In The Name Of God

Report Of HW5 Computer Vision

Tina KhezrEsmaeilzadeh

96101595

الف ) تمارین کامپیوتری

۱) ابتدا با استفاده از cv2.ORB_create() ،الگوی descriptor مربوط به روش orb را ساخته و سپس با استفاده از روش‌هایی نظیر orb.detectAndCompute(img1,None) ، descriptor های مربوط به تصویر 1 و نیز تصویر 2 را ساخته و سپس با استفاده از

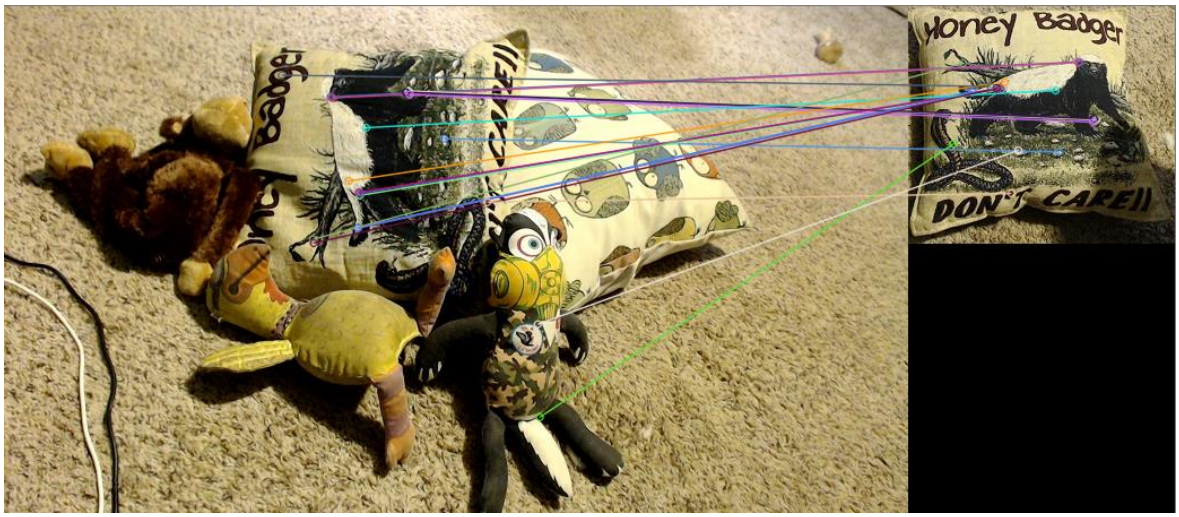
cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True) یک descriptor می‌سازیم که با تابع bf.match(des1,des2) نقاط کلیدی دو تصویر را به هم وصل می‌کند . حال با استفاده از :

```
matches = sorted(matches, key = lambda x:x.distance)

# Draw first 10 matches.

img3 = cv2.drawMatches(img1,kp1,img2,kp2,matches[:20],None, flags=2)
```

20 تا match اول را در دو عکس ترسیم می‌کنیم که به شکل زیر می‌شود :



کد روش SIFT نیز نوشته و اعمال شد اما به علت این که در ورژن opencv من موجود نبود ، قادر به اعمال روی تصاویر داده شده نبودم. اما مراحل نوشتن هر descriptor ای به طور کلی به شکل بالا است که ابتدا باید با استفاده از تابع خاص آن descriptor آن را طراحی کرد. که برای sift به عنوان مثال بتابع به شکل SIFT خواهد بود.

به علت اعمال نشدن متد SIFT از متد cv2.AKAZE_create() استفاده کرده و سپس با استفاده از توابع مشابه بالا به نتیجه‌ی زیر در رابطه با عکس‌های داده شده رسیدیم.

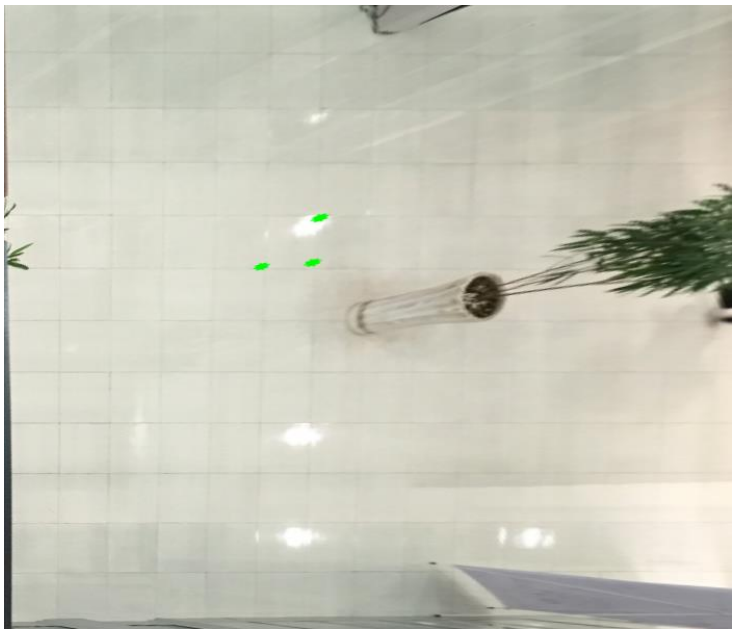

روش AKAZE در این مقاله موفق به تشخیص 150 نقطه‌ی کلیدی و روش ORB موفق به تشخیص 141 نقطه ی کلیدی در عکس‌ها شد. با استفاده از مقاله ای در باره ی مقایسه‌ی دو روش ORB و akaze می‌توان گفت که :

در کل می‌توان گفت که در روش akaze عموما تعداد key point های تشخیص داده شده بیش‌تر و دقت آن ها نیز بیش‌تر خواهد بود ، فلذا خط های match شده در این روش دقیق‌تر خواهند شد . هر دو روش نسبت به rotation و scale ، invariant هستند اما روش akaze بیش‌تر invariant خواهد بود.

به نظر می‌رسد با توجه به موارد تئوری روشakaze از روش SIFT هم بهتر و دقیق تر باشد و invariant بودن نسبت به rotation و scale بیش‌تری در آن دیده می‌شود.

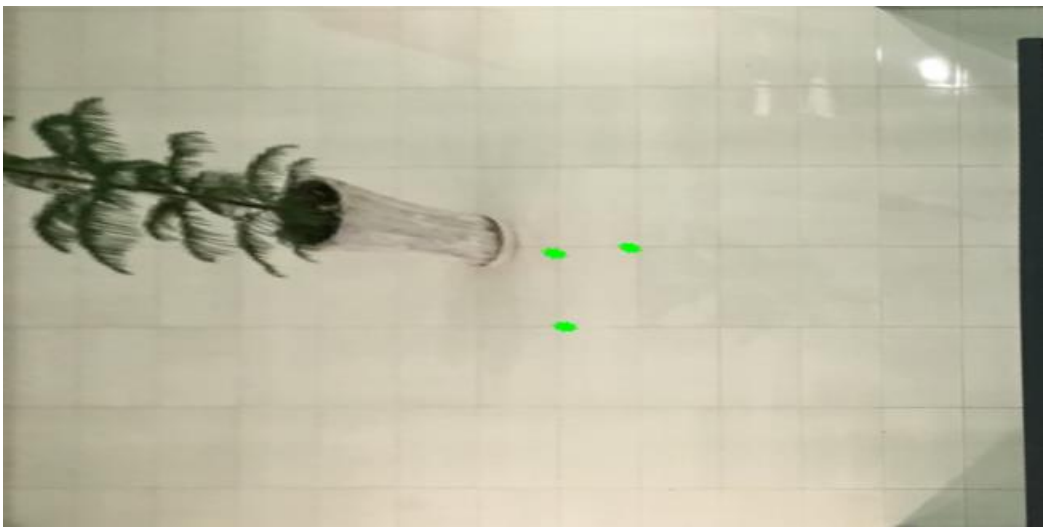
۲) در این سوال باید چند نقطه از تصویر انتخاب شودو با استفاده از گوشه های مربع فرضی تصاویر از دید بالا و بدون perspective به دست آید. برای به دست آوردن چهار نقطه

برای تصویر اول نتیجه‌ی تقریبی زیر به دست آمد:



که مشاهده می‌شود که در باره ی سرامیک ها و سطوح هم عمق با آن به خوبی به تصویر بدون perspective ای خواهیم رسید.

تصویر دوم نیز نتیجه ای به شکل زیر دارد.



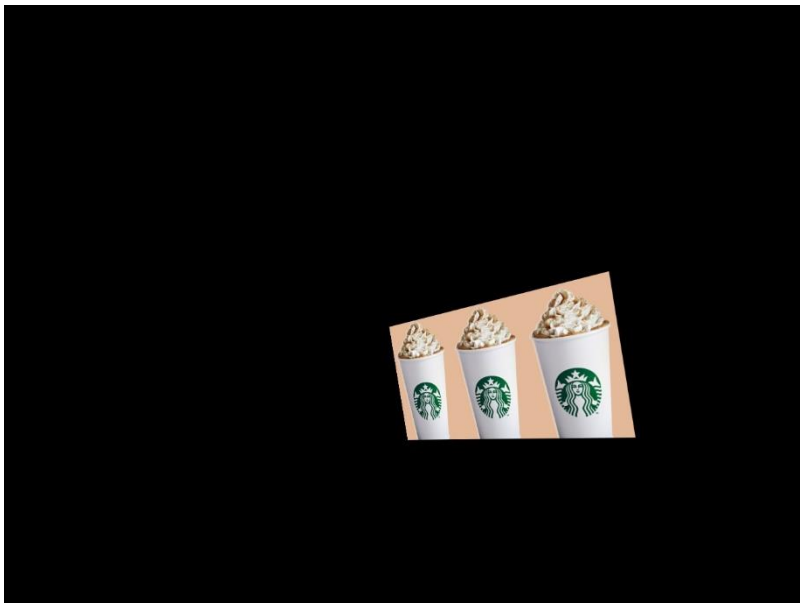
۳ ) ابتدا تصویر خیابان را خوانده و با استفاده از mouse_handler چهار نقطه ی گوشه ای تابلو را از کاربر می‌گیریم . ترتیب گرفتن به صورت چپ بالا ، راست بالا ، راست پایین و در نهایت چپ پایین خواهد بود . بعد زا گزفتن نقاط ، آن ها را با دایره مشخص می‌کنیم. حال عکس خود تابلو را می‌خوانیم. چهار نقطه ای که باید در این جا تطبیق داده شوند ، چهار نقطه‌ی انتهایی تصویر خواهند بود که ان ها را به صورت زیر save می‌کنیم.

```
pts_Tablo = np.array(
                    [
                     [0,0],
                     [size[0] - 1, 0],
                     [size[0] - 1, size[1] -1],
                     [0, size[1] - 1 ]
                     ], dtype=float
                    )
```

بعد از این کار ، با استفاده از h, status = cv2.findHomography( pts_Tablo, pts_Street) هموگرافی میان دو تصویر را می‌یابیم و سپس با استفاده از

```
src_warped = cv2.warpPerspective(im_Tablo, h,
(im_Street.shape[1],im_Street.shape[0]))
```

تصویر perspective شده ی تابلو با به دست می‌آوریم که به شکل زیر خواهد بود :



حال نوبت ادغام دو تصویر است . اگر الان تصویر خیابان و این تصویر warp شده را با هم جمع کنیم ، به نتیجه ی مطلوبی نمی‌رسیم چرا که در این حالت ، مقادیر پیکسلی تصویر تابلو با مقادیر پیکسلی تصویر خیابان در قسمت تابلو جمع می_شوند ، پس با استفاده از یک حلقه‌ی for ، آن ها را ابتدا صفر کرده و سپس تصویر تابلو را بر روی آن می‌نشانیم.

نتیجه برای تصویر اول به شکل زیر است :

پس از اعمال بر تصویر 1-2 نتیجه‌ی زیر به دست آمد :


۴ )

ابتدا به روش akaze نقاط کلیدی دو تصویر را استخراج می‌کنیم. سپس با استفاده از cv2.BFMatcher() ، نقاط کلیدی دو تصویر را match می‌کنیم.

حال با استفاده از کد زیر :

```
matcher = cv2.DescriptorMatcher_create("BruteForce")

rawMatches = matcher.knnMatch(featuresA, featuresB, 2)
```

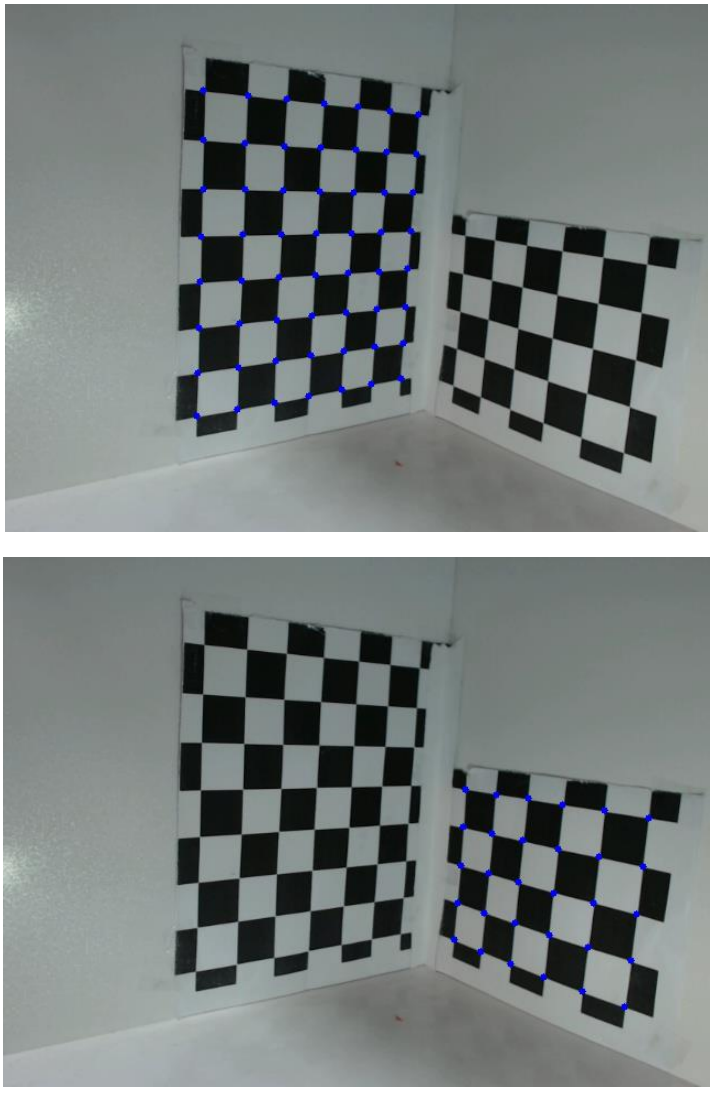
rawMatches را یافته و سپس با استفاده از cv2.findHomgraphy() ، هموگرافی میان دو تصویر را انتخاب می-کنیم . حال که هموگرافی میان دو تصویر یافت شده ، با استفاده از warpperspective آن یکی تصویر را warp کرده تا بتوان به صورت panorama در کنار تصویر یک قرار داد و یک تصویر panorama ساخت.

تصویر panorama به شکل زیر خواهد بود :

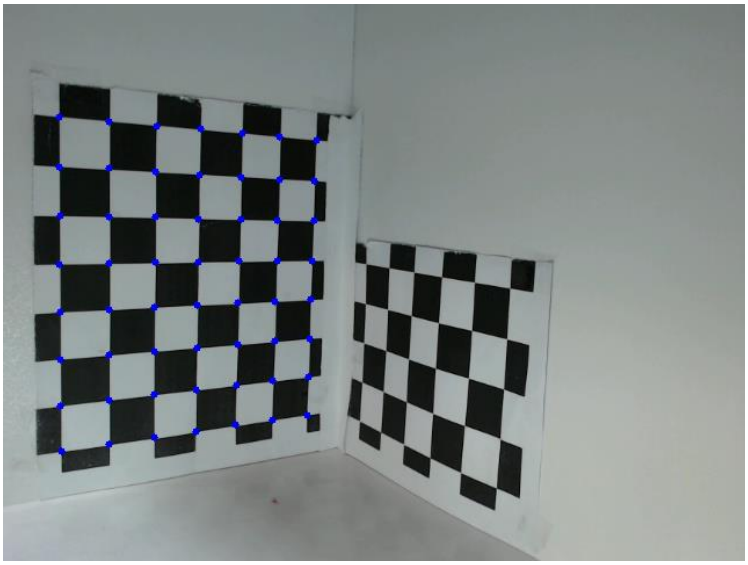
۵ )

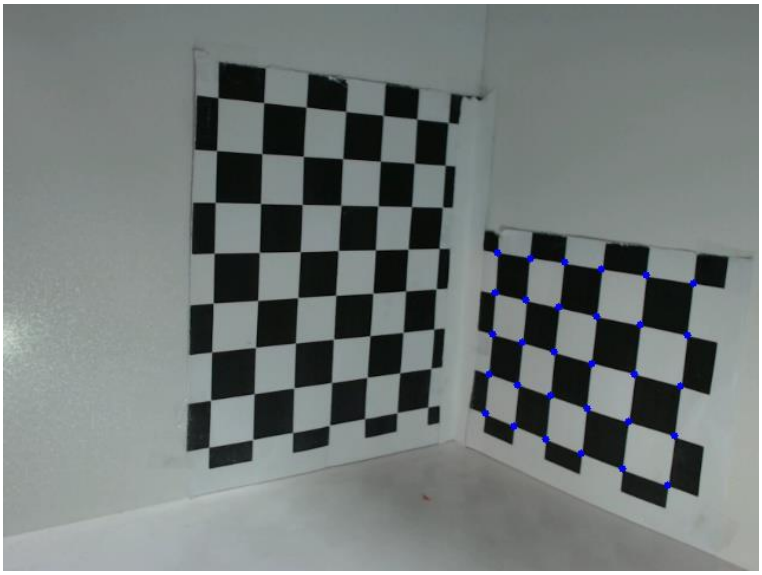
قسمت اول :

ابتدا با استفاده از cv2.findChessboardCorners() که این که pattern صفحه‌ی شطرنجی ما چند در چند است را گرفته و سپس نقاط آن را به ما می‌دهد استفاده کرده و برای صفحه‌های شطرنجی سمت راست و چپ عکس‌های 4-1 و نیز 2-4 این کار را انجام می‌دهیم . به نتایج زیر دست می‌یابیم :



عکس 1-4





عکس 2-4

قسمت دوم :

با استفاده از دستور cv2.findFundamentalMat(corners1, corners2, cv2.FM_LMEDS)

ماتریس فاندامنتال میان دو عکس را نوشته و در workspace ذخیره می‌کنیم تا بتوان در سری های آینده از آن استفاده کرد .

ماتریس فاندامنتال به صورت زیر خواهد بود :

```
array([[-1.05809247e-07,  1.56706755e-06,  2.31028802e-03],
       [-8.30455667e-07,  2.58398172e-06, -2.82059021e-02],
       [-2.55089771e-03,  2.63117048e-02,  1.00000000e+00]])
```

قسمت سوم :

نقطه‌ی داده شده را در نظر می‌گیریم. حال به کمک دستور

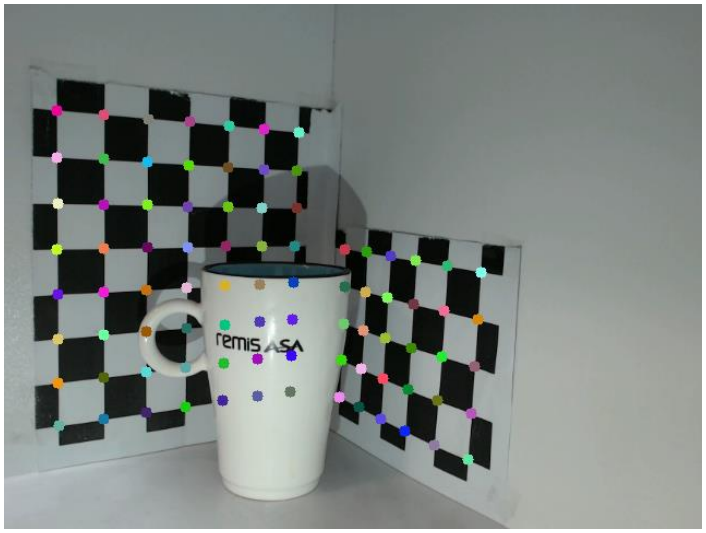
cv2.computeCorrespondEpilines( np.array([265,305]).reshape(-1,1,2), 2, F)

epilines متناظر با نقطه‌ی مذکور را می‌یابیم و با استفاده از تابع drawOneline ، آن را رسم می‌کنیم.

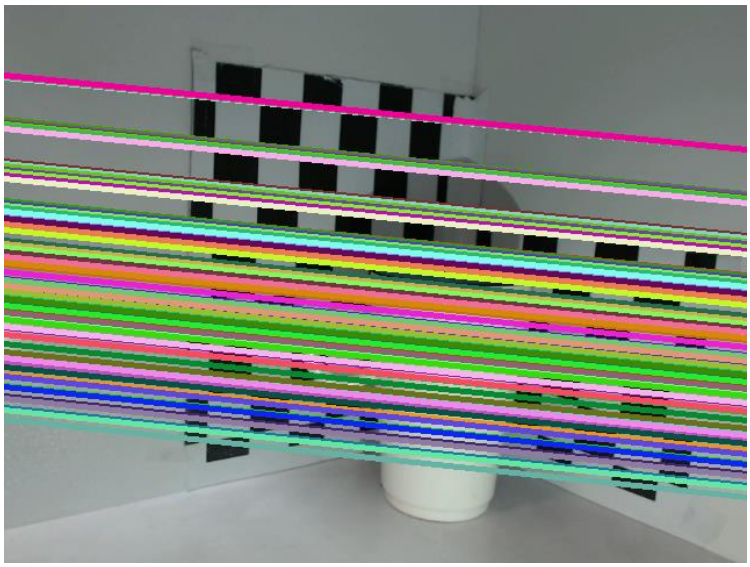

نقطه‌ی داده شده


خط متناظر با نقطه‌ی داده شده

۴ ) حال نوبت به آن رسیده است که خط های متناظر با نقاط کلیدی دو تصویر ، که همان گوشه هاس نقاط شطرنجی هستند ، پیدا شوند. برای این کار، با استفاده از تابع drawSeveralLines که در یک loop برای تمام نقاط داده شده مراحل بالا را اجرا می‌کند ، خط های متناظر با keypoint های سوال یک را می‌کشیم.

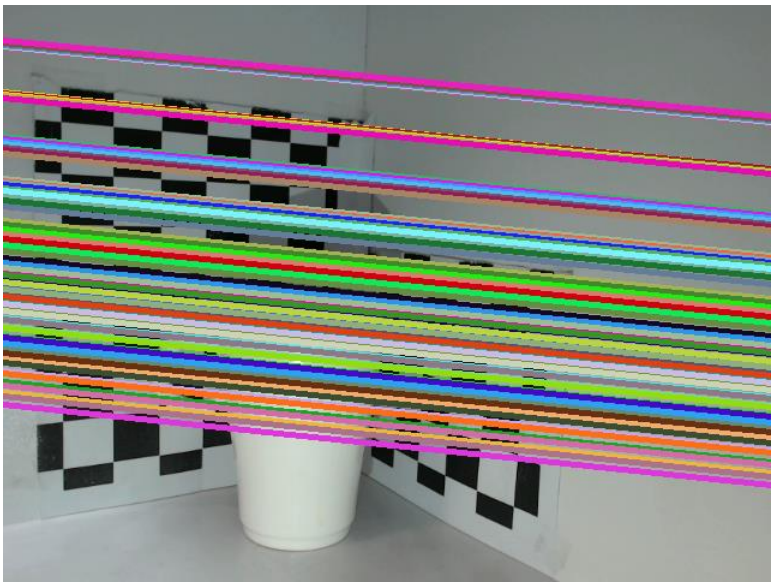


نقاط کلیدی تصویر اول

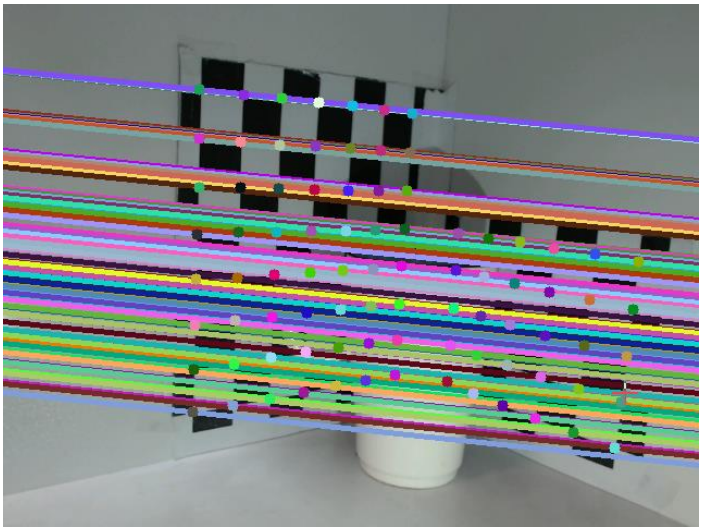


خطوط متناظر با نقاط کلیدی تصویر اول در تصویر دوم

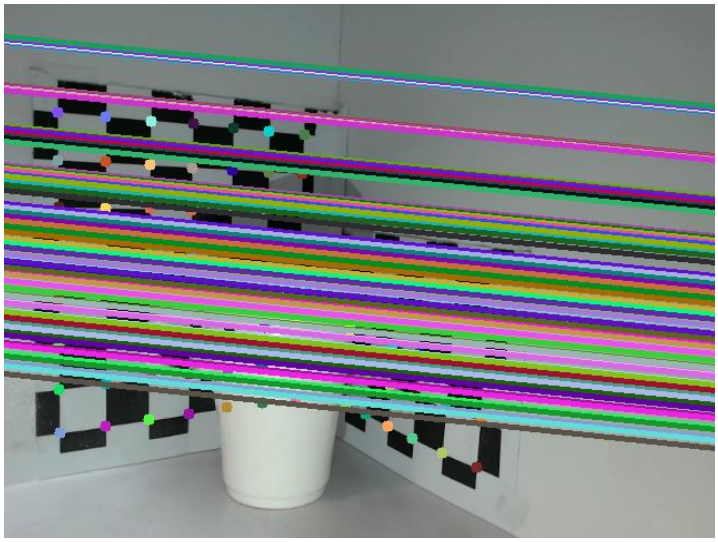

نقاط کلیدی تصویر دوم



خطوط متناظر با نقاط کلیدی تصویر دوم در تصویر اول



خطوط متناظر برای تمام نقاط گوشه ای به همراه نقاط گوشه ای تصویر یک



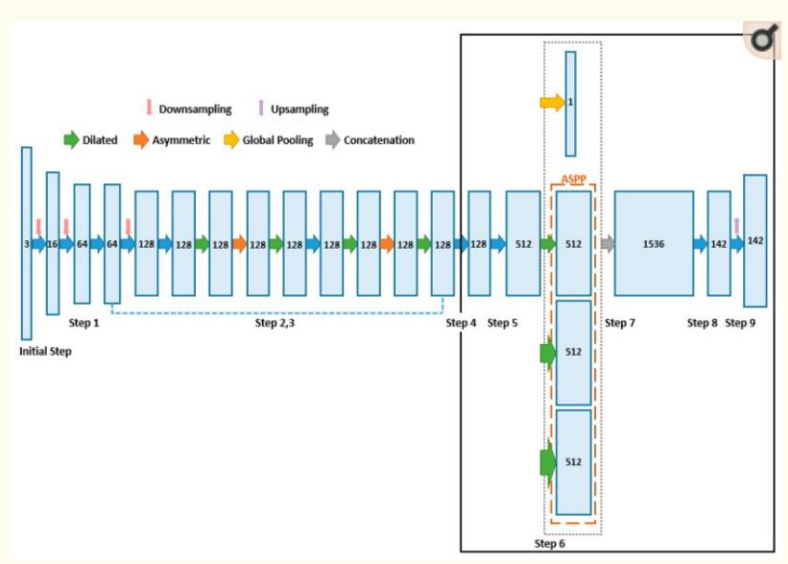
خطوط متناظر برای تمام نقاط گوشه ای به همراه نقاط گوشه ای تصویر دو

## ب ) تمرین پژوهشی

تخمین عمق یک کار بینایی در کامپیوتر است که برای تخمین عمق از یک تصویر2D طراحی شده است. این کار به یک تصویر RGB ورودی نیاز دارد و از یک تصویر عمق خارج می کند. تصویر عمق شامل اطلاعاتی درباره فاصله اشیاء در تصویر از نظرviewpoint است ، که معمولاً دوربین در حال گرفتن تصویر است.

برخی از کاربردهای تخمین عمق عبارتند از: صاف کردن قسمت های تاری یک تصویر ، ارائه بهتر صحنه های سه بعدی ، اتومبیل های رانندگی ، گرفتن در روبات ها ، جراحی با کمک ربات ها ، تبدیل خودکار 2D به 3D در فیلم و نقشه برداری سایه در 3D.

در این بخش ، شبکه عصبی مبتنی بر CNN و روند تخمین عمق پیوسته را توصیف می کنیم که از خروجی شبکه عصبی استفاده می کند. شبکه عصبی پیشنهادی از دو بخش تشکیل شده است. در بخش اول ، یک تکنیک استخراج ویژگی برای استخراج ویژگی dense به یک تصویر داده شده اعمال می شود ، و سپس ویژگی تغییر ناپذیری مقیاس با استفاده از spatial pyramid pooling (ASPP) آموزش داده می شود. در بخش دوم ، دامنه مقدار عمق با استفاده از رگرسیون ordinal تا خروجی شبکه عصبی تخمین زده می شود. در نهایت مقدار عمق پیکسل واقعی به میانگین دو برچسب مجاور اختصاص داده می‌شود. در این مقاله، یک مدل شبکه عصبی کارآمد سبک (L-ENet) طراحی شد که معماری آن در شکل زیر با تغییر مدل پایه ENet نشان داده شده است. ENet اصلی دارای یک ساختار رمزگذار-رمزگشایی بود که برای تقسیم بندی semantic تصویر طراحی شده بود. برای به دست آوردن یک ویژگی غنی از یک تصویر ورودی ، یک convolution به طور کلی در یک receptive field گسترده انجام می شود اما ناکارآمد است ، زیرا با افزایش اندازه ی receptive field ، تعداد پارامترهای مورد نیاز برای آموزش نیز افزایش می یابد.




Sparse Ground Truth thechniques Methods :

که در frameworkآموزش به طور وسیعی استفاده می‌شود .

معیار هایی که برای چک کردن عملکرد متد ها استفاده می‌شوند عبارت اند از :

عملکرد real-time، قابل انتقال بودن ، دقت

# Unsupervised Monocular Depth and Ego-motion Learning with Structure and Semantics

رویکرد ارائه شده در این مقاله شامل هر دو روش ساختاری و معناشناسی برای یادگیری یکپارچه بدون نظارت از عمق و ego_motion.در اینجا رویکردی ارائه داده می‌شود که می تواند با مدل سازی حرکت شیء ، صحنه های پویا را است الگوبرداری کند ، و می تواند استراتژی یادگیری خود را به طور اختیاری با یک پالایش آنلاین تطبیق دهد.روش این مقاله قادر به مدل سازی صحنه های پویا با مدل سازی حرکت جسم است و همچنین می تواند با یک روش پالایش آنلاین اختیاری سازگار شود. مدل سازی حرکات شی منحصر به فرد روش را قادر می سازد صحنه های بسیار پویا را اداره کند. این کار با معرفی یک جزء سوم به مدلی انجام می شود که حرکت اشیاء را به صورت سه بعدی پیش بینی می کند. این ساختار از همان ساختار شبکه به عنوان شبکه حرکتی نفس بهره می برد اما برای جدا کردن وزن ها آموزش می دهد. مدل حرکت بردارهای تحول در هر جسم را در فضای سه بعدی پیش بینی می کند. با استفاده از این دوربین ، ظاهر شی مشاهده شده در قاب هدف مربوطه ایجاد می شود. نتیجه نهایی پیچش ترکیبی از پیچ و تاب فرد از اجسام در حال حرکت و حرکت نفس است. حرکت نفس ابتدا با پوشاندن حرکات شیء تصاویر محاسبه می شود.

این روش بر روی توالی سه تصویر RGB متوالی زیر کار می کند

$$(I_1, I_2, I_3) \in \mathbb{R}^{H\times W\times 3}$$

عمق با یادگیری یک تابع عمیق پیش بینی می شود $\theta : \mathbb{R}^{H\times W\times 3} \rightarrow \mathbb{R}^{H\times W}$

منفرد. ه منظور RGBکه یک شبکه عصبی رمزگذار رمزگشایی کاملاً کانولوشنی با لایه عمق متراکم $D_i = \theta(I_i)$ از یک اداره صحنه های بسیار پویا ، ما حرکات اشیاء فردی را مدل می کنیم.

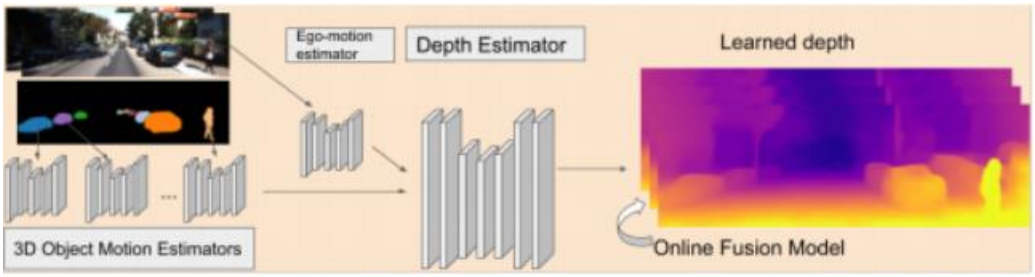



Figure 1: Our method utilizes 3D geometry structure and semantics during learning by modeling motions of individual objects, ego-motion and scene depth in a principled way. Furthermore, a refinement approach adapts the model on the fly in an online fashion.

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

در این مقاله یک معماری کاملاً پیچیده برای حل مسئله برآورد نقشه عمق یک صحنه با توجه به تصویر RGB پیشنهاد شده است. مدل سازی نقشه های مبهم بین تصاویر یکپارچه و نقشه های عمق از طریق یادگیری باقیمانده انجام می شود. از روش Huber معکوس برای بهینه سازی استفاده می شود. این مدل در زمان واقعی بر روی تصاویر یا فیلم ها اجرا می شود.

بخش اول شبکه بر اساس ResNet50 است و با وزنهای از قبل آموزش دیده اولیه سازی می شود. بخش دوم دنباله ای از لایه های محکم و ناپایدار است که شبکه را در یادگیری صعود آن هدایت می کند. Dropout سپس اعمال می شود و به دنبال آن نتیجه گیری نهایی انجام می شود که پیش بینی نهایی را ارائه می دهد.

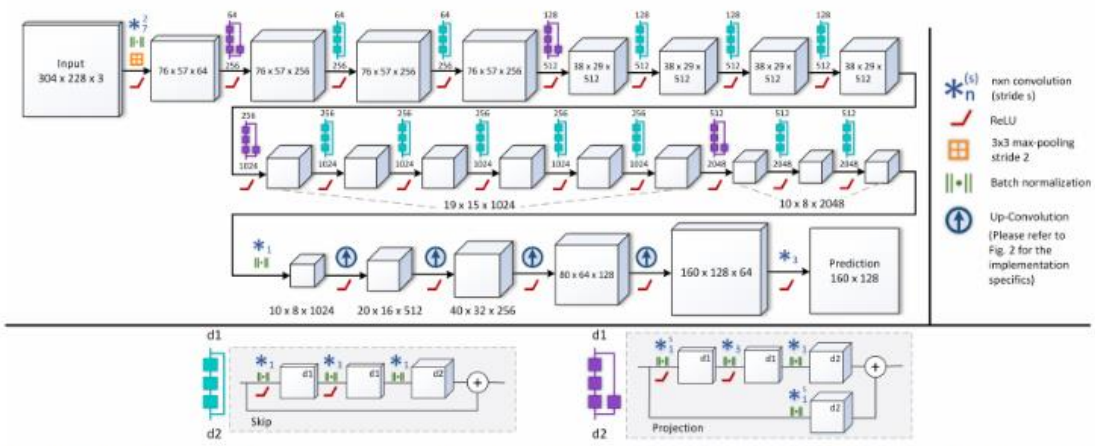



Figure 1. **Network architecture.** The proposed architecture builds upon ResNet-50. We replace the fully-connected layer, which was part of the original architecture, with our novel up-sampling blocks, yielding an output of roughly half the input resolution

لایه های unpooling باعث افزایش وضوح مکانی نقشه های ویژگی می شوند. لایه های باز نشده به گونه ای اجرا می شوند که با نقشه هر ورودی به گوشه بالا سمت چپ یک هسته ۲ × ۲ اندازه را دو برابر می کنند. هر لایه با نتیجه گیری ۵ 5 x دنبال می شود. از این بلوک به عنوان پیچیدگی بالا یاد می شود. نتیجه گیری ۳ × ۳ ساده پس از نتیجه گیری اضافه می شود. اتصال طرح ریزی از نقشه ویژگی با وضوح پایین تر به نتیجه اضافه می شود.