

PULP PLATFORM

Open Source Hardware, the way it should be!

Bitcraze Workshop: Hands-on Session 2

Image acquisition and parallel image filter

**Lorenzo Lamberti, *Hanna Müller*, Vlad Niculescu, Manuele Rusci,
Daniele Palossi**



<http://pulp-platform.org>



[@pulp_platform](https://twitter.com/pulp_platform)



https://www.youtube.com/pulp_platform

The AI-Deck

Crazyflie (STM32)



Radio:
Nordic BTLE



nRF51 2.4GHz
Data rate: 0,25/1/2 Mbit/s

UART Link

Data rate: 1 Mbit/s

Radio dongle



Wi-Fi card

Radio:
NINA Wi-Fi



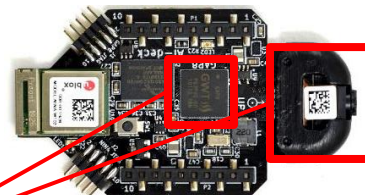
NINA-W102 2.4 GHz
Data rate: 6-54 Mbit/s

Crazyflie + AI-Deck



Hands-on 2: GAP8
programming & camera

AI-Deck (GAP8)



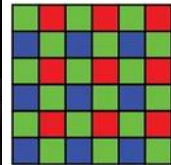


Hands-on: Image acquisition and filtering

1. git clone https://github.com/bitcraze/Aldeck_examples
2. set up your gap-sdk (source configs/ai_deck.sh)
3. Go to GAP8/image_processing_examples/simple_kernel_example
4. Compile and run the code (make clean all run platform=board or gap_run in the VM)
5. You can configure some flags in the Makefile

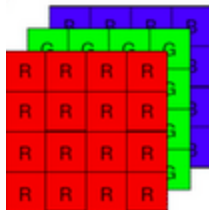
First: execution flow using demosaicking on the fabric controller as example
Then: parallelization with inverting an image on the cluster.

The code is simplified on the slides (but functional)



**Demosaicking
Fabric
controller**

**Inverting
Fabric
controller**



**Demosaicking
Cluster**

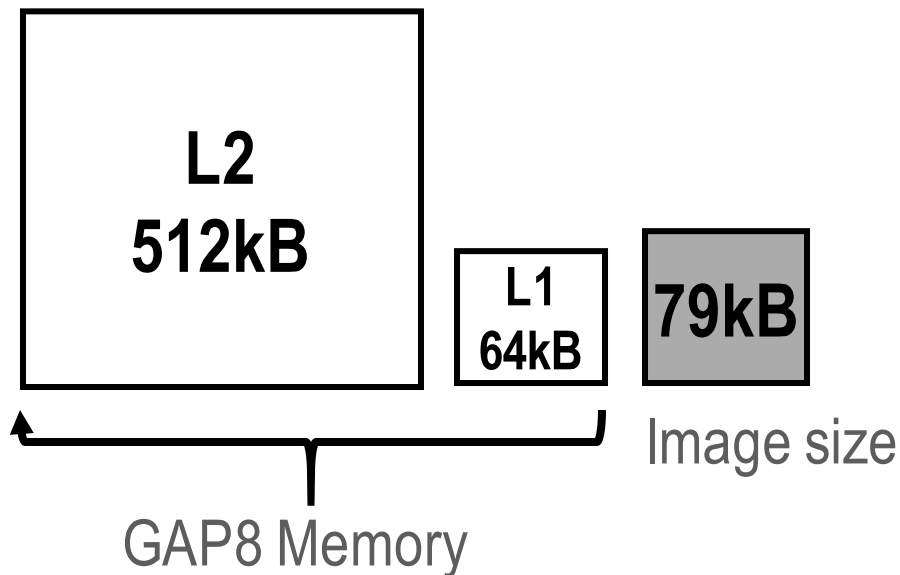
**Inverting
Cluster**





Hands-on: Image acquisition and filtering

Before we start, let's think about memory:
How many QVGA images could you have on
GAP8 at the same time?
Does it matter if they are colored or grey? Hint:
GAP8 L2 Memory: 512kB



Not even a single grey scale one on L1.
6 grey scale or 2 RGB in L2 – BUT do not forget,
you also need space for the code in L2!



Hands-on: Image acquisition and filtering

```

16 #include "pmsis.h"
17 #include "bsp/bsp.h"
18 #include "bsp/camera.h"
19 #include "bsp/camera/himax.h"
20
21 #include "gaplib/ImgIO.h"
22
23 #include "img_proc.h"
24
25 #define WIDTH 324
26 #ifdef QVGA_MODE
27 #define HEIGHT 244
28 #else
29 #define HEIGHT 324
30 #endif
31 #define BUFF_SIZE (WIDTH*HEIGHT)
32
33 PI_L2 unsigned char *buff;
34 PI_L2 unsigned char *buff_demosaick;
35
36
37 static struct pi_device camera;
38 static volatile int done;

```

```

40
41 static void handle_transfer_end(void)
42 {
43     done = 1;
44 }
45
46 static int open_camera(struct pi_dev
47 {
48     printf("Opening Himax camera\n");
49     struct pi_himax_conf cam_conf;
50     pi_himax_conf_init(&cam_conf);
51
52     #if defined(QVGA_MODE)
53     cam_conf.format = PI_CAMERA_QVGA
54     #endif
55
56     pi_open_from_conf(device, &cam_c
57     if (pi_camera_open(device))
58         return -1;
59     pi_camera_control(device, PI_CAM
60
61     return 0;
62 }

```

```

16 #include "pmsis.h"
17 #include "bsp/bsp.h"
18 #include "bsp/camera.h"
19 #include "bsp/camera/himax.h"
20
21 #include "gaplib/ImgIO.h"
22
23 #include "img_proc.h"
24
25 #define WIDTH 324
26 #ifdef QVGA_MODE
27 #define HEIGHT 244
28 #else
29 #define HEIGHT 324
30 #endif
31 #define BUFF_SIZE (WIDTH*HEIGHT)
32
33 PI_L2 unsigned char *buff;
34 PI_L2 unsigned char *buff_demosaick;
35
36
37 static struct pi_device camera;
38 static volatile int done;
39
40

```

Include drivers

Include image IO library

Include own demosaicking function

Define acquisition size

Define variables – place buffer in L2

```

120 #ifdef ASYNC_CAPTURE
121 // Start up async capture task
122 done = 0;
123 pi_task_t task;
124 pi_camera_capture_async(&camera, buff, BUFF_SIZE, pi_task_callback(&task, handle_transfer_end, NULL));
125 #endif

```

```

1 APP = test
2 APP_SRCS += test.c $(GAP_LIB_PATH)/img_io/ImgIO.c img_proc.c
3 APP_INC += $(GAP_LIB_PATH)/include
4
5 APP_CFLAGS += -O3 -g
6
7
8 PMSIS_OS ?= pulp_os
9
10 APP_CFLAGS += -DASYNC_CAPTURE
11 APP_CFLAGS += -DQVGA_MODE
12 APP_CFLAGS += -DCOLOR_IMAGE
13
14
15 clean::
16     rm -rf img_raw.ppm img_color.ppm img_gray.ppm
17
18 include $(RULES_DIR)/pmsis_rules.mk

```

```

159 {
160     printf("\n\t*** PMSIS Camera Example ***\n\n");
161     return pmsis_kickoff((void *) test_camera);
162 }

```



```
#include <stdio.h>
#include "asp.h"
#include "asp/comp/himax.h"
#include "imageIO.h"
#include "own_dmosaicking.h"
```

Define variables – place buffer in L2

Set up OS, then jump to test_camera

Hands-on: Image acquisition and filtering

Include drivers

Include image IO library

Include own demosaicking function

Define acquisition size

Define variables

Open and initialize camera

```

65 int test_camera()
66 {
67     printf("Entering main controller\n");
68
69     #ifdef ASYNC_CAPTURE
70     printf("Testing async camera capture\n");
71
72     #else
73     printf("Testing normal camera capture\n");
74     #endif
75
76     // Open the Himax camera
77     if (open_camera(&camera))
78     {
79         printf("Failed to open camera\n");
80         pmsis_exit(-1);
81     }

```

```

83 // Rotate camera orientation
84 uint8_t set_value=3;
85 uint8_t reg_value;
86
87
88 pi_camera_reg_set(&camera, IMG_ORIENTATION, &set_value);
89 pi_camera_reg_get(&camera, IMG_ORIENTATION, &reg_value);
90 printf("img orientation id\n", reg_value);

```

```

46 static int open_camera(struct pi_device *device)
47 {
48     printf("Opening Himax camera\n");
49     struct pi_himax_conf cam_conf;
50     pi_himax_conf_init(&cam_conf);
51
52     #if defined(QVGA_MODE)
53     cam_conf.format = PI_CAMERA_QVGA;
54     #endif
55
56     pi_open_from_conf(device, &cam_conf);
57     if (pi_camera_open(device))
58     {
59         return -1;
60     }
61     pi_camera_control(device, PI_CAMERA_CMD_AEG_INIT, 0);
62     return 0;

```

```

120 #ifdef ASYNC_CAPTURE
121 // Start up async capture task
122 done = 0;
123 pi_task_t task;
124 pi_async_capture_conf conf;
125 #endif
126
127 int test_camera()
128 {
129     printf("Entering main controller\n");
130
131     #ifdef ASYNC_CAPTURE
132     printf("Testing async camera capture\n");
133
134     #else
135     printf("Testing normal camera capture\n");
136     #endif
137
138     // Open the Himax camera
139     if (open_camera(&camera))
140     {
141         printf("Failed to open camera\n");
142         pmsis_exit(-1);
143     }

```

Open camera



```
16 #include "pmsis.h"
17 #include "bsp/bsp.h"
18 #include "bsp/bsp.h"
19 #include "bsp/bsp/himax.h"
20
21 Include drivers
22 Include image IO library
23 Include own demosaicking function
```

```

30 #endif
31 // Define acquisition size
32 #define ACQ_SIZE (1024*H*W)
33
34 PI_L2 unsigned char *buff;
35
36 PI_L2 unsigned char *buff_demosaick;
37
38 static struct pi_device camera;
39 static volatile int done;
40
41 // Define variables – place buffer in L2

```

Open and initialize camera

Open camera

```

1006     / Reserve buffer space
1007     buff = pmsis_l2_malloc(B
1008     if (buff == NULL){ return
1009
1010     #ifdef COLOR_IMAGE
1011     buff_demaosaic = pmsis_l
1012     #else
1013     buff_demaosaic = pmsis_l
1014     #endif
1015     if (buff_demaosaic == NU
1016     printf("Initialized buffers\n");

```

```

120 // Rotate camera orientation
121 uint8_t set_value=3;
122 uint8_t reg_value;
123
124 pi_camera_reg_set(&camera, IMG_ORIENTATION, &set_value);
125 pi_camera_reg_get(&camera, IMG_ORIENTATION, &reg_value);
126 printf("img orientation %d\n", reg_value);
127
128 #ifdef QVGA_MODE
129 set_value=1;
130 pi_camera_reg_set(&camera, QVGA_WIN_EN, &set_value);
131 pi_camera_reg_get(&camera, QVGA_WIN_EN, &reg_value);
132 printf("qvga window enabled %d\n", reg_value);
133 #endif
134
135 #ifndef ASYNC_CAPTURE
136 set_value=0;
137 pi_camera_reg_set(&camera, VSYNC_HSYNC_PIXEL_SHIFT_EN, &set_value);
138 pi_camera_reg_get(&camera, VSYNC_HSYNC_PIXEL_SHIFT_EN, &reg_value);
139 printf("vsync hsync pixel shift enabled %d\n", reg_value);
140 #endif
141
142 // Configure camera registers

```

Configure camera registers

Hands-on: Image acquisition and filtering

Include drivers
 Include image IO library
 Include own demosaicking function
 Define acquisition size
 Define variables – place buffer in L2
 Open and initialize camera

```

16 #include "pmsis.h"
17 #include "bsp/bsp.h"
18 #include "bsp/camera/himax.h"
19 #include "bsp/camera/qvga.h"
20 #include "bsp/camera/pi_camera.h"
21 #include "bsp/camera/pi_camera_conf.h"
22 #include "bsp/camera/pi_camera_reg.h"
23 #include "bsp/camera/pi_camera_reg_def.h"
24 #include "bsp/camera/pi_camera_reg_def.h"
25 #define WIDTH 324
26 #ifdef QVGA_MODE
27 #define HEIGHT 244
28 #else
29 #define HEIGHT 324
30 #endif
31 #define PI_CAMERA_CMD_AEG_INIT 0
32 #define PI_CAMERA_CMD_START 0
33 PI_L2 unsigned char *buff;
34 PI_L2 unsigned char *buff_demosaick;
35 static struct pi_device camera;
36 static volatile int done;
37 static void handle_transfer_end(void *arg)
38 {
39     done = 1;
40 }
41 static int open_camera(struct pi_device *device)
42 {
43     printf("Opening Himax camera\n");
44     struct pi_himax_conf cam_conf;
45     pi_himax_conf_init(&cam_conf);
46     #if defined(QVGA_MODE)
47     cam_conf.format = PI_CAMERA_QVGA;
48     #endif
49     pi_open_from_conf(device, &cam_conf);
50     if (pi_camera_open(device))
51         return -1;
52     pi_camera_control(device, PI_CAMERA_CMD_AEG_INIT, 0);
53     return 0;
54 }
  
```

Open camera

```

65 int test_camera()
66 {
67     printf("Entering main controller\n");
68     #ifdef ASYNC_CAPTURE
69     printf("Testing async camera capture\n");
70     #else
71     printf("Testing normal camera capture\n");
72     #endif
73     // Open the Himax camera
74     if (open_camera(&camera))
75     {
76         printf("Failed to open camera\n");
77         pmsis_exit(-1);
78     }
79 }
  
```

Configure camera register

```

83 // Rotate camera orientation
84 uint8_t set_value=3;
85 uint8_t reg_value;
86 pi_camera_reg_set(&camera, PI_CAMERA_REG_ORIENTATION, set_value);
87 pi_camera_reg_get(&camera, PI_CAMERA_REG_ORIENTATION, &reg_value);
88 printf("img orientation %d\n", reg_value);
89 #ifdef QVGA_MODE
90 set_value=1;
91 pi_camera_reg_set(&camera, PI_CAMERA_REG_QVGA_WINDOW_ENABLE, set_value);
92 pi_camera_reg_get(&camera, PI_CAMERA_REG_QVGA_WINDOW_ENABLE, &reg_value);
93 printf("qvga window enable %d\n", reg_value);
94 #endif
95 #ifndef ASYNC_CAPTURE
96 set_value=0;
97 pi_camera_reg_set(&camera, PI_CAMERA_REG_ASYNC_CAPTURE, set_value);
98 pi_camera_reg_get(&camera, PI_CAMERA_REG_ASYNC_CAPTURE, &reg_value);
99 printf("vsync hsync pixel clock\n");
100 }
  
```

Allocated buffers in L2

```

106 // Reserve buffer space for image
107 buff = pmsis_l2_malloc(BUFF_SIZE);
108 if (buff == NULL){ return -1;}
109 #ifdef COLOR_IMAGE
110 buff_demosaick = pmsis_l2_malloc(BUFF_SIZE*3);
111 #else
112 buff_demosaick = pmsis_l2_malloc(BUFF_SIZE);
113 #endif
114 if (buff_demosaick == NULL){ return -1;}
115 printf("Initialized buffers\n");
  
```

Set up OS, then jump to test_camera

```

120 #ifdef ASYNC_CAPTURE
121 // Start up async capture task
122 done = 0;
123 pi_task_t task;
124 pi_camera_capture_async(&camera, buff, BUFF_SIZE, pi_task_callback(&task, handle_transfer_end, NULL));
125 #endif
126 // Start the camera
127 pi_camera_control(&camera, PI_CAMERA_CMD_START, 0);
128 #ifdef ASYNC_CAPTURE
129 while(!done){pi_yield();}
130 #else
131 pi_camera_capture(&camera, buff, BUFF_SIZE);
132 #endif
133 }
  
```

Set up OS, then jump to test_camera

```

106 // Reserve buffer space for image
107 buff = pmsis_l2_malloc(BUFF_SIZE);
108 if (buff == NULL){ return -1;}
109 #ifdef COLOR_IMAGE
110 buff_demosaick = pmsis_l2_malloc(BUFF_SIZE*3);
111 #else
112 buff_demosaick = pmsis_l2_malloc(BUFF_SIZE);
113 #endif
114 if (buff_demosaick == NULL){ return -1;}
115 printf("Initialized buffers\n");
  
```

demosaick, RGB888_IO);
 demosaick, GRAY_SCALE_IO);
 GRAY_SCALE_IO);



```
#include "pmsis.h"
#include "bsp/bsp.h"
// #include "bsp/himmax.h"
// #include "bsp/himax.h"
```

Define variables – place buffers

Open and initialize camera

Start camera

Blocking capture

Stop and close camera

Apply a kernel

Write image over openOCD/JTAG to a file on the computer



```

6 #include "pmsis.h"
7 #include "bsp/bsp.h"
8 #include "bsp/bsp.himx.h"
9 #include "bsp/bsp/himax.h"

```

1 Include drivers

2 Include image IO library

3 Include own demosaicking function

But we do not only want to take one image,
we want to continuously take images in a loop!
For simplicity, we focus on synchronus capture

Define variables – place butter in LZ

Asynchronous capture callback

```
46 static int open_camera(struct pi_device *device)
47 {
48     printf("Opening Himax camera\n");
49     struct pi_himax_conf cam_conf;
50     pi_himax_conf_init(&cam_conf);
```

Open and initialize camera

```
65 int test_camera()
66 {
67     printf("Entering main controller\n");
68
69     #ifdef ASYNC_CAPTURE
70     printf("Testing async camera capture\n");
71
72     #else
73     printf("Testing normal camera capture\n");
74     #endif
75 }
```

```

91     printf("img orientation %d\n", reg_value);
92
93     #ifdef QVGA_MODE
94     set_value=1;
95     pi_camera_reg_set(&camera, QVGA_WIN_EN, &set_value);
96     pi_camera_reg_get(&camera, QVGA_WIN_EN, &reg_value);
97     printf("qvga window enabled %d\n", reg_value);
98     #endif
99
100    #ifndef ASYNC_CAPTURE
101    set_value=0;
102    pi_camera_reg_set(&camera, VSYNC_HSYNC_PIXEL_SHIFT_EN, &set_value);
103    pi_camera_reg_get(&camera, VSYNC_HSYNC_PIXEL_SHIFT_EN, &reg_value);
104    printf("vsync hsync pixel shift enabled %d\n", reg_value);

```

Configure camera registers

Allocated buffers in L2

```
120 #!def ASYNC_CAPTURE
121     Asynchronous capture – can queue buffer before starting
122     done = 0;
123     pi_task_t task;
124     pi_camera_capture_async(&camera, buff, BUFF_SIZE, pi_task_callback(&task, handle_transfer_end, camera,
```

```
126 // Start the camera
127
128 pi camera control(&camera, PI_CAMERA_CMD_START, 0);
```

```
129 #lorder ASYNC_CAPTURE
130 Wait for capture to end (pi_yield() blocks until an event happens)
131 #else
```

```
132     pi_camera_capture(&camera, buff, BUFF_SIZE);
133     #endif
134
```

Blocking capture

```
135 // Stop the camera and immediately close it
136 pi_camera_control(&camera, PI_CAMERA_CMD_STOP, 0);
137 pi_camera_close(&camera);
```

```
139     #ifdef COLOR_IMAGE
140     demosaicking(buff, buff_demosaick, WIDTH, HEIGHT, 0);
141     #else
142     demosaicking(buff, buff_demosaick, WIDTH, HEIGHT, 1);
143     #endif
```

```
151 #endif
152
153 Write image over openOCD/JTAG to a file on the computer
```

```
158 int main(void)
159 {
160     printf("\n")*** PMSYS Camera Example ***"\n\n");
161     // ... (rest of the code) ...
162 }
```

Hands-on: Image acquisition and filtering

Include drivers
Include image IO library
Include own demosaicking

But we do not
Define ac we want to c
For simplicity
Define variables – place bu

static int open_camera(struct pi_device
{
printf("Opening Himax camera\n");
struct pi_himax_conf cam_conf;
pi_himax_conf_init(&cam_conf);
#if defined(QVGA_MODE)
cam_conf.format = PI_CAMERA_QVGA;
#endif
pi_open_from_conf(device, &cam_conf;
if (pi_camera_open(device))
return -1;
pi_camera_control(device, PI_CAMERA
Open and initialize camera

```
126
127 // Start the camera
128 pi_camera_control(&camera, PI_CAMERA_CMD_START, 0);
```

Start camera

```
132 pi_camera_capture(&camera, buff, BUFF_SIZE);
133 #endif
```

Blocking capture

```
135 // Stop the camera and immediately close it
136 pi_camera_control(&camera, PI_CAMERA_CMD_STOP, 0);
137 pi_camera_close(&camera);
```

Stop and close camera

```
140 #ifdef COLOR_IMAGE
141 demosaicking(buff, buff_demosaick, WIDTH, HEIGHT, 0);
142 #else
143 demosaicking(buff, buff_demosaick, WIDTH, HEIGHT, 1);
144 #endif
```

Apply a kernel

```
146 // Write to file
147 #ifdef COLOR_IMAGE
148 WriteImageToFile("../img_color.ppm", WIDTH, HEIGHT, sizeof(uint32_t), buff_demosaick, RGB888_IO);
149 #else
150 WriteImageToFile("../img_gray.ppm", WIDTH, HEIGHT, sizeof(uint8_t), buff_demosaick, GRAY_SCALE_IO);
151 #endif
152
153 WriteImageToFile("../img_raw.ppm", WIDTH, HEIGHT, sizeof(uint8_t), buff, GRAY_SCALE_IO);
154
155 pmsis exit(0);
```

Write image over openOCD/JTAG to a file on the computer



Hands-on: Image acquisition and filtering

Include drivers

Include image IO library

Include own demosaicking

But we do not
Define ac we want to c

For simplicity

Define variables – place bu

```
46 static int open_camera(struct pi_device
47 {
48     printf("Opening Himax camera\n");
49     struct pi_himax_conf cam_conf;
50     pi_himax_conf_init(&cam_conf);
51
52     #if defined(QVGA_MODE)
53     cam_conf.format = PI_CAMERA_QVGA;
54     #endif
55
56     pi_open_from_conf(device, &cam_conf);
57     if (pi_camera_open(device))
58         return -1;
59     pi_camera_control(device, PI_CAMERA
60
```

Open and initialize camera

```
126
127 // Start the camera
128 pi_camera_control(&camera, PI_CAMERA_CMD_START, 0);
129 while(1){
130
131
132     pi_camera_capture(&camera, buff, BUFF_SIZE);
133     #endif
134
```

```
138
139
140 #ifdef COLOR_IMAGE
141     demosaicking(buff, buff_demosaick, WIDTH, HEIGHT, 0);
142 #else
143     demosaicking(buff, buff_demosaick, WIDTH, HEIGHT, 1);
144 #endif
145
146 // Write to file
147 #ifdef COLOR_IMAGE
148     WriteImageToFile("../img_color.ppm", WIDTH, HEIGHT, sizeof(uint32_t), buff_demosaick, RGB888_IO);
149 #else
150     WriteImageToFile("../img_gray.ppm", WIDTH, HEIGHT, sizeof(uint8_t), buff_demosaick, GRAY_SCALE_IO);
151 #endif
152
153     WriteImageToFile("../img_raw.ppm", WIDTH, HEIGHT, sizeof(uint8_t), buff, GRAY_SCALE_IO);
154
```

Write image over openOCD/JTAG to a file on the computer

```
135 // Stop the camera and immediately close it
136 pi_camera_control(&camera, PI_CAMERA_CMD_STOP, 0);
137 pi_camera_close(&camera);
```

Stop and close camera

Great! You now have basically an universal pipeline for any kernel you want to run.

Apply a kernel

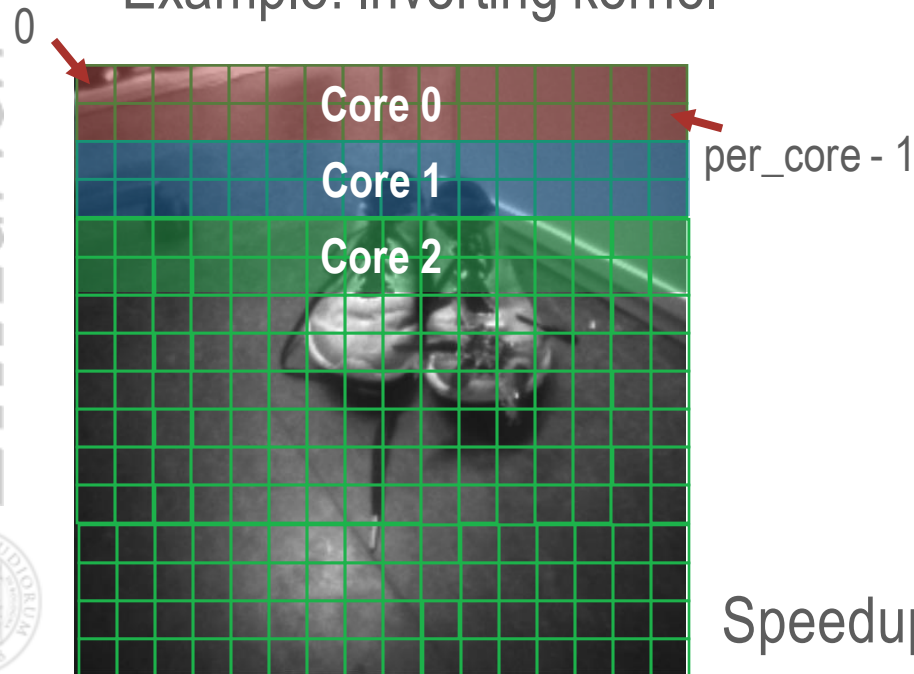
computer



Hands-on: Image acquisition and filtering

How do we improve performance?

- Avoid float operations
- Parallelize code
 - All cores should execute similar code on different data
- Example: Inverting kernel



```

6  typedef struct {
7      char *srcBuffer;    // pointer to the input vector
8      char *resBuffer;    // pointer to the output vector
9      uint32_t width;     // image width
10     uint32_t height;    // image height
11     uint32_t nPE;       // number of cores
12     uint32_t grayscale; // grayscale if one
13 } plp_example_kernel_instance_i32;

```

```

216 void cluster_inverting(void* args)
217 {
218     uint32_t idx = 0;
219     uint32_t core_id = pi_core_id(), cluster_id = pi_cluster_id();
220     plp_example_kernel_instance_i32 *a = (plp_example_kernel_instance_i32*)args;
221     char *srcBuffer = a->srcBuffer;
222     char *resBuffer = a->resBuffer;
223     uint32_t width = a->width;
224     uint32_t height = a->height;
225     uint32_t nPE = a->nPE;
226
227     uint32_t total = width*height;
228
229     // amount of elements per core, rounded up
230     uint32_t per_core = (total+nPE-1)/nPE;
231     // compute the last element of the area each core has to process
232     uint32_t upper_bound = (core_id+1)*per_core;
233     // as we always rounded up before (to distribute the load as equal as possible)
234     // we need to check if the upper bound is still in our matrix
235     if(upper_bound > total) upper_bound = total;
236     // loop over the area assigned to the core
237     for (idx = core_id*per_core; idx < upper_bound; idx++) {
238
239         resBuffer[idx] = 255 - srcBuffer[idx];
240
241     }
242 }

```

Speedup: @50MHz FC and Cluster from 8ms ->1.5ms

