Middle East Technical University
Department of Computer Engineering

CENG 466

FUNDAMENTAL IMAGE PROCESSING TECHNIQUES

Spring 2004

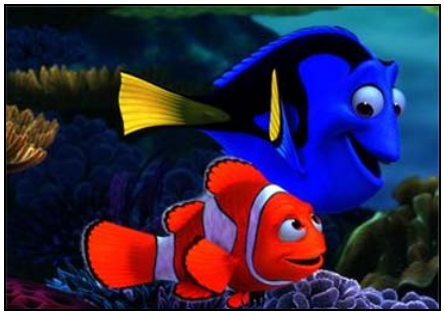
Homework 1

General Rules

- Due date is 5 November 2004, Friday, 17:00.
- In case of cheating, all parts involved (source(s) and receiver(s)) get zero.
- You will submit your homework electronically through Ceng Homework Submission System (https://submit.ceng.metu.edu.tr/) as usual. Additionally, you should submit a hard copy report to my office A-401. Your electronic submission must include only your implementation (hw1.c), you don't need to electronically submit input-output images or soft copy of your report.
- For each group, only one of the students should submit the homework electronically through CENG Homework Submission System.

Homework

1. Develop your own algorithm to quantize an image from 24 (8*3) bits to 12 (4*3) bits, and apply this algorithm to the following images:


In the report,

- Explain your algorithm,
- Show input images together with their output,
- Discuss on your results.

2. Extract the region adjacency graphs of the following images by using their quad-tree representations. You must pre-process (e.g. multithresholding segmentation) the images to obtain piecewise smooth versions before extracting their quad-trees.


In the report,

- Explain your algorithm,
- Show input images together with their output (smoothed versions and extracted adjacency graphs),
- Discuss on your results,

Specifications

For each of the questions in this homework, there is a set of common specifications which are listed below:

1. Input and output image file format is binary encoded Portable Pixel Map (**PPM**). Design your own, functions to read and write PPM files.

   PPM is the portable pixel map format. It is a simple RGB color image description. The definition is as follows:

   - A "magic number" for identifying the file type. For raw PPM files magic number is the two characters "P6".
   - Whitespace (blanks, TABs, etc.).
   - A width, formatted as ASCII characters in decimal.
   - Whitespace.
   - A height, again in ASCII decimal.
   - Whitespace.
   - The maximum color-component value, again in ASCII decimal.
   - Whitespace.
   - Width * height pixels (The pixel values [R,G,B] are stored as **plain bytes**).
   - Characters from a "#" to the next end-of-line are ignored (comments). No line should be longer than 70 characters.

2. All of the questions should necessarily be combined into a single program. The usage of the program for each question should be as follows:

```
hw1 Q1 <input image> <output image>
hw1 Q2 <input image> <output image>
```

3. Your programs will be compiled and run automatically.

4. Use the following template for your homework:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

//macro definitions
#define MAX_IMAGE_WIDTH 512
#define MAX_IMAGE_HEIGHT 512
#define NUMBER_OF_CHANNELS 3

// type definitions
typedef struct
{
	int width;
	int height;
	unsigned char
		pixels[MAX_IMAGE_WIDTH][MAX_IMAGE_HEIGHT][NUMBER_OF_CHANNELS];
} type_imageCO;

// function prototypes
void question1(char *image_filename, char *output_filename);
void question2(char *image_filename, char *output_filename);
void read_image(char *image_filename, type_imageCO *image);
void write_image(char *image_filename, type_imageCO *image);
void print_usage(void);

//--------------------------------------------------------------------
int main(int argc, char *argv[])
{
	try
	{
		if(strcmp(argv[1], "Q1") == 0)
			question1(argv[2], argv[3]);
		else if(strcmp(argv[1], "Q2") == 0)
			question2(argv[2], argv[3]);
	}
	catch(...)
	{
		print_usage();
		exit(1);
	}
	return 0;
}

//--------------------------------------------------------------------
void print_usage(void)
{
	fprintf(stdout, "USAGE:\n------\n");
	fprintf(stdout, "hw1 Q1 <input filename> <output filename>\n\n");
	fprintf(stdout, "hw1 Q2 <input filename> <output filename>\n\n");
}
```

```
//---------------------------------------------------------------------
void question1(char *image_filename, char *output_filename)
{
	type_imageCO input_image;
	type_imageCO output_image;

	// read the input image
	read_image(image_filename, &input_image);

	// INSERT your functions here, to solve the question
	// ...

	// write the output image
	write_image(output_filename, &output_image);
}


//---------------------------------------------------------------------
void question2(char *image_filename, char *output_filename)
{
	type_imageCO input_image;
	type_imageCO output_image;

	// read the input image
	read_image(image_filename, &input_image);

	// INSERT your functions here, to solve the question
	// ..

	// print out the region adjacency graph to standard output as an
	// adjacency matrix

	// write the smooth image
	write_image(output_filename, &output_image);
}


//---------------------------------------------------------------------
void read_image(char *image_filename, type_imageCO *image)
{
	// INSERT your code here, to read a binary encoded Portable Pixel
	// Map (PPM) from a file
}


//---------------------------------------------------------------------
void write_image(char *image_filename, type_imageCO *image)
{
	// INSERT your code here, to write a binary encoded Portable Pixel
	// Map (PPM) to a file.

}
```