Corso di Computer Vision and Image Processing M
Perrone Alessandro [000888508]
Anno Accademico 2019/2020

SignAdvisor

SignAdvisor è un'applicazione di realtà aumentata che permette di ottenere informazioni relative al rating di ristoranti semplicemente scattando una foto della facciata di questi ultimi. Oltre al rating viene inserito un collegamento cliccabile che permette di visionare la pagina Trip Advisor dei ristoranti affiliati.

Partendo dalle insegne dei ristoranti, l'applicazione esegue una Object Detection, andando a ricercare all'interno dell'immagine passata dall'utente un'istanza delle possibili insegne presenti nel database.
Il processo di Object Detection è basato sulla ricerca ed il successivo match delle Local Invariant Features, che permettono di andare a ricercare la model image all'interno della target image.
L'utilizzo delle Local Invariant Features prevede di andare ad eseguire quattro steps:

1. **Detection**: identificazione dei keypoints nella model image e nella target image.
2. **Description**: creazione di un descriptor per ogni keypoint individuato.
3. **Matching**: match tra i descriptor della model image e della target image al fine di verificarne l'uguaglianza
4. **Position Estimation**: determinazione della posizione esatta della model image individuata all'interno della target image

Una volta nota la posizione viene eseguito il warping dell'immagine contenente il rating sulla posizione stimata dell'insegna del ristorante.

Keypoints detection

I Keypoints sono pixel dell'immagine che presentano delle caratteristiche distintive in termini di:

- Ripetibilità: un keypoints detector deve essere in grado di individuare gli stessi keypoints in differenti views della stessa immagine
- Carattere Informativo: i keypoints individuati devono essere circondati da informazioni che successivamente permettano un agile calcolo del descriptor e facilitino la fase di match.

Per individuare i keypoints all'interno della target image e della model image ho utilizzato il SIFT detector.

```
sift = cv2.xfeatures2d.SIFT_create()
kp_train = sift.detect(img_train_bw)
```

Keypoints Description

Nella fase di description, per ogni keypoints individuato si va a calcolare un descriptor basandosi sui pixel appartenenti all'intorno dello stesso.
Anche in questo caso ho sfruttato SIFT, il quale garantisce invarianza rispetto allo scaling ed alla rotazione, andando a calcolare il descriptor nell'immagine smoothata nella quale è stato individuato il keypoints. Per garantire invarianza rispetto alla rotazione, si determina il valore dell'orientazione canonica.

```
kp_train, descriptor_train = sift.compute(img_train_bw, kp_train)
```

KeyPoints Matching

Nella fase di match viene eseguita una NNS (nearest neighbourhood search), in cui per ogni descriptor della target image si cerca il nearest neighbour tra quelli presenti nella model image.
Per rendere la ricerca più efficiente vengono usate delle tecniche di indexing, come ad esempio il kd-tree, simile ad un albero binario ma in grado di effettuare la ricerca all'interno di spazi con k dimensioni.
In Open cv la ricerca tramite kd-tree è implementata nell'algoritmo FLANN,Fast Library for Approximate Nearest Neighbors. Questa libreria raccoglie una serie di algoritmi di NNS opportunamente ottimizzati.
Per utilizzare FLANN è necessario passare come argomento il tipo di algoritmo da utilizzare, ed il numero di volte in cui l'albero viene ricorsivamente

attraversato prima di fornire la soluzione, ovviamente valori maggiori determinano maggior precisione ma minori performance.

```
index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)
search_params = dict(checks=50)
flann = cv2.FlannBasedMatcher(index_params, search_params)
matches = flann.knnMatch(descriptor_query, descriptor_train, k=2)
```

Per validare i match individuati utilizzo il metodo della **distance ratio** proposto da Lowe. Si considera il rapporto tra la distanza dei due match più vicini, che sarà considerato valido solo se minore di una certa soglia.

$$\frac{d_{NN}}{d_{2-NN}} < 0.7$$

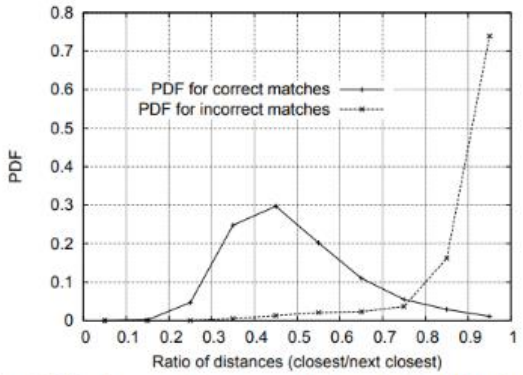

Figure 11: The probability that a match is correct can be determined by taking the ratio of distance from the closest neighbor to the distance of the second closest. Using a database of 40,000 keypoints, the solid line shows the PDF of this ratio for correct matches, while the dotted line is for matches that were incorrect.

```
good_matches = []
for m, n in matches:
  if m.distance < 0.7 * n.distance:
    good_matches.append(m)
return good_matches
```

Validazione della ricerca

Per verificare che la model image sia effettivamente stata trovata, si eseguono due checks:

1. Il primo si basa sul numero di elementi presenti nel vettore *good_matches*, andando a verificare che sia maggiore di una soglia. Nel mio caso Th = 15. Poiché questo metodo non è sempre affidabile, ho deciso di inserire un secondo controllo.

2. Effettuare una GHT sfruttando solo i match corretti.

GHT

La GHT permette di ricercare all'interno di una target image una model image di forma arbitraria, non necessariamente esprimibile tramite equazione.
L'algoritmo si suddivide in due fasi:

1. Fase off-line. Partendo dalla target image, si calcola un reference point, ho utilizzato il centro dell'immagine, e per ogni keypoints si calcola la norma del vettore tra reference point e keypoints. Questi valori saranno utilizzati per stimare il nuovo baricentro.
2. Fase on-line. Si inizializza un accumulator array, con le dimensioni dell'immagine, considerando anche un fattore di scala ed un angolo di rotazione, ottengo un accumulator array a quattro dimensioni. Pero ogni keypoints della target image, stima il valore del reference point considerando la differenza in termini di gradi tra l'orientazione del gradiente dei keypoints della target image e model image, andando a considerare come fattor di scala il rapporto delle scale alle quali sono stati idividuati i keypoints.

Figure 1 Posizione stimata del baricentro

Figure 2 ROI estratto dalla target image partendo dal baricentro stimato

Figure 3 Immagine finale ottenuta dopo il warping della rating image.

Terminata la fase on-line, si estraggono le coordinate del punto che ha ottenuto il maggior numero di voti. Successivamente estraggo il ROI e calcolo il valore di similarità sfruttando la funzione Zero Mean Normalised Cross Correlation. Andando a salvare il valore massimo ed il relativo reference point. Successivamente ho validato un'ultima volta il match tramite la relazione:

if score >= 0.5 **and** score <= 1:

Solo a questo punto il match è valido e si passa alla fase successiva.

Position estimation

L'obiettivo di questa fase è l'individuazione della posizione della model image all'interno della target image. Per fare ciò possiamo andare a determinare un'omografia considerando come punti corrispondenti i keypoints della model

image e della target image. Per ottenere un calcolo dell'omografia che sia robusto al rumore possiamo sfruttare l'approssimazione fornita dall'algoritmo RANSAC, un metodo iterativo per andare a stimare i parametri di un modello matematico, nel nostro caso l'omografia, partendo da un insieme di dati contenenti degli outliers, che in questo caso non andranno ad influenzare la stima del modello.
Una volta nota la posizione, estraggo dal database i dati relativi al rating ed il link per il collegamento a Trip Advisor, per effettuare il warping.

Warping

Per proiettare la rating image sull'insegna determino in primo luogo la trasformazione che permette il mapping tra le immagini.

PT = cv2.getPerspectiveTransform(rating_points, sign_points)

Dove *sign_points* rappresentano i quattro angoli della model image proiettati sulla target image tramite l'omografia stimata precedentemente.
Nota PT eseguo il warping della rating image, successivamente è necessario ristabilire i valori di intensità dei pixel della target image, andando ad utilizzare una maschera, con dimensioni pari alla rating image, e intensità dei pixel bianca. Questa maschera verrà proiettata tramite la matrice PT, e tutti i pixel con valore pari a 0, verranno sostituiti con i pixel della target image.
Lo stesso procedimento è stato eseguito per eliminare lo sfondo della rating image.

Osservazioni

La validazione tramite GHT risulta costosa in termini computazionali, soprattutto nella fase di ricerca del massimo, per questo ho deciso di suddividere l'accumulator array in due array, uno contenente le informazioni relative alla posizione del baricentro e l'altro contenente le informazioni relative a fattore di scala ed angolo.
La validazione tramite GHT in alcuni casi fallisce in quanto va a stimare il valore del baricentro, ed il ROI estratto partendo da questo valore potrebbe contenere parti esterne all'insegna andando a ridurre lo score.