Лекция 2: 2-D цифровая фильтрация и операторы условного перехода в языке Си

Д. А. Караваев

Санкт-Петербургский государственный университет телекоммуникаций им. проф. М. А. Бонч-Бруевича

Факультет РТС, Кафедра РОС

Факультатив «Программирование в ЦОС»

Осень 2019

21.10.2019 Санкт-Петербург

Определение:
Двумерный сигнал $x[n, m]$ - последовательность пронумерованная двумя индексами (n, m). Значения $x[n, m]$ и (n, m) могут иметь смысл интесивностии и пространственных индексов соответственно, тогда $x[n, m]$ есть *изображение*.

КИХ-фильтрация:
Осуществляется через *двумерную (пространственную) свёртку*:

$$y[n, m] = \sum_{i=0}^{T_h-1} \sum_{j=0}^{T_w-1} h[i, j]x[n - i, m - j], \tag{1}$$

где $x[n, m]$ - входной сигнал (N_h, N_w), $h[n, m]$ - ИХ размера (T_h, T_w) и $y[n, m]$ - выходной сигнал (N_h, N_w).

Пример результата 2D КИХ-фильтрации

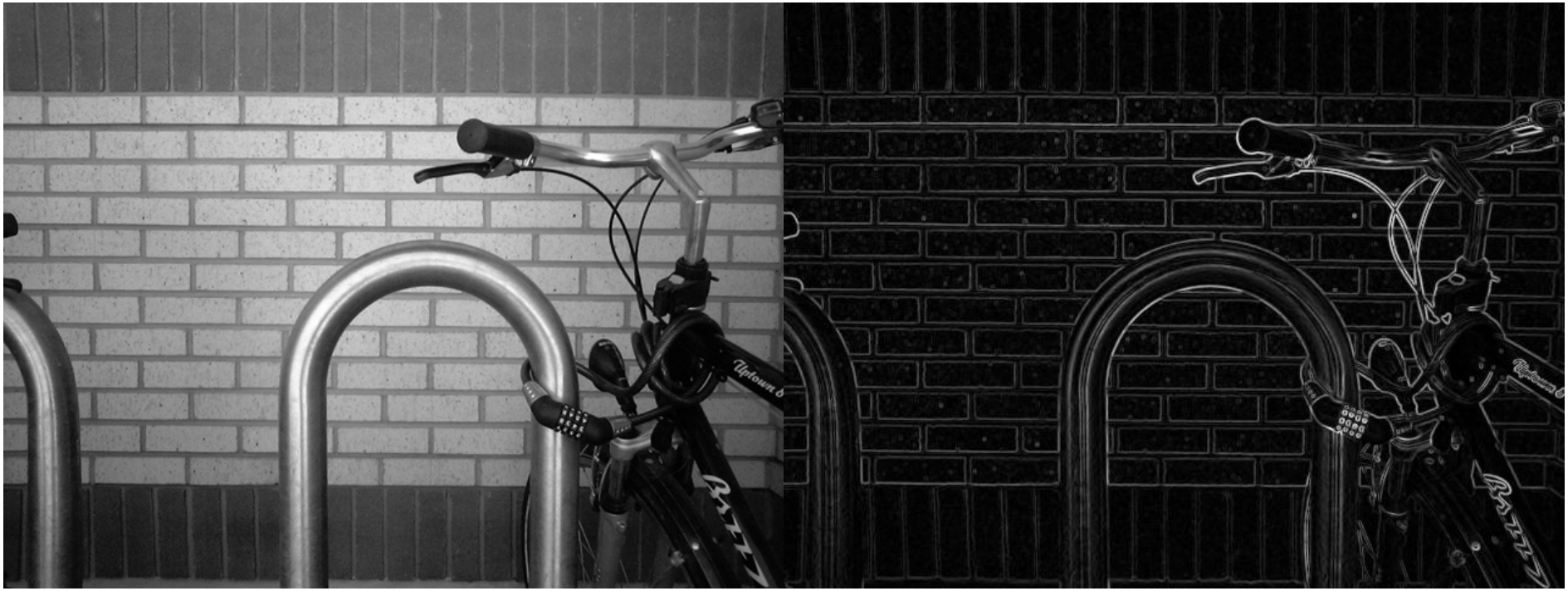
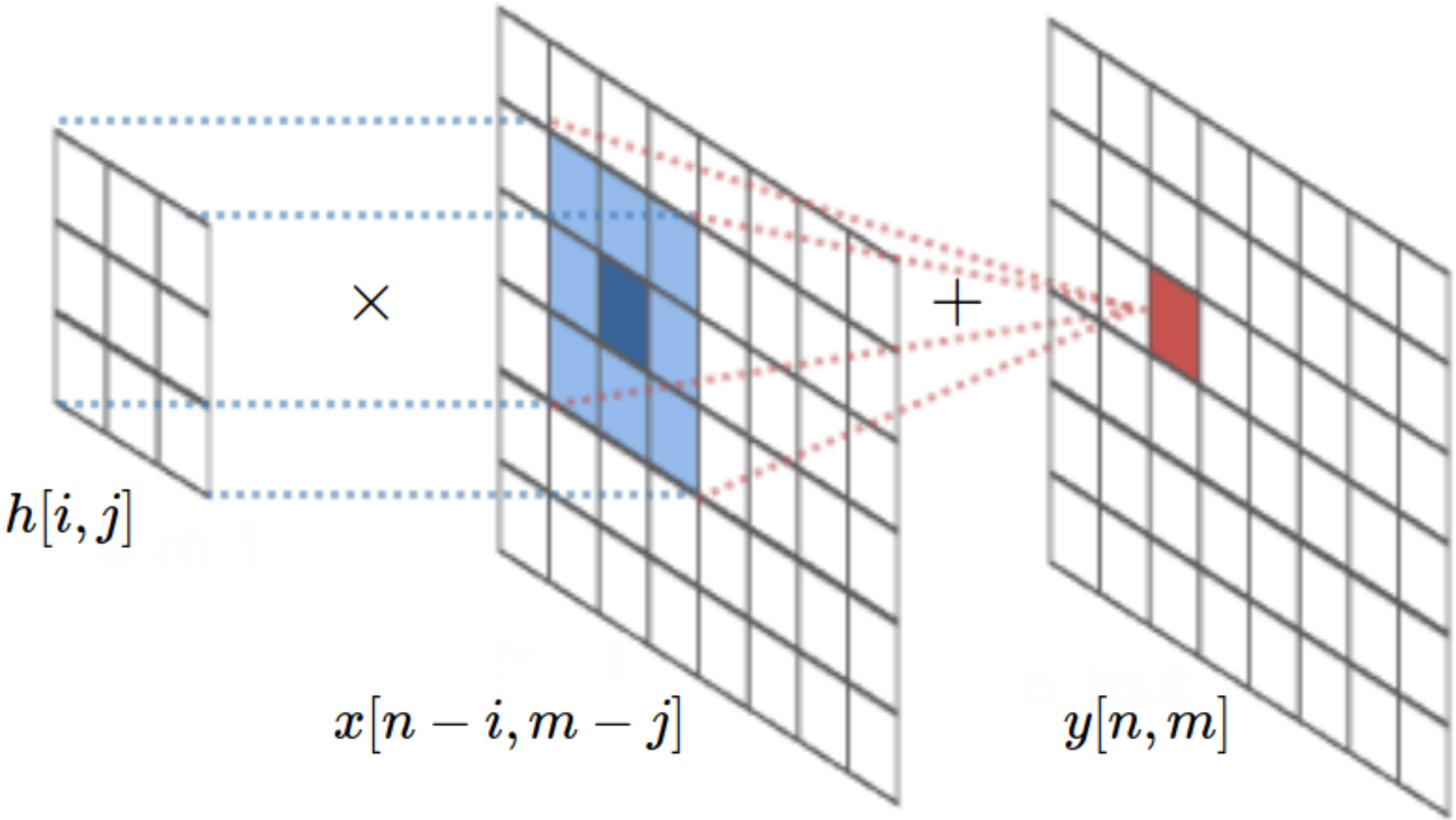


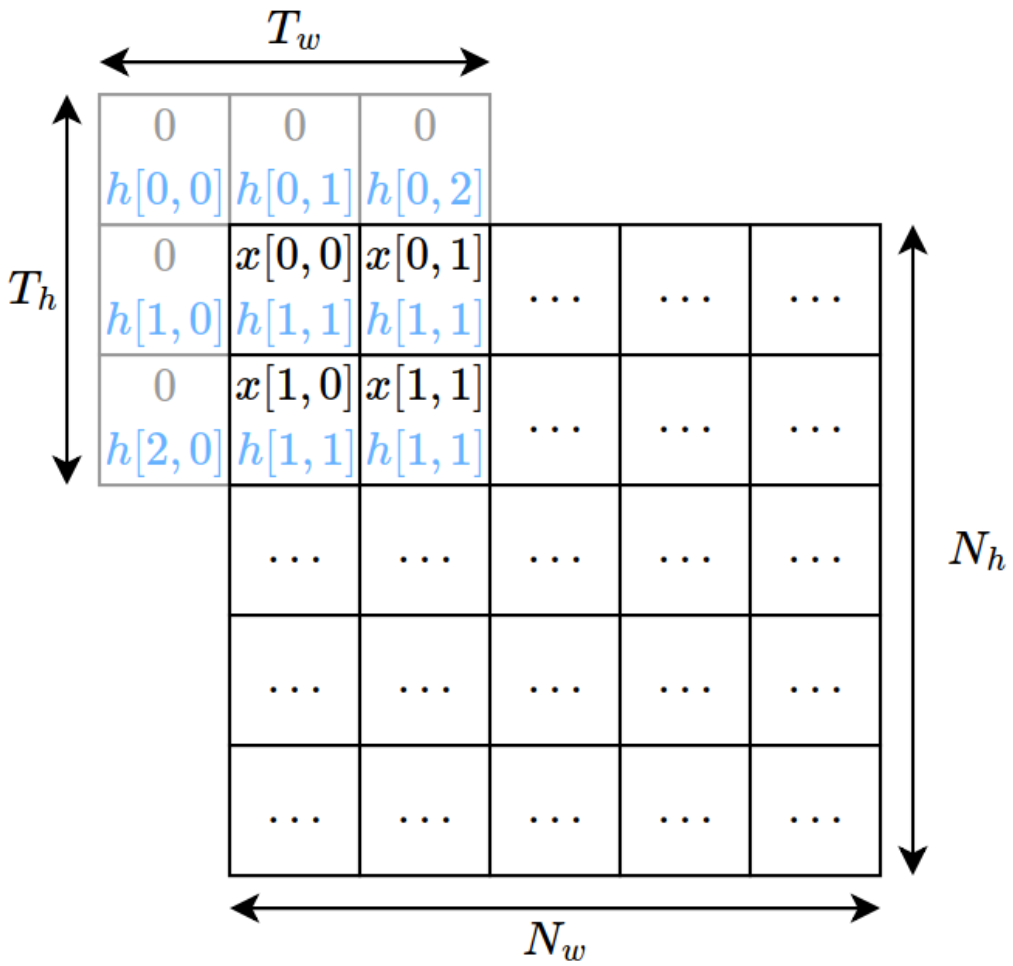
Рис.: Результат применения оператора Собеля (https://en.wikipedia.org/wiki/Sobel_operator).

Схема 2D свёртки



Граничный случай (дополнение нулями)

Замечание: В этот раз мы прибегнем к *неявному* дополнению нулями, проверяя условие выхода за границы сигнала. Для этого нужна конструкция *условного перехода* (a.k.a `if-else`).



Псевдокод алгоритма 2D свёртки

Вход : $x[n, m]$ - размера (N_h, N_w), $h[n, m]$ - размера (T_h, T_w).
Выход: $y[n, m]$ - размера (N_h, N_w).

for $n \leftarrow 0$ **to** $N_h - 1$ **do**
  for $m \leftarrow 0$ **to** $N_w - 1$ **do**
    $y[n, m] \leftarrow 0$
    for $i \leftarrow 0$ **to** $T_h - 1$ **do**
      for $j \leftarrow 0$ **to** $T_w - 1$ **do**
        if $(n - i) \geq 0 \wedge (m - j) \geq 0$ **then**
          $y[n, m] \leftarrow y[n, m] + h[i, j]x[n - i, m - j]$
        end
      end
    end
  end
end

Замечание: Как в языке Си реализовать логические выражения и условный переход?

Логические выражения в Си

```
/* Некоторые переменные: */
float x = 1.0;
float y = -54.12;
float z = -4.12;
/* Логические выражения - выражения, значения которых true/false.*/
/* Состоят из комбинации операций сравнения: (==, !=, <=, >=, <, >) и */
/* логических операций (и, или и не): */
int predicat0 = (x < y) && (x > 0); /* (x < y) и (x > 0).*/
int predicat1 = ((x + 10) < y) || (y > 100); /* (x < y) или (x > 0).*/
int not_predicat0 = !predicat0; /* Значение противоположеное predicat0.*/
/* В языке Си есть тип для хранения результата лог. выражения: bool */
bool truth = (x <= 0) || (x > 0);
/* У операций (и, или и не) есть свой порядок действий, */
/* как у *, + и - соответственно: */
bool example = ((x > 3.14) && (y == 2 * 3.14)) || !(z >= -5);
```

Оператор условного перехода в Си

```c
/* Всё стандартным образом: */
if (/* какое-то логическое выражение */)
{
    /* если выражение истинно исполняется код тут. */
}
else
{
    /* если выражение ложно исполняется код тут. */
}
/* Можно делать так: */
if (/* какое-то логическое выражение */)
{
    /* если выражение истинно исполняется код тут. */
}
/* Без кода на противоположное условие.*/
```

Множественный условный переход в Си

```
/* Оператор множественного условного перехода удобен для
 * проверки нескольких выражений: */
if (/* какое-то логическое выражение_0*/)
{
    /* если выражение_0 истинно исполняется код тут. */
}
else if (/* какое-то логическое выражение 1*/)
{
    /* если выражение_1 ложно исполняется код тут. */
}
/* ... */
else
{
    /* если никакое выражние_(0, 1 ...) неистинно исполняется код тут. */
}
```

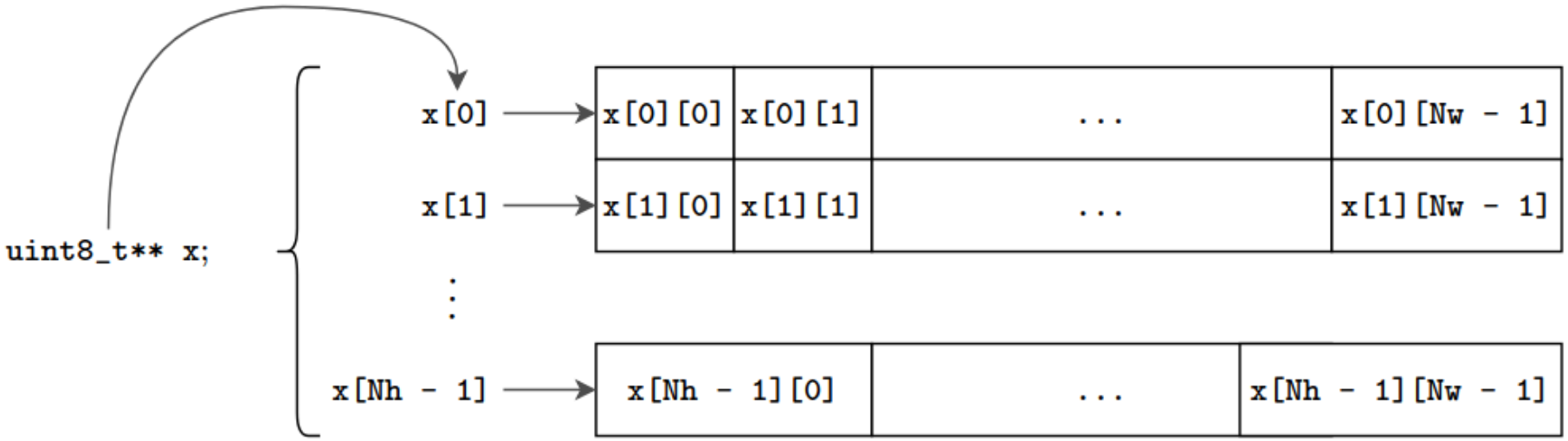
Пример условного перехода в Си

```
float x = 4.2;
float y = -4.3;
float z = 0.0;
if (x >= y) /* Проверяем выражение: */
{
    z = x + y;
}
else /* иначе: */
{
    z = x * y;
} /* или так: */
if ((x > 0) && (y < 0))
{
    z = x / y; /* Если выражение неверно, то z останется равным 0. */
}
```

Представление изображения в компьютере (Си)

```c
/* В Си изображение (ЧБ) можно интерпретировать как двумерные массивы
 * типа uint8_t (интенсивность белого): */
typedef unsigned char uint8_t; /* [0, 255]. */
/* !Внимание велик риск переполнения! */
uint8_t x[100][100]; /* Массив 100 на 100. */
x[10][25] = 24; /* Элемент (отсчёт) в 11 строке и в 26 столбце == 24. */
/* Будем использовать указатели по причине нефиксированного размера: */
uint8_t** y = x; /* Инициализируем указатель на двумерный массив. */
for (size_t n = 0; n < 100; n++) /* По строкам: */
{
    for (size_t m = 0; m < 100; m++) /* По столбцам: */
    {
        y[n][m] = 34; /* (n-ый, m-ый) элемент (пиксель). */
    }
}
```

Иллюстрация двойного указателя



Прототип функции

```
/*!
 * \param[in] x - 2-мерный входной сигнал.
 * \param[out] y - 2-мерный выходной сигнал.
 * \param[in] Nh - высота входного сигнала.
 * \param[in] Nw - ширина входного сигнала.
 * \param[in] h - ИХ.
 * \param[in] Th - высота ИХ.
 * \param[in] Th - ширина ИХ.
 * \return 0 - успех, -1 - не реализована.
 */
int DSP_convolve_2D(const uint8_t** x, uint8_t** y, size_t Nh, size_t Nw,
                    const int8_t**  h, size_t Th,  size_t Tw)
{
    return -1;
}
```

Результат 2D свёртки

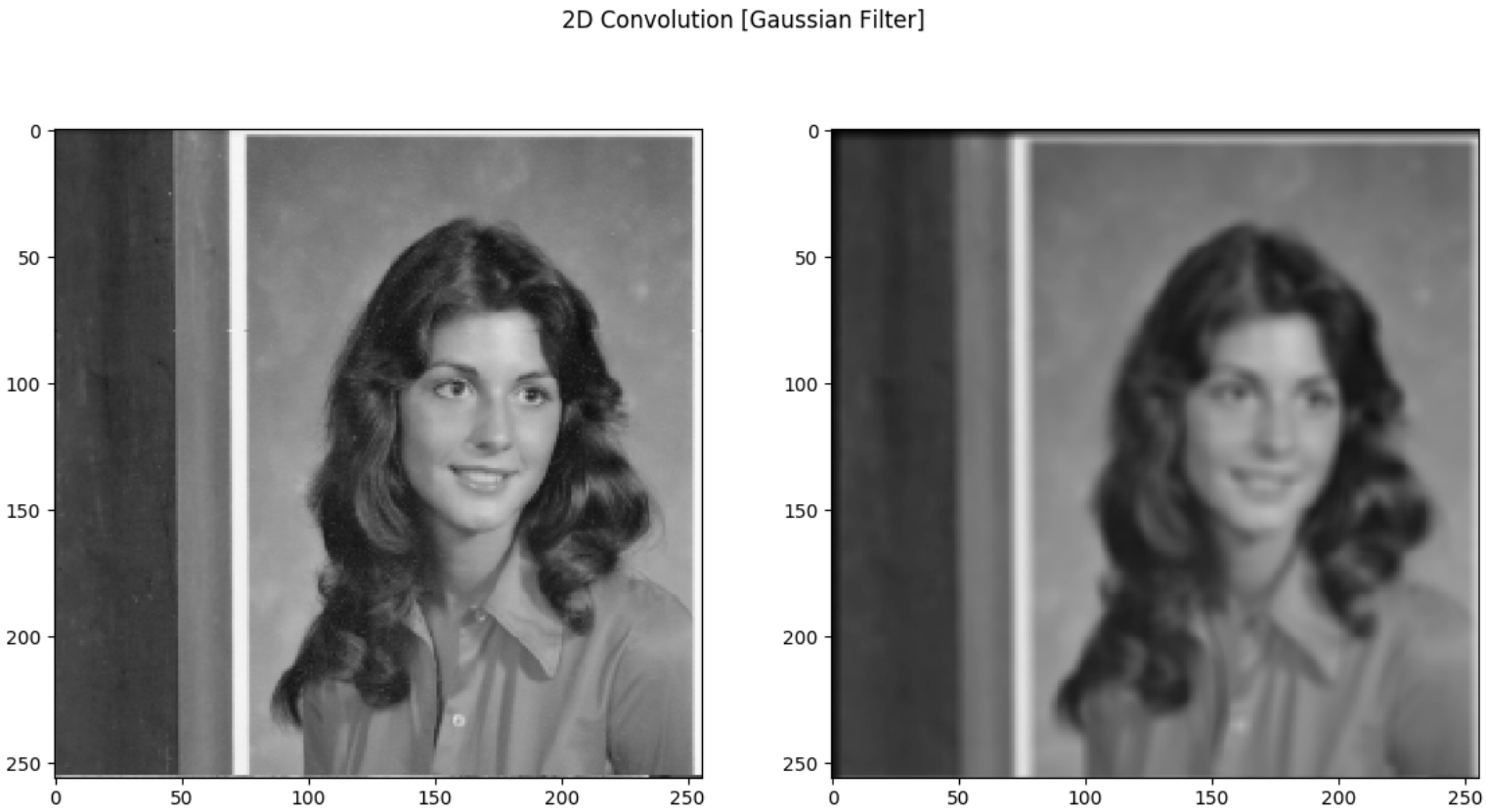


Рис.: Результат применения сглаживающего фильтра Гаусса (https://en.wikipedia.org/wiki/Gaussian_blur).
Команда для отображения: `python3 ../scripts/plot_2D.py`

Спасибо за внимание!