Universidade do Porto

Faculdade de Engenharia

FEUP

Computação Paralela

Trabalho #1

Programa MPI para melhoramento iterativo de imagem

09 / 05 / 2010

João Cristóvão Xavier –060509116

José Nuno Cardoso – 060509113

Índice

Introdução

O trabalho é uma parte integrante do programa da unidade curricular de Computação Paralela, do 4º ano do Mestrado Integrado em Engenharia Informática e Computação na Faculdade de Engenharia da Universidade do Porto.

Para os trabalhos desta unidade curricular, é facultado o acesso a uma cluster com vários processadores (Grid @ FEUP), de forma a se poder aproveitar ao máximo as vantagens que a computação paralela pode oferecer no desenvolvimento de *software*.

O principal objectivo é, através da utilização da linguagem **C** / **C++** e de uma implementação do padrão **MPI** (Message Passing Interface), uma infraestrutura para facilitar a transferência de informações entre vários processadores, resolver um problema de melhoramento iterativo do contraste de uma imagem. Esta tarefa é geralmente demorada para imagens de grande dimensão, de modo que a aplicação de um algoritmo paralelo para um problema deste cariz é crucial para uma redução significativa do tempo de execução.

A imagem a ser melhorada é fornecida no formato **.ppm** (portable pixmap format), que se define no formato **Netpbm**, desenhado para ser facilmente portável entre diferentes plataformas.

De forma a ajustar o contraste de cada pixel, são facultadas no enunciado as fórmulas necessárias de cálculo e avaliação do novo valor que o pixel deve tomar em cada iteração com base nos pixels que o rodeiam.

Implementação

Como linguagem de programação, decidiu-se optar pelo uso do **C++** em deterimento do **C** para poder tirar partido de todas as funcionalidades que caracterizam uma linguagem orientada a objectos, entre outras facilidades implementadas na **STL** (Standard Template Library). Sempre que possível, aproveitou-se o uso de funções de baixo nível para optimização da velocidade de execução.

As implementações de **MPI** usadas foram a **HPC Pack 2008** e a **MPICH2**, a correr no Visual C++ 2008 Express Edition em Windows 7 e em Debian, respectivamente.

Apostou-se em separar o programa a nível de código, identificando as partes de leitura e escrita, algoritmo, macros relativas a acesso a informações da imagem e distribuição do trabalho pelos diferentes processos.

Como fase inicial do desenvolvimento do trabalho, implementou-se o algoritmo de uma forma sequencial, estabeleceu-se um sistema de *benchmarking* e optimizou-se os tempos de leitura e de escrita. Numa fase posterior, implementou-se o algoritmo para ser corrido com múltiplos processadores. Na primeira implementação paralela existia um processo que se limitava a distribuir o trabalho aos outros, não sendo muito eficiente. Numa fase final, procedeu-se a um melhoramento onde todos os processos compartilham o trabalho igualmente e ainda a um outro onde se minimiza o número de comunicações efectuadas entre processos.

A maioria dos testes foram efectuados numa máquina local, a correr num processador Intel(R) Core(TM)2 Duo CPU T7300 2.00GHz. Os restantes testes foram efectuados na cluster, cujos processadores são Intel(R) Xeon(TM) CPU 3.00GHz.

De forma a correr os testes na cluster do Grid, conceberam-se alguns scripts de execução, que podem ser encontrados no Anexo III juntamente com os resultados.

Análise de Resultados

Nesta secção é possível observar os tempos obtidos nos testes efectuados nas diferentes máquinas com um número de processadores diferente.

Todos os testes foram efectuados com 50 iterações e com o ficheiro de teste Harbor.ppm, que pode ser encontrado no Anexo II, assim como a imagem resultante da execução do programa.

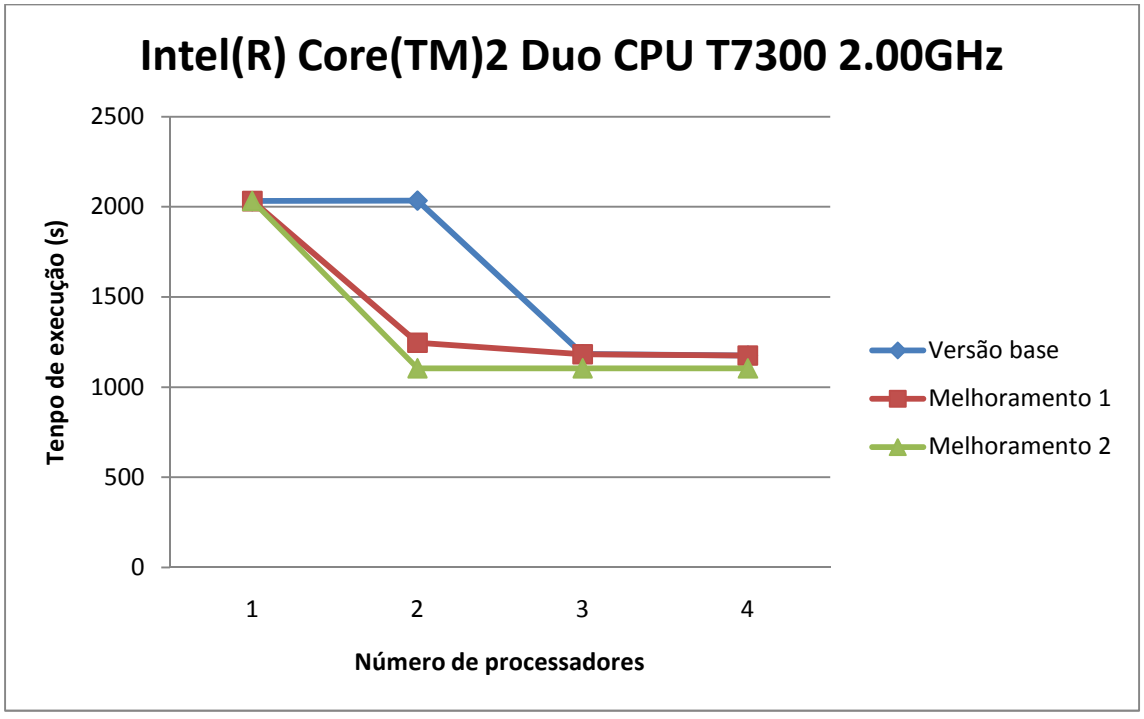
Melhoramento 1: Aproveitamento do processo distribuidor do trabalho para tratar a imagem.

Melhoramento 2: Minimização do número de comunicações.

Intel(R) Core(TM)2 Duo CPU T7300 2.00GHz

Leitura	7.85 s
Escrita	5.32 s

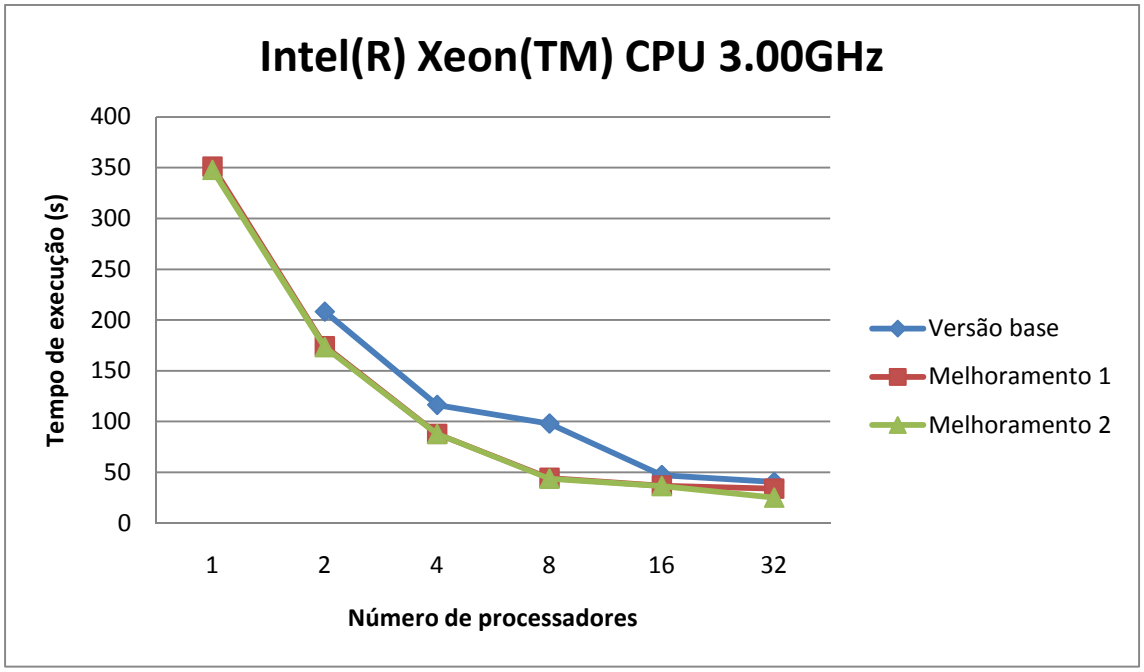
Nº processadores	Versão base	Melhoramento 1	Melhoramento 2
1	2031.79 s	2031.79 s	2031.79 s
2	2034.12 s	1245.64 s	1104.60 s
3	1183.21 s	1182.43 s	1104.43 s
4	1175.06 s	1174.43 s	1104.11 s



Intel(R) Xeon(TM) CPU 3.00GHz

Leitura	1.12 s
Escrita	2.33 s

Nº processadores	Versão base	Melhoramento 1	Melhoramento 2
1	-	351.03 s	347.95 s
2	208.15 s	174.15 s	173.34 s
4	116.35 s	87.82 s	87.75 s
8	97.97 s	44.5 s	43.79 s
16	47.29 s	36.89 s	36.49 s
32	40.47 s	33.94 s	25.13 s



Conclusão

Verificou-se que na máquina local com dois processadores todos os testes efectuados com mais de dois processos dão resultados semelhantes aos resultados obtidos nos testes com dois processos. Já na cluster, é possível denotar uma optimização do tempo de execução proporcional ao número de CPUs utilizados.

Existe uma redução significativa do tempo de execução desde a versão base para o primeiro melhoramento, associado ao facto de se aproveitar mais um processador para o tratamento da imagem. Comparando o segundo melhoramento com o primeiro, deduziu-se que de facto as comunicações inter-processos têm como custo algum *overhead*, tendo resultado numa optimização eficaz do tempo de execução.

O trabalho foi implementado com sucesso, sendo possível comprovar que utilizando um algoritmo paralelo o tempo de execução diminui em função do número de processadores que se usa, e quanto menos comunicações existir entre estes, mais rápido é o desempenho.

Bibliografia

- Quinn, Michael J. - Parallel Programming in C with MPI and OpenMP. New York, NY, United States of America: McGraw-Hill Science/Engineering/Math, 2003.
- Monteiro, Miguel. Programação Paralela, Abril 2010, http://paginas.fe.up.pt/~apm/CPAR/.
- MCS Division. Message Passing Interface, Abril 2010, http://www.mcs.anl.gov/research/projects/mpi/.

Anexos

Anexo I – Código

io.h

```
#ifndef __INPUT_H_
#define __INPUT_H_
#include "image.h"

unsigned char * readImage(const char* path, uint & width, uint &
heigth);
void writeImage(const char* path, const uchar * i, uint width, uint
height);
#endif
```

io.cpp

```
#include <iomanip>
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <cstring>
using namespace std;
#include "io.h"

inline char * clearSpaces(char*c){
	while(isspace(*c))c++;
	return c;
}

inline char * clearLine(char*c){
	while(*c && *c != '\n')c++;
	if(*c == '\n')
		return c + 1;
	return c;
}

int nextInt(char * buf, char**end, char comment='#'){
	buf=clearSpaces(buf);
	while(*buf){
		if(*buf==comment)//comment
			buf=clearLine(buf);
		else if(isdigit(*buf)){//number
			char * btmp;
			int tmp=strtol(buf,&btmp,10);
			if(buf==btmp)//no numbers
				break;
			*end=btmp;
			return tmp;
		}else
```

```
                break;
            buf=clearSpaces(buf);
        }
        throw string("Error while reading file");
}


uchar * readImage(const char* path, uint & w, uint & h){
        ifstream f(path);
        char * data, *current;
        uint l, max;

        f.seekg (0, ios::end);
        l=f.tellg();

        current=data=new char[l+1];

        f.seekg (0, ios::beg);
        f.read (data,l);
        f.close();
        data[l]=0;


        if(strncmp("P3",data,2)){
                data[2]=0;
                string error = "Invalid file type: \"";
                error += string(data);
                error += "\"";
                //delete data;
                throw error;
                //throw string("Invalid file type");
        }

        current+=2;

        w=nextInt(current, &current);
        h=nextInt(current, &current);
        max=nextInt(current, &current);

        cout<<"Resolution: "<<w<<"x"<<h<<endl;

        uint num_samples = numSamples(w, h);

        unsigned char * buf = new unsigned char [num_samples];

        for(uint s=0;s<num_samples;s++)
                buf[s]=nextInt(current, &current)*255l/max;

        delete data;
        return buf;
}


void writeImage(const char* path, const uchar * i, uint w, uint h){
        FILE* ofp;
        ofp = fopen(path, "w");

        if (ofp == NULL)
                throw string("Can't open output file");

        uint n = numSamples(w,h);
```

```
	fprintf(ofp, "P3\n%d %d\n255\n", w, h);

	for (uint a = 0; a != n; a++)
		fprintf(ofp, "%d\n", a[i]);

	fclose(ofp);
}

void writeImageBinary(const char* path, const uchar * i, uint w, uint
h){
	ofstream f(path);
	f<<"P6"<<endl;
	f<<w<<" "<<h<<endl;
	f<<255<<endl;

	f.write((char*)i,numSamples(h,w));
	f.close();
}
```

image.h

```
#ifndef __IMAGE_H__
#define __IMAGE_H__
#include "config.h"
#define getRed(x) ((x)[0])
#define getGreen(x) ((x)[1])
#define getBlue(x) ((x)[2])
#define nextPixel(x) ((x)+3)
#define pixelEquals(x,y) (((x[0])==(y[0])) && ((x[1])==(y[1])) &&
((x[2])==(y[2])))
#define get(buf, w, x, y) ((buf)+(((y)*(w))+(x))*3)
#define numSamples(w,h) ((w)*(h)*3)
#define lineBytes(x) ((x)*3)
#endif
```

config.h

```
#ifndef __CONFIG_H__
#define __CONFIG_H__
typedef unsigned int uint;
typedef unsigned char uchar;
#endif
```

algorithm.h

```
#ifndef __ALGORITHM_H__
#define __ALGORITHM_H__
#include "config.h"
#include "image.h"

#include <algorithm>
#include <limits>
#include <cstdlib>
#include <cstring>
using namespace std;

template <class T>
inline T newValueChannel(const T target, T * N){
	T min, max, med;

	sort(N,N+8);
	if(N[0]>target){
		max=N[7];
		min=target;
		med=N[3];
	}else{
		max=target;
		min=N[0];
		med=N[4];
	}

	float pl, pd, tpl, tpd, val=0, Cdd, Cdl, Cld, Cll,delta;

	delta=(float)max-min;

	tpl=(target-min)/delta;
	tpd=1-tpl;

	Cll=1+(med-min)/delta;
	Cdd=2-(med-min)/delta;
	Cld=0.8f;
	Cdl=0.8f;
	for(int i=0;i<8;i++){ // Compute probabilities
		pl=(N[i]-min)/delta;
		pd=1-pl;
		val += (Cdl*pd+Cll*pl)
                                              /     (Cdd*pd*tpd +
                                                    Cdl*pl*tpd +
                                                    Cld*pd*tpl +
                                                    Cll*pl*tpl);
	}

	val *= 0.125f * tpl;

	if(int(val*10)>5){
		if(target<std::numeric_limits<T>::max())
			return target+1;
	} else if(int(val*10)<5)
		if(target>0)
			return target-1;

	return target;
}
```

```
inline void
newValue(const uchar * target, const uchar *  N0, const uchar * N1,
const uchar * N2,

                  const uchar *  N3, const uchar * N4, const uchar *
N5,

                  const uchar *  N6, const uchar * N7,

                  uchar * dest){

      uchar
Nr[8]={getRed(N0),getRed(N1),getRed(N2),getRed(N3),getRed(N4),getRed(N
5),getRed(N6),getRed(N7)};
      uchar
Ng[8]={getGreen(N0),getGreen(N1),getGreen(N2),getGreen(N3),getGreen(N4
),getGreen(N5),getGreen(N6),getGreen(N7)};
      uchar
Nb[8]={getBlue(N0),getBlue(N1),getBlue(N2),getBlue(N3),getBlue(N4),get
Blue(N5),getBlue(N6),getBlue(N7)};


      getRed(dest)=newValueChannel(getRed(target), Nr);
      getGreen(dest)=newValueChannel(getGreen(target), Ng);
      getBlue(dest)=newValueChannel(getBlue(target), Nb);
}

inline int
step(const uchar * in, uchar* out, uint w, uint h){
      uint num_samples=numSamples(w,h);
      uchar *tmp = new uchar[num_samples];

      int diff=0;
      for(uint y=1;y<h-1;y++){
            memcpy(get(tmp, w, 0, y), get(in, w, 0, y), 3);
            memcpy(get(tmp, w, w - 1, y), get(in, w, w - 1, y), 3);
            for(uint x=1;x<w-1;x++){
                  newValue(get(in, w, x, y),
                              get(in, w, x-1,y-1),    get(in, w, x,y-
1),   get(in, w, x+1,y-1),
                              get(in, w, x-1,y),
                                                get(in, w, x+1,y),
                              get(in, w, x-1,y+1),    get(in, w,
x,y+1),     get(in, w, x+1,y+1),
                              get(tmp, w, x, y)
                              );
                  if (!pixelEquals(get(in, w, x, y), get(tmp, w, x,
y)))
                        diff++;
            }
      }
      memcpy(get(out, w, 0, 1), get(tmp, w, 0, 1), num_samples -
lineBytes(w) * 2);
      delete tmp;
      return diff;
}

#endif
```

Sequencial

enhance.cpp

```
#include <ctime>
#include <string>
#include <iostream>
#include <algorithm>
#include <limits>
#include <cstdlib>
#include <cstring>

#include "algorithm.h"
#include "image.h"
#include "io.h"
using namespace std;


int enhance(uchar * in, uchar * out, uint w, uint h, int steps){
	uint num_samples=numSamples(w,h);
	uchar * a = new uchar[num_samples] , *b = new
uchar[num_samples];
	//Init buffers
	memcpy(a, in, num_samples);
	memcpy(b, in, num_samples);

	int i,diff;
	for(i=0;i<steps;i++){
		diff=step(a, b, w, h);
		cout<<"Step: "<<i+1<<" Diffs: "<<diff<<endl;
		swap(a, b);
		if(!diff)
			break;
	}
	memcpy(out, a, num_samples);
	return i;
}



int main(int argc, char** argv){
	if(argc != 4){
		cout<<"Usage: "<<argv[0]<<" <input> <output>
<steps>"<<endl;
		return 1;
	}

	try{
		clock_t read_init, read_end, proc_init, proc_end,
write_init, write_end;

		// Reading from file
		cout << "Reading from file..." << endl;
		read_init = clock();
		uint w,h;
		uchar * in=readImage(argv[1],w,h);
		uint num_samples=numSamples(w,h);
		uchar * out = new uchar[num_samples];

		read_end = clock();
```

```
            cout << "Success!" << endl << endl;

            // Image processing
            cout << "Processing image..." << endl;
            proc_init = clock();
            int steps=enhance(in,out,w,h,atoi(argv[3]));
            proc_end = clock();
            cout<<"Success! Done in "<<steps<<" steps."<<endl<<endl;

            // Writing to file
            cout << "Writing to file..." << endl;
            write_init = clock();
            writeImage(argv[2],out,w,h);
            write_end = clock();
            cout << "Success!" << endl << endl;
            delete in;

            // Benchmarking
            cout << endl << "BENCHMARKING" << endl;
            cout << "Reading: " << (read_end -
read_init)/(float)CLOCKS_PER_SEC << " s" << endl;
            cout << "Processing: " << (proc_end -
proc_init)/(float)CLOCKS_PER_SEC << " s" << endl;
            cout << "Writing: " << (write_end -
write_init)/(float)CLOCKS_PER_SEC << " s" << endl;

      }catch(string e){
            cout<<"Exception: "<<e<<endl;
      }

}
```

Versão base

enhance_mpi.cpp

```
#include <ctime>
#include <string>
#include <fstream>
#include <iostream>
#include <algorithm>
#include <limits>
#include <cstdlib>
#include <cstring>
#include <mpi.h>
#include <vector>
#include <cmath>
#include <iomanip>

#include "algorithm.h"
#include "image.h"
#include "io.h"
using namespace std;

#define LINES 0
#define WIDTH 1
#define DATA 2
#define DIFF 3

int round(float x) { return int(x+0.5); }

void master(int np, int id, char * inPath, char * outPath, int
iterations) {
            clock_t read_init, read_end, proc_init, proc_end,
write_init, write_end;

            cout << "Reading from file..." << endl;

            read_init = clock();

            MPI_Status status;
            uint w,h, num_samples;
            uchar * in, *out;
            try{
                  in=readImage(inPath,w,h);
                  num_samples=numSamples(w,h);
                  out = new uchar[num_samples];
                  memcpy(out, in, lineBytes(w));
                  memcpy(get(out,w, 0, h-1), get(in, w, 0, h-1),
lineBytes(w));
            }catch (const string & s){
                  cout << "Fail: "<<s<<endl;
                  return ;
            }


            read_end = clock();

            cout << "Success!" << endl << endl;

            //Processing
            vector < pair <uint, uint> > range(np);
```

```
            proc_init = clock();
            //Startup comm
            uint pos = 1;
            uint tmp;
            for (int i=1; i<np; i++){
                  MPI_Send(&w, 1, MPI_UNSIGNED, i, WIDTH,
MPI_COMM_WORLD);
                  range[i].first = pos;
                  range[i].second = round((h - pos - 1) / float(np -
i));
                  pos += range[i].second;
                  cout <<"ID: "<<i<< " Processing "<<range[i].second<<"
pixels"<<endl;

                  tmp=range[i].second + 2;
                  MPI_Send(&tmp, 1, MPI_UNSIGNED, i, LINES,
MPI_COMM_WORLD);
                  MPI_Send(get(in, w, 0, range[i].first - 1),
numSamples(w, range[i].second + 2), MPI_UNSIGNED_CHAR, i, DATA,
MPI_COMM_WORLD);
            }

            cout << endl;

            uint diffs;
            for(int it=0; it<iterations; it++){
                  diffs=0;
                  //Enviar as linhas adjacentesa
                  for (int i=1; i<np; i++){
                        MPI_Send(get(in, w, 0, range[i].first - 1),
lineBytes(w), MPI_UNSIGNED_CHAR, i, DATA, MPI_COMM_WORLD);
                        MPI_Send(get(in, w, 0, range[i].first +
range[i].second), lineBytes(w), MPI_UNSIGNED_CHAR, i, DATA,
MPI_COMM_WORLD);
                  }
                  //Receber as fronteiras
                  for (int i=1; i<np; i++){
                        MPI_Recv(&tmp, 1, MPI_UNSIGNED, i, DIFF,
MPI_COMM_WORLD, &status);
                        diffs+=tmp;
                        MPI_Recv(get(in, w, 0, range[i].first),
lineBytes(w), MPI_UNSIGNED_CHAR, i, DATA, MPI_COMM_WORLD, &status);
                        MPI_Recv(get(in, w, 0, range[i].first +
range[i].second - 1), lineBytes(w), MPI_UNSIGNED_CHAR, i, DATA,
MPI_COMM_WORLD, &status);
                  }
                  cout<<"Iteration "<<it<<": "<<diffs <<" diffs"<<endl;
/*
                  //TODO: Comunicar com os outros a dizer que acabou.
                  if(!diffs)
                        break;
      */
            }

            //Collect all
            for (int i=1; i<np; i++)
                  MPI_Recv(get(out, w, 0, range[i].first),
numSamples(w, range[i].second), MPI_UNSIGNED_CHAR, i, DATA,
MPI_COMM_WORLD, &status);
```

```
            proc_end = clock();

            // Writing to file
            cout << endl << "Writing to file..." << endl;

            write_init = clock();
            writeImage(outPath,out,w,h);
            write_end = clock();
            cout << "Success!" << endl << endl;

            delete in;
            delete out;
            // Benchmarking
            cout << endl << "BENCHMARKING" << endl;
            cout << "Reading: " << (read_end -
read_init)/(float)CLOCKS_PER_SEC << " s" << endl;
            cout << "Processing: " << (proc_end -
proc_init)/(float)CLOCKS_PER_SEC << " s" << endl;
            cout << "Writing: " << (write_end -
write_init)/(float)CLOCKS_PER_SEC << " s" << endl;



}

void worker(int np, int id, int iterations){
      int w, h;
      uint num_samples;
      uchar*buf;
      MPI_Status status;
      MPI_Recv(&w, 1, MPI_UNSIGNED, 0, WIDTH, MPI_COMM_WORLD,
&status);
      MPI_Recv(&h, 1, MPI_UNSIGNED, 0, LINES, MPI_COMM_WORLD,
&status);

      num_samples = numSamples(w,h);
      buf=new uchar[num_samples];

      MPI_Recv(buf, num_samples, MPI_UNSIGNED_CHAR, 0, DATA,
MPI_COMM_WORLD, &status);

      cout <<"ID: "<<id <<" Width: "<<w<<" Height:"<<h<<endl;
      /*
      for(int i=0;i<num_samples;i++)
            cout <<setw(2)<< (int)buf[i] << " ";
      cout << endl;
      */

      for(int i = 0; i < iterations; i++){
            //Receber as linhas adjacentes
            MPI_Recv(get(buf, w, 0, 0), lineBytes(w),
MPI_UNSIGNED_CHAR, 0, DATA, MPI_COMM_WORLD, &status);
            MPI_Recv(get(buf, w, 0, h-1), lineBytes(w),
MPI_UNSIGNED_CHAR, 0, DATA, MPI_COMM_WORLD, &status);
            uint diff=step(buf, buf,w ,h);
            //Enviar linhas de fronteira
            MPI_Send(&diff, 1, MPI_UNSIGNED, 0, DIFF, MPI_COMM_WORLD);
            MPI_Send(get(buf, w, 0, 1), lineBytes(w),
MPI_UNSIGNED_CHAR, 0, DATA, MPI_COMM_WORLD);
```

```
            MPI_Send(get(buf, w, 0, h-2), lineBytes(w),
MPI_UNSIGNED_CHAR, 0, DATA, MPI_COMM_WORLD);
      }
      MPI_Send(get(buf, w, 0, 1), numSamples(w, h - 2),
MPI_UNSIGNED_CHAR, 0, DATA, MPI_COMM_WORLD);
}


int main(int argc, char** argv){
      if(argc != 4){
            cout<<"Usage: "<<argv[0]<<" <input> <output>
<steps>"<<endl;
            return 1;
      }

      int np, id;
      MPI_Init(&argc, &argv);
      MPI_Comm_size(MPI_COMM_WORLD, &np);
      MPI_Comm_rank(MPI_COMM_WORLD, &id);

      if (id == 0)
            master(np, id, argv[1], argv[2], atoi(argv[3]));
      else
            worker(np, id, atoi(argv[3]));

      MPI_Finalize();

      return 0;
}
```

Melhoramento 1

enhance_mpi.cpp

```
#include <ctime>
#include <string>
#include <fstream>
#include <iostream>
#include <algorithm>
#include <limits>
#include <cstdlib>
#include <cstring>
#include <mpi.h>
#include <vector>
#include <cmath>
#include <iomanip>

#include "algorithm.h"
#include "image.h"
#include "io.h"
using namespace std;

#define LINES 0
#define WIDTH 1
#define DATA 2
#define DIFF 3

int round(float x) { return int(x+0.5f); }

void master(int np, int id, char * inPath, char * outPath, int
iterations) {
            clock_t read_init, read_end, proc_init, proc_end,
write_init, write_end;

            cout << "Reading from file..." << endl;

            read_init = clock();

            MPI_Status status;
            uint w,h, num_samples;
            uchar * in, *out;
            try{
                  in=readImage(inPath,w,h);
                  num_samples=numSamples(w,h);
                  out = new uchar[num_samples];
                  memcpy(out, in, lineBytes(w));
                  memcpy(get(out,w, 0, h-1), get(in, w, 0, h-1),
lineBytes(w));
            }catch (const string & s){
                  cout << "Fail: "<<s<<endl;
                  return ;
            }


            read_end = clock();

            cout << "Success!" << endl << endl;

            //Processing
            vector < pair <uint, uint> > range(np);
```

```
            proc_init = clock();
            //Startup comm
            uint pos = 1;
            uint tmp;

            range[0].first = pos;
            range[0].second = round((h - pos - 2) / float(np));
            pos += range[0].second;
            cout <<"ID: "<<0<< " Processing "<<range[0].second<<" lines
Begin: "<<range[0].first<<" End:
"<<(range[0].first+range[0].second)<<endl;

            for (int i=1; i<np; i++){
                  MPI_Send(&w, 1, MPI_UNSIGNED, i, WIDTH,
MPI_COMM_WORLD);
                  range[i].first = pos;
                  range[i].second = round((h - pos - 2) / float(np -
i));
                  pos += range[i].second;
                  cout <<"ID: "<<i<< " Processing "<<range[i].second<<"
lines Begin: "<<range[i].first<<" End:
"<<(range[i].first+range[i].second)<<endl;

                  tmp=range[i].second + 2;
                  MPI_Send(&tmp, 1, MPI_UNSIGNED, i, LINES,
MPI_COMM_WORLD);
                  MPI_Send(get(in, w, 0, range[i].first - 1),
numSamples(w, tmp), MPI_UNSIGNED_CHAR, i, DATA, MPI_COMM_WORLD);
            }

            cout << endl;
            cout <<"ID: "<<id <<" Width: "<<w<<"
Height:"<<range[0].second + 2<<endl;

            uint diffs;
            for(int it=0; it<iterations; it++){
                  diffs=0;
                  //Enviar as linhas adjacentes
                  for (int i=1; i<np; i++){
                        MPI_Send(get(in, w, 0, range[i].first - 1),
lineBytes(w), MPI_UNSIGNED_CHAR, i, DATA, MPI_COMM_WORLD);
                        MPI_Send(get(in, w, 0, range[i].first +
range[i].second), lineBytes(w), MPI_UNSIGNED_CHAR, i, DATA,
MPI_COMM_WORLD);
                  }


                  // Processar (Master at work)
                  diffs += step(in, in, w, range[0].second + 2);


                  //Receber as fronteiras
                  for (int i=1; i<np; i++){
                        MPI_Recv(&tmp, 1, MPI_UNSIGNED, i, DIFF,
MPI_COMM_WORLD, &status);
                        diffs+=tmp;
                        MPI_Recv(get(in, w, 0, range[i].first),
lineBytes(w), MPI_UNSIGNED_CHAR, i, DATA, MPI_COMM_WORLD, &status);
```

```
                        MPI_Recv(get(in, w, 0, range[i].first +
range[i].second - 1), lineBytes(w), MPI_UNSIGNED_CHAR, i, DATA,
MPI_COMM_WORLD, &status);
                  }
                  cout<<"Iteration "<<it<<": "<<diffs <<" diffs"<<endl;

            }

            memcpy(out, in, num_samples);

            //Collect all
            for (int i=1; i<np; i++)
                  MPI_Recv(get(out, w, 0, range[i].first),
numSamples(w, range[i].second), MPI_UNSIGNED_CHAR, i, DATA,
MPI_COMM_WORLD, &status);

            proc_end = clock();

            // Writing to file
            cout << endl << "Writing to file..." << endl;

            write_init = clock();
            writeImage(outPath,out,w,h);
            write_end = clock();
            cout << "Success!" << endl << endl;

            delete in;
            delete out;
            // Benchmarking
            cout << endl << "BENCHMARKING" << endl;
            cout << "Reading: " << (read_end -
read_init)/(float)CLOCKS_PER_SEC << " s" << endl;
            cout << "Processing: " << (proc_end -
proc_init)/(float)CLOCKS_PER_SEC << " s" << endl;
            cout << "Writing: " << (write_end -
write_init)/(float)CLOCKS_PER_SEC << " s" << endl;
}

void worker(int np, int id, int iterations){
      int w, h;
      uint num_samples;
      uchar*buf;
      MPI_Status status;
      MPI_Recv(&w, 1, MPI_UNSIGNED, 0, WIDTH, MPI_COMM_WORLD,
&status);
      MPI_Recv(&h, 1, MPI_UNSIGNED, 0, LINES, MPI_COMM_WORLD,
&status);

      num_samples = numSamples(w,h);
      buf=new uchar[num_samples];

      MPI_Recv(buf, num_samples, MPI_UNSIGNED_CHAR, 0, DATA,
MPI_COMM_WORLD, &status);

      cout <<"ID: "<<id <<" Width: "<<w<<" Height:"<<h<<endl;
      /*
      for(int i=0;i<num_samples;i++)
            cout <<setw(2)<< (int)buf[i] << " ";
      cout << endl;
      */
```

```
	for(int i = 0; i < iterations; i++){
		//Receber as linhas adjacentes
		MPI_Recv(get(buf, w, 0, 0), lineBytes(w),
MPI_UNSIGNED_CHAR, 0, DATA, MPI_COMM_WORLD, &status);
		MPI_Recv(get(buf, w, 0, h-1), lineBytes(w),
MPI_UNSIGNED_CHAR, 0, DATA, MPI_COMM_WORLD, &status);
		uint diff=step(buf, buf,w ,h);
		//Enviar linhas de fronteira
		MPI_Send(&diff, 1, MPI_UNSIGNED, 0, DIFF, MPI_COMM_WORLD);
		MPI_Send(get(buf, w, 0, 1), lineBytes(w),
MPI_UNSIGNED_CHAR, 0, DATA, MPI_COMM_WORLD);
		MPI_Send(get(buf, w, 0, h-2), lineBytes(w),
MPI_UNSIGNED_CHAR, 0, DATA, MPI_COMM_WORLD);
	}
	MPI_Send(get(buf, w, 0, 1), numSamples(w, h - 2),
MPI_UNSIGNED_CHAR, 0, DATA, MPI_COMM_WORLD);
}


int main(int argc, char** argv){
	if(argc != 4){
		cout<<"Usage: "<<argv[0]<<" <input> <output>
<steps>"<<endl;
		return 1;
	}

	int np, id;
	MPI_Init(&argc, &argv);
	MPI_Comm_size(MPI_COMM_WORLD, &np);
	MPI_Comm_rank(MPI_COMM_WORLD, &id);

	if (id == 0)
		master(np, id, argv[1], argv[2], atoi(argv[3]));
	else
		worker(np, id, atoi(argv[3]));

	MPI_Finalize();

	return 0;
}
```

Melhoramento 2

enhance_mpi.cpp

```
#include <ctime>
#include <string>
#include <fstream>
#include <iostream>
#include <algorithm>
#include <limits>
#include <cstdlib>
#include <cstring>
#include <mpi.h>
#include <vector>
#include <cmath>
#include <iomanip>

#include "algorithm.h"
#include "image.h"
#include "io.h"
using namespace std;

#define LINES 0
#define WIDTH 1
#define DATA 2
#define DIFF 3

int round(float x) { return int(x+0.5f); }

void master(int np, int id, char * inPath, char * outPath, int
iterations) {
            clock_t read_init, read_end, proc_init, proc_end,
write_init, write_end;

            cout << "Reading from file..." << endl;

            read_init = clock();

            MPI_Status status;
            uint w,h, num_samples;
            uchar * in, *out;
            try{
                  in=readImage(inPath,w,h);
                  num_samples=numSamples(w,h);
                  out = new uchar[num_samples];
                  memcpy(out, in, lineBytes(w));
                  memcpy(get(out,w, 0, h-1), get(in, w, 0, h-1),
lineBytes(w));
            }catch (const string & s){
                  cout << "Fail: "<<s<<endl;
                  return ;
            }


            read_end = clock();

            cout << "Success!" << endl << endl;

            //Processing
            vector < pair <uint, uint> > range(np);
```

```
            proc_init = clock();
            //Startup comm
            uint pos = 1;
            uint tmp;

            range[0].first = pos;
            range[0].second = round((h - pos - 2) / float(np));
            pos += range[0].second;
            cout <<"ID: "<<0<< " Processing "<<range[0].second<<" lines
Begin: "<<range[0].first<<" End:
"<<(range[0].first+range[0].second)<<endl;

            for (int i=1; i<np; i++){
                  MPI_Send(&w, 1, MPI_UNSIGNED, i, WIDTH,
MPI_COMM_WORLD);
                  range[i].first = pos;
                  range[i].second = round((h - pos - 2) / float(np -
i));
                  pos += range[i].second;
                  cout <<"ID: "<<i<< " Processing "<<range[i].second<<"
lines Begin: "<<range[i].first<<" End:
"<<(range[i].first+range[i].second)<<endl;

                  tmp=range[i].second + 2;
                  MPI_Send(&tmp, 1, MPI_UNSIGNED, i, LINES,
MPI_COMM_WORLD);
                  MPI_Send(get(in, w, 0, range[i].first - 1),
numSamples(w, tmp), MPI_UNSIGNED_CHAR, i, DATA, MPI_COMM_WORLD);
            }

            cout << endl;
            cout <<"ID: "<<id <<" Width: "<<w<<"
Height:"<<range[0].second + 2<<endl;

            uint diffs;
            if (iterations > 0)
            {
                  diffs = 0;
                  diffs += step(in, in, w, range[0].second + 2);

                  //Receber a fronteira
                  if (np > 1)
                        MPI_Recv(get(in, w, 0, range[1].first),
lineBytes(w), MPI_UNSIGNED_CHAR, 1, DATA, MPI_COMM_WORLD, &status);

                  for (int i=1; i<np; i++){
                        MPI_Recv(&tmp, 1, MPI_UNSIGNED, i, DIFF,
MPI_COMM_WORLD, &status);
                        diffs+=tmp;
                  }

                  cout<<"Iteration 0: "<<diffs <<" diffs"<<endl;
            }

            for(int it=1; it<iterations-1; it++){
                  diffs=0;

                  // Enviar a linha adjacente
                  if (np > 1)
```

```
                {
                        // Enviar a linha adjacente
                        MPI_Send(get(in, w, 0, range[1].first - 1),
lineBytes(w), MPI_UNSIGNED_CHAR, 1, DATA, MPI_COMM_WORLD);

                        // Processar (Master at work)
                        diffs += step(in, in, w, range[0].second + 2);

                        //Receber a fronteira
                        MPI_Recv(get(in, w, 0, range[1].first),
lineBytes(w), MPI_UNSIGNED_CHAR, 1, DATA, MPI_COMM_WORLD, &status);
                }
                else
                        diffs += step(in, in, w, range[0].second + 2);


                for (int i=1; i<np; i++){
                        MPI_Recv(&tmp, 1, MPI_UNSIGNED, i, DIFF,
MPI_COMM_WORLD, &status);
                        diffs+=tmp;
                }

                cout<<"Iteration "<<it<<": "<<diffs <<" diffs"<<endl;
        }


        if (iterations > 1)
        {
                diffs=0;

                // Enviar a linha adjacente
                if (np > 1)
                        MPI_Send(get(in, w, 0, range[1].first - 1),
lineBytes(w), MPI_UNSIGNED_CHAR, 1, DATA, MPI_COMM_WORLD);

                diffs += step(in, in, w, range[0].second + 2);

                for (int i=1; i<np; i++){
                        MPI_Recv(&tmp, 1, MPI_UNSIGNED, i, DIFF,
MPI_COMM_WORLD, &status);
                        diffs+=tmp;
                }

                cout<<"Iteration "<<iterations-1<<": "<<diffs <<"
diffs"<<endl;
        }

        memcpy(out, in, num_samples);

        //Collect all
        for (int i=1; i<np; i++)
                MPI_Recv(get(out, w, 0, range[i].first),
numSamples(w, range[i].second), MPI_UNSIGNED_CHAR, i, DATA,
MPI_COMM_WORLD, &status);

        proc_end = clock();

        // Writing to file
        cout << endl << "Writing to file..." << endl;

        write_init = clock();
```

```
            writeImage(outPath,out,w,h);
            write_end = clock();
            cout << "Success!" << endl << endl;

            delete in;
            delete out;
            // Benchmarking
            cout << endl << "BENCHMARKING" << endl;
            cout << "Reading: " << (read_end -
read_init)/(float)CLOCKS_PER_SEC << " s" << endl;
            cout << "Processing: " << (proc_end -
proc_init)/(float)CLOCKS_PER_SEC << " s" << endl;
            cout << "Writing: " << (write_end -
write_init)/(float)CLOCKS_PER_SEC << " s" << endl;
}

void worker(int np, int id, int iterations){
      int w, h;
      uint num_samples;
      uchar*buf;
      MPI_Status status;
      MPI_Recv(&w, 1, MPI_UNSIGNED, 0, WIDTH, MPI_COMM_WORLD,
&status);
      MPI_Recv(&h, 1, MPI_UNSIGNED, 0, LINES, MPI_COMM_WORLD,
&status);

      num_samples = numSamples(w,h);
      buf=new uchar[num_samples];

      MPI_Recv(buf, num_samples, MPI_UNSIGNED_CHAR, 0, DATA,
MPI_COMM_WORLD, &status);

      cout <<"ID: "<<id <<" Width: "<<w<<" Height:"<<h<<endl;
      /*
      for(int i=0;i<num_samples;i++)
            cout <<setw(2)<< (int)buf[i] << " ";
      cout << endl;
      */

      uint diff;

      if (iterations > 0)
      {
            diff=step(buf, buf,w ,h);
            MPI_Send(&diff, 1, MPI_UNSIGNED, 0, DIFF, MPI_COMM_WORLD);
            MPI_Send(get(buf, w, 0, 1), lineBytes(w),
MPI_UNSIGNED_CHAR, id-1, DATA, MPI_COMM_WORLD);
      }

      // Último processo, não envia para a frente
      if (id+1 == np)
      {
            for(int i = 1; i < iterations-1; i++){
            //Receber as linhas adjacentes
                  MPI_Recv(get(buf, w, 0, 0), lineBytes(w),
MPI_UNSIGNED_CHAR, id-1, DATA, MPI_COMM_WORLD, &status);
                  diff=step(buf, buf,w ,h);
                  //Enviar linhas de fronteira
                  MPI_Send(&diff, 1, MPI_UNSIGNED, 0, DIFF,
MPI_COMM_WORLD);
```

```
                  MPI_Send(get(buf, w, 0, 1), lineBytes(w),
MPI_UNSIGNED_CHAR, id-1, DATA, MPI_COMM_WORLD);
            }

            if (iterations > 1)
                  MPI_Recv(get(buf, w, 0, 0), lineBytes(w),
MPI_UNSIGNED_CHAR, id-1, DATA, MPI_COMM_WORLD, &status);
      }
      else
      {
            if (iterations > 0)
                  MPI_Send(get(buf, w, 0, h-2), lineBytes(w),
MPI_UNSIGNED_CHAR, id+1, DATA, MPI_COMM_WORLD);

            for(int i = 1; i < iterations-1; i++){
                  //Receber as linhas adjacentes
                  MPI_Recv(get(buf, w, 0, 0), lineBytes(w),
MPI_UNSIGNED_CHAR, id-1, DATA, MPI_COMM_WORLD, &status);
                  MPI_Recv(get(buf, w, 0, h-1), lineBytes(w),
MPI_UNSIGNED_CHAR, id+1, DATA, MPI_COMM_WORLD, &status);
                  diff=step(buf, buf,w ,h);
                  //Enviar linhas de fronteira
                  MPI_Send(&diff, 1, MPI_UNSIGNED, 0, DIFF,
MPI_COMM_WORLD);
                  MPI_Send(get(buf, w, 0, 1), lineBytes(w),
MPI_UNSIGNED_CHAR, id-1, DATA, MPI_COMM_WORLD);
                  MPI_Send(get(buf, w, 0, h-2), lineBytes(w),
MPI_UNSIGNED_CHAR, id+1, DATA, MPI_COMM_WORLD);
            }

            if (iterations > 1)
            {
                  MPI_Recv(get(buf, w, 0, 0), lineBytes(w),
MPI_UNSIGNED_CHAR, id-1, DATA, MPI_COMM_WORLD, &status);
                  MPI_Recv(get(buf, w, 0, h-1), lineBytes(w),
MPI_UNSIGNED_CHAR, id+1, DATA, MPI_COMM_WORLD, &status);
            }
      }

      if (iterations > 1)
      {
            diff=step(buf, buf,w ,h);
            //Enviar linhas de fronteira
            MPI_Send(&diff, 1, MPI_UNSIGNED, 0, DIFF, MPI_COMM_WORLD);
      }

      MPI_Send(get(buf, w, 0, 1), numSamples(w, h - 2),
MPI_UNSIGNED_CHAR, 0, DATA, MPI_COMM_WORLD);
}


int main(int argc, char** argv){
      if(argc != 4){
            cout<<"Usage: "<<argv[0]<<" <input> <output>
<steps>"<<endl;
            return 1;
      }

      int np, id;
      MPI_Init(&argc, &argv);
      MPI_Comm_size(MPI_COMM_WORLD, &np);
```

```
	MPI_Comm_rank(MPI_COMM_WORLD, &id);

	if (id == 0)
		master(np, id, argv[1], argv[2], atoi(argv[3]));
	else
		worker(np, id, atoi(argv[3]));

	MPI_Finalize();

	return 0;
}
```

Anexo II – Imagens

Neste anexo encontram-se as imagens utilizadas, comprimidas para **JPEG** com 75% da qualidade. O ficheiro Output.jpg é uma compressão do ficheiro resultante do programa corrido com 50 iterações.

Harbor.jpg

Output.jpg

Anexo III – Testes efectuados

Scripts

script1.sh

```
mpimon enhance1/enhance_mpi enhance1/Harbor.ppm enhance1/out.ppm 50 --
node08 1
mpimon enhance1/enhance_mpi enhance1/Harbor.ppm enhance1/out.ppm 50 --
node08 1 node04 1
mpimon enhance1/enhance_mpi enhance1/Harbor.ppm enhance1/out.ppm 50 --
node08 1 node04 1 node06 1 node07 1
mpimon enhance1/enhance_mpi enhance1/Harbor.ppm enhance1/out.ppm 50 --
node08 2 node04 2 node06 2 node07 2
mpimon enhance1/enhance_mpi enhance1/Harbor.ppm enhance1/out.ppm 50 --
node08 2 node04 2 node06 2 node07 2 node03 2 node09 2 node10 2 node11
2
mpimon enhance1/enhance_mpi enhance1/Harbor.ppm enhance1/out.ppm 50 --
node08 2 node04 2 node06 2 node07 2 node03 2 node09 2 node10 2 node11
2 node12 2 node13 2 node14 2 node15 2 node16 2 node17 2 node18 2
node19 2
```

script2.sh

```
mpimon enhance2/enhance_mpi enhance2/Harbor.ppm enhance2/out.ppm 50 --
node08 1
mpimon enhance2/enhance_mpi enhance2/Harbor.ppm enhance2/out.ppm 50 --
node08 1 node04 1
mpimon enhance2/enhance_mpi enhance2/Harbor.ppm enhance2/out.ppm 50 --
node08 1 node04 1 node06 1 node07 1
mpimon enhance2/enhance_mpi enhance2/Harbor.ppm enhance2/out.ppm 50 --
node08 2 node04 2 node06 2 node07 2
mpimon enhance2/enhance_mpi enhance2/Harbor.ppm enhance2/out.ppm 50 --
node08 2 node04 2 node06 2 node07 2 node03 2 node09 2 node10 2 node11
2
mpimon enhance2/enhance_mpi enhance2/Harbor.ppm enhance2/out.ppm 50 --
node08 2 node04 2 node06 2 node07 2 node03 2 node09 2 node10 2 node11
2 node12 2 node13 2 node14 2 node15 2 node16 2 node17 2 node18 2
node19 2
```

script3.sh

```
mpimon enhance3/enhance_mpi enhance3/Harbor.ppm enhance3/out.ppm 50 --
node08 1
mpimon enhance3/enhance_mpi enhance3/Harbor.ppm enhance3/out.ppm 50 --
node08 1 node04 1
mpimon enhance3/enhance_mpi enhance3/Harbor.ppm enhance3/out.ppm 50 --
node08 1 node04 1 node06 1 node07 1
mpimon enhance3/enhance_mpi enhance3/Harbor.ppm enhance3/out.ppm 50 --
node08 2 node04 2 node06 2 node07 2
mpimon enhance3/enhance_mpi enhance3/Harbor.ppm enhance3/out.ppm 50 --
node08 2 node04 2 node06 2 node07 2 node03 2 node09 2 node10 2 node11
2
```

```
mpimon enhance3/enhance_mpi enhance3/Harbor.ppm enhance3/out.ppm 50 --
node08 2 node04 2 node06 2 node07 2 node03 2 node09 2 node10 2 node11
2 node12 2 node13 2 node14 2 node15 2 node16 2 node17 2 node18 2
node19 2
```

Resultados

enhance1

```
Reading from file...
Resolution: 2048x1536
Success!


Iteration 0: 0 diffs
Iteration 1: 0 diffs
Iteration 2: 0 diffs
Iteration 3: 0 diffs
Iteration 4: 0 diffs
Iteration 5: 0 diffs
Iteration 6: 0 diffs
Iteration 7: 0 diffs
Iteration 8: 0 diffs
Iteration 9: 0 diffs
Iteration 10: 0 diffs
Iteration 11: 0 diffs
Iteration 12: 0 diffs
Iteration 13: 0 diffs
Iteration 14: 0 diffs
Iteration 15: 0 diffs
Iteration 16: 0 diffs
Iteration 17: 0 diffs
Iteration 18: 0 diffs
Iteration 19: 0 diffs
Iteration 20: 0 diffs
Iteration 21: 0 diffs
Iteration 22: 0 diffs
Iteration 23: 0 diffs
Iteration 24: 0 diffs
Iteration 25: 0 diffs
Iteration 26: 0 diffs
Iteration 27: 0 diffs
Iteration 28: 0 diffs
Iteration 29: 0 diffs
Iteration 30: 0 diffs
Iteration 31: 0 diffs
Iteration 32: 0 diffs
Iteration 33: 0 diffs
Iteration 34: 0 diffs
Iteration 35: 0 diffs
Iteration 36: 0 diffs
Iteration 37: 0 diffs
Iteration 38: 0 diffs
Iteration 39: 0 diffs
Iteration 40: 0 diffs
Iteration 41: 0 diffs
Iteration 42: 0 diffs
Iteration 43: 0 diffs
Iteration 44: 0 diffs
Iteration 45: 0 diffs
Iteration 46: 0 diffs
Iteration 47: 0 diffs
Iteration 48: 0 diffs
```

```
Iteration 49: 0 diffs

Writing to file...
Success!


BENCHMARKING
Reading: 1.39 s
Processing: 0 s
Writing: 1.88 s



Reading from file...
Resolution: 2048x1536
Success!

ID: 1 Processing 1534 pixels

ID: 1 Width: 2048 Height:1536
Iteration 0: 3137275 diffs
Iteration 1: 3131926 diffs
Iteration 2: 3127384 diffs
Iteration 3: 3125148 diffs
Iteration 4: 3123055 diffs
Iteration 5: 3121055 diffs
Iteration 6: 3119264 diffs
Iteration 7: 3117131 diffs
Iteration 8: 3115319 diffs
Iteration 9: 3113426 diffs
Iteration 10: 3111839 diffs
Iteration 11: 3110280 diffs
Iteration 12: 3108871 diffs
Iteration 13: 3107536 diffs
Iteration 14: 3106203 diffs
Iteration 15: 3105212 diffs
Iteration 16: 3104047 diffs
Iteration 17: 3103038 diffs
Iteration 18: 3101960 diffs
Iteration 19: 3100703 diffs
Iteration 20: 3099590 diffs
Iteration 21: 3098253 diffs
Iteration 22: 3096968 diffs
Iteration 23: 3095601 diffs
Iteration 24: 3094351 diffs
Iteration 25: 3092991 diffs
Iteration 26: 3091741 diffs
Iteration 27: 3090435 diffs
Iteration 28: 3089190 diffs
Iteration 29: 3088009 diffs
Iteration 30: 3086793 diffs
Iteration 31: 3085557 diffs
Iteration 32: 3084423 diffs
Iteration 33: 3083191 diffs
Iteration 34: 3081916 diffs
Iteration 35: 3080661 diffs
Iteration 36: 3079431 diffs
Iteration 37: 3078043 diffs
Iteration 38: 3076608 diffs
Iteration 39: 3075187 diffs
Iteration 40: 3073658 diffs
```

```
Iteration 41: 3072234 diffs
Iteration 42: 3070774 diffs
Iteration 43: 3069319 diffs
Iteration 44: 3067973 diffs
Iteration 45: 3066600 diffs
Iteration 46: 3065222 diffs
Iteration 47: 3063895 diffs
Iteration 48: 3062546 diffs
Iteration 49: 3061127 diffs

Writing to file...
Success!


BENCHMARKING
Reading: 1.13 s
Processing: 208.15 s
Writing: 2.71 s



Reading from file...
Resolution: 2048x1536
Success!

ID: 1 Processing 511 pixels
ID: 2 Processing 512 pixels
ID: 1 Width: 2048 Height:513
ID: 3 Processing 511 pixels
ID: 2 Width: 2048 Height:514

ID: 3 Width: 2048 Height:513
Iteration 0: 3137275 diffs
Iteration 1: 3131926 diffs
Iteration 2: 3127384 diffs
Iteration 3: 3125148 diffs
Iteration 4: 3123055 diffs
Iteration 5: 3121055 diffs
Iteration 6: 3119264 diffs
Iteration 7: 3117131 diffs
Iteration 8: 3115319 diffs
Iteration 9: 3113426 diffs
Iteration 10: 3111839 diffs
Iteration 11: 3110280 diffs
Iteration 12: 3108871 diffs
Iteration 13: 3107536 diffs
Iteration 14: 3106203 diffs
Iteration 15: 3105212 diffs
Iteration 16: 3104047 diffs
Iteration 17: 3103038 diffs
Iteration 18: 3101960 diffs
Iteration 19: 3100703 diffs
Iteration 20: 3099590 diffs
Iteration 21: 3098253 diffs
Iteration 22: 3096968 diffs
Iteration 23: 3095601 diffs
Iteration 24: 3094351 diffs
Iteration 25: 3092991 diffs
Iteration 26: 3091741 diffs
Iteration 27: 3090435 diffs
Iteration 28: 3089190 diffs
```

```
Iteration 29: 3088009 diffs
Iteration 30: 3086793 diffs
Iteration 31: 3085557 diffs
Iteration 32: 3084423 diffs
Iteration 33: 3083191 diffs
Iteration 34: 3081916 diffs
Iteration 35: 3080661 diffs
Iteration 36: 3079431 diffs
Iteration 37: 3078043 diffs
Iteration 38: 3076608 diffs
Iteration 39: 3075187 diffs
Iteration 40: 3073658 diffs
Iteration 41: 3072234 diffs
Iteration 42: 3070774 diffs
Iteration 43: 3069319 diffs
Iteration 44: 3067973 diffs
Iteration 45: 3066600 diffs
Iteration 46: 3065222 diffs
Iteration 47: 3063895 diffs
Iteration 48: 3062546 diffs
Iteration 49: 3061127 diffs

Writing to file...
Success!


BENCHMARKING
Reading: 1.12 s
Processing: 116.35 s
Writing: 2.68 s



Reading from file...
Resolution: 2048x1536
Success!

ID: 1 Processing 219 pixels
ID: 2 Processing 219 pixels
ID: 1 Width: 2048 Height:221
ID: 3 Processing 219 pixels
ID: 2 Width: 2048 Height:221
ID: 4 Processing 219 pixels
ID: 5 Processing 219 pixels
ID: 4 Width: 2048 Height:221
ID: 6 Processing 220 pixels
ID: 3 Width: 2048 Height:221
ID: 5 Width: 2048 Height:221
ID: 7 Processing 219 pixels
ID: 6 Width: 2048 Height:222

ID: 7 Width: 2048 Height:221
Iteration 0: 3137275 diffs
Iteration 1: 3131926 diffs
Iteration 2: 3127384 diffs
Iteration 3: 3125148 diffs
Iteration 4: 3123055 diffs
Iteration 5: 3121055 diffs
Iteration 6: 3119264 diffs
Iteration 7: 3117131 diffs
Iteration 8: 3115319 diffs
```

```
Iteration 9: 3113426 diffs
Iteration 10: 3111839 diffs
Iteration 11: 3110280 diffs
Iteration 12: 3108871 diffs
Iteration 13: 3107536 diffs
Iteration 14: 3106203 diffs
Iteration 15: 3105212 diffs
Iteration 16: 3104047 diffs
Iteration 17: 3103038 diffs
Iteration 18: 3101960 diffs
Iteration 19: 3100703 diffs
Iteration 20: 3099590 diffs
Iteration 21: 3098253 diffs
Iteration 22: 3096968 diffs
Iteration 23: 3095601 diffs
Iteration 24: 3094351 diffs
Iteration 25: 3092991 diffs
Iteration 26: 3091741 diffs
Iteration 27: 3090435 diffs
Iteration 28: 3089190 diffs
Iteration 29: 3088009 diffs
Iteration 30: 3086793 diffs
Iteration 31: 3085557 diffs
Iteration 32: 3084423 diffs
Iteration 33: 3083191 diffs
Iteration 34: 3081916 diffs
Iteration 35: 3080661 diffs
Iteration 36: 3079431 diffs
Iteration 37: 3078043 diffs
Iteration 38: 3076608 diffs
Iteration 39: 3075187 diffs
Iteration 40: 3073658 diffs
Iteration 41: 3072234 diffs
Iteration 42: 3070774 diffs
Iteration 43: 3069319 diffs
Iteration 44: 3067973 diffs
Iteration 45: 3066600 diffs
Iteration 46: 3065222 diffs
Iteration 47: 3063895 diffs
Iteration 48: 3062546 diffs
Iteration 49: 3061127 diffs

Writing to file...
Success!


BENCHMARKING
Reading: 1.11 s
Processing: 97.97 s
Writing: 2.06 s



Reading from file...
Resolution: 2048x1536
Success!

ID: 1 Processing 102 pixels
ID: 2 Processing 102 pixels
ID: 1 Width: 2048 Height:104
ID: 3 Processing 102 pixels
```

```
ID: 2 Width: 2048 Height:104
ID: 4 Processing 102 pixels
ID: 5 Processing 102 pixels
ID: 4 Width: 2048 Height:104
ID: 6 Processing 102 pixels
ID: 5 Width: 2048 Height:104
ID: 7 Processing 102 pixels
ID: 3 Width: 2048 Height:104
ID: 6 Width: 2048 Height:104
ID: 8 Width: 2048 Height:105
ID: 10 Width: 2048 Height:105
ID: 8 Processing 103 pixels
ID: 7 Width: 2048 Height:104
ID: 9 Width: 2048 Height:104
ID: 9 Processing 102 pixels
ID: 11 Width: 2048 Height:104
ID: 10 Processing 103 pixels
ID: 11 Processing 102 pixels
ID: 12 Processing 103 pixels
ID: 13 Processing 102 pixels
ID: 12 Width: 2048 Height:105
ID: 14 Processing 103 pixels
ID: 13 Width: 2048 Height:104
ID: 15 Processing 102 pixels
ID: 14 Width: 2048 Height:105

ID: 15 Width: 2048 Height:104
Iteration 0: 3137275 diffs
Iteration 1: 3131926 diffs
Iteration 2: 3127384 diffs
Iteration 3: 3125148 diffs
Iteration 4: 3123055 diffs
Iteration 5: 3121055 diffs
Iteration 6: 3119264 diffs
Iteration 7: 3117131 diffs
Iteration 8: 3115319 diffs
Iteration 9: 3113426 diffs
Iteration 10: 3111839 diffs
Iteration 11: 3110280 diffs
Iteration 12: 3108871 diffs
Iteration 13: 3107536 diffs
Iteration 14: 3106203 diffs
Iteration 15: 3105212 diffs
Iteration 16: 3104047 diffs
Iteration 17: 3103038 diffs
Iteration 18: 3101960 diffs
Iteration 19: 3100703 diffs
Iteration 20: 3099590 diffs
Iteration 21: 3098253 diffs
Iteration 22: 3096968 diffs
Iteration 23: 3095601 diffs
Iteration 24: 3094351 diffs
Iteration 25: 3092991 diffs
Iteration 26: 3091741 diffs
Iteration 27: 3090435 diffs
Iteration 28: 3089190 diffs
Iteration 29: 3088009 diffs
Iteration 30: 3086793 diffs
Iteration 31: 3085557 diffs
Iteration 32: 3084423 diffs
Iteration 33: 3083191 diffs
```

```
Iteration 34: 3081916 diffs
Iteration 35: 3080661 diffs
Iteration 36: 3079431 diffs
Iteration 37: 3078043 diffs
Iteration 38: 3076608 diffs
Iteration 39: 3075187 diffs
Iteration 40: 3073658 diffs
Iteration 41: 3072234 diffs
Iteration 42: 3070774 diffs
Iteration 43: 3069319 diffs
Iteration 44: 3067973 diffs
Iteration 45: 3066600 diffs
Iteration 46: 3065222 diffs
Iteration 47: 3063895 diffs
Iteration 48: 3062546 diffs
Iteration 49: 3061127 diffs

Writing to file...
Success!


BENCHMARKING
Reading: 1.11 s
Processing: 47.29 s
Writing: 2.37 s



Reading from file...
Resolution: 2048x1536
Success!

ID: 1 Processing 49 pixels
ID: 2 Processing 50 pixels
ID: 1 Width: 2048 Height:51
ID: 3 Processing 49 pixels
ID: 2 Width: 2048 Height:52
ID: 4 Processing 50 pixels
ID: 3 Width: 2048 Height:51
ID: 5 Processing 49 pixels
ID: 4 Width: 2048 Height:52
ID: 6 Processing 50 pixels
ID: 5 Width: 2048 Height:51
ID: 7 Processing 49 pixels
ID: 6 Width: 2048 Height:52
ID: 8 Processing 50 pixels
ID: 7 Width: 2048 Height:51
ID: 9 Processing 49 pixels
ID: 8 Width: 2048 Height:52
ID: 10 Processing 50 pixels
ID: 9 Width: 2048 Height:51
ID: 11 Processing 49 pixels
ID: 10 Width: 2048 Height:52
ID: 12 Processing 50 pixels
ID: 11 Width: 2048 Height:51
ID: 13 Processing 49 pixels
ID: 12 Width: 2048 Height:52
ID: 14 Processing 50 pixels
ID: 13 Width: 2048 Height:51
ID: 15 Processing 49 pixels
ID: 14 Width: 2048 Height:52
```

```
ID: 16 Processing 50 pixels
ID: 15 Width: 2048 Height:51
ID: 17 Processing 49 pixels
ID: 16 Width: 2048 Height:52
ID: 18 Processing 50 pixels
ID: 17 Width: 2048 Height:51
ID: 19 Processing 49 pixels
ID: 18 Width: 2048 Height:52
ID: 20 Processing 50 pixels
ID: 19 Width: 2048 Height:51
ID: 21 Processing 49 pixels
ID: 20 Width: 2048 Height:52
ID: 22 Processing 50 pixels
ID: 21 Width: 2048 Height:51
ID: 23 Processing 49 pixels
ID: 22 Width: 2048 Height:52
ID: 24 Processing 50 pixels
ID: 23 Width: 2048 Height:51
ID: 25 Processing 49 pixels
ID: 24 Width: 2048 Height:52
ID: 26 Width: 2048 Height:52
ID: 26 Processing 50 pixels
ID: 25 Width: 2048 Height:51
ID: 27 Processing 49 pixels
ID: 28 Processing 50 pixels
ID: 29 Processing 49 pixels
ID: 28 Width: 2048 Height:52
ID: 30 Processing 50 pixels
ID: 29 Width: 2048 Height:51
ID: 31 Processing 49 pixels
ID: 27 Width: 2048 Height:51
ID: 30 Width: 2048 Height:52

ID: 31 Width: 2048 Height:51
Iteration 0: 3137275 diffs
Iteration 1: 3131926 diffs
Iteration 2: 3127384 diffs
Iteration 3: 3125148 diffs
Iteration 4: 3123055 diffs
Iteration 5: 3121055 diffs
Iteration 6: 3119264 diffs
Iteration 7: 3117131 diffs
Iteration 8: 3115319 diffs
Iteration 9: 3113426 diffs
Iteration 10: 3111839 diffs
Iteration 11: 3110280 diffs
Iteration 12: 3108871 diffs
Iteration 13: 3107536 diffs
Iteration 14: 3106203 diffs
Iteration 15: 3105212 diffs
Iteration 16: 3104047 diffs
Iteration 17: 3103038 diffs
Iteration 18: 3101960 diffs
Iteration 19: 3100703 diffs
Iteration 20: 3099590 diffs
Iteration 21: 3098253 diffs
Iteration 22: 3096968 diffs
Iteration 23: 3095601 diffs
Iteration 24: 3094351 diffs
Iteration 25: 3092991 diffs
Iteration 26: 3091741 diffs
```

```
Iteration 27: 3090435 diffs
Iteration 28: 3089190 diffs
Iteration 29: 3088009 diffs
Iteration 30: 3086793 diffs
Iteration 31: 3085557 diffs
Iteration 32: 3084423 diffs
Iteration 33: 3083191 diffs
Iteration 34: 3081916 diffs
Iteration 35: 3080661 diffs
Iteration 36: 3079431 diffs
Iteration 37: 3078043 diffs
Iteration 38: 3076608 diffs
Iteration 39: 3075187 diffs
Iteration 40: 3073658 diffs
Iteration 41: 3072234 diffs
Iteration 42: 3070774 diffs
Iteration 43: 3069319 diffs
Iteration 44: 3067973 diffs
Iteration 45: 3066600 diffs
Iteration 46: 3065222 diffs
Iteration 47: 3063895 diffs
Iteration 48: 3062546 diffs
Iteration 49: 3061127 diffs

Writing to file...
Success!


BENCHMARKING
Reading: 1.13 s
Processing: 40.47 s
Writing: 2.06 s
```

enhance2

```
Reading from file...
Resolution: 2048x1536
Success!

ID: 0 Processing 1533 lines Begin: 1 End: 1534

ID: 0 Width: 2048 Height:1535
Iteration 0: 3135229 diffs
Iteration 1: 3129880 diffs
Iteration 2: 3125338 diffs
Iteration 3: 3123102 diffs
Iteration 4: 3121010 diffs
Iteration 5: 3119010 diffs
Iteration 6: 3117219 diffs
Iteration 7: 3115089 diffs
Iteration 8: 3113283 diffs
Iteration 9: 3111397 diffs
Iteration 10: 3109810 diffs
Iteration 11: 3108253 diffs
Iteration 12: 3106846 diffs
Iteration 13: 3105512 diffs
Iteration 14: 3104180 diffs
Iteration 15: 3103191 diffs
Iteration 16: 3102026 diffs
Iteration 17: 3101019 diffs
Iteration 18: 3099942 diffs
Iteration 19: 3098685 diffs
Iteration 20: 3097573 diffs
Iteration 21: 3096237 diffs
Iteration 22: 3094954 diffs
Iteration 23: 3093587 diffs
Iteration 24: 3092337 diffs
Iteration 25: 3090977 diffs
Iteration 26: 3089728 diffs
Iteration 27: 3088422 diffs
Iteration 28: 3087178 diffs
Iteration 29: 3085998 diffs
Iteration 30: 3084782 diffs
Iteration 31: 3083546 diffs
Iteration 32: 3082413 diffs
Iteration 33: 3081182 diffs
Iteration 34: 3079909 diffs
Iteration 35: 3078655 diffs
Iteration 36: 3077425 diffs
Iteration 37: 3076038 diffs
Iteration 38: 3074605 diffs
Iteration 39: 3073184 diffs
Iteration 40: 3071655 diffs
Iteration 41: 3070231 diffs
Iteration 42: 3068771 diffs
Iteration 43: 3067317 diffs
Iteration 44: 3065971 diffs
Iteration 45: 3064598 diffs
Iteration 46: 3063220 diffs
Iteration 47: 3061893 diffs
Iteration 48: 3060544 diffs
Iteration 49: 3059126 diffs
```

```
Writing to file...
Success!


BENCHMARKING
Reading: 1.1 s
Processing: 351.03 s
Writing: 2.39 s



Reading from file...
Resolution: 2048x1536
Success!

ID: 0 Processing 767 lines Begin: 1 End: 768
ID: 1 Processing 766 lines Begin: 768 End: 1534

ID: 0 Width: 2048 Height:769
ID: 1 Width: 2048 Height:768
Iteration 0: 3135229 diffs
Iteration 1: 3129880 diffs
Iteration 2: 3125338 diffs
Iteration 3: 3123102 diffs
Iteration 4: 3121010 diffs
Iteration 5: 3119010 diffs
Iteration 6: 3117219 diffs
Iteration 7: 3115089 diffs
Iteration 8: 3113283 diffs
Iteration 9: 3111397 diffs
Iteration 10: 3109810 diffs
Iteration 11: 3108253 diffs
Iteration 12: 3106846 diffs
Iteration 13: 3105512 diffs
Iteration 14: 3104180 diffs
Iteration 15: 3103191 diffs
Iteration 16: 3102026 diffs
Iteration 17: 3101019 diffs
Iteration 18: 3099942 diffs
Iteration 19: 3098685 diffs
Iteration 20: 3097573 diffs
Iteration 21: 3096237 diffs
Iteration 22: 3094954 diffs
Iteration 23: 3093587 diffs
Iteration 24: 3092337 diffs
Iteration 25: 3090977 diffs
Iteration 26: 3089728 diffs
Iteration 27: 3088422 diffs
Iteration 28: 3087178 diffs
Iteration 29: 3085998 diffs
Iteration 30: 3084782 diffs
Iteration 31: 3083546 diffs
Iteration 32: 3082413 diffs
Iteration 33: 3081182 diffs
Iteration 34: 3079909 diffs
Iteration 35: 3078655 diffs
Iteration 36: 3077425 diffs
Iteration 37: 3076038 diffs
Iteration 38: 3074605 diffs
Iteration 39: 3073184 diffs
Iteration 40: 3071655 diffs
```

```
Iteration 41: 3070231 diffs
Iteration 42: 3068771 diffs
Iteration 43: 3067317 diffs
Iteration 44: 3065971 diffs
Iteration 45: 3064598 diffs
Iteration 46: 3063220 diffs
Iteration 47: 3061893 diffs
Iteration 48: 3060544 diffs
Iteration 49: 3059126 diffs

Writing to file...
Success!


BENCHMARKING
Reading: 1.13 s
Processing: 174.15 s
Writing: 2.47 s



Reading from file...
Resolution: 2048x1536
Success!

ID: 0 Processing 383 lines Begin: 1 End: 384
ID: 1 Processing 383 lines Begin: 384 End: 767
ID: 2 Processing 384 lines Begin: 767 End: 1151
ID: 1 Width: 2048 Height:385
ID: 3 Processing 383 lines Begin: 1151 End: 1534
ID: 2 Width: 2048 Height:386

ID: 0 Width: 2048 Height:385
ID: 3 Width: 2048 Height:385
Iteration 0: 3135229 diffs
Iteration 1: 3129880 diffs
Iteration 2: 3125338 diffs
Iteration 3: 3123102 diffs
Iteration 4: 3121010 diffs
Iteration 5: 3119010 diffs
Iteration 6: 3117219 diffs
Iteration 7: 3115089 diffs
Iteration 8: 3113283 diffs
Iteration 9: 3111397 diffs
Iteration 10: 3109810 diffs
Iteration 11: 3108253 diffs
Iteration 12: 3106846 diffs
Iteration 13: 3105512 diffs
Iteration 14: 3104180 diffs
Iteration 15: 3103191 diffs
Iteration 16: 3102026 diffs
Iteration 17: 3101019 diffs
Iteration 18: 3099942 diffs
Iteration 19: 3098685 diffs
Iteration 20: 3097573 diffs
Iteration 21: 3096237 diffs
Iteration 22: 3094954 diffs
Iteration 23: 3093587 diffs
Iteration 24: 3092337 diffs
Iteration 25: 3090977 diffs
Iteration 26: 3089728 diffs
```

```
Iteration 27: 3088422 diffs
Iteration 28: 3087178 diffs
Iteration 29: 3085998 diffs
Iteration 30: 3084782 diffs
Iteration 31: 3083546 diffs
Iteration 32: 3082413 diffs
Iteration 33: 3081182 diffs
Iteration 34: 3079909 diffs
Iteration 35: 3078655 diffs
Iteration 36: 3077425 diffs
Iteration 37: 3076038 diffs
Iteration 38: 3074605 diffs
Iteration 39: 3073184 diffs
Iteration 40: 3071655 diffs
Iteration 41: 3070231 diffs
Iteration 42: 3068771 diffs
Iteration 43: 3067317 diffs
Iteration 44: 3065971 diffs
Iteration 45: 3064598 diffs
Iteration 46: 3063220 diffs
Iteration 47: 3061893 diffs
Iteration 48: 3060544 diffs
Iteration 49: 3059126 diffs

Writing to file...
Success!


BENCHMARKING
Reading: 1.12 s
Processing: 87.82 s
Writing: 2.41 s



Reading from file...
Resolution: 2048x1536
Success!

ID: 0 Processing 192 lines Begin: 1 End: 193
ID: 1 Processing 192 lines Begin: 193 End: 385
ID: 2 Processing 192 lines Begin: 385 End: 577
ID: 1 Width: 2048 Height:194
ID: 3 Processing 191 lines Begin: 577 End: 768
ID: 2 Width: 2048 Height:194
ID: 4 Processing 192 lines Begin: 768 End: 960
ID: 3 Width: 2048 Height:193
ID: 5 Processing 191 lines Begin: 960 End: 1151
ID: 4 Width: 2048 Height:194
ID: 6 Processing 192 lines Begin: 1151 End: 1343
ID: 5 Width: 2048 Height:193
ID: 7 Processing 191 lines Begin: 1343 End: 1534
ID: 6 Width: 2048 Height:194

ID: 7 Width: 2048 Height:193
ID: 0 Width: 2048 Height:194
Iteration 0: 3135229 diffs
Iteration 1: 3129880 diffs
Iteration 2: 3125338 diffs
Iteration 3: 3123102 diffs
Iteration 4: 3121010 diffs
```

```
Iteration 5: 3119010 diffs
Iteration 6: 3117219 diffs
Iteration 7: 3115089 diffs
Iteration 8: 3113283 diffs
Iteration 9: 3111397 diffs
Iteration 10: 3109810 diffs
Iteration 11: 3108253 diffs
Iteration 12: 3106846 diffs
Iteration 13: 3105512 diffs
Iteration 14: 3104180 diffs
Iteration 15: 3103191 diffs
Iteration 16: 3102026 diffs
Iteration 17: 3101019 diffs
Iteration 18: 3099942 diffs
Iteration 19: 3098685 diffs
Iteration 20: 3097573 diffs
Iteration 21: 3096237 diffs
Iteration 22: 3094954 diffs
Iteration 23: 3093587 diffs
Iteration 24: 3092337 diffs
Iteration 25: 3090977 diffs
Iteration 26: 3089728 diffs
Iteration 27: 3088422 diffs
Iteration 28: 3087178 diffs
Iteration 29: 3085998 diffs
Iteration 30: 3084782 diffs
Iteration 31: 3083546 diffs
Iteration 32: 3082413 diffs
Iteration 33: 3081182 diffs
Iteration 34: 3079909 diffs
Iteration 35: 3078655 diffs
Iteration 36: 3077425 diffs
Iteration 37: 3076038 diffs
Iteration 38: 3074605 diffs
Iteration 39: 3073184 diffs
Iteration 40: 3071655 diffs
Iteration 41: 3070231 diffs
Iteration 42: 3068771 diffs
Iteration 43: 3067317 diffs
Iteration 44: 3065971 diffs
Iteration 45: 3064598 diffs
Iteration 46: 3063220 diffs
Iteration 47: 3061893 diffs
Iteration 48: 3060544 diffs
Iteration 49: 3059126 diffs

Writing to file...
Success!


BENCHMARKING
Reading: 1.12 s
Processing: 44.5 s
Writing: 2.48 s



Reading from file...
Resolution: 2048x1536
Success!
```

```
ID: 0 Processing 96 lines Begin: 1 End: 97
ID: 1 Processing 96 lines Begin: 97 End: 193
ID: 2 Processing 96 lines Begin: 193 End: 289
ID: 1 Width: 2048 Height:98
ID: 3 Processing 96 lines Begin: 289 End: 385
ID: 2 Width: 2048 Height:98
ID: 4 Processing 96 lines Begin: 385 End: 481
ID: 3 Width: 2048 Height:98
ID: 5 Processing 96 lines Begin: 481 End: 577
ID: 4 Width: 2048 Height:98
ID: 6 Width: 2048 Height:98
ID: 6 Processing 96 lines Begin: 577 End: 673
ID: 7 Processing 96 lines Begin: 673 End: 769
ID: 5 Width: 2048 Height:98
ID: 8 Processing 96 lines Begin: 769 End: 865
ID: 7 Width: 2048 Height:98
ID: 8 Width: 2048 Height:98
ID: 9 Processing 96 lines Begin: 865 End: 961
ID: 10 Processing 96 lines Begin: 961 End: 1057
ID: 9 Width: 2048 Height:98
ID: 11 Processing 95 lines Begin: 1057 End: 1152
ID: 10 Width: 2048 Height:98
ID: 12 Processing 96 lines Begin: 1152 End: 1248
ID: 11 Width: 2048 Height:97
ID: 13 Processing 95 lines Begin: 1248 End: 1343
ID: 12 Width: 2048 Height:98
ID: 14 Processing 96 lines Begin: 1343 End: 1439
ID: 13 Width: 2048 Height:97
ID: 15 Processing 95 lines Begin: 1439 End: 1534
ID: 14 Width: 2048 Height:98

ID: 15 Width: 2048 Height:97
ID: 0 Width: 2048 Height:98
Iteration 0: 3135229 diffs
Iteration 1: 3129880 diffs
Iteration 2: 3125338 diffs
Iteration 3: 3123102 diffs
Iteration 4: 3121010 diffs
Iteration 5: 3119010 diffs
Iteration 6: 3117219 diffs
Iteration 7: 3115089 diffs
Iteration 8: 3113283 diffs
Iteration 9: 3111397 diffs
Iteration 10: 3109810 diffs
Iteration 11: 3108253 diffs
Iteration 12: 3106846 diffs
Iteration 13: 3105512 diffs
Iteration 14: 3104180 diffs
Iteration 15: 3103191 diffs
Iteration 16: 3102026 diffs
Iteration 17: 3101019 diffs
Iteration 18: 3099942 diffs
Iteration 19: 3098685 diffs
Iteration 20: 3097573 diffs
Iteration 21: 3096237 diffs
Iteration 22: 3094954 diffs
Iteration 23: 3093587 diffs
Iteration 24: 3092337 diffs
Iteration 25: 3090977 diffs
Iteration 26: 3089728 diffs
Iteration 27: 3088422 diffs
```

```
Iteration 28: 3087178 diffs
Iteration 29: 3085998 diffs
Iteration 30: 3084782 diffs
Iteration 31: 3083546 diffs
Iteration 32: 3082413 diffs
Iteration 33: 3081182 diffs
Iteration 34: 3079909 diffs
Iteration 35: 3078655 diffs
Iteration 36: 3077425 diffs
Iteration 37: 3076038 diffs
Iteration 38: 3074605 diffs
Iteration 39: 3073184 diffs
Iteration 40: 3071655 diffs
Iteration 41: 3070231 diffs
Iteration 42: 3068771 diffs
Iteration 43: 3067317 diffs
Iteration 44: 3065971 diffs
Iteration 45: 3064598 diffs
Iteration 46: 3063220 diffs
Iteration 47: 3061893 diffs
Iteration 48: 3060544 diffs
Iteration 49: 3059126 diffs

Writing to file...
Success!


BENCHMARKING
Reading: 1.13 s
Processing: 36.89 s
Writing: 2.12 s



Reading from file...
Resolution: 2048x1536
Success!

ID: 0 Processing 48 lines Begin: 1 End: 49
ID: 1 Processing 48 lines Begin: 49 End: 97
ID: 2 Processing 48 lines Begin: 97 End: 145
ID: 1 Width: 2048 Height:50
ID: 3 Processing 48 lines Begin: 145 End: 193
ID: 2 Width: 2048 Height:50
ID: 4 Processing 48 lines Begin: 193 End: 241
ID: 3 Width: 2048 Height:50
ID: 5 Processing 48 lines Begin: 241 End: 289
ID: 4 Width: 2048 Height:50
ID: 6 Processing 48 lines Begin: 289 End: 337
ID: 7 Processing 48 lines Begin: 337 End: 385
ID: 6 Width: 2048 Height:50
ID: 5 Width: 2048 Height:50
ID: 8 Processing 48 lines Begin: 385 End: 433
ID: 7 Width: 2048 Height:50
ID: 8 Width: 2048 Height:50
ID: 10 Width: 2048 Height:50
ID: 9 Processing 48 lines Begin: 433 End: 481
ID: 11 Width: 2048 Height:50
ID: 10 Processing 48 lines Begin: 481 End: 529
ID: 9 Width: 2048 Height:50
ID: 11 Processing 48 lines Begin: 529 End: 577
```

```
ID: 12 Processing 48 lines Begin: 577 End: 625
ID: 13 Processing 48 lines Begin: 625 End: 673
ID: 12 Width: 2048 Height:50
ID: 14 Processing 48 lines Begin: 673 End: 721
ID: 13 Width: 2048 Height:50
ID: 15 Processing 48 lines Begin: 721 End: 769
ID: 14 Width: 2048 Height:50
ID: 16 Processing 48 lines Begin: 769 End: 817
ID: 15 Width: 2048 Height:50
ID: 17 Processing 48 lines Begin: 817 End: 865
ID: 16 Width: 2048 Height:50
ID: 18 Processing 48 lines Begin: 865 End: 913
ID: 17 Width: 2048 Height:50
ID: 19 Processing 48 lines Begin: 913 End: 961
ID: 18 Width: 2048 Height:50
ID: 20 Processing 48 lines Begin: 961 End: 1009
ID: 19 Width: 2048 Height:50
ID: 21 Processing 48 lines Begin: 1009 End: 1057
ID: 20 Width: 2048 Height:50
ID: 22 Processing 48 lines Begin: 1057 End: 1105
ID: 21 Width: 2048 Height:50
ID: 23 Processing 48 lines Begin: 1105 End: 1153
ID: 22 Width: 2048 Height:50
ID: 24 Processing 48 lines Begin: 1153 End: 1201
ID: 25 Processing 48 lines Begin: 1201 End: 1249
ID: 24 Width: 2048 Height:50
ID: 26 Processing 48 lines Begin: 1249 End: 1297
ID: 25 Width: 2048 Height:50
ID: 23 Width: 2048 Height:50
ID: 27 Processing 47 lines Begin: 1297 End: 1344
ID: 26 Width: 2048 Height:50
ID: 28 Processing 48 lines Begin: 1344 End: 1392
ID: 29 Processing 47 lines Begin: 1392 End: 1439
ID: 28 Width: 2048 Height:50
ID: 30 Processing 48 lines Begin: 1439 End: 1487
ID: 29 Width: 2048 Height:49
ID: 31 Processing 47 lines Begin: 1487 End: 1534
ID: 27 Width: 2048 Height:49
ID: 30 Width: 2048 Height:50

ID: 0 Width: 2048 Height:50
ID: 31 Width: 2048 Height:49
Iteration 0: 3135229 diffs
Iteration 1: 3129880 diffs
Iteration 2: 3125338 diffs
Iteration 3: 3123102 diffs
Iteration 4: 3121010 diffs
Iteration 5: 3119010 diffs
Iteration 6: 3117219 diffs
Iteration 7: 3115089 diffs
Iteration 8: 3113283 diffs
Iteration 9: 3111397 diffs
Iteration 10: 3109810 diffs
Iteration 11: 3108253 diffs
Iteration 12: 3106846 diffs
Iteration 13: 3105512 diffs
Iteration 14: 3104180 diffs
Iteration 15: 3103191 diffs
Iteration 16: 3102026 diffs
Iteration 17: 3101019 diffs
Iteration 18: 3099942 diffs
```

```
Iteration 19: 3098685 diffs
Iteration 20: 3097573 diffs
Iteration 21: 3096237 diffs
Iteration 22: 3094954 diffs
Iteration 23: 3093587 diffs
Iteration 24: 3092337 diffs
Iteration 25: 3090977 diffs
Iteration 26: 3089728 diffs
Iteration 27: 3088422 diffs
Iteration 28: 3087178 diffs
Iteration 29: 3085998 diffs
Iteration 30: 3084782 diffs
Iteration 31: 3083546 diffs
Iteration 32: 3082413 diffs
Iteration 33: 3081182 diffs
Iteration 34: 3079909 diffs
Iteration 35: 3078655 diffs
Iteration 36: 3077425 diffs
Iteration 37: 3076038 diffs
Iteration 38: 3074605 diffs
Iteration 39: 3073184 diffs
Iteration 40: 3071655 diffs
Iteration 41: 3070231 diffs
Iteration 42: 3068771 diffs
Iteration 43: 3067317 diffs
Iteration 44: 3065971 diffs
Iteration 45: 3064598 diffs
Iteration 46: 3063220 diffs
Iteration 47: 3061893 diffs
Iteration 48: 3060544 diffs
Iteration 49: 3059126 diffs

Writing to file...
Success!


BENCHMARKING
Reading: 1.11 s
Processing: 33.94 s
Writing: 2.04 s
```

enhance3

```
Reading from file...
Resolution: 2048x1536
Success!

ID: 0 Processing 1533 lines Begin: 1 End: 1534

ID: 0 Width: 2048 Height:1535
Iteration 0: 3135229 diffs
Iteration 1: 3129880 diffs
Iteration 2: 3125338 diffs
Iteration 3: 3123102 diffs
Iteration 4: 3121010 diffs
Iteration 5: 3119010 diffs
Iteration 6: 3117219 diffs
Iteration 7: 3115089 diffs
Iteration 8: 3113283 diffs
Iteration 9: 3111397 diffs
Iteration 10: 3109810 diffs
Iteration 11: 3108253 diffs
Iteration 12: 3106846 diffs
Iteration 13: 3105512 diffs
Iteration 14: 3104180 diffs
Iteration 15: 3103191 diffs
Iteration 16: 3102026 diffs
Iteration 17: 3101019 diffs
Iteration 18: 3099942 diffs
Iteration 19: 3098685 diffs
Iteration 20: 3097573 diffs
Iteration 21: 3096237 diffs
Iteration 22: 3094954 diffs
Iteration 23: 3093587 diffs
Iteration 24: 3092337 diffs
Iteration 25: 3090977 diffs
Iteration 26: 3089728 diffs
Iteration 27: 3088422 diffs
Iteration 28: 3087178 diffs
Iteration 29: 3085998 diffs
Iteration 30: 3084782 diffs
Iteration 31: 3083546 diffs
Iteration 32: 3082413 diffs
Iteration 33: 3081182 diffs
Iteration 34: 3079909 diffs
Iteration 35: 3078655 diffs
Iteration 36: 3077425 diffs
Iteration 37: 3076038 diffs
Iteration 38: 3074605 diffs
Iteration 39: 3073184 diffs
Iteration 40: 3071655 diffs
Iteration 41: 3070231 diffs
Iteration 42: 3068771 diffs
Iteration 43: 3067317 diffs
Iteration 44: 3065971 diffs
Iteration 45: 3064598 diffs
Iteration 46: 3063220 diffs
Iteration 47: 3061893 diffs
Iteration 48: 3060544 diffs
Iteration 49: 3059126 diffs
```

```
Writing to file...
Success!


BENCHMARKING
Reading: 1.1 s
Processing: 351.03 s
Writing: 2.39 s



Reading from file...
Resolution: 2048x1536
Success!

ID: 0 Processing 767 lines Begin: 1 End: 768
ID: 1 Processing 766 lines Begin: 768 End: 1534

ID: 0 Width: 2048 Height:769
ID: 1 Width: 2048 Height:768
Iteration 0: 3135229 diffs
Iteration 1: 3129880 diffs
Iteration 2: 3125338 diffs
Iteration 3: 3123102 diffs
Iteration 4: 3121010 diffs
Iteration 5: 3119010 diffs
Iteration 6: 3117219 diffs
Iteration 7: 3115089 diffs
Iteration 8: 3113283 diffs
Iteration 9: 3111397 diffs
Iteration 10: 3109810 diffs
Iteration 11: 3108253 diffs
Iteration 12: 3106846 diffs
Iteration 13: 3105512 diffs
Iteration 14: 3104180 diffs
Iteration 15: 3103191 diffs
Iteration 16: 3102026 diffs
Iteration 17: 3101019 diffs
Iteration 18: 3099942 diffs
Iteration 19: 3098685 diffs
Iteration 20: 3097573 diffs
Iteration 21: 3096237 diffs
Iteration 22: 3094954 diffs
Iteration 23: 3093587 diffs
Iteration 24: 3092337 diffs
Iteration 25: 3090977 diffs
Iteration 26: 3089728 diffs
Iteration 27: 3088422 diffs
Iteration 28: 3087178 diffs
Iteration 29: 3085998 diffs
Iteration 30: 3084782 diffs
Iteration 31: 3083546 diffs
Iteration 32: 3082413 diffs
Iteration 33: 3081182 diffs
Iteration 34: 3079909 diffs
Iteration 35: 3078655 diffs
Iteration 36: 3077425 diffs
Iteration 37: 3076038 diffs
Iteration 38: 3074605 diffs
Iteration 39: 3073184 diffs
Iteration 40: 3071655 diffs
```

```
Iteration 41: 3070231 diffs
Iteration 42: 3068771 diffs
Iteration 43: 3067317 diffs
Iteration 44: 3065971 diffs
Iteration 45: 3064598 diffs
Iteration 46: 3063220 diffs
Iteration 47: 3061893 diffs
Iteration 48: 3060544 diffs
Iteration 49: 3059126 diffs

Writing to file...
Success!


BENCHMARKING
Reading: 1.13 s
Processing: 174.15 s
Writing: 2.47 s



Reading from file...
Resolution: 2048x1536
Success!

ID: 0 Processing 383 lines Begin: 1 End: 384
ID: 1 Processing 383 lines Begin: 384 End: 767
ID: 2 Processing 384 lines Begin: 767 End: 1151
ID: 1 Width: 2048 Height:385
ID: 3 Processing 383 lines Begin: 1151 End: 1534
ID: 2 Width: 2048 Height:386

ID: 0 Width: 2048 Height:385
ID: 3 Width: 2048 Height:385
Iteration 0: 3135229 diffs
Iteration 1: 3129880 diffs
Iteration 2: 3125338 diffs
Iteration 3: 3123102 diffs
Iteration 4: 3121010 diffs
Iteration 5: 3119010 diffs
Iteration 6: 3117219 diffs
Iteration 7: 3115089 diffs
Iteration 8: 3113283 diffs
Iteration 9: 3111397 diffs
Iteration 10: 3109810 diffs
Iteration 11: 3108253 diffs
Iteration 12: 3106846 diffs
Iteration 13: 3105512 diffs
Iteration 14: 3104180 diffs
Iteration 15: 3103191 diffs
Iteration 16: 3102026 diffs
Iteration 17: 3101019 diffs
Iteration 18: 3099942 diffs
Iteration 19: 3098685 diffs
Iteration 20: 3097573 diffs
Iteration 21: 3096237 diffs
Iteration 22: 3094954 diffs
Iteration 23: 3093587 diffs
Iteration 24: 3092337 diffs
Iteration 25: 3090977 diffs
Iteration 26: 3089728 diffs
```

```
Iteration 27: 3088422 diffs
Iteration 28: 3087178 diffs
Iteration 29: 3085998 diffs
Iteration 30: 3084782 diffs
Iteration 31: 3083546 diffs
Iteration 32: 3082413 diffs
Iteration 33: 3081182 diffs
Iteration 34: 3079909 diffs
Iteration 35: 3078655 diffs
Iteration 36: 3077425 diffs
Iteration 37: 3076038 diffs
Iteration 38: 3074605 diffs
Iteration 39: 3073184 diffs
Iteration 40: 3071655 diffs
Iteration 41: 3070231 diffs
Iteration 42: 3068771 diffs
Iteration 43: 3067317 diffs
Iteration 44: 3065971 diffs
Iteration 45: 3064598 diffs
Iteration 46: 3063220 diffs
Iteration 47: 3061893 diffs
Iteration 48: 3060544 diffs
Iteration 49: 3059126 diffs

Writing to file...
Success!


BENCHMARKING
Reading: 1.12 s
Processing: 87.82 s
Writing: 2.41 s



Reading from file...
Resolution: 2048x1536
Success!

ID: 0 Processing 192 lines Begin: 1 End: 193
ID: 1 Processing 192 lines Begin: 193 End: 385
ID: 2 Processing 192 lines Begin: 385 End: 577
ID: 1 Width: 2048 Height:194
ID: 3 Processing 191 lines Begin: 577 End: 768
ID: 2 Width: 2048 Height:194
ID: 4 Processing 192 lines Begin: 768 End: 960
ID: 3 Width: 2048 Height:193
ID: 5 Processing 191 lines Begin: 960 End: 1151
ID: 4 Width: 2048 Height:194
ID: 6 Processing 192 lines Begin: 1151 End: 1343
ID: 5 Width: 2048 Height:193
ID: 7 Processing 191 lines Begin: 1343 End: 1534
ID: 6 Width: 2048 Height:194

ID: 7 Width: 2048 Height:193
ID: 0 Width: 2048 Height:194
Iteration 0: 3135229 diffs
Iteration 1: 3129880 diffs
Iteration 2: 3125338 diffs
Iteration 3: 3123102 diffs
Iteration 4: 3121010 diffs
```

```
Iteration 5: 3119010 diffs
Iteration 6: 3117219 diffs
Iteration 7: 3115089 diffs
Iteration 8: 3113283 diffs
Iteration 9: 3111397 diffs
Iteration 10: 3109810 diffs
Iteration 11: 3108253 diffs
Iteration 12: 3106846 diffs
Iteration 13: 3105512 diffs
Iteration 14: 3104180 diffs
Iteration 15: 3103191 diffs
Iteration 16: 3102026 diffs
Iteration 17: 3101019 diffs
Iteration 18: 3099942 diffs
Iteration 19: 3098685 diffs
Iteration 20: 3097573 diffs
Iteration 21: 3096237 diffs
Iteration 22: 3094954 diffs
Iteration 23: 3093587 diffs
Iteration 24: 3092337 diffs
Iteration 25: 3090977 diffs
Iteration 26: 3089728 diffs
Iteration 27: 3088422 diffs
Iteration 28: 3087178 diffs
Iteration 29: 3085998 diffs
Iteration 30: 3084782 diffs
Iteration 31: 3083546 diffs
Iteration 32: 3082413 diffs
Iteration 33: 3081182 diffs
Iteration 34: 3079909 diffs
Iteration 35: 3078655 diffs
Iteration 36: 3077425 diffs
Iteration 37: 3076038 diffs
Iteration 38: 3074605 diffs
Iteration 39: 3073184 diffs
Iteration 40: 3071655 diffs
Iteration 41: 3070231 diffs
Iteration 42: 3068771 diffs
Iteration 43: 3067317 diffs
Iteration 44: 3065971 diffs
Iteration 45: 3064598 diffs
Iteration 46: 3063220 diffs
Iteration 47: 3061893 diffs
Iteration 48: 3060544 diffs
Iteration 49: 3059126 diffs

Writing to file...
Success!


BENCHMARKING
Reading: 1.12 s
Processing: 44.5 s
Writing: 2.48 s



Reading from file...
Resolution: 2048x1536
Success!
```

```
ID: 0 Processing 96 lines Begin: 1 End: 97
ID: 1 Processing 96 lines Begin: 97 End: 193
ID: 2 Processing 96 lines Begin: 193 End: 289
ID: 1 Width: 2048 Height:98
ID: 3 Processing 96 lines Begin: 289 End: 385
ID: 2 Width: 2048 Height:98
ID: 4 Processing 96 lines Begin: 385 End: 481
ID: 3 Width: 2048 Height:98
ID: 5 Processing 96 lines Begin: 481 End: 577
ID: 4 Width: 2048 Height:98
ID: 6 Width: 2048 Height:98
ID: 6 Processing 96 lines Begin: 577 End: 673
ID: 7 Processing 96 lines Begin: 673 End: 769
ID: 5 Width: 2048 Height:98
ID: 8 Processing 96 lines Begin: 769 End: 865
ID: 7 Width: 2048 Height:98
ID: 8 Width: 2048 Height:98
ID: 9 Processing 96 lines Begin: 865 End: 961
ID: 10 Processing 96 lines Begin: 961 End: 1057
ID: 9 Width: 2048 Height:98
ID: 11 Processing 95 lines Begin: 1057 End: 1152
ID: 10 Width: 2048 Height:98
ID: 12 Processing 96 lines Begin: 1152 End: 1248
ID: 11 Width: 2048 Height:97
ID: 13 Processing 95 lines Begin: 1248 End: 1343
ID: 12 Width: 2048 Height:98
ID: 14 Processing 96 lines Begin: 1343 End: 1439
ID: 13 Width: 2048 Height:97
ID: 15 Processing 95 lines Begin: 1439 End: 1534
ID: 14 Width: 2048 Height:98

ID: 15 Width: 2048 Height:97
ID: 0 Width: 2048 Height:98
Iteration 0: 3135229 diffs
Iteration 1: 3129880 diffs
Iteration 2: 3125338 diffs
Iteration 3: 3123102 diffs
Iteration 4: 3121010 diffs
Iteration 5: 3119010 diffs
Iteration 6: 3117219 diffs
Iteration 7: 3115089 diffs
Iteration 8: 3113283 diffs
Iteration 9: 3111397 diffs
Iteration 10: 3109810 diffs
Iteration 11: 3108253 diffs
Iteration 12: 3106846 diffs
Iteration 13: 3105512 diffs
Iteration 14: 3104180 diffs
Iteration 15: 3103191 diffs
Iteration 16: 3102026 diffs
Iteration 17: 3101019 diffs
Iteration 18: 3099942 diffs
Iteration 19: 3098685 diffs
Iteration 20: 3097573 diffs
Iteration 21: 3096237 diffs
Iteration 22: 3094954 diffs
Iteration 23: 3093587 diffs
Iteration 24: 3092337 diffs
Iteration 25: 3090977 diffs
Iteration 26: 3089728 diffs
Iteration 27: 3088422 diffs
```

```
Iteration 28: 3087178 diffs
Iteration 29: 3085998 diffs
Iteration 30: 3084782 diffs
Iteration 31: 3083546 diffs
Iteration 32: 3082413 diffs
Iteration 33: 3081182 diffs
Iteration 34: 3079909 diffs
Iteration 35: 3078655 diffs
Iteration 36: 3077425 diffs
Iteration 37: 3076038 diffs
Iteration 38: 3074605 diffs
Iteration 39: 3073184 diffs
Iteration 40: 3071655 diffs
Iteration 41: 3070231 diffs
Iteration 42: 3068771 diffs
Iteration 43: 3067317 diffs
Iteration 44: 3065971 diffs
Iteration 45: 3064598 diffs
Iteration 46: 3063220 diffs
Iteration 47: 3061893 diffs
Iteration 48: 3060544 diffs
Iteration 49: 3059126 diffs

Writing to file...
Success!


BENCHMARKING
Reading: 1.13 s
Processing: 36.89 s
Writing: 2.12 s



Reading from file...
Resolution: 2048x1536
Success!

ID: 0 Processing 48 lines Begin: 1 End: 49
ID: 1 Processing 48 lines Begin: 49 End: 97
ID: 2 Processing 48 lines Begin: 97 End: 145
ID: 1 Width: 2048 Height:50
ID: 3 Processing 48 lines Begin: 145 End: 193
ID: 2 Width: 2048 Height:50
ID: 4 Processing 48 lines Begin: 193 End: 241
ID: 3 Width: 2048 Height:50
ID: 5 Processing 48 lines Begin: 241 End: 289
ID: 4 Width: 2048 Height:50
ID: 6 Processing 48 lines Begin: 289 End: 337
ID: 7 Processing 48 lines Begin: 337 End: 385
ID: 6 Width: 2048 Height:50
ID: 5 Width: 2048 Height:50
ID: 8 Processing 48 lines Begin: 385 End: 433
ID: 7 Width: 2048 Height:50
ID: 8 Width: 2048 Height:50
ID: 10 Width: 2048 Height:50
ID: 9 Processing 48 lines Begin: 433 End: 481
ID: 11 Width: 2048 Height:50
ID: 10 Processing 48 lines Begin: 481 End: 529
ID: 9 Width: 2048 Height:50
ID: 11 Processing 48 lines Begin: 529 End: 577
```

```
ID: 12 Processing 48 lines Begin: 577 End: 625
ID: 13 Processing 48 lines Begin: 625 End: 673
ID: 12 Width: 2048 Height:50
ID: 14 Processing 48 lines Begin: 673 End: 721
ID: 13 Width: 2048 Height:50
ID: 15 Processing 48 lines Begin: 721 End: 769
ID: 14 Width: 2048 Height:50
ID: 16 Processing 48 lines Begin: 769 End: 817
ID: 15 Width: 2048 Height:50
ID: 17 Processing 48 lines Begin: 817 End: 865
ID: 16 Width: 2048 Height:50
ID: 18 Processing 48 lines Begin: 865 End: 913
ID: 17 Width: 2048 Height:50
ID: 19 Processing 48 lines Begin: 913 End: 961
ID: 18 Width: 2048 Height:50
ID: 20 Processing 48 lines Begin: 961 End: 1009
ID: 19 Width: 2048 Height:50
ID: 21 Processing 48 lines Begin: 1009 End: 1057
ID: 20 Width: 2048 Height:50
ID: 22 Processing 48 lines Begin: 1057 End: 1105
ID: 21 Width: 2048 Height:50
ID: 23 Processing 48 lines Begin: 1105 End: 1153
ID: 22 Width: 2048 Height:50
ID: 24 Processing 48 lines Begin: 1153 End: 1201
ID: 25 Processing 48 lines Begin: 1201 End: 1249
ID: 24 Width: 2048 Height:50
ID: 26 Processing 48 lines Begin: 1249 End: 1297
ID: 25 Width: 2048 Height:50
ID: 23 Width: 2048 Height:50
ID: 27 Processing 47 lines Begin: 1297 End: 1344
ID: 26 Width: 2048 Height:50
ID: 28 Processing 48 lines Begin: 1344 End: 1392
ID: 29 Processing 47 lines Begin: 1392 End: 1439
ID: 28 Width: 2048 Height:50
ID: 30 Processing 48 lines Begin: 1439 End: 1487
ID: 29 Width: 2048 Height:49
ID: 31 Processing 47 lines Begin: 1487 End: 1534
ID: 27 Width: 2048 Height:49
ID: 30 Width: 2048 Height:50

ID: 0 Width: 2048 Height:50
ID: 31 Width: 2048 Height:49
Iteration 0: 3135229 diffs
Iteration 1: 3129880 diffs
Iteration 2: 3125338 diffs
Iteration 3: 3123102 diffs
Iteration 4: 3121010 diffs
Iteration 5: 3119010 diffs
Iteration 6: 3117219 diffs
Iteration 7: 3115089 diffs
Iteration 8: 3113283 diffs
Iteration 9: 3111397 diffs
Iteration 10: 3109810 diffs
Iteration 11: 3108253 diffs
Iteration 12: 3106846 diffs
Iteration 13: 3105512 diffs
Iteration 14: 3104180 diffs
Iteration 15: 3103191 diffs
Iteration 16: 3102026 diffs
Iteration 17: 3101019 diffs
Iteration 18: 3099942 diffs
```

```
Iteration 19: 3098685 diffs
Iteration 20: 3097573 diffs
Iteration 21: 3096237 diffs
Iteration 22: 3094954 diffs
Iteration 23: 3093587 diffs
Iteration 24: 3092337 diffs
Iteration 25: 3090977 diffs
Iteration 26: 3089728 diffs
Iteration 27: 3088422 diffs
Iteration 28: 3087178 diffs
Iteration 29: 3085998 diffs
Iteration 30: 3084782 diffs
Iteration 31: 3083546 diffs
Iteration 32: 3082413 diffs
Iteration 33: 3081182 diffs
Iteration 34: 3079909 diffs
Iteration 35: 3078655 diffs
Iteration 36: 3077425 diffs
Iteration 37: 3076038 diffs
Iteration 38: 3074605 diffs
Iteration 39: 3073184 diffs
Iteration 40: 3071655 diffs
Iteration 41: 3070231 diffs
Iteration 42: 3068771 diffs
Iteration 43: 3067317 diffs
Iteration 44: 3065971 diffs
Iteration 45: 3064598 diffs
Iteration 46: 3063220 diffs
Iteration 47: 3061893 diffs
Iteration 48: 3060544 diffs
Iteration 49: 3059126 diffs

Writing to file...
Success!


BENCHMARKING
Reading: 1.11 s
Processing: 33.94 s
Writing: 2.04 s
```