Discovery of Latent 3D Keypoints via End-to-end Geometric Reasoning

Панаетов Александр

Higher School of Economic

30 января 2020 г.

Обзор

1. Задача оценки смещения камеры
2. Обучаемые кейпоинты
3. Архитектура
4. Результаты

Оценка смещения камеры

- Положение камеры задаётся матрицей поворота R и вектором сдвига t, полное преобразование обозначается
$$T = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$$
- Нам предоставляют две картинки одного объекта, снятого с двух камер
$$P' = [R' \ t'] \text{ и } P'' = [R'' \ t'']$$
- Необходимо определить матрицу относительного поворота
$$R = R'' R'^T$$

ShapeNet

- ShapeNet - датасет с $3D$ моделями разных объктов - машины, стулья и т.п.
- Присутствует много полезных аннотаций - ориентация объекта, оси симметрии, некоторые стандартные кейпоинты и прочее
- Некоторые стандартные обозначения: x, y, z - координаты точки в простарнстве, u, v - координаты точки на картинке, полученной камерой, f - фокальное расстояние камеры. Переход от однородных координат камеры к координатам пикселя и глубины:

$$\pi([x, y, z, 1]^T) = [\frac{fx}{z}, \frac{fy}{z}, z, 1]^T = [u, v, z, 1]^T$$

Как найти матрицу относительного поворота?

- По изображению предсказываем кейпоинты - неподвижные точки объекта. Если на первом изображении кейпоинт оказался на колесе машины, то на втором изображении он тоже должен оказаться на том же колесе машины в том же месте.
- Кейпоинт представляет из себя три координаты - u, v, z
- По N парам кейпоинтов X и X' можно восстановить матрицу поворота: $\tilde{X}$, $\tilde{X'}$ - вычли среднее. Тогда $U, \Sigma, V^T = SVD(\tilde{X}, \tilde{X'}^T)$. И матрица поворота вычисляется как

$$R = V\ diag(1, 1, ..., det(VU^T))U^T$$

Обучаемые кейпоинты

- Можно использовать уже размеченные кейпоинты, как делалось ранее: подаём на вход картинку камеры, хотим получить N заранее известных размеченных кейпоинтов; потом для второй картинки получаем эти же N кейпоинтов, и находим матрицу поворота.
- Авторы предлагают позволить сетке самой обучать кейпоинты. Идея в том, что можно напрямую оптимизировать получаемую матрицу поворота, а не только кейпоинты. Так, ранее размеченные кейпоинты возможно не оптимальны для вычисления матрицы поворота, а обученные могут оказаться лучше.

Архитектура

- Для предсказания кейпоинтов используется CNN. На вход подается картинка, на выходе - N карт $g_i(u, v)$ для каждого кейпоинта с его вероятностью оказаться в каждом пикселе (u, v) и карта глубины $d_i(u, v)$. Далее конкретные координаты u, v каждого кейпоинта - просто матожидание по соответствуйщей карте, аналогично глубина:

$$[u_i, v_i]^T = \sum_{u,v} [ug_i(u, v), vg_i(u, v)]^T$$

$$z_i = \sum_{u,v} d_i(u, v) g_i(u, v)$$

Multi-view consistency loss

- Для того, чтобы один кейпоинт находился в одной и той же точке объекта используется multi-view consistency loss: предсказанный кейпоинт для первого изображения должен совпадать с предсказанным кейпоинтом второго изображения (и наоборот) после смещения камеры:

$$[\tilde{u}, \tilde{v}, \tilde{z}, 1]^T \sim \pi T \pi^{-1}([u, v, z, 1]^T)$$

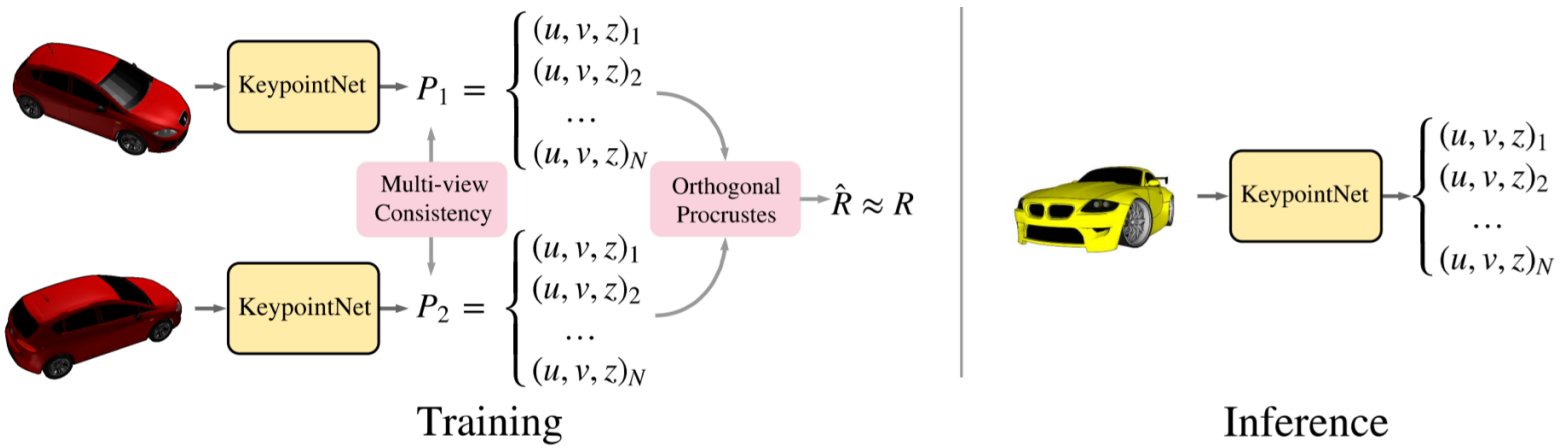
$$[\tilde{u'}, \tilde{v'}, \tilde{z'}, 1]^T \sim \pi T^{-1} \pi^{-1}([u', v', z', 1]^T)$$

$$L_{con} = \frac{1}{2N} \sum_{i=1}^{N} \| [u_i, v_i, u'_i, v'_i]^T - [\tilde{u}_i, \tilde{v}_i, \tilde{u}'_i, \tilde{v}'_i]^T \|^2$$

Relative pose estimation loss

- Поскольку главная задача - предсказать матрицу поворота, то предсказанные кейпоинты должны позволять это делать. Для этого используется relative pose estimation loss:

$$L_{pose} = 2 \arcsin\left(\frac{1}{2\sqrt{(2)}} \|\tilde{R} - R\|_F\right)$$



Separation loss

- Другим довольно очевидным требованием является то, что кейпоинты должны быть различны. Мы не хотим, чтобы 5 кейпоинтов начали указывать на одно крыло самолёта. Поэтому предлагается дополнительный separation loss:

$$L_{sep} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j \neq i}^{N} \max(0, \delta^2 - \|X_i - X_j\|^2)$$

Silhouette consistency loss

- Также разумено требовать, чтобы кейпоинты находились внутри объекта. Без этого они могут располагаться где угодно: например, точка на полу по середине между двух ножек стула. Чтобы этого избежать, используют silhouette consistency loss. Сегментационная маска объекта $b(u, v) \in 0, 1$, где 1 - наш объект.

$$L_{obj} = \frac{1}{N} \sum_{i=1}^{N} - \log \sum_{u,v} b(u, v) g_i(u, v)$$

- Чтобы карта кейпоинта имела явный пик, добавляют

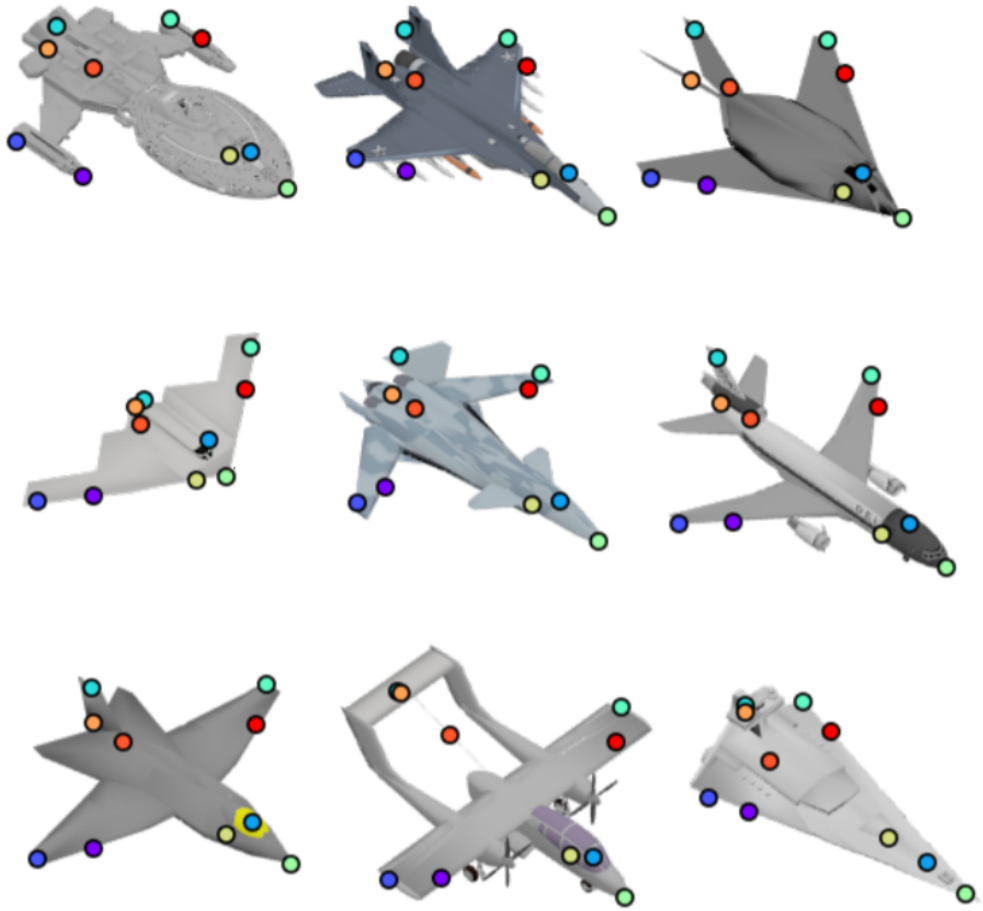
$$L_{var} = \frac{1}{N} \sum_{i=1}^{N} \sum_{u,v} g_i(u, v) \| [u, v]^T - [u_i, v_i]^T \|^2$$

Результаты

- Подход с обучаемыми кейпонитами показал лучшие результаты по сравнению с предудущим подходом (размеченные кейпоинты), в таблице ниже приведены mean angular distance error (то, что стояло в L_{pose}):

Метод	Машины	Самолёты	Стулья
Supervised	16.268	18.350	21.882
KeypointNet	11.310	17.330	14.572

Самолёты

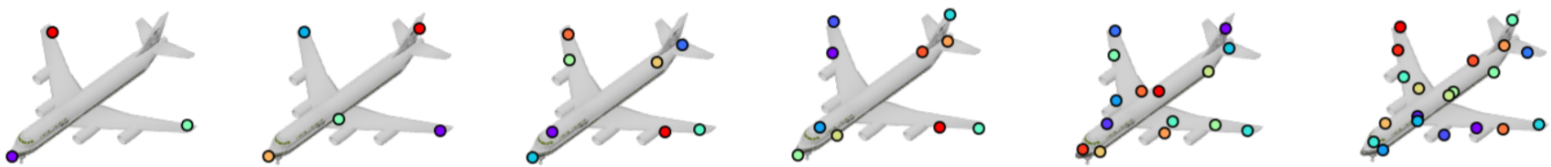


Машинки

Стулья

Деформации

Добавляем кейпоинты



Реальные машинки

Источники

- Supasorn Suwajanakorn, Noah Snavely, Jonathan Tompson, Mohammad Norouzi, "Discovery of Latent 3D Keypoints via End-to-end Geometric Reasoning"(2018)

Вопросы

- Что представляются из себя multi-view consistency loss, relative pose estimation loss? Зачем они нужны?
- Что представляют из себя 3D keypoints из статьи? Как их предсказывает KeypointNet?
- Что способствует предсказанию кейпоинтов в разных местах объекта? Почему они не выходят за границы объекта?