

Домашна задача 5

Препознавање на слики од постери со SIFT

Во директориумот Database се дадени слики од 22 постери. Сликите hw7_poster_1.jpg, hw7_poster_2.jpg, и hw7_poster_3.jpg се прашални слики.

За секоја прашална слика искористете го следниот алгоритам за да ја одредите најсличната слика (сликата која го содржи соодветниот постер) од сликите зададени во директориумот Database:

- Генерирајте SIFT дескриптори за прашалната слика
- Со користење на пребарување со најблиски соседи споредете ги генерираните SIFT дескриптори на прашалната слика со SIFT дескрипторите на сликите од директориумот Database
- За локалните дескриптори кои се совпаднале пронајдете inliers со користење на RANSAC и хомографија
- Прикажете ја сликата со најголем број на inliers која всушност е најслична на прашалната слика

Решение:

```
import cv2
import numpy as np
import os

def find_best_match(query_descriptors, database_descriptors, query_keypoints,
database_keypoints):
    matcher = cv2.BFMatcher()
    matches = matcher.knnMatch(query_descriptors, database_descriptors, k=2)

    good_matches = []
    for m, n in matches:
        if m.distance < 0.75 * n.distance:
            good_matches.append(m)

    if len(good_matches) > 10:
        src_pts = np.float32([query_keypoints[m.queryIdx].pt for m in
good_matches]).reshape(-1, 1, 2)
        dst_pts = np.float32([database_keypoints[m.trainIdx].pt for m in
good_matches]).reshape(-1, 1, 2)

        M, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC, 5.0)
        if M is not None:
            matches_mask = mask.ravel().tolist()
            inliers = sum(matches_mask)
            return inliers

    return 0

directory = os.getcwd()

query_images = ["hw7_poster_1.jpg", "hw7_poster_2.jpg", "hw7_poster_3.jpg"]

database_images = sorted([file for file in os.listdir("Database") if
file.lower().endswith(('.png', '.jpg', '.jpeg'))])

for database_image_path in database_images:
    image = cv2.imread(os.path.join(directory, "Database",
database_image_path))
    image = cv2.resize(image, (800, 600))
    cv2.imwrite(os.path.join(directory, "Database", database_image_path),
image)

for query_image_path in query_images:
    image = cv2.imread(os.path.join(directory, query_image_path))
    image = cv2.resize(image, (800, 600))
    cv2.imwrite(os.path.join(directory, query_image_path), image)

sift = cv2.SIFT_create()
```

```

for query_image_path in query_images:

    query_image = cv2.imread(os.path.join(directory, query_image_path),
cv2.IMREAD_GRAYSCALE)

    query_keypoints, query_descriptors = sift.detectAndCompute(query_image,
None)

    best_match_image = None
    best_match_inliers = 0

    for database_image_path in database_images:

        database_image = cv2.imread(os.path.join(directory, "Database",
database_image_path), cv2.IMREAD_GRAYSCALE)

        database_keypoints, database_descriptors =
sift.detectAndCompute(database_image, None)

        inliers = find_best_match(query_descriptors, database_descriptors,
query_keypoints, database_keypoints)

        if inliers > best_match_inliers:
            best_match_inliers = inliers
            best_match_image = database_image_path

    best_match = cv2.imread(os.path.join(directory, "Database",
best_match_image))
    cv2.imshow("Query Image", query_image)
    cv2.imshow("Best Match", best_match)
    cv2.waitKey(0)

cv2.destroyAllWindows()

```

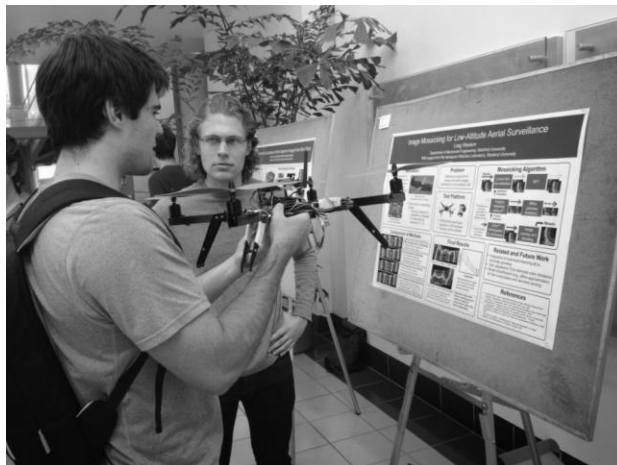


Image Mosaicking for Low-Altitude Aerial Surveillance

Craig Western
Department of Mechanical Engineering, Stanford University
With support from the Aerospace Robotics Laboratory, Stanford University

Motivation

- Growing demand for aerial surveillance and mapping in emergency response, agriculture, and military applications
- Applications in v-dar-based robotic sensing

Problem

Develop an algorithm to mosaic and blend images gathered by a low-altitude UAV

Test Platform

- 3DRobotics AeroVee
- GoPro HERO2

Mosaicking Algorithm

Comparison of Methods

Method	Pros	Cons
Full Homography, No Blending	Simple to implement	Seams are visible
Full Homography, Linear Blending	Seams are less visible	Seams are still visible
Full Homography, Weighted Blending	Seams are almost invisible	Seams are still visible

Final Results

Related and Future Work

- Integration of multi-base blending
- Color correction
- Gain adjustment (e.g., affine approximation)
- Model simplification (e.g., affine approximation)

References

1. [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29] [30] [31] [32] [33] [34] [35] [36] [37] [38] [39] [40] [41] [42] [43] [44] [45] [46] [47] [48] [49] [50] [51] [52] [53] [54] [55] [56] [57] [58] [59] [60] [61] [62] [63] [64] [65] [66] [67] [68] [69] [70] [71] [72] [73] [74] [75] [76] [77] [78] [79] [80] [81] [82] [83] [84] [85] [86] [87] [88] [89] [90] [91] [92] [93] [94] [95] [96] [97] [98] [99] [100]



Foreign Bill Detection, Identification and Currency Conversion Using Sift

Michael Dignan
Christian Elder
Mina Makar

Project Goals

Detect, locate and identify paper currency notes in an image. The value of detected notes is reported to the user in USD.

The Algorithm

Preprocessing: The Golden Samples

Feature Extraction: SIFT

Feature Matching: SIFT

Feature Mosaic: SIFT

Image Overlay and Blending: SIFT

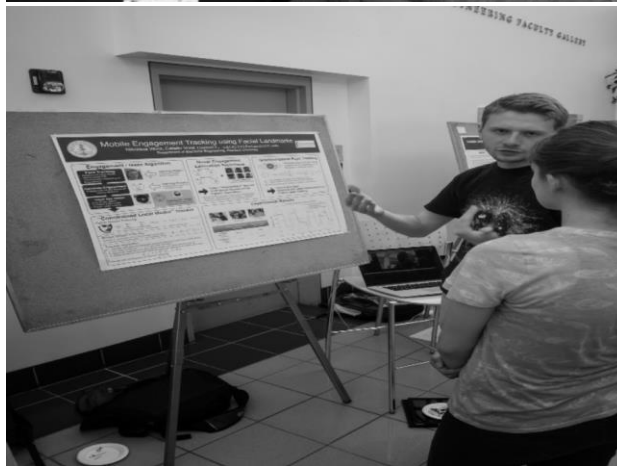
Mosaic: SIFT

Results

Data Flow

Our mobile app uploads a smart phone camera photo to a LAMP server where it is processed. The server returns an image with the matched bills labeled, converted to USD and totalled.

14 bills (front & back) can be detected, distinguished and totalled. Up to 12 bills can be detected at once. Overlapping and similar bills can be properly detected.



Mobile Engagement Tracking using Facial Landmarks

Nikolaus West, Catalin Voss (nwest2, catalin@stanford.edu)
Department of Electrical Engineering, Stanford University

Engagement / Gaze Algorithm

Face Tracking

Compute Engagement

Track eye corner

Aggregate / Adapt

Novel Engagement Estimation Technique

Learn "engagement" across CLM shape modes using logistic regression

Compare testing results, record better engagement

Track by maximizing CLM parameter order

Gradient-based Pupil Tracking

Maximize alignment with image gradient vectors pointing out of the pupil to find eye center

Future direction: Combine with pose information from CLM for gaze tracking

"Constrained Local Model" Tracker

Patch Model Training

Shape Model Training

Tracker Algorithm

Experimental Results

Изработил:

Бојан Рустов (211151)