Цифровая обработка сигналов

Лабораторная работа № 8

Сопоставление изображений

Содержание

Организация:	Самарский университет
Подразделение:	Кафедра геоинформатики и информационной безопасности
Версия:	2023.12.11-beta

1 Теоретические сведения

1.1 Многомерные варианты преобразования Фурье

Изображение можно рассматривать как функцию зависимости яркости от координат пространства (напомним, что до сих пор аргументом было время, а не пространство), поэтому обозначим сигнал как f, а x будем использовать для обозначения координат пространства.

Непрерывные координаты пространства зададим вектором $\mathbf{x} = (x_0, ..., x_{M-1})$, где M — размерность рассматриваемого пространства. Дискретные координаты зададим вектором $\mathbf{n} = (n_0, ..., n_{M-1})$, а размеры дискретного изображения обозначим как $\mathbf{N} = (N_0, ..., N_{M-1})$.

Для одноканального (например, в оттенках серого) изображения $M = 2$ и величины N_0 и N_1 задают размеры изображения (высоту и ширину). Для многоканальных изображений $M = 3$ и величина N_2 задаёт количество каналов. Для цветного RGB-изображения используется три канала ($N_2 = 3$); для гиперспектральных изображений, например, получаемых спутником Ресурс-П, количество каналов может достигать 216.

Многомерное преобразование Фурье:

$$F(\mathbf{\Omega}) = \int\limits_{\mathbb{R}^M} f(\mathbf{x}) e^{-j(\mathbf{\Omega},\ \mathbf{x})} d\mathbf{x},$$

$$f(\mathbf{x}) = \frac{1}{(2\pi)^M} \int\limits_{\mathbb{R}^M} F(\mathbf{\Omega}) e^{j(\mathbf{\Omega},\ \mathbf{x})} d\mathbf{\Omega},$$

где $(\mathbf{\Omega},\ \mathbf{x})$ – скалярное произведение векторов $\mathbf{\Omega}$ и $\mathbf{x}$.

Многомерное дискретное преобразование Фурье:

$$F[\mathbf{m}] = \sum_{\mathbf{n}=\mathbf{0}}^{\mathbf{N}-1} f[\mathbf{n}] e^{-j2\pi(\mathbf{m},\ \frac{\mathbf{n}}{\mathbf{N}})},$$

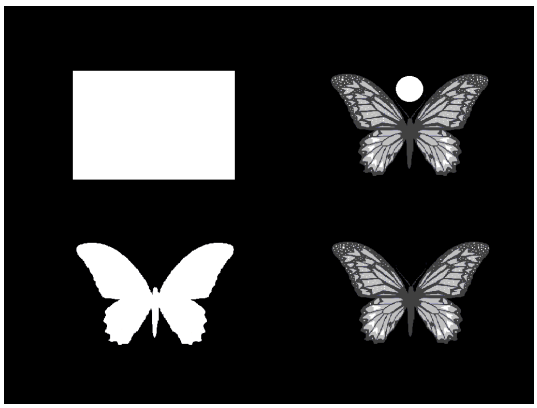
$$f[\mathbf{n}] = \frac{1}{\prod_{i=0}^{M-1} N_i} \sum_{\mathbf{m}=\mathbf{0}}^{\mathbf{N}-1} F[\mathbf{m}] e^{j2\pi(\mathbf{n},\ \frac{\mathbf{m}}{\mathbf{N}})},$$

где $\sum\limits_{\mathbf{n}=\mathbf{0}}^{\mathbf{N}-1}$ обозначает перебор по всем координатам $\sum\limits_{n_1=0}^{N_1-1} ... \sum\limits_{n_{M-1}=0}^{N_{M-1}}$, а $\frac{\mathbf{n}}{\mathbf{N}}$ обозначает покомпонентное деление векторов $\left(\frac{n_0}{N_0}, ..., \frac{n_{M-1}}{N_{M-1}}\right)$.

Обратите внимание, что при $M = 1$ данные формулы превращаются в уже известные вам варианты преобразований.

1.2 Корреляционный метод поиска шаблона

Рассмотрим одноканальное ($M = 2$) изображение $f[\mathbf{n}] = f[n_0, n_1]$ размера $\mathbf{N}$:



Пусть на нём необходимо найти все вхождения шаблона $h[\mathbf{n}]$ размера $\mathbf{M}$:

В простейшем случае можно оценить взаимнокорреляционную функцию:

$$R_{hf}[\mathbf{n}] = \sum_{\mathbf{m}=\mathbf{0}}^{\mathbf{M}-1} h[\mathbf{m}] f[\mathbf{n} + \mathbf{m}].$$

С точностью до знака $\mathbf{m}$ вышеприведённая формула представляет собой свёртку $R_{hf}[\mathbf{n}] = h[-\mathbf{n}] * f[\mathbf{n}]$. Чтобы работать с сигналами без отрицательных индексов, то есть начинающихся с нулевого индекса, необходимо задержать h на величину $\mathbf{M} - 1$, что приведёт и к соответствующей задержке R_{hf}. Если получаемый результат обозначить за g, получим:

$$g[\mathbf{n}] = R_{hf}[\mathbf{n} - \mathbf{M} + 1] = h[-\mathbf{n} + \mathbf{M} - 1] * f[\mathbf{n}].$$

По теореме о свёртке и схеме вычисления линейной свёртки через ДПФ нахождение g можно свести к следующему алгоритму:

1) обратить h по всем координатам (`numpy.flip`);

2) дополнить f и h нулями до размера $\mathbf{N} + \mathbf{M} - 1$ (`numpy.pad`);

3) вычислить дискретные спектры F и H (`numpy.fft.fft2`);

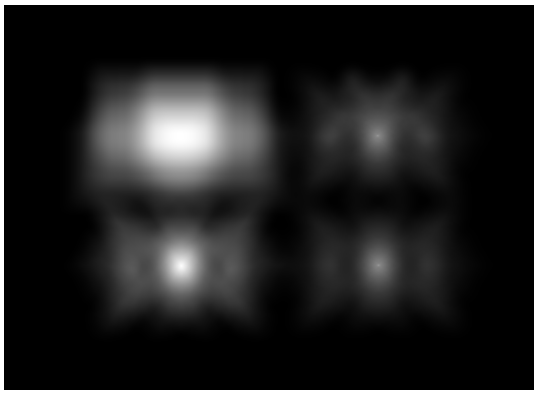
4) вычислить обратное ДПФ от $F \cdot H$ (`numpy.fft.ifft2`);

5) у результата взять модуль или отбросить мнимую часть — она должна равняться 0 с точностью до погрешности вычислений (`numpy.abs`).

Позиция очередного максимума $\mathbf{n}_{\max}$ в g соответствует координатам $\mathbf{n}_{\max} - \mathbf{M} + 1$ в R_{hf}, которые определяют начало координат потенциального совпадения с шаблоном (верхний левый угол). Таким образом, позиция максимума $\mathbf{n}_{\max}$ в координатах g соответствует правому нижнему углу совпадения. Если необходимо

определить верхний левый угол или центр совпадения, найденную позицию максимума нужно скорректировать на соответствующую величину (по сути, компенсировать задержку полностью или наполовину соответственно).

Для приведённых выше изображений g будет иметь следующий вид:



Соответствующие значения максимумов в центрах областей (вверху слева, вверху справа, внизу слева, внизу справа) равны примерно 15463, 9658, 15409 и 9658 соответственно. Видно, что максимум наблюдается не только на месте настоящих совпадений (второй и четвёртый максимумы), но и в местах, где на f были просто яркие области. Более того, максимумы на этих ярких областях превышают максимумы настоящих совпадений.

Избежать этой проблемы можно, если нормировать значения g на среднеквадратичные значения яркостей соответствующих областей f. Для удобства можно попутно добавить и нормировку значений h, в этом случае выходные значения будут гарантированно лежать в диапазоне от 0 до 1:

$$R_{hf}^{\mathrm{norm}}[\mathbf{n}] = \frac{\sum\limits_{\mathbf{m}} h[\mathbf{m}] f[\mathbf{n}+\mathbf{m}]}{\sqrt{\sum\limits_{\mathbf{m}} h^2[\mathbf{m}]} \cdot \sqrt{\sum\limits_{\mathbf{m}} f^2[\mathbf{n}+\mathbf{m}]}},$$

где в знаках сумм для удобства восприятия опущены пределы (напомним, что суммирование ведётся по всем допустимым отсчётам шаблона h).

С учётом $g[\mathbf{n}] = R_{hf}[\mathbf{n} - \mathbf{M} + 1]$, нормировку можно делать на последней стадии покомпонентным делением $g^{\mathrm{norm}}[\mathbf{n}] = \dfrac{g[\mathbf{n}]}{r[\mathbf{n}]}$, где

$$r[\mathbf{n}] = \sqrt{\sum_{\mathbf{m}} h^2[\mathbf{m}]} \cdot \sqrt{\sum_{\mathbf{m}} f^2[\mathbf{n} - \mathbf{M} + 1 + \mathbf{m}]} = r_0 \cdot r_1[\mathbf{n}]$$

представляет из себя матрицу нормировочных коэффициентов.

Первый коэффициент r_0 (нормировочный множитель h) представляет из себя константу (он зависит только от шаблона h). Второй коэффициент r_1 есть нормировочный множитель f в $\mathbf{M}$-окрестности точки $\mathbf{n}$. Вычисление суммы всех отсчётов в

$\mathbf{M}$-окрестности можно вычислить как свёртку с матрицей единиц соответствующего размера.

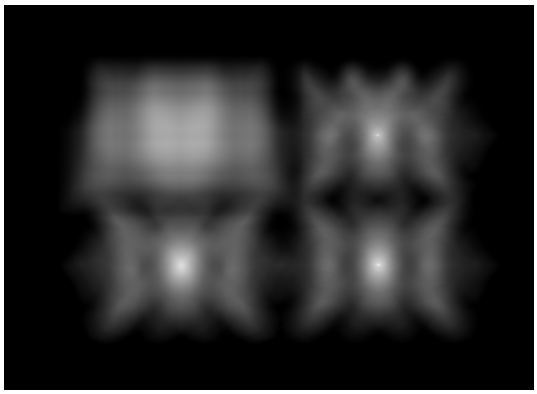
Таким образом, $g^{\mathrm{norm}}[\mathbf{n}]$ можно вычислить по следующему алгоритму:

1) по теореме о свёртке и схеме вычисления линейной свёртки через ДПФ вычислить свёртку f^2 с матрицей $J_{M_0,\ M_1}$ (матрица размера $M_0 \times M_1$, состоящая из одних единиц) — это будет $r_1^2[\mathbf{n}]$;

2) вычислить $r[\mathbf{n}] = \sqrt{\sum\limits_{\mathbf{m}} h^2[\mathbf{m}] \cdot r_1^2[\mathbf{n}]}$;

3) нормировать $g[\mathbf{n}]$ — вычислить $g^{\mathrm{norm}}[\mathbf{n}] = \dfrac{g[\mathbf{n}]}{r[\mathbf{n}]}$.

Использование нормировки даст следующий вид g^{norm}:



Соответствующие значения максимумов равны примерно 0.67, 0.91, 0.88 и 1.00 соответственно. Видно, что максимум, соответствующий настоящему совпадению, имеет максимальное значение.

Обратим внимание, что вышеприведённый метод может использоваться только при условии, что на изображении шаблон присутствует в том же масштабе и в той же ориентации (без поворота). Это достаточно сильное ограничение. Если это ограничение не выполняется, то требуется использовать более сложные алгоритмы (например, SIFT).

Однако в одном частном случае, когда на изображении присутствует только шаблон и задача заключается в нахождении параметров сдвига, поворота и масштаба одного изображения относительного другого, задача может быть решена за счёт некоторых свойств преобразования Фурье. В подразделе 1.3 будет приведена формулировка этих свойств, а в подразделе 1.4 на их основе сформулирован алгоритм сопоставления изображений.

1.3 Свойства сдвига, масштабирования и поворота

Пусть $F(\mathbf{\Omega})$ является преобразованием Фурье от $f(\mathbf{x})$ и задано некоторое афинное преобразование координат $\mathbf{y} = A\mathbf{x} - \mathbf{d}$, где A — некоторая невырожденная матрица, $\mathbf{d}$ — некоторый вектор сдвига.

Рассмотрим сигнал $g(\mathbf{x}) = f(A\mathbf{x} + \mathbf{d})$. Тогда известно, что Фурье-образы $G(\mathbf{\Omega})$ и $F(\mathbf{\Omega})$ связаны следующим образом:

$$G(\mathbf{\Omega}) = \frac{1}{|\det A|} \cdot F(A^{-T}\mathbf{\Omega}) \cdot e^{-j(\mathbf{A^{-T}\Omega},\ \mathbf{d})}.$$

Рассмотрим некоторые важнейшие частные случаи.

Во-первых, рассмотрим случай, когда присутствует только сдвиг: $A = E$ — единичная матрица. Тогда $\mathbf{y} = \mathbf{x} - \mathbf{d}$ и связь Фурье-образов упрощается до вида $G(\mathbf{\Omega}) = F(\mathbf{\Omega}) \cdot e^{-j(\mathbf{\Omega},\ \mathbf{d})}$. Таким образом, при сдвиге Фурье-образы совпадают с точностью до фазового множителя.

Во-вторых, рассмотрим случай, когда присутствует только однородное масштабирование: $A = \alpha E$ и $\mathbf{d} = \mathbf{0}$. Тогда $\mathbf{y} = \alpha\mathbf{x}$ и связь Фурье-образов упрощается до вида $G(\mathbf{\Omega}) = \frac{1}{\alpha^M} \cdot F\left(\frac{1}{\alpha}\mathbf{\Omega}\right)$. Таким образом, сжатие (растяжение) сигнала в α раз приводит к растяжению (сжатию) спектра в α раз и уменьшению (увеличению) его амплитуды в α^M раз.

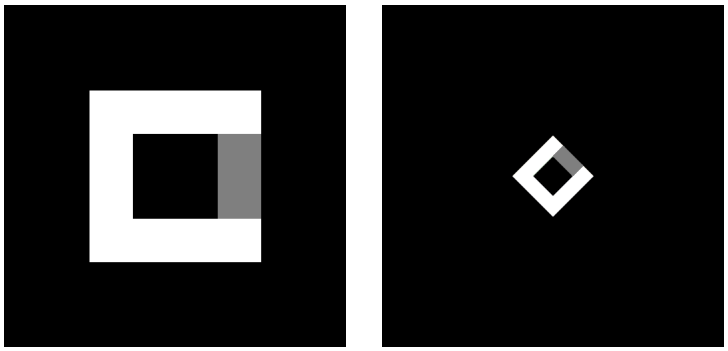
В-третьих, рассмотрим двумерный случай, когда присутствует только поворот: $A = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$ и $\mathbf{d} = \mathbf{0}$. Тогда $\mathbf{y} = A\mathbf{x}$, $A^T = A^{-1}$ и связь Фурье-образов упрощается до вида $G(\mathbf{\Omega}) = F(A\mathbf{\Omega})$. Таким образом, при повороте сигнала его Фурье-образ поворачивается на ту же величину.

1.4 Сопоставление изображений

Пусть дано изображение f, которое является сдвинутой, повернутой и отмасштабированной копией h. Необходимо найти эти самые параметры сдвига, поворота и масштаба.

Сначала необходимо перейти к амплитудным спектрам $|F|$ и $|H|$. Сдвиг никак не влияет на амплитудные спектры и в этом случае остаётся лишь определить параметры масштаба и поворота. Как только эти параметры будут определены, h можно подвергнуть соответствующему преобразованию (повернуть и отмасштабировать) и затем воспользоваться классическим корреляционным методом (описанным в подразделе 1.2) для определения координат сдвига.

Остаётся лишь решить задачу определения масштаба и поворота по амплитудным спектрам. Положим, что амплитудные спектры $|F|$ и $|H|$ **схематично** выглядят следующим образом соответственно:



Обратим внимание на несколько моментов.

Во-первых, на изображениях выше предполагается, что нулевая частота спектра находится в центре изображения. Напомним, что при вычислении ДПФ нулевая частота находится в начале координат (можно воспользоваться циклическим сдвигом `numpy.roll` или специальной функцией `numpy.fft.fftshift`).

Во-вторых, спектр намеренно сделан несимметричным для наглядности дальнейшего изложения (то есть формально сигналы f и h — комплекснозначные).

Кроме того, напомним, что так как $|F|$ выглядит растянутым относительно $|H|$, то в пространственной области f будет сжатым относительно h согласно вышерассмотренному свойству преобразования Фурье.

Далее эти амплитудные спектры можно рассматривать как обычные изображения, одно из которых повёрнуто и отмасштабировано относительно другого. Для определения параметров поворота и масштаба воспользуемся представлением в логполярных координатах.

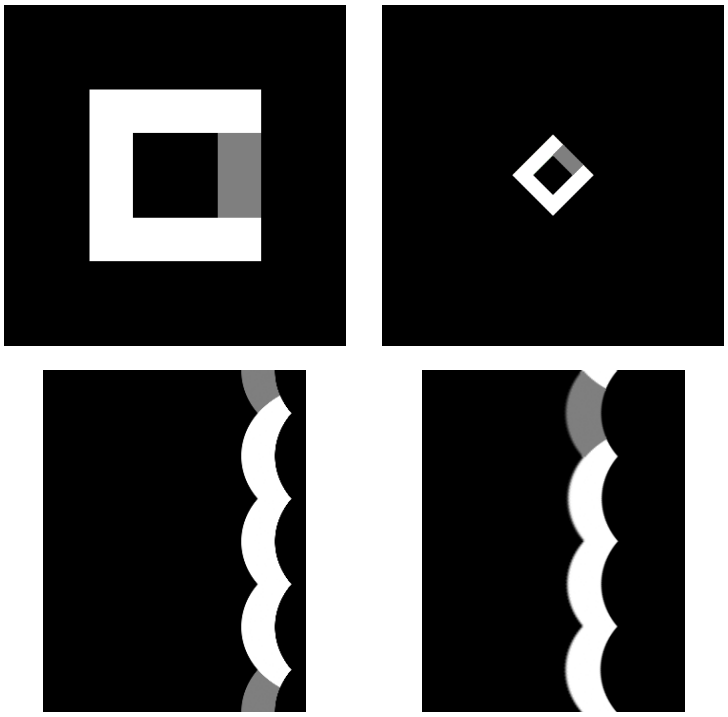
Пусть дано изображение $s[n_0, n_1]$ (s — source). Обозначим его представление в логполярных координатах как $d[n_\phi, n_\rho]$ (d — destination), при этом:

$$d[n_\phi, n_\rho] = s(n_0, n_1),$$
$$n_\rho = k \ln \rho,$$
$$\rho = \sqrt{n_0^2 + n_1^2},$$
$$\phi = \Delta\phi \cdot n_\phi,$$
$$n_0 = N_0 \operatorname{div} 2 - \rho \sin(\phi),$$
$$n_1 = N_1 \operatorname{div} 2 + \rho \cos(\phi),$$

где $\Delta\phi$ — используемый шаг по углу (определяет количество строк в d), k — используемый коэффициент масштабирования координаты ρ (определяет количество столбцов в d).

Обратим внимание, что в выражении $d[n_\phi, n_\rho] = s(n_0, n_1)$ для s использованы круглые скобки. Вычисляемые значения n_1 и n_2 являются вещественными, поэтому для получения соответствующего значения $s(n_0, n_1)$ потребуется интерполяция.

Обозначим представления $|F|$ и $|H|$ в логполярных координатах как $\widehat{F}$ и $\widehat{H}$. Эти представления будут выглядеть следующим образом (для наглядности приведём также представления в оригинальных декартовых координатах):



В логполярных координатах $\widehat{F}$ и $\widehat{H}$ оказываются **ни повёрнутыми**, **ни масштабированными**, а всего лишь **смещенными** друг относительно друга на некоторую величину $(\Delta n_\phi, \Delta n_\rho)$. Эту величину можно найти классическим корреляционным методом (см. подраздел 1.2). Обратим внимание, что вертикальная координата определяет угол, то есть является периодической. Это означает, что по вертикальной координате надо рассчитывать **циклическую** свёртку, а не линейную.

При этом из вертикальной координаты смещения Δn_ϕ можно сразу получить угол поворота. Рассмотрим, как из Δn_ρ можно извлечь коэффициент масштаба α.

Рассмотрим на $|H|$ две произвольные точки, лежащие на одной прямой с точкой центра изображения, и находящиеся от этого центра на расстоянии r_0 и r_1 соответственно. Для определённости положим $r_0 < r_1$, тогда расстояние между этими точками равно $r_1 - r_0$ в декартовых координатах. В логполярном представлении $\widehat{H}$ они окажутся лежащими на горизонтальной прямой с горизонтальными координатами $k \ln r_0$ и $k \ln r_1$ соответственно. Расстояние между ними будет равно $k(\ln r_1 - \ln r_0) = k \ln \dfrac{r_1}{r_0}$ в логполярных координатах.

Рассмотрим теперь $|F|$ — масштабированный вариант $|H|$, растянутый в α раз (напомним, что рассматривается спектр; в пространственной области это соответствует сжатию в α раз). Тогда вышерассмотренные точки окажутся лежащими на расстояниях αr_0 и αr_1 от центра в декартовых координатах. Расстояние между ними в декартовых координатах также отмасштабируется в α раз и станет равным $\alpha(r_1 - r_0)$.

В логполярном представлении $\widehat{F}$ горизонтальные координаты этих точек окажутся равными $k \ln \alpha r_1$ и $k \ln \alpha r_0$, а расстояние между ними окажется равным $k(\ln \alpha r_1 - \ln \alpha r_0) = k \ln \dfrac{\alpha r_1}{\alpha r_0} = k \ln \dfrac{r_1}{r_0}$, то есть в логполярных координатах они окажутся лежащими на том же самом расстоянии друг от друга, что и в неотмасштабированном $\widehat{H}$.

Итак, изменится только абсолютное положение этих точек, но не их положение друг относительно друга. Для смещения Δn_ρ любой из точек (возьмём, например, r_0) можно записать выражение $\Delta n_\rho = k \ln \alpha r_0 - k \ln r_0$, откуда можно выразить искомый параметр α.

2 Задание на лабораторную работу

2.1 Получение файлов

Загрузите файлы к лабораторной работе из репозитория курса.

Для выполнения лабораторной работы вам понадобится пакет `opencv-python`, который в коде импортируется как `cv2`. Кроме того, ещё нужен `numpy`, но он установится как зависимость.

Ознакомьтесь со вспомогательным кодом в файле `imio.py`. Обратите внимание, что вспомогательный класс для отрисовки изображений `Show` использует флаг `cv2.WINDOW_NORMAL`, который позволяет масштабировать и изменять размеры окна, не сохраняя при этом соотношение сторон. При необходимости этот флаг можно убрать.

В предоставляемом коде для амплитудных спектров используется суффикс `_s`, для амплитудных спектров в логполярных координатах — суффикс `_s_lp`.

⚠ При реализации **запрещено** использовать высокоуровневые функции, например, `cv2.matchTemplate`, `cv2.warpPolar` и т. п. Допускается использовать только базовые numpy-операции (`abs`, `concatenate`, `flip`, `pad` и т. п.) над массивами и явно разрешённые в тексте функции.

2.2 Реализация корреляционного метода

Выполняется в файле `corr2d.py`.

Реализуйте функцию `corr2d_unnormalized` для вычисления ненормированной корреляции. Используйте `numpy.fft.fft2` и `numpy.fft.ifft2`.

Реализуйте функцию `corr2d` для вычисления нормированной корреляции (в этой функции нужно вызвать уже реализованную `corr2d_unnormalized`, рассчитать нормировочные коэффициенты $r[\mathbf{n}]$ и почленно поделить). Обратите внимание, что могут встретиться случаи, когда $r[\mathbf{n}] = 0$. Для обхода проблемы деления на ноль можно, например, все значения r, которые меньше 1, заменить на 1.

Косвенно убедитесь в правильности реализации. Во-первых, сохраняемые изображения визуально должны совпадать с референсными (с суффиксом `_ref`). Во-вторых, выводимые в консоли максимумы должны быть следующими:

`(15463.18, 9658.18, 15409.05, 9658.18)`

`(0.67, 0.91, 0.88, 1.0)`

2.3 Реализация перевода декартовых координат в логполярные

Выполняется в файле `log_polar.py`.

В функции `bilinear_interpolate` реализуйте билинейную интерполяцию.

В функции `log_polar` реализуйте преобразование коодинат из декартовых в логполярные. Для простоты можно считать, что $\Delta\phi = 1^\circ$, то есть высота выходного изображения равна 360, что соответствует полному обороту. Косвенно проверьте правильность реализации на изображениях `log_polar_in_?_s.png` — результаты работы должны совпадать с соответствующими референсными изображениями `log_polar_out_?_s_lp_ref.png`.

В функции `log_polar_match` преобразуйте данные амплитудные спектры в логполярное представление, рассчитайте корреляцию (`corr2d` из прошлого шага), найдите максимумы и по этим максимумам рассчитайте потенциальные параметры поворота и масштаба. Все полученные данные упакуйте в объект класса `LogPolarMatch`. Обратите внимание на следующие моменты.

Во-первых, считается, что на вход функции подаются амплитудные спектры (на данном этапе это искусственные `log_polar_in_?_s.png`).

Во-вторых, напомним, что в логполярном представлении вертикальная координата является углом и по ней нужно рассчитывать **циклическую** свёртку, а не линейную.

Итогом вычисления корреляции будет некоторый сигнал `corr_raw`. Перед анализом максимумов его желательно отфильтровать — провести пороговую обработку и отобрать локальные максимумы с помощью морфологической операции:

```
corr = corr_raw.copy()
corr[corr < corr_threshold] = 0
kernel = np.ones((dilate_kernel_size, dilate_kernel_size))
corr[corr < cv2.dilate(corr, kernel)] = 0
```

Тогда далее перебор всех максимумов сводится до следующей конструкции:

```
for row, col in np.argwhere(corr > 0):
    pass
```

Косвенно убедитесь в правильности реализации. Во-первых, корреляционный сигнал `corr_raw` и отфильтрованный `corr` должны визуально совпадать с соответствующими референсными изображениями. Во-вторых, выводимые в консоли максимумы и соответствующие рассчитанные параметры поворота и масштаба должны быть следующими:

```
corr[ 45, 331]=0.91     angle=  45     scale=3.00
corr[315, 331]=0.98     angle= -45     scale=3.00
```

2.4 Определение поворота, масштаба и сдвига

Выполняется в файле `main.py` с использованием реализованных на прошлых шагах функций.

Для данных двух изображений (`main_in_base.png` и `main_in_pattern.png`) найдите возможные комбинации поворота и масштаба. Для каждой такой комбинации трансформируйте соответствующим образом шаблон (`scipy.ndimage.zoom` и `scipy.ndimage.rotate`) и с помощью `corr2d` рассчитайте соответствующий корреляционный сигнал g_i, где i — номер рассматриваемой комбинации.

Сформируйте итоговый результат g как изображение с тремя каналами, где:

1) первый канал хранит значение корреляции, выбираемое как максимальное из всех g_i;

2) второй и третий каналы хранят угол и поворот, соответствующие выбранному максимальному значению корреляции в этой точке.

Итоговый результат g можно отфильтровать аналогично тому способу, который использовался в `log_polar_match`. Распечатайте в консоль все максимумы (а именно: соответствующие значения корреляции, параметры сдвига, поворота и масштаба). Условный пример вывода максимума:

```
0.90 shift=[800, 700] angle=60 scale=5.00
```

3 Формат сдачи

Программа, которая по данным двум изображениям `main_in_base.png` и `main_in_pattern.png` автоматически определяет и распечатывает в консоль параметры сдвига, масштаба и поворота, которые надо применить ко второму изображению, чтобы получить первое.