Компьютерное зрение

Практический курс

Савельева Юлия Олеговна
i.o.saveleva.kpfu@gmail.com

2-й семестр, 12.03.2020 г.

BRIEF Descriptor

Что необходимо реализовать на первом этапе

1. Реализовать формирование пар координат в патче, которые будут использоваться для сравнения (генерация случайных координат из равномерного и нормального распределений, GI и GIII в статье по BRIEF). Необходимо при генерации указать определенный seed, чтобы при вызове дескриптора для следующей картинки использовались те же координаты внутри патча. Множество первых точек пары назовем pos1, а множество вторых точек пары - pos2

2. Отбросить те ключевые точки, которые находятся слишком близко к границе изображения и поэтому не позволяют сравнить все пары точек в патче

3. Написать цикл по всем ключевым точкам, вычислить в каждой из них бинарный вектор, путем сравнения точек, вычисленных на шаге 1

BRIEF Descriptor

Учет угла поворота и масштаба

1. Так как ключевые точки вычисляются отдельно для каждого масштаба изображения (x, x/2, x/4, x/8), нужно подавать на вход дескриптору картинку такого же размера, как и на вход детектору, а уменьшать размер патча НЕ нужно

2. Поворот патча на угол, соответствующий данной ключевой точке, будет осуществляться следующим образом:

    a) Перед началом вычисления бинарных векторов в каждой ключевой точке необходимо создать маски поворота для всех возможных углов. Так как множество возможных вещественных значений угла бесконечно, то разделим окружность 0° - 360° на сектора в 12° и для каждого значения угла α = {0°, 12°, 24°, 36°, 48°, …, 348°, 360°} заранее создадим матрицу поворота:
    m = cv2.getRotationMatrix2D(center=(0, 0), angle=α, scale=1)

BRIEF Descriptor

Учет угла поворота и масштаба

b) Во время вычисления бинарного вектора для конкретной ключевой точки необходимо взять матрицу поворота угла, который ближе всего по значению к углу, соответствующему ключевой точке (Например, oriented FAST вернул вам угол π/4, значит ему будет лучше всего соответствовать матрица поворота 48°)

c) Для каждой пары точек (x1, y1) из pos1 и (x2, y2) из pos2 (до прибавления к ним значения x и y ключевой точки) посчитать их новые значения одновременно, путем умножения матрицы всех сгенерированных координат на транспонированную матрицу поворота:

```
new_pos1 = np.stack([pos1[:, 1], pos1[:, 0], np.zeros(len(pos1))], axis=1)
new_pos1 = np.round(np.dot(new_pos1, rotation_matrix.T))
new_pos2 = np.stack([pos2[:, 1], pos2[:, 0], np.zeros(len(pos2))], axis=1)
new_pos2 = np.round(np.dot(new_pos2, rotation_matrix.T))
```

BRIEF Descriptor

Учет угла поворота и масштаба

d) Затем уже прибавить ко всем новым значениям координат new_pos1, new_pos2 координаты ключевой точки x, y и вычислить бинарный вектор для этой ключевой точки:

```
keypoint_pos1 = (new_pos1[:, 1] + y, new_pos1[:, 0] + x)
keypoint_pos2 = (new_pos2[:, 1] + y, new_pos2[:, 0] + x)
descriptor = img_gray[keypoint_pos1] < img_gray[keypoint_pos2]
```

e) Проделать описанные выше операции b-d для каждой ключевой точки

ORB

План объединения компонент в систему

Из полученной на вход в ORB картинки строим, например, Gaussian Scale Pyramid (https://docs.opencv.org/2.4/doc/tutorials/imgproc/pyramids/pyramids.html). Для картинки в каждом масштабе находим ключевые точки и дескрипторы следующим образом:

1. Подаем картинку в текущем масштабе на вход FAST, получаем координаты ключевых точек K_FAST={xi, yi} и углы поворота A={ai}

2. Подаем картинку в текущем масштабе на вход Harris Corner Detector, получаем координаты ключевых точек K_Harris={xi, yi}, уверенность R = {ri}

3. Получаем пересечение множеств: indices = where(K_Harris & K_FAST)
   K = K_Harris[indices]
   A = A[indices]
   R = R[indices]

ORB

План объединения компонент в систему

4. Фильтруем с помощью NMS координаты K и углы A используя уверенности R
5. Подаем отфильтрованные координаты K, углы A, а так же картинку в текущем масштабе (patch_size не меняем!) и получаем дескрипторы ключевых точек $D=\{di, di \in \{0, 1\}^n\}$

Полученные ключевые точки (пересчитанные по возвращении ORB под размер оригинального изображения) и дескрипторы для каждого масштаба картинки складываются в один контейнер, и все они вместе будут участвовать в последующем матчинге.

ORB

Проверить работу системы

1. Возьмите две картинки, оригинальную img1 и измененную картинку img2 (поменяйте масштаб, поверните картинку)

2. Подайте обе картинки на вход в ORB и получите kp1, des1 и kp2, des2, где kp - это список объектов cv2.KeyPoint

3. Найдите наиболее похожие точки с помощью cv2.BFMatcher и нарисуйте их с помощью cv2.drawMatches (пример можно найти здесь https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_matcher/py_matcher.html)

На следующее занятие

1. BRIEF Descriptor 
2. Rotated BRIEF 
3. ORB 
4. Object Localization with Key Points
5. Bag of Visual Words