

Введение в фотограмметрию

Уточнение положения камер и автокалибровка, часть 2

Фотограмметрия. Лекция 9



- Ceres solver
- Автодифференцирование
- Функции потерь
- Hierarchical SFM
- Геопривязка, RTK

Rolling shutter

- 1) Некоторые камеры делают снимок строка за строкой
- 2) Если объект или камера движется, то каждая строка наблюдает сцену с разной перспективой

Rolling shutter

- 1) Некоторые камеры делают снимок строка за строкой
- 2) Если объект или камера движется, то каждая строка наблюдает сцену с разной перспективой
- 3) Линейная модель rolling shutter:
 - а) камера движется и вращается равномерно и прямолинейно (пропорционально у-координате)

Rolling shutter

- 1) Некоторые камеры делают снимок строка за строчкой
- 2) Если объект или камера движется, то каждая строка наблюдает сцену с разной перспективой
- 3) Линейная модель rolling shutter:
 - a) камера движется и вращается равномерно и прямолинейно (пропорционально у-координате)
 - b) за время съемки кадра камера сдвинулась на ΔT , повернулась на ΔR
 - c) апдейт параметров модели камеры:

$$T, R, \{k_1-k_3, c_x, c_y, f\}, \underline{\Delta T, \Delta R}$$

Rolling shutter

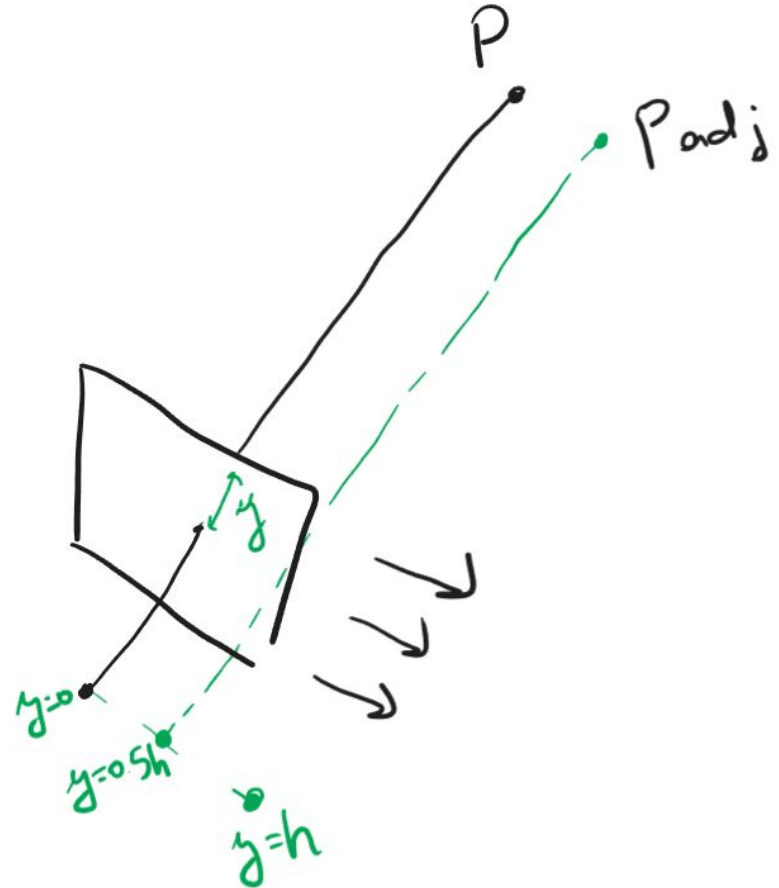
- 1) Некоторые камеры делают снимок строка за строчкой
- 2) Если объект или камера движется, то каждая строка наблюдает сцену с разной перспективой
- 3) Линейная модель rolling shutter:
 - a) камера движется и вращается равномерно и прямолинейно (пропорционально у-координате)
 - b) за время съемки кадра камера сдвинулась на ΔT , повернулась на ΔR
 - c) апдейт параметров модели камеры:

$$T, R, \{k_1, k_3, c_x, c_y, f\}, \underline{\Delta T, \Delta R}$$

- d) если камера ускорялась то наши полномочия все

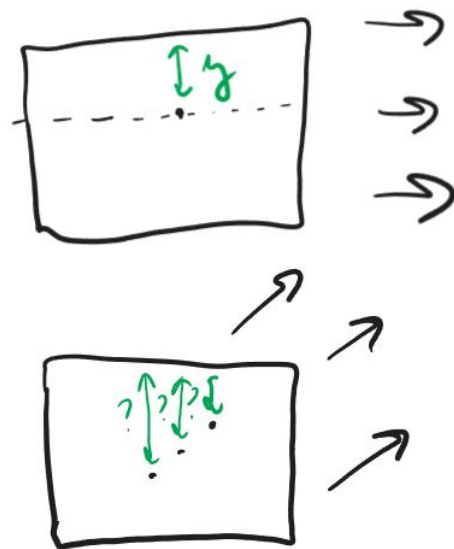
Rolling shutter

- 1) Как выглядит **unproject**?
- 2) Без хитростей, знаем y -координату точки, следовательно знаем на сколько сдвинулась камера относительно начала снимка



Rolling shutter

- 1) Как выглядит **project**?
- 2) у-координата зависит от того, в какой точке была камера в момент съемки, а мы этого не узнаем (в общем случае), пока не найдем у-координату

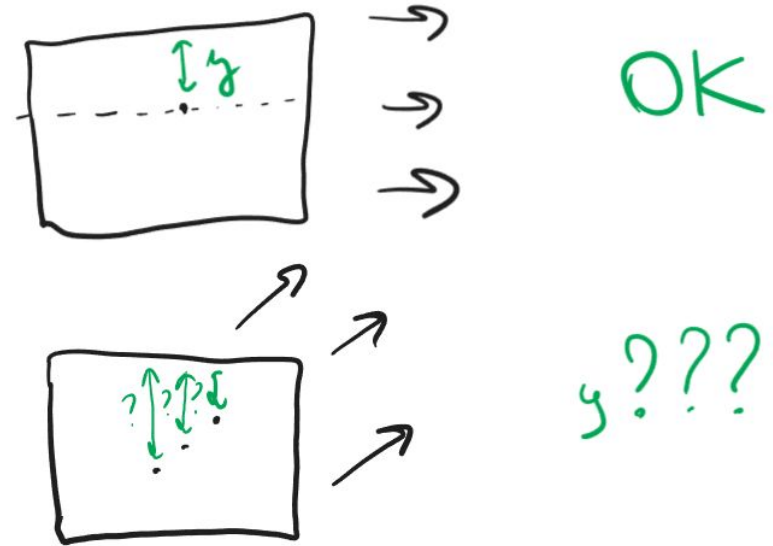


OK

y???

Rolling shutter

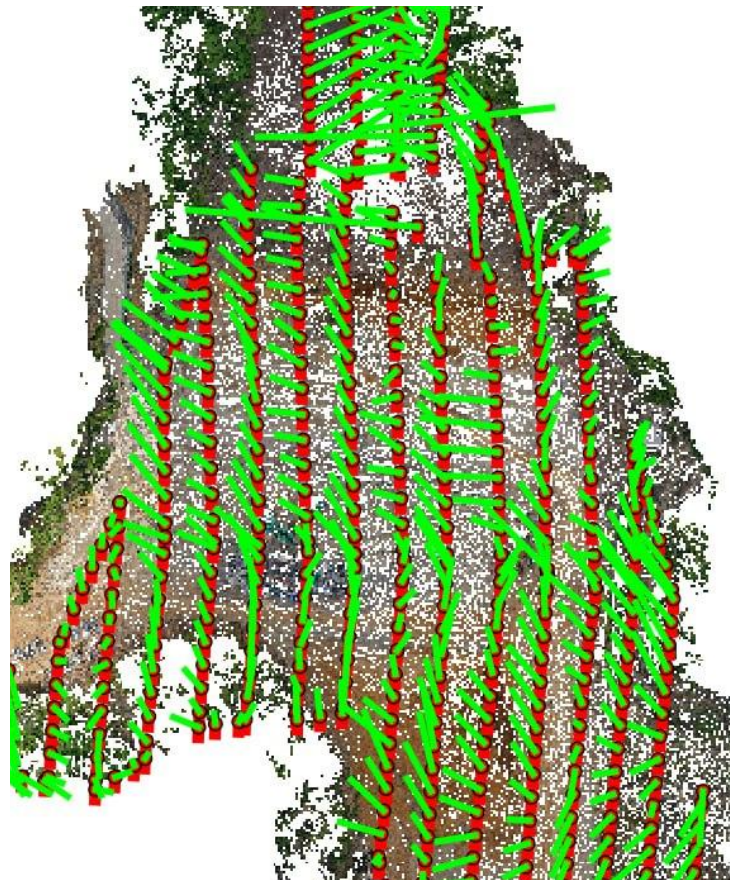
- 1) Как выглядит **project**?
- 2) у-координата зависит от того, в какой точке была камера в момент съемки, а мы этого не узнаем (в общем случае), пока не найдем у-координату
- 3) при движении ровно вбок все хорошо сразу
- 4) при произвольном движении - итерации



$$p_x = \text{calib.project}(p)$$
$$\tilde{p} = \text{shutter.project}(p, p_x \cdot y)$$
$$n \text{ times } \uparrow p_x = \text{calib.project}(\tilde{p})$$

Rolling shutter

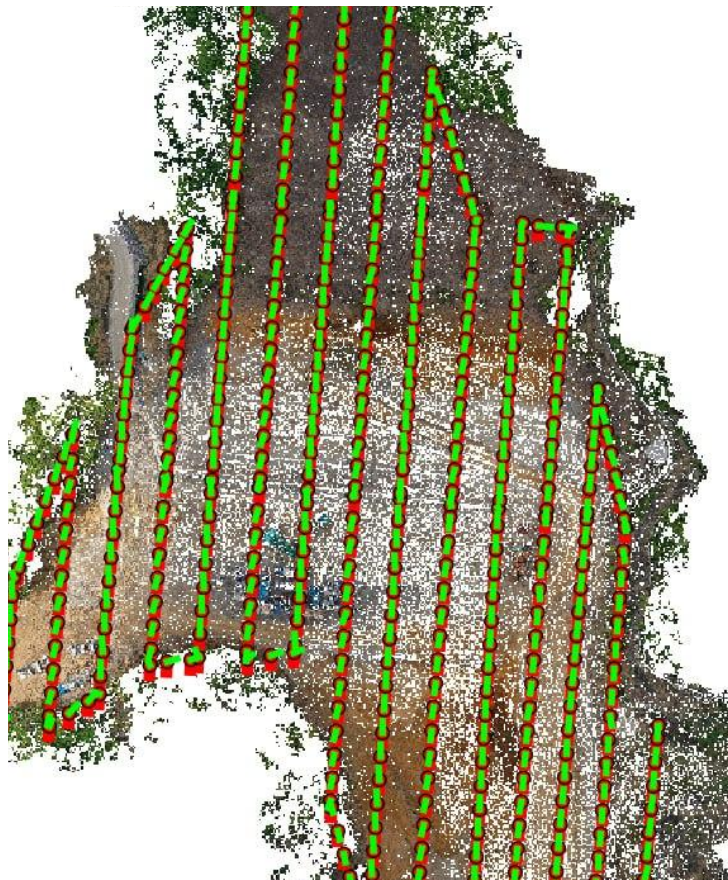
- 1) иногда 6 параметров слишком много и калибровочные параметры разъезжаются во время ВА
- 2) может помочь зажать модель: например, для дрона снимающего заметающей прямой можно оставить два параметра T_x , T_y вместо шести



Rolling shutter

- 1) иногда 6 параметров слишком много и калибровочные параметры разъезжаются во время ВА
- 2) может помочь зажать модель: например, для дрона снимающего заметающей прямой можно оставить два параметра T_x , T_y вместо шести

похоже на траекторию полета!



Model selection

- 1) Бывают вырожденные конфигурации сцены, когда оптимальное решение оптимизационной задачи ВА не единственно

Model selection

- 1) Бывают вырожденные конфигурации сцены, когда оптимальное решение оптимизационной задачи ВА не единственно
- 2) Пример: Плоскость
 - a) снимок вблизи с маленьким f (большим FOV)
 - b) снимок издалека с большим f (маленьким FOV) (большим зумом)
 - c) если поверхность не плоская то как меняется?

Model selection

- 1) Бывают вырожденные конфигурации сцены, когда оптимальное решение оптимизационной задачи ВА не единственно
- 2) Пример: Плоскость
 - a) снимок вблизи с маленьким f (большим FOV)
 - b) снимок издалека с большим f (маленьким FOV) (большим зумом)
 - c) если поверхность не плоская то как меняется?
- 3) В случае вырожденной конфигурации не можем найти калибровочные параметры однозначно, появляется степень свободы

Model selection

- 1) Бывают вырожденные конфигурации сцены, когда оптимальное решение оптимизационной задачи ВА не единственно
- 2) Пример: Плоскость
 - a) снимок вблизи с маленьким f (большим FOV)
 - b) снимок издалека с большим f (маленьким FOV) (большим зумом)
 - c) если поверхность не плоская то как меняется?
- 3) В случае вырожденной конфигурации не можем найти калибровочные параметры однозначно, появляется степень свободы
- 4) Какие-то параметры надо зафиксировать

Model selection

1) Какие параметры фиксировать?

Model selection

- 1) Какие параметры фиксировать?
- 2) Заранее не знаем. Можем к нашим наблюдениям добавить тяготение всех калибровочных параметров к нулю (sticky prior)

Model selection

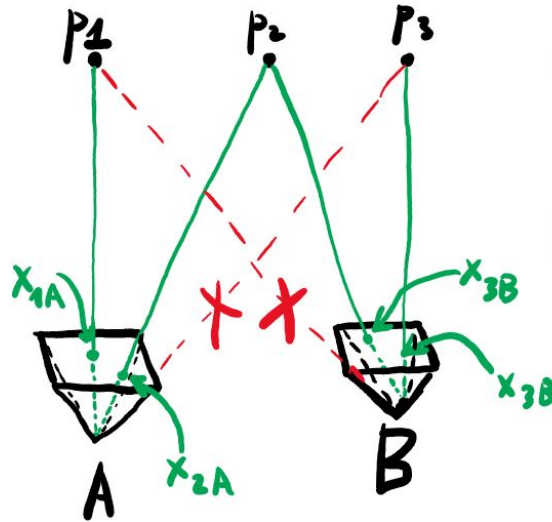
- 1) Какие параметры фиксировать?
- 2) Заранее не знаем. Можем к нашим наблюдениям добавить тяготение всех калибровочных параметров к нулю (sticky prior)
- 3) Можем посчитать корреляцию калибровочных коэффициентов и отключить слишком скоррелированные параметры

Model selection

- 1) Какие параметры фиксировать?
- 2) Заранее не знаем. Можем к нашим наблюдениям добавить тяготение всех калибровочных параметров к нулю (sticky prior)
- 3) Можем посчитать корреляцию калибровочных коэффициентов и отключить слишком скоррелированные параметры
- 4) [Bundle Adjustment: A Modern Synthesis](#) (глава Model Selection)

Bundle Adjustment

- 1) Оптимизационная задача в виде наименьших квадратов
- 2) Решать самому (трудоемко :)



проекция $g(x): \mathbb{R}^3 \rightarrow \mathbb{R}^2$

остатки (residuals):

$$r_{1A}(x) = g_A(P_1) - x_{1A}$$

$$r_{1B}(x) = g_B(P_1) - x_{1B}$$

$$r_{2A}(x) = g_A(P_2) - x_{2A}$$

$$r_{2B}(x) = g_B(P_2) - x_{2B}$$

наблюдения

минимизация

$$f(x) = \frac{1}{2} \sum_{i,j=1..m} (g_j(P_i) - x_{ij})^2$$

Ceres Solver

Ceres Solver - библиотека на **C++**. Позволяет просто написав функцию которая считает “что мы хотим минимизировать” - найти решение.

Ceres Solver

Ceres Solver - библиотека на **C++**. Позволяет просто написав функцию которая считает “что мы хотим минимизировать” - найти решение.

- Функция должна быть выпуклой (или почти)
- Реализованы все нужные методы для Bundle Adjustment
- Библиотека автоматически найдет производные

Ceres-solver

Non-linear Least Squares

Introduction

Ceres can solve bounds constrained robustified non-linear least squares problems of the form

$$\min_{\mathbf{x}} \quad \frac{1}{2} \sum_i \left(\|f_i(x_{i_1}, \dots, x_{i_k})\|^2 \right)$$

Ceres-solver

Non-linear Least Squares

Introduction

Ceres can solve bounds constrained robustified non-linear least squares problems of the form

$$\begin{aligned} \min_{\mathbf{x}} \quad & \frac{1}{2} \sum_i \rho_i \left(\|f_i(x_{i_1}, \dots, x_{i_k})\|^2 \right) \\ \text{s.t.} \quad & l_j \leq x_j \leq u_j \end{aligned}$$

Ceres-solver

Non-linear Least Squares

Introduction

Ceres can solve bounds constrained robustified non-linear least squares problems of the form

$$\min_{\mathbf{x}} \frac{1}{2} \sum_i \rho_i \left(\|f_i(x_{i_1}, \dots, x_{i_k})\|^2 \right)$$

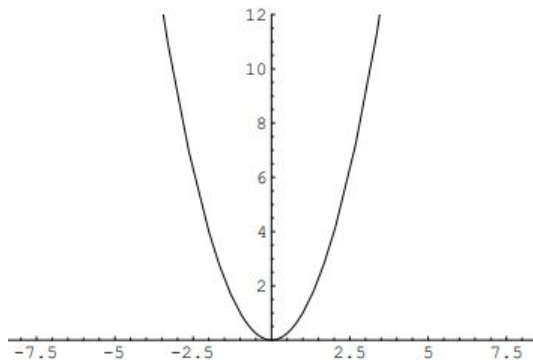
s.t. $l_j \leq x_j \leq u_j$

The diagram illustrates the mathematical formulation of a non-linear least squares problem with bounds and robustification, annotated with handwritten Russian text:

- Loss Function - функция потерь**: Points to the ρ_i term in the summation.
- Residual Func. не-линейная**: Points to the f_i term in the summation.
- (Parameter Block) блок параметров от конкретных значений Cost Function $f_i(\dots)$** : Points to the parameters x_{i_1}, \dots, x_{i_k} inside the residual function.
- Residual Block #i блок нелинейки**: Points to the entire residual term $\rho_i \left(\|f_i(x_{i_1}, \dots, x_{i_k})\|^2 \right)$.
- ограничения (constraints) на конкретный параметр**: Points to the constraint $l_j \leq x_j \leq u_j$.
- искомый параметр**: Points to the minimization variable \mathbf{x} .

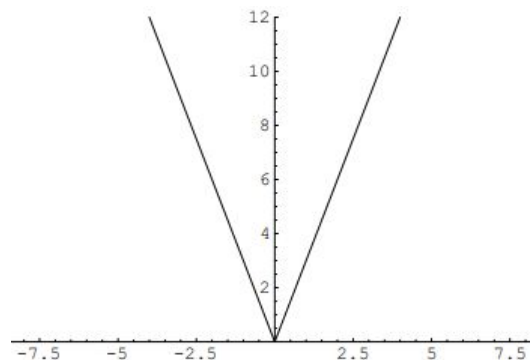
Loss functions (функции потерь)

$$C(\delta) = \delta^2$$



Squared-Error
L2 norm
Trivial Loss

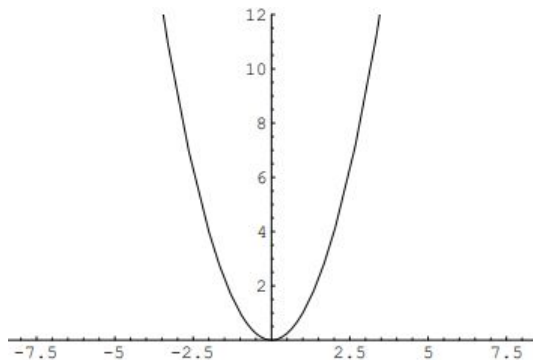
$$C(\delta) = |\delta|$$



L1 norm
Total Variation
Absolute Loss

Loss functions (функции потерь)

$$C(\delta) = \delta^2$$



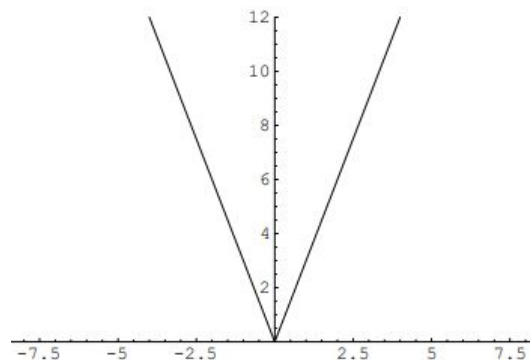
Squared-Error

L2 norm

Trivial Loss

- Наблюдения-выбросы (outliers) имеют сильное влияние

$$C(\delta) = |\delta|$$



L1 norm

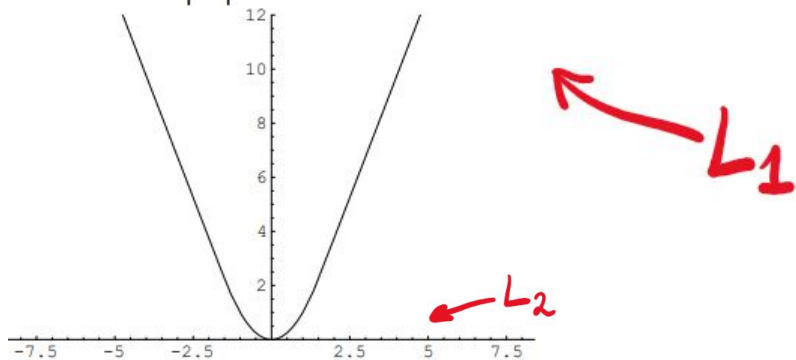
Total Variation

Absolute Loss

- Устойчива к выбросам (robust to outliers)
- Ищет медиану значений, игнорирует остальные значения

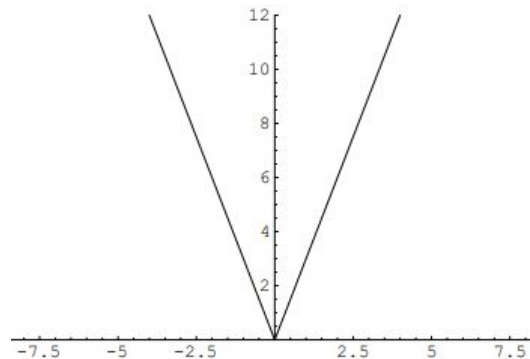
Loss functions (функции потерь)

$$\begin{aligned} C(\delta) &= \delta^2 \text{ for } |\delta| < b \\ &= 2b|\delta| - b^2 \text{ otherwise} \end{aligned}$$



Huber Loss

$$C(\delta) = |\delta|$$



L1 norm

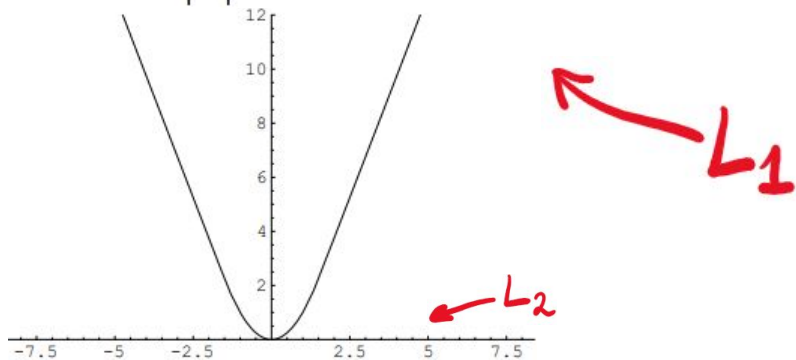
Total Variation

Absolute Loss

- Устойчива к выбросам (robust to outliers)
- Ищет медиану значений, игнорирует остальные значения

Loss functions (функции потерь)

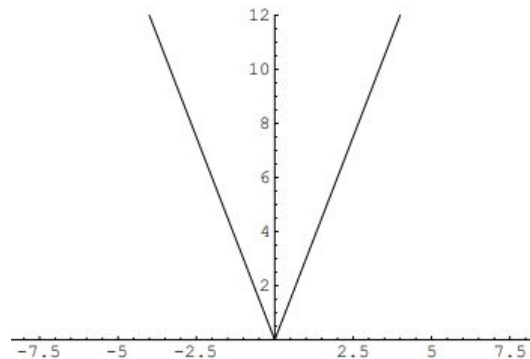
$$\begin{aligned} C(\delta) &= \delta^2 \text{ for } |\delta| < b \\ &= 2b|\delta| - b^2 \text{ otherwise} \end{aligned}$$



Huber Loss

- Устойчива к выбросам (robust to outliers)
- Нет проблемы про медиану (т.е. хорошо приближает норм. распределение)

$$C(\delta) = |\delta|$$



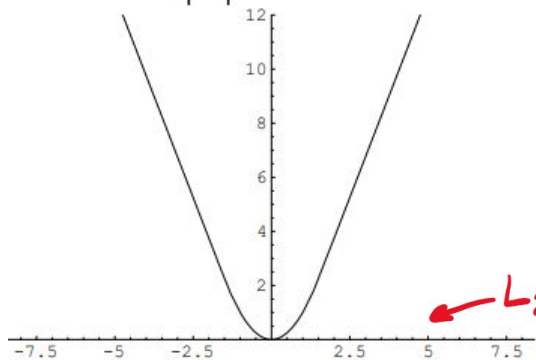
L1 norm

Total Variation
Absolute Loss

- Устойчива к выбросам (robust to outliers)
 - Ищет медиану значений, игнорирует остальные значения

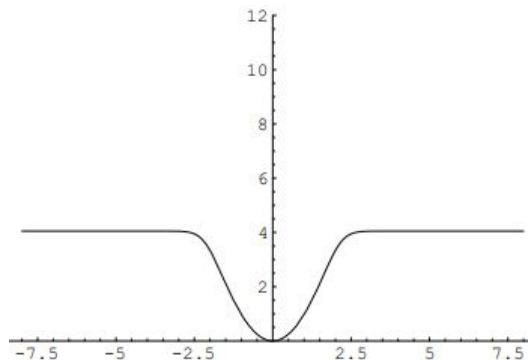
Loss functions (функции потерь)

$$\begin{aligned} C(\delta) &= \delta^2 \text{ for } |\delta| < b \\ &= 2b|\delta| - b^2 \text{ otherwise} \end{aligned}$$



Huber Loss

$$C(\delta) = -\log(\exp(-\delta^2) + \epsilon) \text{ and } \epsilon = \exp(-\alpha^2)$$

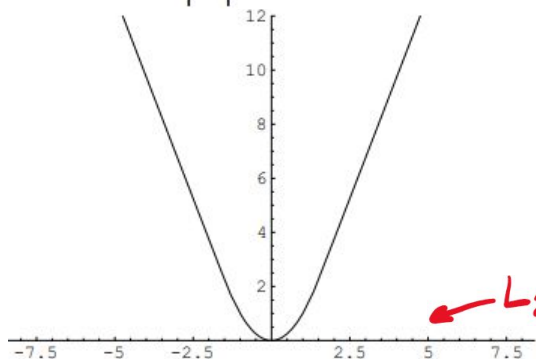


Blake-Zisserman

- Устойчива к выбросам
(robust to outliers)
- Нет проблемы про медиану
(т.е. хорошо приближает норм.
распределение)

Loss functions (функции потерь)

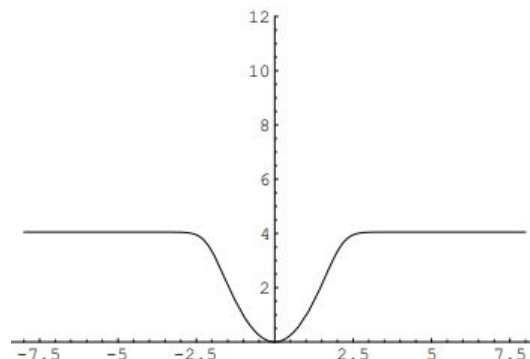
$$\begin{aligned} C(\delta) &= \delta^2 \text{ for } |\delta| < b \\ &= 2b|\delta| - b^2 \text{ otherwise} \end{aligned}$$



Huber Loss

- Устойчива к выбросам (robust to outliers)
- Нет проблемы про медиану (т.е. хорошо приближает норм. распределение)

$$C(\delta) = -\log(\exp(-\delta^2) + \epsilon) \text{ and } \epsilon = \exp(-\alpha^2)$$



Blake-Zisserman

- Супер устойчива к выбросам
- Если изначальное приближение - плохое, то в нем и застрянем

Ceres-solver

Non-linear Least Squares

Introduction

Ceres can solve bounds constrained robustified non-linear least squares problems of the form

$$\min_{\mathbf{x}} \frac{1}{2} \sum_i \rho_i \left(\|f_i(x_{i_1}, \dots, x_{i_k})\|^2 \right)$$

s.t. $l_j \leq x_j \leq u_j$

The diagram illustrates the mathematical formulation of a non-linear least squares problem with bounds and robustification, annotated with handwritten Russian text:

- Loss Function - функция потерь**: Points to the ρ_i term in the summation.
- Residual Func. не-линейна**: Points to the f_i term in the summation.
- (Parameter Block) блок параметров от конкретных значений Cost Function $f_i(\dots)$** : Points to the parameters x_{i_1}, \dots, x_{i_k} inside the residual function.
- Residual Block #i блок нелинейн**: Points to the entire residual term $\rho_i \|f_i\|^2$.
- ограничения (constraints) на конкретный параметр**: Points to the constraint $l_j \leq x_j \leq u_j$.
- искомый параметр**: Points to the minimization variable \mathbf{x} .

Автоматическое дифференцирование

$$f(x+\varepsilon) = f(x) + f'(x)\varepsilon + o(\varepsilon^2)$$

Автоматическое дифференцирование

$$f(x+\varepsilon) = f(x) + f'(x)\varepsilon + o(\varepsilon^2)$$

$$\begin{aligned} f(x+\varepsilon)g(x+\varepsilon) &= (f + f'\varepsilon)(g + g'\varepsilon) = \\ &= fg + (f'g + fg')\varepsilon + f'g'\varepsilon^2 \end{aligned}$$

Автоматическое дифференцирование

$$f(x+\varepsilon) = f(x) + f'(x)\varepsilon + o(\varepsilon^2)$$

$$\begin{aligned} f(x+\varepsilon)g(x+\varepsilon) &= (f + f'\varepsilon)(g + g'\varepsilon) = \\ &= fg + (f'g + fg')\varepsilon + f'g'\varepsilon^2 \end{aligned}$$

$$\frac{f(x+\varepsilon)g(x+\varepsilon) - f(x)g(x)}{\varepsilon} = f'g + fg' + f'g' \cdot \varepsilon$$

Автоматическое дифференцирование

$$f(x+\varepsilon) = f(x) + f'(x)\varepsilon + o(\varepsilon^2)$$

$$\begin{aligned} f(x+\varepsilon)g(x+\varepsilon) &= (f + f'\varepsilon)(g + g'\varepsilon) = \\ &= fg + (f'g + fg')\varepsilon + f'g'\varepsilon^2 \end{aligned}$$

$$\frac{f(x+\varepsilon)g(x+\varepsilon) - f(x)g(x)}{\varepsilon} = f'g + fg' + f'g' \cdot \varepsilon$$

$$(fg)' \equiv \lim_{\varepsilon \rightarrow 0} \frac{f(x+\varepsilon)g(x+\varepsilon) - f(x)g(x)}{\varepsilon} = f'g + fg'$$

Автоматическое дифференцирование

$$f(g(x+\varepsilon)) = f(g + g' \varepsilon) =$$

Автоматическое дифференцирование

$$\begin{aligned} f(g(x+\varepsilon)) &= f(g + g' \varepsilon) = \\ &= f(g(x)) + f'(g(x)) \cdot g'(x) \cdot \varepsilon \end{aligned}$$

Автоматическое дифференцирование

$$\begin{aligned} f(g(x+\varepsilon)) &= f(g + g' \varepsilon) = \\ &= f(g(x)) + f'(g(x)) \cdot g'(x) \cdot \varepsilon \end{aligned}$$

$$f'(g) \equiv \lim_{\varepsilon \rightarrow 0} \frac{f(g(x+\varepsilon)) - f(g(x))}{\varepsilon} \rightarrow f'(g(x)) g'(x)$$

Автоматическое дифференцирование: **Dual Numbers**

В исходниках **Ceres Solver** есть очень хорошее описание:

[github/ceres-solver/include/ceres/jet.h](https://github.com/ceres-solver/include/ceres/jet.h)

По аналогии с комплексными числами $a + b \cdot i$ дополним число бесконечно малой компонентой. И введем ϵ по аналогии с мнимой единицей i ($i^2 = -1$)

Автоматическое дифференцирование: **Dual Numbers**

В исходниках **Ceres Solver** есть очень хорошее описание:

[github/ceres-solver/include/ceres/jet.h](https://github.com/ceres-solver/include/ceres/jet.h)

По аналогии с комплексными числами $a + b \cdot i$ дополним число бесконечно малой компонентой. И введем ϵ по аналогии с мнимой единицей i ($i^2 = -1$)

$x + x' \cdot \epsilon$
↑
бесконечно малое:
 $\epsilon^2 = 0$

Автоматическое дифференцирование: **Dual Numbers**

A handwritten diagram illustrating the concept of dual numbers. On the left, the symbol \mathbb{R} is written. An arrow points from \mathbb{R} to the expression $x + x' \cdot \epsilon$. Another arrow points from $x + x' \cdot \epsilon$ to the symbol ϵ . To the right of ϵ , the text "бесконечно малое:" is written in red, with an upward arrow pointing to ϵ . Below this text, the equation $\epsilon^2 = 0$ is written in red.

$$x + x' \cdot \epsilon$$

бесконечно малое:
 $\epsilon^2 = 0$

Автоматическое дифференцирование: **Dual Numbers**

\mathbb{R} \nearrow $x + x' \cdot \epsilon$

\uparrow
бесконечно
малое:
 $\epsilon^2 = 0$

$$f(x) = x^2$$

Автоматическое дифференцирование: **Dual Numbers**

$x + x' \cdot \epsilon$
↓ ↙ ↘
 \mathbb{R} ↗
↑
бесконечно
малое:
 $\epsilon^2 = 0$

$$f(x) = x^2$$

$$f(10) = 100 \quad f'_x(10) = 20$$

Автоматическое дифференцирование: **Dual Numbers**

\mathbb{R} $x + x' \cdot \epsilon$
↑
бесконечно
малое:
 $\epsilon^2 = 0$

$$f(x) = x^2$$

$$f(10) = 100 \quad f'_x(10) = 20$$

$$f(10 + \epsilon) = (10 + \epsilon)^2 = 100 + 20 \cdot \epsilon + \epsilon^2$$

Автоматическое дифференцирование: **Dual Numbers**

\mathbb{R} $x + x' \cdot \epsilon$
↑
бесконечно
малое:
 $\epsilon^2 = 0$

$$f(x) = x^2$$

$$f(10) = 100 \quad f'_x(10) = 20$$

$$f(10 + \epsilon) = (10 + \epsilon)^2 = 100 + 20 \cdot \epsilon + \cancel{\epsilon^2}$$

"0"

Автоматическое дифференцирование: Dual Numbers

\mathbb{R} $x + x' \cdot \epsilon$
↑
бесконечно
малое:
 $\epsilon^2 = 0$

$$f(x) = x^2$$

$$f(10) = 100 \quad f'_x(10) = 20$$

$$f(10 + \epsilon) = (10 + \epsilon)^2 = 100 + 20 \cdot \epsilon + \cancel{\epsilon^2}$$

"0"

dual
number

Автоматическое дифференцирование: Dual Numbers

\mathbb{R} $x + x' \cdot \epsilon$
↑
десконечно
малое:
 $\epsilon^2 = 0$

$$f(x) = x^2$$
$$f(10) = 100 \quad f'_x(10) = 20$$
$$f(10 + \epsilon) = (10 + \epsilon)^2 = \boxed{100 + 20 \cdot \epsilon} + \cancel{\epsilon^2}$$

$\begin{array}{ccc} \parallel & \parallel & \parallel \\ f(10) & f'_x(10) & 0 \end{array}$

dual number

Автоматическое дифференцирование: Dual Numbers

\mathbb{R} $x + x' \cdot \epsilon$
↑
дифференциально
малое:
 $\epsilon^2 = 0$

$$f(x) = x^2$$

$$f(10) = 100 \quad f'_x(10) = 20$$

$$f(10 + \epsilon) = (10 + \epsilon)^2 = \boxed{100 + 20 \cdot \epsilon} + \cancel{\epsilon^2}$$

\parallel \parallel \parallel
 $f(10)$ $f'_x(10)$ 0

dual
number

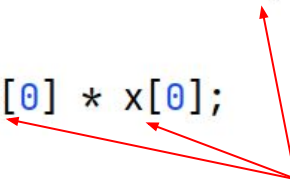
$$f(x_0 + x' \cdot \epsilon) = f(x_0) + f'_x(x_0) \cdot x' \cdot \epsilon$$

Автоматическое дифференцирование: **Dual Numbers**

```
1 struct CostFunctor {  
2     template <typename T>  
3     bool operator()(const T* const x, T* residual) const {  
4         //  $f(x) = x^2$   
5         residual[0] = x[0] * x[0];  
6         return true;  
7     }  
8 };
```

Автоматическое дифференцирование: Dual Numbers

```
1 struct CostFunctor {  
2     template <typename T>  
3     bool operator()(const T* const x, T* residual) const {  
4         //  $f(x) = x^2$   
5         residual[0] = x[0] * x[0];  
6         return true;  
7     }  
8 };
```



искомое
(оптимизируемые параметры)

Автоматическое дифференцирование: Dual Numbers

```
1 struct CostFunctor {  
2     template <typename T>  
3     bool operator()(const T* const x, T* residual) const {  
4         //  $f(x) = x^2$   
5         residual[0] = x[0] * x[0];  
6         return true;  
7     }  
8 };
```

T - подставляемый тип
T=double или **Jet**

искмое
(оптимизируемые параметры)

Автоматическое дифференцирование: Dual Numbers

```
1 struct CostFunctor {  
2     template <typename T>  
3     bool operator()(const T* const x, T* residual) const {  
4         //  $f(x) = x^2$   
5         residual[0] = x[0] * x[0];  
6         return true;  
7     }  
8 };
```

T - подставляемый тип
T=double или Jet

искемое
(оптимизируемые параметры)

```
211 struct Jet {  
285     // The scalar part.  
286     T a;  
288     // The infinitesimal part.  
289     Eigen::Matrix<T, N, 1> v;  
294 };
```

Автоматическое дифференцирование: Dual Numbers

```
1 struct CostFunctor {  
2     template <typename T>  
3     bool operator()(const T* const x, T* residual) const {  
4         //  $f(x) = x^2$   
5         residual[0] = x[0] * x[0];  
6         return true;  
7     }  
8 };
```

T - подставляемый тип

T=double или Jet

искомое
(оптимизируемые параметры)

```
211 struct Jet {  
285     // The scalar part.  
286     T a;  
288     // The infinitesimal part.  
289     Eigen::Matrix<T, N, 1> v;  
294 };
```

Dual Number

Автоматическое дифференцирование: **Dual Numbers**

$$f(x_0 + \epsilon) = f(x_0) + f'_x(x_0) \cdot \epsilon$$

Проверим правило взятия производной сложной функции (**Chain Rule**).

Автоматическое дифференцирование: **Dual Numbers**

$$f(x_0 + \epsilon) = f(x_0) + f'_x(x_0) \cdot \epsilon$$

Проверим правило взятия производной сложной функции (**Chain Rule**).

$$\left[g(f(x_0)) \right]_x' = g'_x(f(x_0)) \cdot f'_x(x_0)$$

Автоматическое дифференцирование: **Dual Numbers**

$$f(x_0 + \epsilon) = f(x_0) + f'_x(x_0) \cdot \epsilon$$

Проверим правило взятия производной сложной функции (**Chain Rule**).

$$\left[g(f(x_0)) \right]'_x = g'_x(f(x_0)) \cdot f'_x(x_0)$$

$$g(f(x_0 + \epsilon)) = g(f(x_0) + f'_x(x_0) \cdot \epsilon)$$

Автоматическое дифференцирование: **Dual Numbers**

$$f(x_0 + \epsilon) = f(x_0) + f'_x(x_0) \cdot \epsilon$$

Проверим правило взятия производной сложной функции (**Chain Rule**).

$$\left[g(f(x_0)) \right]'_x = g'_x(f(x_0)) \cdot f'_x(x_0)$$

$$\begin{aligned} g(f(x_0 + \epsilon)) &= g(f(x_0) + f'_x(x_0) \cdot \epsilon) \\ &= g(f(x_0)) + g'_x(f(x_0)) \cdot f'_x(x_0) \cdot \epsilon \end{aligned}$$

Автоматическое дифференцирование: **Dual Numbers**

$$f(x_0 + \epsilon) = f(x_0) + f'_x(x_0) \cdot \epsilon$$

Проверим правило взятия производной сложной функции (**Chain Rule**).

$$\begin{aligned} g(f(x_0 + \epsilon)) &= g(f(x_0) + f'_x(x_0) \cdot \epsilon) \\ &= g(f(x_0)) + g'_x(f(x_0)) \cdot f'_x(x_0) \cdot \epsilon \\ &= \left[g(f(x_0)) \right]'_x = g'_x(f(x_0)) \cdot f'_x(x_0) \end{aligned}$$

Автоматическое дифференцирование: **Dual Numbers**

Осталось **распространить** все базовые операции с вещественных чисел на **dual numbers**:

- Сложение, умножение, вычитание, деление
- Возведение в степень
- Тригонометрические функции
- ...

Автоматическое дифференцирование: **Dual Numbers**

Осталось **распространить** все базовые операции с вещественных чисел на **dual numbers**:

- Сложение, **умножение**, вычитание, деление
- Возведение в степень
- Тригонометрические функции
- ...

Автоматическое дифференцирование: **Dual Numbers**

Умножение:

$$(f(x_0) \cdot g(x_0))' =$$

Автоматическое дифференцирование: **Dual Numbers**

Умножение:

$$(f(x_0) \cdot g(x_0))' = f'(x_0) \cdot g(x_0) + f(x_0) \cdot g'(x_0)$$

Автоматическое дифференцирование: **Dual Numbers**

Умножение:

$$\left(f(x_0) \cdot g(x_0) \right)'_x = f'_x(x_0) \cdot g(x_0) + f(x_0) \cdot g'_x(x_0)$$
$$(a + b\varepsilon) \cdot (c + d\varepsilon) =$$

Автоматическое дифференцирование: **Dual Numbers**

Умножение:

$$\left(f(x_0) + \epsilon f'(x_0) \right) \cdot \left(g(x_0) + \epsilon g'(x_0) \right) = f(x_0) \cdot g(x_0) + f'(x_0) \cdot g(x_0) \cdot \epsilon + f(x_0) \cdot g'(x_0) \cdot \epsilon + \epsilon^2 f'(x_0) g'(x_0)$$

$$(a + b\epsilon) \cdot (c + d\epsilon) = a \cdot c + (b \cdot c + a \cdot d) \cdot \epsilon + \cancel{b \cdot d \cdot \epsilon^2}$$

Автоматическое дифференцирование: Dual Numbers

Умножение:

теперь это равенство неверно

$$\left(f(x_0) + \epsilon f'(x_0) \right) \cdot \left(g(x_0) + \epsilon g'(x_0) \right) \stackrel{?}{=} f'(x_0) \cdot g(x_0) + f(x_0) \cdot g'(x_0)$$

$$(a + b\epsilon) \cdot (c + d\epsilon) = a \cdot c + (b \cdot c + a \cdot d) \cdot \epsilon + \cancel{b \cdot d \cdot \epsilon^2}$$

Автоматическое дифференцирование: **Dual Numbers**

Умножение:

теперь это равенство неверно

$$\left(f(x_0) + \epsilon \cdot f'(x_0) \right) \cdot \left(g(x_0) + \epsilon \cdot g'(x_0) \right) \stackrel{?}{=} f'(x_0) \cdot g(x_0) + f(x_0) \cdot g'(x_0)$$

$$(a + b\epsilon) \cdot (c + d\epsilon) = \underbrace{a \cdot c}_{f(x_0) \cdot g(x_0)} + (b \cdot c + a \cdot d) \cdot \epsilon + \cancel{b \cdot d \cdot \epsilon^2}$$

$\epsilon^2 = 0$

Автоматическое дифференцирование: **Dual Numbers**

Умножение:

$$\begin{aligned} \left(f(x_0) + f'(x_0) \epsilon \right) \cdot \left(g(x_0) + g'(x_0) \epsilon \right) &= f(x_0) \cdot g(x_0) + f'(x_0) \cdot g(x_0) \epsilon + f(x_0) \cdot g'(x_0) \epsilon + f'(x_0) \cdot g'(x_0) \epsilon^2 \\ &= \underbrace{f(x_0) \cdot g(x_0)}_{f(x_0) \cdot g(x_0)} + (f'(x_0) \cdot g(x_0) + f(x_0) \cdot g'(x_0)) \epsilon + \cancel{f'(x_0) \cdot g'(x_0) \epsilon^2} \end{aligned}$$

Автоматическое дифференцирование: **Dual Numbers**

Умножение:

$$\begin{aligned} (f(x_0) + f'(x_0)\epsilon) \cdot (g(x_0) + g'(x_0)\epsilon) &= f'(x_0) \cdot g(x_0) + f(x_0) \cdot g'(x_0) \\ (a + b\epsilon) \cdot (c + d\epsilon) &= \underbrace{a \cdot c}_{f(x_0) \cdot g(x_0)} + (b \cdot c + a \cdot d) \cdot \epsilon + \cancel{b \cdot d \cdot \epsilon^2} \end{aligned}$$

Автоматическое дифференцирование: **Dual Numbers**

Умножение:

$$\begin{aligned} (f(x_0) + f'(x_0)\epsilon) \cdot (g(x_0) + g'(x_0)\epsilon) &= f'(x_0) \cdot g(x_0) + f(x_0) \cdot g'(x_0) \\ (a + b\epsilon) \cdot (c + d\epsilon) &= \underbrace{a \cdot c}_{f(x_0) \cdot g(x_0)} + (b \cdot c + a \cdot d) \cdot \epsilon + \cancel{b \cdot d \cdot \epsilon^2} \end{aligned}$$

Автоматическое дифференцирование: **Dual Numbers**

Умножение:

$$\begin{aligned} \left(f(x_0) + f'(x_0) \epsilon \right) \cdot \left(g(x_0) + g'(x_0) \epsilon \right) &= f(x_0) \cdot g(x_0) + f'(x_0) \cdot g(x_0) \epsilon + f(x_0) \cdot g'(x_0) \epsilon + \cancel{f'(x_0) \cdot g'(x_0) \epsilon^2} \\ &= \underbrace{f(x_0) \cdot g(x_0)}_{f(x_0) \cdot g(x_0)} + (f'(x_0) \cdot g(x_0) + f(x_0) \cdot g'(x_0)) \epsilon \end{aligned}$$

Автоматическое дифференцирование: **Dual Numbers**

Умножение:

$$\begin{aligned} \left(f(x_0) + f'(x_0) \epsilon \right) \cdot \left(g(x_0) + g'(x_0) \epsilon \right) &= f(x_0) \cdot g(x_0) + f'(x_0) \cdot g(x_0) \epsilon + f(x_0) \cdot g'(x_0) \epsilon + \cancel{f'(x_0) \cdot g'(x_0) \epsilon^2} \\ &= \underbrace{f(x_0) \cdot g(x_0)}_{f(x_0) \cdot g(x_0)} + (f'(x_0) \cdot g(x_0) + f(x_0) \cdot g'(x_0)) \epsilon \end{aligned}$$

Автоматическое дифференцирование: **Dual Numbers**

Умножение:

$$\begin{aligned} \left(f(x_0) \cdot g(x_0) \right)'_x &= f'_x(x_0) \cdot g(x_0) + f(x_0) \cdot g'_x(x_0) \\ (a + b\varepsilon) \cdot (c + d\varepsilon) &= \underbrace{a \cdot c}_{f(x_0) \cdot g(x_0)} + (b \cdot c + a \cdot d) \cdot \varepsilon + \cancel{b \cdot d \cdot \varepsilon^2} \\ &= 0 \end{aligned}$$

Автоматическое дифференцирование: **Dual Numbers**

Умножение:

$$\begin{aligned} \left(f(x_0) + f'(x_0) \epsilon \right) \cdot \left(g(x_0) + g'(x_0) \epsilon \right) &= \underbrace{f(x_0) \cdot g(x_0)}_{f(x_0) \cdot g(x_0)} + \underbrace{f'(x_0) \cdot g(x_0) + f(x_0) \cdot g'(x_0)}_{(b \cdot c + a \cdot d) \cdot \epsilon} + \cancel{f'(x_0) \cdot g'(x_0) \epsilon^2} \\ (a + b \epsilon) \cdot (c + d \epsilon) &= \underbrace{a \cdot c}_{f(x_0) \cdot g(x_0)} + (b \cdot c + a \cdot d) \cdot \epsilon + \cancel{b \cdot d \cdot \epsilon^2} \end{aligned}$$

Автоматическое дифференцирование: Dual Numbers

Умножение:

$$\begin{aligned} (f(x_0) + \epsilon f'(x_0)) \cdot (g(x_0) + \epsilon g'(x_0)) &= f(x_0) \cdot g(x_0) + f(x_0) \cdot \epsilon g'(x_0) + \epsilon f'(x_0) \cdot g(x_0) + \epsilon^2 f'(x_0) g'(x_0) \\ &= f(x_0) \cdot g(x_0) + (f'(x_0) \cdot g(x_0) + f(x_0) \cdot g'(x_0)) \cdot \epsilon + \epsilon^2 f'(x_0) g'(x_0) \end{aligned}$$

The diagram illustrates the multiplication of dual numbers. The first line shows the general form of the product of two dual numbers, with terms highlighted in yellow and red. The second line shows the expansion of the product, with terms grouped and simplified. The final term, $\epsilon^2 f'(x_0) g'(x_0)$, is crossed out with a red 'X' and labeled as zero, indicating that ϵ^2 is negligible.

Автоматическое дифференцирование: Dual Numbers

Умножение:

$$\begin{aligned} (f(x_0) + \epsilon f'(x_0)) \cdot (g(x_0) + \epsilon g'(x_0)) &= f(x_0) \cdot g(x_0) + f(x_0) \cdot \epsilon g'(x_0) + \epsilon f'(x_0) \cdot g(x_0) + \epsilon^2 f'(x_0) g'(x_0) \\ &= (a + b\epsilon) \cdot (c + d\epsilon) = \underbrace{a \cdot c}_{f(x_0) \cdot g(x_0)} + (b \cdot c + a \cdot d) \cdot \epsilon + \cancel{b \cdot d \cdot \epsilon^2} \end{aligned}$$

Автоматическое дифференцирование: **Dual Numbers**

Умножение:

$$\begin{aligned} \left(f(x_0) + f'(x_0) \epsilon \right) \cdot \left(g(x_0) + g'(x_0) \epsilon \right) &= \underbrace{f(x_0) \cdot g(x_0)}_{f(x_0) \cdot g(x_0)} + \underbrace{f'(x_0) \cdot g(x_0) + f(x_0) \cdot g'(x_0)}_{(b \cdot c + a \cdot d) \cdot \epsilon} + \cancel{f'(x_0) \cdot g'(x_0) \epsilon^2} \\ (a + b \epsilon) \cdot (c + d \epsilon) &= \underbrace{a \cdot c}_{f(x_0) \cdot g(x_0)} + (b \cdot c + a \cdot d) \cdot \epsilon + \cancel{b \cdot d \cdot \epsilon^2} \end{aligned}$$

Автоматическое дифференцирование: **Dual Numbers**


Тип Jet

```
211 struct Jet {  
285     // The scalar part.  
286     T a;  
288     // The infinitesimal part.  
289     Eigen::Matrix<T, N, 1> v;  
294 };
```

Автоматическое дифференцирование: **Dual Numbers**

Тип Jet

```
211 struct Jet {  
285     // The scalar part.  
286     T a;  
288     // The infinitesimal part.  
289     Eigen::Matrix<T, N, 1> v;  
294 };
```



Автоматическое дифференцирование: **Dual Numbers**

Тип Jet

```
211 struct Jet {  
285     // The scalar part.  
286     T a;  
288     // The infinitesimal part.  
289     Eigen::Matrix<T, N, 1> v;  
294 };
```

**для векторных
аргументов**

Автоматическое дифференцирование: **Dual Numbers**

Умножение: [github/ceres-solver/include/ceres/jet.h](https://github.com/ceres-solver/include/ceres/jet.h) `operator*`

```
347 // Binary *
348 template <typename T, int N>
349 inline Jet<T, N> operator*(const Jet<T, N>& f, const Jet<T, N>& g) {
350     return Jet<T, N>(f.a * g.a, f.a * g.v + f.v * g.a);
351 }
```


Автоматическое дифференцирование: Dual Numbers

Умножение: [github/ceres-solver/include/ceres/jet.h](https://github.com/ceres-solver/include/ceres/jet.h) operator*

```
347 // Binary *
348 template <typename T, int N>
349 inline Jet<T, N> operator*(const Jet<T, N>& f, const Jet<T, N>& g) {
350     return Jet<T, N>(f.a * g.a, f.a * g.v + f.v * g.a);
351 }
```

```
211 struct Jet {
285     // The scalar part.
286     T a;
288     // The infinitesimal part.
289     Eigen::Matrix<T, N, 1> v;
294 };
```

Автоматическое дифференцирование: Dual Numbers

Умножение: [github/ceres-solver/include/ceres/jet.h](https://github.com/ceres-solver/include/ceres/jet.h) operator*

```
347 // Binary *
348 template <typename T, int N>
349 inline Jet<T, N> operator*(const Jet<T, N>& f, const Jet<T, N>& g) {
350     return Jet<T, N>(f.a * g.a, f.a * g.v + f.v * g.a);
351 }
```

211 struct Jet {
285 // The scalar part.
286 T a;
288 // The infinitesimal part.
289 Eigen::Matrix<T, N, 1> v;
294 };

real

ϵ

Автоматическое дифференцирование: Dual Numbers

Умножение: [github/ceres-solver/include/ceres/jet.h](https://github.com/ceres-solver/include/ceres/jet.h) operator*

```
347 // Binary *
348 template <typename T, int N>
349 inline Jet<T, N> operator*(const Jet<T, N>& f, const Jet<T, N>& g) {
350     return Jet<T, N>(f.a * g.a, f.a * g.v + f.v * g.a);
351 }
```

```
211 struct Jet {
285     // The scalar part.
286     T a;
288     // The infinitesimal part.
289     Eigen::Matrix<T, N, 1> v;
294 };
```

$$(a + b\varepsilon) \cdot (c + d\varepsilon) = a \cdot c + (b \cdot c + a \cdot d) \cdot \varepsilon + \cancel{b \cdot d \cdot \varepsilon^2}$$

$\varepsilon^2 = 0$

Автоматическое дифференцирование: Dual Numbers

Умножение: [github/ceres-solver/include/ceres/jet.h](https://github.com/ceres-solver/include/ceres/jet.h) operator*

```
347 // Binary *
348 template <typename T, int N>
349 inline Jet<T, N> operator*(const Jet<T, N>& f, const Jet<T, N>& g) {
350     return Jet<T, N>(f.a * g.a, f.a * g.v + f.v * g.a);
351 }
```

```
211 struct Jet {
285     // The scalar part.
286     T a;
288     // The infinitesimal part.
289     Eigen::Matrix<T, N, 1> v;
294 };
```

$$\overbrace{(a+b\varepsilon)}^f \cdot \overbrace{(c+d\varepsilon)}^g = a \cdot c + (b \cdot c + a \cdot d) \cdot \varepsilon + \cancel{b \cdot d \cdot \varepsilon^2}$$

$\varepsilon^2 = 0$

Автоматическое дифференцирование: Dual Numbers

Умножение: [github/ceres-solver/include/ceres/jet.h](https://github.com/ceres-solver/include/ceres/jet.h) operator*

```
347 // Binary *
348 template <typename T, int N>
349 inline Jet<T, N> operator*(const Jet<T, N>& f, const Jet<T, N>& g) {
350     return Jet<T, N>(f.a * g.a, f.a * g.v + f.v * g.a);
351 }
```

```
211 struct Jet {
285     // The scalar part.
286     T a;
288     // The infinitesimal part.
289     Eigen::Matrix<T, N, 1> v;
294     };
```

$$\overbrace{(a + b\varepsilon)}^f \cdot \overbrace{(c + d\varepsilon)}^g = a \cdot c + (b \cdot c + a \cdot d) \cdot \varepsilon + \cancel{b \cdot d \cdot \varepsilon^2}$$

$\varepsilon^2 = 0$

Автоматическое дифференцирование: Dual Numbers

Умножение: [github/ceres-solver/include/ceres/jet.h](https://github.com/ceres-solver/include/ceres/jet.h) `operator*`

```
347 // Binary *
348 template <typename T, int N>
349 inline Jet<T, N> operator*(const Jet<T, N>& f, const Jet<T, N>& g) {
350     return Jet<T, N>(f.a * g.a, f.a * g.v + f.v * g.a);
351 }
```

```
211 struct Jet {
285     // The scalar part.
286     T a;
288     // The infinitesimal part.
289     Eigen::Matrix<T, N, 1> v;
294 }
```

$$\overbrace{(a + b\varepsilon)}^f \cdot \overbrace{(c + d\varepsilon)}^g = a \cdot c + (b \cdot c + a \cdot d) \cdot \varepsilon + \cancel{b \cdot d \cdot \varepsilon^2}$$

$\varepsilon^2 = 0$

Автоматическое дифференцирование: Dual Numbers

Умножение: [github/ceres-solver/include/ceres/jet.h](https://github.com/ceres-solver/include/ceres/jet.h) operator*

```
347 // Binary *
348 template <typename T, int N>
349 inline Jet<T, N> operator*(const Jet<T, N>& f, const Jet<T, N>& g) {
350     return Jet<T, N>(f.a * g.a, f.a * g.v + f.v * g.a);
351 }
```

```
211 struct Jet {
285     // The scalar part.
286     T a;
288     // The infinitesimal part.
289     Eigen::Matrix<T, N, 1> v;
294 };
```

$$\overbrace{(a + b\varepsilon)}^f \cdot \overbrace{(c + d\varepsilon)}^g = \overbrace{a \cdot c} + (b \cdot c + a \cdot d) \cdot \varepsilon + \cancel{b \cdot d \cdot \varepsilon^2}$$

$\varepsilon^2 = 0$

Автоматическое дифференцирование: Dual Numbers

Умножение: [github/ceres-solver/include/ceres/jet.h](https://github.com/ceres-solver/include/ceres/jet.h) operator*

```

347 // Binary *
348 template <typename T, int N>
349 inline Jet<T, N> operator*(const Jet<T, N>& f, const Jet<T, N>& g) {
350     return Jet<T, N>(f.a * g.a, f.a * g.v + f.v * g.a);
351 }

```

```
211 struct Jet {
285     // The scalar part.
286     T a;
288     // The infinitesimal part.
289     Eigen::Matrix<T, N, 1> v;
294 };
```

$$\overbrace{(a + b\varepsilon)}^f \cdot \overbrace{(c + d\varepsilon)}^g = \overbrace{a \cdot c}^{||} + (b \cdot c + a \cdot d) \cdot \varepsilon + \cancel{b \cdot d \cdot \varepsilon^2}$$

Автоматическое дифференцирование: Dual Numbers

Умножение: [github/ceres-solver/include/ceres/jet.h](https://github.com/ceres-solver/include/ceres/jet.h) operator*

```

347 // Binary *
348 template <typename T, int N>
349 inline Jet<T, N> operator*(const Jet<T, N>& f, const Jet<T, N>& g) {
350     return Jet<T, N>(f.a * g.a, f.a * g.v + f.v * g.a);
351 }

```

```
211 struct Jet {
285     // The scalar part.
286     T a;
288     // The infinitesimal part.
289     Eigen::Matrix<T, N, 1> v;
294 };
```

$$\overbrace{(a+b\varepsilon)}^f \cdot \overbrace{(c+d\varepsilon)}^g = \overbrace{a \cdot c}^{\parallel} + \overbrace{(b \cdot c + a \cdot d)}^{\parallel} \cdot \varepsilon + \cancel{b \cdot d \cdot \varepsilon^2} \\ \varepsilon^2 = 0$$

Автоматическое дифференцирование: Dual Numbers

Умножение: [github/ceres-solver/include/ceres/jet.h](https://github.com/ceres-solver/include/ceres/jet.h) operator*

```

347 // Binary *
348 template <typename T, int N>
349 inline Jet<T, N> operator*(const Jet<T, N>& f, const Jet<T, N>& g) {
350     return Jet<T, N>(f.a * g.a, f.a * g.v + f.v * g.a);
351 }

```

```
211 struct Jet {
285     // The scalar part.
286     T a;
288     // The infinitesimal part.
289     Eigen::Matrix<T, N, 1> v;
294 };
```

$$\overbrace{(a+b\varepsilon)}^f \cdot \overbrace{(c+d\varepsilon)}^g = \overbrace{a \cdot c}^{=} + \overbrace{(b \cdot c + a \cdot d)}^{=} \cdot \varepsilon + \cancel{b \cdot d \cdot \varepsilon^2} \\ \varepsilon^2 = 0$$

Автоматическое дифференцирование: Dual Numbers

Умножение: [github/ceres-solver/include/ceres/jet.h](https://github.com/ceres-solver/include/ceres/jet.h) operator*

```

347 // Binary *
348 template <typename T, int N>
349 inline Jet<T, N> operator*(const Jet<T, N>& f, const Jet<T, N>& g) {
350     return Jet<T, N>(f.a * g.a, f.a * g.v + f.v * g.a);
351 }

```

```
211 struct Jet {
285     // The scalar part.
286     T a;
288     // The infinitesimal part.
289     Eigen::Matrix<T, N, 1> v;
294 };
```

$$\overbrace{(a+b\varepsilon)}^f \cdot \overbrace{(c+d\varepsilon)}^g = \overbrace{a \cdot c} + \overbrace{(b \cdot c + a \cdot d)} \cdot \varepsilon + \cancel{b \cdot d \cdot \varepsilon^2}$$

Автоматическое дифференцирование: Dual Numbers

Умножение: [github/ceres-solver/include/ceres/jet.h](https://github.com/ceres-solver/include/ceres/jet.h) operator*

```

347 // Binary *
348 template <typename T, int N>
349 inline Jet<T, N> operator*(const Jet<T, N>& f, const Jet<T, N>& g) {
350     return Jet<T, N>(f.a * g.a, f.a * g.v + f.v * g.a);
351 }

```

```
211 struct Jet {
285     // The scalar part.
286     T a;
288     // The infinitesimal part.
289     Eigen::Matrix<T, N, 1> v;
294 };
```

$$\overbrace{(a+b\varepsilon)}^f \cdot \overbrace{(c+d\varepsilon)}^g = \overbrace{a \cdot c} + \overbrace{(b \cdot c + a \cdot d)} \cdot \varepsilon + \cancel{b \cdot d \cdot \varepsilon^2}$$

Автоматическое дифференцирование: Dual Numbers

Синус: [github/ceres-solver/include/ceres/jet.h](https://github.com/ceres-solver/include/ceres/jet.h) `sin()`

```
// sin(a + h) ~= sin(a) + cos(a) h
template <typename T, int N>
inline Jet<T, N> sin(const Jet<T, N>& f) {
    return Jet<T, N>(sin(f.a), cos(f.a) * f.v);
}
```

```
211 struct Jet {
285     // The scalar part.
286     T a;
288     // The infinitesimal part.
289     Eigen::Matrix<T, N, 1> v;
};
```

Примеры задач

1) Минимизировать $\frac{1}{2}(10 - x)^2$

Какие блоки параметров? Какая функция невязки (residual)?

Примеры задач

- 1) Минимизировать $\frac{1}{2}(10 - x)^2$
- 2) Есть фиксированная прямая и фиксированный параболоид. Найти точку пересечения.

Какие блоки параметров? Какая функция невязки (residual)?

Примеры задач

- 1) Минимизировать $\frac{1}{2}(10 - x)^2$
- 2) Есть фиксированная прямая и фиксированный параболоид. Найти точку пересечения.
- 3) Есть сколько-то шумных замеров (с выбросами), нужно зафитить прямой.

Какие блоки параметров? Какая функция невязки (residual)? Какая функция потерь (Loss function)?

Примеры задач

- 1) Минимизировать $\frac{1}{2}(10 - x)^2$
- 2) Есть фиксированная прямая и фиксированный параболоид. Найти точку пересечения.
- 3) Есть сколько-то шумных замеров (с выбросами), нужно зафитить прямой.
- 4) Bundle Adjustment.

Какие блоки параметров? Какая функция невязки (residual)? Какая функция потерь (Loss function)?

Примеры задач

- 1) Минимизировать $\frac{1}{2}(10 - x)^2$
- 2) Есть фиксированная прямая и фиксированный параболоид. Найти точку пересечения.
- 3) Есть сколько-то шумных замеров (с выбросами), нужно зафитить прямой.
- 4) Bundle Adjustment.

Какие блоки параметров? Какая функция невязки (residual)? Какая функция потерь (Loss function)?

Как фильтровать шумные ключевые точки из разреженного облака точек для более надежного последующего добавления очередной камеры?

Bundle Adjustment. Ссылки

Две ключевые книги-библии:

- [Multiple View Geometry in Computer Vision, Richard Hartley & Andrew Zisserman \(здесь особенно много деталей, в т.ч. про L-M\)](#)
- [Computer Vision: Algorithms and Applications, Richard Szeliski](#)

Статьи про детали BA:

- [Bundle adjustment—a modern synthesis, Triggs et al.](#)
- [Bundle Adjustment in the Large, Agarwal et. al.](#)

Лекции:

- [youtube/Behnam Asadi: про методы оптимизации и BA](#)
- [youtube/Cyrill Stachniss: BA 1, BA 2](#)
- [coursera/Robotics: Perception \(BA\)](#)

Hierarchical SFM

- 1) Что делать если в датасете много кадров (например, 10K)
 - а) даже самый мощный компьютер будет обрабатывать долго

Hierarchical SFM

- 1) Что делать если в датасете много кадров (например, 10K)
 - a) даже самый мощный компьютер будет обрабатывать долго
 - b) а если кадров еще в 10 раз больше?

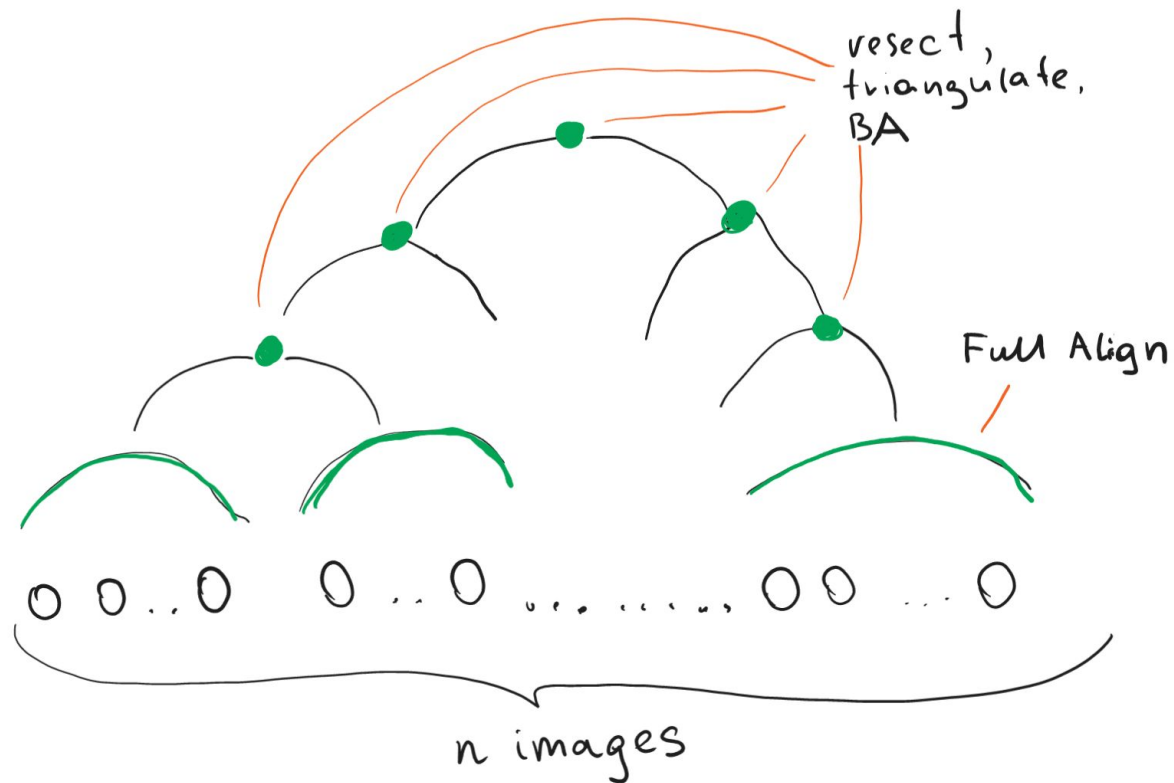
Hierarchical SFM

- 1) Что делать если в датасете много кадров (например, 10K)
 - а) даже самый мощный компьютер будет обрабатывать долго
 - б) а если кадров еще в 10 раз больше?
- 2) Инкрементальный подход уязвим к ошибкам выравнивания на раннем этапе

Hierarchical SFM

- 1) Что делать если в датасете много кадров (например, 10K)
 - a) даже самый мощный компьютер будет обрабатывать долго
 - b) а если кадров еще в 10 раз больше?
- 2) Инкрементальный подход уязвим к ошибкам выравнивания на раннем этапе
- 3) **Hierarchical SFM**: используем сразу много компьютеров (узлов) соединенных по локальной сети
 - a) разобьем датасет на небольшие группы (например по 100 кадров)
 - b) подзадача для каждого узла: выровнять группу из 100 кадров, либо взять две выровненные группы, объединить облака точек и выполнить BA

Hierarchical SFM



Геопривязка

- 1) Что если мы хотим по построенной геометрии делать измерения?
Например, построение планов местности, кадастровый учет, измерения объемов руды на карьерах
- 2) Можно с помощью точного GPS датчика измерить несколько характерных хорошо заметных точек на поверхности земли (Контрольные точки, control points), и добавить их в ВА как дополнительные измерения с большим весом
- 3) Как оценить качество выравнивания?

Геопривязка

- 1) Что если мы хотим по построенной геометрии делать измерения?
Например, построение планов местности, кадастровый учет, измерения объемов руды на карьерах
- 2) Можно с помощью точного GPS датчика измерить несколько характерных хорошо заметных точек на поверхности земли (Контрольные точки, control points), и добавить их в ВА как дополнительные измерения с большим весом
- 3) Как оценить качество выравнивания? Можно половину точек не добавлять как наблюдения в ВА, а после выравнивания оценить точность выравнивания через разницу посчитанной координаты и измеренной вручную

Realtime kinematic (RTK)

- 1) Контрольные точки ставить трудоемко. Как получить геопривязку и хорошую точность но с меньшими трудозатратами?

Realtime kinematic (RTK)

- 1) Контрольные точки ставить трудоемко. Как получить геопривязку и хорошую точность но с меньшими трудозатратами?
- 2) Можно использовать точный GPS для центров фотографирования

Realtime kinematic (RTK)

- 1) Контрольные точки ставить трудоемко. Как получить геопривязку и хорошую точность но с меньшими трудозатратами?
- 2) Можно использовать точный GPS для центров фотографирования
- 3) Обычный GPS имеет большую ошибку (~10 метров)

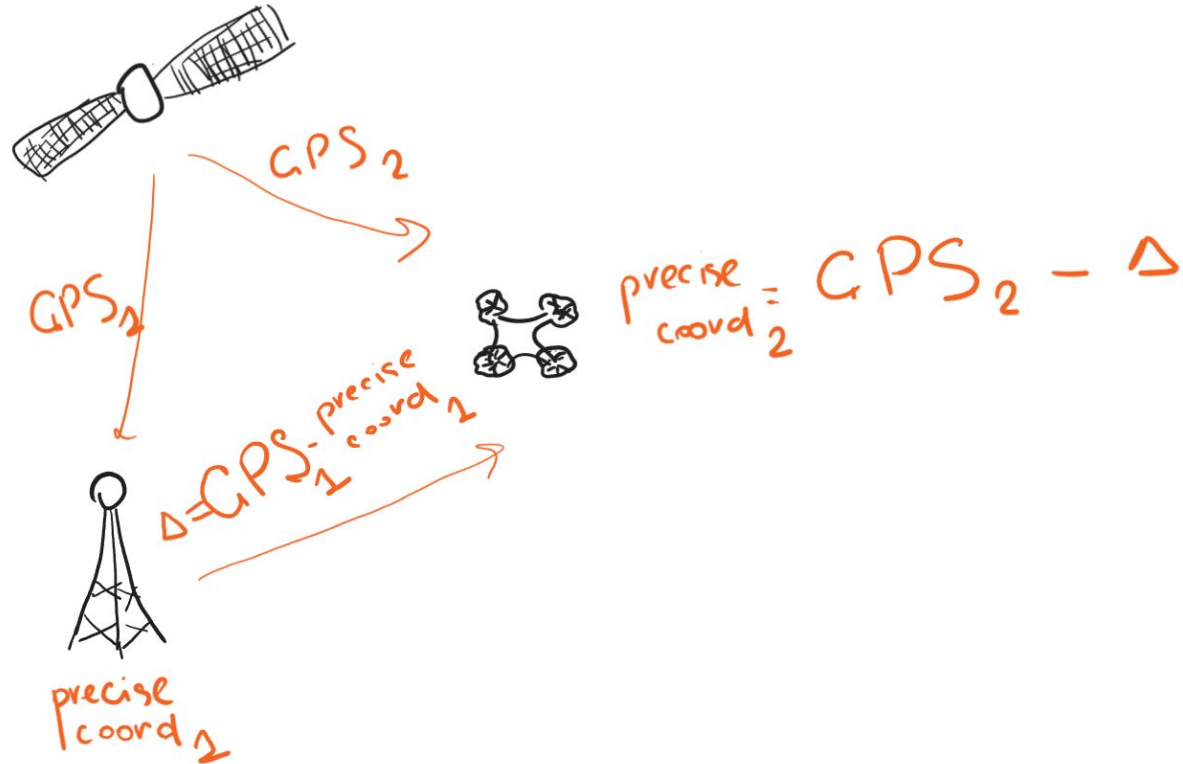
Realtime kinematic (RTK)

- 1) Контрольные точки ставить трудоемко. Как получить геопривязку и хорошую точность но с меньшими трудозатратами?
- 2) Можно использовать точный GPS для центров фотографирования
- 3) Обычный GPS имеет большую ошибку (~10 метров)
- 4) **RTK GPS**: использует базовую станцию на земле (мобильную либо стационарную с подключением по сотовой сети, NTRIP)

Realtime kinematic (RTK)

- 1) Контрольные точки ставить трудоемко. Как получить геопривязку и хорошую точность но с меньшими трудозатратами?
- 2) Можно использовать точный GPS для центров фотографирования
- 3) Обычный GPS имеет большую ошибку (~10 метров)
- 4) **RTK GPS**: использует базовую станцию на земле (мобильную либо стационарную с подключением по сотовой сети, NTRIP)
- 5) Базовая станция знает свою точную координату
- 6) В каждый момент времени измеряет координату обычным GPS, и разницу отправляет на борт дрону
- 7) У дрона ошибка почти совпадает с ошибкой у базовой станции
- 8) Дрон вычитает из своего шумного измерения ошибку и получает точную координату, которую можно записать в EXIF мета данные фотографии

Realtime kinematic (RTK)



Вопросы?

