GRAPHICS & MEDIA LAB
VIDEO GROUP

Использование графического оборудования для обработки видеоданных

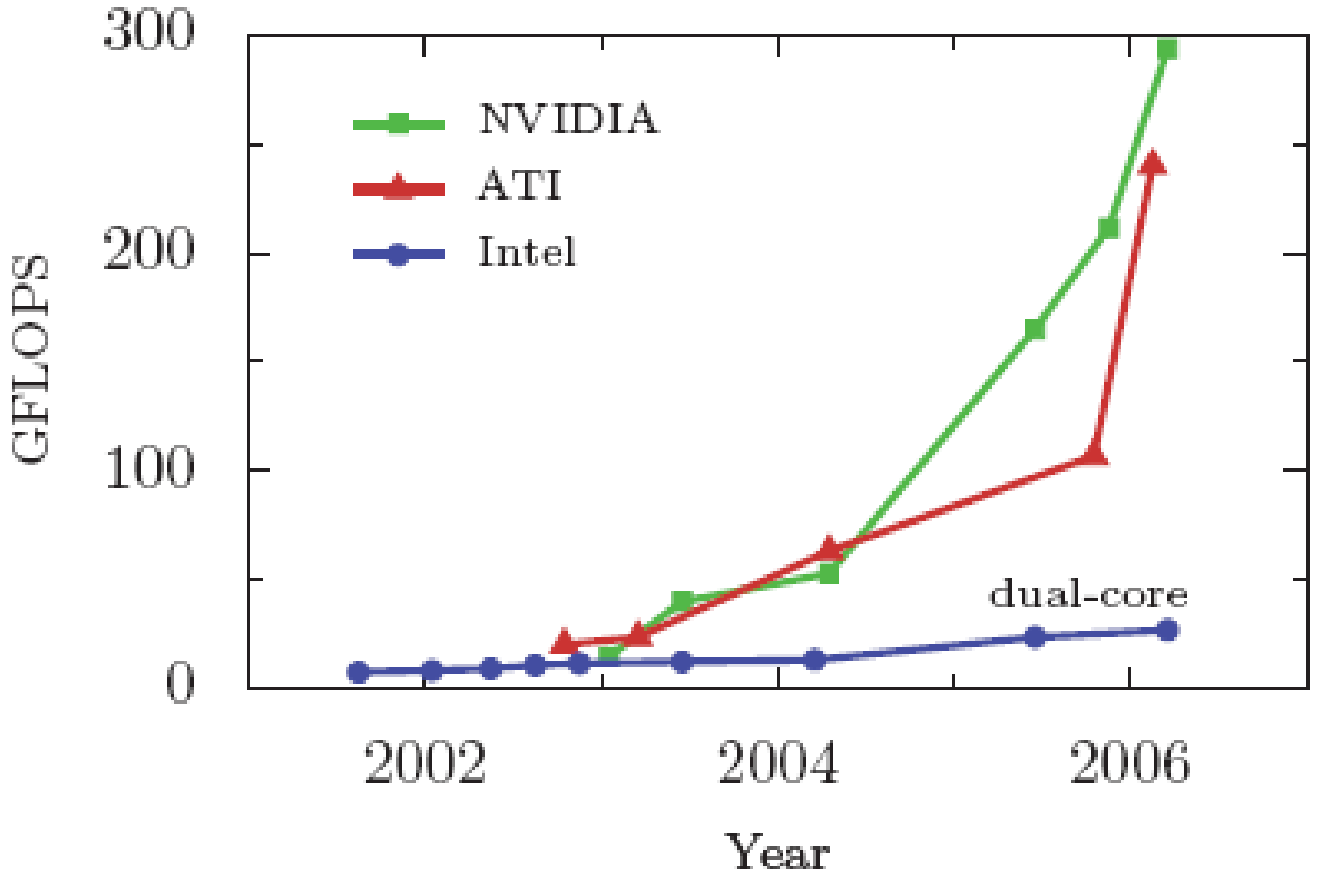
Марат Арсаев

Video Group
CS MSU Graphics & Media Lab

Содержание доклада

- **Введение**
- Обзор архитектуры графического оборудования
- Использование на практике
- Мои цели и задачи

Введение



“A Survey of General-Purpose Computation on Graphics Hardware”, John D. Owens, David Luebke, Naga Govindaraju, Computer Graphics Forum, 2007

Введение

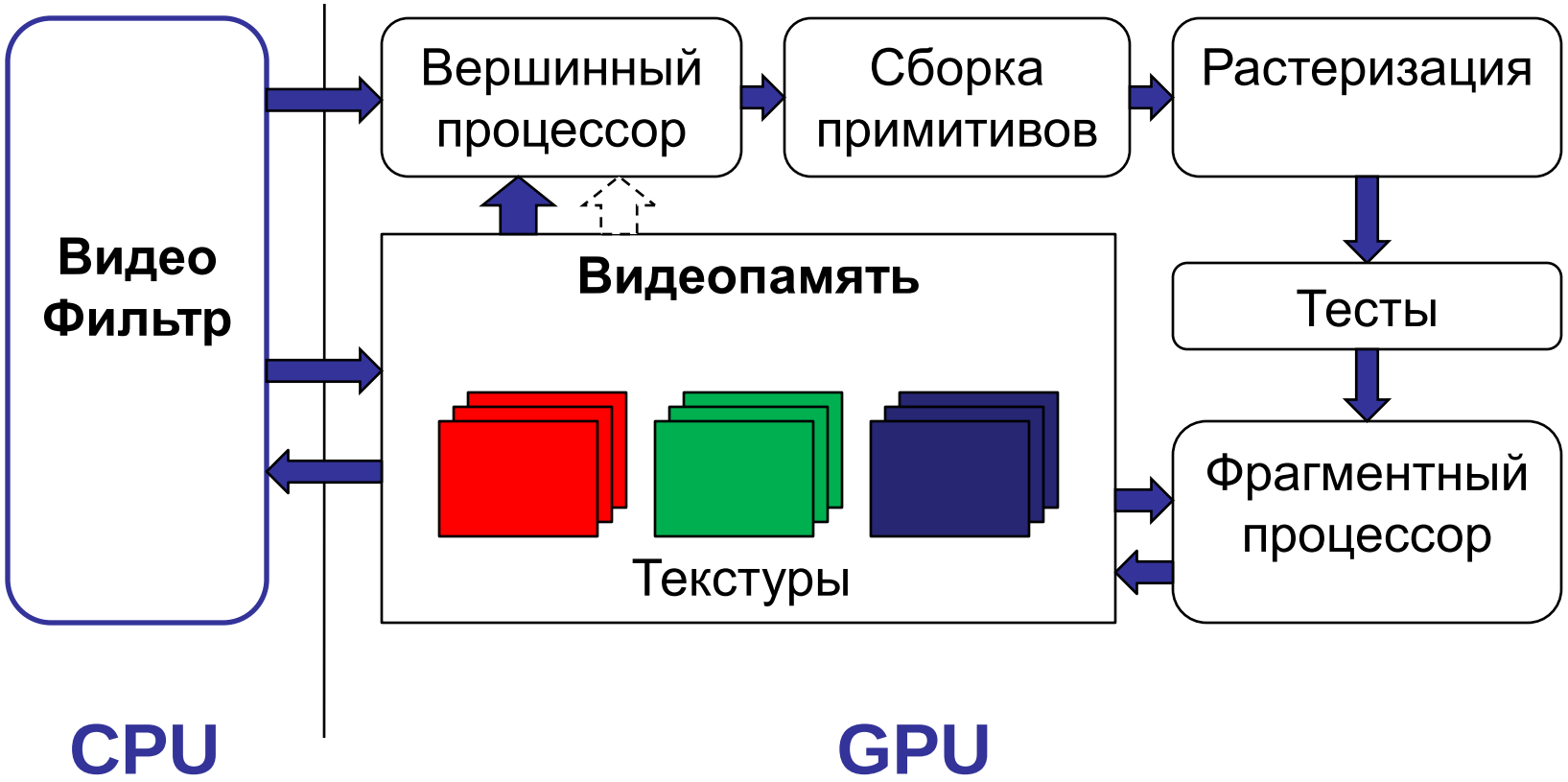
Чем обусловлено преимущество?

- до 160(ATI)/240(NVIDIA) программируемых процессоров
- высокая степень распараллеливания
- наличие специфических операций

Содержание доклада

- Введение
- **Обзор средств программирования и архитектуры GPU**
- Использование на практике
- Мои цели и задачи

Архитектура GPU



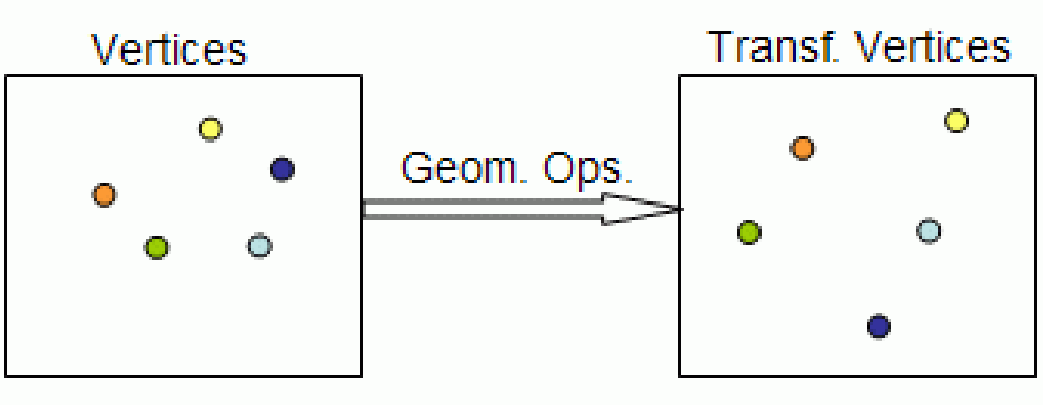
Перенос задачи на GPU

Основные концепции:

- Массив данных – текстура
- Вычисления – шейдерная программа
- Запуск вычислений – рисование в буфер кадров

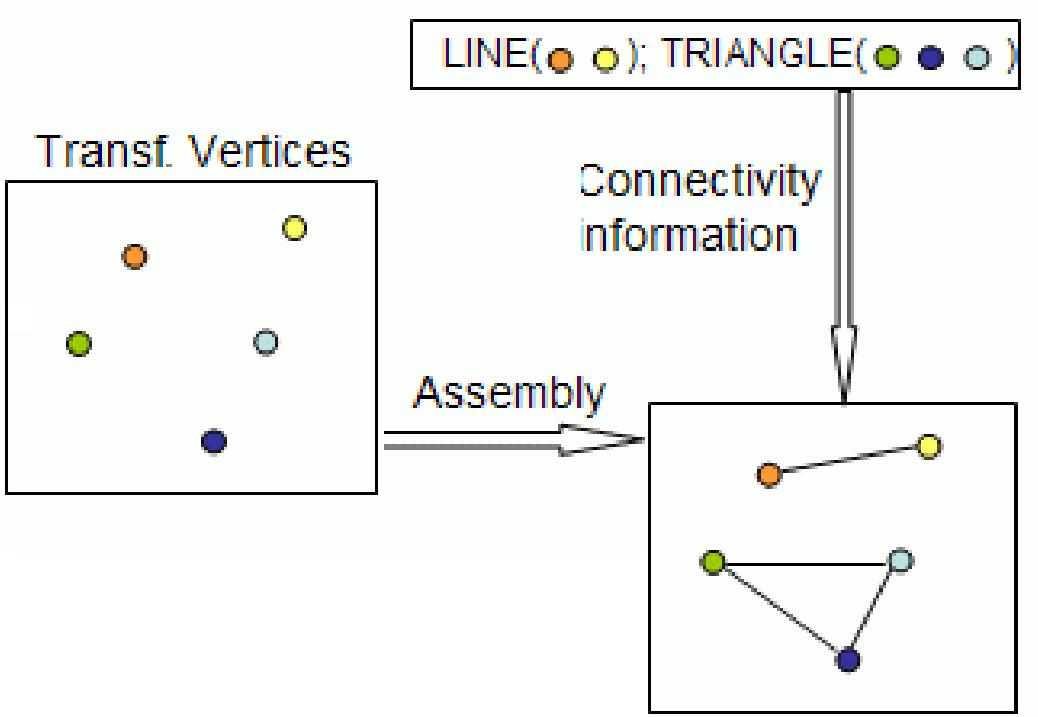
Вершинный процессор

- На вход – вершины с атрибуты
- Добавление, изменение аттрибутов
- На последнем оборудовании - медленный доступ к текстурам



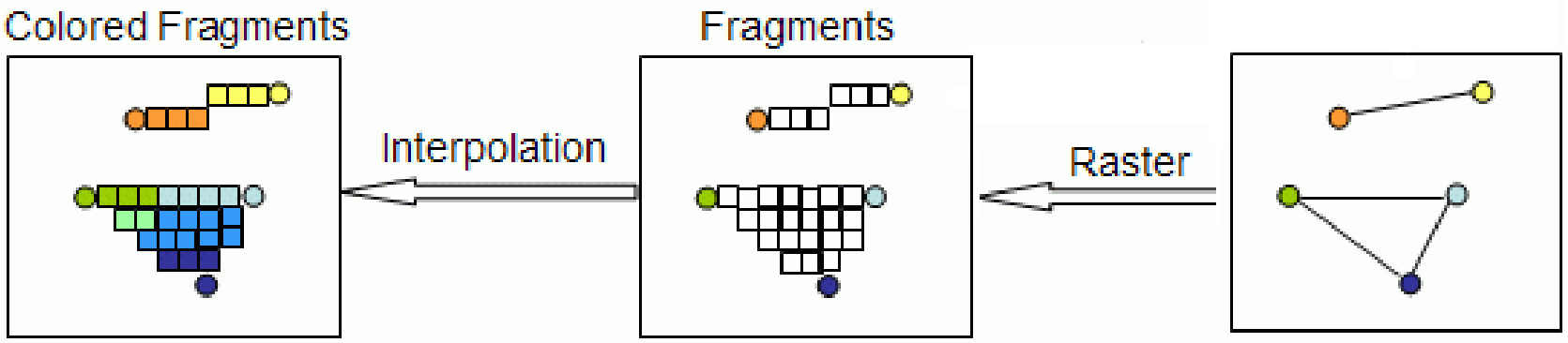
Сборка

- На вход - преобразованные вершины и информацию о их связях
- Формирование фигур



Растеризация

- Формирует фрагменты и координаты конечных пикселей
- Интерполирует атрибуты вершин



Тесты

- Тест глубины
- Тест по шаблону
- Позволяют разбить или избежать ненужных вычислений

Фрагментный процессор

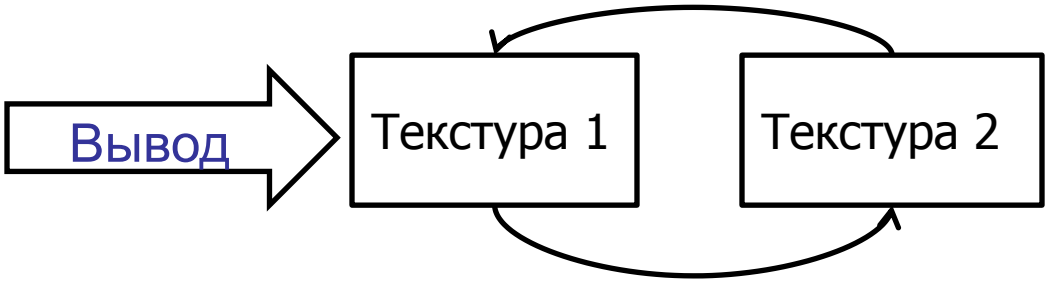
- На вход - пиксель с атрибутами
- Формирование цвета
- Одновременное вычисление нескольких наборов данных

Видеопамять

- Текстура – место хранения промежуточных и конечных результатов вычислений
- Параметры текстуры:
  - Texture target
  - Texture format
  - Render target

Вычисления в несколько проходов

- Используется метод пинг-понга



- Применение:
  - редукция
  - вычисление сложных алгоритмов

Редукция

- Производится для вычислений по некоторой области
- Рисование производится в 4 раза меньшую область

47	2	3	57	5	12	7	8
10	20	6	13	14	15	16	17
19	11	21	22	23	68	25	26
38	29	64	31	32	33	35	34
37	28	39	49	53	42	41	52
46	1	48	40	61	51	44	43
55	71	4	58	69	62	50	60
30	65	66	67	24	59	70	56

47	57	15	17
38	64	68	35
46	49	61	52
71	67	69	70

64	68
71	70

71

Обзор средств программирования

- ARB
- High level languages
  - HLSL\Cg
  - GLSL
  - other – Brook+, Lib Sh…
- CUDA, AMD Stream

ARB

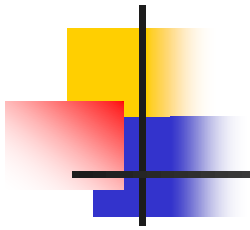
- Ассемблерный язык
- Не поддерживает контроль программы
- Используется для совместимости на низком уровне

```
!!ARBfp1.0
TEMP color;
MUL color, fragment.texcoord[0].y
, 2.0;
ADD color, 1.0, -color;
ABS color, color;
ADD result.color, 1.0, -color;
MOV result.color.a, 1.0;
END
```

HLSL

- C-подобный язык программирования
- Пользовательские функции
- Типы данных – **bool**, **fixed**, **half**
- Работает под DirectX начиная с версии 9.0

```
void main( in v2p IN, out p2f OUT )
 {
        float4 color = tex2D(tex0, IN.Texcoord0);
        OUT.Color = brightness * IN.Color * color;  }
```



Cg

- Практически идентичен HLSL
- Работает через OpenGL или DirectX
- Расширение структур
- Использование профайлов

GLSL

- C-подобный язык программирования
- Часть спецификации OpenGL 2.0
- Некоторые отличия от HLSL в производительности

```
varying vec2 texture_coordinate;
uniform sampler2D  my_color_texture;
void main()
{
        gl_FragColor = texture2D(my_color_texture,
texture_coordinate); }
```

CUDA

- Расширение языка C
- Не требует знания архитектуры видеокарты
- Доступен только на последнем оборудовании NVIDIA

AMD Stream

- Расширение языка Brook
- SDK содержит оптимизированные библиотеки основных функций для ATI
- Поддерживает OpenCL
- Совместим только с потоковыми GPU ATI

Обзор средств программирования

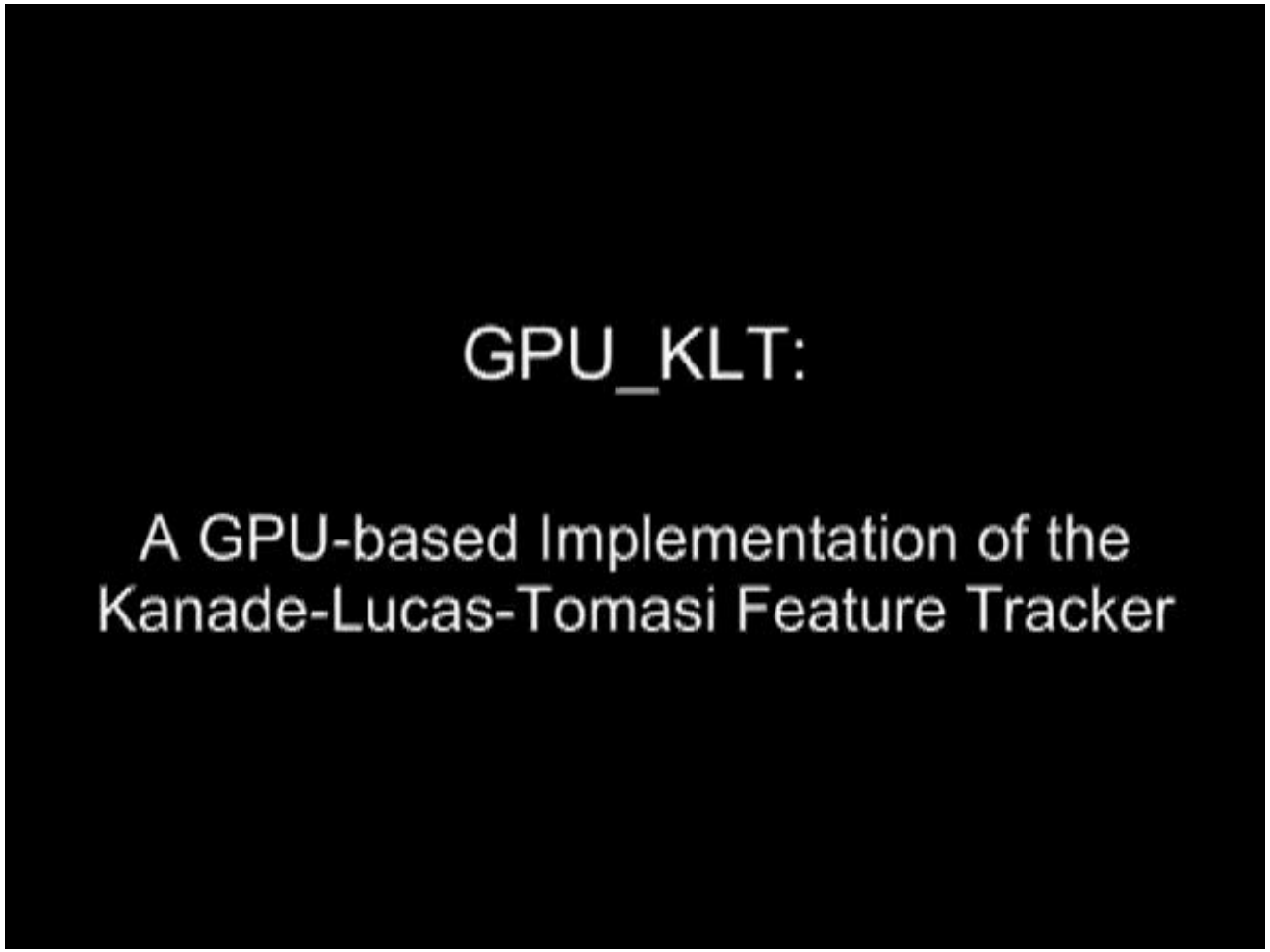
GRAPHICS & MEDIA LAB
VIDEO GROUP

	Доступ к данным	Набор инструкций	Поддержка GPU	Абстракция
ARB	текстура	суженный	практически все	на уровне языка
GLSL	текстура	полный	зависит от версии шейдеров	на уровне шейдера
Cg (HLSL)	текстура	полный	зависит от версии шейдеров	на уровне шейдера
CUDA (AMD Stream)	прямой доступ	C-подобный (Brook+)	NVidia (ATI) последних поколений	на уровне основной программы

Содержание доклада

- Введение
- Обзор архитектуры графического оборудования
- **Использование на практике**
- Мои цели и задачи

GPU-KLT feature tracking



Необходимые вычисления

- цель – нахождение смещения **d** с целью минимизации ошибки **r**

$$r = \sum_W (I(\mathbf{x}+\mathbf{d}, t) - I(\mathbf{x}, t+\Delta t))^2$$

x – текущая точка
d - смещение
I - изображение
$[\frac{\partial I}{\partial x} \quad \frac{\partial I}{\partial y}]$ - вектор градиента изображения в т. **x**

- нахождение точек интереса

$$\mathbf{c} = min\,(eig\,(\sum_W [\frac{\partial I}{\partial x} \quad \frac{\partial I}{\partial y}]^T [\frac{\partial I}{\partial x} \quad \frac{\partial I}{\partial y}]))$$

"Feature Tracking and Matching in Video Using Programmable Graphics Hardware" Sudipta N. Sinha, Jan-Michael Frahm, Marc Pollefeys, Yakup Genc
Submitted to Machine Vision and Applications, July 2006

Необходимые вычисления

- трекинг реализуется решением уравнения

$$\underbrace{(\sum_{W} \mathbf{G}^T\mathbf{G})(\mathbf{d})}_{\mathbf{A}} = \underbrace{\sum_{W} \mathbf{G}^T \, \Delta \, \mathbf{I}(\mathbf{x}, \Delta t)}_{\mathbf{b}}$$

- G – градиент в точке x, представляется в симметрическом виде:

$$\mathbf{G} = [\frac{\partial(I(*,t) + I(*,t+\Delta t))}{\partial x} \quad \frac{\partial(I(*,t) + I(*,t+\Delta t))}{\partial y}]$$

GPU-KLT feature tracking

KLT Tracking (*ft_list*, F_0, F_1) {

(1) Build-Pyramid: builds multi-resolution intensity and gradient pyramid from images F_0, F_1

(2) Track:

For all pyramid levels from coarse to fine

For multiple iterations

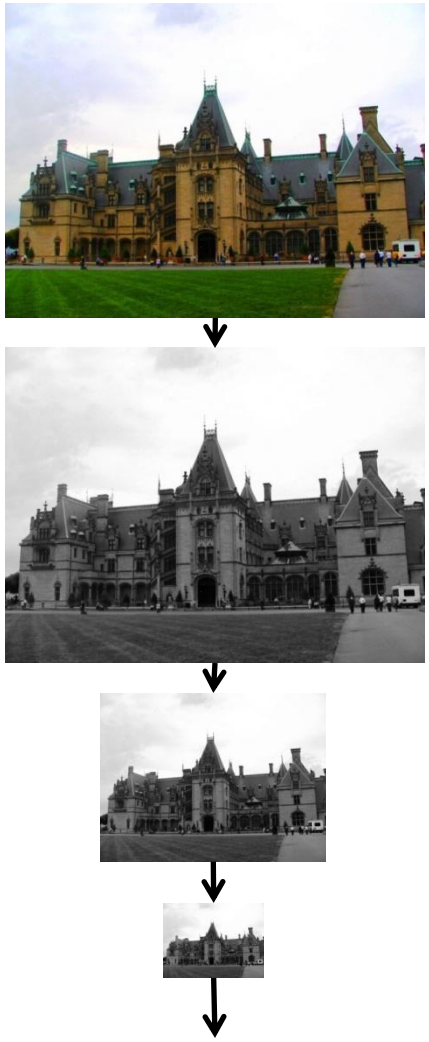
For each feature *f* in *ft_list*

compute coefficients of **A** and **b** as shown in Equation 1.

solve **A d = b**

evaluate **d** and update track of feature

}

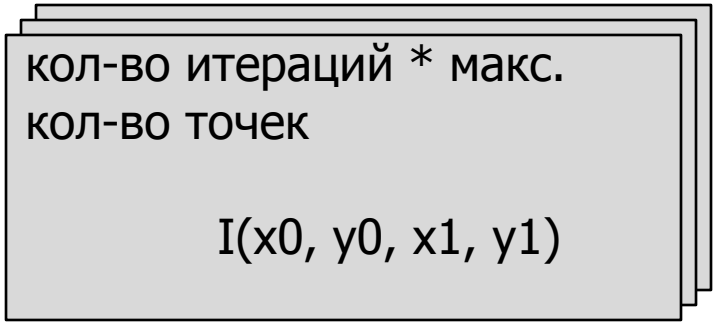


Build-Pyramid

- Используемые данные :
  - 2 входных изображения
  - набор текстур для пирамиды формата RGBA
- Размытие для градиента

Track

- Используемые данные:
  - набор текстур для представления точек



* кол-во уровней

  - временные текстуры для решения уравнения

Track

- 4 прохода :
  - интерполяция яркости и градиента вокруг каждой следующей точки (7x7)
  - вычисление матрицы **A** и вектора **b**
  - решение уравнения
- после каждой итерации проверка и выгрузка результатов в следующую строку текстуры
- выгрузка на CPU

GPU-KLT feature tracking

```
Re-select-Features (ft_list) {
    mask   = mask_out_region (ft_list)
    c_map = evaluate corner-ness
            measure c over whole image
    // Perform non-maximal suppression
    pts    = find_features (#max_feats,
             mask, sort (c_map) )
    add_new_features (ft_list, pts)
}
```

Feature re-select

- использование ранней фильтрации для определения шаблона поиска
- 2 прохода
  - вычисление элементов матрицы окна 7x7
  - подсчет минимального хар. значения
- результат – 8битовая текстура

Feature re-select

- фильтрация алгоритмом подавления в точках отсутствия максимума

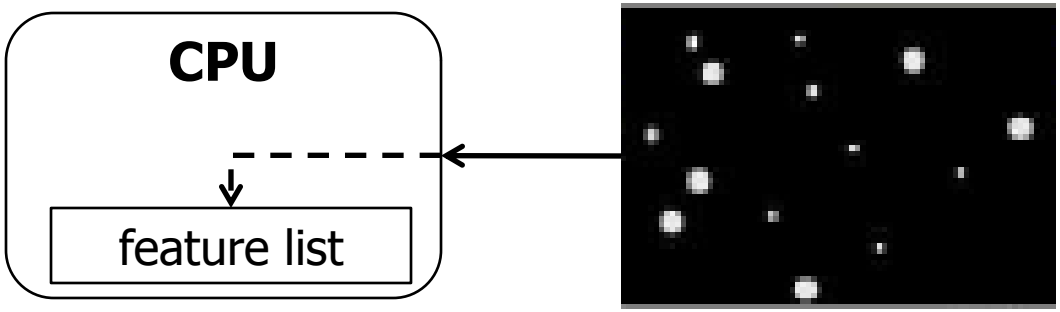
```
void main(uniform sampler2D src_tex : TEXUNIT0,
          float2 st0 : TEXCOORD0,
          uniform float2 ds,
          out float4 color : COLOR)
{
     float maxCornerness = pack_4ubyte(tex2D(src_tex, st0));
     …
```

Feature re-select

```
for (int i = -MIN_DIST; i < 0; ++i)
  {
    float cornerness = abs(pack_4ubyte(tex2D(src_tex, st0 + i*ds)));
    maxCornerness = (cornerness >= abs(maxCornerness)) ?
                                        (-cornerness) : maxCornerness;
  }
    …
color = unpack_4ubyte(maxCornerness);
}
```

Feature re-select

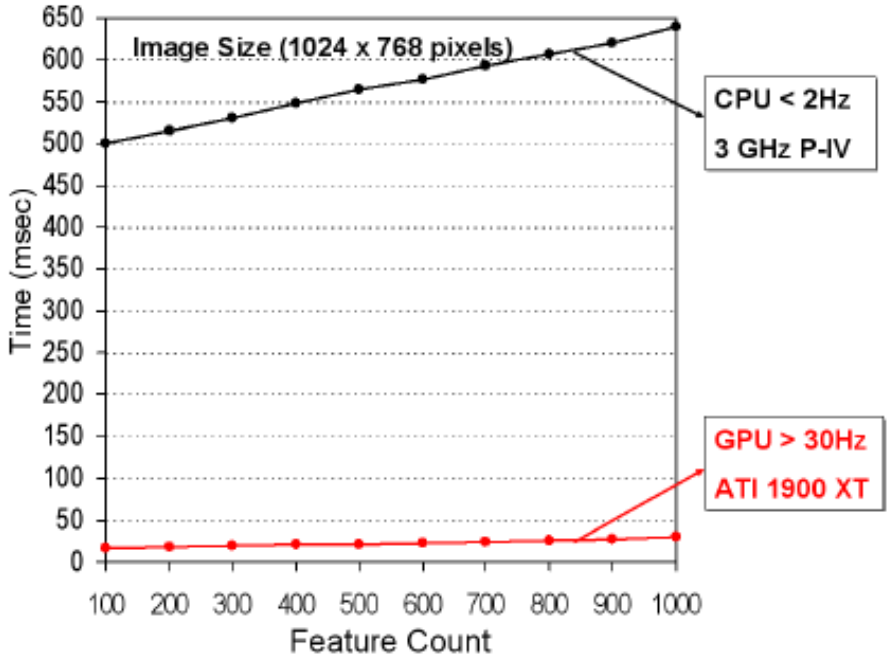
- результат выгружается на CPU

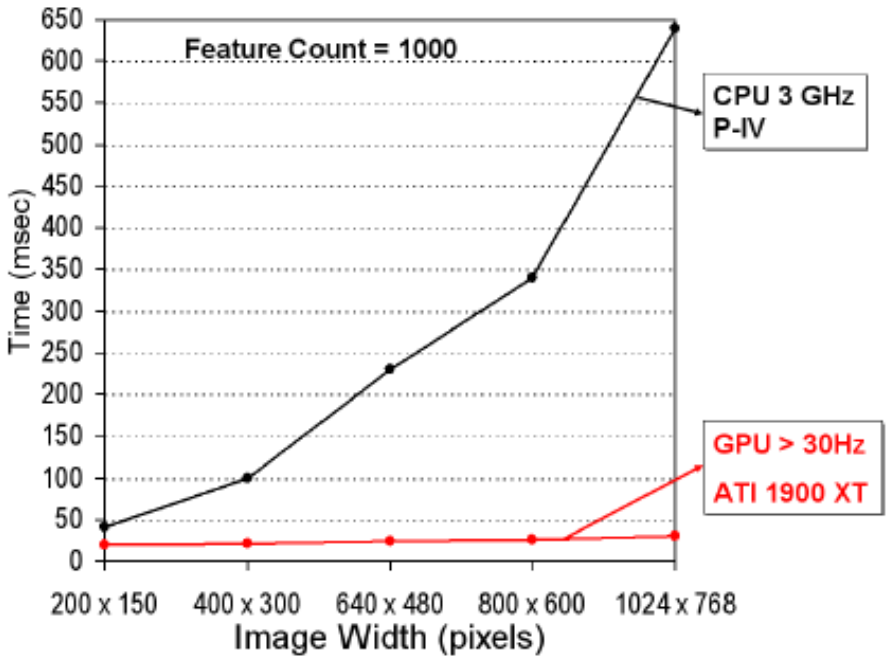


- сортировка и добавление лучших точек

GPU-KLT feature tracking: производительность

Real-time обработка видео 1024x768 – 30fps





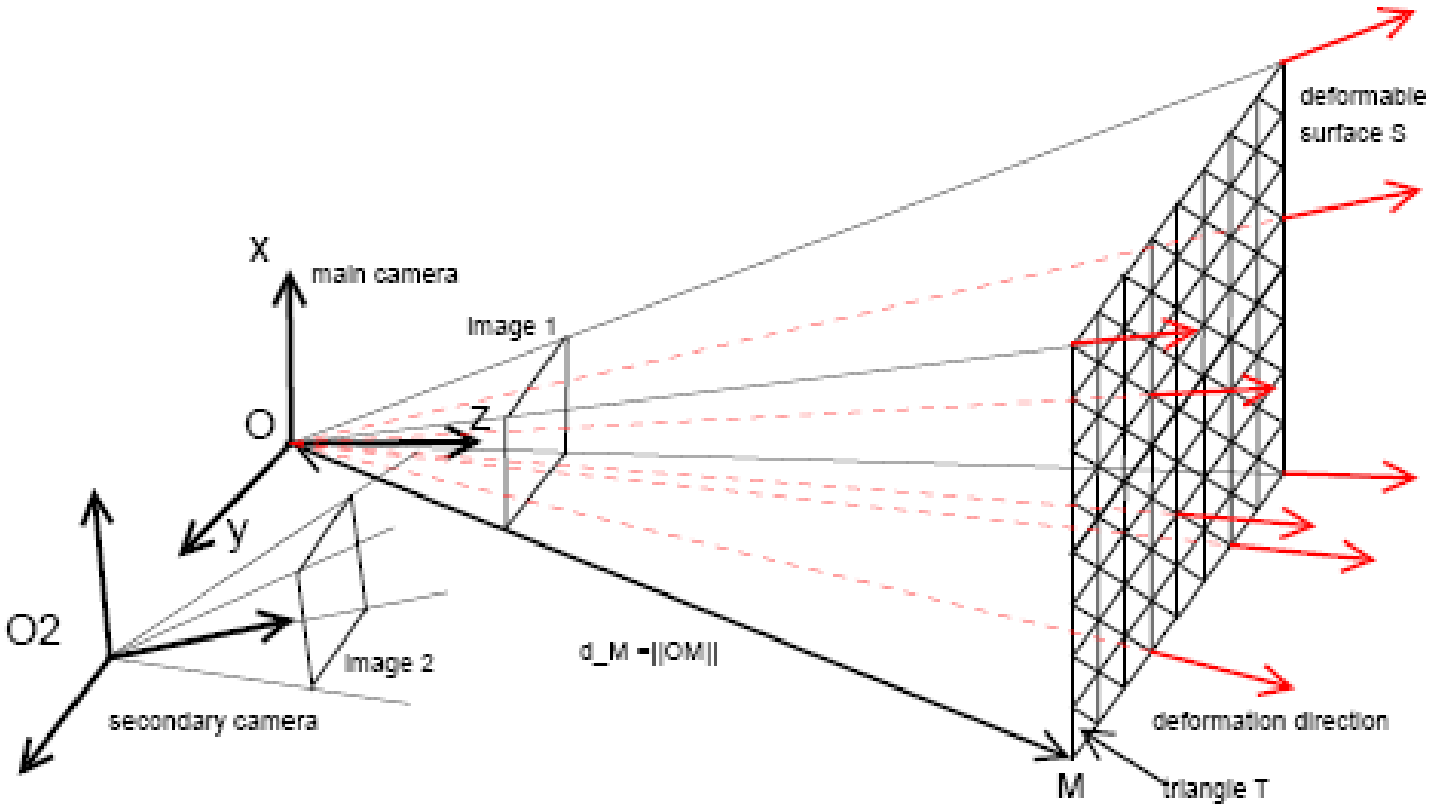
Fast and efficient dense variational stereo on GPU

GRAPHICS & MEDIA LAB
VIDEO GROUP

- Цель – построение 3D модели по 2-м изображениям
- Увеличение точности за счет 3-ей камеры
- Построение видео 3D модели

Fast and efficient dense variational stereo on GPU

- Цель – построение 3D модели по стерео-изображению



Необходимые вычисления

1) Подсчет энергии:

$$E(S) = \int_S (1 - \rho(I_1 \circ \Pi_1, I_2 \circ \Pi_2, m))dS(m)$$

$$E(S) = \sum_T E_T = \sum_T (1 - \rho_T(I_1 \circ \Pi_1, I_2 \circ \Pi_2))$$

$$\rho_T = \frac{< I_1, I_2 >}{|I_1|.|I_2|}$$

$$< I_1, I_2 > = \frac{1}{A_T} \int_T (I_1 \circ \Pi_1(m) - \overline{I_1 \circ \Pi_1})$$
$$(I_2 \circ \Pi_2(m) - \overline{I_2 \circ \Pi_2})dm$$

$$\overline{I_i \circ \Pi_i} = \frac{1}{A_T} \int_T I_i \circ \Pi_i(m)dm$$

$$|I_i| = \sqrt{< I_i, I_i >}$$

S – 3D модель

T – отдельный треугольник

I – изображение

П(m) – проекция точки **m** на фигуру

A – площадь тр-ка

Необходимые вычисления

2) Подсчет градиента:

M – точка

V(M) – ее соседи

d – расстояние до камеры

$$\frac{\partial E(S)}{\partial d_M} = \sum_{T \in V(M)} \frac{\partial E_T}{\partial d_M}$$

сводится к вычислению величины $\frac{\partial I_i \circ \Pi_i(m)}{\partial d_M}$

“Fast and efficient dense variational stereo on GPU”
Julien Mairal, Renaud Keriven and Alexandre Chariot
Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)

Алгоритм на GPU

Вычисление градиента:

$$D_{i,k}(\alpha_1, \alpha_2, \alpha_3) = \frac{\partial I_i \circ \Pi_i(p)}{\partial d_{M_k}}$$

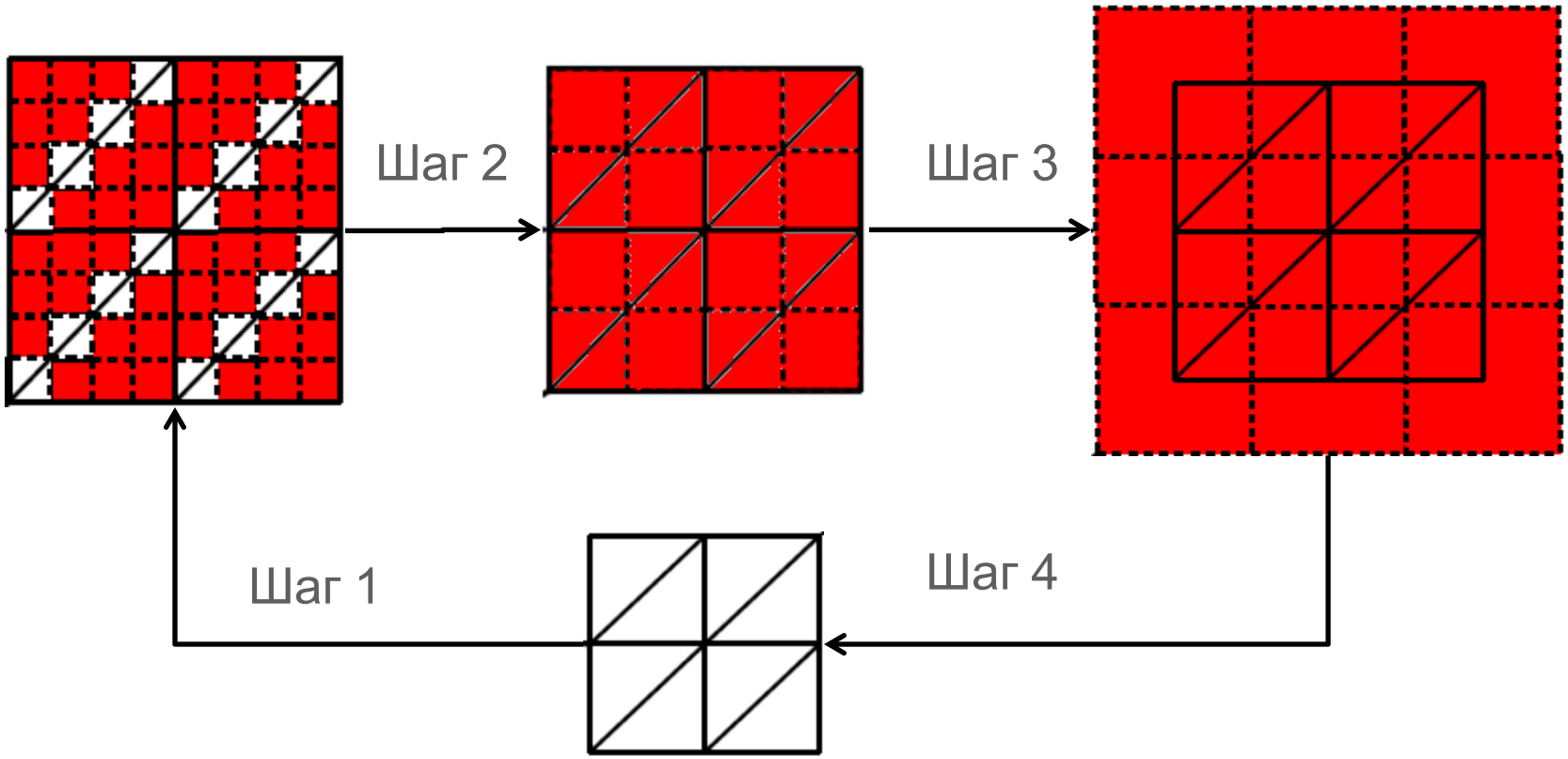
После прямого вычисления :

$$D_{i,k}(\alpha_1, \alpha_2, \alpha_3) = g_i(\alpha_k f_i(M_k), \Pi_i(p), (\nabla I_i) \circ \Pi_i(p))$$

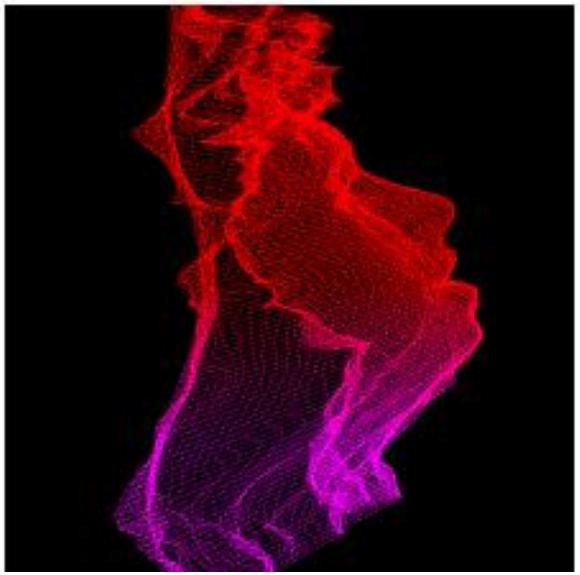
$$\Pi_i(p) = \alpha_1 \Pi_i(M_1) + \alpha_2 \Pi_i(M_2) + \alpha_3 \Pi_i(M_3)$$

i = 1, 2; k = 1,2,3; $p \in T$

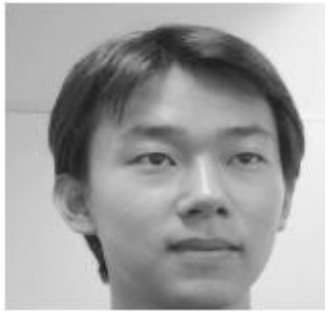
Алгоритм на GPU

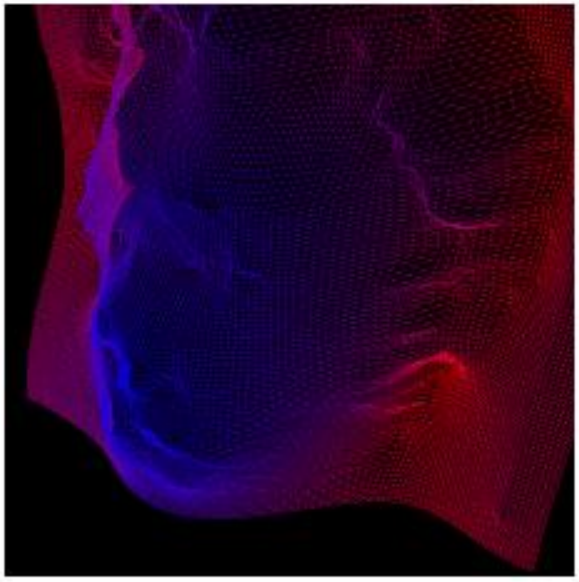


Fast and efficient dense variational stereo on GPU



Две камеры - неточности

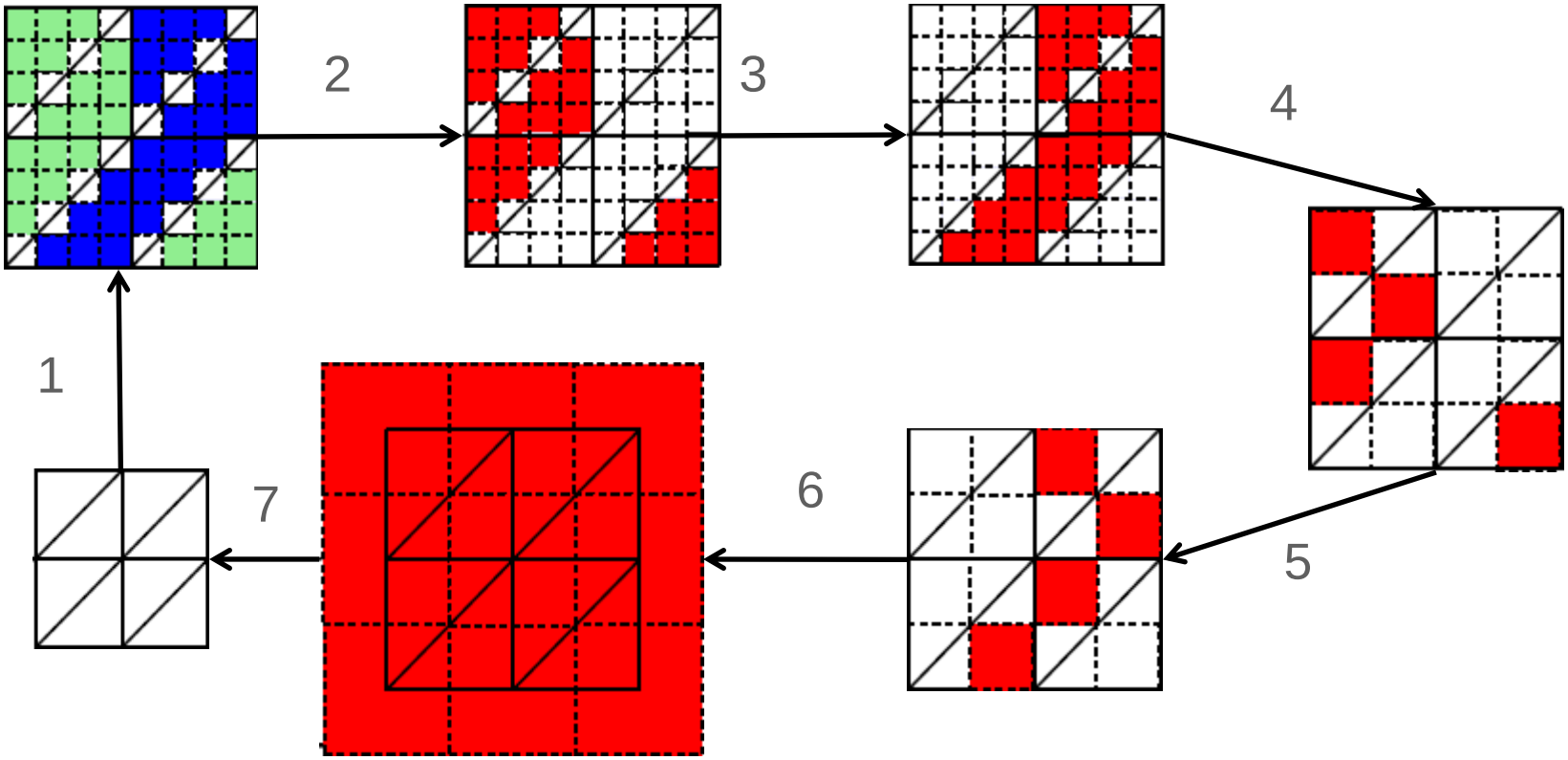




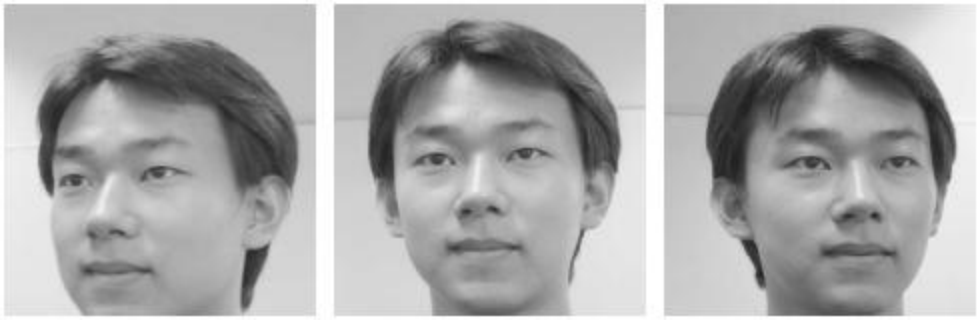
Три камеры

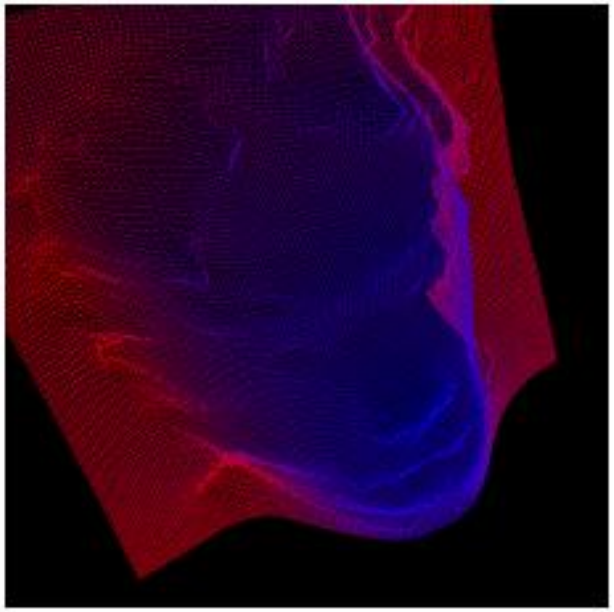
- Классификация треугольников
  - Отбрасывание невидимых частей
  - Принцип ближнего
- Последовательный просчет модели

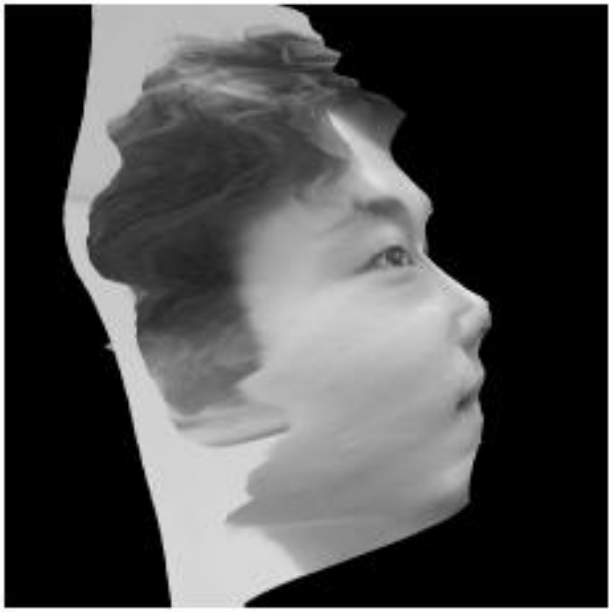
Три камеры



Результаты







Содержание доклада

- Введение
- Обзор архитектуры графического оборудования
- Использование на практике
- **Мои цели и задачи**

Список литературы

- **“A Survey of General-Purpose Computation on Graphics Hardware”,** John D. Owens, David Luebke, Naga Govindaraju, Computer Graphics Forum, 2007
- “**Feature Tracking and Matching in Video Using Programmable Graphics Hardware**” Sudipta N. Sinha, Jan-Michael Frahm, Marc Pollefeys, Yakup Genc, submitted to Machine Vision and Applications, July 2006
- “**Fast and efficient dense variational stereo on GPU**”, Julien Mairal, Renaud Keriven and Alexandre Chariot, Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)
- “**High-Level GPU Programming, Domain-specific optimization and inference**” Calle Lejdfors, Doctoral dissertation, Department of Computer Science, Lund University, 2008
- “**The GPU and Graphic Algorithms**” , Ivo Hanak, State of the Art and Concept of PhD Thesis, 2005

Лаборатория компьютерной графики и мультимедиа

Видеогруппа это:

- Выпускники в аспирантурах Англии, Франции, Швейцарии (в России в МГУ и ИПМ им. Келдыша)
- Выпускниками защищено 5 диссертаций
- Наиболее популярные в мире сравнения видеокодеков
- Более 3 миллионов скачанных фильтров обработки видео