Program description

The program takes as input two images in pgm format and a corresponding look-up table for each image. One image is the view from the left and another - from the right.

The program workflow consists of the following parts:

1. Reading input data
    a) Reading input for left image
    b) Reading input for right image
2. Processing images
    a) Performing rectification for both input images
    b) Performing census transform for rectified images
3. Saving output data on persistent storage

Reading input data

Description. Pgm images are stored in binary format and parsed according to the format specification https://en.wikipedia.org/wiki/Netpbm#PGM_example. Each image has size (width, length) and each pixel data is represented as *unit8_t* number. For each loaded image look-up table is loaded. For each pixel *i* we also store two *int16_t* values that will be helpful for image rectification. All the input data is firstly read to **CPU RAM** and then copied to **GPU RAM**. In case if required files are missing or some other error occurred, exception is thrown and the error description is written to *cerr* stream and 1 is returned from *main*.

Assumptions. It is assumed that input data is located in ‘res’ folder and named *left.pgm*, *left.blt* and *right.pgm, right.blt* for left and right images respectively. The images are assumed to be of the same size.

Processing images.

Description. After images data is successfully parsed and copied to **GPU RAM**, the second step - image processing, is performed. For each image we create a separate **CUDA** stream in order to perform independent and parallel processing of each image. For each image we firstly perform rectification and then for rectified image - census transform. We shall describe each procedure in detail in the following sections.

Rectification.

Kernel parameters. Rectification kernel has the following parameters that will be further optimized for performance: *threadsPerBlock* - number of threads per **CUDA** block in rectification kernel and *num_threads* - total number of threads for rectification kernel.

Description. Each **CUDA** thread processes several pixels depending on the kernel parameters. This is done in order to fully utilize **GPU** resources and increase performance. For each output pixel we firstly load two corresponding values from look-up table. These values contain

indices of input picture for rectification and information for calculating weights for each input pixel. In other words, for each output pixel (i, j) the look-up table provides indices (m, n), (m+1, n), (m, n+1), (m+1, n+1) and coefficients k0, k1, k2, k3. Then output value is calculated as out[i, j] = k0 * in[m, n] + k1 * in[m+1, n] + k2 * in[m, n+1] + k3 * in[m+1, n+1]. The output picture is provided to the census transform.

Census.

Kernel parameters. Census kernel has the following parameters that will be further optimized for performance: *threadsPerBlockX* and *threadsPerBlockY* - number of threads in block in x and y direction respectively.

Description. Each **CUDA** thread processes exactly one pixel. For each input pixel (i, j) it counts the number of pixels in the range [i-2,i+2]x[j-2,j+2] greater than the input pixel and writes this value to output pixel. In order to increase performance this kernel uses shared memory to store each input pixel needed for given block.

Performance analysis.

Performance analysis is done for **GPU** - **NVIDIA GTX 1050ti**, **CPU** - **Intel i7-7700HQ**. We firstly optimize cuda kernel parameters and then compare **CPU** and **GPU** performance.

census		**blockx**			
		8	16	32	64
blocky	4		104.51	104.48	101.54
	8	104.94	102.08	102.56	101.47
	16	102.37	110.18	103.49	110.66
	32	101.41	102.27	114.21	
	64	103.33	112.86		

Table 1. Comparison of performance for different kernel parameters for census kernel.

rectification		**threadsPer Block**				
		64	128	256	512	1024
num_threads	2048	104.76	109.22	104.86	106.85	112.54
	4096	61.47	61.70	62.08	62.18	63.07
	8192	40.64	41.92	40.86	45.79	43.14
	16384	39.62	39.46	39.92	40.54	39.78
	32768	36.35	36.93	36.03	36.67	37.06
	65536	35.65	35.42	34.98	35.58	36.80
	131072	34.85	34.46	35.01	35.49	38.02

Table 2. Comparison of performance for different kernel parameters for rectification kernel.

Performance results for census and rectification are shown in tables 1 and two respectively where values mean microseconds per kernel call. Measurements are done in **NVIDIA** profiling tool called **Nsight Compute**. Optimal parameters are highlighted with green color.

	Time
CPU	42550
GPU	255

Table 3. Comparison of algorithm performance for CPU and GPU implementation.

Comparison of performance for **CPU** and **GPU** algorithms is shown in table 3 where time is measured in microseconds as well. For **GPU** only algorithm time is measured without taking into account **CPU**-**GPU** data transfer. In order to reduce measurement noise the algorithm time is taken as minimum from 1000 runs for **GPU** and 100 runs for **CPU** algorithm.

Error handling

In case if some **CUDA**-related error occurred during kernel call, it is checked and the corresponding error text is written to cerr stream and 1 is returned from main.

Saving output data on persistent storage

Description. If all previous steps were successfully performed, output images called *left_processed.pgm* and *right_processed.pgm* respectively are copied to the **CPU RAM** and then saved to build folder. In case if for some reason files cannot be created, exception is thrown and the corresponding message is written to *cerr* and 1 is returned from *main*.

For two processed images we also compute hamming distance (number of different pixels). This computation is done also on gpu via thrust library utilities. Then, this distance is saved to *different_pixels.txt* file.