



دانشگاه صنعتی امیر کبیر
(پلی تکنیک تهران)

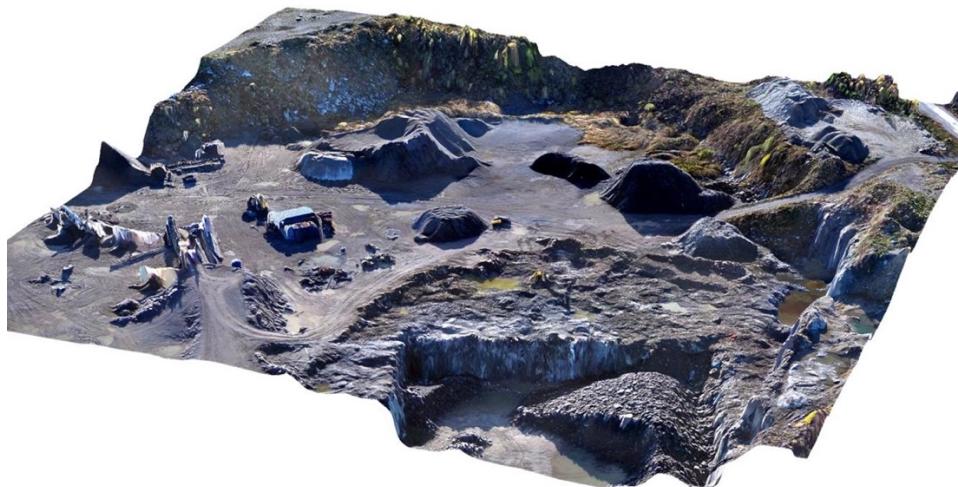
درس پیوپی کامپیوونر سه بعدی

استاد درس جناب آقای دکتر جوانمردی

(تکلیف اول)

محسن عبادپور | ۴۰۰۱۳۱۰۸۰ | m.ebadpour@aut.ac.ir

نیمسال دوم سال تحصیلی ۱۴۰۱-۱۴۰۲



فهرست پاسخ ها

فهرست پاسخ ها

۳	بخش اول: آشنایی با ویژگی های تبدیلات در فضای P^2
۳	الف) اثبات هموگرافی
۴	ب) ثابت بودن VANISHING LINE تحت تبدیل AFFINE
۴	ج) محل VANISHING LINE
۵	د) یافتن یک تبدیل H_P برای انتقال VANISHING LINE ایده آل به VANISHING LINE بدست آمده
۷	۵) اصلاح تصویر بصورت SIMILARITY
۹	بخش دوم: تعیین پارامتر های داخلی یک دوربین با یک تصویر
۹	الف) بدست آوردن ماتریس W با سه نقطه $VANISHING POINT$
۱۱	ب) تجزیه CHOLESKY
۱۱	ج-امتیازی) اثبات منطبق بودن محل تلاقی میانه ها و نقطه PRINCIPAL
۱۳	بخش سوم: تعیین موقعیت دوربین
۱۲	الف) فرآیند کلی الگوریتم CPE
۱۴	ب) تعیین فاصله و جهت خودرو

آشنایی با ویژگی‌های تبدیلات در فضای P^2

بخش اول: آشنایی با ویژگی‌های تبدیلات در فضای P^2

الف) اثبات هموگرافی

برای بدست آوردن ماتریس هموگرافی لازم است که ابتدا ماتریس دوربین را محاسبه و بدست آوریم. در کلاس درس و در اسلایدها ملاحظه کردیم که مدل یک دوربین معمولی نظیر pin hole یک نقطه در فضای واقعیت سه بعدی را به یک نقطه‌ی دو بعدی در صفحه‌ی تصویر دوربین نگاشت می‌کند. حال با فرض اینکه صحنه‌ی مورد نظر ما یک تصویر است و در موقعیت $(X, Y, 0)$ قرار دارد، می‌توانیم ماتریس دوربین معمولی خود را به صورت زیر آپدیت کنیم که در نتیجه ماتریس حاصل یک ماتریس هموگرافی است (ماتریس با ابعاد 3×3). تصویر زیر از اینترنت انتخاب و آورده شده است:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} \\ C_{21} & C_{22} & C_{23} & C_{24} \\ C_{31} & C_{32} & C_{33} & C_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$
$$= \begin{bmatrix} C_{11} & C_{12} & \cancel{C_{13}} & C_{14} \\ C_{21} & C_{22} & \cancel{C_{23}} & C_{24} \\ C_{31} & C_{32} & \cancel{C_{33}} & C_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ \cancel{Z} \\ 1 \end{bmatrix} \quad H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix}$$
$$= \begin{bmatrix} C_{11} & C_{12} & C_{14} \\ C_{21} & C_{22} & C_{24} \\ C_{31} & C_{32} & C_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

مسئله‌ی فوق بیان می‌کند که تغییر از یک نمایی به نمای دیگر که projective بوده و سمت راست دنیای واقعی و سمت چپ فضای دو بعدی تصویر است، به ازای یک صحنه‌ی یکسان از دنیای واقعی سه بعدی و با حذف بعد سوم و صفر شدن درایه‌های متناظر، تبدیلی از یک صفحه projective به صفحه‌ی دیگری می‌باشد چرا که در نهایت به این رسیده‌ایم که سمت راست دو بعد و سمت چپ نیز دو بعد است و توسط یک ماتریس 3×3 می‌توان این نگاشت از یک صفحه دو بعدی به صفحه‌ی دو بعدی دیگر را انجام داد و از جایی که h_{33} وجود ندارد می‌توانیم آن را یک جا گذاری کنیم.

ب) ثابت بودن vanishing line تحت تبدیل affine

مقصود از vanishing line یک خطی است که همهی vanishing point ها روی آن قرار دارد؛ از طرفی می‌دانیم تبدیل affine تبدیلی است که خطوط موازی را موازی نگه داشته اما تبدیل projective این را لزوماً ندارد. طبق نمایش تبدیل‌های affine و projective که از اینترنت برداشته شده و در زیر قابل مشاهده است؛ اگر vanishing line را با تبدیل affine بدهست آوریم، خود vanishing line مجدد بدهست می‌آید اما اگر آن را با تبدیل projective بدهست آوریم، لزوماً همان خط بدهست نمی‌آید و میتواند مقادیر دیگری حاصل شود لذا میتوان نتیجه گرفت که vanishing line تحت تبدیل affine ثابت است اما تحت یک تبدیل projective به یک خط با مختصات محدود تبدیل می‌شود.

$$\begin{array}{l} \text{A square transforms to:} \\ \text{Projective 8dof} \quad \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \\ \text{Affine 6dof} \quad \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \end{array}$$

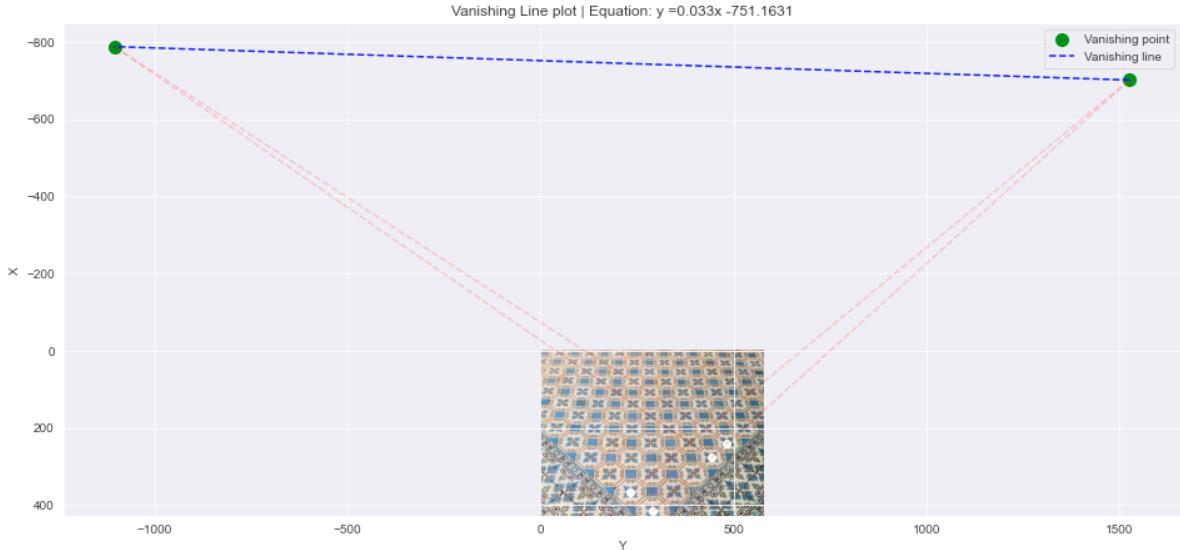
$$\text{Vanishing Line} = [0, 0, 1]^T$$

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} * \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{13} \\ h_{23} \\ h_{33} \end{bmatrix} \text{ for projective and, } \begin{bmatrix} a_{11} & a_{12} & tx \\ a_{21} & a_{22} & ty \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \text{ for affine}$$

ج) محل vanishing line

برای این منظور ابتدا دو جفت خط موازی (در واقعیت) را انتخاب می‌کنیم؛ این کار با انتخاب هشت نقطه بصورت دستی vanishing line انجام شده است. هر جفت خط موازی در یک point همیگر را قطع کنند. بدین صورت ما دو vanishing point خواهیم داشت که با یافتن معادلهی خط (شیب و عرض از مبدأ) بین آن دو میتوان معادلهی vanishing line را نیز بطور دقیق بدهست آورد. تصویر بعدی نقاط انتخاب شده را به همراه vanishing point ها و vanishing line نشان می‌دهد و معادله نیز بصورت بصورتی که در عنوان

درج شده است بدهست می‌آید:



د) یافتن یک تبدیل H برای انتقال vanishing line ایده آل به بدست آمده

اگر بخواهیم نقاط تصویر ورودی را مد نظر قرار دهیم، از اسلاید درس داریم که $x' = Hx$ که x' و x میتوانند در هر فضایی دلخواهی باشند و از طرفی اگر بخواهیم خطوط را مد نظر قرار دهیم داریم که $l' = H^{-T}l$ که همانند مورد مذکور میتوانند در هر فضایی باشند. حال اگر ما l را vanishing line ایده آل در نظر بگیریم و l' را خطی که در قسمت قبل بدست آورده‌ایم (و بطور مشابه نتیجه شود که x' ما تصویر ورودی و x تصویر اصلاح شده است)، میتوانیم تعدادی از عناصر H^{-T} را p نامیده و سه مورد را بصورت مستقیم و مابقی را بصورت پارامتریک بدست بیاوریم که با توجه به اعداد حاصل خواهیم داشت:

$$\begin{bmatrix} 0.33 \\ -1 \\ -751.1631 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \rightarrow p_{13} = 0.033, \quad p_{23} = -1, \quad p_{33} = -751.1631$$

طبق توضیحات فوق، ما بایستی x را بدست آوریم، در نتیجه $x' = H^{-1}x$ را نیاز داریم؛ در عبارت فوق H^{-T} را محاسبه کرده ایم، حال اگر صرفا آن را ترانهاده کنیم، به پاسخ خواهیم رسید:

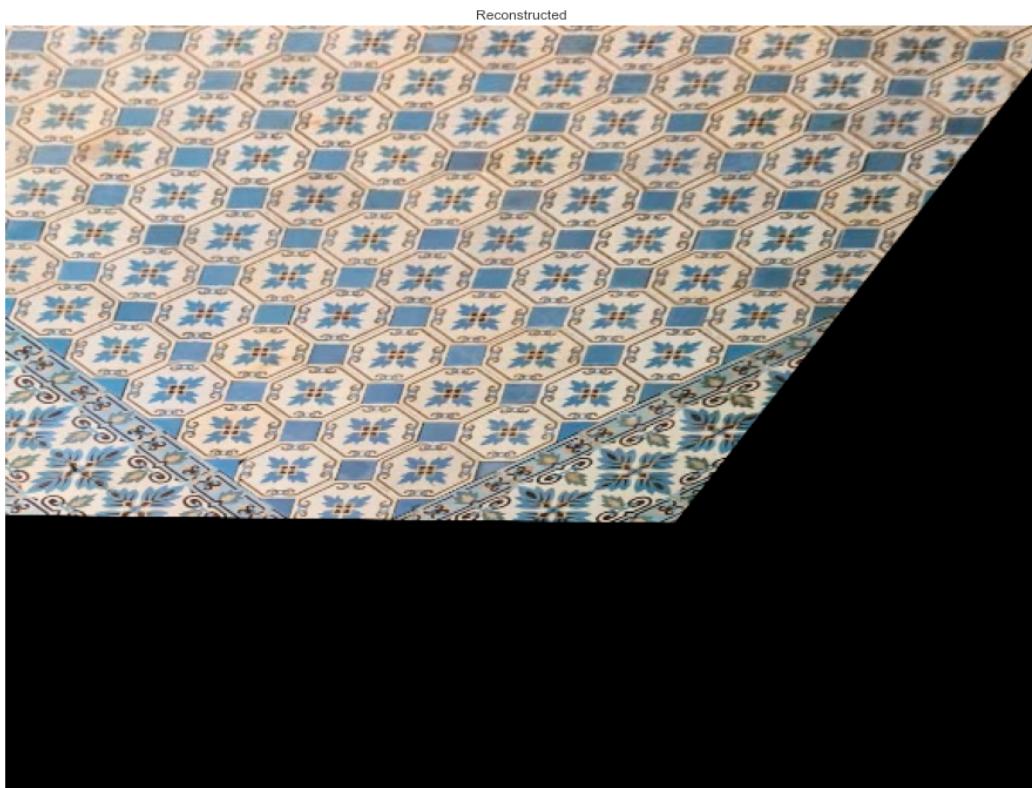
$$\begin{bmatrix} p_{11} & p_{21} & p_{31} \\ p_{12} & p_{22} & p_{32} \\ p_{13} & p_{23} & p_{33} \end{bmatrix} = \begin{bmatrix} p_{11} & p_{21} & p_{31} \\ p_{12} & p_{22} & p_{32} \\ 0.033 & -1 & -751.1631 \end{bmatrix}$$

لذا میتوانیم به عنوان دلخواه، مقادیر نامشخص را تعیین کنیم که بصورت زیر حاصل می‌شود. البته به این نکته توجه شود در ماتریس فوق p_{11} و p_{12} و p_{21} و p_{22} تشکیل دهنده‌ی یک ماتریس rotation هستند و باید رابطه‌ی \sin/\cos رعایت شده باشد و راحت ترین مقدار برای جاگذاری انتخاب شده است و مقادیر دیگر میتواند مورد استفاده قرار گیرد و این است که نشان میدهد یکتا نیست:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0.033 & -1 & -751.1631 \end{bmatrix}$$

همچنین بایستی توجه نمود که در ماتریس تبدیل projective بایستی عنصر سوم در سطر سوم یک شود تا معنی دار شود؛ لذا با استناد به این نکته، سطر سوم را نرمالایز میکنیم که حاصل می‌شود:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -0.00004393 & 0.0013 & 1 \end{bmatrix}$$



۵) اصلاح تصویر بصورت similarity

با دانستن موقعیت چهار نقطه از تصویر و چهار نقطه ای متناظر در واقعیت که تشکیل یک مستطیل می‌دهند، میتوانیم نگاشتی 3×3 مانند H بدست آوریم که در بحث بینایی به این مفهوم Homography نیز گفته می‌شود. هموگرافی ماتریس تبدیلی است که طی آن مختصات نقاط از یک صفحه به مختصات متناظر در صفحه‌ی دیگر نگاشت می‌شود که معمولاً در مباحث پردازش تصویر در تبدیل عکس‌های پرسپکتیو استفاده می‌شود. برای این کار اگر ما بدانیم که مختصات گوشه‌های مستطیل در تصویری که بصورت پرسپکتیو می‌باشد در واقعیت و از رو به رو در چه موقعیتی قرار می‌گیرد میتوانیم ماتریس H را بصورت دقیق و معین بدست آوریم. به عبارتی دیگر، مستطیلی که در واقعیت مستطیل بوده اما در تصویر بخاطر پرسپکتیو بصورت ذوزنقه دیده می‌شود را می‌توان با تبدیلی مانند H بازسازی کرد طوری که مجدد مستطیل از دید رو به رو حاصل گردد.

برای اثبات در نظر می‌گیریم که مختصات دنیای واقعی R بوده و دستگاه دو بعدی آن بصورت (x,y) تعیین می‌شود و مختصات چهار نقطه‌ی A, B, C, D از آن در اختیار است و مختصات تصویر نیز I بوده و بصورت (u,v) تعیین می‌شود و مختصات چهار نقطه‌ی متناظر بترتیب بصورت A', B', C', D' در اختیار است. برای اینکه نشان دهیم نگاشتی مانند H (و مشابهاً معکوس آن) وجود دارد که میتواند تصویر اصلی را بازسازی کند، کافی است وجود و محاسبه‌ی آن را نشان دهیم.

فرض می‌کنیم چنین تبدیل بین R و I وجود دارد. حال برای هر نقطه‌ی دلخواه بایستی این تبدیل بصورت $[u, v, w]^T = H * [x, y, 1]^T$ قابل انجام باشد که این رابطه گویای یک تبدیل خطی است که با یک ماتریس نمایش داده می‌شود و برای یافتن عناصر آن می‌توان دستگاه معادلات خطی ایجاد و با حل آن به خود تبدیل رسید؛ حال برای ساختن این ماتریس داریم که (دلیل تخریب این است که ممکن است در تصویر پیکسل‌ها دقیقاً روی هم نیافتدند و نیاز به درون یابی باشد):

$$[u_1, v_1, 1]^T \approx H * [x_1, y_1, 1]^T$$

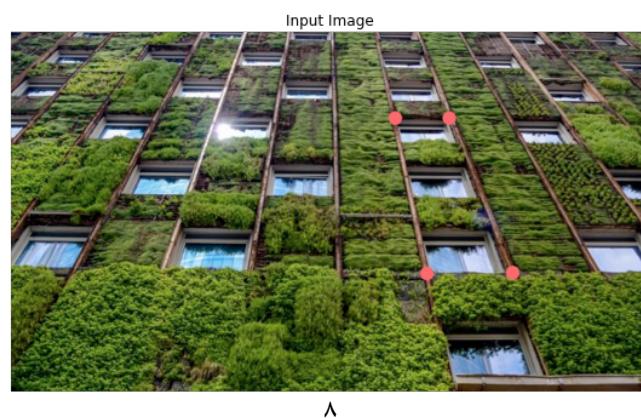
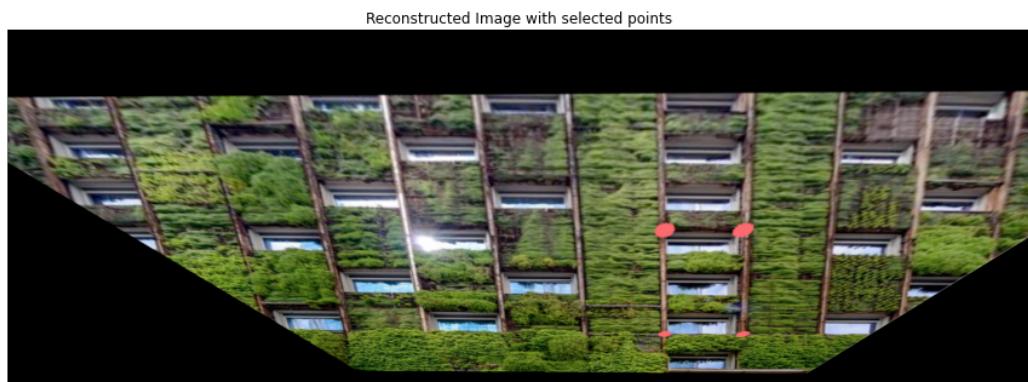
$$[u_2, v_2, 1]^T \approx H * [x_2, y_2, 1]^T$$

$$[u_3, v_3, 1]^T \approx H * [x_3, y_3, 1]^T$$

$$[u_4, v_4, 1]^T \approx H * [x_4, y_4, 1]^T$$

$$H = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \rightarrow \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & 1 \end{bmatrix}$$

همانطور که مشهود است ما هشت متغیر داشته و هشت معادله داریم و با حل آن می‌توانیم ماتریس تبدیل را کامل تعیین کنیم. برای عکس خواسته شده چهار نقطه که در واقعیت یک مستطیل می‌باشد را انتخاب کرده و مختصات چهار نقطه‌ی دلخواه که در تصویر دقیقاً بصورت مستطیل می‌باشد را هدف قرار می‌دهیم (با متغیرهای ver_shift, hor_shift تصویر را انتقال می‌دهیم تا کل تصویر در کادر قابل نمایش باشد) و از تابع findHomography استفاده کرده و ماتریس تبدیل را بدست می‌آوریم و با تابع wrapPerspective تصویر بازسازی شده را بدست خروجی می‌دهیم. بنده برای بازسازی و اینکه کل تصویر در کادر دیده شود، ابعاد تصویر بازسازی شده را resize و scale را نیز بزرگتر کرده ام تا در نمایش بهتر دیده شود. همانطور که میبینیم خطوط موازی با صفحه دوربین ظاهر شده اند.



A

تعیین پارامتر های داخلی یک دوربین با یک تصویر

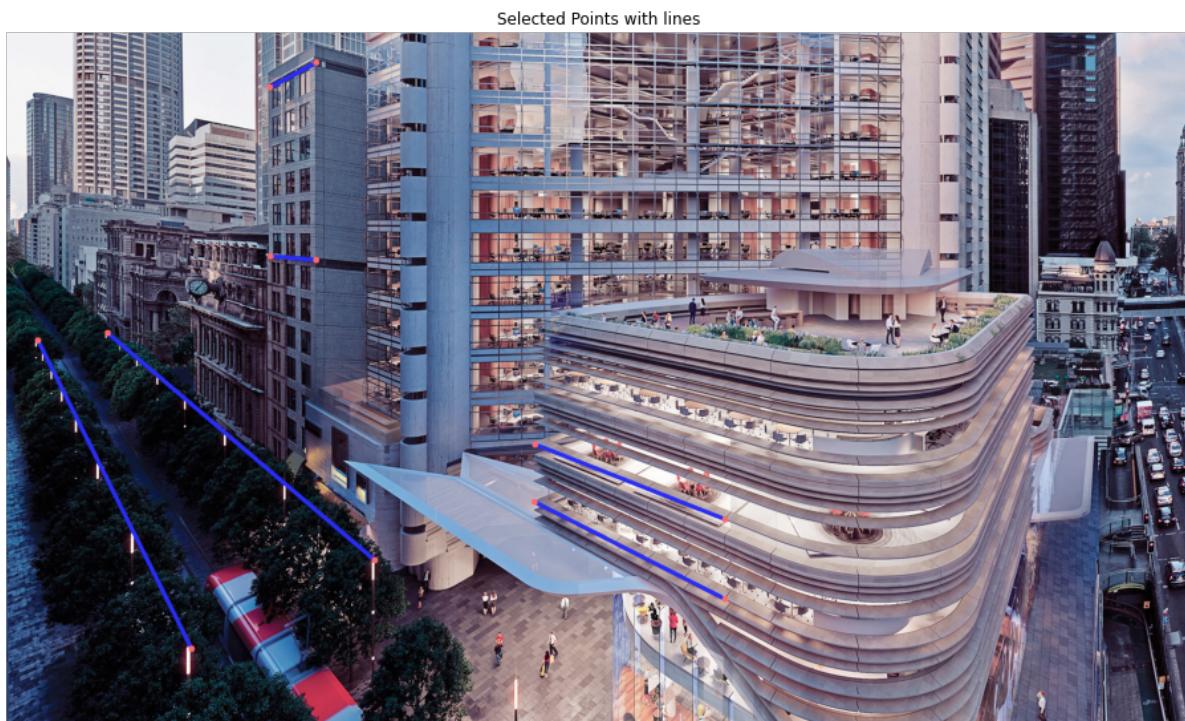
بخش دوم: تعیین پارامتر های داخلی یک دوربین با یک تصویر

الف) بدست آوردن ماتریس W با سه نقطه ای vanishing point

در این قسمت ما می خواهیم با استفاده از سه نقطه ای vanishing point اقدام به یافتن ماتریس W کنیم؛ ماتریس W یک ماتریس 3×3 می باشد که ۹ عنصر دارد. بدلیل صفر بودن skew دو عنصر از ماتریس مذکور برابر با صفر می شود. عنصر آخر ماتریس نیز برابر با یک می باشد و در نتیجه ۶ مجھول باقی می ماند و با تشکیل معادله با سه نقطه ای vanshinhg که هر کدام دو بعد (y,x) دارد میتوانیم همه عناصر ماتریس مذکور را بدست آوریم.

برای اینکه مختصات سه نقطه ای vanishing را داشته باشیم بایستی سه جفت خط موازی که در واقعیت موازی بوده ولی در تصویر موازی با صفحه ای دوربین نیستند را انتخاب کنیم لذا در کل به ۶ خط نیاز داریم. برای نشان دادن هر خط نیز به دو نقطه نیاز داریم تا بتوانیم خط را بدست آوریم و در نتیجه از تصویر بایستی ۱۲ نقطه انتخاب کنیم.

تصویر زیر ۱۲ نقطه ای انتخابی و ۶ خط حاصل را نشان می دهد:



حال همانند بخش یک، نقطه‌ی قطع دو خط موازی را بدست می‌آوریم تا نقطه‌ی vanishing حاصل شود و این کار را برای هر دو خط موازی تکرار می‌کنیم تا سه نقطه‌ی vanishing حاصل شود. این سه نقطه به همراه خطوط و نقاط اولیه در تصویر زیر قابل مشاهده است:



حال دو به دو نقاط vanishing را در نظر گرفته (نقطه‌ی vanishing اول با دوم، اول با سوم و دوم با سوم) و با توجه به رابطه‌ی $v_1^t W v_2 = 0$ ، شش معادله‌ی مدنظر را بدست می‌آوریم و با استفاده از حل svd مقادیر مجهولات را بدست می‌آوریم تا ماتریس W حاصل شود. برای نقاط و خطوط انتخابی، W بصورت زیر حاصل می‌شود:

```
W:
[[ 4.10822608e-05  0.00000000e+00 -1.17852167e-03]
 [ 0.00000000e+00  4.10822608e-05 -7.67308896e-03]
 [-1.17852167e-03 -7.67308896e-03  9.99969866e-01]]
```

ب) تجزیه Cholesky

تجزیه Cholesky تجزیه‌ای است که یک ماتریس نیمه مثبت معین مربعی را دریافت کرده و آن را به بصورت حاصل ضرب دو ماتریس تجزیه می‌کند که این دو ماتریس یکی پایین مثلثی و دیگری بالا مثلثی و ترانهاده‌ی یکدیگر هستند و این تجزیه دقیقاً چیزی است که ما برای یافتن ماتریس K از ماتریس w با توجه به رابطه‌ی $w = K^{-1}w$ به آن نیاز داریم. ابتدا بایستی ماتریس وارون w را محاسبه و سپس آن را به تابع Cholesky از کتابخانه numpy بدھیم تا تجزیه محاسبه و ماتریس K حاصل شود. در این فرآیند و محاسبات ممکن است ماتریس w و وارونش نیمه‌مثبت معین حاصل نشده باشد و از جایی شرط تجزیه Cholesky نیمه مثبت معین بودن ماتریس ورودی می‌باشد، از نیمه مثبت معین بودن آن اطمینان حاصل و آن را به یک ماتریس نیمه مثبت معین تبدیل می‌کنیم؛ نتایج بصورت زیر حاصل می‌شود:

```
Positive semi-definite:  
[[ 0.01004108  0.          -0.00117852]  
 [ 0.          0.01004108 -0.00767309]  
 [-0.00117852 -0.00767309  1.00996987]]  
----  
k:  
[[ 0.1002052   0.          -0.01176108]  
 [ 0.          0.1002052  -0.07657376]  
 [ 0.          0.          1.00198204]]
```

ج-امتیازی) اثبات منطبق بودن محل تلاقی میانه‌ها و نقطه Principal

برای اثبات این که نقطه principal مانند P تصویر در محل برخورد میانه‌های یک مثلث قرار دارد که توسط سه نقطه vanishing ساخته شده است، باید از اصول پرسپکتیو و خصوصیات نقاط vanishing استفاده نماییم. میدانیم خطوط موازی در دنیای واقعی به یک نقطه vanishing در تصویر همگرا می‌شوند. این ویژگی نتیجه‌ای است که از تصویرسازی هندسی یک صحنه سه‌بعدی بر روی صفحه‌ی تصویر دو بعدی است. همچنین میدانیم نقاط vanishing، نقاطی در تصویر هستند که خطوط موازی در دنیای واقعی، در آن محل به هم همگرا می‌شوند که از این ویژگی‌ای است که در حل قسمت‌های قبل استفاده کرده ایم؛ حال اگر فرض کنیم یک صحنه در دنیای واقعی وجود دارد که سه مجموعه خط

موازی در آن وجود دارد و هر مجموعه به یک نقطه vanishing متفاوت همگرا می‌شود، میتوانیم این سه نقطه را با نام V1، V2 و V3 نشان دهیم. حال P که نقطه تلاقی بین صفحه‌ی تصویر و خطی است که کانون دوربین و نقطه vanishing با جهت عمود بر صفحه‌ی تصویر را به هم وصل می‌کند. حالا، میتوانیم یک مثلث با استفاده از سه نقطه vanishing P3P1 و P2P3 طول اضلاع این مثلث را به عنوان P_1P_2 و P_2P_3 نشان می‌دهیم که P_1P_2 و P_2P_3 ناقاط تلاقی صفحه‌ی تصویر با خطوطی هستند که نقطه P و نقاط محو شونده متناظر V1، V2 و V3 را به هم وصل می‌کنند.

حال برای اثبات اینکه نقطه اصلی P در وسط مثلث قرار دارد، باید نشان دهیم که طول اضلاع مثلث برابر است و این را بایستی برای هر سه ضلع داشته باشیم. (اگر نشان بدیم که طول اضلاع مثلث برابر بوده و مثلث متساوی الاضلاع واقع شده است، محل تلاقی میانه‌ها منطبق بر محل تلاقی عمود منصف‌ها، نیمسازها و خصوصاً ارتفاع‌ها بوده و در نتیجه در مرکز مثلث قرار دارد که همان نقطه P مدنظر ما می‌باشد)

族群 اول: $P_1P_2 = P_1V_1 + V_1P_2$: بر اساس تعریف نقطه P، P_1V_1 فاصله بین P1 و V1 در تصویر را نشان می‌دهد و V_1P_2 نشان‌دهنده فاصله بین V1 و P2 در تصویر است. بنابراین، P_1P_2 مجموع این دو فاصله است.

族群 دوم: $P_2P_3 = P_2V_2 + V_2P_3$: به طور مشابه P_2P_3 مجموع فاصله بین P2 و $(P_2V_2 + V_2P_3)$ و فاصله بین V2 و (V2P3) است.

族群 سوم: $P_3P_1 = P_3V_3 + V_3P_1$: دوباره به طور مشابه P_3P_1 مجموع فاصله بین P3 و $(P_3V_3 + V_3P_1)$ و فاصله بین V3 و (V3P1) است.

برای نشان دادن اینکه نقطه P در وسط مثلث قرار دارد، باید نشان دهیم که $P_1P_2 = P_2P_3 = P_3P_1$ است. با توجه به خصوصیات پرسپکتیو و نقاط vanishing، می‌دانیم که فواصل بین نقطه P و نقاط vanishing V1 و V2 و V3 برابر است. بنابراین، می‌توانیم نتیجه بگیریم که $P_1P_2 = P_2P_3 = P_3P_1$ و در نتیجه نقطه P در عمل در وسط مثلث قرار دارد.

تعیین موقعیت دوربین

بخش سوم: تعیین موقعیت دوربین

الف) فرآیند کلی الگوریتم CPE

ابتدا خود مفهوم تعیین موقعیت دوربین را بررسی می‌کنیم؛ تعیین موقعیت دوربین به فرآیندی اطلاق می‌شود که طی آن بتوان موقعیت، محل قرارگیری، جهت/چرخش دوربین را نسبت به یک صحنه یا نسبت به یک سیستم مختصات سه بعدی محاسبه نمود که پایه‌ی آن بر اساس روابط هندسی حاکم بین نقاط در سه بعد و تصویر آن در دو بعد است. دانستن موقعیت دوربین می‌تواند کمک شایانی در مسائل بینایی کامپیوتر به خصوص مکان‌یابی نماید. فرآیند کلی الگوریتم‌های CPE مبتنی بر چهارم گام اصلی و دو گام جانبی می‌باشد که در ادامه بطور مختصر مورد بررسی قرار می‌گیرد.

اولین گام کالیبراسیون دوربین و بدست آوردن ویژگی‌های ذاتی دوربین می‌باشد. دومین گام، استخراج ویژگی از تصویر یا فریم‌های ورودی ویدیویی مورد نظر می‌باشد. این استخراج ویژگی بر حسب مسئله می‌تواند لبه، نقاط کلیدی، گوشها یا سایر توصیف‌های بصری و بینایی از آن باشد. سومین گام تطبیق و انطباق ویژگی‌های استخراجی از تصاویر می‌باشد؛ این تطبیق و تناظر یابی ویژگی می‌تواند بین فریم‌های ورودی یک ویدیو یا بین یک عکس و عکس مرجع از یک صحنه باشد. گام چهارم تخمین موقعیت با استفاده از ویژگی‌های تطبیق شده بین تصاویر می‌باشد(میتوان از دانش سه‌بعدی نیز مانند نقشه عمق نیز استفاده نمود). روند کلی تخمین موقعیت بدین صورت است که از روابط هندسی/معنایی بین ویژگی‌های متناظر استفاده کرده و جهت و موقعیت دوربین تخمین زده می‌شود. از روش‌های متعددی برای این گام استفاده می‌شود؛ به عنوان مثال از روش Perspective-n-Point (PnP) می‌توان استفاده نمود که در آن از n نقطه در فضای سه‌بعدی و تناظر آن در تصویر دو بعدی استفاده شده و بر اساس ساختار حاصل میتوان موقعیت دوربین را تخمین زد.

به عنوان مثالی دیگر مه در کلاس درس نیز مورد بررسی قرار گرفت، الگوریتم Iterative Closest Point (ICP) می‌باشد که در آن از ابر نقاط استفاده شده و بصورت تکراری تلاش می‌شود خطای تناظریابی و فاصله‌ی ابر نقاط با نقاط تصویر دو بعدی کمینه شده و بتوان موقعیت دوربین را تخمین زد. گام پنجم که یک گام فرعی محسوب می‌شود، بهبود

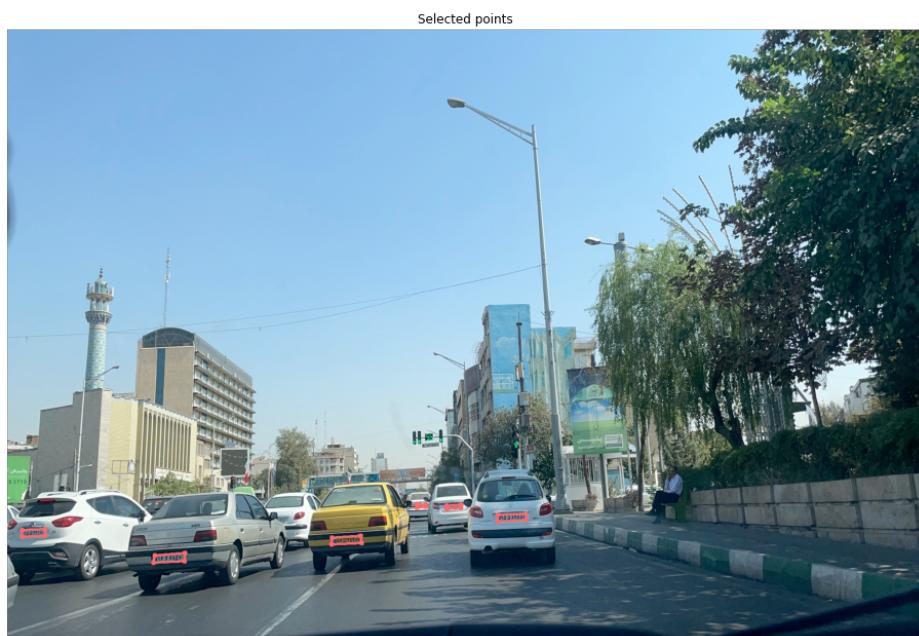
موقعیت تخمین زده شده با استفاده از روش‌های دیگر بوده و گام ششم نیز رهگیری و پیوستگی تعیین موقعیت دوربین در عکس‌های متوالی یا ویدیو می‌باشد.

ب) تعیین فاصله و جهت خودرو

برای این منظور بایستی نقاط چهار گوشه‌ی پلاک را از عکس تعیین نمود. برای اینکار من از کتابخانه‌ی open-cv و event های حاصل از کلیک روی عکس موقعیت چهار گوشه‌ی پلاک را بدست آورده‌ام.(این کد قبلا در سوال چهارم تمرین سری چهارم درس پردازش تصویر و مبحث face morphing برای تعیین نقاط استفاده کرده بودم)

نقاط انتخاب شده و خطوط حاصل از این نقاط در تصویر بعدی قابل مشاهده است. حال بایستی از نقاط چهار گوشه‌ی هر پلاک استفاده کرده و با کمک از ابعاد اصلی پلاک‌های استاندارد ایران اقدام به دست آوردن بردارهای انتقال و چرخش می‌کنیم. طبق جست و جو، ابعاد پلاک‌های ایران ۱۱ سانتی‌متر در ۵۲ سانتی‌متر است. نقاط گوشه‌ی چپ و بالای هر پلاک را مرجع گرفته و به تناسب ابعاد پلاک نقاط سه بعدی را بدست می‌آوریم بصورت زیر حاصل می‌شود که در آن ترتیب نقاط بصورت ساعت گرد در نظر گرفته شده است:

```
_ ref3D = np.array([[0, 0, 0], [52, 0, 0], [52, 11, 0], [0, 11, 0]], dtype=np.float32)
```



حال با استفاده از تابع solvePnP و ماتریس پارامترهای داخلی و با فرض اینکه هیچ تخریبی وجود ندارد، بردارهای انتقال و چرخش را برای هر پلاک(خودرو) بدست می‌وریم که بصورت زیر حاصل می‌شود؛ عنصر سوم آرایه انتقال، فاصله از مرجع در راستای عمق را نشان می‌دهد. برای بدست آوردن فاصله‌ی مستقیم(فاصله‌ی هوایی) در صورت نیاز می‌توانیم از روابط مثلثی و فیثاغورث استفاده نماییم. (تعیین ابعاد استاندارد پلاک تاثیر بزرگی در فواصل بدست آمده دارد) ترتیب فاصله‌های بدست آمده و انطباق آن ترتیب بصورت چشمی نشان میدهد عمق های بدست آمده درست می‌باشد.

```
=====
=> White 206:
--> Translation vector : [ 54.3762  303.27   1240.1381]
--> Rotation vector -> [ 0.2444 -0.0456 -0.0197]
=====
=> White Sayni:
--> Translation vector : [ -77.2277  567.1187 2433.2979]
--> Rotation vector -> [ 0.2186 -0.1351 -0.0103]
=====
=> White Samand:
--> Translation vector : [-257.7213  790.6281 3412.1209]
--> Rotation vector -> [-0.2143 -0.6478  0.056 ]
=====
=> Taxi:
--> Translation vector : [-243.5973  329.4619 1207.4068]
--> Rotation vector -> [-0.6737 -0.2619 -0.0667]
=====
=> 405:
--> Translation vector : [-544.2144  345.99   1173.7015]
--> Rotation vector -> [-0.5039 -0.037  -0.0508]
=====
=> White JAC:
--> Translation vector : [-888.8763  354.9886 1347.2099]
--> Rotation vector -> [-0.0886  0.1716 -0.0684]
```