Interactive Image Processing: Discrete Cosine Transformation

Peter Plaimer
Johannes Kepler University Linz
dct-tool@tk.jku.at

January 24, 2018

Abstract

This paper looks into methods of teaching Discrete Cosine Transformation (DCT) in the context of images. DCT is a mathematical transformation used in lossy compression of images and video. As such, it is a topic addressed in university curricula for computer science. While it is obvious to approach teaching DCT from a mathematical point of view, this approach is not well-suited for all students alike. We propose a software application that supports students in learning DCT by offering an alternative, visual and interactive means to explore forward an inverse DCT on arbitrary images. We further provide exercises and concrete questions the students can work on, in order to deepen their understanding of DCT based on example images built into that software.

Contents

1 Introduction

Many contemporary and everyday multimedia systems make use of Discrete Cosine Transformation (DCT) at their very core. DCT [3]is the basis for lossy compression of JPEG images [2] and MPEG videos. The undergraduate curriculum for computer science at the Johannes Kepler University Linz [1] reflects the relevance of this technique in the respective field. The curriculum comprises multiple classes where Discrete Cosine Transformation (DCT) is an important topic, for example, lectures and exercises on multimedia systems and a course on digital image processing.

From our first hand experience we know that the level of mathematical skills of students in these classes varies significantly, and so does their success in understanding DCT. Some colleagues struggle to get their heads around the role of coefficients in the frequency domain matrix and their role in the image domain matrix or vice-versa. Informal interviews suggest that most of these students are seeking a visual connection between the coefficients in the two domains. They express the desire to see such connections especially when DCT is applied to blocks of "just" 8 x 8 pixels during JPEG compression and decompression.

We plan to contribute teaching material for classroom training, exercise assignments, and self-study, which aids in understanding DCT in a visual way. Our approach consists of two major parts. The first one is a software application that shows DCT in the context of image processing and JPEG. It offers an image representation of the spatial domain matrix and the frequency domain matrix. Its user interface allows for interactive manipulation of any coefficients in each matrix, respectively the images, using a brush metaphor controlled by the mouse pointer and other means. Changes in one domain trigger instant execution of the DCT, providing instant visual feedback in the other domain's image. The second part is this paper which briefly introduces the topic, explains the application and its user interface, and proposes exercises and questions for deepening the understanding of DCT. TODO: the java sources for the application and the latex/lyx sources paper, as well as PDF version of the paper, shall be publicly available on github, ok?

The remainder of this paper offers a short recap of the definition of the DCT and its application in JPEG in section 2. In section 3 we present the software application in the style of a user manual. That section moves on with a list of exercises we propise. It ends with an overview of the application's components and technical requirements. Section 4 evaluates the application based on the planned contribution. We use section 5 to propose topics for future work based on the application or the general theme of visualizing interactive image processing demos. Section 6 concludes our work.

2 Related work

TODO: was is das umfeld, "wir beschäftigen uns mit dct in bildern". referenzen sind bücher und papers. über jede refernz schreibat man was für das eigene thema drin intressant ist. unterkapitel thematisch bauen, zb bildkompression überhaupt verlustftei/verlustbehaftet, verwendung in konkreten anwendungen. und was no cooler ist, zb fast forward dct (firma bioelektromech).

TODO: section intro

2.1 DCT

TODO: talk about DCT paper[3], how DCT was defined there and how image compression is never mentioned there ;)

2.2 JPEG

TODO: talk about JPEG standard [2], main topic is annex A.3 DCT. grab image of encoding/decoding process in order to put our application into the right context.

3 Interactive Software Application

TODO: dann kapitel "was hab i jetzt eigenclih gmacht" mit schönen titel. kurz komponenten bzw elemte aufzähle nud wie sie zsammarbeiten, zb maven, opencv artifact, jtable genau das aktualiesierne was pinseln geändert.

This section contains the user manual and gives examples for questions to be answered by students. It furthermore provides an architecture overview of the application and details for building the application from source.

3.1 User Manual

This section describes the elements of the user interface, explains their effects, and outlines relevant user cases. The user interface comprises multiple windows of different types. The UI overview in figure 1 depicts a typical situation during DCT exploration and serves as the reference for the following description.

The top left window contains all settings for the brush, the tool for interactive manipulation of the matrices via image windows. The other two windows in the top row are image windows, each one showing an image representation of a matrix. The left window depicts the spatial domain matrix, and right one the frequency domain matrix. The bottom row has two table windows. A table window shows the components of a matrix as table, and allows value manipulation via standard means of input. Their matrices correspond to those in the image windows above.

Any manipulation of a matrix value via an image window or via a table window will instantly affect the other matrix, and reflect in all windows of all matrices. Manipulation of the spatial domain matrix triggers a forward DCT (FDCT) and changes the frequency domain matrix. Manipulation of the frequency domain matrix triggers an inverse DCT (IDCT) and changes the spatial do main matrix.

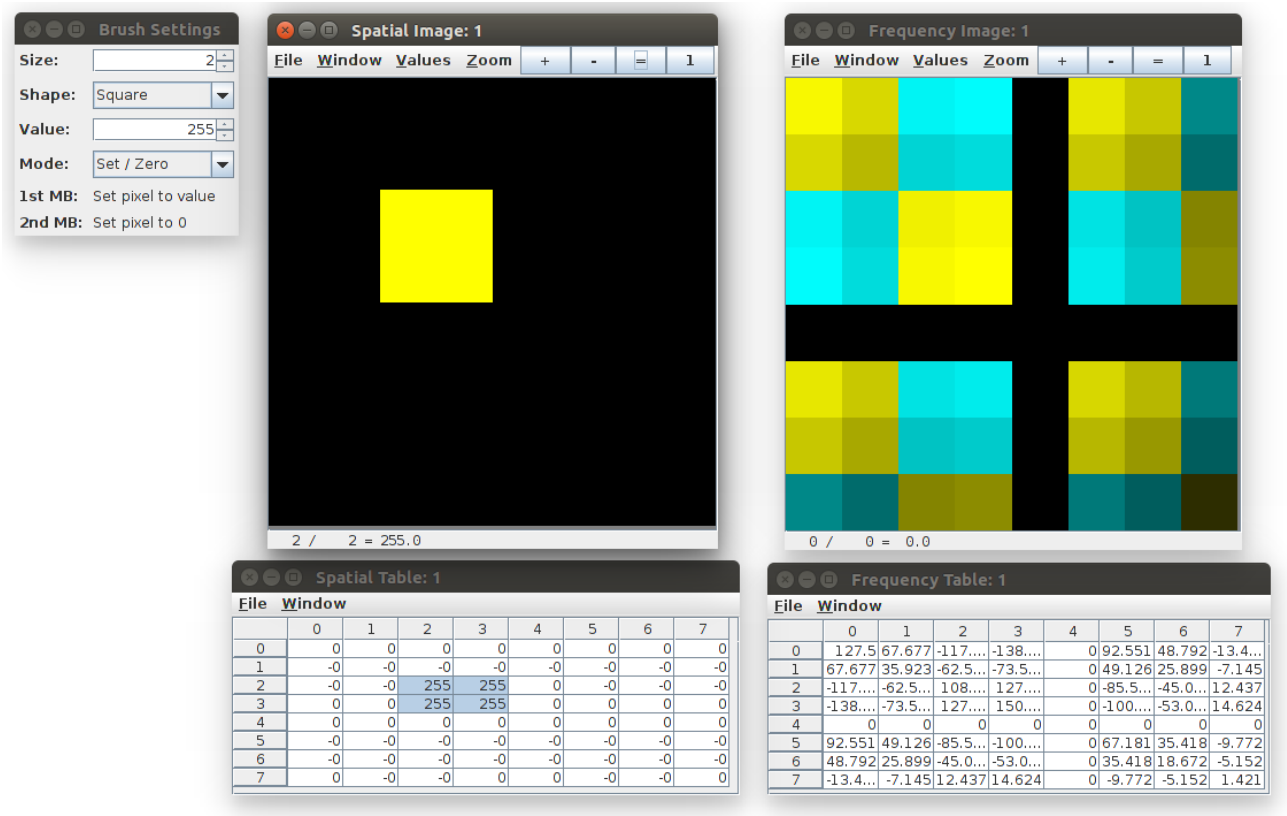

Figure 1: This UI overview is an example of a typical situation during DCT exploration.

3.1.1 Image Window

Each image window provides the same functions, regardless of the matrix it is displaying. The window uses a viewport showing all or a part of the image representation of a matrix, see section 3.1.2 for details. In this representation, one pixel corresponds to one component of the matrix.

The window features a set of buttons [+], [-], [=], and [1] to control the magnification of the image, in a manner similar to image viewer applications. Spinning the mouse wheel has the effect of zooming in and out, too. The application provides scrollbars to navigate the image, if the magnified view of the image does not fit into the space inside the window. The status bar shows the value of the matrix component beneath the mouse pointer.

Each image window features a *File Menu*, which operates on the respective matrix and its visualization. The most important menu items from top to bottom:

New Matrix creates a new zero matrix. A dialog will query the size. As a consequence of the nature of DCT, this will also create the other matrix anew, and all components will be zero as well.

Open Example Image >> offers a sub menu containing a list of built-in example images for easy opening, with the same behavior like the next menu item.

Open Image File ... opens an image file from the file system via the standard file dialog. Colored images will automatically convert to grayscale. The image window scales the matrix visualization to fit into the window.

Re-Open Last Image quickly opens the last image opened from the examples list or the file system, discarding all changes to the matrix. This is a very basic way of undoing changes.

Save Matrix As CSV ... saves the matrix components into a comma-separated-values file for external processing.

Save Visible Image As PNG ... saves the whole matrix visualization image to the file system, ignoring the image window viewport. The image file is always of type PNG, no matter what extension the file name features.

Copy Visible Image To Clipboard copies the whole matrix visualization image into the system clipboard, ignoring the image window viewport.

The window's menu bar features a Window Menu, containing items to create further image windows and table windows.

3.1.2 Matrix Visualization

DCT is a transformation on a matrix, yielding another matrix. In order to create the image representation of a matrix, the application creates an image of the same size in pixels as the matrix has components. One image pixel corresponds to one matrix component in the same row and column as the pixel.

Each component of both matrices is a real number, thus it has decimal digits and a sign, which may be negative. We can not just use the component value for the brightness of a pixel, since contemporary computer displays can not show negative brightness, and positive brightness is limited to integer values from 0 to 255 inclusive.

The application offers multiple strategies to overcome this crucial limitation, amongst which the user may choose one in each image window's *Value Menu*.

Original mat clipped grey implements simple clamping of values in a grayscale image. Negative component values show as brightness 0 in the pixel, values of 255 and above will have brightness 255.

Auto contrast/brightness grey will uniformly and linearly scale the matrix values such that their resulting brightness exactly fits into the range 0 to 255, if any component exceeds these values. In image processing terms, this roughly corresponds to an automatic contrast and brightness correction.

Log(1+abs(v)) to grey is based on TODO:REF-tönnies2005?, which recommends a conversion to a grayscale image using a logarithmic scale. In order to avoid the impossible calculation of the logarithm of values less than one, he introduces the addition of 1 before the log operation. This way, the intuitive mapping of component value 0 to pixel brightness 0 is still intact. The actual implementation employs automatic contrast adaption after that, in order to use the maximum brightness value 255.

Log(1+v) to +yellow/-cyan is our extension of the aforementioned strategy, additionally encoding the sign of the component value in the color of the pixel. See figure 1 for an example. Positive component values use yellow color, negative values use cyan, zero is black still. TODO: the following exmaplanation into a separate section, maybe technical stuff?

We choose these colors with contemporary RGB displays and human perception in mind, and we aim for high brightness and contrast, while still retaining good color separation for positive and negative signs. The green color component of pixels in these displays is the color with the best reception and resolution in the human eye TODO:REF_something, yielding high brightness and contrast. We use green for both signs in order to leverage these favorable attributes in all cases. With the same brightness, we add either red or blue, depending on the sign. This has multiple advantages. We further gain brightness and contrast, vital for visual value estimation. We leave a minimum of available brightness and contrast capability of the display unused. We achieve high color separation, due to red and blue being on opposite sides of green in the color spectrum. We add no crosstalk, because we use a separate display subpixel per sign.

3.1.3 Brush Window

Interactive manipulation of a matrix via its image representation inside an image window uses the brush metaphor. Pressing the primary mouse button (PMB) while the mouse pointer is inside the image viewport will modify the matrix by applying the brush's primary function, the secondary mouse button (SMB) applies the brush's secondary function, respectively. Dragging the mouse pointer while pressing a mouse button will repeat the function once for every matrix component entered with the mouse pointer, thus enabling a natural way of painting on the matrix.

The Brush Window allows for configuration of a number of brush attributes:

Size determines the size of the brush. Its exact semantics depend on the brush shape.

Shape chooses a shape of the brush and indirectly the strength of its impact on the components affected. The available shapes are:

- **Square** is a filled square with hard edges with length *size*. The mouse pointer determines the center component of the square.
- **Hard Circle** is a disk of diameter *size* with a hard edge. The mouse pointer determines the center component of the disk. The brush function affects all components covered by the brush equally.
- **Soft Circle** is a disk of diameter *size* with a soft edge. The mouse pointer determines the center component of the disk. The brush function fully affects the center component. The strength of the effect on other components decreases with their distance from the center of the brush. It is minimal for components on the edge of the disk. TODO: explain this particular application of alpha blending in a separate section?

Value determines the strength of the brush function. Can be negative. Its exact semantics depend on the brush mode.

Mode chooses the pair of functions the brush applies to the matrix:

- **Set Value** sets the affected components value to equal *brush value* when pressing the PMB. The SMB sets the components value 0.
- **Add Value** adds *brush value* to the components value when applied with the PMB, and the SMB subtracts the *brush value* from the component value.

Multiply % interprets *brush value* as percentage and applies the resulting number to the component value using a multiplication for the PMB and a division for the SMB. For example, value 125 represents 125 percent, which equals the number 1.25 in decimal notation. I will therefore increase the component value by one fourth when using the PMB and reduce it by one fifth when applied by the SMB.

3.1.4 Table Window

A table window shows the matrix and its component values in a table. Standard UI interaction allows for changing single component values via keyboard. When drawing the brush in an image window, table windows of the same matrix change their cell selection such that it highlights the components affected by the brush. A table window's menu bar offers the File Menu and the Window Menu as described in section 3.1.1. minus the image export functions.

3.2 Exercises

This section contains a list of topics investigate and proposes exercises and questions suitable for classroom demonstration or deepening self-studies. For each topic we introduce the context, give instructions to reconstruct the situation under discussion, and we pose questions in order to excite the students' curiosity. The instructions often refer to the Spatial Domain (SD), Spatial Domain Image (SDI), Frequency Domain (FD), and the Frequency Domain Image (FDI), using the respective abbreviations.

TODO: move this subsection into section 4 - evaluation?

TODO: add one or two small pictures of the respective SDI/FDI to each exercise?

3.2.1 Identity Matrix

The identity matrix clearly shows the matrix transformation nature of the DCT. When loaded into the SD or the FD, the resulting matrix is always the identity matrix.

Exercise: Load the example image identity-8 into either one matrix.

Additional Questions: Are there any other simple matrices showing visual similarities in the SDI and FDI?

3.2.2 JPEG DCT

JPEG compression feeds 8 by 8 blocks of pixels into an FDCT to obtain the FD matrix. Similarly, the decompression feeds an 8 by 8 matrix into its IDCT in order to reconstruct the SDI. Visual explanations of these step often use a diagram showing the SDI pattern every single FD component adds to the image TODO:REF_english_wikipedia_DCT_or_better?. This application allows to reproduce the elements of these diagrams interactively. Furthermore, one can study the superposition of the patterns from multiple coefficients.

Exercise: Configure a size 1 square brush with value 255 in set-value mode. Set SDI visualization to *yellow/cyan*. Apply the PMB on single components of the FDI, use SMB to remove the value.

Additional Questions: What is the effect of FD component in row 0, column 0 on the SDI? How do FD components n/0 and 0/n relate to each other? How and why is the SDI for the combination of FDI n/0 and 0/n different to n/n alone?

3.2.3 JPEG Quantization Loss / Low-Pass Filter

JPEG compression is lossy due to the quantization step after the FDCT. This step basically discards high-frequency components with small absolute values, in other terms, it is a low-pass filter. When losing too much information, the decompressed image shows blurred edges, blocky colors, ringing, and other types of artifacts. We can simulate this loss of information in quantization and immediately see its effect in the SDI.

Exercise: In an SDI window, open example image lena-512 in an SDI window, *zoom to original size*, choose visualization *original*. Configure a size 80 square brush and choose mode *set/zero*. In an FDI set visualization yellow-cyan. In that FDI, start applying the brush in the bottom right corner. Press and hold SMB and start sweeping in a diagonal direction from bottom-left to top-right and back. While still sweeping, slowly advance to the top-left corner, deleting lower and lower frequencies. You are most likely to see no detrimental effect in the SDI up until the point where the bottom-right half of the FDI is cleared. Keep sweeping and watch closely as the first artifacts become visible.

Additional questions: What is similar or different to the effect seen before, if one starts clearing coefficients at the bottom edge of the FDI, sweeping from left to right and back, slowly moving to the top?

3.2.4 Edge detection / High-Pass Filter

Edges in SDI correspond to high amplitudes in high-frequency coefficients. One can use this property to find edges by manipulating the FDI mimicking a high-pass filter.

Exercise: In an SDI window, open example image lena-512 in an SDI window, choose visualization *auto-contrast/brightness*. Configure a size 240 hard circle brush and choose mode *set/zero*. In an FDI apply the brush by SMB at component 0/0.

Additional Questions: How does the brush size influence the kind of edges this method finds? When clearing the coefficient 0/0 only, how does it change the character of the filter operation?

3.2.5 Large-Scale Self-Similarity

Exercise: Create a new matrix of size 256 by 256. Configure brush size 32, *hard circle* shape, value 255, mode *add/subtract*. Choose *yellow/cyan* matrix visualization for SDI and FDI. Apply the brush in SDI at 128/128 using PMB. This produces a pattern of dozens of wiggly lines and loops in the FDI. Now apply the brush in the FDI at 128/128. Now SDI and FDI show striking similarities. Question: What are the major reasons for this result?

3.3 Technical description

TODO: rundown of components, tools and runtime requirements: Java 7, swing library and programming model, swing jtable extension for selection feedback, opencv 2.4.9 for dct (mention reason: digitla image processing class), maven fragment nu.pattern.opencv for opencv/java-bindings, Eclipse Neon 4.6, ubuntu 14.04 LTS, github https://github.com/jkutk/iaip-dct

4 Evaluation

TODO: dann das Evaluation als volles kapitel. ist es gut gegangen, was kann man damit jetzt wie gut tun. hat man ziele erreicht, gschickterweise in introduction was ziel ist und heir nur ja habs erreicht.

TODO: Exercises section goes here?

TODO: explanation of design choices for matrix visualization yellow/cyan goes here?

TODO: short story bout problems during implementation. zb opencv artifact und java.nio passe nicht zsamm, oder opencv libpng problem und wie gelöst.

TODO: any positive surprises along the whole process of designing, implementing, evaluation ...?

5 Future Work

TODO: nicht was mach i damit konkret sondern was kann sonstwer damit machen. aus unseer sicht feature complete, aber ideen. zb man macht das mit wavelets, oder audio, oder in übung statt opencv dct die selber implementierte aus übung 3 (?) verwednen.

TODO: propose application features: load csv file. swap SD and FD.

6 Conclusion

TODO: wie einleitung aber in vergangneheitsform aber kürzer und hinweis auf ergebnis.

TODO: fix bibtex generated bibliography. missing authors of webpage refs crash bst style alphadin. working styles e.g. acm do not print webpage urls and miss other info. which bst style is the best / suitable, ...? resort to writing the bibliography manually?

References

[1] CURRICULUM GUIDE BACHELOR IN COMPUTER SCIENCE. valid as of WS 2017/18.

[2] ISO/IEC 10918-1:1994 information technology – digital compression and coding of continuous-tone still images: Requirements and guidelines.

[3] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. C-23(1):90–93.