretical complexity distortion function for the FFT.

4.3 Adjustment of Operating Point

In this section we show to adjust to dynamically changing resource conditions. We shall assume that as part of the image
metadata, we are provided with the class BCDF. We shall also assume that the image metadata contains the approximating error \( e_{0i}, e_{1i}, e_{2i} \) for each block with polynomial degree 0,1,2 approximation (from encoder) with regard to the polynomial basis set. We assume that the image is currently operating at an initial operating point \((C_0,D_0)\). The goal is to adjust the operating point optimally (i.e. close to the BSCDF), when \( C_i < C_0 \), where \( C_i \) is the target complexity that is slightly less than \( C_0 \). The fast algorithm for downward adjusting is described as follows:

- Initialize: compute \( e_{0i}, e_{1i} \) for blocks using linear and quadratic approximation and \( e_{1i}, e_{2i} \) for blocks using quadratic approximation and sort \( e_{0i}, e_{1i}, e_{2i} \) to generate two ascending lists.

- If the smallest component in \( e_{0i}, e_{1i}, e_{2i} \) list is less than the smallest in \( e_{1i}, e_{2i} \) list, change approximation function of the block corresponding to the smallest component in \( e_{0i}, e_{1i}, e_{2i} \) list from linear to DC and remove this smallest component from \( e_{0i}, e_{1i}, e_{2i} \) list. If the smallest component in \( e_{0i}, e_{1i}, e_{2i} \) list is larger than the smallest in \( e_{1i}, e_{2i} \) list, change approximation function of the block corresponding to the smallest component in \( e_{1i}, e_{2i} \) list from quadratic to linear, insert the \( e_{0i}, e_{1i}, e_{2i} \) value of this block into an appropriate position of \( e_{0i}, e_{1i}, e_{2i} \) list and remove this smallest component from \( e_{1i}, e_{2i} \) list.

- Update complexity \( C \) and distortion \( D \). If result \( C \) is larger than complexity constraint, go to step 2, else stop adjusting and the desired \((C,D)\) is achieved.

In a similar way, we can adjust the operating point upward. More details can be found in [1]

5. Experimental Results

We now present our experimental results. In this section we shall show how we computed the BSCDF for a polynomial basis, and present results with respect to the class of face images. We now show how to derive the polynomial BSCDF. For an image divided into M*N blocks, 8*8 pixels each, an alternative definition of complexity distortion function using polynomial basis set. We shall show how we computed the BSCDF for a polynomial basis, and present results with respect to the class of face images. It shows that images with similar texture have similar polynomial basis \( C(D) \) functions which guarantee that we can use one polynomial basis \( C(D) \) function for similar images.

6. Conclusion

In this paper, we have attempted to create a systematic framework for trading off computational complexity with distortion. There were three key ideas – (a) using basis functions to approximate the input, we broke the input signal dependence problem, (b) we showed the existence of a convex complexity-distortion curve, and (c) we showed a dynamic adaptation algorithm. In the future, we are planning to investigate other basis sets that give a more fine grained complexity approximations, and investigate \( C(D) \) curves for more classes.

7. References


