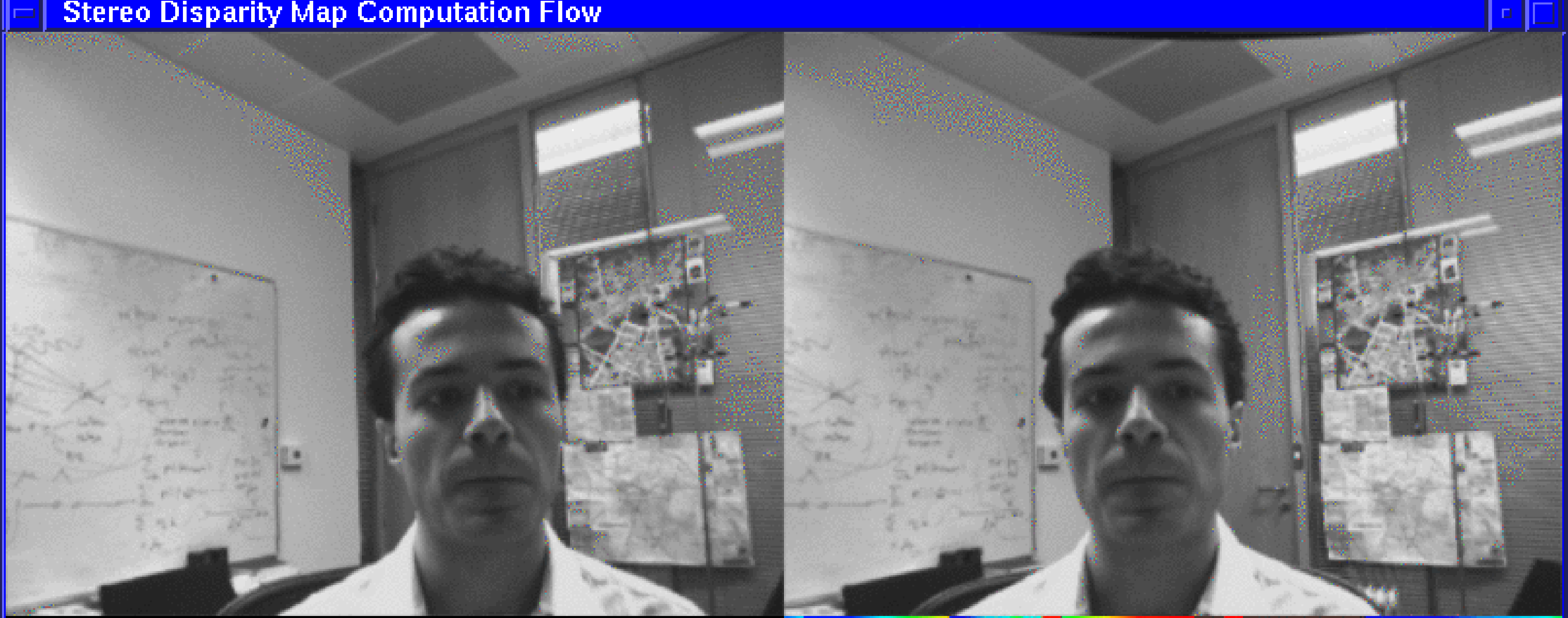
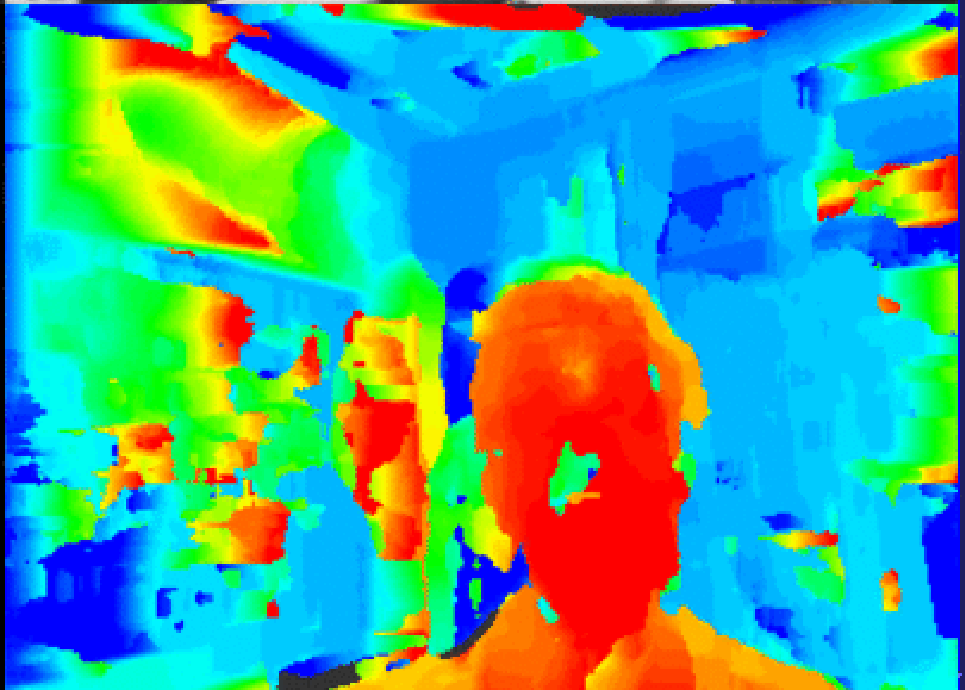


Genetically Improved CUDA C++ Software

W. B. Langdon

Computer Science, UCL, London

26.4.2014



Genetically Improved CUDA C++ Software

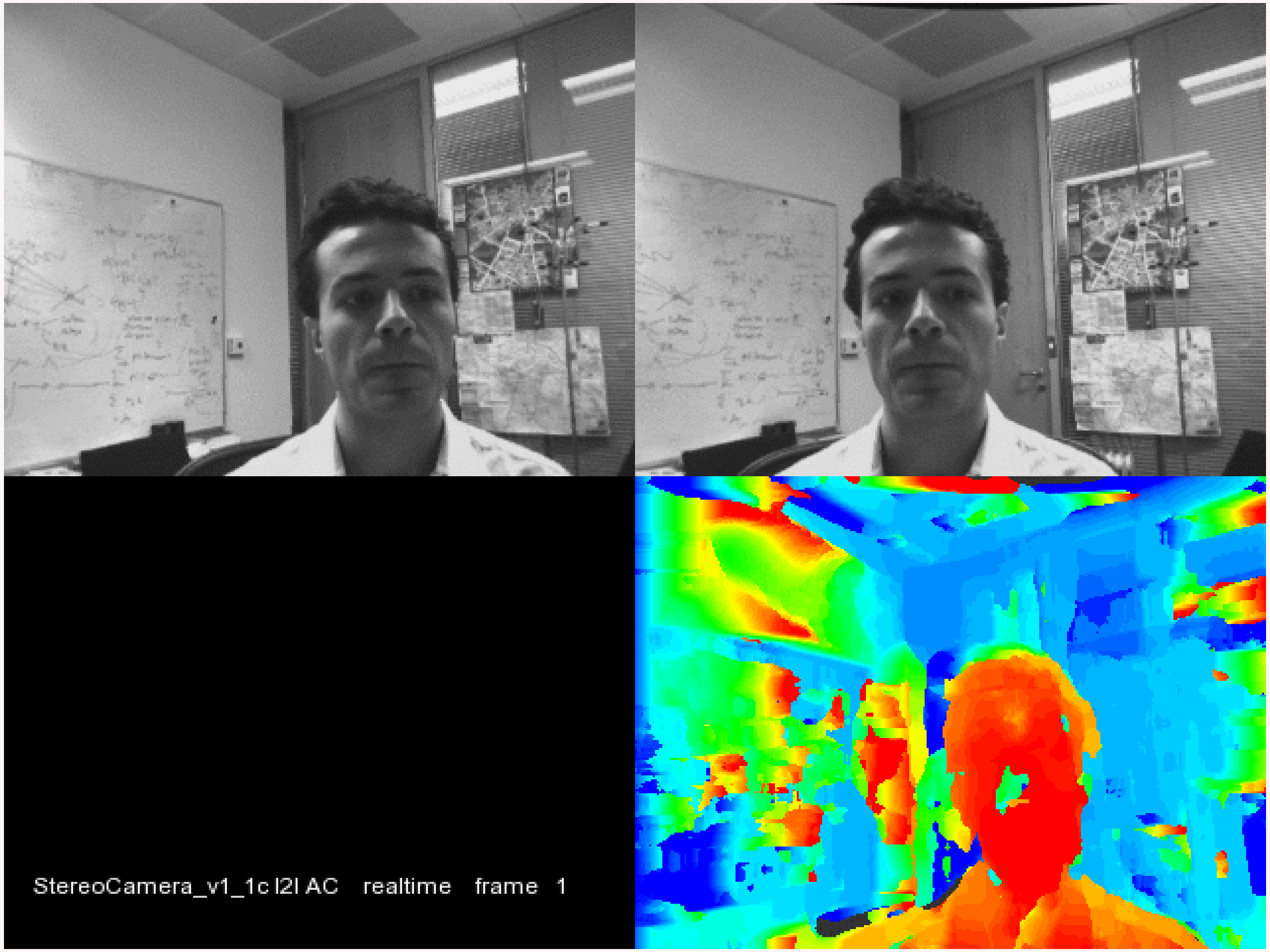
W. B. Langdon

Centre for Research on Evolution, Search and Testing

Computer Science, UCL, London

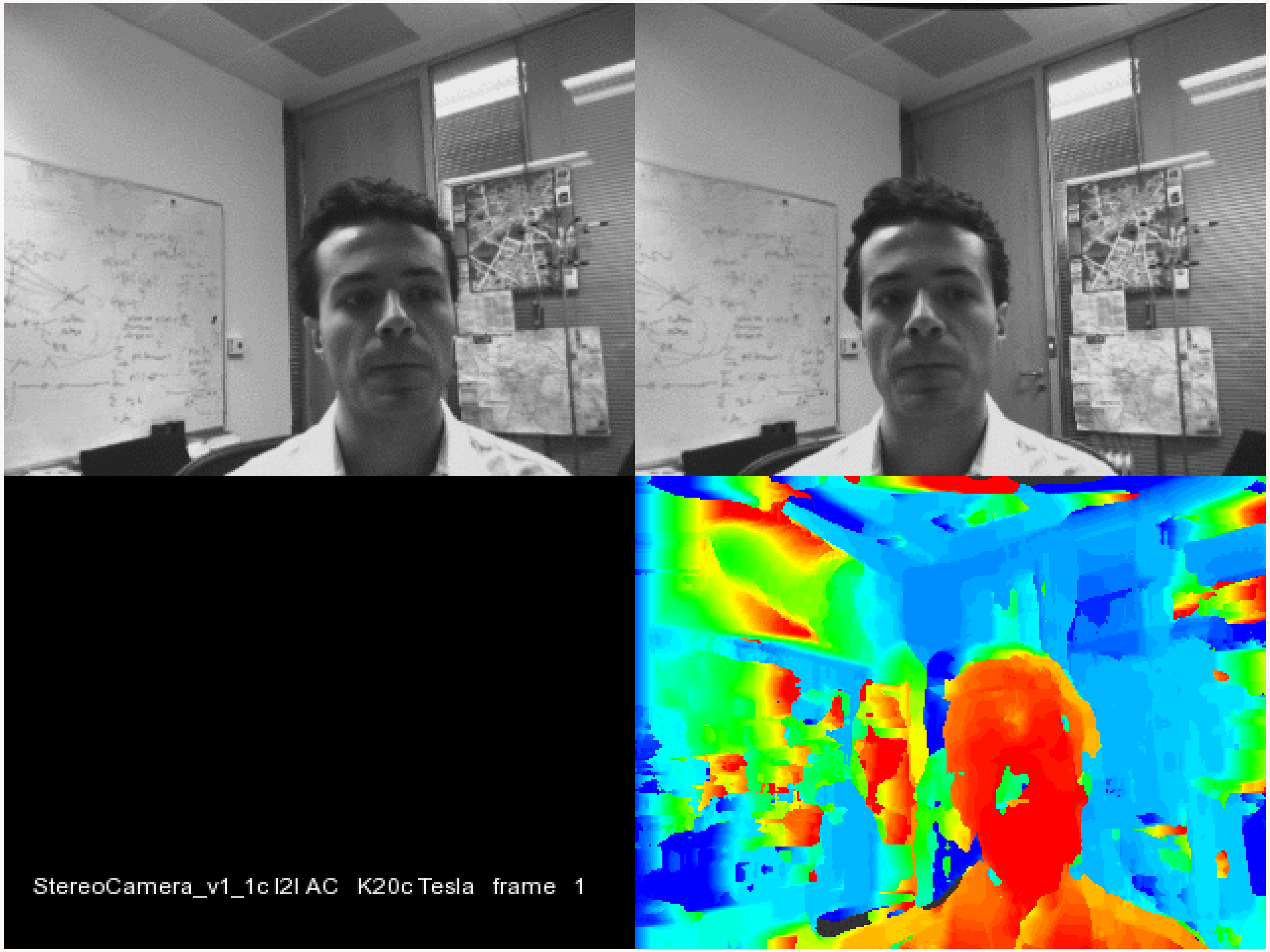
GISMOE: Genetic Improvement of Software for Multiple Objectives

Real time stereo vision
200 frames 30fps=6.7sec



Video

GP improved Tesla K20c 200 frames 710microsecs each



Video

Genetic Programming to Improve Legacy Stereo Image Processing Software

- Target software: StereoCamera
- Target hardware: 6 graphics cards(GPUs)
- Manual and auto changes. Tuning
- GP code changes
- Results: 1.05 to 6.8 times faster

CUDA is nVidia’s C++ dialect for nVidia’s hardware

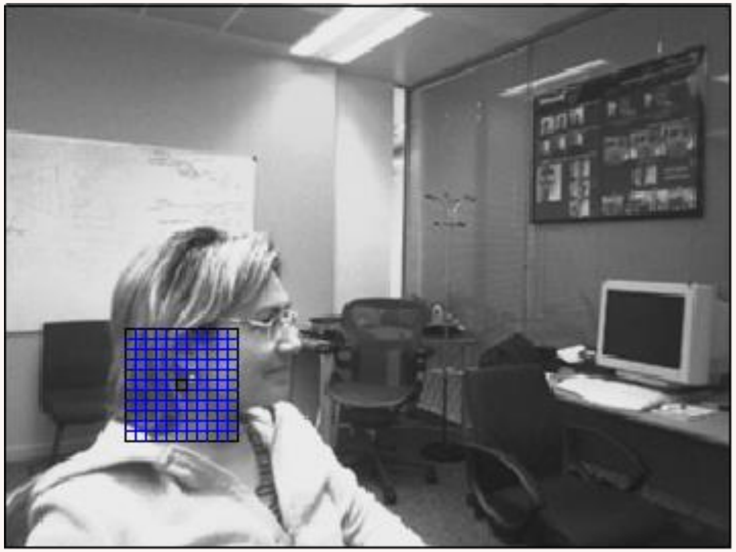
GP Automatic Coding

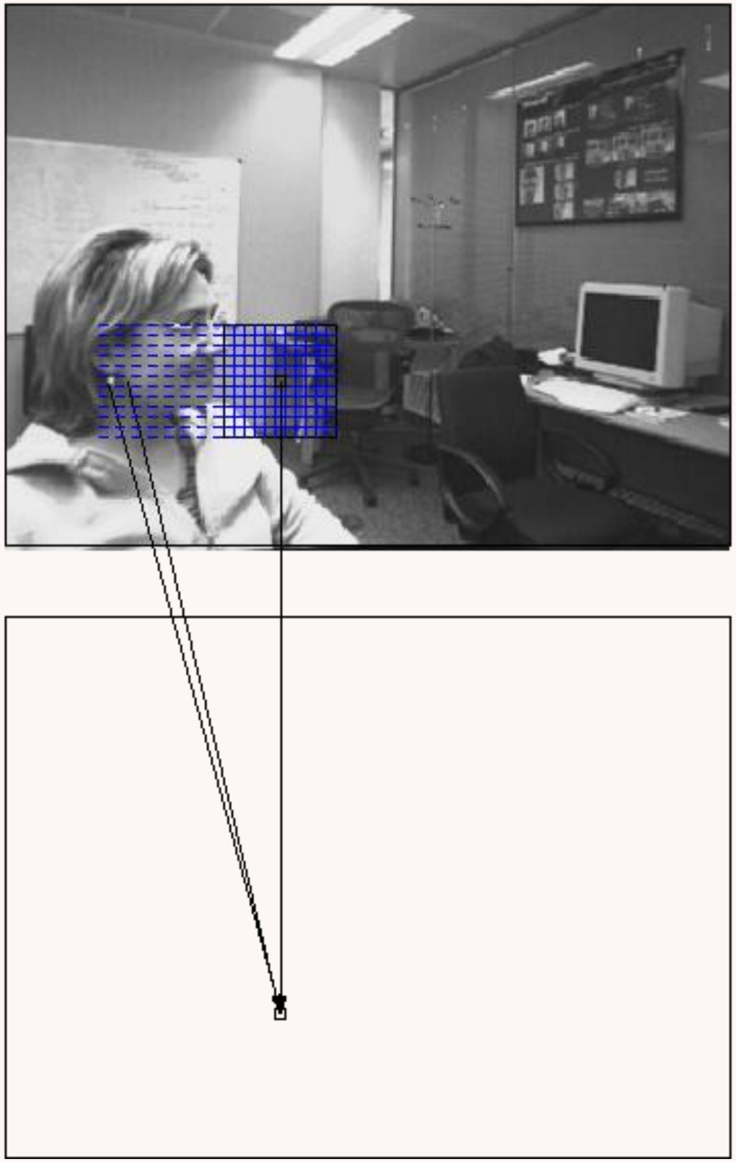
- Target non-trivial open source system stereo image processing GPU code.
- Tailor existing system for specific use:
  - microsoft I2I 320×240 grey stereo pairs
  - 6 different GPUs (16 to 2496 cores)
- Run existing code and GP back-to-back. Fitness:
  - difference orginal code's answer and GP's
  - speed of evolved code
- Remove bloat

Why StereoCamera

- StereoCamera is high quality CUDA code
- Since written (2007) changes to both hardware and software
- Target stereoKernel
    - Full kernel 276 lines CUDA
    - Expanded existing code by hand
        - XHALO, DPER
- Tailor existing system for specific cases:
- Microsoft I2I image database

CUDA GPU stereoKernel C++ code





For each left pixel find horizontal offset of best corresponding right pixel

Not to scale.

Min Sum (diff^2 11×11)

Six Types of nVidia GPUs Parallel Graphics Hardware

Name	year		MP	Cores	Clock
Quadro NVS 290	2007	1.1	2 × 8	16	0.92 GHz
GeForce GTX 295	2009	1.3	30 × 8	240	1.24 GHz
Tesla T10	2009	1.3	30 × 8	240	1.30 GHz
Tesla C2050	2010	2.0	14 × 32	448	1.15 GHz
GeForce GTX 580	2010	2.0	16 × 32	512	1.54 GHz
Tesla K20c	2012	3.5	13 × 192	2496	0.71 GHz

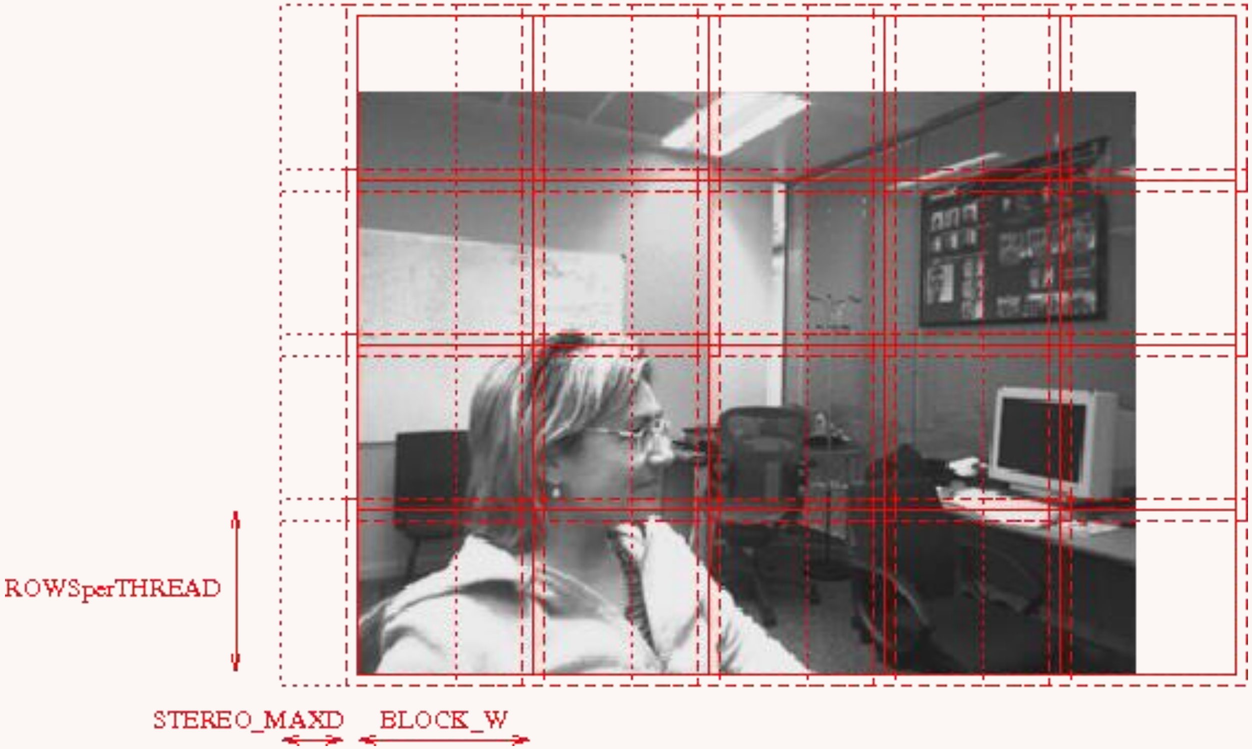
Evolving stereoKernel

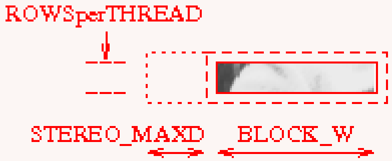
- Convert code to grammar
- Grammar used to control modifications to code
- Genetic programming manipulates patches
  - Copy/delete/insert lines of existing code
  - Patch is small
  - New kernel source is syntactically correct
  - Essentially no compilation errors

Before GP

- As a result of initial GP, we discovered
  - 2 Objectives: low error and fast, too different
  - Existing code badly tuned
- Therefore:
  - Abandoned multi-objective GP (NSGA-II fails)
    - one objective: go faster with zero error
  - Pre and post tune 2 or 3 key parameters
  - GP now optimises both configuration parameters (12) and code (variable length)

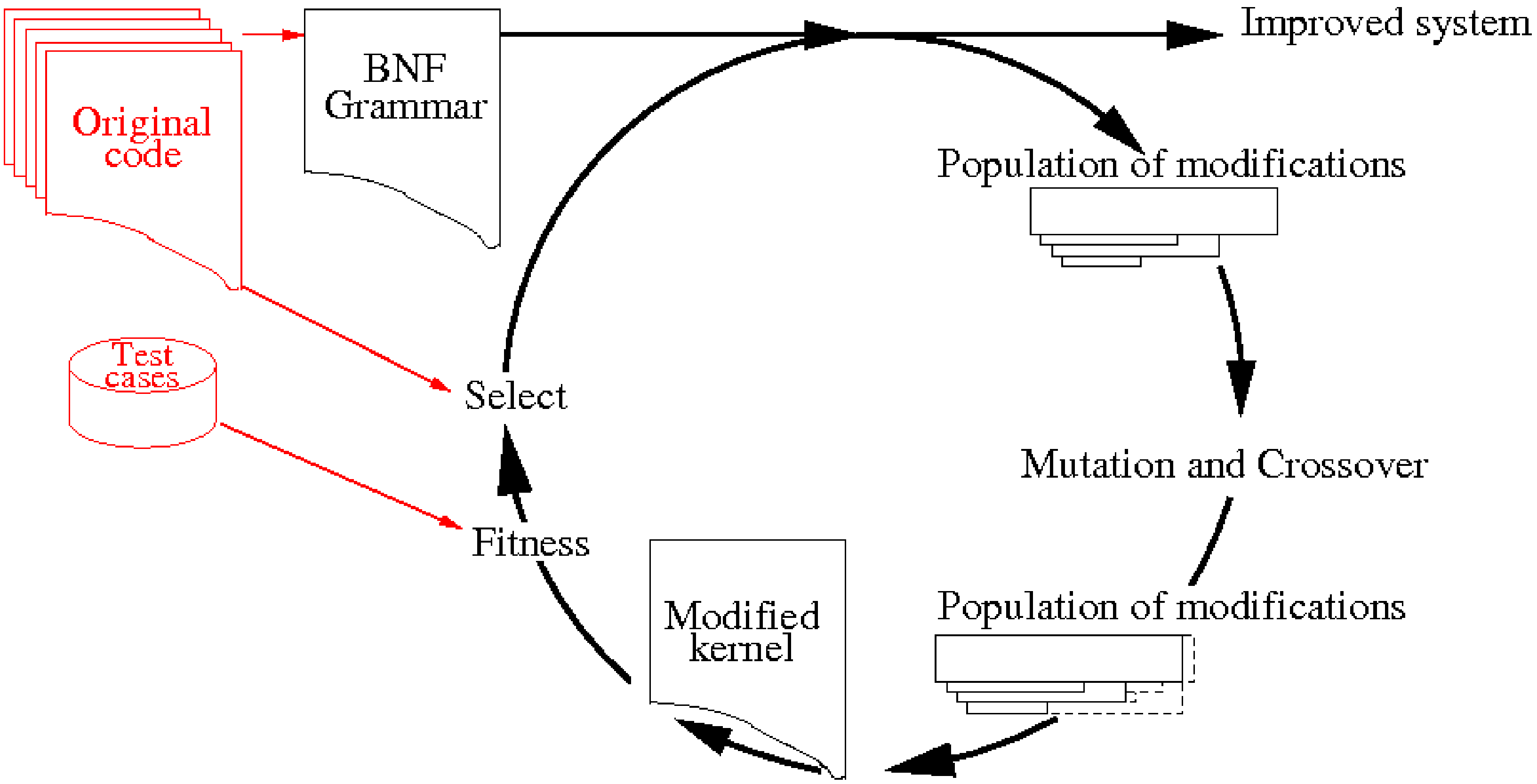
Images split into tiles
Each tile run in parallel





STEREO_MAXD = 50
BLOCK_W = 64 (5 columns)
ROWSperTHREAD = 40 (6 rows) **tune** → 5 (48 rows) 48×5=240 (was 30)

GP Evolving Patches to CUDA



BNF Grammar for code changes

```
if(X < width && Y < height)
   {
    dmin = d;
```

Lines 326-328 Stereo_union.cuh

```
<KStereo.cuh_326>   ::=    " if" <IF_KStereo.cuh_326> " \n"
#"if
<IF_KStereo.cuh_326>::=    "(X < width && Y < height)"
<KStereo.cuh_327>   ::=    "{\n"
<KStereo.cuh_328>   ::=    "" <_KStereo.cuh_328> "\n"
#other
<_KStereo.cuh_328>  ::=    "dmin = d;"
```

Fragment of Grammar (Total 424 rules)

Grammar Rule Types

- Type indicated by rule name
- Replace rule only by another of same type
- 55 statement (eg assignment, **Not** declaration)
- 27 IF
- <KStereo.cuh_356> ::= " if" <IF_KStereo.cuh_356> " \n"
- <IF_KStereo.cuh_356> ::= "(dblockIdx==0)"
- 10 for1, for2, for3
- <KStereo.cuh_221> ::= <pragma_KStereo.cuh_221> "for(" <for1_KStereo.cuh_221> ";” "OK()&&” <for2_KStereo.cuh_221> ";" <for3_KStereo.cuh_221> ") \n"
- <pragma_KStereo.cuh_221> ::= ""
- <for1_KStereo.cuh_221> ::= "i = 0"
- <for2_KStereo.cuh_221> ::= "i<=(2*RADIUS_H)"
- <for3_KStereo.cuh_221> ::= "i++"
- CUDA types

Representation

- 12 fixed configuration parameter: variable length list of grammar patches.
- uniform crossover on 12 fixed: tree like 2pt crossover on grammar changes.
- mutation change one of 12 configuration params or adds one randomly chosen grammar change
- 3 possible grammar changes:
  - Delete line of source code (or replace by “”, 0)
  - Replace with line of stereoKernel (same type)
  - Insert a copy of another stereoKernel line

Example Mutating Grammar

```
<IF_KStereo.cuh_326>::=    "(X < width && Y < height)"
<IF_KStereo.cuh_154>::=    "(dblockIdx==0)"
```

2 lines from grammar

```
<IF_KStereo.cuh_326><IF_KStereo.cuh_154>
```

Fragment of list of mutations

Says replace line 326 by line 154

```
if(X < width && Y < height)
```

Original code

```
if(dblockIdx==0)
```

New code

StereoKernel Fitness Function

- Fitness run twice
  - – Run with error checking
    - CUDA memcheck (GPU dependent)
    - Force termination of excessive for loop iterations
  - – If ok, run again for accurate timing

Fitness

- Run patched stereoKernel on 1 example image
  - 96% compile and run ok
  - Compare results with original answer
  - Sort population by
    - Error (actually only selected zero error)
    - Kernel GPU clock ticks (minimise)
  - Select top half of population.
- Mutate, crossover to give 2 children per parent.
- Repeat 50 generations
- Remove bloat
- Automatic tune again

Results

- GP run failed on old hardware
  - memcheck problem?
- Patched/Tuned code run on 3010 images
- New kernels work for **all**. **Always** faster.
- Speed up depends on image size
  - All 320×240 speed up as training cases
  - Other sizes speed up some times less

stereoKernel Results

GPU name	Original	Pretuned	Ratio	GP	Speedup
Quadro NVS 290	27.0ms	26.0ms			1.05
GeForce GTX 295	5.4ms	1.5ms			3.6
Tesla T10	5.3ms	1.4ms	3.7	1.4ms	3.9
Tesla C2050	4.6ms	3.0ms	1.5	1.1ms	4.1
GeForce GTX 580	3.1ms	1.6ms	1.9	0.7ms	4.2
Tesla K20c	4.4ms	1.8ms	2.4	0.6ms	6.8

Milliseconds to process 320x240 stereo image pairs.
Averages over 2516 pairs.

NVS290 and GTX 295 hardware locked up in GP run.

GP can Improve Software

- Existing code provides
  - It is its own defacto specification
  - High quality starting code
  - Framework for both:
    - Functional fitness: does evolve code give right answers? (unlimited number of test cases)
    - Performance: how fast, how much power, how reliable,…
- Evolution has tuned old code for six very different graphics hardware and new software.

END

http://www.cs.ucl.ac.uk/staff/W.Langdon/ http://www.epsrc.ac.uk/ EPSRC

Tesla K20c 6.8 times faster Fixed parameters

Non-default configuration	value
ROWSperTHREAD	5
BLOCK_W	64
DPER	enabled
dperblock	2
XHALO	enabled
STORE_disparityMinSSD	SHARED
STORE_disparityPixel	SHARED

Evolution decided to enable DPER and XHALO and move `disparityMinSSD` and `disparityPixel` from global memory to shared memory.

ROWSperTHREAD, BLOCK_W and dperblock set by post evolution auto tuner

Tesla K20c 6.8 times faster Code changes

Remove CUDA code	New CUDA code
int * __restrict__ disparityMinSSD,	
volatile extern __attribute__ ((shared)) int col_ssd[];	extern __attribute__ ((shared)) int col_ssd[];
volatile int* const reduce_ssd = &col_ssd[(64)*2 -64];	int* const reduce_ssd = &col_ssd[(64)*2 -64];
	#pragma unroll 11
if(X < width && Y < height)	if(dblockIdx==0)
__syncthreads();	
	#pragma unroll 3

Parameter disparityMinSSD no longer needed as made shared (ie not global)

All volatile removed

Two #pragma inserted

if() replaced

__syncthreads() removed