علی آراسته ۹۶۱۰۱۱۶۵

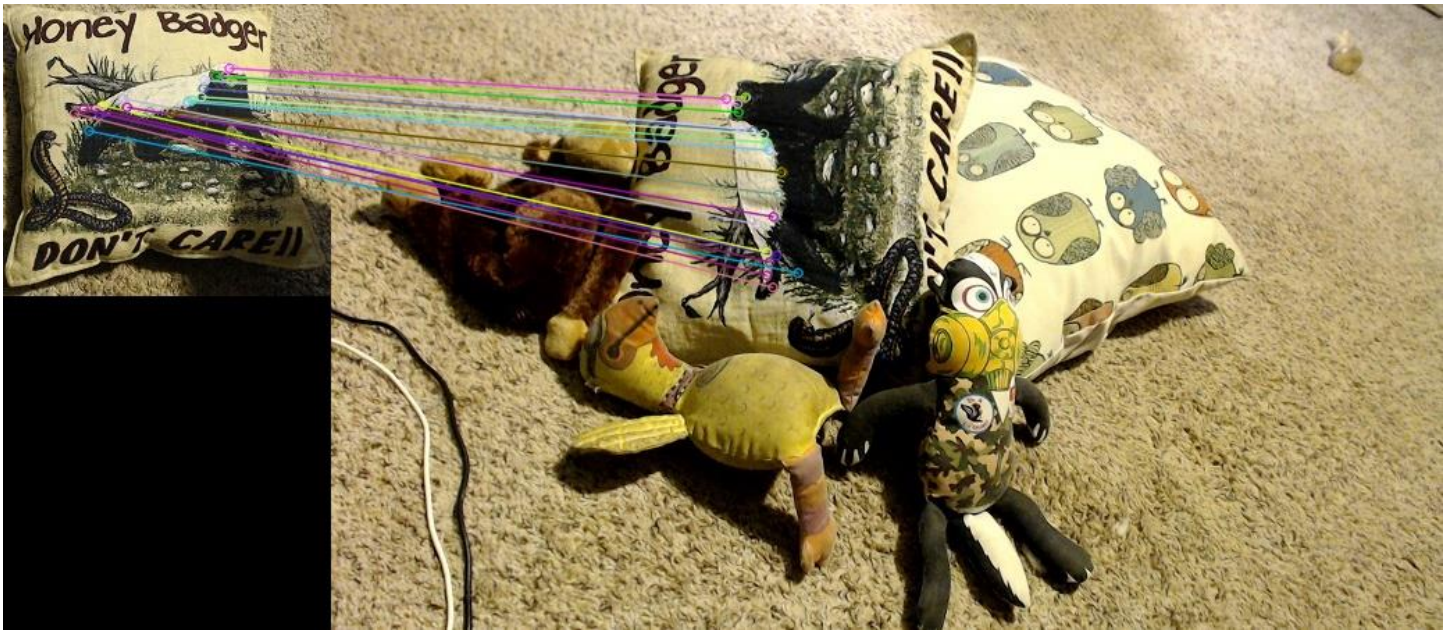
دید کامپیوتری – تمرین چهارم

الف) تمارین کامپیوتری:

سوال ۱:

به علت مشکل موجود در استفاده از روش SIFT، به عنوان جایگزین از روش AKAZE استفاده می‌کنیم. همچنین برای بدست آوردن آن دسته از نقاط کلیدی که بهترین match را دارند، از تابع cv2.BFMatcher استفاده می‌کنیم. سپس از میان match های شناسایی شده بین دو تصویر، ۲۰ جفت نقطه کلیدی‌ای که بیشترین میزان matching را دارند، انتخاب کرده، نمایش می‌دهیم.

روش AKAZE:



روش ORB:

همان طور که از خروجی برنامه مشخص است، در روش AKAZE تمام نقاطی که match شده اند، درست هستند، اما در روش ORB تعداد قابل توجهی matching نادرست صورت گرفته است.

مقایسه را برای ۵۰ جفت نقطه کلیدی‌ای که بیشترین میزان matching را دارند، تکرار می‌کنیم.

روش AKAZE:

روش ORB:

در این حالت در هر دو روش خطاهایی وجود دارد؛ اما به وضوح روش AKAZE عملکرد بهتری دارد و نسبت match های درست به غلط آن، در مقایسه با روش ORB بسیار بهتر است.

از این سوال می‌توان نتیجه گرفت که میزان robust بودن روش AKAZE نسبت به روش ORB بسیار بیشتر است و در نتیجه می‌تواند عملکرد بهتری داشته باشد.

سوال ۲:

برای بدست آوردن نگاشت هموگرافی، حداقل به چهار جفت نقطه یکسان در دو تصویر نیاز داریم. از آن جایی که برای دید از بالا تصویری در اختیار نداریم، با در نظر گرفتن یک تصویر فرضی و شمارش تعداد سرامیک‌های موجود در طول و عرض تصویر، مختصات پیکسلی چهار گوشه از چهار سرامیک در چهار گوشه تصویر را تخمین می‌زنیم. سپس با بدست آوردن مختصات پیکسلی همین نقاط در تصویر اصلی، ماتریس هموگرافی بین تصویر اصلی و تصویر فرضی و با استفاده از آن تصویر دید از بالا را بدست می‌آوریم.

تصویر 1-1:

تصویر 1-2:

سوال ۳:

برای این کار ابتدا با استفاده از دستور cv2.imshow تصویر مورد نظر را در پنجره‌ای نمایش می‌دهیم. سپس با استفاده از دستور cv2.setMouseCallback، یک شنودگر رویدادی‌های موس به این پنجره متصل می‌کنیم. در تابع callback این شنودگر، رویداد اتفاق افتاده را بررسی می‌کنیم. اگر رویداد، فشرده شدن کلیک چپ موس باشد، در محل آن، نقطه‌ای قرمز رنگ رسم می‌کنیم. همچنین مختصات آن را در آرایه‌ای ذخیره می‌کنیم. بعد از انتخاب شدن همه نقاط، با بدست آوردن ماتریس هموگرافی، تصویر 3-2 را جایگزین تابلو تصویر اصلی می‌کنیم.

ترتیب انتخاب نقاط باید به صورت بالا چپ، پایین چپ، پایین راست و بالا راست باشد.

تصویر 1-2:

تصویر 2-2:

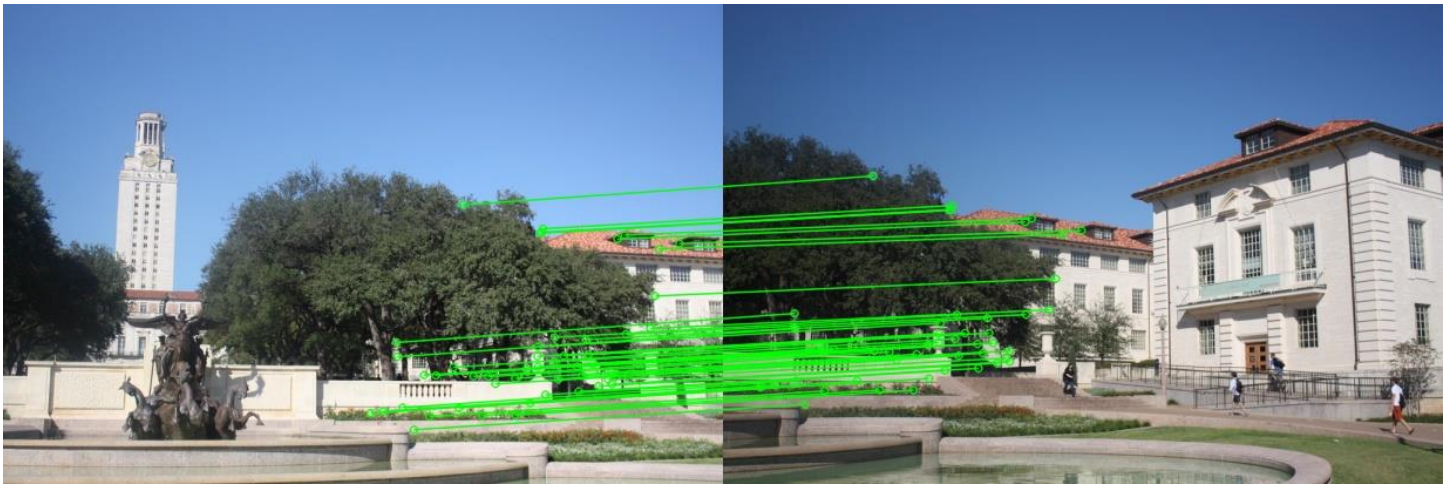
سوال ۴:

ابتدا با استفاده از روش AKAZE نقاط کلیدی هر دوتصویر را بدست می‌آوریم.

تصویر 3-1:

تصویر 3-2:

با استفاده از تابع‌های cv2.BFMatcher و knnMatch، به طور خودکار match های مناسب میان نقاط کلیدی دو تصویر را بدست می‌آوریم.



حال با استفاده از این جفت نقاط یکسان در دو تصویر، ماتریس هموگرافی میان دو تصویر را بدست می‌آوریم. سپس با استفاده از دستور cv2.warpPerspective تصویر دو را به مختصات تصویر دوم منتقل کرده، دو تصویر را به یکدیگر متصل می‌کنیم.

الگوریتم RANSAC:

Random sample consensus (RANSAC) algorithm is a learning technique to estimate parameters of a model by random sampling of observed data. Given a dataset whose data elements contain both inliers and outliers, RANSAC uses the voting scheme to find the optimal fitting result. Data elements in the dataset are used to vote for one or multiple models. The implementation of this voting scheme is based on two assumptions: that the noisy features will not vote consistently for any single model (few outliers) and there are enough features to agree on a good model (few missing data).

RANSAC achieves its goal by repeating the following steps:

1. Select a random subset of the original data. Call this subset the hypothetical inliers.

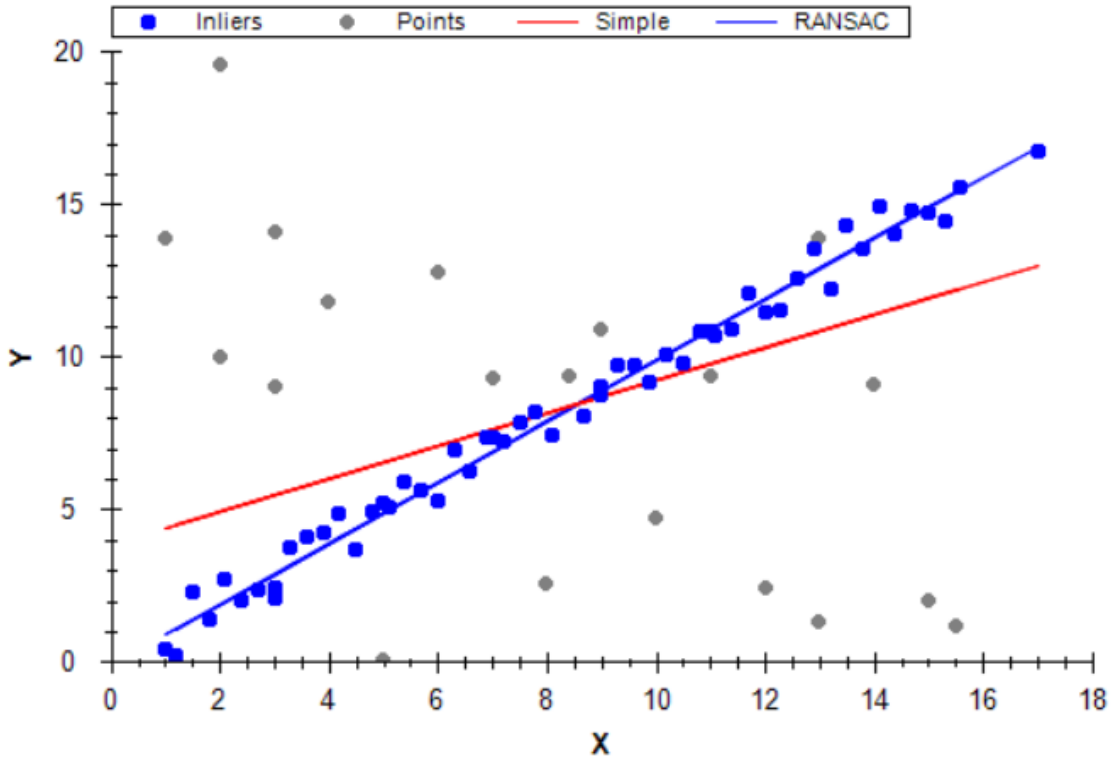
2. A model is fitted to the set of hypothetical inliers.

3. All other data are then tested against the fitted model. Those points that fit the estimated model well, according to some model-specific loss function, are considered as part of the consensus set.

4. The estimated model is reasonably good if sufficiently many points have been classified as part of the consensus set.

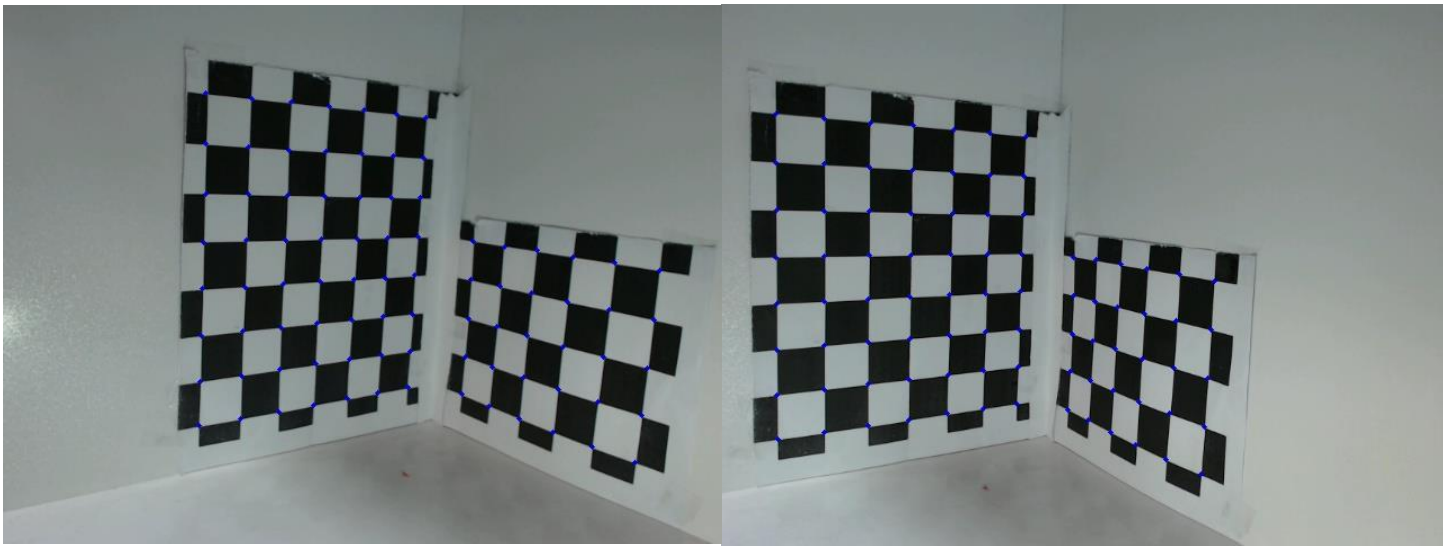
5. Afterwards, the model may be improved by reestimating it, using all members of the consensus set.

This procedure is repeated a fixed number of times, each time producing either a model which is rejected because too few points are part of the consensus set, or a refined model together with a corresponding consensus set size. In the latter case, we keep the refined model if its consensus set is larger than the previously saved model.



سوال ۵:

۱. ابتدا با استفاده از تابع cv2.findChessboardCorners، مختصات نقاطی متناظر میان دو تصویر (نقاط گوشه‌ای صفحه شطرنج) را بدست می‌آوریم.



۲. با استفاده از تابع cv2.findFundamentalMat ماتریس فاندامنتال میان دو تصویر را بدست می‌آوریم.

۳. با استفاده از تابع cv2.computeCorrespondEpilines، در تصویر 3-4، epipolar line متناظر با نقطه (۳۰۵ و ۲۶۵) تصویر 4-4 را بدست می‌آوریم.

۴. به روش مشابه،در هر تصویر epipolar line های متناظر با نقاط گوشه‌ای صفحه شطرنج در تصویر دیگر را رسم می‌کنیم.

ب) تمرین پژوهشی:

Depth estimation is a computer vision task designed to estimate depth from a 2D image. The task requires an input RGB image and outputs a depth image. The depth image includes information about the distance of the objects in the image from the viewpoint, which is usually the camera taking the image. Some of the applications of depth estimation include smoothing blurred parts of an image, better rendering of 3D scenes, self-driving cars, grasping in robotics, robot-assisted surgery, automatic 2D-to-3D conversion in film, and shadow mapping in 3D computer graphics, just to mention a few. A lot of papers aimed at solving these problems using deep learning. We'll look some of them in this report.

Deeper Depth Prediction with Fully Convolutional Residual Networks (IEEE 2016):

This paper proposes a fully convolutional architecture to address the problem of estimating the depth map of a scene given an RGB image. Modeling of the ambiguous mapping between monocular images and depth maps is done via residual learning. The reverse Huber loss is used for optimization. The model runs in real-time on images or videos.

The approach proposed in this paper uses a CNN for depth estimation. The model is fully convolutional and includes efficient residual up-sampling blocks that track high-dimensional regression problems. The first section of the network is based on ResNet50 and is initialized with pre-trained weights. The second part is a sequence of convolutional and unpooling layers that guide the network in learning its upscaling. Dropout is then applied, followed by a final convolution that yields the final prediction. The unpooling layers increase the spatial resolution of feature maps. Unpooling layers are implemented so as to double the size by mapping each entry into the top-left corner of a 2 x 2 kernel. Each such layer is followed by a 5 x 5 convolution. This block is referred to as up-convolution. A simple 3×3 convolution is added after the up-convolution. A projection connection is added from the lower resolution feature map to the result. The authors also recalibrate the up-convolution operation in order to decrease training time of the network by at least 15%. As seen in the figure below, in the top left, the original feature map is unpooled and convolved by a 5 x 5 filter.

Unsupervised Learning of Depth and Ego-Motion from Video (CVPR 2017):

The authors present an unsupervised learning framework for the task of monocular depth and camera motion estimation from unstructured video sequences. The method uses single-view depth and multi-view pose networks. Loss is based on warping nearby views to the target using the computed depth and pose.

The authors propose a framework for jointly training a single-view depth CNN and a camera pose estimation CNN from unlabeled video sequences. The supervision pipeline is based on view synthesis. The depth network takes the target view as the input and outputs a per-pixel depth map. A target view can be synthesized given per-pixel depth in an image and the pose & visibility in a nearby view. This synthesis can be implemented in a fully differentiable manner with CNNs as the geometry and pose estimation modules. The authors adopt the DispNet architecture, which is an encoder-decoder design with skip connections and multi-scale side predictions. A ReLU activation follows all convolution layers except the prediction ones. The target view concatenated with all the source views forms the input to the pose estimation network. The output is the relative pose between the target view and each of the source views. The network is made up of 7 stride-2 convolutions followed by a 1 x 1 convolution with $6 * (N - 1)$ output channels. These correspond to 3 Euler angles and 3-D translation for each source. The global average is applied to aggregate predictions at all spatial locations. Apart from the last convolution layer, where a nonlinear activation is applied, all the others are followed by a ReLU activation function. The explainability prediction network shares the first five feature encoding layers with the pose network. This is followed by 5 deconvolution layers with multi-scale side predictions. Apart from the prediction layers, all the conv/deconv layers are followed by ReLU.

Unsupervised Monocular Depth Estimation with Left-Right Consistency (CVPR 2017):

This paper proposes a convolutional neural network that's trained to perform single image depth estimation without ground-truth depth data. The authors propose a network architecture that performs end-to-end unsupervised monocular depth estimation with a training loss that enforces left-right depth consistency inside the network.

The network estimates depth by inferring disparities that warp the left image to match the right one. The left input image is used to infer the left-to-right and right-to-left disparities. The network generates the predicted image with backward mapping using a bilinear sampler. This results in a fully differentiable image formation model. The convolutional architecture is inspired by DispNet. It's made up of two parts—an encoder and a decoder. The decoder uses skip connections from the encoder's activation blocks to resolve higher resolution details. The network predicts two disparity maps — left-to-right and right-to-left. In the training process, the network generates an image by sampling pixels from the opposite stereo image. The image formation model uses the image sampler from the spatial transformer network (STN) to sample the input image using a disparity map. The bilinear sample used is locally differentiable.

Unsupervised Learning of Depth and Ego-Motion from Monocular Video Using 3D Geometric Constraints (2018):

The authors propose a method for unsupervised learning of depth and ego-motion from single-camera videos. It takes into consideration the inferred 3D geometry of the whole scene and enforces the consistency of the estimated 3D point clouds and ego-motion across consecutive frames. A backpropagation algorithm is used for aligning 3D structures. The model is tested on the KITTI dataset and a video dataset captured on a mobile phone camera.

Learning depth in an unsupervised manner relies on the existence of ego-motion in the video. The network produces a single-view depth estimate given two consecutive frames from a video. An ego-motion estimate is also produced from the pair. Supervision for the training model is achieved by requiring the depth and ego-motion estimates from adjacent frames to be consistent. The authors propose a loss that penalizes inconsistencies in the estimated depth without relying on backpropagation via image reconstruction.

Depth Prediction Without the Sensors: Leveraging Structure for Unsupervised Learning from Monocular Videos (AAAI 2019):

This paper is concerned with the task of unsupervised learning of scene depth and robot ego-motion, where supervision is provided by monocular videos. This is done by introducing geometric structure in the learning process. It involves modeling the scene and the individual objects, camera ego-motion and object motions learned from monocular video inputs. The authors also introduce an online refinement method.

The authors introduce an object motion model that shares the same architecture as the ego-motion network. It is, however, specialized for predicting motions of individual objects in 3D. It takes an RGB image sequence as input. It's complemented by pre-computed instance segmentation masks. The work of the motion model is to learn to predict the transformation vectors per object in 3D space. This creates the observed object appearance in the respective target frame.