用 Stable Diffusion 的 feature 來做 template matching

出自

題目

Object Pose Estimation via the Aggregation of Diffusion Features

Introduction

背景

動機

有些 pose estimation 的方法會用 template 的 feature 與 query 的 feature 進行匹配，而這個 feature 非常重要，不能隨便亂學

text-to-image diffusion models 可以產出 discriminative features， 因為

1. condition on text
2. different time step ⇒ 可以抓到 diverse 特徵

目的

diffusion model freeze，接著加總 features of different granularity

分別做出了三種架構 : Arch. (a), Arch. (b) and Arch. (c) 來加總 diffusion features

最後覺得 Arch c 的效果最好

相關研究

object pose estimation 這個任務，是**輸入一張 2D 圖片**，然後**輸出**圖片中**物體的 6D 姿態**（位置和旋轉）。

object pose estimation 的方法有三類

1. Indirect methods : 著重於求出 2D-3D correspondences，並會接著以 PnP and RANSAC 來求出 6d pose

   BB8 [2017] : input query image 與 object 3d cad model，這個 model 中有該物體的 8 個 3d bounding box頂點，接著 BB8 預測這 8 個點在 query image 的哪裡，稱為 eight_point_predict，接著用 PnP 推出 object 的 6d pose

   PnP 的目的是求出 3D 旋轉矩陣 R 以及 3D 位移矩陣 T
   他會把 cad model 經過 R, T 之後投影到 2D 得到 eight_point_reproject， 希望求出正確的 R, T 使 eight_point_reproject 能夠和 eight_point_predict 越近越好

PVNet [2019]：
我們會幫 query image 中的每一個 pixel 預測 8 個 2d 向量，代表說這個 pixel 覺得 eight_point_predict 在 query image 中的哪裡，接著透過 RANSAC 找到聚焦最多選票的 8 個位置，他們就是 eight_point_predict。 接著我們需要 cad model 並使用 PnP 算出 object 的 6d pose

2. direct methods : directly outputting the object's pose regression

   SSD-6D：平移用 Regression 來解，旋轉拆成 out of plane rotation 與 in-plane rotation，前者用球平面切割來決定分類，後者每 10 度切成一個分類。 6d pose estimation 使用 regression + classification 來解

   將 PnP 可微，那 PnP 的結果也可以 backpropagation 到前面預測 eight_point_predict 時的參數

3. template-based methods : 從 cad model 中做出 template

   [1] 的資料集中，有多張 同一個物體但是不同角度的訓練資料，讓 model 更學會「同一個物體在不同 pose 中看起來不一樣」 這件事情

   AAE：用 auto encoder 來將 template 與 query image 編碼
   這個 auto encoder 是 input augmented image output original image
   所以可以學會不被雜訊干擾的 encoding

   DeepIM：iteratively progressive 直到達到 max iteration count 或是 residual < episolon時就會停止
   他的做法是接在其他的 pose estimator 後面，從 pose estimator 猜的 初始pose 下去 refine
   接著他會去將 3d cad model 依照初始 pose render，這張 render 與 query image 比較得到差異，用這個差異來預測 residual

   CosyPose continuous rotation parametrization

novel object pose estimation： 常假設 object 已經被 localize 了 (by object detection 或是 object segmentation)

1. Multi-path learning for object pose estimation across domains. : 用 multiple decoders 來處理不同 object ⇒ inference 時一定要用相似的 object 才能做出來
2. [24, 46] 用 local object representations 但效果仍然不好 ⇒ 是因為 discriminative features 不夠，無法有效的區分 objects

3. Template-pose : input image，將 image 經過 model 成為 representation 後，再去預測 se(3)。 這個 model 是 他們 fine-tune 一個 pre-trained model

Diffusion 前置知識

PDM：input 是一個雜訊，它的大小要和 output 圖片大小相同

LDM：input 是一個雜訊 embedding， output 是一個清晰的 embedding

Reverse Process：Diffusion Model 的每一個 step 都透過將 x_t image 與 step t 丟給 Unet 預測了一個 noise ϵ_t ，接著透過以下公式算出 x_t-1

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right) + \sigma_t z$$

DDIM vs DDPM

DDIM 是 Discriminative ，所以相同的 input 會有相同的 output，而他的生成時間比較短，因為某些 step 可以跳過。 他的 reverse process 公式為 :

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right)$$

DDPM：生成的時候使用 DDPM，才會同一個 Prompt 有不同的結果

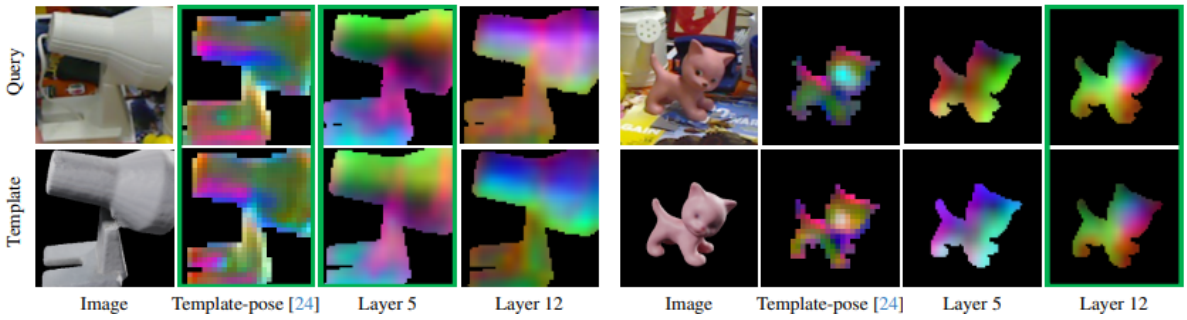
不過他們兩個的 forward process 公式都長一樣 :
要特別注意的是，雖然公式看起來像一步到位，但她實作時仍是經過 t 步才生成的

$$x_t = \sqrt{\alpha_t} x_0 + \sqrt{1-\alpha_t} \epsilon_t, \text{ where } \epsilon_t \sim \mathcal{N}(0,1)$$

方法

1. Feature : query image 與 template 都用 encoder 做成 feature

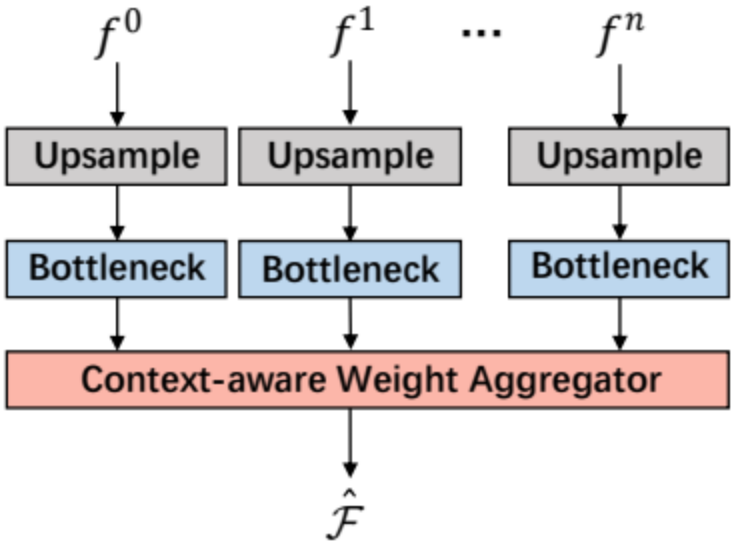
$$\mathcal{F} = \Phi_{encoder}\left(I\right)$$



將 feature map 的每一個 pixel 做 PCA 將 c 維壓成 3 為後塗色而成，而左側是 seen object 右邊是 unseen object ， 可以看到在 unseen object 時 diffusion feature 的效果比較好

2. 如何使用本實驗的 Stable Diffusion Model :
    a. text input 的地方都給予 “ ”
    b. 我們會給 Stable Diffusion t = 0 以及一張 image (template 或是 query)，接著 Stable Diffusion 會吃掉之後經過 Unet 給予 feature
    c. 因為 Denoise 需要詳細的了解 image 才能達到，所以說 Stable Diffusion 給予的 feature 也有豐富的 image 特徵，完全可以用來做 pose estimation
    d. 這些 feature 外接 aggregator

3. aggregator (這邊只介紹效果最佳的 arch c)



Arch. (c)

1. bottleneck layer = 3 層 conv + Relu + skip connection
2. 為了避免 overfitting 以及因為訓練不穩定導致 Stable Diffusion 的 output 變調，最後那層 (aggregator) 的參數 initialize 成 0，讓訓練初期的 gradient 不要那麼劇烈變化，
   也就是說讓 model 溫和學習

Experiments

可以學習的地方

問題