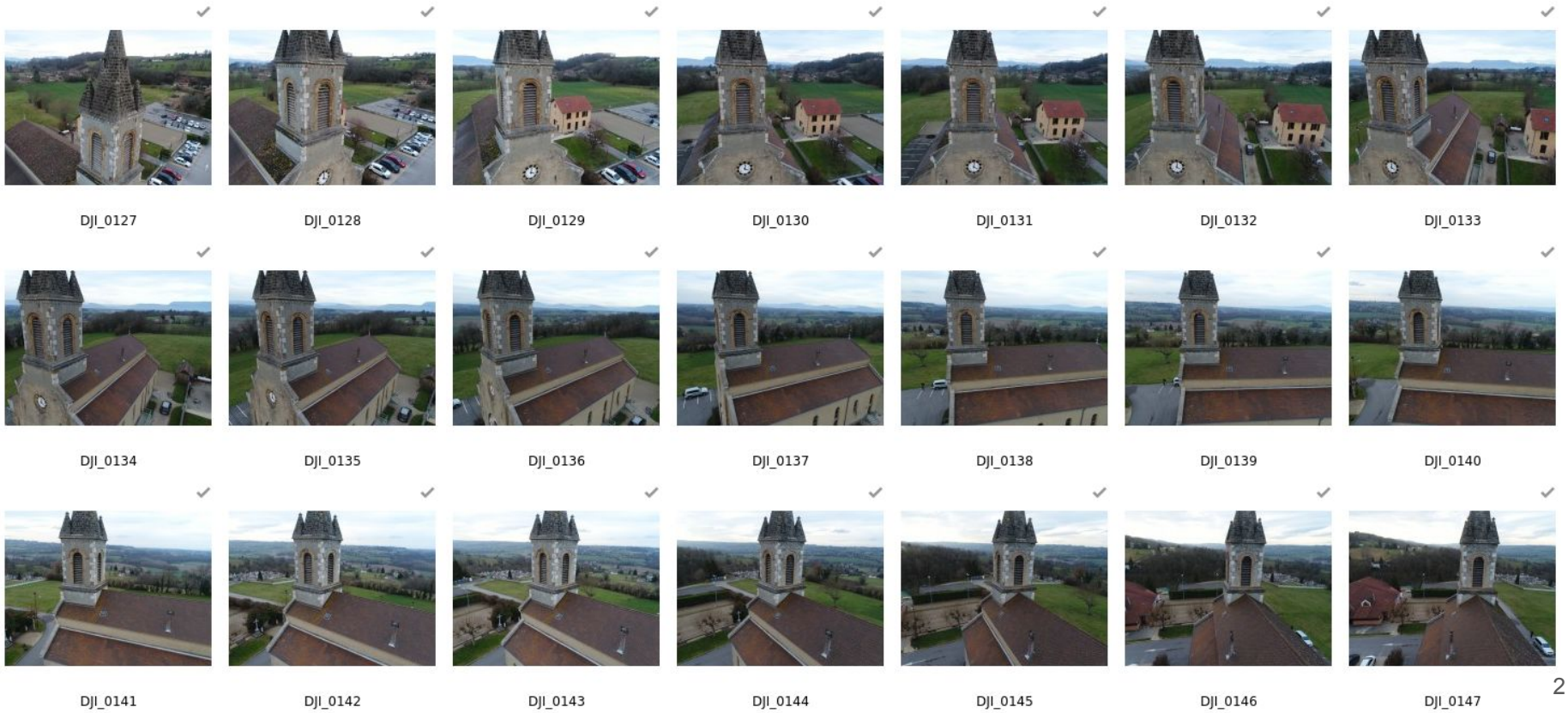
Agisoft Metashape

Полярный Николай
polarnick@agisoft.com

Metashape

Основная задача:

По множеству фотографий восстановить трехмерную модель.



Metashape

Основная задача:
По множеству фотографий восстановить трехмерную модель.

Данные предоставил Stéphane Prodent

Предложенные задачи

1. Генерация синтетического датасета (GTA V)
2. 3D обнаружение проводов ЛЭП
3. Построение карты высот по картам глубины численными методами

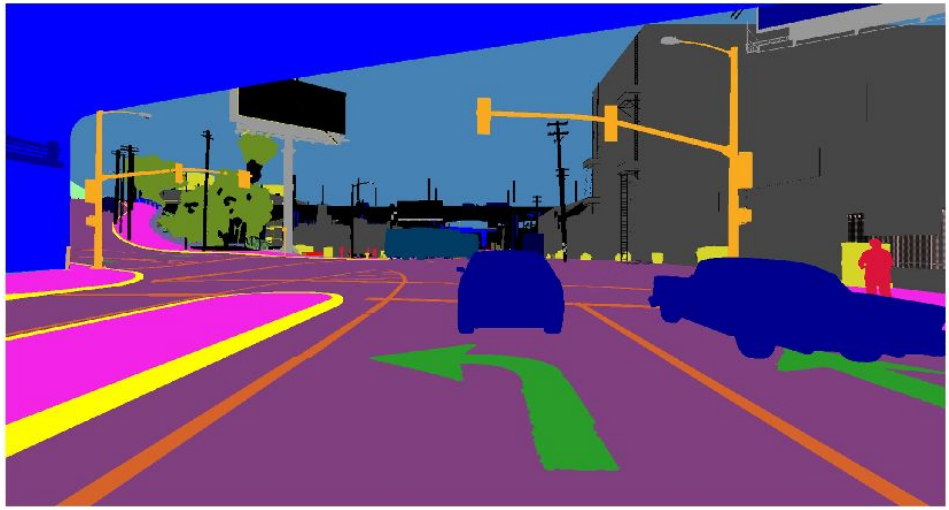
1. Генерация синтетического датасета (GTA V)

В Metashape есть классификация точек на:
землю, дороги, здания, растительность и машины.

1. Генерация синтетического датасета (GTA V)

Проблема:
Мало размеченных данных - хочется больше (чтобы увеличить точность).

Вариант решения:
Генерировать искусственные фотографии игрового мира **GTA V** с идеальной семантической разметкой.



1. Генерация синтетического датасета (GTA V)

Хочется воспроизвести путь по которому пошли авторы URSA **[4]** датасета:

1) Закодировать в каждый пиксель **FMSS-комбинацию** из идентификатора модели, шейдера и текстуры.

2) Найти сопоставление из **FMSS-комбинации** в предсказываемые классы. (с помощью **Amazon Mechanical Turk**)

3) Сгенерировать размеченные датасеты сфотографированные с воображаемого квадрокоптера. (авторы URSA опубликовали набор данных с ракурса воображаемой машины, что нам не подходит)

4) Обучить модель по новым данным. Посмотреть как изменится точность.

5) Опубликовать подробную инструкцию для простого воспроизведения результатов и сопоставление из **FMSS** в предсказываемые классы.

1. Генерация синтетического датасета (GTA V)

Сгенерировать **FMSS**-изображение можно одним из двух вариантов:

1. Перехватить и сохранить все **DirectX API-вызовы** и тем самым получить возможность воспроизвести рендеринг вне игрового движка с произвольными изменениями текстур и шейдеров **[1]**

2. Подменить в игре текстуры на одноцветные и шейдеры на примитивные (чтобы отключить тени, освещение и т.п.) **[4]**

Неизвестно какой вариант проще. Но первый вариант легче в будущем адаптировать для других игр. При этом второй вариант быстрее работает.

Требуется понимать как работают фрагментные шейдеры.

1. Генерация синтетического датасета (GTA V)

Ссылки:

[1] Playing for Data: Ground Truth from Computer Games, **Richter et al., 2016**

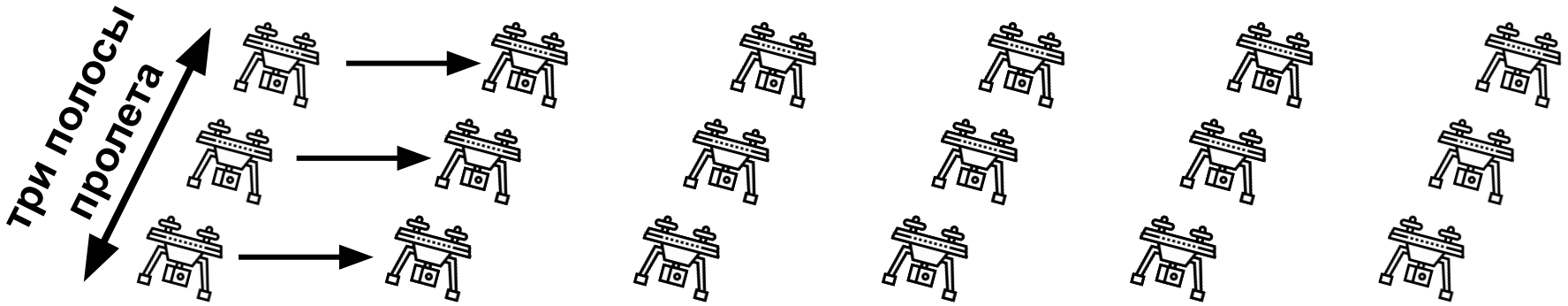
[2] Driving in the Matrix: Can Virtual Worlds Replace Human-Generated Annotations for Real World Tasks? **Johnson-Roberson et al., 2016**

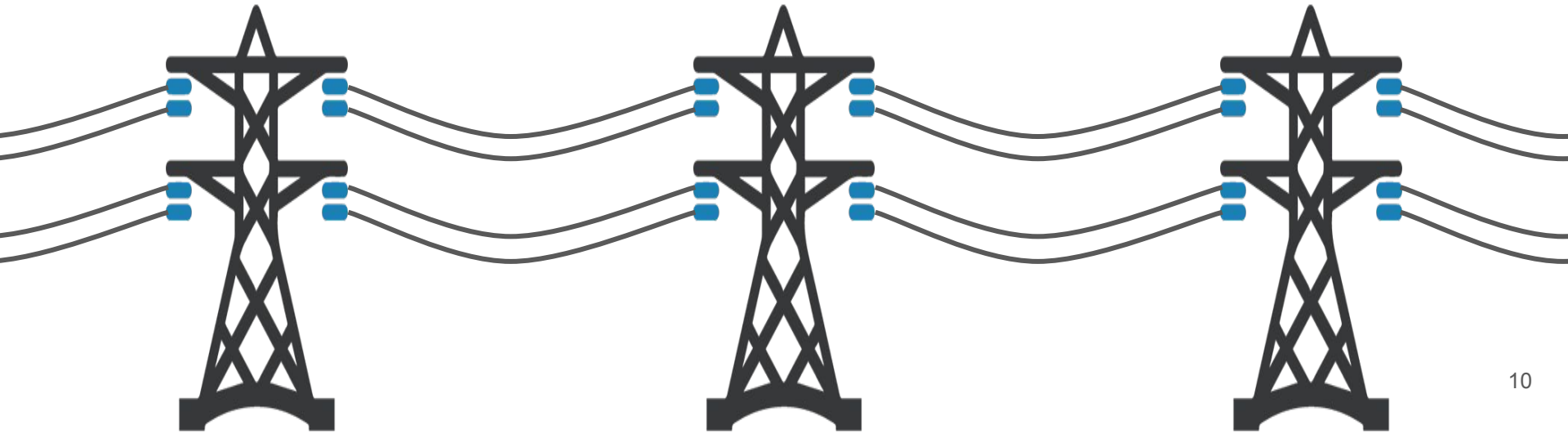
[3] Playing for Benchmarks, **Richter et al., 2017**

[4] Unlimited Road-scene Synthetic Annotation (URSA) Dataset, **Angus et al., 2018**

2. 3D обнаружение проводов ЛЭП

Пусть есть многокилометровая цепочка ЛЭП сфотографированная с беспилотника в несколько полос (десятки тысяч фотографий):

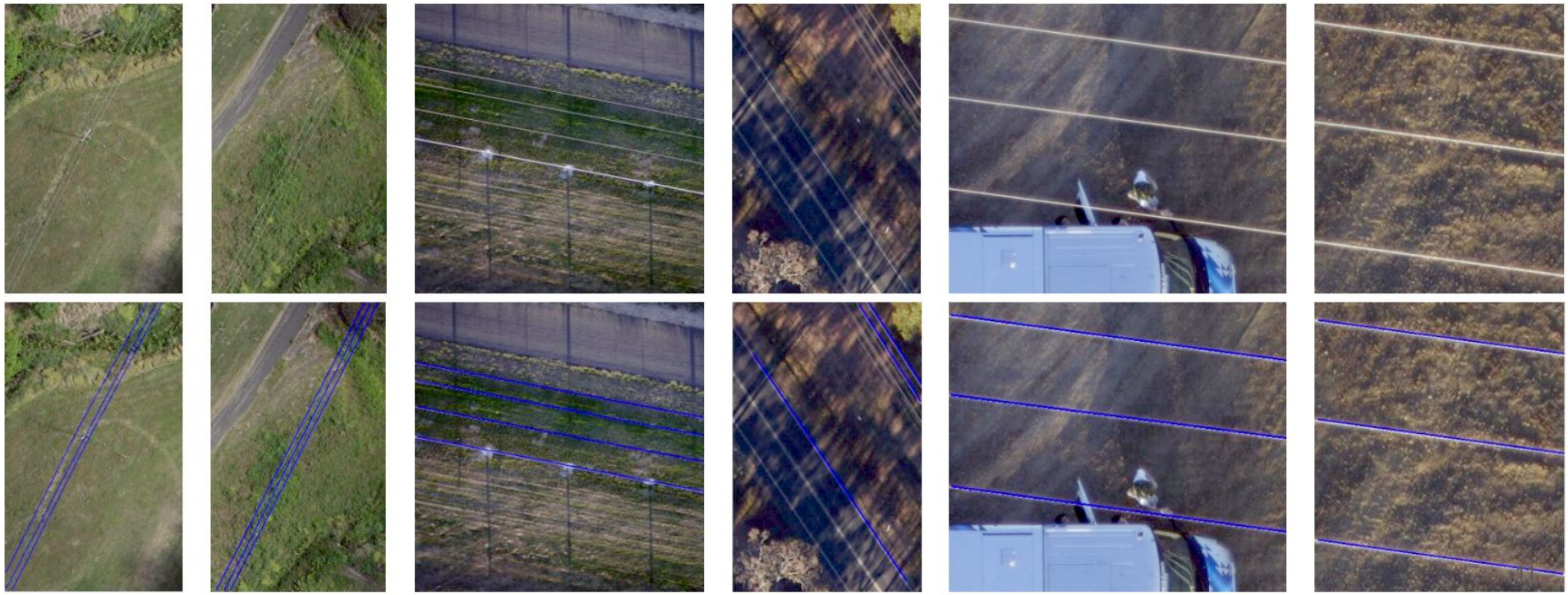




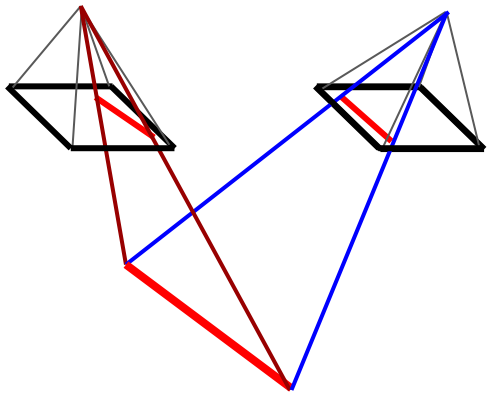
2. 3D обнаружение проводов ЛЭП

Положение каждой фотографии восстанавливается автоматически с большой точностью.

Нужно обнаружить на каждой фотографии длинные прямые (на самом деле слегка кривые) детектором границ и преобразованием Хаффа:



2. 3D обнаружение проводов ЛЭП



Имея положение двух камер, их ракурс и обнаруженный на фотографиях провод - находится прямая в 3D пространстве:

Чтобы найти наблюдаемый с двух камер провод в 3D пространстве - достаточно пересечь две плоскости (т.е. провод из каждого изображения порождает плоскость).

Неоднозначность сопоставления проводов между наблюдениями на фотографиях решается за счет избыточности - **требуется хотя бы три полосы** фотографий, и это требование позволяет устроить голосование и фильтрацию ложных соответствий проводов между фотографиями.

2. 3D обнаружение проводов ЛЭП

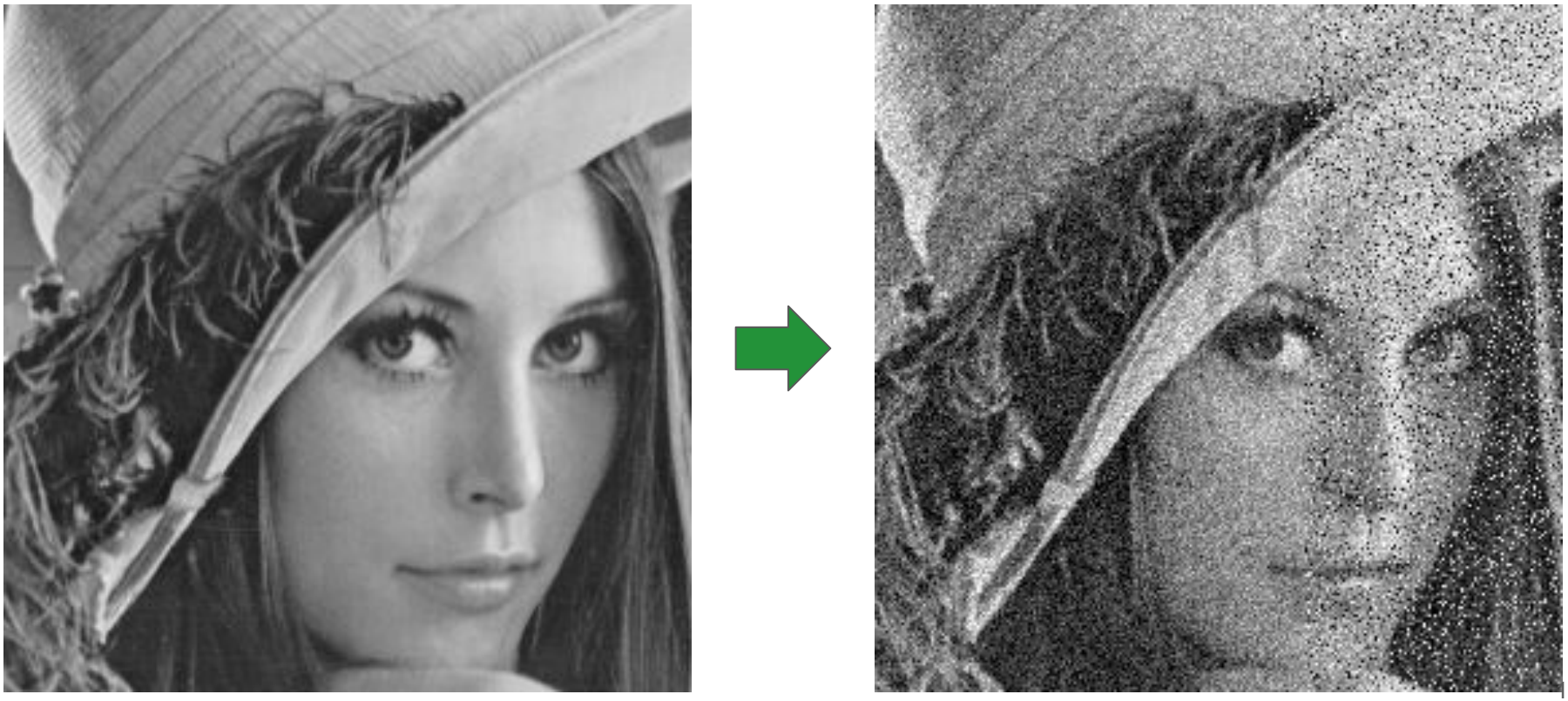
Ссылки:

[1] FAST POWER LINE DETECTION AND LOCALIZATION USING STEERABLE FILTER FOR ACTIVE UAV GUIDANCE, **Yuee et al., 2012**

[2] 3D Power Line Extraction from Multiple Aerial Images, **Oh et al., 2017**

3. Пусть есть шумное изображение

- Гауссовый шум по всей поверхности изображения
- Salt-and-pepper шум на правой части изображения



где градиент по изображению:

$$\begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}$$

3. Image Denoising, формулировка задачи

Нужно найти такое изображение x чтобы сумма полной вариации $\| \nabla x \|$ и отличия от оригинального наблюдения $\| x - f \|^2$ было минимально:

$$\min_x \| \nabla x \| + \frac{\lambda}{2} \| x - f \|^2$$

x

f

3. Image Denoising (TV-L1, **много наблюдений**)

TV-L1 model для множества наблюдений:

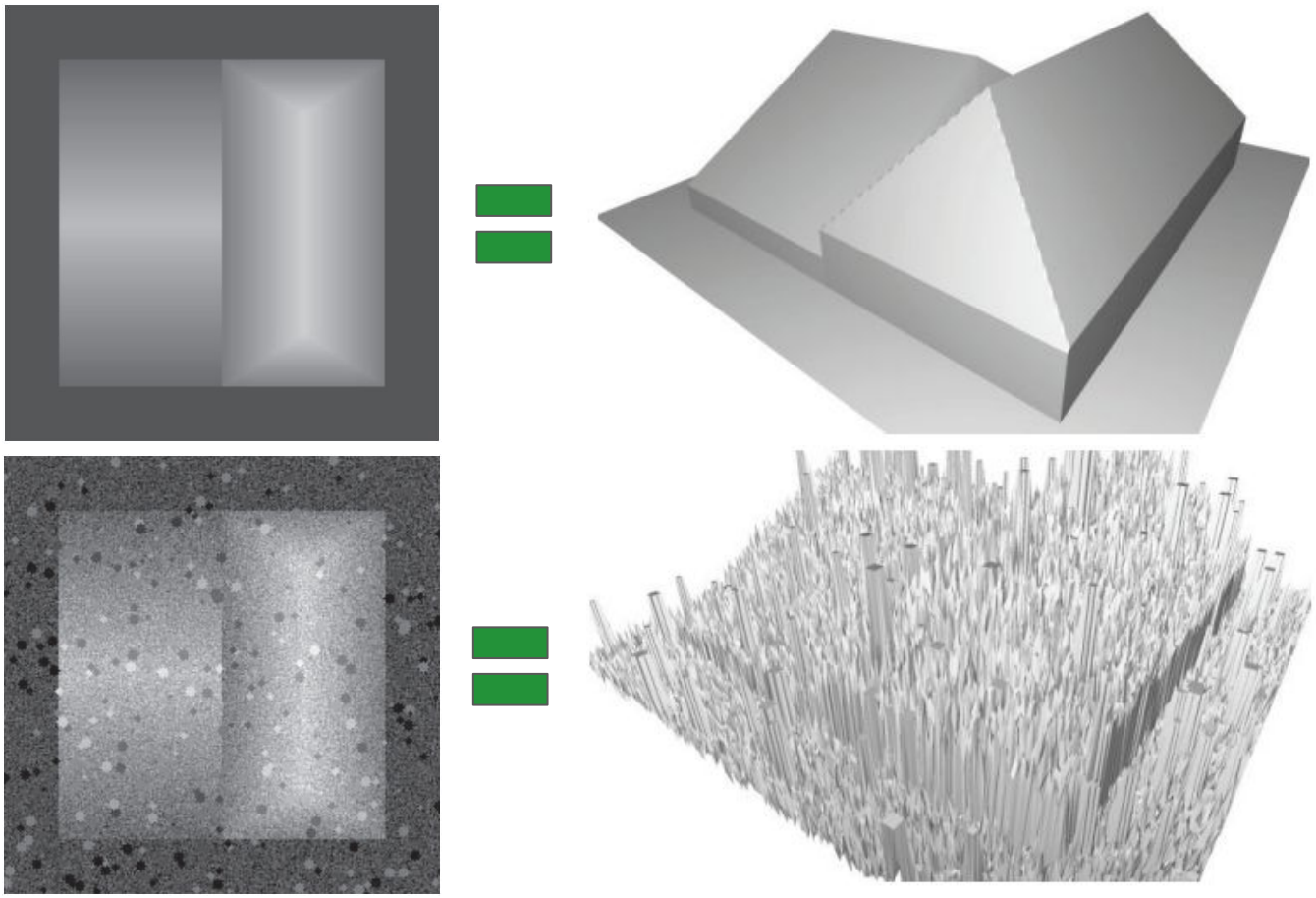
$$\min_x \| \nabla x \| + \lambda \sum_i \| x - f_i \|$$

одно из 5 очень шумных наблюдений:

Подробнее: [IPython notebook with ROF and TVL1 denoising (with math!)]

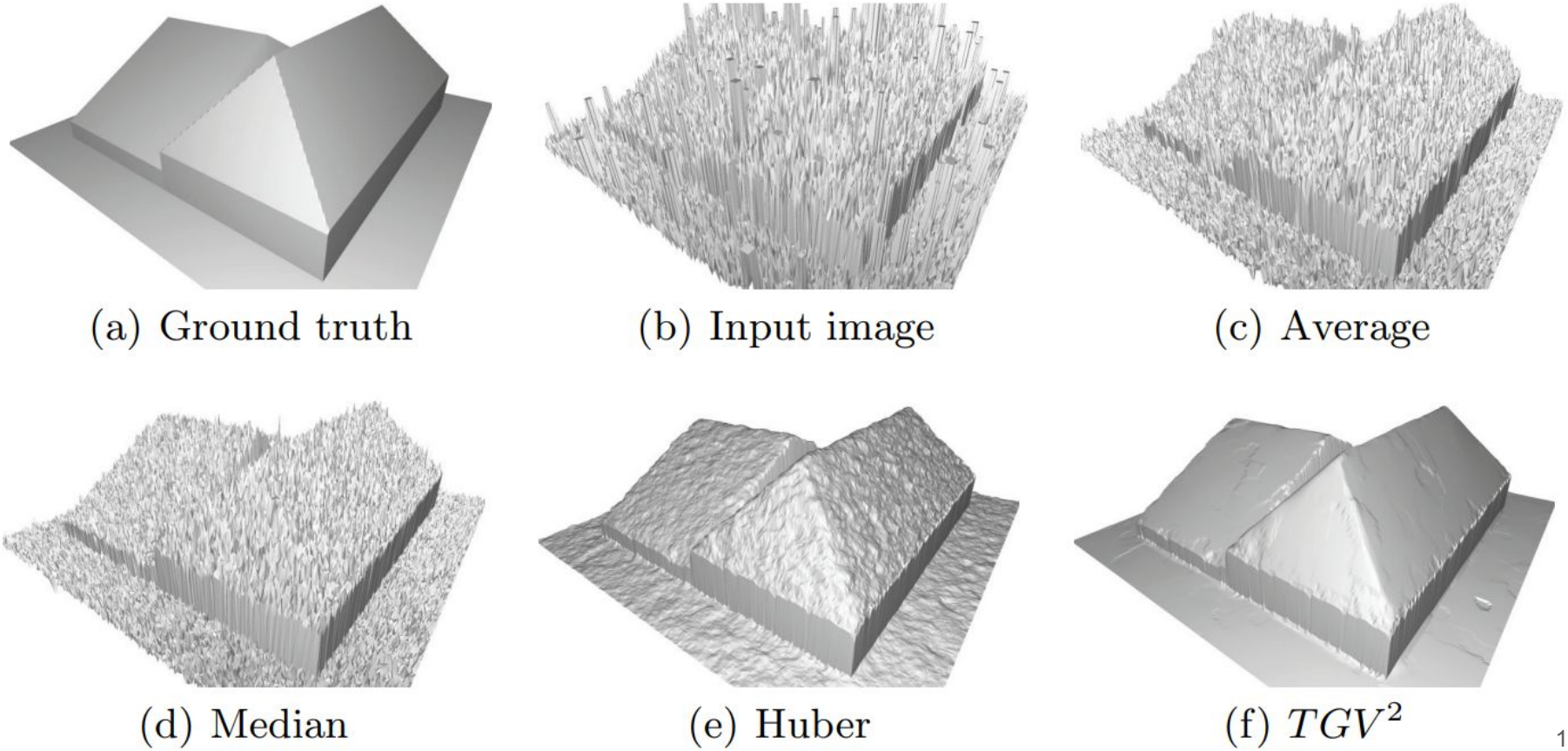
3. Аналогично реконструируется DEM

Вместо шумных картинок - шумные попиксельные замеры высоты. Нужно найти по ним DEM (**Digital Elevation Model**), т.е. карту высот поверхности:



3. DEM по нескольким наблюдениям

Сначала хочется реализовать TGV^2 метод (нижняя правая картинка) - это понятно как делать и это удобно писать на **numpy**. **[2, 3]**



(a) Ground truth (b) Input image (c) Average

(d) Median (e) Huber (f) TGV^2

3. План на задачу

1) Сначала хочется **реализовать** TGV^2 **метод**: это ясно как сделать и это просто сделать на **numpy**. **[2, 3]**

2) Затем хочется **переложить на GPU (pyOpenCL)** ради очень большого ускорения и т.о. ради возможности легко экспериментировать. **[5]**

3) Затем хочется придумать масштабируемую для реальных данных вычислительную схему. **Это основная часть задачи.**

   Проблема: перекрытие наблюдений может быть сколь угодно великим, а хранить в видеопамяти десятки или сотни наблюдений (т.е. сотни значений наблюдений высот для каждого пикселя) не всегда возможно.

   Решение: в каждом пикселе хранить лишь несколько наиболее вероятных наблюдаемых высот/закодировать наблюдаемые высоты в гистограммы/использовать soft-binning/что-то еще.

3. Построение карты высот (DEM) по картам глубины численными методами

Ссылки:

[1] IPython notebook with ROF and TVL1 denoising (with math!)

[2] TGV-Fusion, **Pock et al., 2011** - TGV^2 метод

[3] Fusion of multi-resolution digital surface models, **Kuschk et al., 2013** - почти тот же метод, но нет ошибок в численных формулах

[4] Лекция про image denoising и TV reconstruction

[5] Пример иллюстрирующий использование pyOpenCL

Организационные детали

- Язык для первых двух задач: **C++**
- Для третьей задачи: либо **Python с OpenCL** (через pyOpenCL, см. пример), либо **C++ с OpenCL/CUDA**
- Адрес офиса: Дегтярный Переулок, 11 лит. Б
- С любыми вопросами можно писать на polarnick@agisoft.com или http://t.me/PolarNick239
- К предложенным темам есть тестовые задания - вышлю по запросу
- Во всех практиках нужно будет читать статьи

Вопросы?

Agisoft

Полярный Николай

polarnick@agisoft.com

Предложенные задачи

1. Генерация синтетического датасета (GTA V)
2. 3D обнаружение проводов ЛЭП
3. Построение карты высот по картам глубины численными методами

Agisoft
Полярный Николай
polarnick@agisoft.com