بینایی ماشین

(کارشناسی ارشد فتوگرامتری)

(بخش اول، نسخه اولیه)

تالیف و ترجمه:

مسعود ورشوساز، مهر 1394

Contents

پیش گفتار

درس بینایی ماشین از دروس مهم کارشناسی ارشد فتوگرامتری است که در آن به روشها و تکنیک های استخراج اطلاعات هندسی از تصاویر به صورت اتوماتیک و هوشمند پرداخته می شود. این درس در دو بخش ارائه می شود بخش اول در بر گیرنده مبانی ریاضی و بخش دوم شامل مباحث پردازشی می باشد. نوشتار جاری، بحث اول یعنی مبانی ریاضی را مورد توجه قرار می دهد. شالوده اصلی این نوشته از دروس بینایی ماشین آقای Professor Hoff در Colorado School of Mines آمریکا استخراج شده است. اگرچه تجارب نویسنده و دانش وی در تکمیل، ساختار دهی محتوا و همسان سازی آن با نیازهای دانشجویان کارشناسی ارشد نقش مهمی در شکل گیری این مجموعه داشته است.

در هر فصل تلاش شده با تمرینات و اشکال مختلف دانشجو مرحله به مرحله با اصول کار آشنا گردد. این نوشتار اولیه بوده و از نقص و خطا به هیچ وجه مبرا نیست. لذا، بازخوردهای شما مسلما نه تنها مورد توجه که تقدیر این جانب قرار می گیرد.

مسعود ورشوساز

۱۳۹۳-8-17

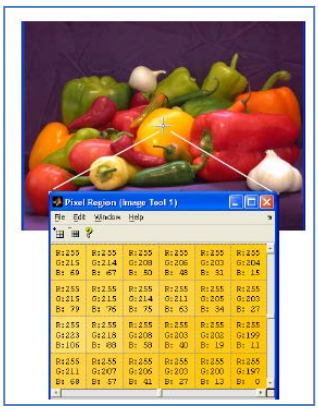
۱ سنسورها و تشکیل تصویر

۱-۱ تصاویر دیجیتال:

یک تصویر دیجیتال مجموعه ای (ماتریسی) از اعداد است که می توانند نشان دهنده شدت نور دریافتی (gray value)، فاصله تا عارضه (depth)، میزان جذب نور توسط عارضه و مسائلی از این دست باشند. منظور از عارضه شیء تصویر برداری شده می باشد. در شکل زیر تصویر یک فرد (سمت چپ) و بخشی از ماتریس مربوطه (قسمتی از صورت وی که در تصویر سمت راست نشان داده شده است) می باشد. در این ماتریس هر درایه عددی بین ۰ تا ۲۵۵ و متناسب با شدت نور دریافتی می باشد. در تصاویر سیاه و سفید، عدد صفر نمایانگر سیاه محض و عدد ۲۵۵ سفید محض می باشد.

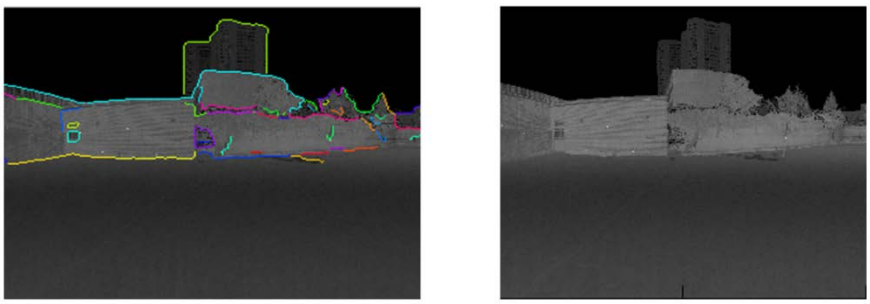
شکل ۱-۱: تصویر مرد دوربینی و مقادیر پیکسل تصویر مربوطه

یک تصویر رنگی در بر گیرنده سه تصویر به ترتیب برای رنگهای اصلی قرمز، سبز و آبی می باشد (شکل ۱-۲). در عمل، هر المان تصویر در بر گیرنده سه مقدار روشنایی برای طیف های قرمز، سبز و آبی دریافتی از عارضه می باشند



شکل ۱-۲: یک تصویر رنگی و لایه های رقومی آن

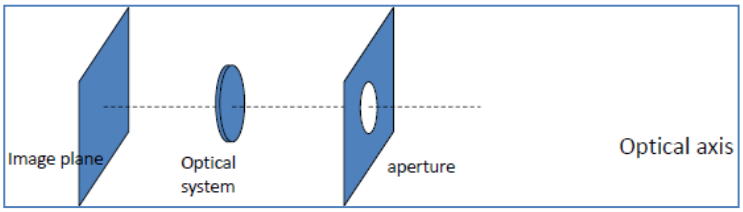
از دیگر تصاویر می توان به تصویر فاصله (Range Image) اشاره نمود که به آن depth map نیز گفته می شود. در این تصویر هر جزء تصویر به جای عدد Gray Level عددی متناسب با فاصله تا عارضه قرار دارد. نمونه مشهور این تصاویر، تصاویر لیزر اسکنرها می باشند. همانند دیگر تصاویر می توان از طریق پردازش های مختلف اطلاعات مفیدی از این نوع تصاویر استخراج نمود. شکل زیر نمونه ای از این تصاویر را نشان می دهد که با استخراج لبه های آن، تغییرات عمق به دست آمده است.



شکل ۱-۳: تصویر رنج داده های لیزر اسکن زمین والیبال دانشکده های عمران و نقشه برداری دانشگاه صنعتی خواجه نصیرالدین طوسی

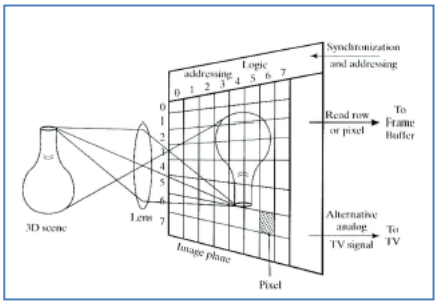
۱-۲ دوربین:

دوربین (شکل ۱-۴) وسیله ای است که وظیفه آن تصویر برداری از عارضه می باشد. با توجه به شکل المان های یک دوربین عبارتند از دیافراگم (aperture)، عدسی (lens) که برای فوکوس نور استفاده می شود و نیز سطح حساس که یا فیلم است یا یک سنسور دیجیتال (digital sensor).



شکل ۱-۴: ساختار یک دروبین معمولی

البته در کامپیوتر ویژن از دوربین های دیجیتال استفاده می شود. که شمای کلی آنها درشکل ۱-۵ دیده می شود.



شکل ۱-۵: ساختار یک دوربین رقومی

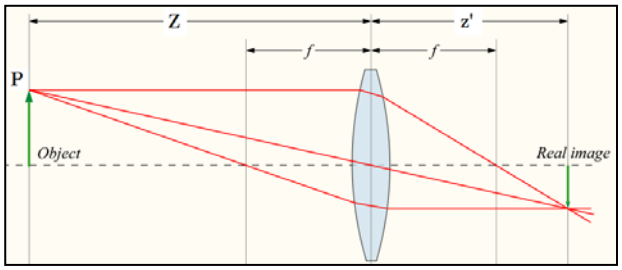
در این دوربین ها ثبت انرژی توسط سنسورهای دیجیتال صورت می پذیرد. سنسور یک دوربین معمولا مجموعه ای از CCDها (Charge Coupled Device) می باشد که در زمان دریافت نور، انرژی را جذب و پس از خاتمه ورود نور آن را آزاد می کند. نوع رایج دیگر سنسورها CMOS ها (Complementary Metal Oxide on Silicon) می باشند. CCD و CMOS هر دو سنسورهای دیجیتال بوده که وظیفه آنها تبدیل نور به سیگنال های الکتریکی می باشد. نسل اول دوربین ها بیشتر از CCD استفاده می کردند. CCD ها به گونه ای ساخته شده اند که پروسه تبدیل در خود chip صورت می پذیرد لذا تصاویر خروجی با نویز پایین و با کیفیت تولید می شوند. CMOS ها در حقیقت ترانزیستورهایی اند که شارژ دریافتی را از طریق کابل های معمولی انتقال می دهند. در نتیجه هر پیکسل می تواند به تنهایی و به سادگی تشکیل و مورد استفاده قرار گیرد. در نتیجه CMOS ها ارزانتر از CCD تولید می شوند. به طور خلاصه می توان گفت که CCD ها در زمان کوتاهتر، تصاویری با کیفیت و با نویز کمتر تولید می کنند. در مقابل CMOS ها برق کمتری را مصرف نموده و ارزانتر تولید می شوند. امروزه تکنولوژی CMOS به سرعت در حال پیشرفت است و کیفیت آنها قابل رقابت با CCD ها می باشد.

۱-۳ لنزهای باریک

همان طور که در بالا اشاره شد، بخش دوم یک دوربین، لنز است. در دوربین ها عموما از مدل لنزهای باریک استفاده می شود. یک لنز باریک لنزی است که ضخامت شیشه آن کم می باشد. در چنین حالتی با توجه به شکل ۱-۶ رابطه زیر برقرار می باشد.

$$\frac{1}{Z}+\frac{1}{z'}=\frac{1}{f}$$

طبیعتا در صورتی که عارضه در بی نهایت (فاصله دور از عدسی) باشد، 1/Z برابر با صفر و در نتیجه f با z' برابر خواهد بود. به عبارت دیگر در صورتی که عارضه در بی نهایت باشد برای داشتن یک تصویر واضح و شارپ از آن، باید فاصله سنسور تا مرکز عدسی برابر با f باشد.

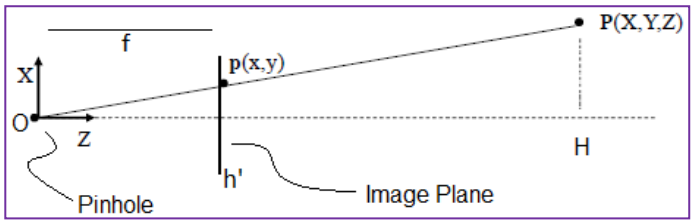


شکل ۱-۶: رابطه بین شیء و تصویر آن در یک دروبین

یک لنز خوب می تواند توسط مدل یک دوربین سوزنی (Pinhole Camera) تعریف شود که در آن نور بدون هیچگونه تغییری از نقطه سوزن (دیافراگم) عبور کرده و بر سطح فیلم تصویر می شود. به پروسه ای که نقاط عارضه پس از عبور از یک نقطه مرکزی بر روی سنسور تصویر می شوند perspective projection گفته می شود. دوربین های سوزنی در دوران رنسانس و برای بررسی اثر پرسپکتیو طراحی و به کار گرفته شدند.

شکل ۱-۷: ساختار یک دوربین سوزنی (Pinhole Camera)

در فتوگرامتری و کامپیوتر ویژن معمولا از مدل لنز باریک استفاده نمی شود بلکه از مدل دوربین های سوزنی بهره گرفته می شود. در عمل با استفاده از این مدل، رفتار نور در تشکیل تصویر باز سازی شده و سپس از طریق تصحیحاتی که بعدا راجع به آنها بحث خواهد شد مسیر نور در یک دوربین واقعی مدل سازی خواهد شد. شکل شکل ۱-۸ این مدل را نشان می دهد.



شکل ۱-۸: رابطه بین یک نقطه زمینی و تصویر آن

در این مدل نقطه O نقطه Pinhole یا همان مرکز عدسی است که به آن Perspective Centre[1] گفته می شود. نقطه P(X,Y,Z) نقطه ای از عارضه است که تصویر آن گرفته شده است. برای سهولت در

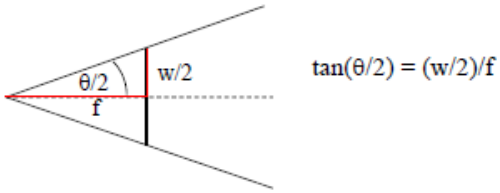
[1] توجه شود در دیکته آمریکایی این کلمه به صورت Center نوشته می شود. در سراسر این کتاب، هر جا که اختلافی بین نوشتار انگلیسی و آمریکایی باشد، دیکته انگلیسی مورد استفاده قرار خواهد گرفت. نمونه دیگر آن Organise در انگلیسی و Organize در آمریکایی می باشد.

محاسبات (جلوگیری از منفی شدن معادلات) معمولا فرض می شود صفحه تصویر در جلوی نقطه O قرار دارد نه در پشت آن. همان طور که در شکل دیده می شود، مثلث های 'Oph و OPH با هم مشابه می باشند. در نتیجه می توان نوشت:

$$x=f\,X/Z$$

$$y=f\,Y/Z$$

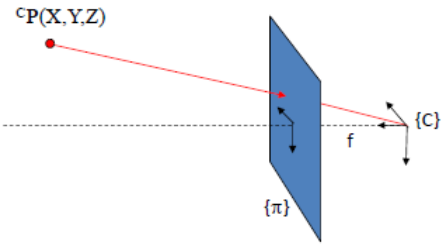
حال اگر طول محدوده دید (FOV[۲])سنسور دوربین را w بنامیم (شکل ...) رابطه زیر بین فاصله کانونی و w برقرار می باشد.



شکل ۱-۹: محاسبه محدوده دید یک دوربین

۱-٤ سیستم های مختصات:

اهم سیستم های مختصات در کامپیوتر ویژن عبارتند از سیستم مختصات دوربین، سنسور و تصویر. معمولا سیستم های مختصات سنسور و دوربین به صورت زیر تعریف می شوند که در آن π صفحه سنسور، {C} سیستم مختصات دوربین، $^{C}P(X,Y,Z)$ مختصات نقطه P در سیستم مختصات دوربین ({C})و f فاصله کانونی دوربین می باشد.



شکل ۱-۱۰: رابطه بین سیستم مختصات تصویر و دوربین

سیستم مختصات دوربین {C} کارتزین، سه بعدی و متریک بوده که واحد آن معمولا متر در نظر گرفته می شود. مبدا آن در Pinhole یا همان مرکز عدسی دوربین، Z آن در راستای محور نوری دوربین و به

---

[۲] Field of View

سمت بیرون، محور X به سمت راست و محور Y به سمت پایین می باشد. توجه شود که در اینجا فرض می شود محور Z بر صفحه سنسور (یعنی π) عمود است. سیستم مختصات سنسور یک سیستم مختصات متریک کارتزین دو بعدی است که واحد آن معمولا میلی متر می باشد. مبدا آن محل تقاطع محور نوری دوربین با صفحه سنسور می باشد. جهت محورهای x و y نیز به ترتیب موازی جهت محورهای X و Y می باشند.

تمرین:

الف:اگر f=5mm، مختصات تصویری نقطه ای از صحنه (عارضه) با مختصات (1,2,5) را حساب کنید.

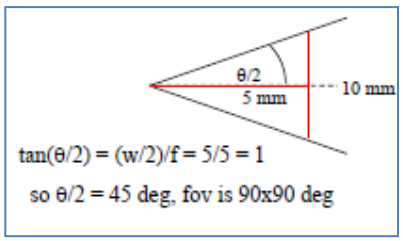
ب: همچنین اگر ابعاد صفحه سنسور 10mm در 10mm باشد، FOV چقدر است؟

ج: در نهایت اگر یک ساختمان که عرض آن ۱۰۰ متر است را بخواهیم تصویر برداری کنیم، دوربین در چه فاصله ای باشد تا تمام عرض ساختمان در تصویر ظاهر شود؟

حل بخش الف:

$$x = f\,X/Z = (5\text{ mm})\,(1\text{ m})/(5\text{ m}) = 1\text{ mm}$$
$$y = f\,Y/Z = (5\text{ mm})\,(2\text{ m})/(5\text{ m}) = 2\text{ mm}$$

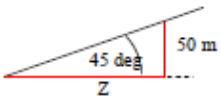
حل بخش ب:



شکل ۱-۱۱: محاسبه زاویه دید یک دوربین: یک مثال

حل بخش ج:

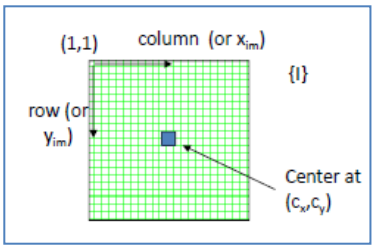
از قسمت قبل داشتیم که نصف زاویه دید برابر با ۴۵ درجه می باشد. در نتیجه داریم:



$$\tan(45\text{ deg}) = (W/2)/Z = 50/Z$$
$$Z = 50\text{ m}$$

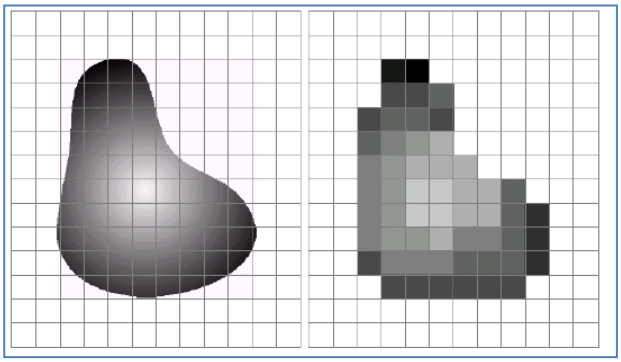
همانطور که اشاره شد، نور دریافتی از اشیاء، در صفحه سنسور در مقیاس واقعی (mm) بر روی CCD ها ثبت می شود. با این حال، علی رغم پیوستگی نور دریافتی، ثبت اطلاعات به صورت گسسته می باشد.

برای نشان دادن این گسستگی سیستم مختصات تصویر که به آن Image Buffer گفته می شود تعریف می شود. سیستم مختصات تصویر در حقیقت یک سیستم ماتریسی است که واحد آن پیکسل است و مبدا آن در سمت چپ بالای تصویر بوده و مختصات آن در زبان برنامه نویسی متلب برابر با (1,1) و در زبان c برابر با (0,0) می باشد (شکل ۱-۱۲).



شکل ۱-۱۲: Image Buffer یا همان سیستم مختصات تصویر

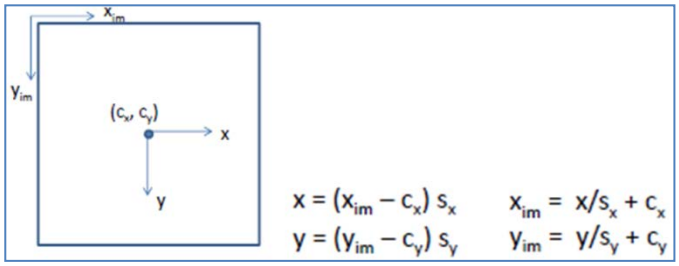
مختصات مرکز تصویر(C_x,C_y)می باشند. محور x نشان دهنده ستون های تصویر و محور y نشان دهنده سطرهای ماتریس تصویر می باشند. شکل ۱-۱۳ نشان دهنده یک شیء پیوسته و تصویر آن توسط سنسور دوربین می باشد.



شکل ۱-۱۳: صفحه سنسور و صفحه تصویر

۱-۴-۱ تبدیل بین صفحه سنسور و صفحه تصویر

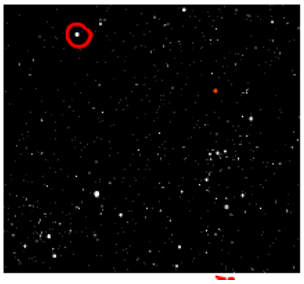
فرض کنیم که سایز یک پیکسل در راستای x و y برابر است با S_x و S_y به میلیمتر. بنابراین با توجه به شکل ۱-۱۴ می توان گفت که رابطه زیر بین مختصات یک نقطه در سنسور و تصویر برقرار است:



شکل ۱-۱۴: رابطه بین مختصات سنسوری و تصویری یک نقطه

تمرین:

فرض کنیم تصویری که توسط یک تلسکوپ گرفته شده است را داریم (شکل ۱-۱۵). مطلوب است به دست آورید بردار سه بعدی یک ستاره که در مختصات تصویری (r,c) = (100,200) قرار دارد. فرض کنید که تصویر ما 1000x1000 پیکسل بوده و ابعاد صفحه سنسور CCD برابر با 10mm x 10mm و فاصله کانونی تلسکوپ 1m باشد.



شکل ۱-۱۵: تصویری از آسمان

حل:

می دانیم که سیستم مختصات تصویری عبارتند از: $x = (x_{im} - c_x)\, s_x$ $y = (y_{im} - c_y)\, s_y$. با توجه به اینکه تصویر ۱۰۰۰ در ۱۰۰۰ پیکسل است در نتیجه داریم: $C_x=C_y=500$. در ضمن با توجه به اینکه سنسور دوربین 10mm x 10mm است می توان گفت: $S_x=S_y=10/1000=0.01mm$. بنابراین خواهیم داشت:

$$v = \begin{pmatrix} x \\ y \\ f \end{pmatrix} = \begin{pmatrix} (200-500)*0.01 \\ (100-500)*0.01 \\ 1 \end{pmatrix}$$

همان طور که در بالا گفته شد داریم: $x = (x_{im} - c_x)\, s_x$ ، $y = (y_{im} - c_y)\, s_y$ یا $x_{im} = x/s_x + c_x$ ، $y_{im} = y/s_y + c_y$. از طرفی می دانیم: $x = f\, X/Z$ ، $y = f\, Y/Z$.

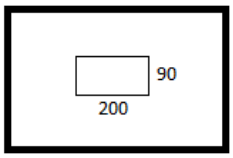
بنابراین می توانیم بنویسیم:

$$x_{im} = (f / s_x)\, X/Z + c_x$$
$$y_{im} = (f / s_y)\, Y/Z + c_y$$

حال اگر فرض کنیم: $f_x = (f / s_x)$ ، $f_y = (f / s_y)$ برای به دست آوردن مختصات تصویری یک نقطه به f_x و f_y نیاز داریم. اما از طرفی می دانیم که این دو هیچ نیستند جز اندازه فاصله کانونی f در راستای x و y منتهی در واحد پیکسل. این نشان می دهد ما نیازی به اندازه متریک f نداریم بلکه اگر نسبت های f/S_x یا f/S_y یا همان مقادیر پیکسلی f یعنی f_x و f_y نیز معلوم باشند می توانیم ارتباط بین سیستم مختصات تصویری و شیء را برقرار نماییم.

تمرین:

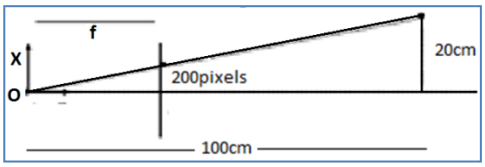
الف: فرض کنیم یک دوربین یک مستطیل به ابعاد 20cm x 10cm را که در فاصله یک متری آن قرار دارد با ابعاد 200 x 90 پیکسل تصویری ثبت کرده است. محاسبه کنید فاصله کانونی را بر اساس پیکسل.



شکل ۱-۱۶: تصویر یک مستطیل

ب: فرض کنید که تصویر 480 x 640 پیکسل باشد. در این صورت FOV دوربین را حساب کنید.

حل الف: می دانیم که فاصله تصویر تا صفحه برابر با یک متر یا 100cm است. بنابراین با توجه به شکل زیر می توانیم از تشابه دو مثلث بنویسیم: $20/100 = 200/f$ در نتیجه $f = 1000$ می باشد. البته این f در راستای x یا همان f_x می باشد. به همین ترتیب می توانیم f_y را به دست آوریم یعنی $f_y = 900$. توجه شود که اختلاف بین f_x و f_y نشان می دهد که پیکسل های دوربین مربعی نیستند.



شکل ۱-۱۷: محاسبه ارتفاع یک ساختمان

حل ب: برای به دست آوردن زاویه دید دوربین داریم: $w_x = 640/2 = 320$ در نتیجه می توانیم بنویسیم: $tg(\theta_x/2) = 320/1000$ در نتیجه θ_x تقریبا برابر با ۳۵ درجه و به همین ترتیب θ_y برابر با ۳۰ درجه محاسبه می شود.

۱-۵ پارامترهای دوربین

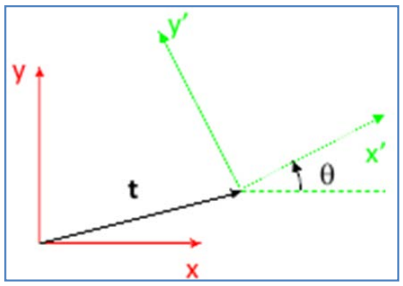
المان های یک دوربین را می توان به دو دسته داخلی و خارجی تقسیم نمود. المان های داخلی آنهایی اند که یک نقطه از تصویر رقومی را در سیستم مختصات دوربین تعریف می کنند. این المان ها عبارتند (f_x, f_y, C_x, C_y) و نیز المان های اعوجاج عدسی که بعدا مورد بحث قرار می گیرند. المان های خارجی دوربین موقعیت و دوران دوربین در سیستم مختصات مرجع زمینی را تعریف می کنند که به مجموعه آنها در کامپیوتر ویژن Pose (Position & Orientation) دوربین گفته می شود[۳].

۲ تبدیل بین سیستم های مختصات

در این بخش به سیستم های مختصات، ترانسفورماسیون و دوران در دو بعد و سه بعد پرداخته می شود.

۲-۱ تبدیل های دو بعدی

همانطور که در شکل ۲-۱ دیده می شود تبدیل یک سیستم مختصات صلب دو بعدی نسبت به یک سیستم مختصات دیگر بوسیله یک شیفت ($t=(\Delta x,\Delta y)^T$) و یک دوران (θ) تعریف می گردد. در چنین ترانسفورماسیونی شکل و ابعاد ثابت می مانند.



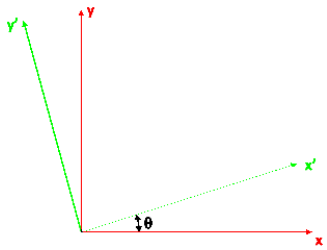
شکل ۲-۱: رابطه بین دو سیستم مختصات سه بعدی

---

[۳] در فتوگرامتری به آن توجیه دوربین (Camera Orientation) گفته می شود.

در این جا تعداد درجات آزادی معادل ۳ می باشد (دو شیفت در راستای x و y و یک دوران). حال ببینیم نحوه تاثیر دوران و شیفت بر مختصات یک نقطه چگونه است.

همان طور که در شکل ۲-۲ دیده می شود، فرض کنیم که سیستم مختصات (x',y') به اندازه θ حول محور عمود بر سیستم مختصات (x,y) دوران یافته باشد. این دوران در جهت خلاف عقربه های ساعت و از x به سمت y می باشد. در این صورت می توان نوشت: $\begin{pmatrix} x' \\ y' \end{pmatrix} = \mathbf{R} \begin{pmatrix} x \\ y \end{pmatrix}$ که در آن ثابت می شود که $\mathbf{R} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$. همان طور که دیده می شود ماتریس R یک ماتریس متعامد است یعنی اینکه حاصل ضرب داخلی هر دو سطر یا دو ستون آن در هم برابر با صفر ($\mathbf{r}_1 \cdot \mathbf{r}_2 = 0$ و $\mathbf{c}_1 \cdot \mathbf{c}_2 = 0$)، حاصل ضرب ماتریس در ترانهاده آن یکه ($\mathbf{R}\mathbf{R}^T = \mathbf{I}$) و دترمینان آن برابر با یک ($|\mathbf{R}| = 1$) می باشد. بنابراین می توان نوشت: $\begin{pmatrix} x \\ y \end{pmatrix} = \mathbf{R}^T \begin{pmatrix} x' \\ y' \end{pmatrix}$.



شکل ۲-۲: دوران یک سیستم مختصات سه بعدی حول محور Z

۲-۱-۱ مختصات هموژن (Homogeneous Coordinates)

در کامپیوتر ویژن با توجه به اینکه نقاط عوارض در قالب یک سیستم پروژکتیو در دوربین تصویر می شوند، از سیستم مختصاتی به نام سیستم مختصات پروژکتیو (Projective Space) استفاده می شود. در این سیستم نقاط دارای مختصات هموژن یا Homogeneous Coordinates می باشند. برای ساخت مختصات هموژن هر نقطه کافی است عدد ۱ را به عنوان المان سوم (یا چهارم در حالت سه بعدی) به مختصات اقلیدسی نقطه اضافه نمود یعنی $\tilde{\mathbf{x}} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$. استفاده از مختصات هموژن، محاسبات را ساده می کند. به عبارت دیگر می توان به جای $\begin{pmatrix} x' \\ y' \end{pmatrix} = \mathbf{R}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$ یا $\mathbf{x}' = \mathbf{R}\mathbf{x} + \mathbf{t}$ نوشت: $\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ 0 \;\; 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$ یا $\tilde{\mathbf{x}}' = \mathbf{H}\tilde{\mathbf{x}}$.

در فضای پروژکتیو نقاطی که مضرب ثابتی از یک نقطه باشند همگی با هم برابر اند یعنی: $\tilde{\mathbf{x}} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} sx \\ sy \\ s \end{pmatrix}$. در حقیقت بردارهایی که تفاوت آنها فقط در مقیاس آنها (s) است با هم برابر در نظر گرفته می شوند. گفته می شود فضای پروژکتیو یک فضای دو بعدی (P^2) است که از طریق حذف نقطه (0,0,0) از فضای سه بعدی R^3 حاصل می شود یعنی: $P^2 = R^3 - (0,0,0)$. در حقیقت فضای پروژکتیو تمامی ویژگی‌های فضای اقلیدسی همچون زاویه، خط، نقطه و... را دارد. با این تفاوت که بر خلاف فضای اقلیدسی در این فضا هر دو خط حتماً در یک نقطه همدیگر را قطع می‌کنند و چیزی بنام توازی در این فضا معنی نمی‌دهد. در این فضا مختصات هر نقطه به جای (x,y) ، (x,y,1) می‌باشد. حال سئوالی مطرح است آنست که چرا ۱ و چرا یک عدد دیگر به جای یک استفاده نمی‌شود مثل (x,y,z) در پاسخ باید گفت نقطه‌ای به نام (x,y,1) با نقطه دیگری بنام (2x, 2y, 2) با هم برابر می‌باشند.

و یا به طور کلی (Kx, Ky, K) همه با هم برابر می‌باشند که در آن $k \neq 0$است. به عبارت دیگر در این فضا، نقاط با کلاس‌هایی از نقاط تعریف می‌شوند که در این صورت در نقطه با هم برابرند در صورتیکه اختلاف مختصات آنها یک عدد ثابت باشد. در این صورت می‌توان با داشتن یک مجموعه همگن نقطه (Kx, Ky, K) نقطه اصلی در فضای اقلیدسی را یعنی x,y را بدست آورد. این کار براحتی از طریق تقسیم مختصات بر k حذف مولفه سوم یعنی k امکان پذیر می‌باشد.

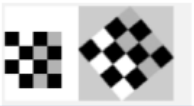
با این حال در صورتیکه ما (x,y,0) باشد، هیچ نقطه‌ای در فضای اقلیدسی نمی‌توان پیدا کرد چرا که آن نقطه بایستی $\left(\frac{x}{0}, \frac{y}{0}, \frac{0}{0}\right)$باشد که البته همان مفهوم بینهایت است که براحتی در فضای پروژکتیو تعریف می‌شود.

توضیحات تفصیلی این سیستم مختصات و مفاهیم مرتبط با آن را می توان در کتاب Multiview Geometry نوشته آقایان Hartley & Zisermann یافت.

۲-۱-۲ *ترانسفورماسیون های دو بعدی*

در فضای پروژکتیو، ترانسفورماسیون های کانفورمال (Similarity Transformation) و افاین (Affine) به صورت زیر نوشته می شوند:

$$\begin{pmatrix} x_B \\ y_B \\ 1 \end{pmatrix} = \begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_A \\ y_A \\ 1 \end{pmatrix}$$



شکل ۲-۳: ترانسفورماسیون کانفورمال (Similarity Transformation)

$$\begin{pmatrix} x_B \\ y_B \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_A \\ y_A \\ 1 \end{pmatrix}$$

شکل ۲-۴: ترانسفوماسیون افاین

همان گونه که می دانیم در ترانسفورماسیون کانفورمال زوایا حفظ اما طول ها به دلیل تغییر مقیاس حفظ نمی شوند. این در حالی است که در ترانسفورماسیون افاین علاوه بر طول ها، زوایا نیز تغییر می کنند اما خطوط موازی، موازی باقی می مانند. ترانسفورماسیون جامعی که در کامپیوتر ویژن استفاده می شود ترانسفورماسیون پروژکتیو می باشد که در حقیقت بواسطه آن تصویر شیء ایجاد شده و به آن هوموگرافی (Homography) گفته می شود. این ترانسفورماسیون یک سیستم مختصات پروژکتیو را به دیگری به طور عام تبدیل می کند و به این صورت نوشته می شود:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x_A \\ y_A \\ 1 \end{pmatrix}$$

۲-۱-۳ *ترانسفورماسیون پروژکتیو یا هوموگرافی:*

در حقیقت توسط این ترانسفورماسیون یک صفحه به صفحه دیگر تبدیل می شود یعنی:

شکل ۲-۵: تبدیل یک صفحه از طریق یک هوموگرافی

جدول زیر به طور خلاصه ویژگی های ترانسفورماسیون ها را نشان می دهد. در این جدول هر ترانسفورماسیون ویژگی های ترانسفورماسیون سطر بالای خود را نیز دارا می باشد. به عنوان مثال ترانسفورماسیون کانفورمال نه تنها زوایا بلکه توازی و مستقیم بودن خطوط را حفظ می کند. توجه شود که به جهت سادگی فرم ماتریس های هر ترانسفورماسیون به صورت دو سطری نوشته شده است. به

عبارت دیگر برای تشکیل ماتریس کامل هر ترانسفورماسیون باید $[0^T\ 1]$ به انتهای تمامی ماتریس ها اضافه شود.

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} I \mid t \end{bmatrix}_{2\times3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} R \mid t \end{bmatrix}_{2\times3}$	3	lengths	
similarity	$\begin{bmatrix} sR \mid t \end{bmatrix}_{2\times3}$	4	angles	
affine	$\begin{bmatrix} A \end{bmatrix}_{2\times3}$	6	parallelism	
projective	$\begin{bmatrix} \tilde{H} \end{bmatrix}_{3\times3}$	8	straight lines	

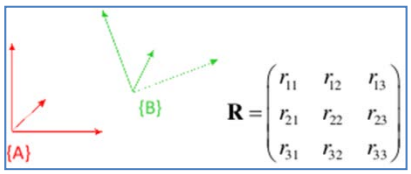
شکل ۲-۶: ویژگی ترانسفورماسیون های مختلف

۲-۲ تبدیل های سه بعدی

نمونه هایی از سیستم های مختصات سه بعدی سیستم مختصات زمینی، دوربین و مدل می باشند. فرض کنیم دو سیستم داریم A و B و بخواهیم B را نسبت به A توجیه کنیم که این توجیه شامل یک جابجایی (t) و یک دوران R می باشد که بر خلاف حالت دو بعدی شامل سه دوران می باشد. در ادامه به چگونگی به دست آوردن المان های توجیه می پردازیم.

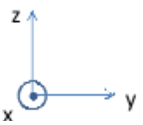
۲-۲-۱ *دوران های سه بعدی*

به طور کلی می توان ارتباط دو سیستم مختصات را که یکی نسبت به دیگری دوران یافته است را از طریق یک ماتریس دوران به دست آورد (شکل ۲-۸).



شکل ۲-۷: دوران یک سیستم مختصات سه بعدی

شکل زیر را در نظر بگیرید که در بر گیرنده محورهای y و z بوده و محور x عمود بر صفحه می باشد. در کامپیوتر ویژن برای نشان دادن جهتی محوری همچون x از علامت های ⊙ و ⊗ استفاده می شود که به ترتیب به معنی جهت محور x به سمت بیرون (بیننده) و داخل می باشد.



شکل ۲-۸: دوران سه بعدی حول محور z

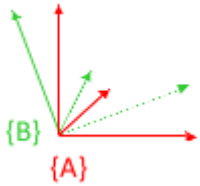
اگر فرض کنیم که سیستم مختصات {A} را حول محور x سیستم مختصات {B} به میزان θ_x دوران داده ایم داریم:

$$\begin{pmatrix} {}^{B}x \\ {}^{B}y \\ {}^{B}z \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_X & -\sin\theta_X \\ 0 & \sin\theta_X & \cos\theta_X \end{pmatrix} \begin{pmatrix} {}^{A}x \\ {}^{A}y \\ {}^{A}z \end{pmatrix}$$

در رابطه فوق ${}^{B}x$ و ${}^{A}x$ نشان دهنده مختصه x نقطه های A و B در سیستم های مختصات {A} و {B} می باشند. به همین ترتیب برای دوران حول محورهای y و z می توانیم بنویسیم:

$$\begin{pmatrix} {}^{B}x \\ {}^{B}y \\ {}^{B}z \end{pmatrix} = \begin{pmatrix} \cos\theta_Y & 0 & \sin\theta_Y \\ 0 & 1 & 0 \\ -\sin\theta_Y & 0 & \cos\theta_Y \end{pmatrix} \begin{pmatrix} {}^{A}x \\ {}^{A}y \\ {}^{A}z \end{pmatrix} \begin{pmatrix} {}^{B}x \\ {}^{B}y \\ {}^{B}z \end{pmatrix} = \begin{pmatrix} \cos\theta_Z & -\sin\theta_Z & 0 \\ \sin\theta_Z & \cos\theta_Z & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} {}^{A}x \\ {}^{A}y \\ {}^{A}z \end{pmatrix}$$

بنابراین برای ماتریسی که سیستم مختصات {A} را به ترتیب حول محورهای ${}^{B}x$، ${}^{B}y$ و ${}^{B}z$ دوران می دهد می توان نوشت:



$${}^{B}_{A}\mathbf{R} = \mathbf{R}_Z\,\mathbf{R}_Y\,\mathbf{R}_X = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

بنابراین اگر بخواهیم برداری همچون v را که در سیستم مختصات {A} است را دوران دهیم تا معادل آن را در {B} به دست آوریم داریم: ${}^{B}\mathbf{v} = {}^{B}_{A}\mathbf{R}\ {}^{A}\mathbf{v}$. نکته ای که وجود دارد آنست که جهت اعمال دوران ها بر روی مقادیر ماتریس R تاثیر دارد. به عبارت دیگر می دانیم که ${}^{A}_{B}R_{XYZ}(\theta_X,\theta_Y,\theta_Z)$ (ترتیب اعمال دوران X، Y و پس از آن Z) برابر است با $R_Z(\theta_Z)\,R_Y(\theta_Y)\,R_X(\theta_X)$ و نیز ${}^{A}_{B}R_{ZYX}(\theta_X,\theta_Y,\theta_Z)$ (ترتیب اعمال دوران Z، Y و پس از آن X) برابر است با $R_X(\theta_X)R_Y(\theta_Y)\,R_Z(\theta_Z)$. با توجه به نبود خاصیت جابه جایی در ضرب ماتریس ها طبیعتا ${}^{A}_{B}R_{XYZ}(\theta_X,\theta_Y,\theta_Z) \neq {}^{A}_{B}R_{ZYX}(\theta_X,\theta_Y,\theta_Z)$. نکته دیگر

آنکه، همان طور که قبلا گفتیم، ماتریس دوران یک ماتریس متعامد است. بنابراین می توانیم بنویسیم: ${}^{A}_{B}\mathbf{R} = \left({}^{B}_{A}\mathbf{R}\right)^{T} = \left({}^{B}_{A}\mathbf{R}\right)^{-1}$. به عبارت دیگر برای به دست آوردن معکوس ماتریس دوران کافی است ترانهاده آن را به دست آوریم که کاری ساده و آسان می باشد.

تمرین:

فرض کنید مقادیر دوران حول محورهای x، y و z به ترتیب در واحد رادیان برابر باشند با 0.1، 0.2- و 0.3. مطلوب است محاسبه کنید ماتریس R را در حالتی که ترتیب اعمال دوران ها حول محور x، حول محور y و حول محور z باشد. پس از آن R را در حالت معکوس یعنی حول محور z، حول محورy و حول محور x به دست آورید.

حل: این برنامه به شکل زیر می باشد:

```
ax = 0.1; ay = -0.2; az = 0.3;
% radians

Rx = [ 1 0 0; 0 cos(ax) -
sin(ax); 0 sin(ax) cos(ax)];
Ry = [ cos(ay) 0 sin(ay); 0 1
0; -sin(ay) 0 cos(ay)];
Rz = [ cos(az) -sin(az) 0;
sin(az) cos(az) 0; 0 0 1];
R = Rz * Ry * Rx
R =
0.9363 -0.3130 -0.1593
0.2896 0.9447 -0.1538
0.1987 0.0978 0.9752
```

۲-۲-۲ *به دست آوردن زوایا از طریق المان های ماتریس دوران*

در صورت معلوم بودن المان های ماتریس دوران می توان از طریق روابط زیر زوایای دوران را محاسبه نمود.

$$\theta_Y = \text{atan2}\left(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2}\right)$$
$$\theta_Z = \text{atan2}\left(r_{21}/cy, r_{11}/cy\right)$$
$$\theta_X = \text{atan2}\left(r_{32}/cy, r_{33}/cy\right)$$

توجه: در رابطه فوق، cy همان کسینوس θ_y است که برای خلاصه شدن به این صورت نوشته شده است.

همان طور که دیده می شود برای θ_y دو جواب وجود دارد (با توجه به جذر رادیکال). البته اگر θ_y بین مثبت و منفی ۹۰ درجه باشد مشکلی ایجاد نمی شود. البته اگر θ_y دقیقا برابر با ۹۰ یا 90- باشد، مقدار $\cos(\theta_y)$ برابر با صفر شده و مقادیر θ_z و θ_y به دست نمی آیند. در چنین حالتی ما θ_z را صفر فرض می کنیم (یعنی یک از جوابها). در نتیجه برای حالتی که $\theta_y = 90$ داریم $\theta_X = \operatorname{atan}2(r_{12}, r_{22})$ و برای حالتی که $\theta_y = -90$ داریم $\theta_X = -\operatorname{atan}2(r_{12}, r_{22})$

۲-۲-۳ *ماتریس دوران با زوایای کوچک*

برخی اوقات دوران ها بسیار کوچک می باشند مثل حالتی که عارضه در یک سکانس ویدئویی به آرامی در حال حرکت باشد. در چنین حالتی اگر بخواهیم ارتباط بین فریم ها را به دست آوریم با حالتی رو به رو هستیم که در بین دو فریم متوالی عارضه به میزان کمی دوران داشته است. در چنین حالتی فرم ماتریس دوران ساده می شود. از ضرب ماتریس های دوران حول های مجزا داریم:

$$ {}^{A}_{B}R_{XYZ}(\theta_X,\theta_Y,\theta_Z)=\begin{pmatrix} \cos\theta_Z\cos\theta_Y & \cos\theta_Z\sin\theta_Y\sin\theta_X-\sin\theta_Z\cos\theta_X & \cos\theta_Z\sin\theta_Y\cos\theta_X+\sin\theta_Z\sin\theta_X \\ \sin\theta_Z z\cos\theta_Y & \sin\theta_Z z\sin\theta_Y\sin\theta_X+\cos\theta_Z\cos\theta_X & \sin\theta_Z\sin\theta_Y\cos\theta_X-\cos\theta_Z\sin\theta_X \\ -\sin\theta_Y & \cos\theta_Y\sin\theta_X & \cos\theta_Y\cos\theta_X \end{pmatrix} $$

در حالتی که زوایای چرخش کوچک اند می توان گفت $\cos\theta \approx 1$ و نیز $\sin\theta \approx \theta$ (زوایا به رادیان). علاوه بر این می توان از مقدار حاصل ضرب دو سینوس صرف نظر نمود. بنا براین داریم:

$$ {}^{A}_{B}R_{XYZ}(\theta_X,\theta_Y,\theta_Z)\approx\begin{pmatrix} 1 & -\theta_Z & \theta_Y \\ \theta_Z & 1 & -\theta_X \\ -\theta_Y & \theta_X & 1 \end{pmatrix} $$

۲-۲-۴ *ارتباط المان های ماتریس دوران و کسینوس های محورها*

المان های ماتریس دوران در حقیقت حاصل ضرب داخلی کسینوس های هادی دو سیستم در هم اند. به عبارت دیگر داریم:

$$ {}^{B}_{A}\mathbf{R}=\begin{pmatrix} \hat{\mathbf{x}}_A\cdot\hat{\mathbf{x}}_B & \hat{\mathbf{y}}_A\cdot\hat{\mathbf{x}}_B & \hat{\mathbf{z}}_A\cdot\hat{\mathbf{x}}_B \\ \hat{\mathbf{x}}_A\cdot\hat{\mathbf{y}}_B & \hat{\mathbf{y}}_A\cdot\hat{\mathbf{y}}_B & \hat{\mathbf{z}}_A\cdot\hat{\mathbf{y}}_B \\ \hat{\mathbf{x}}_A\cdot\hat{\mathbf{z}}_B & \hat{\mathbf{y}}_A\cdot\hat{\mathbf{z}}_B & \hat{\mathbf{z}}_A\cdot\hat{\mathbf{z}}_B \end{pmatrix} $$

برای روشن تر شدن موضوع، فرض کنید بخواهیم ماتریس دوران را به بردار یکه محور x اعمال کنیم. در این صورت داریم:

$$ {}^{B}_{A}\mathbf{R}\ {}^{A}\hat{\mathbf{x}}_A = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{pmatrix} r_{11} \\ r_{21} \\ r_{31} \end{pmatrix} $$

اما از طرفی می دانیم که: ${}^{B}_{A}\mathbf{R}\ {}^{A}\hat{\mathbf{x}}_A = {}^{B}\hat{\mathbf{x}}_A$ که معنی آن اینست که بردار نقطه A در سیستم مختصات A پس از دوران تبدیل می شود به بردار یکه نقطه A در سیستم مختصات B. بنابراین از مقایسه این رابطه با رابطه بالا، می توان دید که ستون های ماتریس R در حقیقت دوران یافته بردار های یکه سیستم مختصات A می باشند یعنی ${}^{B}\hat{\mathbf{x}}_A$. پس می توانیم R را به صورت زیر بنویسیم:

$$ {}^{B}_{A}\mathbf{R} = \begin{pmatrix} \begin{pmatrix} {}^{B}\hat{\mathbf{x}}_A \end{pmatrix} & \begin{pmatrix} {}^{B}\hat{\mathbf{y}}_A \end{pmatrix} & \begin{pmatrix} {}^{B}\hat{\mathbf{z}}_A \end{pmatrix} \end{pmatrix} $$

به همین ترتیب، سطرهای ماتریس R، بردارهای واحد سیستم مختصات B پس از دوران و در سیستم مختصات A می باشند. یعنی:

$$ {}^{B}_{A}\mathbf{R} = \begin{pmatrix} \begin{pmatrix} {}^{A}\hat{\mathbf{x}}_B{}^{T} \end{pmatrix} \\ \begin{pmatrix} {}^{A}\hat{\mathbf{y}}_B{}^{T} \end{pmatrix} \\ \begin{pmatrix} {}^{A}\hat{\mathbf{z}}_B{}^{T} \end{pmatrix} \end{pmatrix} $$

۲-۲-۵ *دوران حول یک محور غیر از محورهای x، y و z*

در ریاضیات نشان داده می شود که هر دوران را می توان با یک چرخش (θ) حول یک محور خاص (k) نیز تعریف نمود. در چنین حالتی داریم:

$$ R_k(\theta) = \begin{pmatrix} k_x k_x v\theta + c\theta & k_x k_y v\theta - k_z s\theta & k_x k_z v\theta + k_y s\theta \\ k_x k_y v\theta + k_z s\theta & k_y k_y v\theta + c\theta & k_y k_z v\theta - k_x s\theta \\ k_x k_z v\theta - k_y s\theta & k_y k_z v\theta + k_x s\theta & k_z k_z v\theta + c\theta \end{pmatrix} $$

که در آن:

$$ c\theta = \cos\theta,\ s\theta = \sin\theta,\ v\theta = 1 - \cos\theta $$

$$ \hat{\mathbf{k}} = (k_x, k_y, k_z)^T $$

این روش یکی از راه های غلبه بر ابهاماتی است که در به دست آوردن زوایای دوران از ماتریس دوران ایجاد می شود. در چنین حالتی می توان θ و k را از روابط زیر به دست آورد.

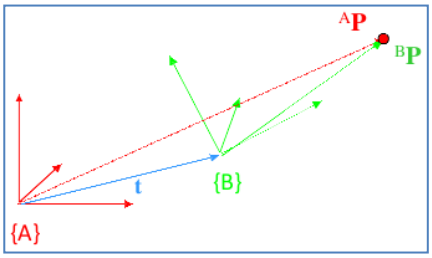
$$ \theta = \operatorname{acos}\left(\frac{r_{11} + r_{22} + r_{33} - 1}{2} \right) $$

$$ \hat{\mathbf{k}} = \frac{1}{2\sin\theta} \begin{pmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{pmatrix} $$

توجه: در اینجا θ– و k– نیز جواب می باشند.

۲-۲-۶ *ترانسفورماسیون یک نقطه از یک سیستم مختصات به سیستم دیگر*

حال ببینیم چگونه می توانیم یک نقطه در فریم **B** به فریم **A** ترانسفورماسیون شباهت (Similarity Transformation) بکنیم. رابطه زیر فرم کلی این ترانسفورماسیون که در بر گیرنده دوران و انتقال می باشد را نشان می دهد. در این رابطه، ${}^{A}\mathbf{P}$ مختصات نقطه در فریم (همان سیستم مختصات) **A** و ${}^{B}\mathbf{P}$ مختصات **P** در فریم **B** را نشان می هد. **t** نیز بردار انتقال می باشد که نشان دهنده مختصات مبدا سیستم مختصات فریم B در فریم A یعنی ${}^{A}\mathbf{t}_{Borg}$ می باشد.



$${}^{A}\mathbf{P} = {}^{A}_{B}\mathbf{R}\ {}^{B}\mathbf{P} + \mathbf{t}$$

شکل ۲-۹: تعیین مختصات یک نقطه از طریق تبدیل بین سیستم های مختصات سه بعدی

همانند حالت دو بعدی، در صورتی که مختصات هموژن نقطه P را در نظر بگیریم می توانیم بنویسیم:

${}^{B}\mathbf{P} = \mathbf{H}\ {}^{A}\mathbf{P}$ که در آن:

$${}^{A}\mathbf{P} = \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} sX \\ sY \\ sZ \\ s \end{pmatrix} \quad \text{و} \quad \mathbf{H} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

و یا به طور کلی می توانیم بنویسیم:

$${}^{A}\mathbf{P} = {}^{A}_{B}\mathbf{H} \cdot {}^{B}\mathbf{P} \qquad {}^{B}_{A}\mathbf{H} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & x_0 \\ r_{21} & r_{22} & r_{23} & y_0 \\ r_{31} & r_{32} & r_{33} & z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

تمرین:

فرض کنیم مختصات P در سیستم A برابر باشد با (1,0,1). اگر سیستم B در مختصات (0,0,10) بوده و به اندازه 180 درجه حول محور x نسبت به سیستم A چرخیده باشد، مختصات P در سیستم مختصات B را محاسبه کنید.

حل:

همان طور که می بینیم مختصات P در سیتم A معلوم است. حال می خواهیم ببینیم مختصات آن در فریم B چیست. آنچه باید به دست بیاوریم ماتریس تراتسفورماسیون از A به B یا به عبارت بهتر همان هوموگرافی از A به B یعنی ${}^{B}_{A}\mathbf{H}$. در چنین صورتی خواهیم داشت: ${}^{B}P = {}^{B}_{A}\mathbf{H} \cdot {}^{A}P$.

می دانیم که ${}^{B}_{A}\mathbf{H} = \left(\begin{array}{c|c} {}^{B}_{A}\mathbf{R} & {}^{B}\mathbf{t}_{Aorg} \\ \hline 0\ 0\ 0 & 1 \end{array}\right)$ که در آن ${}^{B}\mathbf{t}_{Aorg} = \begin{pmatrix} 0 \\ 0 \\ 10 \end{pmatrix}$. از طرفی داریم:

$$ {}^{B}_{A}\mathbf{R} = \mathbf{R}_x(180) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos 180 & -\sin 180 \\ 0 & \sin 180 & \cos 180 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} $$

در نتیجه خواهیم داشت:

$$ {}^{B}\mathbf{P} = {}^{B}_{A}\mathbf{H}\ {}^{A}\mathbf{P} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 10 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 9 \\ 1 \end{pmatrix} $$

برنامه متلبی که ترانسفورماسیون بالا را انجام می دهد به شرح زیر می باشد.

```
% Construct 4x4 transformation matrix to
transform A to B
R_A_B = [1 0 0; 0 -1 0; 0 0 -1] % 3x3 rotation
matrix
tAorg_B = [0; 0; 10] % translation (origin of A
in B)
H_A_B = [ R_A_B tAorg_B; % H_A_B means transform
A to B
                 0 0 0 1 ]
P_A = [1; 0; 1; 1] % A point in the A frame
 P_B = H_A_B * P_A % Convert to B frame
```

نکته ای که وجود دارد آنست که می توان چندین هوموگرافی را پشت سر هم اعمال نمود. یعنی داریم:

$$ {}^{C}_{A}\mathbf{H} = {}^{C}_{B}\mathbf{H}\ {}^{B}_{A}\mathbf{H} \qquad {}^{D}_{A}\mathbf{H} = {}^{D}_{C}\mathbf{H}\ {}^{C}_{B}\mathbf{H}\ {}^{B}_{A}\mathbf{H} $$

۲-۲-۷ معکوس ماتریس های هموژن

در اینجا ماتریس معکوس هوموگرافی در حقیقت معکوس ترانسفورماسیون است که در دل آن هم دوران دارد و هم شیفت. داریم: ${}^{A}_{B}\mathbf{H} = \left({}^{B}_{A}\mathbf{H}\right)^{-1}$. باید توجه نمود که با توجه به اینکه ماتریس H متعامد نیست لذا ${}^{A}_{B}\mathbf{H} \neq \left({}^{B}_{A}\mathbf{H}\right)^{T}$. بنابراین نمی توانیم مانند ماتریس دوران که معکوس آن را از طریق ترانهاده آن به دست می آوردیم در اینجا معکوس ماتریس H را به دست آوریم. در حقیقت راه اصلی که برای به دست آوردن ${}^{A}_{B}\mathbf{H}$ (یعنی همان معکوس ${}^{B}_{A}\mathbf{H}$ یا $\left({}^{B}_{A}\mathbf{H}\right)^{-1}$) وجود دارد از طریق روشهای معمول تعیین معکوس یک ماتریس می باشد. با این حال چنین راه هایی ممکن است کند باشند یا در هر صورت بخواهیم در صورت امکان معکوس ماتریس را به صورت مستقیم تری شبیه آنچه در به دست آوردن معکوس ماتریس دوران از طریق ترانهاده آن داشتیم به دست آوریم. راهی که در اینجا وجود دارد به این شکل است. همان طور که می دانیم:

$${}^{B}_{A}\mathbf{H} = \left(\begin{array}{c|c} {}^{B}_{A}\mathbf{R} & {}^{B}\mathbf{t}_{Aorg} \\ \hline 0\ 0\ 0 & 1 \end{array}\right)$$

لذا می توان المان های ${}^{B}_{A}\mathbf{H}$ را به شکل زیر در ${}^{A}_{B}\mathbf{H}$ قرار داد تا معکوس آن به دست آید.

$${}^{A}_{B}\mathbf{H} = \left(\begin{array}{c|c} {}^{B}_{A}\mathbf{R}^{T} & {}^{A}_{B}\mathbf{R}\left(-{}^{B}\mathbf{t}_{Aorg}\right) \\ \hline 0 & 1 \end{array}\right)$$

۲-۳ ترانسفورماسیون 3D به 2D

استفاده از این تبدیل در این درس مدل سازی ترانسفورماسیون پرسپکتیو (Perspective Projection) است که بر اساس آن تصویر شکل می گیرد. اساسا می توان یک مختصات سه بعدی را از طریق ماتریس زیر به دو بعدی تبدیل کرد.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

از طرفی می دانیم برای مختصات دو بعدی هموژن ، می توانیم مختصات را در ضریب 1/Z ضرب کنیم (یعنی بر Z) تقسیم کرد (چرا که در مختصات هموژن هر نقطه با مضارب آن برابر است) یعنی:

$$\tilde{\mathbf{x}} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} X/Z \\ Y/Z \\ 1 \end{pmatrix}$$

که در آن $\tilde{\mathbf{x}}$ مختصات دو بعدی هموژن است. از تلفیق دو رابطه بالا و نیز اضافه کردن ماتریس f (به شکل زیر) می توانیم بنویسیم:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

در این رابطه در صورتی که f برابر با یک فرض شود، به مختصات تصویری، مختصات نرمال شده می گویند. رابطه فوق، همان رابطه مختصات پرسپکتیو است که قبلا دیدیم منتهی در سیستم مختصات هموژن. برای نشان دادن این موضوع کافی است سمت راست رابطه را به دست آورده و سپس درایه ها را به درایه سطر سوم تقسیم کنیم یعنی:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}}_{\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix}$$

حال از تقسیم کردن طرفین بر درایه سوم داریم:

$$\begin{pmatrix} \frac{x_1}{x_3} \\ \frac{x_2}{x_3} \\ 1 \end{pmatrix} = \begin{pmatrix} fX/Z \\ fY/Z \\ 1 \end{pmatrix}$$

در رابطه فوق، x_1/x_3 همان مختصات x تصویری و x_2/x_3 مختصات y تصویری یک نقطه است پس:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} fX/Z \\ fY/Z \\ 1 \end{pmatrix}$$

در هر صورت می دانیم که فرم ماتریس المان های داخلی بیش از آنچه است که در بالا آمده است یعنی:

$$\mathbf{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \quad \text{یا} \quad \mathbf{K} = \begin{pmatrix} f/s_x & 0 & c_x \\ 0 & f/s_y & c_y \\ 0 & 0 & 1 \end{pmatrix}.$$

K ماتریس المان های داخلی دوربین، S_x,S_y مقیاس در محورهای x,y (یا همان سایز پیکسل در راستای x,y) و C_x,C_y نشان دهنده مختصات نقطه اصلی می باشند. پس به طور خلاصه اگر بخواهیم مختصات تصویری نقطه ای را در سیستم مختصات دوربین به دست آوریم داریم $\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x_1/x_3 \\ x_2/x_3 \\ 1 \end{pmatrix}$ که در آن:

$$\mathbf{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \quad \text{و} \quad \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \mathbf{K} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} {}^{C}\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}.$$

اندیس بالای **c** نشان دهنده مختصات نقطه در سیستم مختصات دوربین می باشد. رابطه فوق در حقیقت مختصات سه بعدی یک نقطه در سیستم دوربین را به مختصات دو بعدی نقطه در سیستم تصویر، تبدیل می کند.

حال ببینیم المان های توجیه خارجی را چگونه می توانیم به دست آوریم. ماتریس المان ای توجیه خارجی در حقیقت pose دوربین (Position and Rotation) در سیستم مختصات زمینی که در کامپیوتر ویژن به آن World Coordinate System گفته می شود را نشان می دهد. برای رسیدن از مختصات زمینی یک نقطه به تصویری آن، باید در ابتدا آن را به سیستم مختصات دوربین تبدیل کنیم. رابطه زیر این تبدیل را نشان می دهد:

$${}^{C}\mathbf{P} = {}^{C}_{W}\mathbf{H}\,{}^{W}\mathbf{P} = \begin{pmatrix} {}^{C}_{W}\mathbf{R} & {}^{C}\mathbf{t}_{Worg} \\ \mathbf{0} & 1 \end{pmatrix} {}^{W}\mathbf{P}$$

در این رابطه، بخشی از ماتریس H که دوران و جابه جایی از سیستم مختصات جهانی به سیستم مختصات دوربین را انجام می دهد، ماتریس توجیه خارجی نام داشته و عبارت است از:

$$\mathbf{M}_{ext} = \begin{pmatrix} {}^{C}_{W}\mathbf{R} & {}^{C}\mathbf{t}_{Worg} \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_X \\ r_{21} & r_{22} & r_{23} & t_Y \\ r_{31} & r_{32} & r_{33} & t_Z \end{pmatrix}$$

باید دقت نمود که در ماتریس بالا، ${}^{C}_{W}\mathbf{R}$ بر مبنای دوران سیستم مختصات W حول محورهای ثابت C نوشته می شود. ضمن آنکه ${}^{C}\mathbf{t}_{Worg}$ نشان دهنده مختصات مبدا سیستم مختصات جهانی در سیستم مختصات دوربین می باشد. در صورتی که مختصات مبدا سیستم مختصات دوربین در سیستم مختصات جهانی معلوم باشد می توانیم بنویسیم:

$$\mathbf{M}_{ext} = \begin{pmatrix} {}^{C}_{W}\mathbf{R} & {}^{C}\mathbf{t}_{Worg} \end{pmatrix} = \begin{pmatrix} {}^{C}_{W}\mathbf{R} & -{}^{C}_{W}\mathbf{R}\,{}^{W}\mathbf{t}_{Corg} \end{pmatrix}$$

از جمع دو ترانسفورماسیون داخلی و خارجی، می توان تصویر کامل پرسپکتیو یک نقطه زمینی بر روی تصویر (یعنی تصویر یک نقطه سه بعدی در سیبستم مختصات جهانی یا ${}^{W}\mathbf{P}$ به مختصات پیکسلی آن در تصویر یعنی x_{im}, y_{im}) را به شکل زیر به دست آورد:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \mathbf{K}\,\mathbf{M}_{ext} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}^{W}$$

که در آن داریم $x_{im} = x_1 / x_3,\ \ y_{im} = x_2 / x_3$

تمرین متلب:

فرض کنید که یک روبات داریم که بر روی آن یک دوربین نصب شده است. برنامه ای بنویسید که مختصات یک نقطه زمینی را بر روی تصویر گرفته شده توسط روبات ترانسفر می کند. فرض کنید که f=512 pix, (cx,cy)=(256,256). برنامه را برای نقطه ای به مختصات (16,0,-1,1) اجرا کنید.

```
H_V_W = [ 1 0 0 5; 0 -1 0 0; 0 0 -1 1; 0
0 0 1]
H_S_V = [ 0 0 1 1;
1 0 0 0;
0 1 0 -2;
0 0 0 1]
P_W = [ 16; 0; -1; 1];
K = [512 0 256; 0 512 256; 0 0 1 ];
R_C_V = [ 0 0 1; 1 0 0; 0 1 0];
R_V_C = R_C_V';
R_W_V = [1 0 0;
0 -1 0;
0 0 -1]';
R_W_C = R_V_C * R_W_V;
tCorg_V = [1; 0; -2; 1];
tCorg_W = H_V_W * tCorg_V; tCorg_W =
tCorg_W(1:3);
Mext = [ R_W_C -R_W_C * tCorg_W ];
p = K * Mext * P_W; p = p / p(3)
```

در صورتی که فاصله تا عارضه در مقایسه با تغییر عمق عوارض زیاد باشد، می توان تقریبی از تصویر پرسپکتیو (Perspective Projection) را استفاده نمود که در آن به جای Z از Z متوسط یا Z_{avg} استفاده می شود. به چنین حالتی Weak Perspective یا Scaled Orthography گفته می شود. در این جا داریم: $x = f\,X/Z_{avg},\ y = f\,Y/Z_{avg}$. بنابراین بردار تصویر نقطه به شکل $\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x_1/x_3 \\ x_2/x_3 \\ 1 \end{pmatrix}$ به دست می آید که در آن:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & Z_{avg} \end{pmatrix} {}^{C}\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

مزیت چنین حالتی آنست که مختصات تصویری نقاط، تابعی خطی از مختصات زمینی آنها می باشند. با استفاده از چنین شرطی، در صورتی که یک عارضه نسبتا مسطح داشته باشیم که در فاصله دور از دوربین قرار داشته و با دوران های کوچک در حال چرخش باشد ثابت می شود که:

$$\begin{pmatrix} x_B \\ y_B \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_A \\ y_A \\ 1 \end{pmatrix}$$

به عبارت دیگر، تبدیل از حالت سه بعدی به دو بعدی، به دو بعدی به دو بعدی تغییر می یابد.

۳ اعمال تغییرات دو بعدی بر روی تصاویر (Image Transforms)

در این فصل در ابتدا مروری خواهیم داشت بر روش کمترین مربعات. پس از آن خواهیم دید که چگونه می توانیم ترانسفورماسیون بین دو عکس را با روش کمترین مربعات پیدا کنیم. ترانسفورماسیون مربوط به یک دوران را پیدا نموده و نحوه اعمال آن بر یک تصویر را مرور خواهیم کرد. نمونه عملی که در این فصل در نهایت اجرا خواهیم کرد تولید یک تصویر ترمیم شده (Rectified) است.

۳-۱ مروری بر روش کمترین مربعات

فرض کنید مجموعه ای از x_i, y_i داریم به گونه ای که $y=f(x)=mx+b$ و بخواهیم که m و b را به دست آوریم. در روش کمترین مربعات پارامترهای m و b به گونه ای تعیین می شوند که رابطه $E=\sum_i |y_i - f(x_i)|^2$ حداقل شود. به طور کلی در روش کمترین مربعات داده های ورودی می تواند بردار باشند و تابع ما نیز رابطه خطی بین این مقادیر باشد. به عبارت دیگر داریم: $\mathbf{A}\,\mathbf{x} = \mathbf{b}$ که در آن مقادیر مجهول در بردار x و داده های ورودی در A و b قرار داده می شوند. به عنوان مثال در حالت برازاندن یک خط به مجموعه ای از نقاط داریم:

$$\mathbf{A}=\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \\ x_N & 1 \end{pmatrix},\quad \mathbf{b}=\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}\quad \mathbf{x}=\begin{pmatrix} m \\ b \end{pmatrix}$$

بنابراین برای حل مقادیر مجهولات باید رابطه $E=|\mathbf{Ax}-\mathbf{b}|^2$ (مربع نرم Ax-b) به حداقل برسد. برای این منظور با بسط رابطه فوق داریم:

$$E=\mathbf{x}^T\left(\mathbf{A}^T\mathbf{A}\right)\mathbf{x}-2\mathbf{x}^T\left(\mathbf{A}^T\mathbf{b}\right)+|\mathbf{b}|^2$$

حال برای اینکه مقدار مینیمم E را پیدا کنیم مشتق آن را نسبت به x گرفته و برابر با صفر قرار می دهیم. در این صورت ثابت می شود که خواهیم داشت:

$$\left(\mathbf{A}^T\mathbf{A}\right)\mathbf{x}=\mathbf{A}^T\mathbf{b}$$

به معادلات فوق، معادلات نرمال گفته می شود که با استفاده از آنها می توان x را به دست آورد. برای این منظور داریم

$$\mathbf{x}=\left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b}$$

به مقدار $(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$ معکوس مجازی (یا A^+)گفته می شود که در متلب از طریق pinv(A) (مخفف کلمات Pseudo Inverse) به دست می آید. در نتیجه با استفاده از دستورات زیر، با داشتن A و b می توان x را به دست آورد:

- x = pinv(A)*b;
- or x = A\b;

۳-۱-۱ *مثال: محاسبه ترانسفورماسیون بین دو عکس*

دو تصویر زیر را در نظر بگیرید. مطلوب است محاسبه کنید دوران و شیفت کتاب بین دو عکس را.

شکل ۳-۱: تصاویریک کتاب چرخش داده شده

حل: برای این منظور لازم است مختصات یک سری نقطه مشترک بین دو عکس را داشته باشیم. می دانیم که با توجه به اینکه سیستم مختصات دو تصویر دو بعدی است رابطه زیر بین آنها برقرار است که در آن c=cosθ و s=sinθ.

$$\begin{pmatrix} x_B \\ y_B \\ 1 \end{pmatrix} = \begin{pmatrix} c & -s & t_x \\ s & c & t_t \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_A \\ y_A \\ 1 \end{pmatrix}$$

اگر رابطه فوق را به صورت باز شده اصلی آن بنویسیم:

$$x_B = cx_A - sy_A + t_x$$
$$y_B = sx_A + cy_A + t_y$$

که در آن c، s، t_x و t_y مجهولات ما می باشند. در معادلات بالا توجه شود که c و s اگرچه وابسته اند (به θ) با این حال به صورت مستقیم در معادلات وارد شده اند تا معادلات به صورت خطی و حل آنها

ساده تر باشد. البته در هر صورت می توان پس از حل، مقدار θ ار از طریق $\cos^{-1}(x(1))$ به دست آورد. در هر صورت، با نوشتن معادلات در فرم ماتریسی خواهیم داشت:

$$\mathbf{A} = \begin{pmatrix} x_A^{(1)} & -y_A^{(1)} & 1 & 0 \\ y_A^{(1)} & x_A^{(1)} & 0 & 1 \\ \vdots & & & \\ y_A^{(N)} & x_A^{(N)} & 0 & 1 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} c \\ s \\ t_x \\ t_y \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} x_B^{(1)} \\ y_B^{(1)} \\ \vdots \\ y_B^{(N)} \end{pmatrix}$$

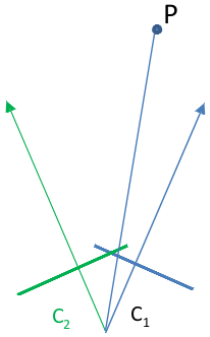
حال می توان با یک برنامه متلب مقادیر مجهولات را محاسبه نمود. در ابتدا با استفاده از دستور ('نام تصویر')imtool یا cpselect مختصات نقاط کتاب در دو تصویر را قرائت می کنیم[۴]. در این صورت، برنامه زیر محاسبات لازم برای محاسبه دوران و شیفت کتاب در دو تصویر را نشان میدهد.

```
% Using imtool, we find corresponding
% points (x;y), which are the four
% corners of the book
pA = [
    213 398  401  223;
    29   20  293  297];
pB = [
    207 391  339  164;
    7    34  302  270];
N = size(pA,2);
A = zeros(2*N,4);
for i=1:N
    A( 2*(i-1)+1, :) = [ pA(1,i) -pA(2,i)  1  0];
    A( 2*(i-1)+2, :) = [ pA(2,i)  pA(1,i)  0  1];
end
b = reshape(pB, [], 1);

x = A\b;

theta = acos(x(1));
tx = x(3);
ty = x(4);
```

[۴] دستور imtool یک فانکشن متلب است که یک تصویر را بالا آورده و امکان قرائت مختصات آن را فراهم می سازد

۳-۱-۲ *تعیین رابطه تصویر دوران یافته*

در این قسمت می خواهیم رابطه بین تصاویری که توسط دوربین در حال چرخش گرفته شده است را پیدا کنیم. در این جا فرض می کنیم که دوربین فقط دوران یافته و تحت هیچگونه جابجایی قرار نگرفته است. شکل زیر وضعیت دو تصویر را نشان می دهد. دوربین از حالت C_1 به حالت C_2 دوران یافته است.



شکل ۳-۲: دوران دوربین حول یک محور

در صورتی که یک نقطه زمینی P را در نظر بگیریم همان طوری که می دانیم می توان تصویر آن در سیستم C_1 که مبدا است را از طریق رابطه زیر به دست آورد:

$$\tilde{\mathbf{x}}_1 = \mathbf{K}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} {}^{C_1}X \\ {}^{C_1}Y \\ {}^{C_1}Z \\ 1 \end{pmatrix} = \mathbf{K}\begin{pmatrix} {}^{C_1}X \\ {}^{C_1}Y \\ {}^{C_1}Z \end{pmatrix}$$

در رابطه فوق، K ماتریس المان های داخلی دوربین و ماتریس پس از آن، ماتریس المان های توجیه خارجی است که با فرض مبدا بودن سیستم مختصات C_1 و صفر بودن میزان جابجایی ها نوشته شده است. $({}^{C1}X, {}^{C1}Y, {}^{C1}Z, 1)$ نیز نشان دهنده مختصات نقطه P در سیستم مختصات C_1 می باشد. از رابطه فوق به دست می آید:

$$\begin{pmatrix} {}^{C_1}X \\ {}^{C_1}Y \\ {}^{C_1}Z \end{pmatrix} = \mathbf{K}^{-1}\tilde{\mathbf{x}}_1$$

حال ببینیم مختصات نقطه P پس از دوران دوربین چگونه است. در این جا داریم:

$$\begin{pmatrix} {}^{C_2}X \\ {}^{C_2}Y \\ {}^{C_2}Z \\ 1 \end{pmatrix} = \begin{pmatrix} {}^{C_2}_{C_1}\mathbf{R} & \mathbf{0} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} {}^{C_1}X \\ {}^{C_1}Y \\ {}^{C_1}Z \\ 1 \end{pmatrix}, \quad \text{or} \quad \begin{pmatrix} {}^{C_2}X \\ {}^{C_2}Y \\ {}^{C_2}Z \end{pmatrix} = {}^{C_2}_{C_1}\mathbf{R} \begin{pmatrix} {}^{C_1}X \\ {}^{C_1}Y \\ {}^{C_1}Z \end{pmatrix}$$

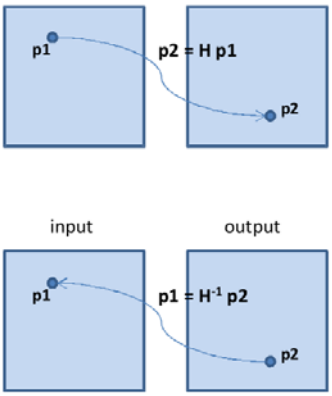
از تلفیق دو رابطه فوق با هم داریم:

$$\tilde{\mathbf{x}}_2 = \mathbf{K} \begin{pmatrix} {}^{C_2}X \\ {}^{C_2}Y \\ {}^{C_2}Z \end{pmatrix} = \mathbf{K}\ {}^{C_2}_{C_1}\mathbf{R}\ \mathbf{K}^{-1}\ \tilde{\mathbf{x}}_1$$

در رابطه فوق، ماتریس KRK^{-1} یک ماتریس ترانسفورماسیون پروژکتیو است که مختصات نقاط تصویر اول را سیستم مختصات دوربین دوم تصویر می کند. به عبارت بهتر، در صورت مشخص بودن H می توان از طریق رابطه $P_2=HP_1$ تمامی نقاط تصویر اول را به تصویر دوم تبدیل نمود.

۳-۱-۳ *تبدیل یک تصویر به تصویر دیگر از طریق یک هوموگرافی*

مشکلی که در عملیات ذکر شده در بالا وجود دارد آنست که ممکن است به تعدادی از نقاط در تصویر دوم هیچ نقطه ای از تصویر اول تبدیل نشده و در نتیجه درجه خاکستری این نقاط تعیین نشود. برای حل این مساله کاری که می توان انجام داد آنست که از طریق رابطه $P_1=H^{-1}P_2$ ، به ازاء هر نقطه از P_2 مختصات نقطه مربوطه در P_1 را به دست آورده و مقدار پیکسل (درجه خاکستری) مربوط به آن نقطه (در تصویر دوم) را تعیین نماییم. البته اگر مقداری که برای P_1 به دست می آوریم دقیقا منطبق بر نقاط تصویر اول نباشد، می توانیم درجه خاکستری نزدیکترین نقطه استفاده کنیم. هر دو حالت بالا در شکل زیر نشان داده شده اند.



شکل ۳-۳: تبدیل بین مختصات تصویری نقاط

تمرین متلب:

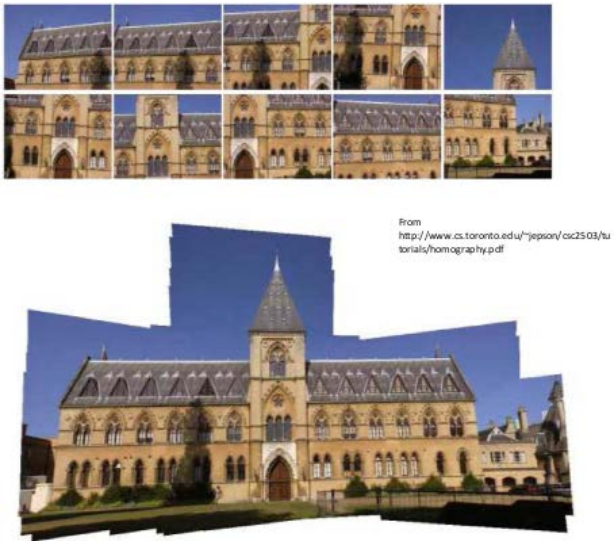
فرض کنید یک تصویر دارید. با فرض بر اینکه دوربین مربوطه ۳۰ درجه چرخیده باشد، تصویر حاصل را به دست آورید.

حل: برای حل این مساله کافی است که ماتریس هوموگرافی تبدیل از تصویر مبدا به مقصد را محاسبه نموده و آن را در تصویر ضرب کنیم. حاصل، تصویر چرخش یافته به میزان ۳۰ درجه می باشد. کد زیر این کار را انجام می دهد.

```
clear all
close all
I1 = imread('cameraman.tif');
I2 = zeros(size(I1));
% Say we have a rotation about the camera's y axis
 th = 30.0 * pi/180;
Ry = [ cos(th) 0 sin(th);
0 1 0; -sin(th) 0 cos(th)];
R_1_2 = Ry;
K = [ 128 0 128; 0 128 128; 0 0 1];
Kinv = inv(K);
% This is the projective transform (homography) from image 1 to image
2
H = K*R_1_2*Kinv
Hinv = inv(H);
for x2=1:size(I2,2)
          for y2=1:size(I2,1)
             p2 = [x2; y2; 1];
             p1 = Hinv * p2;
              p1 = p1/p1(3);
% We'll just pick the nearest point to p1 (better way is to %
interpolate).
               x1 = round(p1(1));
              y1 = round(p1(2));
if x1>0 && x1<=size(I1,2) && y1>0 && y1<=size(I1,1)
                        I2(y2,x2) = I1(y1,x1);
                    end
        end
end
```

۳-۱-۴ *یک مثال کامل: تهیه موزاییک از مجموعه ای از تصاویر*

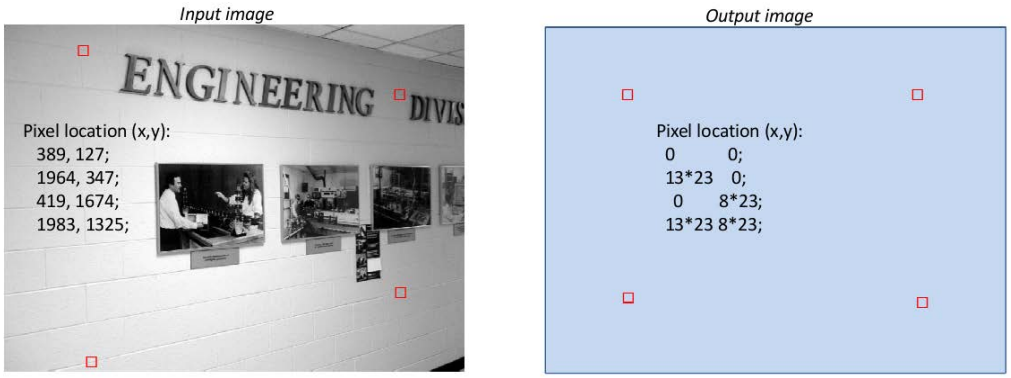
فرض کنیم که دو تصویر داریم که تنها تفاوت آنها آنست که یکی نسبت به دیگری دوران کرده است. بدیهی است ارتباط بین این دو یک هوموگرافی است که برای تعیین آن نیازمند تعدادی نقطه مشترک می باشیم. از آن جایی که ماتریس هوموگرافی دارای ۸ درجه آزادی است برای تعیین آن به حداقل ۴ نقطه مشترک احتیاج است. هر نقطه دو معادله داده و در نتیجه المانهای مجهول به دست می آیند. برای به دست آوردن المان ها از روش کمترین مربعات استفاده می کنیم. پس از آنکه ماتریس هوموگرافی شکل گرفت، ماتریس را به یک تصویر اعمال می کنیم تا آن را به صفحه تصویر دیگر تبدیل کنیم. شکل زیر این پروسه را نشان می دهد.



شکل ۳-۴: تصاویر و موزائیک تهیه شده

۳-۱-۵ *یک تمرین متلب کامل: ترمیم کردن دو تصویر و موزاییک کردن آنها به هم*

الف) دو تصویر زیر را در نظر بگیرید. در تصویر اول مختصات تصویری ۴ نقطه کنترل بر روی تصویر آمده است. ارتفاع مستطیل مشخص شده در شکل ۸ آجر و طول آن ۱۳ آجر می باشد. هر آجر هم 23cm طول دارد. بنابراین محدوده مستطیل 184x299cm می باشد. برای ترمیم تصویر، فرض کنید که هر پیکسل تصویر خروجی معادل 1cm باشد. مختصات نقطه بالا- سمت چپ را نیز 0,0 در نظر بگیرید.



شکل ۳-۵: ترمیم یک تصویر

حل:

با توجه به اطلاعات داده شده، مختصات تصویر نقاط مستطیل در تصویر دوم عبارتند از:

نقطه سمت چپ بالا: 0,0

نقطه سمت راست بالا: 0 ,(13*23)

نقطه سمت چپ پایین: 0,(8*23)

نقطه سمت راست پایین: (13*23), (8*23)

برای نوشتن این برنامه از دو فانکشن مفید متلب استفاده می کنیم: cp2tform و imtransform

cp2tform: ورودی این تابع دو دسته نقاط کنترل (Pts1, Pts2) می باشند. در این صورت، این تابع از طریق کمترین مربعات المان های ماتریس هوموگرافی پروژکتیو ('projective') را محاسبه می کند.

مثال: Tform = cp2tform(Pts1,Pts2,'projective');

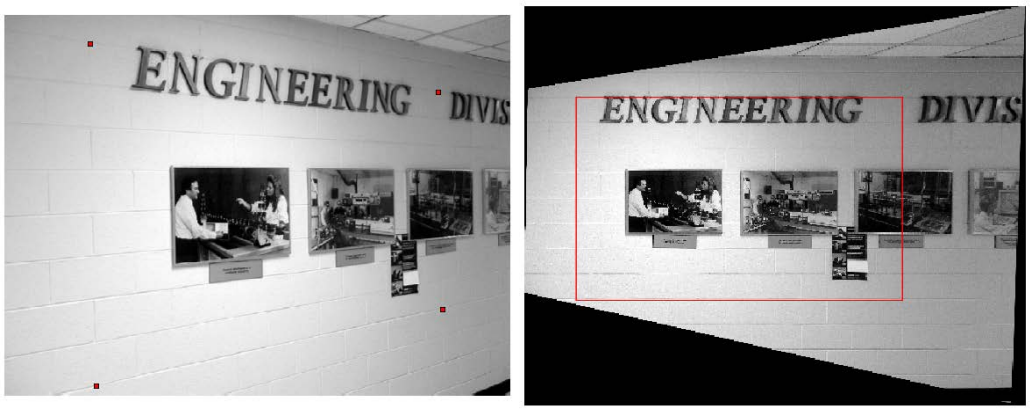
imtransform: این تابع یک تصویر را با استفاده از یک ماتریس هوموگرافی تبدیل می کند.

مثال: ITrans = imtransform(I1,Tform);

با توجه به مجموعه اطلاعات گفته شده، برنامه متلب مربوطه به شکل زیر است:

```
clear all
close all
Iin1 = imread('wall1.jpg');
imshow(Iin1,[]), impixelinfo;
% Location of control points in (x,y) input image coords (pixels)
% These are the corners of a rectangle that is 8 bricks high by 13
bricks % wide. Each brick is about 23 cm.
Pin1 = [
389, 127; 1964, 347; 419, 1674; 1983, 1325;
];
% Mark control points on input image
for i=1:size(Pin1, 1)
                rectangle('Position', [Pin1(i,1)-10 Pin1(i,2)-10 20
20], 'FaceColor', 'r');
  end
% Define location of control points in output image. We'll make 1
pixel
% equal to 1 cm. Also put the upper left control point at 0,0 pixels.
Pout1 = [
0,                  0;                  13*23,                  0;
0, 8*23; 13*23, 8*23;
];
% Compute transform, from corresponding control points
Tform1 = cp2tform(Pin1,Pout1,'projective');
% Transform input image to output image
Iout1 = imtransform(Iin1,Tform1);
figure, imshow(Iout1,[]);
```

خروجی برنامه بالا به شکل زیر خواهد بود:



شکل ۳-۶: یک تصویر و تصویر ترمیم شده آن

توجه شود که گوشه سمت چپ بالا، 0,0 نیست. این به آن دلیل است که فانکشن imtransform تصویر خروجی را بزرگ می کند تا تمامی تصویر تولید شده را در بر بگیرد. البته می توان جلوی این کار را گرفت.

اگر بخواهیم بخش تبدیل شده را در تصویر خروجی دقیقا نشان دهیم می توانیم از دستور زیر استفاده کنیم:

[ITrans, xdata, ydata] = imtransform(Iin1,Tform1);

در حقیقت Tform (خروجی imtransform) یک استراکچر (structure) است

۳-۱-۵-۱ ترمیم دو تصویر و اتصال آنها به هم

ب) حال فرض کنید بخواهیم تصویر دومی (ادامه تصویر قبلی) را نیز ترمیم و این دو تصویر را به هم متصل کنیم. مختصات ورودی و خروجی تصویر دوم بر روی آن نوشته شده اند.

شکل ۳-۷: اثر پرسپکتیو در یک تصویر ترمیم نشده

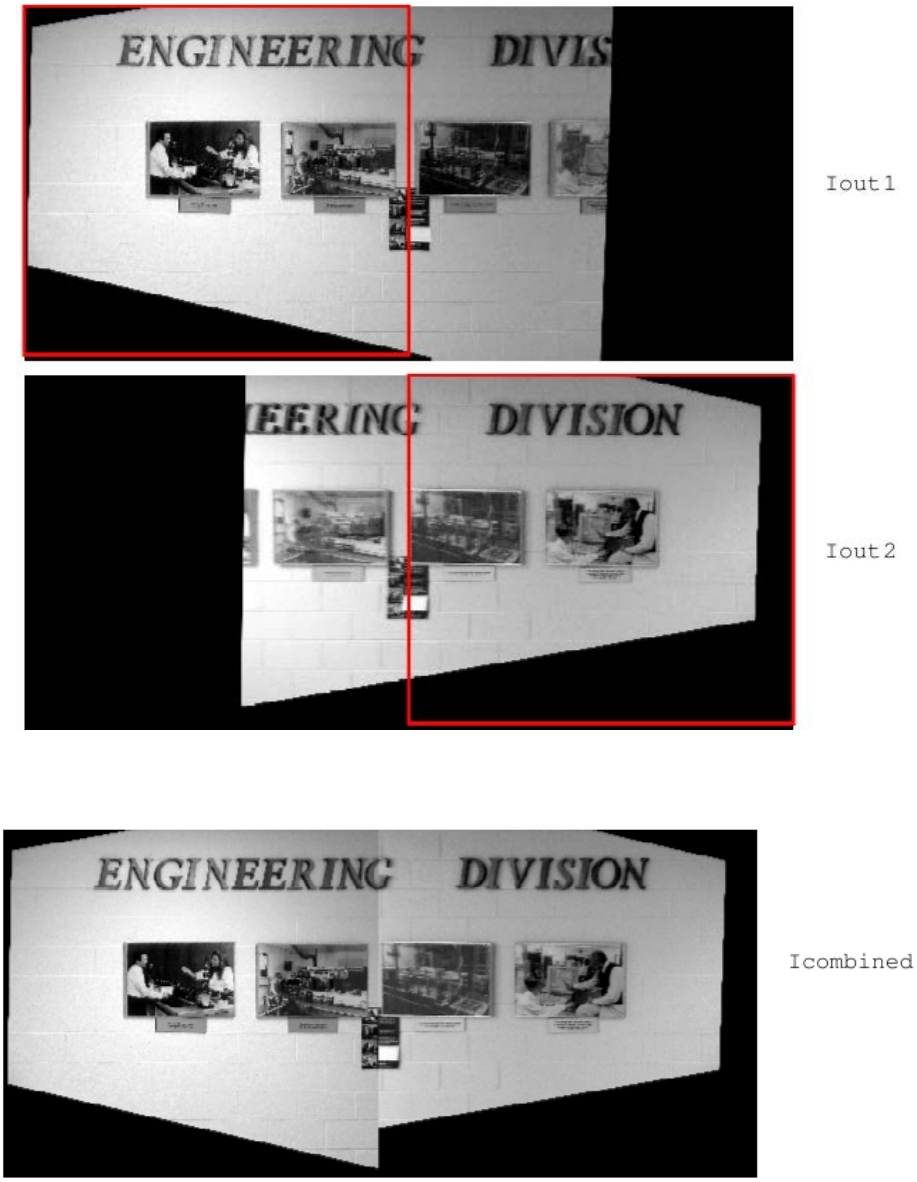
حل: برای این منظور لازم است تا تصویر دوم را نیز ترمیم نموده و سپس دو تصویر را با هم ادغام (merge) کنیم. البته برای اینکه این کار به خوبی انجام شود باید هر دو تصویر به یک سیستم مختصات مشخص تبدیل شوند. برای این منظور باید در دستور imtransform محدوده تصویر خروجی را دقیقا تعیین کنیم. در imtransform دو بخش Xdata و Ydata وجود دارد. Xdata، مختصه x اولین و آخرین ستون تصویر خروجی را تعیین می کند. به همین ترتیب Ydata مختصه Y اولین و آخرین سطر تصویر خروجی را. بنابراین ما باید برای هر دو تصویر فوق، یک محدوده را در تصویر خروجی تعیین کنیم. یعنی:

```
Iout1 = imtransform(Iin1, Tform1, ...
 'XData',[-50 550], ...
'YData',[-25 250]);
Iout2 = imtransform(Iin2, Tform2, ...
 'XData',[-50 550], ...
'YData',[-25 250]);
```

و در نهایت دو تصویر را با هم ادغام کنیم. یعنی:

```
[H,W] = size(Iout2);
Icombined = [Iout1(:,1:floor(W/2)) Iout2(:,floor(W/2)+1:end) ];
```

در زیر تصاویر خروجی و تصویر تلفیق شده نهایی دیده می شوند.



شکل ۳-۸: اتصال دو تصویر ترمیم شده

۴ SVD

در این درس می‌خواهیم راجع به SVD (Singular Value Decomposition) و کاربرد آن در حل معادلات صحبت کنیم. SVD یک روش حل ماتریسی است که دارای کاربردهای متنوعی در کامپیوتر ویژن می‌باشد. دو نمونه مهم از کاربردهای آن عبارتند از:

- حل یک سری معادلات خطی هموژن در فرم Ax=0
- اطمینان حاصل کردن از اینکه مقادیر محاسبه شده برای المان‌های یک ماتریس از شرط خاصی تبعیت می‌کنند. مثلاً وقتی ماتریس R را محاسبه می‌کنیم و می‌خواهیم ببینیم که آیا ماتریس حاصله متعامد هست یا نه؟

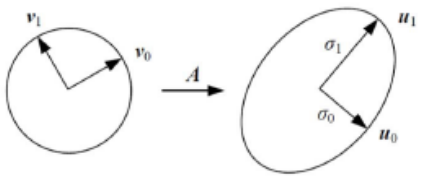
حال ببینیم SVD چیست. SVD در حقیقت یک روش تجزیه ماتریسی است. در ریاضیات نشان داده می‌شود که هر ماتریس MxN را می‌توان به این صورت حاصل ضرب سه ماتریس به این شکل نوشت:. $A = U D V^T$. در این رابطه، ماتریس U و V ماتریس‌هایی با ستون‌هایی‌اند که در حقیقت بردارهای یکه می‌باشند. ماتریس‌های U و V از این شرط تبعیت می‌کنند: $\mathbf{U}^T\mathbf{U} = \mathbf{I}, \mathbf{V}^T\mathbf{V} = \mathbf{I}$. به عبارت دیگر $\mathbf{u}_i \cdot \mathbf{u}_j = \mathbf{v}_i \cdot \mathbf{v}_j = \delta_{ij}$ که در آن ij=0□ اگر i مساوی با j نباشد و نیز مساوی با ۱ است اگر i مساوی با j باشد. D یک ماتریس قطری است که درایه های قطر آن مقادیر σi می‌باشند که به ترتیب $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_n \geq 0$ در D قرار داده شد اند و به آن‌ها مقادیر سینگولار (singular values) گفته می‌شود.

بنابراین اگر ما یک ماتریس A_{MxN} داشته باشیم می‌توانیم بنویسیم:

$$\begin{aligned} \boldsymbol{A}_{M\times N} &= \boldsymbol{U}_{M\times P}\boldsymbol{\Sigma}_{P\times P}\boldsymbol{V}^T_{P\times N} \\ &= \left[\begin{array}{c|c|c} \boldsymbol{u}_0 & \cdots & \boldsymbol{u}_{p-1} \end{array}\right] \begin{bmatrix} \sigma_0 & & \\ & \ddots & \\ & & \sigma_{p-1} \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_0^T \\ \hline \cdots \\ \hline \boldsymbol{v}_{p-1}^T \end{bmatrix} \end{aligned}$$

در رابطه بالا توجه شود که المان های V^T سطری اند چون ترانسپوز v می‌باشند).

تکنیک SVD دارای ویژگی هایی است که در اینجا به اختصار به آنها اشاره می شود. همان طور که دیدیم $A = U D V^T$. در این صورت با ضرب V از سمت راست در هر دو طرف رابطه داریم: AV=UD (توجه شود که $V^TV = I$) یا $A v_j = \sigma_j u_j$ (نکته: در اینجا اندیس j قرار داده شده است چرا که ماتریس‌ها ستون به ستون فرق می‌کنند). پس می‌بینیم که می‌توان با استفاده از A بردار v_j را به یک جهت u_j با طول σ_j تبدیل نمود یعنی:



شکل ۵-۱: تبدیل v_j را به یک جهت u_j با طول σ_j

حال اگر طرفین را در V^T مجدداً ضرب کنیم داریم:

$$\mathbf{A} = \sum_{j=0}^{p-1} \sigma_j \mathbf{u}_j \mathbf{v}_j^T$$

که به بردارهای u_j ، جزء اصلی A (principal components of A) گفته می‌شود. همانطور که دیدیم $\mathbf{A} = \mathbf{U}\,\mathbf{D}\,\mathbf{V}^T$. حال اگر از طرفین ترانهاده گرفته و حاصل را در طرفین رابطه بالا ضرب کنیم خواهیم داشت:

$$\mathbf{A}\,\mathbf{A}^T = (\mathbf{U}\,\mathbf{D}\,\mathbf{V}^T)\,(\mathbf{U}\,\mathbf{D}\,\mathbf{V}^T)^T = \mathbf{U}\,\mathbf{D}\,\mathbf{V}^T\mathbf{V}\,\mathbf{D}\,\mathbf{U}^T = \mathbf{U}\,\Lambda\,\mathbf{U}^T$$

در این رابطه Λ حاصلضرب D.D است که المان‌های آن یعنی λ_i برابراند با $\lambda_i = \sigma_i^2$. حال اگر طرفین را در U ضرب کنیم داریم $(\mathbf{A}\,\mathbf{A}^T)\,\mathbf{U} = \mathbf{U}\,\Lambda$ یا $(\mathbf{A}\,\mathbf{A}^T)\,\mathbf{u}_j = \lambda_j\,\mathbf{u}_j$. بنابراین دیده می شود که ضرب $\mathbf{A}\,\mathbf{A}^T$ در بردار ستونی u_j برابر است با λ_i در همان بردار ستونی و این در حقیقت همان تعریف مقادیر و بردارهای ویژه می‌باشد. به عبارت بهتر u_j ها در حقیقت بردارهای ویژه (eigenvectors) ماتریس $\mathbf{A}\,\mathbf{A}^T$ و λ_i ها همان مقادیر ویژه (eigenvalues) آن می‌باشند. به همین ترتیب در صورتی که از $\mathbf{A} = \mathbf{U}\,\mathbf{D}\,\mathbf{V}^T$ ترانهاده گرفته، و از سمت چپ در طرفین ضرب کنیم داریم:

$$\mathbf{A}^T\,\mathbf{A} = (\mathbf{U}\,\mathbf{D}\,\mathbf{V}^T)^T\,(\mathbf{U}\,\mathbf{D}\,\mathbf{V}^T) = \mathbf{V}\,\mathbf{D}\,\mathbf{U}^T\mathbf{U}\,\mathbf{D}\,\mathbf{V}^T = \mathbf{V}\,\Lambda\,\mathbf{V}^T$$

که با ضرب V از سمت راست در طرفین داریم $(\mathbf{A}^T\,\mathbf{A})\,\mathbf{V} = \mathbf{V}\,\Lambda$ یا $(\mathbf{A}^T\,\mathbf{A})\,\mathbf{v}_j = \lambda_j\,\mathbf{v}_j$. بنابراین دیده می‌شود که ستون‌های V در حقیقت بردارهای ویژه $\mathbf{A}\,\mathbf{A}^T$ و λ_jها مقادیر ویژه آن می‌باشند.حال ببینیم با توجه به این مطالب چگونه می‌توان معادلات هموژن را حل نمود.

۱-٤ حل یک سیستم معادلات هموژن:

فرض کنیم داریم Ax=0 که بوسیله m معادلات و n مجهول تشکیل شده اند و نیز داریم $m >= n-1$ و رنک A برابر با n-1 می باشد یعنی $\text{rank}(\mathbf{A})=n-1$. توجه کنید در چنین حالتی هر مضربی از x نیز یک جواب است. حتی x=0 نیز یکی از جواب‌ها است که البته ما علاقه‌ای به آن نداریم. خواهیم دید که جواب ما بردار ویژه تنها مقدار ویژه ای از $\mathbf{A}\,\mathbf{A}^T$می‌باشد که مقدار آن برابر صفر است. برای اثبات این

موضوع می‌دانیم که ما باید نرم Ax یعنی $\|\mathbf{Ax}\|^2$ را مینیمم کنیم که در آن $\|\mathbf{Ax}\|^2 = (\mathbf{Ax})^T\mathbf{Ax} = \mathbf{x}^T\mathbf{A}^T\mathbf{Ax}$.
با توجه به اینکه هر مضربی از x جواب است، ما حالتی را انتخاب می کنیم که $\mathbf{x}^T\mathbf{x} = 1$. بنابراین برای اینکه این شرط بالا مینیمم شده و شرط $\mathbf{x}^T\mathbf{x} = 1$ نیز صادق باشد، مقداری که باید مینیمم شود عبارت است از:

$$L(\mathbf{x}) = \mathbf{x}^T\mathbf{A}^T\mathbf{Ax} - \lambda(\mathbf{x}^T\mathbf{x} - 1)$$

که در آن λ یک مقدار ثابت است و به آن، مضرب لاگرانژ گفته می‌شود. برای مینیمم کردن رابطه فوق، از آن نسبت به x مشتق گرفته و برابر با صفر قرار می‌دهیم یعنی:

$$\mathbf{A}^T\mathbf{Ax} - \lambda\mathbf{x} = 0 \text{ و یا } \mathbf{A}^T\mathbf{Ax} = \lambda\mathbf{x}$$

پس همان طور که می‌بینیم λ، مقدار ویژه A^TA است و x نیز که بردار ویژه A^TA می‌باشد. حال سوالی که وجود دارد آنست که مقدار λ چیست؟ برای به دست آوردن آن باید $L(\mathbf{e}_\lambda) = \lambda$ مینیمم شود که این در حالتی اتفاق می‌افتد که λ=0 باشد. بنابراین $\mathbf{x} = \mathbf{e}_0$ بردار ویژه مربوط به مقدار ویژه صفر می‌باشد.

تمرین:

فرض کنید $\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ باشد و داشته باشیم Ax=0 در این صورت x را به دست آورید.

حل: می‌دانیم x یک بردار ۳x۱ است. حال اگر A^TA را به دست آوریم داریم:

$$\mathbf{A}^T\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

حال اگر مقادیر ویژه و بردارهای ویژه ATA را به دست آوریم داریم:

$$\lambda_1 = 0, \mathbf{e}_1 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad \lambda_2 = 1, \mathbf{e}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad \lambda_3 = 1, \mathbf{e}_3 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

همانطور که گفتیم بردار ویژه ای که مقدار ویژه آن برابر صفر باشد (یعنی $\lambda_1=0$) همان جواب ما است یعنی: $x=e_1$. برای بررسی صحت این موضوع کافی است که Ax را به دست آوریم یعنی:

$$\mathbf{Ax}=\begin{pmatrix}1&0&0\\0&1&0\end{pmatrix}\begin{pmatrix}0\\0\\1\end{pmatrix}=\begin{pmatrix}0\\0\end{pmatrix}=\mathbf{0}$$

پس می‌بینیم که رابطه Ax=0 صادق استذ و این فقط در حالی اتفاق افتاده است که $x=e_1$ به عبارت دیگر اگر x را مساوی e_2 یا e_3 قرار دهیم رابطه Ax=0 صادق نیست.

حال ببینیم چگونه با SVD معادلات Ax=0 را حل می کنیم. گفتیم که x بردار ویژه مقدار ویژه ای است که اندازه آن برابر صفر باشد. حال اگر از A، SVD بگیریم داریم: $\mathbf{A}=\mathbf{U}\,\mathbf{D}\,\mathbf{V}^T$. گفتیم که ستون‌های V ، بردارهای ویژه A^TA اند. ضمناً x ستونی از V است که مقدار ویژه آن صفر است. بنابراین با توجه به اینکه ستون‌های V به ترتیب در V قرار داده می شوند، x در حقیقت آخرین ستون V (اولین ستون از راست) می‌باشد.

تمرین: در تمرین قبل، x را از طریق SVD به دست آورید.

حل: گفتیم که $\mathbf{A}=\begin{pmatrix}1&0&0\\0&1&0\end{pmatrix}$. بنابراین با استفاده از متلب داریم:

$$\mathbf{A}=\mathbf{UDV}^T=\begin{pmatrix}1&0\\0&1\end{pmatrix}\begin{pmatrix}1&0&0\\0&1&0\end{pmatrix}\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}$$

همان طور که دیده می شود، ستون سوم V^T (اولین ستون از راست) یعنی همان $\begin{pmatrix}0\\0\\1\end{pmatrix}$ بردار x می باشد که در تمرین قبل به دست آمد.

تمرین: برنامه متلبی بنویسید که Ax=0 را با استفاده از SVD حل کند.

حل: با فرض معلوم بودن A داریم [U,D,V] = svd(A) با استفاده از این دستور ماتریس‌های U ، V و D تعیین می شوند. بنابراین می توان x را به دست آورد یعنی: x = V(:,end). در این دستور x همان ستون آخر V می باشد. برنامه زیر به طور کامل مراحل فوق را انجام می دهد:

```
clear all
close all
% Solve the system of equations Ax = 0
A = [ 1 0 0;
0 1 0 ];
[U,D,V] = svd(A);
x = V(:,end); % get last column of V
```

در صورت اجرای برنامه فوق، خروجی ها به شکل زیر می باشند:

```
>> U
U =
   1   0
   0   1
>> D
D =
   1   0   0
   0   1   0
```

```
>> V
V =
   1   0   0
   0   1   0
   0   0   1
>> x
x =
   0
   0
   1
```

شکل ۵-۲: خروجی های حل یک ماتریس از طریق SVD

۲-٤ اعمال شرایط اجباری به کمک SVD

همان طور که می دانیم المان های یک ماتریس به طور کلی از هم مستقل نیستند و برخی اوقات باید از شرط خاصی تبعیت کنند. به عنوان مثال اگر ماتریس A ما یک ماتریس دوران باشد، سطر و ستون های آن باید متعامد باشند. حال فرض کنیم که چنین شرطی دقیقا در المان های ماتریس رعایت نشده باشند. اگر این ماتریس را A' بنامیم، در این صورت به کمک SVD می توانیم بر اساس نرم ماتریس نزدیک ترین ماتریس (A) به A' را پیدا کنیم که شرط تعامد در آن صادق باشد. برای این منظور کافی است از ماتریس A' خود SVD بگیریم یعنی: $\mathbf{A'} = \mathbf{U\,D\,V}^T$. حال اگر شرط ما برقرار می بود، ما یک سری مقادیر سینگولار (Singular Values) می داشتیم. بنابراین در مرحله دوم، ماتریس D' ای تشکیل می دهیم که بر اساس این مقادیر شکل گرفته است. در این صورت $\mathbf{A} = \mathbf{U\,D'\,V}^T$ ماتریسی است که شرط مورد نظر در آن تامین شده و طبیعتاً نزدیکترین ماتریس به A می باشد.

تمرین: فرض کنید دوران های یک ماتریس دوران را داشته باشید. در این صورت برنامه ای بنویسید که ماتریس دوران را حساب نموده و آن را به صورت رندوم به مقدار کمی تغییر دهد. سپس با استفاده از تکنیک SVD و اعمال شرط تعامد، نزدیکترین تقریب R را به دست آورید.

حل: برنامه زیر این کار را انجام می دهد. در ابتدا با استفاده از مقادیر دوران ماتریس R حساب می شود. سپس با استفاده از دستور randn مقادیر R به میزان کمی تغییر می یابند. ماتریس حاصل Rp نامیده شده است. سپس برای به دست آوردن نزدیکترین تقریب R که در این برنامه Rc اطلاق شده است، از Rp، SVD گرفته شده است. در این مرحله، ماتریس های U، D و V به دست آمده اند. حال، می دانیم اگر R متقارن می بود، ماتریس D باید یک ماتریس قطری که المان های قطر آن برابر با ۱ اند می بود. پس برای اعمال شرط تعامد، به جای D از D' که یک ماتریس قطری است و المان های قطر آن همه ۱ اند استفاده کرده و ماتریس Rc یا همان نزدیکترین تقریب R را به دست می آوریم. یادآور می شود که با استفاده از دستور eye(3,3) می توان یک ماتریس قطری ساخت که در آن همه المان های غیر قطر برابر با صفر بوده و المان های قطر آن برابر با ۱ می باشند.

برنامه زیر گفته های بالا را اجرا می کند:

```
clear all
close all
% Make a valid rotation matrix
ax = 0.1; ay = -0.2; az = 0.3; % radians
Rx = [ 1 0 0; 0 cos(ax) -sin(ax); 0 sin(ax) cos(ax)];
Ry = [ cos(ay) 0 sin(ay); 0 1 0; -sin(ay) 0 cos(ay)];
Rz = [ cos(az) -sin(az) 0; sin(az) cos(az) 0; 0 0 1];
R = Rz * Ry * Rx
% Ok, perturb the elements of R a little
Rp = R + 0.01*randn(3,3)
[U,D,V] = svd(Rp); % Take SVD of Rp
D % Here is the actual matrix of singular values
% Recover a valid rotation matrix by enforcing constraints
Rc = U * eye(3,3) * V'
```

---

۵ Linear Pose Estimation

در اینجا می‌خواهیم راجع به DLT یا Direct Linear Transform صحبت کنیم. هدف در اینجا محاسبه المان‌های ماتریس پروژکتیو به صورت مستقیم می‌باشد. در دروس گذشته دیدیم که اگر بخواهیم

مختصات تصویری (x_{im},y_{im}) یک نقطه $^{W}\mathbf{P}$ را به دست آوریم داریم: $\tilde{\mathbf{p}} = \mathbf{K}\mathbf{M}_{ext}\,{}^{W}\mathbf{P}$. یا به عبارت بهتر:

$$\tilde{\mathbf{p}} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \mathbf{K}\mathbf{M}_{ext}\,{}^{W}\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}.$$

که در آن: $x_{im} = x_1 / x_3,\ y_{im} = x_2 / x_3$ و همچنین داریم:

$$\mathbf{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \quad \text{و نیز} \quad \mathbf{M}_{ext} = \left({}^{C}_{W}\mathbf{R} \quad {}^{C}\mathbf{t}_{Worg} \right) = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_X \\ r_{21} & r_{22} & r_{23} & t_Y \\ r_{31} & r_{32} & r_{33} & t_Z \end{pmatrix}$$

در روش DLT ما بدون توجه به وابستگی پارامترهای r_{11} تا r_{33} آن‌ها را به صورت مستقیم به دست می‌آوریم. البته برای این منظور ما باید از مفهومی به نام مختصات نرمال شده تصویری (Normalised Image Coordinates) استفاده کنیم. برای این منظور با فرض معلوم بودن المان داخلی دوربین، ما می‌توانیم نقاط تصویر را نرمال کنیم. سیستم مختصات نقاط نرمال شده سیستمی است که در آن:

- مبدأ سیستم مختصات در مرکز تصویر است.
- f=1

در این صورت مختصات نرمال شده عبارتند از :

$$x_{normalized} = X/Z,\ y_{normalized} = Y/Z$$

نکته: رابطه مختصات اصلی (غیر نرمال) و نرمال شده عبارت است از:

$$\mathbf{p}_{normalized} = (\mathbf{K})^{-1}\,\mathbf{p}_{unnormalized} \quad \text{یا} \quad \mathbf{p}_{unnormalized} = \mathbf{K}\,\mathbf{p}_{normalized}$$

که در آن K ماتریس المان‌های داخلی دوربین می‌باشد. در روش DLT ما در حقیقت یک نقطه $^{W}\mathbf{P}$ را به موقعیت نرمال شده تصویری آن تبدیل می‌کنیم. بنابراین در اینجا داریم:

$$\tilde{\mathbf{p}}_n = \mathbf{M}_{ext}\mathbf{P} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{32} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

با توجه به اینکه $\tilde{\mathbf{p}}_n$ مختصات هموژن تصویری است باید برای به دست آوردن مختصات تصویری x_1 و x_2 را به x_3 تقسیم کنیم. بنابراین پس از ضرب ماتریس‌های سمت راست رابطه و تقسیم سطرهای اول و دوم به سطر سوم خواهیم داشت:

$$x = \frac{r_{11}X + r_{12}Y + r_{13}Z + t_x}{r_{31}X + r_{32}Y + r_{33}Z + t_z}, \quad y = \frac{r_{21}X + r_{22}Y + r_{23}Z + t_y}{r_{31}X + r_{32}Y + r_{33}Z + t_z}$$

حال اگر دو رابطه بالا را طرفین وسطین کنیم داریم:

$$x(r_{31}X + r_{32}Y + r_{33}Z + t_z) = r_{11}X + r_{12}Y + r_{13}Z + t_x$$
$$y(r_{31}X + r_{32}Y + r_{33}Z + t_z) = r_{21}X + r_{22}Y + r_{23}Z + t_y$$

حال برای اینکه این رابطه را حل کنیم باید آن را در فرم Ax=0 قرار دهیم که در آن x در برگیرنده r_{11} تا r_{33} و t_x ، t_y و t_z می‌باشد. در اینجا داریم:

$$\mathbf{Ax} = \begin{pmatrix} X & Y & Z & 0 & 0 & 0 & -xX & -xY & -xZ & 1 & 0 & -x \\ 0 & 0 & 0 & X & Y & Z & -yX & -yY & -yZ & 0 & 1 & -y \end{pmatrix} \begin{pmatrix} r_{11} \\ r_{12} \\ r_{13} \\ r_{21} \\ r_{22} \\ r_{23} \\ r_{31} \\ r_{32} \\ r_{33} \\ t_x \\ t_y \\ t_z \end{pmatrix} = \mathbf{0}$$

با توجه به اینکه ۱۲ مجهول داریم و هر نقطه ۲ معادله می‌دهد، حداقل ۶ نقطه نیاز داریم تا مجهولات را به دست آوریم.

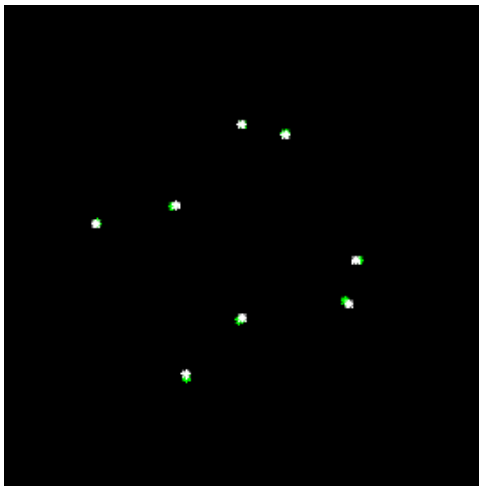
تمرین: برنامه‌ای بنویسید که براساس مقادیر فرضی دوربین (f=512) و نیز $C_x=C_y=256$ و داشتن مختصات زمینی ۸ نقطه، موقعیت نقاط تصویری را بدست آورده و روی تصویر نشان دهد. سپس همین مقادیر را بوسیله DLT بدست آورده و روی تصویر نشان دهید.

حل: در زیر این برنامه دیده می شود.

```
clear all
close all
% Create camera matrix
f = 512; % focal length in pixels
cx = 256;
cy = 256;
K = [ f 0 cx; 0 f cy; 0 0 1 ]; % intrinsic parameter matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Create input data
% This is used to reset the random number generator to the same
sequence
s = RandStream('swb2712','Seed',0);
RandStream.setDefaultStream(s);
% Create known 3D points (at least 6)
N = 8;
P_M = [
1 -1 1 -1 1 -1 1 -1;
1 1 -1 -1 1 1 -1 -1;
1 1 1 1 -1 -1 -1 -1;
1 1 1 1 1 1 1 1];
% Create true model-to-camera transform
ax = 0; ay = 20*pi/180; az = -30*pi/180;
tx = 0; ty = 0; tz = 6;
Rx = [ 1 0 0; 0 cos(ax) -sin(ax); 0 sin(ax) cos(ax)];
Ry = [ cos(ay) 0 sin(ay); 0 1 0; -sin(ay) 0 cos(ay)];
Rz = [ cos(az) -sin(az) 0; sin(az) cos(az) 0; 0 0 1];
Rtrue = Rz * Ry * Rx;
% Project points onto image
Mtrue = [ Rtrue [tx;ty;tz] ];
disp('Ground truth pose (Mtrue):'); disp(Mtrue);
% Project model points onto image
p = K*Mtrue*P_M;
p(1,:) = p(1,:)./p(3,:);
p(2,:) = p(2,:)./p(3,:);
p(3,:) = p(3,:)./p(3,:);
% Display input data
disp('Known model points:'); disp(P_M);
disp('Measured image points:'); disp(p);
I = zeros(512,512);
imshow(I);
hold on
plot(p(1,:), p(2,:), 'g*');

% Add some noise to the image points
p(1:2,:) = p(1:2,:) + 2.0*randn(2,N);
plot(p(1,:), p(2,:), 'w*');
```

برای انجام این برنامه در بخش اول با داشتن المان‌های داخلی، ماتریس K ساخته می‌شود. سپس رئوس یک مکعب را به عنوان نقاط مدل یعنی P_M معرفی می‌کنیم.در بخش دوم با معرفی زوایا و شیفت نقاط، R و در نهایت ترانسفورماسیون مدل به دوربین (که در اینجا Mtrue نامیده شده) را تشکیل داده و با استفاده از دستور disp نقاط Mtrue را نشان می‌دهیم. سپس با استفاده از رابطه P=K*Mtrue * P_M نقاط مدل را بر روی تصویر تبدیل و با تقسیم بر سطر سوم نرمال می‌کنیم.

در مرحله بعد، برای دیدن مشاهدات فرضی با دستور disp (P_M) در ابتدا مختصات مدل و سپس با دستور disp(p) مختصات تصویری را نشان می‌دهیم. سپس یک تصویر I به ابعاد 512×512 ساخته (با دستور I=zeros(512,512)) و آن را نشان داده (imshow(I)) و روی آن مختصات محاسبه شده p را با دستور plot نشان می‌دهیم یعنی (plot(p(1,:), p(2,:)), 'g*') که در آن 'g*' برای نشان دادن به رنگ سبز می باشد. برای اینکه مقداری خط به نقاط اضافه کنیم از دستور randn استفاده می‌کنیم و نقاط را با رنگ سفید ترسیم می‌کنیم. با اجرای برنامه بالا شکل زیر را خواهیم داشت:



شکل ۶-۱: موقعیت صحیح (سبز) و خطادار (سفید) نقاط در یک تبدیل DLT

۵-۱ به دست آوردن ضرایب DLT

برای نوشتن برنامه DLT گفتیم که نقاط باید نرمال شده باشند. لذا با ضرب نقاط تصویری خطا دار بالا در معکوس ماتریس K ، نقاط نرمال شده را به دست آورده و نمایش می‌دهیم. حال برای به دست آوردن مقادیر از طریق DLT همانطور که گفتیم اولین مرحله تشکیل ماتریس‌های Ax=0 می‌باشد. لذا با توجه به شکل A که قبلاً برای DLT گفتیم چگونه است، با داشتن مقادیر مختصات تصویری اندازه

گیری شده x (همان نقاط خطادار بالا) و نیز مختصات زمینی نقاط (همان نقاط مکعب) ماتریس A را تشکیل می‌دهیم. در این مرحله باید با استفاده از SVD ، x را بدست آوریم. برای این منظور دیدیم که کافی است با استفاده از SVD ماتریس‌های D،U و V را از طریق `[U,D,V] = svd(A)` به دست آوریم. در این صورت `x = V(:,end)` است که در آن المان‌های x آخرین ستون V خواهند بود. البته همانطور که می‌دانیم x در حقیقت المان‌های ماتریس تبدیل ما می باشند که در اینجا آن را M می‌نامیم بنابراین باید در اولین مرحله، M را تشکیل دهیم یعنی:

```
M = [ x(1)  x(2)  x(3)  x(10);
      x(4)  x(5)  x(6)  x(11);
      x(7)  x(8)  x(9)  x(12) ];
```

در ماتریس بالا: x_1 تا x_9 المان‌های ماتریس دوران و x_{10} تا x_{12} شیفت های ما می‌باشند.

```
% Normalize image points
pn = inv(K)*p;

% Ok, now we have pn = Mext*P_M.
% If we know P_M and pn, we can solve for the elements of Mext.
% The equations for x,y are:
% x = (r11*X + r12*Y + r13*Z + tx)/(r31*X + r32*Y + r33*Z + tz)
% y = (r21*X + r22*Y + r23*Z + ty)/(r31*X + r32*Y + r33*Z + tz)
% or
% r11*X + r12*Y + r13*Z + tx - x*r31*X - x*r32*Y - x*r33*Z - x*tz = 0
% r21*X + r22*Y + r23*Z + ty - y*r31*X - y*r32*Y - y*r33*Z - y*tz = 0
% Put elements of Mext into vector w:
% w = [r11 r12 r13 r21 r22 r23 r31 r32 r33 tx ty tz]
% We then have Ax = 0. The rows of A are:
% X Y Z 0 0 0 -x*X -x*Y -x*Z 1 0 -x
% 0 0 0 X Y Z -y*X -y*Y -y*Z 1 0 -y
A = zeros(N,12);
for i=1:N
X = P_M(1,i); Y = P_M(2,i); Z = P_M(3,i);
x = pn(1,i); y = pn(2,i);
A( 2*(i-1)+1, :) = [ X Y Z 0 0 0 -x*X -x*Y -x*Z 1 0 -x ];
A( 2*(i-1)+2, :) = [ 0 0 0 X Y Z -y*X -y*Y -y*Z 0 1 -y ];
end
% Solve for the value of x that satisfies Ax = 0.
% The solution to Ax=0 is the singular vector of A corresponding to
the
% smallest singular value; that is, the last column of V in A=UDV'
[U,D,V] = svd(A);
x = V(:,end); % get last column of V
% Reshape x back to a 3x4 matrix, M = [R t]
M = [ x(1) x(2) x(3) x(10);
x(4) x(5) x(6) x(11);
x(7) x(8) x(9) x(12) ];
% here is the rotation and translation portion
R = M(1:3,1:3);
```

```
t = M(1:3, 4);
```

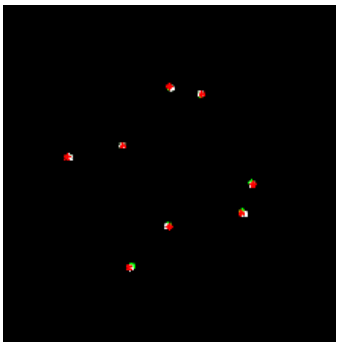
۵-۲ به دست آوردن مقیاس

نکته ای که وجود دارد آنست که جوابهایی که تا کنون به دست آورده ایم در مقیاس واقعی نیستند. پس لازم است تا مقیاس x را نیز پیدا کنیم. در اینجا می‌توانیم از دو شرط استفاده کنیم. اول آنکه دترمینان R باید برابر با 1 باشد. ثانیاً شرط اینکه ماتریس R دارای مقادیر ویژه یکسان و برابر با ۱ را اعمال کنیم. نهایتاً برای اینکه مقیاس شیفت را نیز درست کنیم از trace (D)/3 استفاده کرده و المان‌های t را بر آن تقسیم می‌کنیم. و نهایتاً نقاط را به دست آورده و تصویر می‌کنیم و نمایش می‌دهیم.

در برنامه زیر مراحل بالا نشان داده شده اند

```
% R must be a right handed rotation matrix; ie det(R)>0
if det(R)<0
R = -R;
t = -t;
end
% Enforce fact that R has equal singular values, and they are = 1
[U,D,V] = svd(R);
R = U*diag([1 1 1])*V';
myscale = trace(D)/3; % Also scale translation portion
M = [R t/myscale];
disp('Derived pose (M):'); disp(M);
% Reproject points back onto the image
p = K*M*P_M;
p(1,:) = p(1,:)./p(3,:);
p(2,:) = p(2,:)./p(3,:);
p(3,:) = p(3,:)./p(3,:);
plot(p(1,:), p(2,:), 'r*');
```

با اجرای برنامه بالا خواهیم داشت:



شکل ۶-۲: موقعیت نقاط پس از اعمال مقیاس

۵-۳ محاسبه خطا:

حال ببینیم خطای محاسبات چقدر است. طبیعتاً این خطا دو بخش دارد یکی مربوط به شیفت و یکی مربوط به دوران. در خصوص شیفت که موضوع بسیار آسان است چرا که کافی است که نرم رابطه زیر را به دست آوریم:

$$\Delta \mathbf{t} = \mathbf{t}_{estimated} - \mathbf{t}_{groundtruth}$$

اما در مورد R وضعیت کمی پیچیده تر است چون ما در اینجا سه دوران داریم. کاری که در اینجا می‌توان کرد آنست که ماتریس دورانی که وضعیت محاسبه شده را به وضعیت زمینی تبدیل می‌کند پیدا کنیم. در این راستا داریم:

$$\prescript{gt}{est}{R} = \prescript{gt}{w}{R}\ \prescript{w}{est}{R} = \prescript{w}{gt}{R}^{T}\ \prescript{w}{est}{R}$$

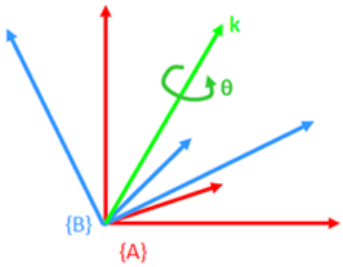
در حالت ایده آل $\prescript{gt}{est}{R} = \mathbf{I}$ ولی اگر این رابطه صادق نباشد به علت خطایی است که می خواهیم میزان آن را تخمین بزنیم. برای این موضوع می توانیم محور و زاویه معادل ماتریس دوران را پیدا و کنیم. در این راستا همان طور که قبلا دیدیم می توانیم به طور کلی دوران حول سه محور ثابت X، Y و Z را با یک دوران θ حول یک محور k نیز تعریف کرد که در آن داریم:

$$R_k(\theta) = \begin{pmatrix} k_x k_x v\theta + c\theta & k_x k_y v\theta - k_z s\theta & k_x k_z v\theta + k_y s\theta \\ k_x k_y v\theta + k_z s\theta & k_y k_y v\theta + c\theta & k_y k_z v\theta - k_x s\theta \\ k_x k_z v\theta - k_y s\theta & k_y k_z v\theta + k_x s\theta & k_z k_z v\theta + c\theta \end{pmatrix}$$

where

$$c\theta = \cos\theta,\ s\theta = \sin\theta,\ v\theta = 1 - \cos\theta$$

$$\hat{\mathbf{k}} = (k_x, k_y, k_z)^T$$



$$\theta = \operatorname{acos}\left(\frac{r_{11} + r_{22} + r_{33} - 1}{2}\right)$$

$$\hat{\mathbf{k}} = \frac{1}{2\sin\theta}\begin{pmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{pmatrix}$$

شکل ۶-۳: دوران و محور معادل یک ماتریس دوران

بنابراین می‌توانیم خطای دوران محاسبات را با توجه به میزان خطای θ تعریف کنیم. که آن هم چیزی نیست جز همان θ. پس به طور خلاصه کاری که باید انجام دهیم آنست که $\prescript{gt}{est}{R}$ را حساب کرده و سپس θ معادل آن که نشان دهنده میزان خطا است را به دست آوریم.

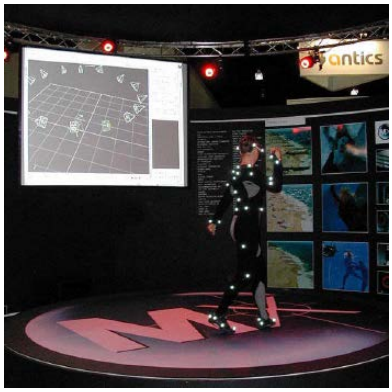
تمرین: برنامه ای بنویسید که خطای ناشی از محاسبه المان های توجیه خارجی را به دست آورد.

حل: برنامه فوق این کار را انجام می دهد:

```
% Evaluate error (from ground truth)
tdiff = M(1:3, 4) - Mtrue(1:3, 4);
fprintf('Translation error: %f\n', norm(tdiff));
Rdiff = R*Rtrue'; % Rotation between our answer and ground truth
ang = acos( (trace(Rdiff)-1)/2 );
fprintf('Rotation error (degrees): %f\n', ang*180/pi);
```

۵-٤ DLT for Motion Capture

آنچه تاکنون گفته‌ایم آن بود که چگونه با داشتن نقاط کنترل، المان‌های توجیه بوسیله DLT را حساب کنیم. حال ببینیم در صورتی که المان‌های توجیه دوربین‌ها معلوم بودند چگونه می‌توانیم مختصات نقاط از عارضه را به دست آوریم. می دانیم که به این کار تقاطع گفته می شود. یکی از کاربردهای چنین موضوعی در Motion Capture است که در آن فردی لباسی می پوشد که در برگیرنده تارگت های مختلفی است در حالی که دوربین‌ها در اطراف نصب شده اند. کاری که انجام می شود آنست که مختصات تصویری نقاط تارگت اندازه گیری و مختصات سه بعدی آن‌ها محاسبه می شود و از این طریق مدل حرکت فرد به دست می آید. چنین کاری در مطالعات بیومکانیکی و یا در تولید کارتون با کارکترهای واقعی کاربرد دارد. در زیر نمونه ای از تصاویر گرفته شده در چنین پروژه هایی نشان داده شده است.



شکل ۶-۴: اندازه گیری مختصات سه بعدی برای تعیین مدل حرکتی یک انسان

در هر صورت روند کلی کار به این شکل است:

۱. نصب دوربین‌ها

۲. کالیبراسیون دوربین‌ها برای به دست آوردن المان‌های داخلی و DLT

۳. قرار دادن تارگت ها بر روی بدن فرد
٤. تصویر برداری از تارگت های نصب شده و قرائت مختصات تصویری آن‌ها
۵. محاسبه مختصات سه بعدی تارگت ها.

حال ببینیم چگونه می‌توانیم مختصات تارگت ها را حساب کنیم. خوشبختانه مساله در اینجا ساده است. برای این منظور از همان معادلات DLT که قبلاً دیدیم استفاده می کنیم. یعنی:

$$x=\frac{r_{11}X+r_{12}Y+r_{13}Z+t_x}{r_{31}X+r_{32}Y+r_{33}Z+t_z},\quad y=\frac{r_{21}X+r_{22}Y+r_{23}Z+t_y}{r_{31}X+r_{32}Y+r_{33}Z+t_z}$$

که پس از طرفین وسطین کردن داریم:

$$\begin{aligned}&r_{11}X+r_{12}Y+r_{13}Z+t_x-x(r_{31}X+r_{32}Y+r_{33}Z+t_z)=0\\&r_{21}X+r_{22}Y+r_{23}Z+t_y-y(r_{31}X+r_{32}Y+r_{33}Z+t_z)=0\end{aligned}$$

البته در اینجا تمامی المان‌های $r_{11}\rightarrow r_{33}$ و نیز $t_x\rightarrow t_z$ معلوم و مجهولات ما مختصات (X,Y,Z) نقاط می‌باشند. پس می‌توانیم مجدّداً معلومات و مجهولات را در فرم Ax=0 قرار داده و با حل آن از طریق SVD ، مختصات نقاط را به دست آوریم. با توجه به اینکه هر نقطه دو معادله می‌دهد، با توجه به اینکه برای هر نقطه سه مجهول داریم کافی است که معادله هر نقطه در دو دوربین نوشته شود. فقط باید توجه داشت که المان‌های دوران و شیفت هر دوربین متفاوت می‌باشند.

اجازه بدهید ببینیم این کار به صورت دقیق چگونه انجام می شود. از روابط بالا داریم:

$$\begin{aligned}&(r_{11}-xr_{31})X+(r_{12}-xr_{32})Y+(r_{13}-xr_{33})Z=-t_x+xt_z\\&(r_{21}-yr_{31})X+(r_{22}-yr_{32})Y+(r_{23}-yr_{33})Z=-t_y+yt_z\end{aligned}$$

حال اگر بخواهیم این معادلات را در فرم Ax=0 قرار دهیم داریم:

$$\begin{pmatrix} r_{11} - x r_{31} & r_{12} - x\ r_{32} & r_{13} - x\ r_{33} \\ r_{21} - y r_{31} & r_{22} - y\ r_{32} & r_{23} - y\ r_{33} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} -t_x + x t_z \\ -t_y + y t_z \end{pmatrix}$$

توجه شود که این معادلات برای یک نقطه در یک دوربین می باشند. طبیعتاً فرم ماتریس می تواند برای دیگر نقاط تکمیل گردد فقط همان طور که در بالا نیز اشاره شد المان های خارجی (و در صورت نیاز داخلی) هر دوربین متفاوت می باشند. بنابراین برای دو دوربین داریم:

$$\begin{pmatrix} r_{11}^{(c1)} - x^{(c1)} r_{31}^{(c1)} & r_{12}^{(c1)} - x^{(c1)} r_{32}^{(c1)} & r_{13}^{(c1)} - x^{(c1)} r_{33}^{(c1)} \\ r_{11}^{(c1)} - x^{(c1)} r_{31}^{(c1)} & r_{12}^{(c1)} - x^{(c1)} r_{32}^{(c1)} & r_{13}^{(c1)} - x^{(c1)} r_{33}^{(c1)} \\ r_{11}^{(c2)} - x^{(c2)} r_{31}^{(c2)} & r_{12}^{(c2)} - x^{(c2)} r_{32}^{(c2)} & r_{13}^{(c2)} - x^{(c2)} r_{33}^{(c2)} \\ r_{11}^{(c2)} - x^{(c2)} r_{31}^{(c2)} & r_{12}^{(c2)} - x^{(c2)} r_{32}^{(c2)} & r_{13}^{(c2)} - x^{(c2)} r_{33}^{(c2)} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} -t_x^{(c1)} + x^{(c1)} t_z^{(c1)} \\ -t_y^{(c1)} + y^{(c1)} t_z^{(c1)} \\ -t_x^{(c2)} + x^{(c2)} t_z^{(c2)} \\ -t_y^{(c2)} + y^{(c2)} t_z^{(c2)} \end{pmatrix}$$

که در آن مثلا برای دوربین اول داریم:

$${}^{c1}_{w}H = \begin{pmatrix} r_{11}^{(c1)} & r_{12}^{(c1)} & r_{13}^{(c1)} & t_x^{(c1)} \\ r_{21}^{(c1)} & r_{22}^{(c1)} & r_{23}^{(c1)} & t_y^{(c1)} \\ r_{31}^{(c1)} & r_{32}^{(c1)} & r_{33}^{(c1)} & t_z^{(c1)} \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \text{image point in camera 1} \begin{pmatrix} x^{(c1)} \\ y^{(c1)} \end{pmatrix}$$

معادلات بالا فرم ماتریس های Ax=L را تشکیل می دهند که با استفاده از آنها مختصات سه بعدی نقاط را می توان محاسبه نمود.

تمرین: برنامه ای بنویسید که دو تصویر مجازی از تعدادی نقطه زمینی را تولید نموده و سپس با استفاده از معادلات DLT مختصات آنها را مجددا به دست آورده و بر روی تصاویر اولیه بیندازد.

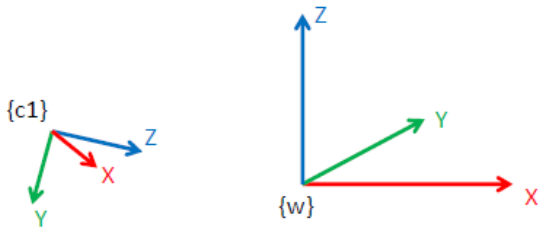
حل: حل این برنامه در مراحل مختلفی انجام می شود که در ادامه می آید.

- تولید تعدادی نقاط کنترل و نمایش آنها: در اولین مرحله تعدادی نقطه زمینی در اطراف مرکز مختصات تولید می کنیم. برای این منظور از سیستم قطبی استفاده شده است. برنامه زیر این کار را انجام می دهد:

```
clear all
close all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Create some 3D points in the world
N = 10;
P_w = zeros(4,N);
for i=1:N
r = 2 - i/N;
z = 2*i/N;
a = 4*pi*i/N;
P_w(1,i) = r*cos(a);
P_w(2,i) = r*sin(a);
P_w(3,i) = z;
P_w(4,i) = 1;
end
plot3(P_w(1,:), P_w(2,:), P_w(3,:), 'o-');
axis equal
axis vis3d
```

- تعیین موقعیت و دوران دوربین ها:. برای این منظور در دو نقطه فرضی از طریق تعیین شیفت های (t) موقعیت دوربین ها را مشخص می کنیم. فرض می کنیم برای دوربین اول داریم: (tx,ty,tz) = (-3, -6, 5) و برای دوربین دوم نیز داریم: (tx,ty,tz) = (+3, -6, 5). حال باید توجیه دورانی دوربین ها را تعیین کنیم. با توجه به اینکه نقاط زمینی در اطراف مرکز مختصات تعریف شده اند، فرض می کنیم که دوربین ها به سمت مرکز سیستم مختصات نشانه روی کرده اند. شکل زیر این موضوع را برای دوربین اول نشان می دهد:



شکل ۶-۵: سیستم مختصات جهانی و سیستم مختصات دوربین شبیه سازی شده

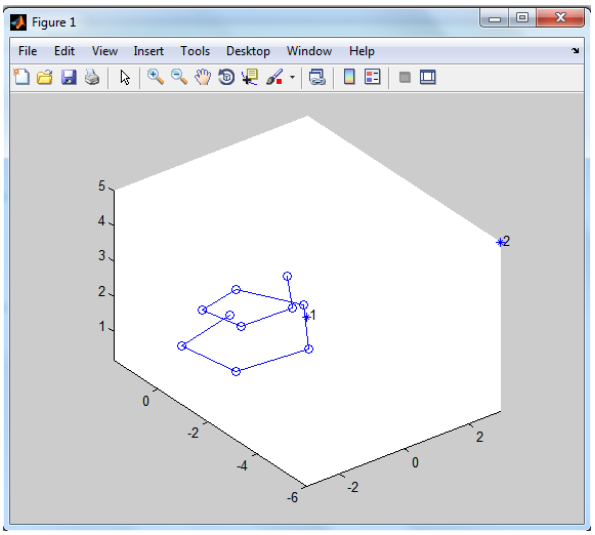
در این حالت باید R را به دست آوریم یعنی مثلا برای دوربین اول داریم: ${}^{w}_{c1}R = \left({}^{w}\hat{\mathbf{x}}_{c1} \quad {}^{w}\hat{\mathbf{y}}_{c1} \quad {}^{w}\hat{\mathbf{z}}_{c1}\right)$. از آنجائی که محور z به سمت مرکز است، طبیعتاً بردار واحد آن یعنی ${}^{w}\hat{\mathbf{z}}_{c1} = \frac{-{}^{w}\mathbf{t}_{c1org}}{\left|{}^{w}\mathbf{t}_{c1org}\right|}$ محور z را مشخص می کند. حال ببینیم که دو محور دیگر را چگونه می توانیم تعیین کنیم. فرض کنیم که محور x دوربین در صفحه x,y جهانی (w) قرار دارد. در نتیجه ضرب خارجی (cross product) محور z دوربین و z سیستم مختصات

جهانی، محور x دوربین را مشخص خواهد کرد یعنی: ${}^{w}\hat{\mathbf{x}}_{c1} = \frac{{}^{w}\hat{\mathbf{z}}_{c1} \times {}^{w}\hat{\mathbf{z}}_{w}}{\left|{}^{w}\hat{\mathbf{z}}_{c1} \times {}^{w}\hat{\mathbf{z}}_{w}\right|}$. در نهایت می توان از رابطه ${}^{w}\hat{\mathbf{y}}_{c1} = {}^{w}\hat{\mathbf{z}}_{c1} \times {}^{w}\hat{\mathbf{x}}_{c1}$ نیز محور y دوربین را تعریف نمود. به همین ترتیب می توانیم محورهای دوربین دوم را به دست آوریم. همان طور که دیدیم با استفاده از بردار یکه محورها می توان المان های ماتریس دوران را مشخص نمود.

برنامه زیر این بخش از کار را انجام می دهد. در ابتدا المان های دوربین ها را مشخص و در نهایت آنها را نمایش می دهد.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Place two cameras in the world
t1org_w = [-3; -6; 5]; % Origin of camera 1 in world
uz  =  -t1org_w  /  norm(t1org_w);  %  Camera  z  points  toward
origin
ux = cross(uz, [0;0;1]); % Camera x is in the world xy plane
ux = ux/norm(ux);
uy = cross(uz, ux); % Camera y is just z cross x
H_c1_w = [ ux uy uz t1org_w; 0 0 0 1];
t2org_w = [+3; -6; 5]; % Origin of camera 2 in world
uz  =  -t2org_w  /  norm(t2org_w);  %  Camera  z  points  toward
origin
ux = cross(uz, [0;0;1]); % Camera x is in the world xy plane
ux = ux/norm(ux);
uy = cross(uz, ux); % Camera y is just z cross x
H_c2_w = [ ux uy uz t2org_w; 0 0 0 1];
% Show cameras on the 3D plot
hold on
plot3(t1org_w(1), t1org_w(2), t1org_w(3), '*');
text(t1org_w(1)+0.1, t1org_w(2), t1org_w(3), '1');
plot3(t2org_w(1), t2org_w(2), t2org_w(3), '*');
text(t2org_w(1)+0.1, t2org_w(2), t2org_w(3), '2');
axis equal
axis vis3d
```

در صورت اجرای این دو بخش از برنامه، شکل زیر حاصل می شود که در آن هم نقاط شبیه سازی شده و هم موقعیت دوربین ها نشان داده شده اند.



شکل ۶-۶: موقعیت نقاط و دوربین های شبیه سازی شده در سیستم مختصات جهانی

- به دست آوردن ماتریس المان های داخلی دوربین: در مرحله سوم، با فرض f=400، Cx=Cy=200 ماتریس المان های داخلی دوربین یعنی K را به دست می آوریم. یعنی:

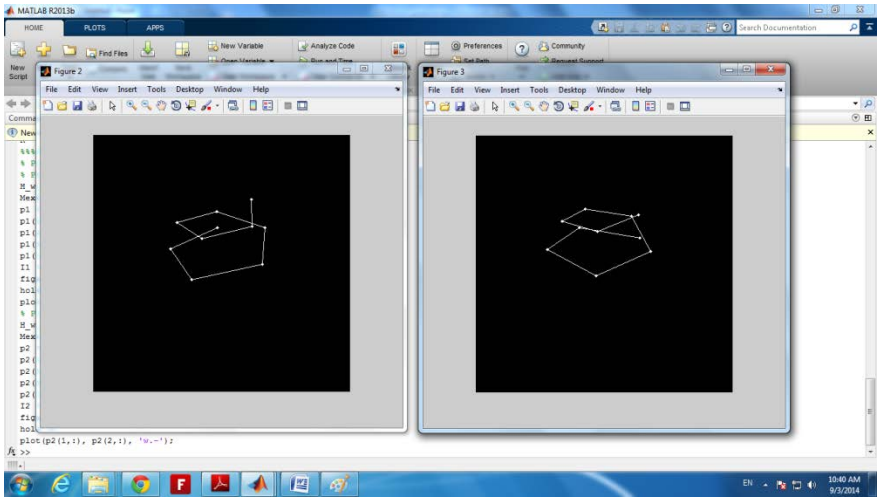
```
% Create intrinsic camera matrix
f = 400; % focal length in pixels
cx = 200;
cy = 200;
K = [ f 0 cx; 0 f cy; 0 0 1 ]; % intrinsic parameter matrix
```

در این بخش از کار، با استفاده از المان های دو دوربین، مختصات تصویری نقاط زمینی را حساب کرده و البته پس از اضافه کردن کمی نویز آنها را نمایش می دهیم.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Project the points onto the two cameras
% Project points onto image1
H_w_c1 = inv(H_c1_w);
Mext = H_w_c1(1:3,:);
p1 = K*Mext*P_w;
p1(1,:) = p1(1,:)./p1(3,:);
p1(2,:) = p1(2,:)./p1(3,:);
p1(3,:) = p1(3,:)./p1(3,:);
p1(1:2,:) = p1(1:2,:) + 2.0*randn(2,N); % Add a little noise
I1 = zeros(400,400);
figure, imshow(I1, []);
hold on
plot(p1(1,:), p1(2,:), 'w.-');
% Project points onto image2
H_w_c2 = inv(H_c2_w);
Mext = H_w_c2(1:3,:);
```

```
p2 = K*Mext*P_w;
p2(1,:) = p2(1,:)./p2(3,:);
p2(2,:) = p2(2,:)./p2(3,:);
p2(3,:) = p2(3,:)./p2(3,:);
p2(1:2,:) = p2(1:2,:) + 2.0*randn(2,N); % Add a little noise
I2 = zeros(400,400);
figure, imshow(I2, []);
hold on
plot(p2(1,:), p2(2,:), 'w.-');
```

شکل زیر تصاویر شبیه سازی شده را نشان می دهد.



شکل ۶-۷: تصاویر شبیه سازی شده از نقاط زمینی

- محاسبه مختصات نقاط زمینی: حال تمامی داده های شبیه سازی شده ما به دست آمده اند یعنی مختصات تصویری نقاط و المان های خارجی و داخلی هر دو دوربین. در این مرحله باید مختصات زمینی نقاط را به دست آوریم. برای این منظور در ابتدا مختصات تصویری را نرمال می کنیم. پس از آن ماتریس های A و b را تشکیل داده و سپس بوسیله کمترین مربعات، مختصات زمینی نقاط را به دست می آوریم. برنامه زیر این مراحل را انجام می دهد.

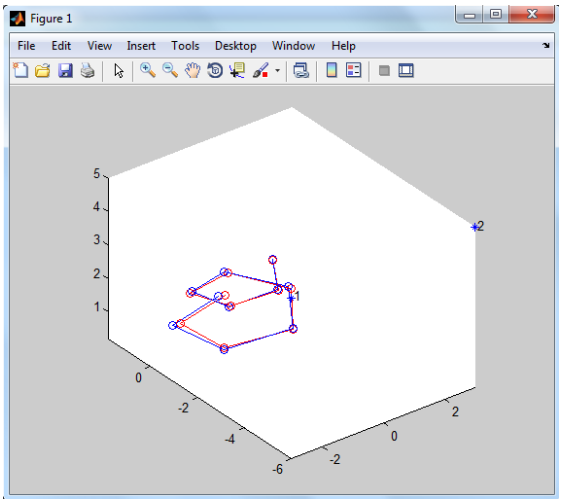
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Reconstruct points
% First normalize the image points
pn1 = inv(K)*p1;
pn2 = inv(K)*p2;
Pr_w = zeros(3,N); % This holds the reconstructed points
```

```
for i=1:N
% The equations for image point x,y are:
% x = (r11*X + r12*Y + r13*Z + tx)/(r31*X + r32*Y + r33*Z +
tz)
% y = (r21*X + r22*Y + r23*Z + ty)/(r31*X + r32*Y + r33*Z +
tz)
% or
% (r11-x*r31)*X + (r12-x*r32)*Y + (r13-x*r33)*Z = -tx + x*tz
% (r21-y*r31)*X + (r22-y*r32)*Y + (r23-y*r33)*Z = -ty + y*tz
% Here, (X;Y;Z) are the unknowns. Put into the form Ax=b.
A = zeros(4,3);
b = zeros(4,1);
% Camera 1
r11 = H_w_c1(1,1); r12 = H_w_c1(1,2); r13 = H_w_c1(1,3);
r21 = H_w_c1(2,1); r22 = H_w_c1(2,2); r23 = H_w_c1(2,3);
r31 = H_w_c1(3,1); r32 = H_w_c1(3,2); r33 = H_w_c1(3,3);
tx = H_w_c1(1,4); ty = H_w_c1(2,4); tz = H_w_c1(3,4);
x = pn1(1,i); y = pn1(2,i);
A(1,:) = [ r11-x*r31 r12-x*r32 r13-x*r33 ];
A(2,:) = [ r21-y*r31 r22-y*r32 r23-y*r33 ];
b(1) = -tx + x*tz;
b(2) = -ty + y*tz;
% Camera 2
r11 = H_w_c2(1,1); r12 = H_w_c2(1,2); r13 = H_w_c2(1,3);
r21 = H_w_c2(2,1); r22 = H_w_c2(2,2); r23 = H_w_c2(2,3);
r31 = H_w_c2(3,1); r32 = H_w_c2(3,2); r33 = H_w_c2(3,3);
tx = H_w_c2(1,4); ty = H_w_c2(2,4); tz = H_w_c2(3,4);
x = pn2(1,i); y = pn2(2,i);
A(3,:) = [ r11-x*r31 r12-x*r32 r13-x*r33 ];
A(4,:) = [ r21-y*r31 r22-y*r32 r23-y*r33 ];
b(3) = -tx + x*tz;
b(4) = -ty + y*tz;
x = A\b; % Solve for (X;Y;Z) point position
Pr_w(:,i) = x;
end
```

در نهایت می توانیم نقاط را نمایش دهیم یعنی:

```
% Display reconstructed 3D points
figure(1);
hold on
plot3(Pr_w(1,:), Pr_w(2,:), Pr_w(3,:), 'ro-');
```

در صورت اجرای برنامه فوق داریم:



شکل ۶-۸: نمایش نقاط شبیه سازی شده و بازسازی شده آنها توسط DLT

در شکل بالا، نقاط زمینی با رنگ آبی و نقاط باز سازی شده با رنگ قرمز نمایش داده شده اند.

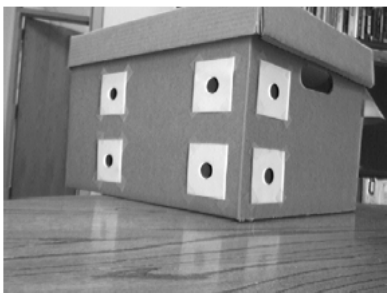
---

۶ Model Based Pose Estimation

تا کنون راجع به تبدیل بین تصاویر صحبت کرده ایم. همان طور که در بخش های بعدی خواهیم دید، در صورتی که بخواهیم مختصات نقاط را از تصویر استخراج کنیم نیازمند دانستن توجیه فضایی دوربین (یا camera pose) می باشیم. این توجیه با استفاده از یک ماتریس که دوران و موقعیت دوربین نسبت به یک سیستم مختصات مشخص (معمولا زمینی) را تعریف می کند مشخص می شود. این المان ها معمولا از طریق یک سری نقاط کنترل که مختصات زمینی و تصویری آنها مشخص است به دست می آیند. در این درس می‌خواهیم ببینیم که در حالت کلی توجیه یک دوربین با استفاده از نقاط کنترل چگونه انجام می شود. ورودی‌های ما در این بخش عبارتند از:

- تصویر عارضه
- هندسه عارضه (به طور مشخص موقعیت نقاطی از عارضه در سیستم مختصات زمینی)
- مختصات تصویری نقاط متناظر

در این صورت خروجی مورد انتظار ما عبارت است از موقعیت و دوران (pose) عارضه نسبت به دوربین (یا همان المان‌های توجیه خارجی دوربین) یا همان ${}^{C}_{M}\mathbf{H}$ (هوموگرافی تبدیل از مختصات مدل به دوربین). نمونه ای از مجموعه یک تصویر که در بر گیرنده عارضه های با مختصات مشخص بوده که مجموعا هندسه را تعریف می کنند در شکل زیر دیده می شود.



شکل ۴-۱: تصویر یک جعبه

در این جا دو فرض داریم. اول آنکه عارضه صلب (Rigid) است بنابراین فقط ۶ درجه آزادی داریم یعنی سه دوران و سه انتقال. فرض دوم آنست که المان‌های داخلی دوربین مشخص می‌باشند. در این صورت می خواهیم المان‌های خارجی دوربین را به گونه ای تعیین کنیم که مجموع مربعات خطای موقعیت نقاط در تصویر (یعنی اختلاف بین مختصات قرائت‌شده نقاط با مقدار محاسباتی آن‌ها / پیش‌بینی‌شده برای آن‌ها) حداقل گردد.

در این مساله، با توجه به اینکه هر نقطه منجر به ۲ معادله مستقل (یکی برای x و یکی برای y) می‌گردد ما حداقل به سه نقطه مدل (زمینی) نیاز داریم. مساله ای که در اینجا باید حل شود یک مساله بهینه سازی غیر خطی از طریق روش کمترین مربعات می‌باشد. در اینجا داریم y=f(x). در این رابطه x بردار مجهولات ما (المان‌های توجیه خارجی دوربین) می باشد. یعنی :

$$\mathbf{x} = \begin{pmatrix} \theta_x \\ \theta_y \\ \theta_z \\ t_x \\ t_y \\ t_z \end{pmatrix}$$

f تابعی است که با استفاده از المان‌های توجیه خارجی x موقعیت نقاط مدلی را در فضای تصویر پیش بینی (تقریب / تصویر) می‌کند. در حقیقت تابع f نقاط مدل را با استفاده از المان‌های x به موقعیت‌های تصویر آن‌ها یعنی بردار y تصویر می‌کند که در آن:

$$\mathbf{y} = \begin{pmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ \vdots \\ x_N \\ y_N \end{pmatrix}.$$

البته ما در اینجا با مقادیر اولیه y (y_0) محاسبات را شروع می‌کنیم که این مقادیر در حقیقت همان مختصات تصویری اندازه گیری شده نقاط می‌باشند. در حل مساله فوق x به گونه ای تعیین می‌شود که مقدار $E = |\mathbf{f(x)} - \mathbf{y0}|^2$ مینیمم شود.

پس روال کار به این شرح است:

۱. تعیین مقادیر اولیه (تقریبی) المان‌های مجهول x یعنی xo

۲. محاسبه y=f(x) و محاسبه باقیمانده‌ها یعنی $\mathbf{dy = y-y0}$

۳. محاسبه مشتق f نسبت به x یعنی $\mathbf{J = [\partial f/ \partial x]}$ و برآورد آن با استفاده از مقادیر فعلی x (در شروع همان xo)

٤. به دست آوردن مقادیر dx در رابطه dy=j dx از طریق $\mathbf{dx = (J^TJ)^{-1}J^T dy}$

۵. به دست آوردن x جدید از طریق **x <= x + dx**

۶. تکرار مراحل تا جایی که dx از یک مقدار threshold کمتر شود (یعنی دیگر تغییر نکند).

۶-۱ تعیین رابطه f(x)

حال ببینیم که تابع f(x) و مشتقات آن چگونه تشکیل می‌شوند. همانطور که به خاطر داریم برای اینکه بتوانیم مختصات یک نقطه در سیستم مختصات جهانی ${}^{W}\mathbf{P}$ را به نقطه ای از تصویر با مختصات (x_{im},y_{im}) تصویر کنیم از تبدیل پرسپکتیو استفاده می‌کنیم. یعنی: $\tilde{\mathbf{p}} = \mathbf{K}\mathbf{M}_{ext}\,{}^{W}\mathbf{P}$. در اینجا داریم:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \mathbf{K}\mathbf{M}_{ext}\,{}^{W}\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

با توجه به اینکه بردار x در فضای پروژکتیو است لذا باید برای بدست آوردن x_{im} و y_{im} مختصات x_1,x_2 را به x_3 تقسیم کنیم یعنی: $x_{im} = x_1 / x_3,\ y_{im} = x_2 / x_3$. علاوه براین دیدیم که ماتریس M_{ext} (ماتریس المان های خارجی) و K (ماتریس المان های داخلی) نیز به شکل زیر بودند:

$$\mathbf{M}_{ext} = \begin{pmatrix} {}^{C}_{W}\mathbf{R} & {}^{C}\mathbf{t}_{Worg} \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_X \\ r_{21} & r_{22} & r_{23} & t_Y \\ r_{31} & r_{32} & r_{33} & t_Z \end{pmatrix} \quad \mathbf{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

البته اگر در اینجا ما به جای سیستم مختصات جهانی از سیستم مختصات مدل استفاده کنیم خواهیم داشت:

$$\tilde{\mathbf{p}} = \mathbf{K}\mathbf{M}_{ext}\,{}^{M}\mathbf{P} = \mathbf{K}\begin{pmatrix} {}^{C}_{M}\mathbf{R} & {}^{C}t_{Morg} \end{pmatrix}{}^{M}\mathbf{P}$$

در نهایت همانطور که قبلاً اشاره شده بود ماتریس دوران ما به صورت R = Rz Ry Rx که در آن داریم:

$$R_Z = \begin{pmatrix} \cos\theta_Z & -\sin\theta_Z & 0 \\ \sin\theta_Z & \cos\theta_Z & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad R_Y = \begin{pmatrix} \cos\theta_Y & 0 & \sin\theta_Y \\ 0 & 1 & 0 \\ -\sin\theta_Y & 0 & \cos\theta_Y \end{pmatrix} \quad R_X = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_X & -\sin\theta_X \\ 0 & \sin\theta_X & \cos\theta_X \end{pmatrix}$$

البته در متلب برای به دست آوردن R ، ما فقط ماتریس‌های فوق را تشکیل داده و ضرب آن‌ها را به عهده متلب قرار می‌دهیم.

تمرین:برنامه متلبی بنویسید که یک نقطه از مدل را به نقطه متناظر آن بر روی تصویر تبدیل کند با فرض بر اینکه المان‌های دوربین معلوم باشند.

حل: این تابع را fproject می‌نامیم که ورودی‌های آن x (بردار المان‌های توجیه خارجی دوربین)، P_M (مختصات نقطه ای از مدل) و K (المان‌های داخلی دوربین) می‌باشند. خروجی فانکشن نیز مختصات تصویری نقطه می‌باشند. در زیر این برنامه نشان داده شده است.

```
function p = fProject(x, P_M, K)
% Project 3D point onto image
% Get pose params
ax = x(1); ay = x(2); az = x(3);
tx = x(4); ty = x(5); tz = x(6);
% Rotation matrix, model to camera
Rx = [ 1 0 0; 0 cos(ax) -sin(ax); 0 sin(ax) cos(ax)];
Ry = [ cos(ay) 0 sin(ay); 0 1 0; -sin(ay) 0 cos(ay)];
Rz = [ cos(az) -sin(az) 0; sin(az) cos(az) 0; 0 0 1];
R = Rz * Ry * Rx;
% Extrinsic camera matrix
Mext = [ R [tx;ty;tz] ];
% Project point
ph = K*Mext*P_M;
ph = ph/ph(3);
p = ph(1:2);
return
```

در این برنامه در ابتدا المان‌های توجیه را از ماتریس x استخراج می‌کنیم. این ماتریس در بر گیرنده مقادیر دوران حول محورهای x، y و z به رادیان و نیز شیفت های t_x، t_y و t_z می باشد. سپس با استفاده از آن‌ها ماتریس دوران را محاسبه می‌کنیم وماتریس المان های توجیه خارجی (M_{ext})را تشکیل می‌دهیم. پس از آن مختصات تصویری نقاط را به دست آورده و بر مختصه سوم تقسیم می‌کنیم. در نهایت، مختصات تصویری نقطه یعنی x_{im},y_{im} را به عنوان خروجی فانکشن قرار می‌دهیم.

تمرین: فانکشنی بنویسید که مختصات مجموعه ای از نقاط زمینی را با داشتن المان های توجیه خارجی و داخلی دوربین گرفته و مختصات تصویر آنها را حساب کند.

این برنامه در صورتی که بخواهیم فانکشن فوق مجموعه‌ای از نقاط را به مختصات تصویری آن‌ها تبدیل کند در اینجا داریم:

$$ {}^{M}\mathbf{P} = \begin{pmatrix} X_1 & X_2 & \cdots & X_N \\ Y_1 & Y_2 & \cdots & Y_N \\ Z_1 & Z_2 & \cdots & Z_N \\ 1 & 1 & \cdots & 1 \end{pmatrix} \qquad \mathbf{p} = \begin{pmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ \vdots \\ x_N \\ y_N \end{pmatrix} $$

همچنین داریم:

x=[ax; ay; az; tx; ty; tz] و نیز p = [x1;y1;x2;y2;...] فانکشنی که می‌نویسیم عین حالت قبلی است به جز قسمت آخر آن که مختصات تصویر را از مختصات پروژکتیو (از طریق تقسیم بر مختصه سوم) به دست می‌آوریم که به شکل زیر نوشته می‌شود:

```
ph(1,:) = ph(1,:)./ph(3,:);
ph(2,:) = ph(2,:)./ph(3,:);
```

در آخر برای حذف سطر سوم می نویسیم:

```
ph = ph(1:2,:);
```

نهایتا برای تشکیل ماتریس مختصات نقاط که ابعاد آن 2Nx1 است می نویسیم:

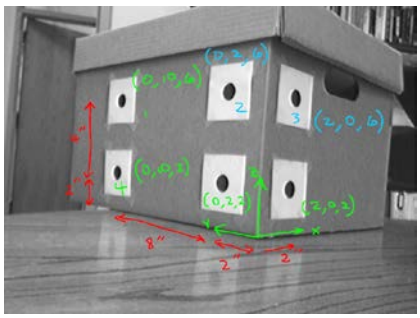
```
p = reshape(ph, [], 1);
```

در زیر برنامه کامل دیده می شود:

```
function p = fProject(x, P_M, K)
% Project 3D points onto image

% Get pose params
ax = x(1); ay = x(2); az = x(3);
tx = x(4); ty = x(5); tz = x(6);

% Rotation matrix, model to camera
Rx = [ 1 0 0; 0 cos(ax) -sin(ax); 0 sin(ax) cos(ax)];
Ry = [ cos(ay) 0 sin(ay); 0 1 0; -sin(ay) 0 cos(ay)];
Rz = [ cos(az) -sin(az) 0; sin(az) cos(az) 0; 0 0 1];
R = Rz * Ry * Rx;

% Extrinsic camera matrix
Mext = [ R  [tx;ty;tz] ];
```

```
% Project points
ph = K*Mext*P_M;

% Divide through 3rd element of each column
ph(1,:) = ph(1,:)./ph(3,:);
ph(2,:) = ph(2,:)./ph(3,:);
ph = ph(1:2,:); % Get rid of 3rd row

p = reshape(ph, [], 1); % reshape into 2Nx1 vector
return
```

تمرین: تصویرزیر را در نظر بگیرید. در این تصویر موقعیت (مختصات) زمینی نقاط کنترل و المان های توجیه داخلی (f=715 pixel، Cx=354، Cy=245) و نیز مقادیر تقریبی المان های توجیه خارجی معلوم است. برنامه‌ای بنویسید که موقعیت نقاط را بر روی تصویر به دست آورده و آن‌ها را نمایش دهد.



شکل ۴-۲: تصویر یک جعبه به همراه موقعیت نقاط کنترل بر روی آن

حل: برنامه متلب زیر نقاط کنترل را از طریق معادلات پروژکتور بر روی تصویر انداخته و آن‌ها را نمایش می‌دهد.

```
clear all
close all

I = imread('img1_rect.tif');
imshow(I, [])

% These are the points in the model's coordinate system (inches)
P_M = [  0      0       2       0       0       2;
        10      2       0      10       2       0;
         6      6       6       2       2       2;
         1      1       1       1       1       1 ];
```

```
% Define camera parameters
f = 715;      % focal length in pixels
cx = 354;
cy = 245;

K = [ f 0 cx; 0 f cy; 0 0 1 ];   % intrinsic parameter matrix


% Make an initial guess of the pose [ax ay az tx ty tz]
%x = [1.5; -1.0; 0.0; 0; 0; 30];
%x = [1.5481; -0.8323; 0.0971; 0.9342; 2.9837; 8.3286];  % True
x = [0.5; 0; 0; 0; 0; 30];

% Get predicted image points by substituting in the current pose


y = fProject(x, P_M, K);

for i=1:2:length(y)
    rectangle('Position', [y(i)-8 y(i+1)-8 16 16], 'FaceColor', 'r');
end
```

بخش‌های این برنامه عبارتند از:

۱. خواندن و نمایش تصویر (دستورات imread و imshow)

۲. تشکیل ماتریس نقاط کنترل زمینی (البته در سیستم مختصات پروژکتیو یعنی $\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$)

۳. تشکیل ماتریس K از طریق مقادیر f ، Cx و Cy

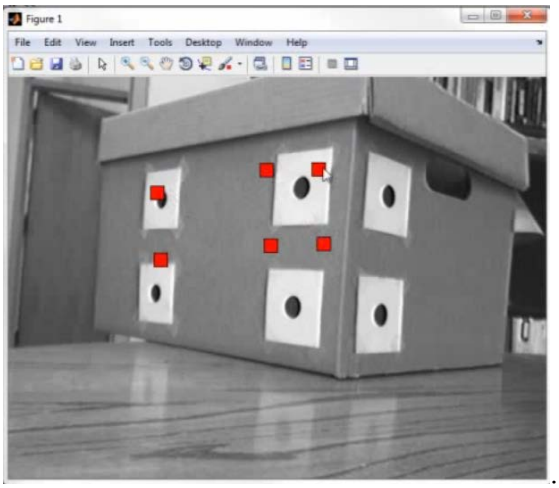
٤. تشکیل ماتریس مشاهده مختصات تصویری نقاط کنترل(y_0)

٥. تعریف بردار المان‌های تقریبی مجهولات x

٦. محاسبه مختصات تصویری نقاط با استفاده از تابع f project (ورودی‌ها عبارتند از: (x,P_M, K

۷. نمایش نقاط بر روی تصویر (از طریق فانکشن rectangle)

با اجرای برنامه فوق شکل زیر حاصل می‌شود که در آن نقاط قرمز، موقعیت‌های تصویری محاسبه شده توسط برنامه فوق نشان داده شده اند که با توجه به اینکه مقادیر x تقریبی بوده‌اند، از مکان‌های صحیح خود فاصله زیادی دارند



شکل ۴-۳: موقعیت تصویری نقاط کنترل با استفاده از المان های تقریبی توجیه خارجی

حال اگر بخواهیم موقعیت نقاط را بهبود بخشیم باید از روش کمترین مربعات استفاده کرده و المان های خارجی را با دقت تعیین نماییم. سپس با استفاده از آنها نقاط کنترل را بر روی تصویر ترانسفورم کنیم.

تمرین: برنامه‌ای بنویسید که در ادامه برنامه بالا المان‌های توجیه خارجی دوربین را با استفاده از نقاط کنترل حساب نموده و موقعیت تصویری نقاط کنترل را بهبود بخشد.

حل: همانطور که قبلاً اشاره شد برای به دست آوردن مقادیر المان‌های توجیه خارجی باید از رابطه f(x) مشتق بگیریم. همانطور که می‌دانیم مشتق (یا ژاکوبین Jacobean) تابع ما نسبت به مجهولات عبارت است از:

$$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_i} \approx \frac{\mathbf{f}(\mathbf{x}+\varepsilon\hat{\mathbf{u}}_i)-\mathbf{f}(\mathbf{x})}{\varepsilon}$$

در این رابطه $\varepsilon\hat{\mathbf{u}}_i$ برداریکه مربوط به المان x_i ام می‌باشد. ε نیز یک مقدار کوچکی است. در ریاضیات مشتق تابع حد رابطه بالا است وقتی ε به سمت صفر میل می‌کند. بنابراین اگر در یک برنامه برای مقدار ε عدد بسیار کوچکی در نظر بگیریم با استفاده از رابطه بالا با تقریب بسیار خوبی مشتق تابع محاسبه می‌شود. پس در صورتی که تابع ما y=fproject (x,P_M,K) باشد، با استفاده از برنامه زیر می‌توانیم ماتریس J را به دست آوریم.

```
y = fProject(x,P_M,K);
e = 0.000001;
J(:,1) = ( fProject(x+[e;0;0;0;0;0],P_M,K) - y )/e;
J(:,2) = ( fProject(x+[0;e;0;0;0;0],P_M,K) - y )/e;
J(:,3) = ( fProject(x+[0;0;e;0;0;0],P_M,K) - y )/e;
J(:,4) = ( fProject(x+[0;0;0;e;0;0],P_M,K) - y )/e;
J(:,5) = ( fProject(x+[0;0;0;0;e;0],P_M,K) - y )/e;
J(:,6) = ( fProject(x+[0;0;0;0;0;e],P_M,K) - y )/e;
```

بنابراین برنامه متلب زیر که یک Loop است به صورت مرحله‌ای نتایج به دست آمده برای x را (تکرار به تکرار) بهبود می‌بخشد. ورودی های این برنامه عبارتند از:

ورودی‌ها: y_o (مشاهدات تصویری نقاط کنترل)

x_o : مقادیر اولیه x

y=f(x) : تابع ما (همان fProject)

در این صورت مراحل اجرای برنامه به این شکل می باشند:

۱. x را مساوی x_o قرار بده

۲. y=f(x) و خطای باقیمانده $dy = y-y_o$ را حساب کن

۳. ماتریس J را با استفاده از مقادیر x (البته در لوپ اول $x = x_o$) برآورد کن.

a. توجه: تاکنون داریم: dy=J dx

٤. dx را از طریق $\mathbf{dx = (J^TJ)^{-1}J^T dy}$ محاسبه کن

۵. مقدار norm(dx)/norm(x) را حساب کن. اگر از یک، T=0.000001 کوچکتر نبود x = x+dx در غیر این صورت، برنامه را خاتمه بده.

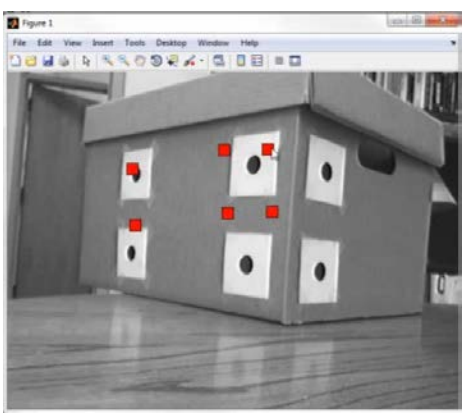
در زیر برنامه کامل که در ابتدا نقاط کنترل و نیز مقادیر المان های داخلی و خارجی اولیه را گرفته و با استفاده از آنها نقاط کنترل بر روی تصویر نشان می دهد و سپس مقادیر المان های خارجی نهایی را نیز تعیین می کند آمده است.

```
clear all
close all

I = imread('img1_rect.tif');
imshow(I, [])

% These are the points in the model's coordinate system (inches)
P_M = [  0      0       2       0       0       2;
        10      2       0      10       2       0;
         6      6       6       2       2       2;
```

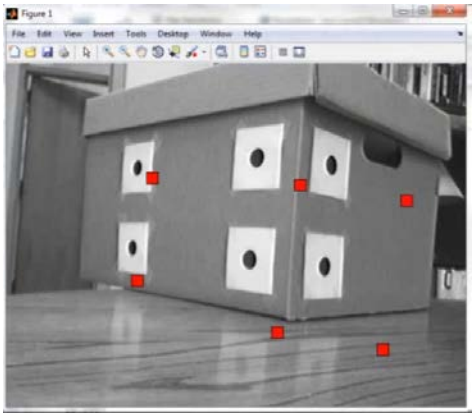
```
          1       1       1       1       1       1 ];

% Define camera parameters
f = 715;      % focal length in pixels
cx = 354;
cy = 245;

K = [ f 0 cx; 0 f cy; 0 0 1 ];   % intrinsic parameter matrix


% Make an initial guess of the pose [ax ay az tx ty tz]
%x = [1.5; -1.0; 0.0; 0; 0; 30];
%x = [1.5481; -0.8323; 0.0971; 0.9342; 2.9837; 8.3286];  % True
x = [1.5; -1; 0; 0; 0; 30];

% Get predicted image points by substituting in the current pose


y = fProject(x, P_M, K);

for i=1:2:length(y)
    rectangle('Position', [y(i)-8 y(i+1)-8 16 16], 'FaceColor',
'r');
end

pause

% Now improve the results;

% First measure the coordinates of control points.
y0 = [ 183; 147;    % 1
    350; 133;       % 2
    454; 144;       % 3
    176; 258;       % 4
    339; 275;       % 5
    444; 286 ];     % 6

for i=1:25
    fprintf('\nIteration %d\nCurrent pose:\n', i);
    disp(x);

    % Get predicted image points
    y = fProject(x, P_M, K);

    imshow(I, [])
    for i=1:2:length(y)
        rectangle('Position', [y(i)-8 y(i+1)-8 16 16], ...
            'FaceColor', 'r');
    end
    pause(0.2);

    % Estimate Jacobian
    e = 0.00001;    % a tiny number
    J(:,1) = ( fProject(x+[e;0;0;0;0;0],P_M,K) - y )/e;
```

```
    J(:,2) = ( fProject(x+[0;e;0;0;0;0],P_M,K) - y )/e;
    J(:,3) = ( fProject(x+[0;0;e;0;0;0],P_M,K) - y )/e;
    J(:,4) = ( fProject(x+[0;0;0;e;0;0],P_M,K) - y )/e;
    J(:,5) = ( fProject(x+[0;0;0;0;e;0],P_M,K) - y )/e;
    J(:,6) = ( fProject(x+[0;0;0;0;0;e],P_M,K) - y )/e;


    % Error is observed image points - predicted image points
    dy = y0 - y;
    fprintf('Residual error: %f\n', norm(dy));

    % Ok, now we have a system of linear equations   dy = J dx
    % Solve for dx using the pseudo inverse
    dx = pinv(J) * dy;

    % Stop if parameters are no longer changing
    if abs( norm(dx)/norm(x) ) < 1e-6
        break;
    end

    x = x + dx;   % Update pose estimate

    pause;
end
```

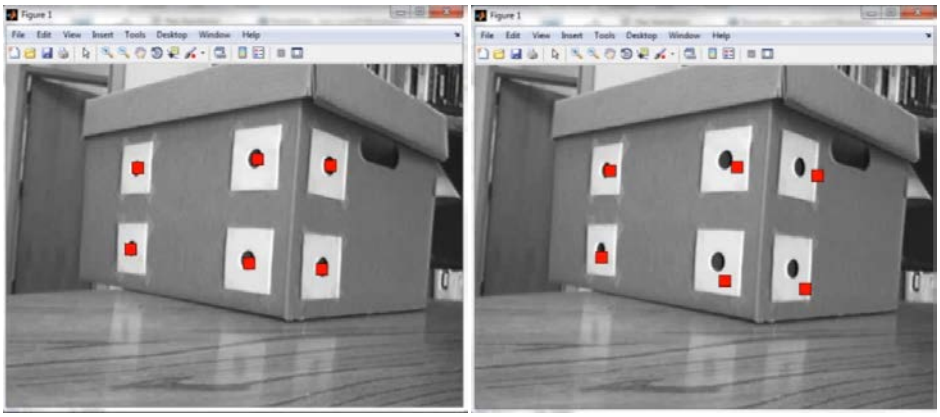
حال اگر برنامه فوق را اجرا کنیم. تصاویر زیر حاصل می‌شوند که همانطور که می‌بینیم در هر تکرار موقعیت نقاط تصویر شده به مکان های واقعی آنها نزدیک تر می شود یعنی در هر مرحله موقعیت تصویری نقاط کنترل بهبود یافته تا در نهایت بر روی موقعیت دقیق آن‌ها در تصویر قرار می گیرند.



a



b



d c

شکل ۴-۴: بهبود موقعیت تبدیل نقاط کنترل بر روی تصویر آنها

توجه: norm مجموع مربعات یک بردار را نشان می‌دهد یعنی $\|\mathbf{A}\|_F = \sum_{i,j} a_{i,j}^2$

حال با داشتن المان های دقیق توجیه خارجی، می توانیم با استفاده از تابع fProject هر نقطه زمینی را به صورت دقیق بر روی تصویر آن ترانسفورم کنیم.

تمرین: برنامه‌ای بنویسید که با توجه به مقادیر محاسبه شده، محورهای مختصات زمینی را روی تصویر نشان دهید.

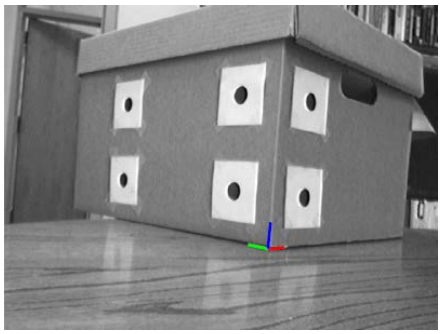
حل:

برای این برنامه کافی است که پس از محاسبه x ، مبدأ و سه برداری که محورهای سیستم مختصات زمینی را محاسبه و نشان دهیم.

کد زیر (که در باید انتهای برنامه قبلی اضافه شود) این کار را انجام می‌دهد.

```
u0 = fProject(x, [0;0;0;1], K); % origin
uX = fProject(x, [1;0;0;1], K); % unit X vector
uY = fProject(x, [0;1;0;1], K); % unit Y vector
uZ = fProject(x, [0;0;1;1], K); % unit Z vector
line([u0(1) uX(1)], [u0(2) uX(2)], 'Color', 'r', 'LineWidth', 3);
line([u0(1) uY(1)], [u0(2) uY(2)], 'Color', 'g', 'LineWidth', 3);
line([u0(1) uZ(1)], [u0(2) uZ(2)], 'Color', 'b', 'LineWidth', 3);
```

در کد فوق، u_o مبدأ مختصات و u_x ، u_y ، u_z بردارهای یکه محورهای x ، y ، z می‌باشند. با اجرای برنامه فوق خواهیم داشت:



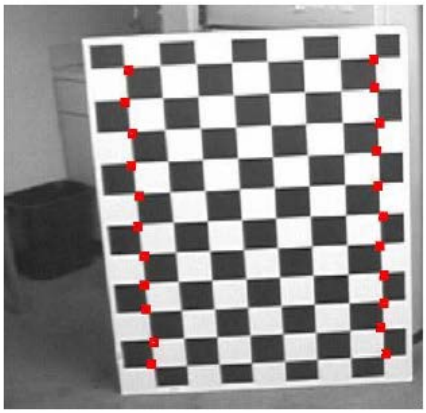
شکل ۴-۵: ترسیم سیستم مختصات یک عارضه

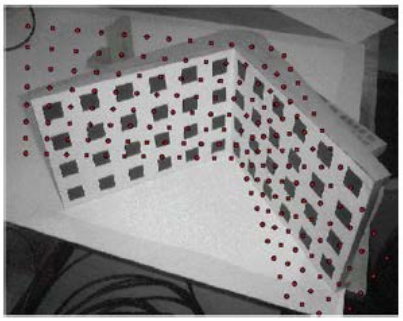
---

۷ Camera Calibration

کالیبراسیون دوربین برای بسیاری از فعالیت‌های کامپیوتر ویژن و فتوگرامتری مورد نیاز می‌باشد. هدف کالیبراسیون به دست آوردن المان‌های خارجی و داخلی دوربین می‌باشد. المان‌های خارجی در برگیرنده موقعیت و دوران (position & orientation) دوربین و المان‌های داخلی شامل فاصله کانونی، ابعاد پیکسل، ضرایب عدسی و مختصات مرکز تصویر می‌باشد. در تعیین المان‌های داخلی دوربین استفاده از یک الگوی کالیبراسیون یا مجموعه‌ای از نقاط[۵] یکی از مطمئن‌ترین روش‌های تعیین المان‌ها می‌باشد. برای این منظور معمولاً از یک مجموعه نقاط که بر روی یک صفحه قرار دارند از موقعیت‌های مختلف تصویربرداری می‌شود. در این پروسه سعی می‌شود که تصویرها بگونه ای گرفته شوند که حتی الامکان نقاط تارگت در تمامی یا حداکثر تصاویر ظاهر شوند. معمولاً سعی می‌شود که در مرحله اول المان های دوربین به غیر از ضرایب اعوجاج عدسی به دست آمده و سپس طی یک پروسه بهینه سازی غیر خطی (معمولاً با استفاده از روش کمترین مربعات) المان‌های عدسی تعیین شوند.

در زیر نمونه‌هایی از تست فیلدهای مورد استفاده در کامپیوتر ویژن دیده می‌شوند.





شکل ۷-۱: نمونه هایی از تست فیلدهای مورداستفاده برای کالیبراسیون دوربین

حال ببینیم این المان‌ها چگونه به دست می‌آیند. همانطور که قبلاً دیدیم، نقاط شیء سه بعدی با استفاده از یک ماتریس تبدیل پرسپکتیو (به ابعاد ۴×۳) برروی تصویر ، ترانسفورم می‌شوند. یعنی:

---

[۵] در فتوگرامتری به این مجموعه از نقاط تست فیلد (testfield) گفته می شود

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \mathbf{M} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

که البته (u,v,w) در سیستم هموژن می باشند. بنابراین برای به دست آورن مختصات تصویری باید درایه های اول و دوم را به درایه سوم تقسیم کنیم یعنی: $x = u/w, \quad y = v/w$. همانطور که نشان دادیم این ماتریس (یعنی $\mathbf{M} = \mathbf{K}\mathbf{M}_{ext}$) المان‌های زیر را مدل می‌کند:

- دوران و انتقال دوربین (t; R)
- فاصله کانونی (f)
- نسبت طول و عرض پیکسل (S_x,S_y)
- مختصات مرکز تصویر (C_x,C_y)

البته المان‌های لنز را مدل نمی‌کند. حال در صورتی که تست فیلد ما مسطح باشد، می‌توانیم برای همه نقاط Z=۰ را در نظر بگیریم. بنابراین ماتریس M و X ما خلاصه می شوند یعنی:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

تبدیل می شود به

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{14} \\ m_{21} & m_{22} & m_{24} \\ m_{31} & m_{32} & m_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

این ماتریس ۳×۳ یک ماتریس هرموگرافی است که مختصات (x,y,0) تست فیلد را به مختصات تصویری آن‌ها یعنی (x ، y) که در آن x=u/w و y=v/w می باشد تبدیل می‌کنند. مجهولات m_{11} تا m_{34} را با تصویربرداری از نقاط معلوم تست فیلد می‌توان به دست آورد. به این روش DLT گفته می‌شود. در این راستا می‌توانیم بنویسیم:

$$x = \frac{u}{w} = \frac{m_{11}X + m_{12}Y + m_{14}}{m_{31}X + m_{32}Y + m_{34}}, \quad y = \frac{v}{w} = \frac{m_{21}X + m_{22}Y + m_{14}}{m_{31}X + m_{32}Y + m_{34}}$$

در نتیجه داریم:

$$m_{11}X + m_{12}Y + m_{14} - x(m_{31}X + m_{32}Y + m_{34}) = 0$$
$$m_{21}X + m_{22}Y + m_{14} - y(m_{31}X + m_{32}Y + m_{34}) = 0$$

در صورتی که روابط بالا را در فرم ماتریس بنویسیم، اگر مجهولات (۹ المان ماتریس M) را m بنامیم داریم: $Am=0$ این رابطه یک رابطه هموژن است که با حل آن تمامی المان‌ها به جز مقیاس، تعیین می‌شوند. و همانطور که دیدیم جواب m چیزی نیست جز بردار ویژه A^TA وقتی که مقدار ویژه آن صفر باشد. و گفتیم که می‌توانیم m را با استفاده از SVD نیز بدست آوریم. به عبارت دیگر در صورتی که از A ، SVD بگیریم داریم $A = U D V^T$ که در این صورت، x یا همان m ، ستون آخر v (اولین ستون از سمت راست به چپ) می‌باشد. حال با داشتن M می‌توانیم از رابطه $\mathbf{M} = \mathbf{K}\,\mathbf{M}_{ext}$ ضرایب داخلی و خارجی دوربین را به دست آوریم. در این صورت المان‌های K (f_y ، f_x ، C_x ، C_y) و نیز M_{ext} (یعنی $r_{11} \rightarrow r_{33}$ و t_z ، t_y ، t_x) به دست می‌آیند.

چگونگی انجام این کار در مقاله (zhang,2000) آمده است:
Zhang, Z., 2000. A flexible new technique for camera calibration

۷-۱ تعیین المان‌های اعوجاج عدسی:

همانطور که می‌دانیم وقتی نور از عدسی عبور می‌کند شکست پیدا می‌کند. در نتیجه نقاط از مسیر فرضی که در دوربین pinhole در نظر گرفتیم عبور نمی‌کنند بلکه نقاط معمولاً به صورت شعاعی یا به سمت مرکز و یا به سمت خارج عدسی برروی تصویر، تبدیل می‌شوند. به حالت اول اعوجاج بالشی (Barrel) و به حالت دوم (pincushion) گفته می‌شود. البته علاوه بر اعوجاج شعاعی اعوجاج مماسی (Tangential) نیز ممکن است رخ دهد. اعوجاج بالشی در دوربین های با زاویه دید باز (wide anagle) و اعوجاج شعاعی در دوربین های با زاویه دید محدود (narrow angle) اتفاق می افتند. شکل زیر اثر اعوجاج‌های شعاعی را بر روی یک تصویر نشان می‌دهد. خطای اعوجاج مماسی معمولا در حالاتی اتفاق می افتد که عدسی دوربین کاملا موازی با صفحه تصویر برداری نباشد.

wideangle (barrel)

telephoto (pincushion)

شکل ۷-۲: اعوجاج شعاعی در دوربین

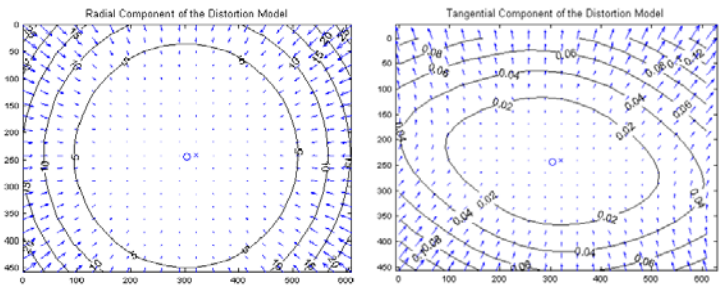
برای تعیین المان‌های اعوجاج از مدل‌های مختلفی بهره گرفته می‌شود. مدلی که ما در اینجا استفاده می‌کنیم مدل Heikkil & Silven است که در دانشگاه oulu فنلاند توسعه داده شده است. در متلب از مدل Stroble و همکاران وی استفاده شده است. در این رابطه‌ها معمولاً جهت x از چپ و به راست و y از بالا به پایین می‌باشد. نقطه چپ- بالا نیز مرکز سیستم مختصات است. اعوجاج شعاعی را می توان با رابطه زیر نشان داد:

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} (1 + k_1 r^2 + k_2 r^4 + k_5 r^6)$$

که در آن $r^2=x^2+y^2$ فاصله از مرکز را نشان می‌دهد. (شکل زیر را ببینید). با مشتق گیری از رابطه فوق، خطای مماسی به دست می آید که برابر است با:

$$\mathbf{dx} = \begin{bmatrix} 2k_3 xy + k_4\left(r^2 + 2x^2\right) \\ k_3\left(r^2 + 2y^2\right) + 2k_4 xy \end{bmatrix}$$

در شکل زیر اثر اعوجاج شعاعی و مماسی برای نقاط مختلف تصویر نشان داده شده اند:



شکل ۷-۳: اثر اعوجاج شعاعی و مماسی برای نقاط مختلف تصویر

بنابراین مدل نهایی عبارت است از:

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix}(1 + k_1 r^2 + k_2 r^4 + k_5 r^6) + \mathbf{dx}$$

در این صورت مختصات پیکسلی نقاط عبارتند از:

$$\begin{pmatrix} x_p \\ y_p \end{pmatrix} = \begin{pmatrix} f_1 x_d + cc_1 \\ f_2 y_d + cc_2 \end{pmatrix}$$

در رابطه بالا f_1 ، f_2 فاصله کانونی در راستای x ، y بود که البته با هم معمولاً برابراند. به نسبت f_2 / f_1 Aspect ratio گفته می‌شود که معمولاً برابر با ۱ است مگر آنکه CCD ها مربع نباشند.

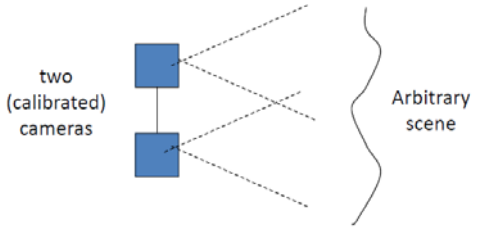
تعداد مجهولاتی که باید تعیین شوند ۹ است یعنی $f_1,f_2,cc_1,cc_2,K_1,K_2,\ldots,K_5$. برای حل المان‌ها معمولاً می‌توان مقادیر المان های توجیهات و فاصله کانونی که در مرحله اول (مرحله حل المان ها بدون لحاظ نمودن اعوجاجات عدسی) به دست آمده‌اند را به عنوان مقادیر اولیه در این مرحله مورد استفاده قرار داد. سپس با استفاده از روش کمترین مربعات به صورت غیر خطی (که مشابه آن قبلاً انجام شد) مجهولات را به دست آورد.

مجهولات در اینجا عبارتند از ۹ المان داخلی و ۶ المان خارجی برای هر دوربین. پس در صورتی که تعداد تصاویر K و تعداد نقاط M باشد، با توجه به اینکه هر نقطه ۲ معادله می‌دهد تعداد معادلات عبارتند از 2KM در حالی که تعداد مجهولات عبارتند از 9+6K. لذا باید 2KM>9+6K باشد تا مجهولات حل شوند. پس به طور مثال اگر تعداد تصاویر ۳ باشد، حداقل ۵ نقطه در هر تصویر نیاز است تا بتوان مجهولات حساب نمود.

کلیه عملیات فوق در Camera Calibration Toolbox متلب انجام می‌شوند.

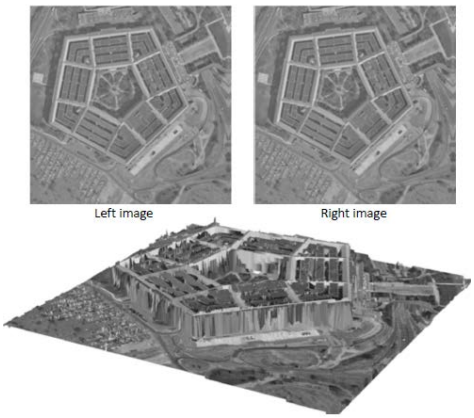
۸ Stereo Vision

استریو ویژن تکنیکی است برای به دست آوردن اطلاعات سه بعدی از ۲ (یا بیشتر) تصویر. همان طور که در شکل زیر دیده می شود می توان با استفاده از تصاویر پوشش دار مختصات نقاط عوارض را از طریق تقاطع به دست آورد.



اصل استریو در به دست آوردن مختصات سه بعدی نقاط

یک سایت اینترنتی که در آن برنامه‌ها و داده‌های زیادی در این زمینه قرار داده شده است به آدرس *vision.**middlebury**.edu/**stereo**/* می باشد. نمونه‌هایی از تصاویر و خروجی آن در شکل زیر دیده می‌شود



یک زوج تصویر هوایی و مدل سطح (DSM) مستخرج از آنها

همانطور که در شکل می‌بینیم بین نقاط مشابه در دو تصویر یک شیفت افقی وجود دارد که به آن disparity (در فتوگرامتری پارالاکس) گفته می‌شود.

تمرین: برنامه‌ای بنویسید که یک تصویر را گرفته و آن را شیفت داده و در هر مرحله اختلاف آن با تصویر اولیه را نشان دهد.

حل: برنامه زیر این کار را انجام می دهد.

```
Iright = im2double(imread('pentagonRight.png'));
Ileft = im2double(imread('pentagonLeft.png'));
% Disparity is d = xleft-xright
% So Ileft(x,y) = Iright(x+d,y)
for d=-20:20
```

```
d
Idiff = abs(Ileft(:, 21:end-20) - Iright(:, d+21:d+end-20));
imshow(Idiff, []);
pause
end
```

حال ببینیم مبنای تکنیک استریو چیست و چگونه می‌توانیم با استفاده از آن اطلاعات سه بعدی را استخراج کنیم. فرض کنیم دو دوربین داشته باشیم که المان‌های داخلی و توجیه نسبی آن‌ها نسبت به یکدیگر معلوم باشد. در این صورت اگر مختصات یک نقطه در تصویر اول و متناظر آن در تصویر دوم (معمولا گفته می شود تصاویر چپ و راست) را اندازه گیری کنیم با استفاده از تقاطع خط‌های حاصل از اتصال نقطه اندازه گیری شده به مرکز دوربین مختصات سه بعدی نقطه به دست می‌آید. برای فهم بهتر، شکل زیر را در نظر بگیرید.

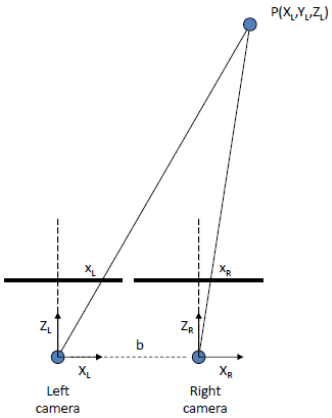


Figure ۱: محاسبه موقعیت سه بعدی یک نقطه به کمک دو تصویر پوشش دار

فرض کنیم که محور نوری دو دوربین کاملاً موازی با هم بوده و تنها تفاوت آنها فاصله آن‌ها (باز b) می‌باشد ضمناً صفحات دوربین‌ها هم صفحه (Co-planar) اند. در این صورت داریم:

$$x_L = f\frac{X_L}{Z_L}, x_R = f\frac{X_R}{Z_R}$$

حال با توجه به اینکه دو دوربین هم راستا اند پس $Z_L = Z_R = Z$. از طرفی، با فرض بر اینکه سیستم مختصات دوربین سمت چپ سیستم مختصات مرجع ما باشد: $X_L = X_R + b$. در نتیجه داریم:

$$x_L = f\frac{X_R + b}{Z}$$

با توجه به رابطه بالا می توان نوشت:

$$d = x_L - x_R = f\frac{(X_R + b) - X_R}{Z} = f\frac{b}{Z}$$

بنابراین: $Z = f\frac{b}{d}$. در این رابطه، d اختلاف مختصه x نقطه زمینی در تصاویر چپ و راست را نشان می دهد و به آن disparity یا پارالاکس می گویند. توجه شود که در رابطه فوق xL و xR باید هر دو مربوط به یک نقطه زمینی باشند و این فقط در صورتی است که شرایط بالا صادق باشند یعنی محور نوری دو دوربین کاملاً موازی با هم بوده و صفحات دوربین‌ها هم صفحه (Co-planar) باشند که اخذ تصاویر با این شرایط عملا غیر ممکن می باشد. لذا پس از اخذ تصاویر معمولا تصاویر تحت مرحله ای به نام ترمیم (rectification) قرار می گیرند تا اثر تغییر زاویه دید در آنها که ناشی از عدم توازی محورهای نوری آنها است بر طرف گردد.

به هر حال، در صورت مشخص بودن d در هر نقطه، می توان Z (فاصله نقطه زمینی تا دوربین) را با استفاده از d مربوطه حساب نمود. هر چه d بزرگ تر باشد Z کوچکتر و هر چه d کمتر باشد، Z بزرگتر می‌شود. اگر محاسبه Z برای همه نقاط در محدوده پوشش دو تصویر انجام و ترسیم شود، نقشه ای تولید می شود که به آن disparity map یا نقشه عمق گفته می شود. در شکل زیر دو تصویر و نقشه عمق آنها نشان داده شده است.

Figure ۲: تصاویر پوشش دار و نقشه عمق مربوطه

۸-۱ تعیین میزان خطای مدلسازی

همانطور که از رابطه تعیین عمق بر می‌آید خطای عمق(ΔZ) بستگی دارد به Z ، b ،f و نیز Δx_L و Δx_R. در این بخش می خواهیم میزان خطای مورد انتظار و نیز واریانس آن را پیدا کنیم. در ابتدا اجازه دهید خطای موجود در disparity (Δd) را پیدا کنیم. فرض کنیم که میزان خطای قرائت تشخیص نقاط متناظر در عکس را بدانیم و آن‌ها را Δx_L و Δx_R بنامیم. حال اگر از طرفین مشتق بگیریم داریم:

$$d(d) = d(x_L) - d(x_R)$$
$$\Delta d = \Delta x_L - \Delta x_R$$

اگر فرض کنیم Δx_L و Δx_R مستقل از هم باشند و متوسط آن‌ها صفر باشد، داریم:

$$\mu = E[\Delta d] = E[\Delta x_L] - E[\Delta x_R] = 0$$
$$Var[\Delta d] = E\left[(\Delta d - \mu)^2\right] = E\left[(\Delta d)^2\right]$$

از بسط رابطه واریانس داریم:

$$Var[\Delta d] = E\left[(\Delta x_L - \Delta x_R)^2\right] = E\left[\Delta x_L^{\ 2} - 2\Delta x_L \Delta x_R + \Delta x_R^{\ 2}\right]$$
$$= E\left[\Delta x_L^{\ 2}\right] - 2E\left[\Delta x_L \Delta x_R\right] + E\left[\Delta x_R^{\ 2}\right]$$

مقدار $\Delta x_L \Delta x_R$ برابر صفر است چون این دو مستقل از هم و متوسط آن‌ها صفر است پس وقتی یکی مثبت است دیگری منفی است لذا خطای مورد انتظار ضرب آن‌ها برابر صفر است. پس می توانیم بنویسیم: $Var[\Delta d] = E\left[\Delta x_L^{\ 2}\right] + E\left[\Delta x_R^{\ 2}\right]$ و یا $\sigma_d^2 = \sigma_L^2 + \sigma_R^2$.

حال اگر بخواهیم خطای Z را حساب کنیم باید از رابطه $Z = f\frac{b}{d}$ مشتق بگیریم پس داریم: $\Delta Z = f\frac{b}{d^2}(-\Delta d)$. در اینجا فرض کردیم فقط d دارای خطاست. خطای متوسط عبارت است از $\mu_Z = E[\Delta Z]$. بنابراین σ_z^2 عبارت است از $\sigma_Z^2 = E[(\Delta Z - \mu_Z)^2]$. اما با توجه به اینکه $\mu z = 0$ داریم: $E[(\Delta Z - \mu Z)^2] = E[\Delta Z^2] = \frac{fb}{d^2} = E[(-\Delta d)^2] = \left(f.\frac{b}{d^2}\right) * \sigma_z^2$ پس به طور خلاصه می توانیم بگوییم $\sigma_z^2 = (fb/d^2)\sigma_d^2$ و یا بخواهیم انحراف معیار را داشته باشیم داریم: $\sigma_z = (fb/d^2)\sigma_d$. در نهایت با توجه به اینکه داریم $Z = f\frac{b}{d}$ می‌توانیم بنویسیم $\sigma_z = (Z\sigma_d/d)$

تمرین: یک سیستم استریو داریم که میزان پارالاکس نقطه ای را 10 pixel برآورده کرده است. مطلوب است حساب کنید:

الف) عمق نقطه‌ای را با فرض بر آنکه f=500pixel و b=10cm

حل: z=fb/d=500*100/10=500cm

ب) خطای محاسبه z چقدر است اگر فرض کنیم خطای اندازه گیری یک نقطه در تصویر برابر با 1pixel باشد.

حل: $\sigma_Z = Z\sigma_d/d = Z\,(\sigma^2_{xr} + \sigma^2_{xl})^{1/2}/10 = z(1+1)^{1/2}/10 = 500 + 2^{1/2}/10 = 70cm$

سوالی که وجود دارد آنست که اگر هم قرائت پارالاکس و هم فاصله کانونی هر دو دارای خط باشند چه کار باید بکنیم. در اینجا داریم:$Z=fb/d$ در نتیجه: $\Delta Z=\Delta f.b/d+(fb/d^2)(-\Delta d)$. برای به دست آوردن واریانس کافی است از طرفین واریانس بگیریم. در این صورت شبیه به آنچه قبلاً گفتیم داریم:

$$\sigma^2 z=(f.b/d2).\sigma^2 d+(b/d).\sigma f$$

۸-۲ محاسبه مختصات در حالت کلی:

در حالت کلی یعنی وقتی که محور نوری دو دوربین ما هم صفحه نباشند، مطابق شکل زیر فرض کنیم که نقطه ما $P(X_L,Y_L,Z_L)$ باشد که تصویر آن بر تصاویر چپ و راست به ترتیب P_l و P_r باشد.

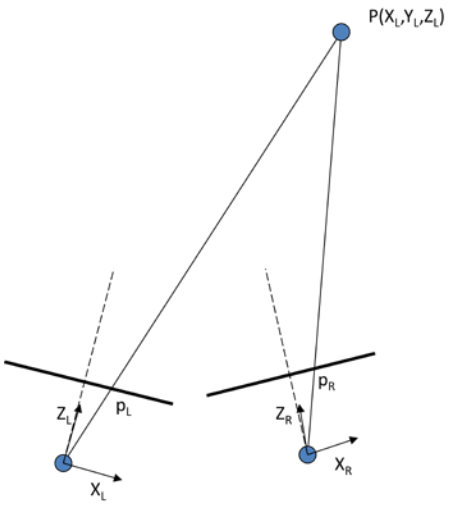


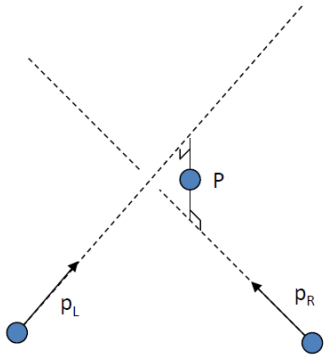
Figure ۳: هندسه تقاطع در استریو

در صورتی که سیستم مختصات نرمال شده را در نظر بگیریم (یعنی $f=1$) در این صورت داریم:

$$\mathbf{p}_L=\begin{pmatrix} x_L \\ y_L \\ 1 \end{pmatrix},\quad \mathbf{p}_R=\begin{pmatrix} x_R \\ y_R \\ 1 \end{pmatrix}$$

اگر توجیه نسبی دو دوربین معلوم باشد، ما می‌توانیم شعاع‌های O_Lp_L و O_Rp_R را تشکیل داده و با هم تقاطع دهیم که محل تقاطع طبیعتاً نقطه P خواهد بود. مشکلی که وجود دارد آنست که ممکن است این

دو خط با توجه به وجود خطا در اندازه‌گیری‌ها و محاسبات همدیگر را قطع نکنند. بنابراین ما می‌توانیم به جای پیدا کردن تقاطع در خط نقطه وسط قطعه‌ای که بر هر دو شعاع عمود است را به عنوان نقطه تقاطع در نظر بگیریم. این موضوع در شکل زیر نشان داده شده است.



۸-۲-۱ به دست آوردن مختصات سه بعدی یک نقطه در حالت کلی استریو

برای اینکه بتوانیم این نقطه را پیدا کنیم می دانیم تصویر نقطه P بر روی دو تصویر به این صورت است: $Z_L\,\mathbf{p}_L = \mathbf{M}_L\,\mathbf{P}$ و $Z_R\,\mathbf{p}_R = \mathbf{M}_R\,\mathbf{P}$ که در آنها داریم:

$$\mathbf{M}_L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\mathbf{M}_R = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{pmatrix} = \begin{pmatrix} {}^R_L\mathbf{R} & {}^R\mathbf{t}_{Lorg} \end{pmatrix}$$

در این روابط فرض آنست که مبدأ سیستم مختصات در دوربین سمت چپ است پس زوایای دوران صفر و مقادیر شیفت برابر با صفر می باشند. حال آنکه با توجه به اینکه وضعیت نسبی دوربین دوم نسبت به دوربین اول معلوم است، می توانیم ماتریس M_R را با استفاده از ماتریس دوران و بردار شیفت (نسبت به دوربین اول) تشکیل دهیم که در بالا آمد. حال با توجه به اینکه بردار سه بعدی p_L و تصویر آن یعنی M_LP موازی هم اند، ضرب خارجی آن‌ها باید صفر باشد همچنین است برای M_RP و P_R. طبیعتاً نقطه P باید هر دو شرط را تأمین کند پس داریم:

$$p_L \times \mathbf{M}_L\mathbf{P} = 0$$
$$p_R \times \mathbf{M}_R\mathbf{P} = 0$$

دستگاه بالا مجموعه‌ای از معادلات را نشان می‌دهد که در بر گیرنده ۴ معادله بوده و می توان با استفاده از آن مختصات P (سه مجهول X_L,Y_L,Z_L) را با استفاده از روش کمترین مربعات به دست آورد. حال اگر تعداد تصاویر افزایش یابد، طبیعتاً تعداد معادلات بیشتری داشته و قاعدتاً دقت تعیین P بهبود می‌یابد.
پس به طور کلی مراحل برآورد مختصات نقاط با استفاده روش استریو به این شرح اند:

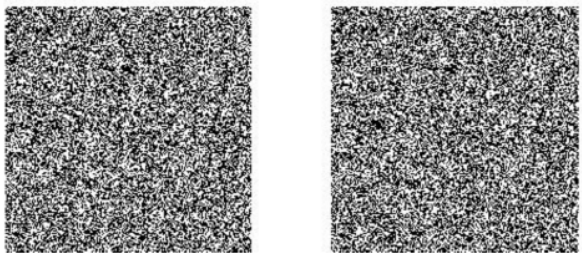
۱. استخراج نقاط (یا عوارض features) در تصاویر چپ و راست
۲. تعیین نقاط متناظر (برای محاسبه disparity)
۳. استفاده از disparity برای محاسبه عمق

از بین سه مرحله بالا، مرحله تناظر یابی یا Matching دشوارترین مرحله می‌باشد. برای اینکه ببینیم این کار را چگونه می توانیم انجام دهیم می توانیم ببینیم انسان چگونه آن را انجام می دهد. اولین نکته ای که وجود دارد آنست که برای تناظریابی اشیاء در بین عکس ها باید آنها به صورت شبیه به هم در تصاویر ظاهر شده باشند. بنابراین اگر از دو تصویر یکی نگاتیو و دیگری پوزتیو باشد (شکل زیر) علی رغم اینکه هر دو از یک منظر گرفته شده اند اما با توجه به عدم شباهت نوری آنها با هم چشم نمی تواند آنها را با هم ترکیب نماید. نمونه ای از چنین تصاویری در زیر دیده می شود.

Figure ۴ :یک تصویر و نگاتیو آن

نکته دیگری که وجود دارید آنست که در سیستم بینایی انسان فقط محدوده کوچکی از عمق یک عارضه به صورت همزمان می تواند برجسته دیده شود. به عبارت دیگر وقتی چشم در جایی فوکوس می شود فقط محدوده کوچکی قبل و بعد از آن را می تواند در همان زمان برجسته ببیند. در حقیقت چشم برای اینکه بتواند اعماق مختلف را درک کند، محور نوری چشم ها به هم متمایل یا از هم دور می شوند. در هر صورت نکته جالبی که وجود دارد آنست که چشم انسان برای تشخیص عمق به هیچ چیز جز اختلاف بین منظرها یا همان disparity احتیاج ندارد. Bela Julesz در سال 1971 آزمایشاتی با مجموعه ای از نقاط رندوم انجام داد که تنها اختلاف آنها شیفت در راستای x می باشد. به این مجموعه

از نقاط stereogram گفته می شود که چشم می تواند اختلاف عمق موجود بین آنها را به خوبی تشخیص دهد. این نشان می دهد که چشم انسان می تواند فقط با استفاده از اختلاف مناظر یا همان disparity عمق را تشخیص دهد. به عبارت دیگر، استریو خود به تنهایی و بدون نیاز به هیچ اطلاعات اضافی برای تشخیص عمق کافی است. شکل زیر یک زوج تصویر نقطه های رندوم را نشان می دهد که تنها اختلاف آنها یک disparity است. در صورت تنظیم چشم می توان عارضه سه بعدی بین نقاط را مشاهده نمود.



Bela Julesz, 1971

Figure ۵: استریو گرام Julesz

تمرین: برنامه ای بنویسید که یک Random Dot Stereogram بسازد.

حل: برنامه زیر این کار را انجام میدهد.

```
L = rand(400,400);
R = L;
% Shift center portion by 50 pixels
R(100:300, 150:350) = L(100:300, 100:300);
% Fill in part that moved
R(100:300, 100:149) = rand(201, 50);
```

در هر صورت برای محاسبه disparity باید بتوانیم نقاط متناظر را در تصاویر تشخیص بدهیم. مشکلی که وجود دارد آنست که برای هر نقطه در تصویر چپ تعداد زیادی نقطه می توان در تصویر دوم یافت که احتمالا متناظر نقطه مورد نظر باشند. البته ما میتوانیم با استفاده از اطلاعاتی که از هندسه تصویر برداری داریم محدوده جستجو در تصویر دوم را کاهش دهیم. در این میان مهمترین ویژگی که به ما در این زمینه کمک می کند شرط epipolar است. با استفاده از این شرط ما می توانیم جستجو برای پیدا کردن متناظر یک نقطه در تصویر دوم را محدود به یک خط بکنیم. این کار علاوه بر افزایش سرعت محاسبه، از تعدد کاندیداها و بالتبع از ابهام موجود می کاهد. این شرط در شکل زیر دیده می شود.

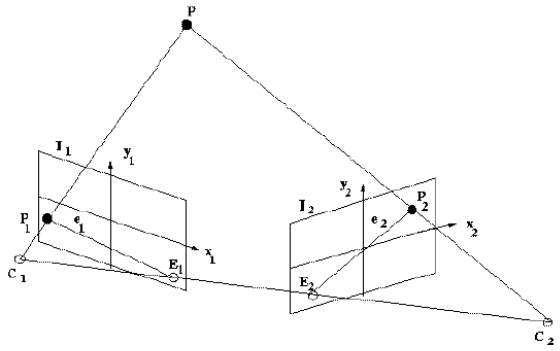


Figure ۶: شرط اپی پولار

همان ظور که می دانیم از مراکز دوربین ها و هر نقطه سه بعدی یک صفحه عبور می کند که تقاطع آن با هر کدام از تصویر های چپ و راست یک خط است. که به آنها خطوط اپی پولار گفته می شود. همانطور که در شکل نیز دیده می شود متناظر نقطه P_1 (تصویر P در عکس سمت چپ) در تصویر سمت راست، P_2 است که طبیعتاً در راستای خط P_2e می باشد. این موضوع چه دو تصویر هم راستا بوده و چه نباشند صادق است. بنابراین به جای جستجوی همه تصویر راست کافی است محدوده خط اپی پولار مربوط به نقطه P_1 در تصویر P_2 جستجو شود. پس کافی است برای هر نقطه، خط اپی پولار آن در تصویر دیگر محاسبه شده و برای پیدا کردن متناظر آن، جستجو در راستای این خط صورت پذیرد.

البته در عمل بهتر است اگر تصاویر متقارب باشند اول آنها را تغییر شکل بدهیم تا خطوط اپی پولار مستقیم و موازی شوند. به این کار ترمیم عکس با Rectification گفته می شود. شکل زیر نمونه ای از تصاویر و ترمیم شده آنها را نشان می دهد.

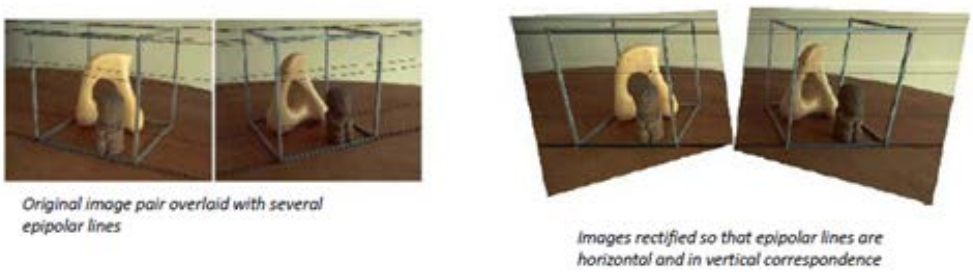


Figure ۷: دو تصویر و ترمیم شده آنها (توجه کنید که خطوط اپی پولار در سمت راست با هم موازی شده اند)

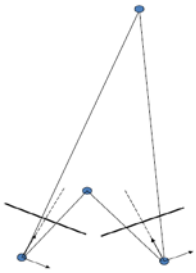
در هر صورت حتی اگر از شرط اپی پولار استفاده کنیم، باز حالاتی وجود دارند که در آنها یا برای یک نقطه هیچ نقطه متناظری پیدا نمی شود (مثل حالتی که دو تصویر از یک وایت برد گرفته ایم) و یا تعداد زیادی نقطه متناظر برای یک نقطه پیدا می شود (مثل تصاویر گرفته شده از یک صفحه شطرنجی). در

چنین حالاتی گفته می شود که مساله Under constrained است یعنی مساله یک جواب منحصر به فرد ندارد و بلکه برای آن جواب های متعددی وجود دارد. تنها راهی که برای خروج از این وضعیت وجود دارد آنست که از برخی اطلاعات جانبی که از واقعیت داریم استفاده نموده و مساله را حل کنیم. البته این فرضیات ممکن است که در همه جا صادق نباشند اما هر جا که باشند می توانند به حل مساله کمک و تناظر یابی را سهولت بخشند. حال مروری بر این شرایط بکنیم.

۸-۳ شرایط اجباری تناظریابی

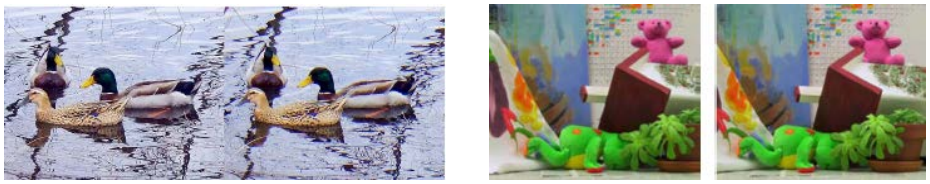
شرایط مختلفی برای تناظر یابی وجود دارد که اهم آنها در این بخش مرور می شود:

- محدوده disparity (Disparity Limits): بر اساس این فرض متناظر یک نقطه در تصویر دوم در یک محدوده کوچک از خط اپی پولار قرار دارد و لذا لازم نیست تمامی خط را جستجو کنیم. در زمانی که تصاویر ویدئویی گرفته می شود این شرط به خوبی می تواند به پروسه تناظریابی کمک کند چرا که اختلاف منظر تصاویر محدود بوده و با توجه به فاصله فریم ها از هم تقریبا شناخته شده میباشد.



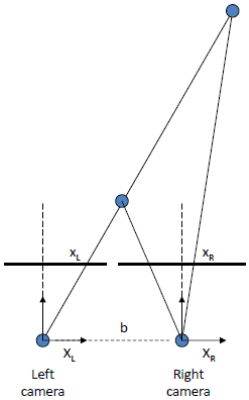
محدوده disparity

- Appearance (شباهت درجات خاکستری): در اینجا فرض می شود که عوارض متناظر از مقادیر رنگی یا الگوی رنگی مشابه تبعیت می کنند. اشیاء براق و متحرک مثل آب نمونه های خوبی از اجسامی اند که این ویژگی در مورد آنها صدق نمی کند. شکل زیر دو حالت را نشان می دهد که در یکی (سمت راست) شرط صدق و در دیگری (سمت چپ) صدق نمی کند.



شباهت بین درجات خاکستری عوارض مشابه

- یکه بودن نقطه متناظر (Uniqueness): در این جا فرض بر این است که هر نقطه در تصویر چپ فقط یک نقطه متناظر در تصویر سمت راست دارد. این شرط همیشه درست است مگر مواقعی که شیئ ما آئینه ای است.



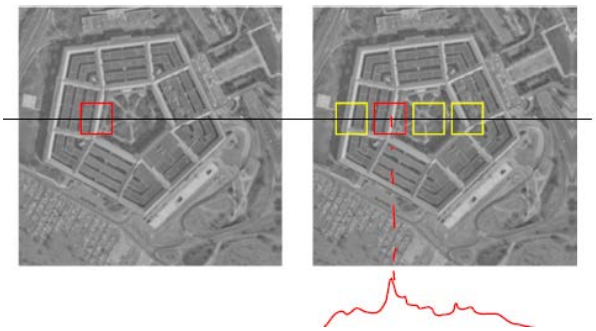
وجود دو نقطه متناظر برای یک نقطه

- ترتیب (Ordering): در این شرط فرض بر آنست که ترتیب نقاط در تصاویر چپ و راست مثل هم است. در صورتی که نقاط هم ارتفاع و یا نزدیک به هم باشند این شرط معمولا درست است اما وقتی اختلاف عمق نقاط زیاد شود ممکن است که این شرط دیگر صادق نباشد.
- نرمی (Smoothness): فرض می شود که سطح نرم است. در نتیجه تغیرات disparity از نقطه ای به نقطه دیگر شدید نیست بلکه به نرمی تغییر می کند. این فرض معمولا برای عوارض طبیعی درست است ولی برای عوارض ساخت دست بشر مثل ساختمان ها که تغییرات عمق در آنها بعضا شدید است معمولا صادق نیست.

۸-٤ روشهای تناظریابی

به طور کلی دو نوع روش تناظر یابی وجود دارد که عبارتند از Correlation-Based Matching و Feature-Based Matching. در ادامه به اختصار به این روشها اشاره می شود.

- Correlation-Based Matching: در این روش تصاویر به صورت قطعه (Patch) به قطعه مقایسه و نقاط متناظر تعیین می شوند. در این روشها فرض می شود که اختلاف بین دو قطعه در دو تصویر صرفا ناشی از شیفت بین تصاویر است یعنی دو قطعه نسبت به هم هیچ دوران و یا اختلاف در Appearance ناشی از اثرات پرسپکتیو ندارند. این روش معمولا برای موقعیت هایی که یک قطعه در سمت چپ و متناظر آن هر دو مربوط به یک سطح باشند و نیز همچنین موقعی که فاصله تا عارضه به نسبت فاصله بین دو دوربین (Base) زیاد باشد. علاوه بر این با توجه به اینکه مبنای روش میزان شباهت قطعه ها با توجه به ضریب وابستگی آنها است این روش در مناطق با texture (تعداد عوارض در تصویر) بالا خوب جواب می دهد. روند کلی این روش در شکل زیر نشان داده شده است.



روش Correlation-Based Matching

- روشهای Feature-Based: در این روش به جای مقایسه قطعه ها در تصاویر، عوارض مشخص همچون لبه ها، خطوط یا گوشه های استخراج شده در یک تصویر با معادل های آنها در تصویر/تصاویر دیگر مقایسه و عوارض مشابه تعیین می شوند. خروجی این روش پیوسته نیست که البته می توان از طریق درون یابی حد فاصل عوارض متناظر را نیز تناظریابی نمود. این روش شاید برای جاهایی که texture کمی دارند مناسبتر باشد.

تمرین: برنامه ای بنویسید که بر مبنای روش وابستگی تصاویر، نقاط متناظر را در دو تصویر پیدا کند.

حل: همان طور که در شکل زیر (اسلاید ۳۷) دیده می شود روش کار به این شکل است که ما در ابتدا یک پنجره را در تصویر سمت چپ مشخص می کنیم. سپس در راستای خط اپی پولار مربوط به آن در تصویر دیگر، پنجره به پنجره جستجو می کنیم. پنجره ای که دارای حداکثر شباهت (وابستگی) با پنجره مبدا باشد را به عنوان متناظر آن در نظر می گیریم. بنابراین تنها شرطی که در این تناظر یابی اعمال می شود شرط اپی پولار است. در زیر این برنامه آورده شده است.

```
% Simple stereo system using cross correlation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%FirstPart%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all
close all
% Constants
W=16; % size of cross-correlation template is (2W+1 x 2W+1)
DH = 50; % disparity horizontal search limit is -DH .. DH
DV = 8; % disparity vertical search limit is -DV .. +DV
Ileft = imread('left.png');
Iright = imread('right.png');
figure(1), imshow(Ileft, []), title('Left image');
figure(2), imshow(Iright, []), title('Right image');
pause;
% Calculate disparity at a set of discrete points
xborder = W+DH+1;
yborder = W+DV+1;
xTsize = W+DH; % horizontal template size is 2*xTsize+1
yTsize = W+DV; % vertical template size is 2*yTsize+1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%Second
Part%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


npts = 0; % number of found disparity points
for x=xborder:W:size(Ileft,2)-xborder
for y=yborder:W:size(Ileft,1)-yborder
% Extract a template from the left image centered at x,y
figure(1), hold on, plot(x, y, 'rd'), hold off;
T = imcrop(Ileft, [x-W y-W 2*W 2*W]);
%figure(3), imshow(T, []), title('Template');
% Search for match in the right image, in a region centered at
x,y
% and of dimensions DW wide by DH high.
IR = imcrop(Iright, [x-xTsize y-yTsize 2*xTsize 2*yTsize]);
%figure(4), imshow(IR, []), title('Search area');
% The correlation score image is the size of IR, expanded by W
in
```

```
% each direction.
ccscores = normxcorr2(T,IR);
%figure(5), imshow(ccscores, []), title('Correlation scores');
% Get the location of the peak in the correlation score image
[max_score, maxindex] = max(ccscores(:));
[ypeak, xpeak] = ind2sub(size(ccscores),maxindex);
hold on, plot(xpeak, ypeak, 'rd'), hold off;
% If score too low, ignore this point
if max_score < 0.85 continue;
 end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%Third
Part%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% These are the coordinates of the peak in the search image
ypeak = ypeak - W;
xpeak = xpeak - W;
%figure(4), hold on, plot(xpeak, ypeak, 'rd'), hold off;
% These are the coordinates in the full sized right image
xpeak = xpeak + (x-xTsize);
ypeak = ypeak + (y-yTsize);
figure(2), hold on, plot(xpeak, ypeak, 'rd'), hold off;
% Save the point in a list, along with its disparity
npts = npts+1;
xPt(npts) = x;
yPt(npts) = y;
dPt(npts) = xpeak-x; % disparity is xright-xleft
%pause
end
end
figure, plot3(xPt, yPt, dPt, 'd');
```

پارامترهای ورودی این برنامه عبارتند از: ابعاد پنجره مبدا (template patch size)، ابعاد پنجره جستجو افقی در تصویر دوم (horizontal disparity search window) و ابعاد پنجره جستجو عمودی در تصویر دوم (vertical disparity search window). در بخش اول برنامه پارامترها تعیین و ابعاد پنجره ها مشخص می شوند. در بخش دوم برنامه، یک پنجره در عکس سمت چپ در نظر گرفته می شود. با استفاده از Normalised Cross-Correlation متناظر پنجره در تصویر دیگر جستجو می شود. در صورتی که ارزش تناظر از یک حد آستانه بیشتر باشد، تناظر مورد قبول قرار می گیرد. این کار برای تمام تصویر تکرار می شود. در نهایت در بخش سوم، موقعیت های peak و disparity های مربوطه ذخیره و در انتها نمایش داده می شوند.

۸-۴-۱ نکات روش *Area-Based Matching*

در به کار گیری این روشها در نظر گرفتن نکات زیر مفید است.

در این روشها هر چه پنجره تناظر یابی بزرگتر باشد احتمال پیدا نمودن جواب های یکه بیشتر می شود حال آنکه در پنجره های کوچکتر احتمال وجود ناپیوستگی کمتر می شود. بنابراین سایز پنجره باید با توجه به این موارد تعیین شود.

روشهای مختلفی برای اندازه گیری میزان شباهت پنجره ها استفاده می شود که عبارت اند از:

Cross-Correlation (cc)
Sum of Squared Differences (SSD)
Sum of Absolute Differences (SAD)

یک سایت بسیار خوب که کد روشهای مختلف را به همراه برنامه مقایسه آنها در اختیار قرار می دهد همان سایتی است که درابتدای این درس معرفی شد یعنی: vision.middlefury.edu/stereo ضمن آنکه یک برنامه تجاری خوب در این زمینه برنامه PointGrey است که آدرس آن www.ptgrey.com می باشد.

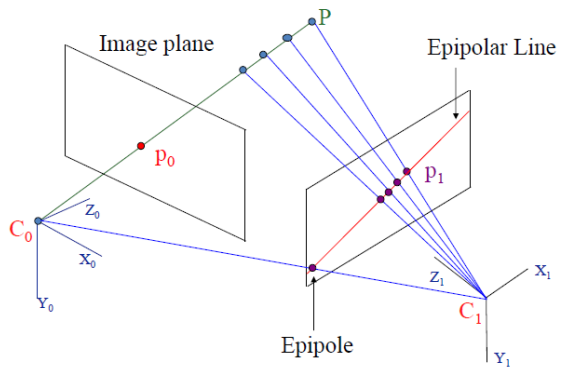
۹ Structure From Motion

تاکنون دو روش برای به دست آوردن اطلاعات سه بعدی از تصاویر دو بعدی را با هم بررسی کرده‌ایم. یکی از آن‌ها Model Based Pose Estimation بود که در آن با یک دوربین کالیبره شده از یک عارضه با هندسه مشخص تصویربرداری می شود. با استفاده از آن تصاویر گرفته شده می توان وضعیت و دوران عارضه نسبت به دوربین (یعنیpose) را تعیین کرد. راه دوم استفاده روش استریو بود که در آن از دو یا چند دوربین به صورت همزمان استفاده می شود که با توجه به معلوم بودن رابطه بین دوربین‌ها امکان به دست آوردن مختصات زمینی نقاط وجود دارد.

روشی که امروز می‌خواهیم راجع به آن صحبت کنیم روشStructure from motion است که شاید بهتر بود آن را Structure and motion from a moving camera می‌نامیدیم. در این روش ما فرض می‌کنیم که یک دوربین کالیبره شده داریم که از یک صحنه فرضی که مختصات نقاط آن را نداریم تصویربرداری می‌کند. در این صورت آنچه به دنبال آن هستیم عبارت اند از مختصات نقاط در صحنه مورد نظر و وضعیت نسبی دوربین‌ها نسبت به هم (یعنی Motion) بوسیله R و t یعنی دوران و انتقال. به عبارت بهتر، در کامپیوتر ویژن مختصات نقاط زمینی همان structure است و به دست آوردن المان های توجیه خارجی دوربین همان motion. البته همانطور که خواهیم دید، مقیاس واقعی در این روش بدون استفاده از شرایط / نقاط اضافی تعیین نخواهد شد. در ادامه این بخش، در ابتدا هندسه اپی پولار و ماتریس ضروری و نحوه به دست آوردن آن بحث شده و پس از آن نحوه به کارگیری این ماتریس برای تعیین دوران و انتقال بین دوربین‌ها و نیز مختصات نقاط صحنه بررسی می‌شود.

۹-۱ هندسه اپی پولار (Epipolar geometry)

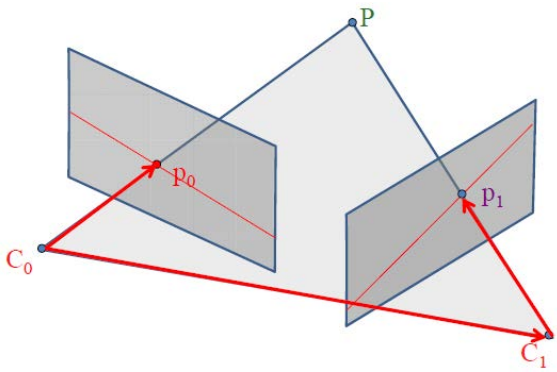
همانطور که در شکل زیر دیده می‌شود، اگر یک نقطه سه بعدی چون P داشته باشیم، این نقطه در دوربین اول (C_0) و دوم (C_1) در نقاط p_0 و p_1 تصویر می‌شود.



یک نقطه و متناظر آن در تصویر دوم

نکته‌ای که وجود دارد آنست که دیدن p_0 در تصویر اول به آن معنی است که این نقطه می‌توانسته هر کدام از نقاطی باشد که بر روی شعاع مربوط به آن (نقاطی که بین p_0 و P بر روی خط C_0P قرار دارند) قرار دارند. این نقاط دارای تصاویری بر روی تصویر C_1 هستند که هم خط بوده و بر روی خطی قرار دارند که به آن خط اپی پولار (epipolar Line) گفته می‌شود.

دیده می‌شود که از نقاط P، C_0، C_1 یک صفحه عبور می‌کند که نقاط p_0 و p_1 نیز بر روی آن قرار دارند. تقاطع این صفحه (سه بعدی) با تصویر (دو بعدی) دوم خطی است که در حقیقت خط اپی پولار مربوط به نقطه p_0 در تصویر اول می‌باشد. به همین ترتیب تقاطع این صفحه با تصویر اول، خط اپی پولار مربوط به نقطه p_1 می‌باشد. همان طور که در شکل زیر نیز دیده می شود، باید توجه کرد که در عمل ما موقعیت P در صفحه را نداریم آنچه داریم در حقیقت نقاط p_0، p_1، C_0، C_1 می‌باشند که با استفاده از این نقاط می‌توانیم بردارهای $C_0\ p_0$، $C_1\ p_1$ و نیز $C_0\ C_1$ را تشکیل دهیم که همگی در یک صفحه قرار دارند.



شرط هم صفحه ای

با توجه به اینکه این بردارها هم صفحه‌اند می‌توانیم بنویسیم:

$$\overrightarrow{\mathbf{C}_0\mathbf{p}_0}\cdot\left(\overrightarrow{\mathbf{C}_0\mathbf{C}_1}\times\overrightarrow{\mathbf{C}_1\mathbf{p}_1}\right)=0$$

به عبارت دیگر ضرب داخلی یکی از بردارها (مثلا C_0p_0) در ضرب خارجی دو بردار دیگر (مثلا C_0C_1 و C_1p_1) برابر با صفر می‌باشد. به این رابطه شرط هم صفحه‌ای گفته می‌شود. حال اگر بخواهیم رابطه‌ای برای هر نقطه از تصویر بنویسیم، با فرض بر اینکه این نقطه دارای مختصات نرمال شده است (یعنی f=1) می‌توانیم نقاط p_0 و p_1 را به این شکل نشان دهیم:

$$\mathbf{p}_0=\begin{pmatrix}x_0\\y_0\\1\end{pmatrix}^{C_0} \quad \text{و} \quad \mathbf{p}_1=\begin{pmatrix}x_1\\y_1\\1\end{pmatrix}^{C_1}$$

p_0 و p_1 در حقیقت بردارهای سه بعدی در دوربین های C_0 و C_1 می‌باشند که مختصات نقطه تصویری آن‌ها هر کدام در سیستم مختصات تصویر خود اندازه‌گیری شده است. بنابراین می‌توان تصویر بردار p_0 را از طریق رابطه $\mathbf{p}_0={}^{C_0}_{C_1}\mathbf{R}\,\mathbf{p}_1$ به دست آورد. توجه شود که در این رابطه همان طور که گفته شد p_0 یک بردار است نه یک نقطه. از طرفی می‌دانیم که بردار C_0C_1 نیز در حقیقت مشخص‌کننده انتقال بین دو دوربین یعنی t می‌باشد. بنابراین شرط هم صفحه‌ای را می‌توانیم به این شکل: $\mathbf{p}_0\cdot(\mathbf{t}\times\mathbf{R}\mathbf{p}_1)=0$ بنویسیم. که در این رابطه R، همان ${}^{C_0}_{C_1}\mathbf{R}$ و t نیز بردار انتقال C_1 نسبت به C_0 (یا همان ${}^{C_0}\mathbf{t}_{C_1org}$) می‌باشد.
همانطور که دیدیم توجیه نسبی دو دوربین را می‌توان با یک ماتریس هوموگرافی به صورت زیر نوشت:

$${}^{C_0}_{C_1}\mathbf{H}=\begin{pmatrix}{}^{C_0}_{C_1}\mathbf{R} & {}^{C_0}\mathbf{t}_{C_1org}\\ \mathbf{0} & 1\end{pmatrix}$$

حال ببینیم آیا می‌توان رابطه شرط هم صفحه‌ای را می‌توان به صورت ماتریسی نوشت؟می‌دانیم که حاصل ضرب خارجی دو بردار را می‌توان به صورت یک ماتریس و یک بردار نیز نوشت به عبارت اگر a و b دو بردار باشند حاصل ضرب خارجی آنها عبارت است از $[\mathbf{a}]_\times\mathbf{b}$ که در آن داریم:

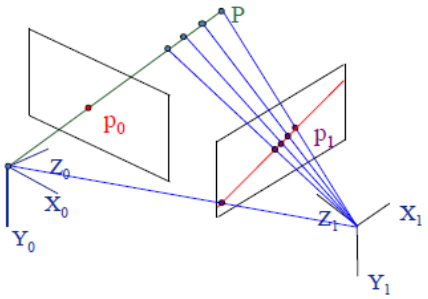
$$[\mathbf{a}]_\times=\begin{pmatrix}0 & -a_3 & a_2\\ a_3 & 0 & -a_1\\ -a_2 & a_1 & 0\end{pmatrix}$$

ضمن آنکه می‌دانیم حاصل ضرب داخلی دو بردار b و a نیز به این صورت است: $a.b = a^T.b$. بنابراین رابطه شرط هم صفحه‌ای (یعنی $\mathbf{p}_0 \cdot (\mathbf{t} \times \mathbf{R}\mathbf{p}_1) = 0$) را می‌توانیم به صورت ماتریسی بنویسیم یعنی $\mathbf{p}_0^T [\mathbf{t}]_\times \mathbf{R}\mathbf{p}_1 = 0$ که در آن $[\mathbf{t}]_\times$ یک ماتریس پادمتقارن (skew symmetric) با فرم نشان داده شده در بالا می‌باشد.

حال اگر $[\mathbf{t}]_\times \mathbf{R}$ را E بنامیم، داریم:

$$\begin{pmatrix} x_0 & y_0 & 1 \end{pmatrix} \begin{pmatrix} E_{11} & E_{12} & E_{13} \\ E_{21} & E_{22} & E_{23} \\ E_{31} & E_{32} & E_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = 0 \quad \text{یا} \quad \mathbf{p}_0^T \, \mathbf{E} \, \mathbf{p}_1 = 0$$

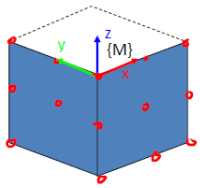
در رابطه بالا، به E ماتریس ضروری (Essential Matrix) گفته می‌شود که با استفاده از آن می‌توان یک نقطه در یک دوربین را به نقطه متناظر آن در دوربین دیگر ارتباط داد. برای این منظور کافی است ماتریس دوران R و بردار انتقال t را داشته باشیم که در این صورت $\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$. نحوه این ارتباط مجددا در شکل زیر آورده شده است.



برای اینکه نحوه ارتباط یک نقطه از یک تصویر با متناظر آن در تصویر دیگر از طریق ماتریس ضروری بهتر درک شود تمرین پایین را انجام دهید.

تمرین:مکعب زیر را در نظر بگیرید. با فرض بر اینکه المان‌های توجیه نسبی دو دوربین که از دو طرف مکعب تصویر گرفته‌اند را به شرح زیر داشته باشیم، مختصات نقاطی از مکعب را ترسیم و خطوط اپی پولار آن را نیز ترسیم کنید. در ضمن فرض کنید محورهای XYZ ثابت باشند.

- المان‌های توجیه نسبی مکعب نسبت به دوربین اول: ax=120°, ay=0°, az=60°, tx=3, ty=0, tz=0
- المان‌های توجیه نسبی دوربین دوم نسبت به دوربین اول: ax=0°, ay=-25°, az=0°, tx=3, ty=0, tz=1



حل: حل این مساله و انجام آن در چند مرحله است.

الف) ترسیم نقاط: برنامه زیر نقاط را ترسیم می کند به عبارت دیگر دو تصویر می سازد که تصویر نقاط مکعب بر روی آنها شبیه سازی شده است.

```
clear all
close all

%% Create a virtual image
L = 300; % size of virtual image in pixels
I = zeros(L,L);
%Define intrinsic camera matrix
% Define f, u0, v0
f = L;
u0 = L/2;
v0 = L/2;
% Create the matrix of intrinsic camera parameters
Mint = [ f 0 u0;
0 f v0;
0 0 1];
DEG_TO_RAD = pi/180;
% Create some points on the face of the cube
P_M = [
0 0 0 0 0 0 0 0 0 1 2 1 2 1 2;
2 1 0 2 1 0 2 1 0 0 0 0 0 0 0;
0 0 0 -1 -1 -1 -2 -2 -2 0 0 -1 -1 -2 -2;
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1;
];
NPTS = length(P_M);   %% Number of control points
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Define pose of model with respect to camera1
ax = 120 * DEG_TO_RAD;
ay = 0 * DEG_TO_RAD;
az = 60 * DEG_TO_RAD;
Rx = [ 1 0 0;
0 cos(ax) -sin(ax);
0 sin(ax) cos(ax) ];
Ry = [ cos(ay) 0 sin(ay);
0 1 0;
-sin(ay) 0 cos(ay) ];
Rz = [ cos(az) -sin(az) 0;
sin(az) cos(az) 0;
0 0 1 ];
R_m_c1 = Rx * Ry * Rz;
Pmorg_c1 = [0; 0; 5]; % translation of model wrt camera
M = [ R_m_c1 Pmorg_c1 ]; % Extrinsic camera parameter matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%Now create virtual Image1
% Render image 1
p1 = M * P_M;

% Divide on the third element to get xim and yim from x1,x2,x3
(as in homogeneous coordinates)
p1(1,:) = p1(1,:) ./ p1(3,:);
p1(2,:) = p1(2,:) ./ p1(3,:);
p1(3,:) = p1(3,:) ./ p1(3,:);
figure(1), imshow(I, []), title('View 1');

% Convert image points from normalized to unnormalized
u = Mint * p1;
for i=1:length(u)
rectangle('Position', [u(1,i)-2 u(2,i)-2 4 4], 'FaceColor',
'r');
end
pause

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up second view. For use the relative pose of C2 with
respect to C1 to get to the pose of the model with respect to
camera 2


% Define rotation of camera1 with respect to camera2
ax = 0 * DEG_TO_RAD;
ay = -25 * DEG_TO_RAD;
az = 0;
Rx = [ 1 0 0;
0 cos(ax) -sin(ax);
0 sin(ax) cos(ax) ];
Ry = [ cos(ay) 0 sin(ay);
0 1 0;
-sin(ay) 0 cos(ay) ];
Rz = [ cos(az) -sin(az) 0;
sin(az) cos(az) 0;
0 0 1 ];
R_c2_c1 = Rx * Ry * Rz;
% Define translation of camera2 with respect to camera1
Pc2org_c1 = [3; 0; 1];
% Figure out pose of model with respect to camera 2.
H_m_c1 = [ R_m_c1 Pmorg_c1 ; 0 0 0 1];
H_c2_c1 = [ R_c2_c1 Pc2org_c1 ; 0 0 0 1];
H_c1_c2 = inv(H_c2_c1);
H_m_c2 = H_c1_c2 * H_m_c1;
R_m_c2 = H_m_c2(1:3,1:3);
Pmorg_c2 = H_m_c2(1:3,4);
% Now time to form the camera2 Extrinsic parameter matrix
M = [ R_m_c2 Pmorg_c2 ];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Now we cam Render image 2 similar to what we did for Image 1
p2 = M * P_M;
p2(1,:) = p2(1,:) ./ p2(3,:);
p2(2,:) = p2(2,:) ./ p2(3,:);
p2(3,:) = p2(3,:) ./ p2(3,:);
figure(2), imshow(I, []), title('View 2');
```

```
% Convert image points from normalized to unnormalized
u = Mint * p2;
for i=1:length(u)
rectangle('Position', [u(1,i)-2 u(2,i)-2 4 4], 'FaceColor',
'r');
end
pause
```

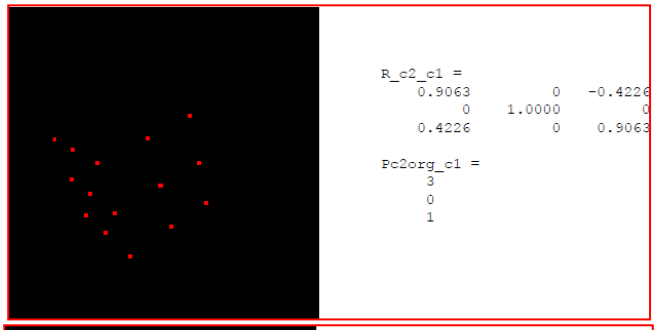
در این برنامه، در ابتدا یک تصویر ۳۰۰×۳۰۰ در نظر گرفته و فاصله کانونی را معادل ۳۰۰ پیکسل و مختصات مرکز تصویر را در وسط تصویر در نظر می‌گیریم یعنی ۱۵۰ و ۱۵۰. با استفاده از این مقادیر ماتریس المان‌های داخلی را تشکیل داده و مختصات نقاطی از مکعب را به صورت دستی وارد می‌کنیم. پس از آن ماتریس توجیه مکعب نسبت به دوربین اول را به کمک R و t (با توجه به مقادیر ذکر شده در بالا) تشکیل می‌دهیم. در این مرحله می‌توانیم تصویر اول را بسازیم (که به این کار rendering گفته می شود). برای این منظور مختصات تصویری نقاط را از ضرب مقادیر زمینی آن‌ها در ماتریس توجیه خارجی (M) به دست می‌آوریم. البته این مختصات به صورت نرمال شده می‌باشند که باید آن‌ها را به صورت نرمال نشده در آوریم که برای این منظور مختصات حساب شده را در معکوس ماتریس المان‌های داخلی ضرب کرده ایم. در اینجا می‌توانیم نقاط را ترسیم کنیم. این تصویر که در حقیقت شبیه سازی شده تصویری است که توسط دوربین اول گرفته می‌شود را View1 می‌نامیم.

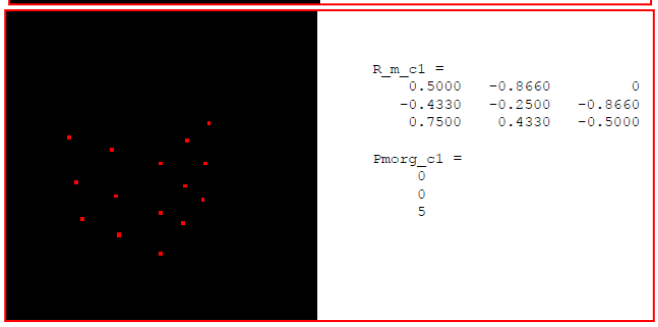
در بخش دوم برنامه، برای ترسیم View2 باید توجیه مکعب نسبت به دوربین شماره ۲ را محاسبه کنیم. برای این منظور در ابتدا ماتریس دوران و انتقال دوربین اول به دوم را شبیه حالت قبل حساب می‌کنیم سپس با استفاده از رابطه زیر، ماتریس مدل نسبت به دوربین دوم را نیز حساب می‌کنیم یعنی: ${}^{C2}_{M}\mathbf{H} = {}^{C2}_{C_1}\mathbf{H}\ {}^{C1}_{M}\mathbf{H}$. بنابراین تصویر جدید ترسیم شده در همان سیستم مختصات مدل می‌باشد.

در صورت اجرای این برنامه خواهیم داشت:

```
R_m_c1 =
    0.5000   -0.8660         0
   -0.4330   -0.2500   -0.8660
    0.7500    0.4330   -0.5000

Pmorg_c1 =
     0
     0
     5
```

همچنین نمایی از View1 (چپ)و View2 (راست) به این صورت خواهند بود:





ب) محاسبه ماتریس ضروری

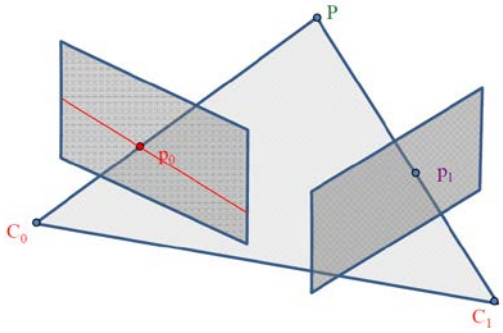
با توجه به اینکه $\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$ می‌توانیم E را حساب کنیم. برنامه زیر این کار را انجام می دهد.

```
% Calculate essential matrix from [t]x R_C2_C1
t = Pc2org_c1;
E = [ 0 -t(3) t(2); t(3) 0 -t(1); -t(2) t(1) 0] * R_c2_c1;
```

با اجرای برنامه خواهیم داشت:

```
E =
         0   -1.0000         0
   -0.3615         0   -3.1415
         0    3.0000         0
```

ج) ترسیم خطوط اپی پولار: در این بخش از برنامه می خواهیم خطوط اپی پولار را ترسیم کنیم. برای این منظور اجازه دهید ببینیم که اساساً یک خط چگونه ساخته و ترسیم می‌شود. همانطور که می‌دانیم معادله یک خط در صفحه به شکل ax+by+c=0 می‌باشد. حال اگر المان‌های خط را $\mathbf{l} = (a,b,c)^T$ بنامیم، می‌دانیم که هر نقطه P در صورتی برروی خط قرار دارد که $\mathbf{p}^T\mathbf{l} = 0$ باشد. از مقایسه این رابطه با رابطه $\mathbf{p}_0^T\,\mathbf{E}\,\mathbf{p}_1 = 0$ می‌توان دید که $E\mathbf{p}_1$ در حقیقت خط اپی پولار مربوط به نقطه p_1 در تصویر C_0 (شکل زیر) می‌باشد.



خط اپی پولار

بنابراین برای ترسیم این خطوط به این شرح عمل می‌کنیم

۱. یک نقطه در تصویر دوم در نظر می‌گیریم.

۲. خط اپی پولار مربوط به نقطه را در تصویر اول محاسبه می‌کنیم با استفاده از l=Ep1

- با توجه به اینکه l= a,b,c

- با توجه به اینکه معادله خط عبارت است از:

$$\begin{pmatrix} x_0 & y_0 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = ax_0 + by_0 + c = 0$$

۳. خط به دست آمده را بر روی تصویر اول ترسیم می‌کنیم. برای این منظور دو نقطه بر روی خط پیدا نموده و خط بین آن‌ها را ترسیم می‌کنیم برای این منظور یکباره x را برابر با -1 قرار داده و y را حساب می‌کنیم یکبار هم برابر با +1 قرار داده و y را حساب می‌کنیم.

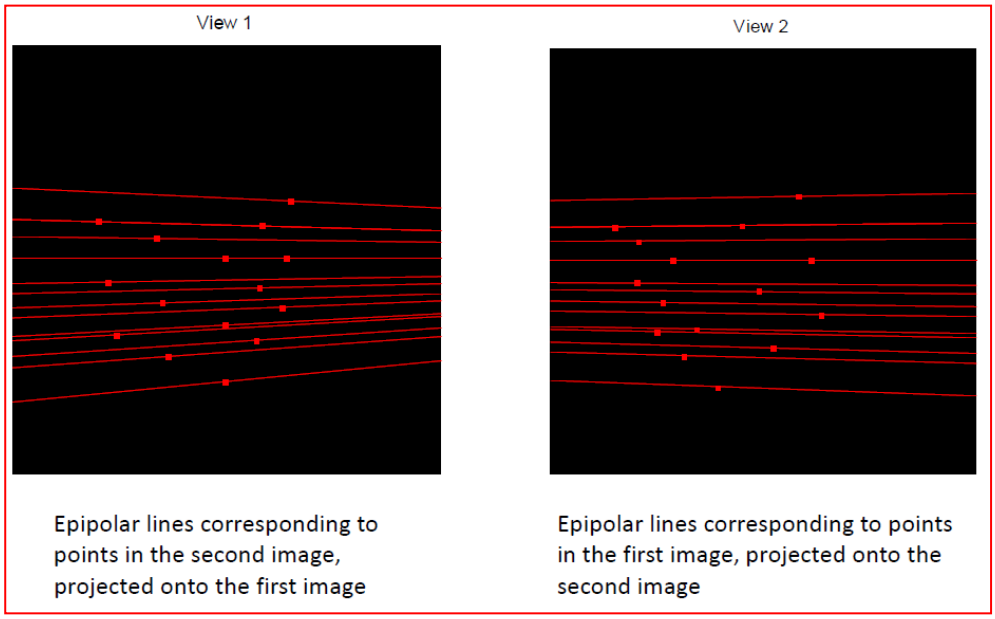
نکته: برای محاسبه y از رابطه y=(-c-ax)/b استفاده می‌کنیم.

برنامه زیر هر کدام از بخش‌های فوق را انجام می‌دهد.

```
% Draw epipolar lines
for i=1:length(p2)
figure(1);
% The product l=E*p2 is the equation of the epipolar line
corresponding
% to p2, in the first image. Here, l=(a,b,c), and the equation
of the
% line is ax + by + c = 0.
l = E * p2(:,i);
% Let's find two points on this line. First set x=-1 and solve
% for y, then set x=1 and solve for y.
pLine0 = [-1; (-l(3)-l(1)*(-1))/l(2); 1];
pLine1 = [1; (-l(3)-l(1))/l(2); 1];
```

```
% Convert from normalized to unnormalized coords
pLine0 = Mint * pLine0;
pLine1 = Mint * pLine1;
line([pLine0(1) pLine1(1)], [pLine0(2) pLine1(2)], 'Color',
'r');
pause
end
```

در صورت اجرای برنامه خواهیم داشت:



نمایی از خطوط اپی پولار ترسیم شده

۹-۲ تعیین E از طریق نقاط متناظر

در درس‌های قبلی دیدیم که $\mathbf{p}_0^T \mathbf{E}\, \mathbf{p}_1 = 0$ که در صورتی که R و $[\mathbf{t}]_\times$ معلوم باشند. با استفاده از $\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$ می‌توان E را حساب نمود. حال می خواهیم ببینیم که چگونه می توانیم با داشتن یک سری نقاط متناظر، E را به دست آوریم. در این درس برای تشریح کار، برنامه های متعددی را نوشته و به کار می گیریم. در ابتدا برنامه ای می نویسیم که تعدادی نقطه زمینی راتولید و دو تصویر گرفته شده از این نقاط را شبیه سازی می کند. پس از آن با استفاده مختصات تصویری تعدادی از نقاط متناظر در تصاویر شبیه سازی شده، E را به دست آورده و خطوط اپی پولار را نمایش می دهیم. در مرحله بعد، المان

های توجیه دوربین ها یا همان motion را با به دست می آوریم. در نهایت با استفاده از motion استخراج شده، مختصات زمینی بقیه نقاط را محاسبه می کنیم.

بنابراین می بینیم که روند کار به طور خلاصه به این شکل است که در ابتدا با استفاده از تعدادی از نقاط مشابه ماتریس E را حساب می کنیم[۶]. پس از آن با استفاده از ماتریس E و مختصات نقاط تصویری، مختصات زمینی یا مدلی نقاط را حساب می کنیم.[۷]

تمرین: برنامه ای بنویسید که مجموعه‌ای از نقاط یک مکعب را تولید، تصاویر view1 و view2 را شبیه سازی و نتیجه را در تصاویر l1.tif و l2.tif ذخیره کند.

حل: برنامه زیر این نقاط و تصاویر را تولید می کند.

```
% Create an image pair with known rotation and translation
between the
% views, and corresponding image points.

clear all
close all

L = 300;        % size of image in pixels
I1 = zeros(L,L);

% Define f, u0, v0
f = L;
u0 = L/2;
v0 = L/2;

%  Create the matrix of intrinsic camera parameters
K = [ f  0  u0;
      0  f  v0;
      0  0   1];

DEG_TO_RAD = pi/180;

% Create some points on the face of a cube
P_M = [
    0   0   0   0   0   0   0   0   0   1   2   1   2   1   2;
    2   1   0   2   1   0   2   1   0   0   0   0   0   0   0;
    0   0   0  -1  -1  -1  -2  -2  -2   0   0  -1  -1  -2  -2;
    1   1   1   1   1   1   1   1   1   1   1   1   1   1   1;
    ];
```

---

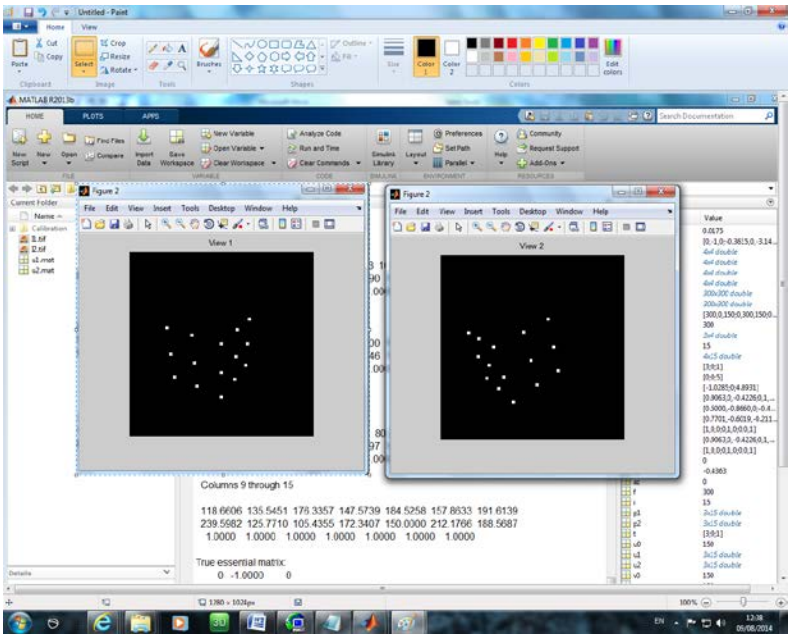
[۶] می دانیم به این کار فتوگرامتری توجیه یا ترفیع گفته می شود.

[۷] که در فتوگرامتری به این بخش از کار تقاطع یا ترسیم گفته می شود.

```
NPTS = length(P_M);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Define pose of model with respect to camera1
ax = 120 * DEG_TO_RAD;
ay = 0 * DEG_TO_RAD;
az = 60 * DEG_TO_RAD;

Rx = [ 1    0            0;
       0    cos(ax)     -sin(ax);
       0    sin(ax)      cos(ax) ];
Ry = [ cos(ay)    0     sin(ay);
       0          1     0;
      -sin(ay)    0     cos(ay) ];
Rz = [ cos(az)    -sin(az)   0;
       sin(az)     cos(az)   0;
       0           0         1 ];
R_m_c1 = Rx * Ry * Rz;
Pmorg_c1 = [0; 0; 5];   % translation of model wrt camera

M = [ R_m_c1 Pmorg_c1 ];    % Extrinsic camera parameter matrix

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Render image 1
p1 = M * P_M;

p1(1,:) = p1(1,:) ./ p1(3,:);
p1(2,:) = p1(2,:) ./ p1(3,:);
p1(3,:) = p1(3,:) ./ p1(3,:);

u1  =  K  *  p1;   %  Convert  image  points  from  normalized  to
unnormalized
for i=1:length(u1)
    x = round(u1(1,i));   y = round(u1(2,i));
    I1(y-2:y+2, x-2:x+2) = 255;
end
figure(1), imshow(I1, []), title('View 1');
pause


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up second view.
% Define rotation of camera1 with respect to camera2
ax = 0 * DEG_TO_RAD;
ay = -25 * DEG_TO_RAD;
az = 0;

Rx = [ 1    0            0;
       0    cos(ax)     -sin(ax);
       0    sin(ax)      cos(ax) ];
Ry = [ cos(ay)    0     sin(ay);
       0          1     0;
      -sin(ay)    0     cos(ay) ];
Rz = [ cos(az)    -sin(az)   0;
       sin(az)     cos(az)   0;
       0           0         1 ];
```

```
R_c2_c1 = Rx * Ry * Rz;

% Define translation of camera2 with respect to camera1
Pc2org_c1 = [3; 0; 1];

% Figure out pose of model wrt camera 2.
H_m_c1 = [ R_m_c1 Pmorg_c1 ;  0 0 0 1];
H_c2_c1 = [ R_c2_c1 Pc2org_c1 ;   0 0 0 1];
H_c1_c2 = inv(H_c2_c1);
H_m_c2 = H_c1_c2 * H_m_c1;

R_m_c2 = H_m_c2(1:3,1:3);
Pmorg_c2 = H_m_c2(1:3,4);

% Extrinsic camera parameter matrix
M = [ R_m_c2 Pmorg_c2 ];


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Render image 2
I2 = zeros(L,L);
p2 = M * P_M;

p2(1,:) = p2(1,:) ./ p2(3,:);
p2(2,:) = p2(2,:) ./ p2(3,:);
p2(3,:) = p2(3,:) ./ p2(3,:);

% Convert image points from normalized to unnormalized
u2 = K * p2;
for i=1:length(u2)
    x = round(u2(1,i));   y = round(u2(2,i));
    I2(y-2:y+2, x-2:x+2) = 255;
end
figure(2), imshow(I2, []), title('View 2');

disp('Points in image 1:');
disp(u1);
disp('Points in image 2:');
disp(u2);
imwrite(I1, 'I1.tif');
imwrite(I2, 'I2.tif');


% This is the "true" essental matrix between the views
t = Pc2org_c1;
Etrue = [ 0 -t(3) t(2); t(3) 0 -t(1); -t(2) t(1) 0] * R_c2_c1;
disp('True essential matrix:');
disp(Etrue);

save('u1.mat', 'u1');   % Save points to files
save('u2.mat', 'u2');
```

در زیر نتیجه اجرای این برنامه دیده می‌شود.



حال ببینیم که چگونه می‌توانیم با استفاده از این نقاط E را حساب کنیم. در این صورت می‌توانیم آن را با مقادیر شبیه‌سازی‌شده برنامه فوق مقایسه کنیم. فرض کنیم که تعدادی نقطه p_0 و متناظرهای آن‌ها یعنی p_1 را داشته باشیم می‌دانیم که برای این نقاط رابطه هم صفحه‌ای یعنی $\mathbf{p}_0^T\,\mathbf{E}\,\mathbf{p}_1 = 0$ صادق می‌باشد. به عبارت دیگر به ازای هر نقطه یک معادله می‌توانیم بنویسیم. با توجه به اینکه E یک ماتریس 3x3 است پس ۹ مجهول دارد. البته همانطور که می‌بینیم از آنجاییکه طرف دوم معادله صفر است لذا با هر ضریبی از آن نیز رابطه صادق است. پس در حقیقت ما ۸ مجهول داریم. در نتیجه با داشتن حداقل ۸ نقطه (یا بیشتر) نیز می‌توانیم ۸ معادله تشکیل و المان‌های E را محاسبه کنیم. به این روش الگوریتم 8 point گفته می شود.

۹-۲-۱ *محاسبه المان‌های ماتریس E از طریق الگوریتم ۸ نقطه*

همانطور که دیدیم $\mathbf{p}_0^T\,\mathbf{E}\,\mathbf{p}_1 = 0$ این رابطه را اگر به صورت ماتریسی بنویسیم داریم:

$$\begin{pmatrix} x_0 & y_0 & 1 \end{pmatrix} \begin{pmatrix} E_{11} & E_{12} & E_{13} \\ E_{21} & E_{22} & E_{23} \\ E_{31} & E_{32} & E_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = 0$$

همانند قبل در اینجا نیز فرض می‌کنیم که p_0 و p_1 نرمال شده‌اند یعنی f=1 و مرکز مختصات در وسط تصویر است. برای به دست آوردن المان‌های E باید آن را به صورت Ax=0 بنویسیم که در آن **x = (E11, E12, E13, ... , E33)** می‌باشد در این راستا از ضرب ماتریسی معادله بالا داریم:

$$E_{11}x_0x_1 + E_{12}x_0y_1 + E_{13}x_0 + E_{21}y_0x_1 + E_{22}y_0y_1 + E_{13}y_0 + E_{31}x_1 + E_{32}y_1 + E_{33} = 0$$

بدیهی است هر زوج نقاط متناظر، یک خط از ماتریس A را تشکیل می‌دهد. بنابراین با داشتن حداقل ۸ نقطه ماتریس A و بالتبع رابطه Ax=0 تشکیل می‌شود. همانطوری که قبلاً نیز دیدیم این یک مجموعه معادلات هموژن است که بدون در نظر گرفتن x=0 می‌توانیم یک جواب منحصر به فرد برای آن پیدا کنیم که بر اساس کمترین مربعات $\sum (\mathbf{p}_0^T \mathbf{E}\, \mathbf{p}_1)^2$ را مینیمم می‌کند. البته فراموش نکنیم که مقیاس در این مرحله تعیین نمی‌شود همان طوری که در فتوگرامتری نیز در مرحله توجیه نسبی مقیاس مدل به دست نمی آمد.

برای حل x همانند گذشته از طریق SVD(A) داریم: $\mathbf{A = U\,D\,V^T}$ که پاسخ x همان آخرین ستون ماتریس V (اولین ستون از سمت راست) می‌باشد.

تمرین: برنامه‌ای بنویسید که ماتریس E را با استفاده از نقاط مشترک متناظر حل کند.

حل: با توجه به مطالب گفته شده برنامه زیر E را به دست می‌دهد.

```
% Compute essential matrix E from point correspondences.
% We know that p1' E p2 = 0, where p1,p2 are the normalized
image coords.
% We write out the equations in the unknowns E(i,j)
% A x = 0

%% Asume some tie points
p1s=[61.4195  102.1798  150.0000   68.3768  106.2098  150.0000
74.3208  109.6134 150.0000  176.0870  196.1538  174.0000
192.8571  172.2222  190.0000;
  124.4290  136.1955  150.0000  167.2490  181.1490  197.2377
203.8325  219.1146 236.6025  127.4080  110.0296  170.7846
150.0000  207.7350  184.6410;
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
1.0000    1.0000 1.0000    1.0000    1.0000    1.0000    1.0000
1.0000    1.0000];

p2s=[ 45.5272   63.4568   86.9447   61.6620   80.2653  104.1468
75.7981   94.7507  118.6606  135.5451  176.3357  147.5739
184.5258  157.8633  191.6139;
  126.5989  136.7293  150.0000  165.9997  180.2731  198.5963
200.5196  217.7991  239.5982  125.7710  105.4355 172.3407
```

```
150.0000  212.1766  188.5687;
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
1.0000    1.0000    1.0000    1.0000    1.0000 1.0000    1.0000
1.0000    1.0000];
A = [p1s(1,:)'.*p2s(1,:)' p1s(1,:)'.*p2s(2,:)' p1s(1,:)' ...
p1s(2,:)'.*p2s(1,:)' p1s(2,:)'.*p2s(2,:)' p1s(2,:)' ...
p2s(1,:)' p2s(2,:)' ones(length(p1s),1)];
% The solution to Ax=0 is the singular vector of A corresponding
to the
% smallest singular value; that is, the last column of V in
A=UDV'
[U,D,V] = svd(A);
x = V(:,size(V,2)); % get last column of V
% Put unknowns into a 3x3 matrix. Transpose because Matlab's
"reshape"
% uses the order E11 E21 E31 E12 ...
Escale = reshape(x,3,3)';
```

۹-۲-۱-۱ ضعف حل معادلات از طریق الگوریتم ۸ نقطه

مشاهده شده است که به دست آوردن المان‌های E به شکلی که گفته شد روشی مستحکم نبوده و نتایج از ثبات مناسب برخوردار نمی باشند در نتیجه در برخی موارد وجود نویز در مشاهدات بعضاً منجر به جواب‌های نادرست می‌شود. برای برطرف نمودن این مساله از دو تکنیک preconditioning و postconditioning استفاده می‌شود.

- بهبود تکنیک ۸ نقطه از طریق preconditioning: در این تکنیک نقاط تصویری به‌گونه‌ای شیفت داده شده و مقیاس آن‌ها تغییر می‌کند که مبدأ سیستم مختصات آن‌ها به مرکز ثقل آن‌ها منتقل شده (they are centered at the origin) و نیز متوسط فاصله نقاط تا مبدأ مختصات $\sqrt{2}$ می‌باشد.
- بهبود تکنیک ۸ نقطه از طریق postcanditioning: همانگونه که می‌دانیم اعضای ماتریس E از هم مستقل نیستند. در حقیقت فقط ۵ پارامتر مستقل وجود دارد. بنابراین رنک E باید برابر با ۲ باشد.

در تکنیک postcanditioning، E به‌گونه‌ای تعیین می‌شود که رنک آن برابر با ۲ باشد.

تمرین: برنامه‌ای بنویسید که داده‌ها را precondition کند.

حل: برنامه زیر این کار را انجام می‌دهد. توجه شود که این برنامه در حقیقت قبل از برنامه بالا باید نوشته شود به عبارت دیگر باید نقاط اول precondition شوند و بعد ماتریس E با استفاده از آنها به دست آید. بنابراین اگر بخواهیم این برنامه و برنامه بالا را با هم استفاده کنیم باید در انتهای برنامه دستور زیر را اضافه کنیم:

```
 p1=p1s; p2=p2s;    %% To get point correspondence from previous
program.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Scale and translate image points so that the centroid of
% the points is at the origin, and the average distance of the
points to the
% origin is equal to sqrt(2).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
xn = p1(1:2,:); % xn is a 2xN matrix
N = size(xn,2);    %% Number of points
t = (1/N) * sum(xn,2); % this is the (x,y) centroid of the
points
xnc = xn - t*ones(1,N); % center the points; xnc is a 2xN matrix
dc = sqrt(sum(xnc.^2)); % dist of each new point to 0,0; dc is
1xN vector
davg = (1/N)*sum(dc); % average distance to the origin
s = sqrt(2)/davg; % the scale factor, so that avg dist is
sqrt(2)
T1 = [s*eye(2), -s*t ; 0 0 1];
p1s = T1 * p1;
xn = p2(1:2,:); % xn is a 2xN matrix
N = size(xn,2);
t = (1/N) * sum(xn,2); % this is the (x,y) centroid of the
points
xnc = xn - t*ones(1,N); % center the points; xnc is a 2xN matrix
dc = sqrt(sum(xnc.^2)); % dist of each new point to 0,0; dc is
1xN vector
davg = (1/N)*sum(dc); % average distance to the origin
s = sqrt(2)/davg; % the scale factor, so that avg dist is
sqrt(2)
T2 = [s*eye(2), -s*t ; 0 0 1];
p2s = T2 * p2;    % Finally, this preconditions the points
```

در این برنامه، در ابتدا مرکز ثقل نقاط در تصویر اول تعیین شده، سپس ضریب مقیاس محاسبه می‌شود. از ضرب نقاط در ماتریس مقیاس و انتقال، مختصات جدید نقاط به‌گونه‌ای

به دست می‌آید که مبدأ آن‌ها مرکز ثقل نقاط بوده و فاصله آن‌ها تا مبدأ جدید به نسبت (متوسط فواصل نقاط تا مبدا)/$\sqrt{2}$ کوچک شود. این عملیات برای نقاط تصویر دوم نیز انجام می‌شود. نتیجه این محاسبات در برنامه با p1s و p2s نشان داده شده است.

تمرین: برنامه‌ای بنویسید که نتایج را Postcendition کند.

حل: برای این منظور باید شرطی به معادلات اضافه کنیم که به واسطه آن E فقط دو مقدار ویژه غیر صفر که با هم برابراند داشته باشد. برای اعمال این شرط به این شکل عمل می‌کنیم.در ابتدا از E، SVD می‌گیریم. داریم: $\mathbf{E = U\,S\,V^T}$. حال شرط داشتن فقط دو مقدار ویژه غیر صفر را باید اعمال کنیم. برای این منظور S باید یک ماتریس قطری باشد که المان‌های قطر آن (۱.۱.۰) اند. پس برای اعمال این شرط ماتریس قطری با المان های فوق نوشته و جایگزین S می کنیم. یعنی: $\mathbf{E' = U\,diag(1,1,0)\,V^T}$. بنابراین در متلب برای اعمال این شرط می‌نویسیم:

```
[U,D,V] = svd(Escale);
Escale = U*diag([1 1 0])*V';
```

نکته مهمی که وجود دارد آنست که باید نقاط پس از حل معادلات به حالت اولیه بازگردانده شوند. یعنی پس از خاتمه محاسبات باید E نهایی به حالت اصلی خود بازگردد یعنی اثر تغییر مقیاس ناشی از Preconditioniy باید برطرف شود برای این منظور می‌نویسیم: `E = T1' * Escale * T2`. در این رابطه T1 و T2 ضرایبی اند که برای تغییر مقیاس نقاط در تصویر اول و دوم از آن‌ها استفاده شده است.

تمرین: برنامه‌ای بنویسید که با توجه به تمامی مطالب گفته شده تاکنون، ماتریس E را با استفاده از نقاط متناظر حل کند.

حل: برنامه زیر این کار را انجام می‌دهد.

```
% Calculate the essential matrix.
clear all
```

```
close all
K = [ 300 0 150; % intrinsic camera parameters
0 300 150;
0 0 1];
% These are the points in image 1
u1 = [
61.4195 102.1798 150.0000 68.3768 106.2098 150.0000 74.3208
109.6134 150.0000 176.0870 196.1538 174.0000 192.8571 172.2222
190.0000;
124.4290 136.1955 150.0000 167.2490 181.1490 197.2377 203.8325
219.1146 236.6025 127.4080 110.0296 170.7846 150.0000 207.7350
184.6410;
1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 ];
% These are the corresponding points in image 2
u2 = [
45.5272 63.4568 86.9447 61.6620 80.2653 104.1468 75.7981 94.7507
118.6606 135.5451 176.3357 147.5739 184.5258 157.8633 191.6139;
126.5989 136.7293 150.0000 165.9997 180.2731 198.5963 200.5196
217.7991 239.5982 125.7710 105.4355 172.3407 150.0000 212.1766
188.5687;
1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 ];
I1 = imread('I1.tif');
I2 = imread('I2.tif');


% Display points on the images for visualization
imshow(I1, []);
for i=1:length(u1)
x = round(u1(1,i)); y = round(u1(2,i));
rectangle('Position', [x-4 y-4 8 8], 'EdgeColor', 'r');
text(x+4, y+4, sprintf('%d', i), 'Color', 'r');
end
figure, imshow(I2, []);
for i=1:length(u2)
x = round(u2(1,i)); y = round(u2(2,i));
rectangle('Position', [x-4 y-4 8 8], 'EdgeColor', 'r');
text(x+4, y+4, sprintf('%d', i), 'Color', 'r');


% Get normalized image points
p1 = inv(K)*u1;
p2 = inv(K)*u2;


%%  PRECONDITIONING
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Scale and translate image points so that the centroid of
% the points is at the origin, and the average distance of the
points to the
% origin is equal to sqrt(2).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
xn = p1(1:2,:); % xn is a 2xN matrix
N = size(xn,2);
t = (1/N) * sum(xn,2); % this is the (x,y) centroid of the
```

```
points
xnc = xn - t*ones(1,N); % center the points; xnc is a 2xN matrix
dc = sqrt(sum(xnc.^2)); % dist of each new point to 0,0; dc is
1xN vector
davg = (1/N)*sum(dc); % average distance to the origin
s = sqrt(2)/davg; % the scale factor, so that avg dist is
sqrt(2)
T1 = [s*eye(2), -s*t ; 0 0 1];
p1s = T1 * p1;
xn = p2(1:2,:); % xn is a 2xN matrix
N = size(xn,2);
t = (1/N) * sum(xn,2); % this is the (x,y) centroid of the
points
xnc = xn - t*ones(1,N); % center the points; xnc is a 2xN matrix
dc = sqrt(sum(xnc.^2)); % dist of each new point to 0,0; dc is
1xN vector
davg = (1/N)*sum(dc); % average distance to the origin
s = sqrt(2)/davg; % the scale factor, so that avg dist is
sqrt(2)
T2 = [s*eye(2), -s*t ; 0 0 1];
p2s = T2 * p2;



% Compute essential matrix E from point correspondences.
% We know that p1s' E p2s = 0, where p1s,p2s are the scaled
image coords.
% We write out the equations in the unknowns E(i,j)
% A x = 0
A = [p1s(1,:)'.*p2s(1,:)' p1s(1,:)'.*p2s(2,:)' p1s(1,:)' ...
p1s(2,:)'.*p2s(1,:)' p1s(2,:)'.*p2s(2,:)' p1s(2,:)' ...
p2s(1,:)' p2s(2,:)' ones(length(p1s),1)];
% The solution to Ax=0 is the singular vector of A corresponding
to the
% smallest singular value; that is, the last column of V in
A=UDV'
[U,D,V] = svd(A);
x = V(:,size(V,2)); % get last column of V
% Put unknowns into a 3x3 matrix. Transpose because Matlab's
"reshape"
% uses the order E11 E21 E31 E12 ...
Escale = reshape(x,3,3)';


%%%%%%%   POSTCONDITIONING
% Force rank=2 and equal eigenvalues
[U,D,V] = svd(Escale);
Escale = U*diag([1 1 0])*V';

end


%% REMOVE THE EFFECT OF PRECONDITIONING
% Undo scaling
E = T1' * Escale * T2;
```

```
disp('Calculated essential matrix:');
disp(E);
save E
```

این برنامه از چند بخش تشکیل شده است:

- ورود نقاط در هر در تصویر
- نمایش نقاط
- نرمال کردن مختصات نقاط
- مقیاس کردن و شیفت دادن به نقاط
- محاسبه E
- اعمال شرط rank(E)=2
- بازگرداندن E به حالت قبل از مرحله ٤
- نمایش E نهایی

برنامه‌های essential.m و drawepipolar.m ماتریس E را حساب و خطوط اپی پولار را ترسیم می‌کنند. برنامه drawepipolar.m دو تصویر، مجموعه‌ای از نقاط متناظر و ماتریس E را گرفته و خطوط اپی پولار را ترسیم می‌کند که ساختار آن قبلا بحث شد.

۹-۳ محاسبه Motion: (Recoveering Motion Paramaters)

هدف این بخش آنست که بتوانیم با استفاده از E، المان‌های R و t یعنی توجیه نسبی دوربین دوم نسبت به اول را پیدا کنیم. همانطور که به خاطر دارید دیدیم که $\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$. نشان داده می شود که با استفاده از تکنیک SVD می‌توان دوران و انتقال را از R استخراج نمود. برای این منظور اگر از SVD،E بگیریم داریم: $\mathbf{E} = \mathbf{U}\,\mathbf{D}\,\mathbf{V}^T$ که در آن D همان ماتریس قطری (۱.۱.۰) diag ناشی از postconditioning می‌باشد. در اینجا نشان داده می‌شود که بردار t سومین ستون (آخرین) ماتریس U (یعنی u_3) یا معکوس آن ($-u_3$) می‌باشد. R نیز یا $\mathbf{U}\,\mathbf{W}\,\mathbf{V}^T$ یا $\mathbf{U}\,\mathbf{W}^T\,\mathbf{V}^T$ می‌باشد که در آن:

$$\mathbf{W} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

بنابراین همانگونه که دیده می‌شود ما ۴ حالت برای t و R داریم که سه حالت آن‌ها بی معنی اند یعنی اینکه با فرض درست بودن آن‌ها، عارضه (scene) باید پشت یک یا هر دو دوربین‌ها باشد تا مقابل آن‌ها. ما در حقیقت حالتی را می‌خواهیم که نقاط صحنه در روبروی دوربین‌ها باشند. بنابراین برای پیدا کردن ترکیب صحیح t و R باید مختصات نقطه یا نقاطی از صحنه را به دست آورده و ببینیم که آیا در مقابل هر دو دوربین هست یا نه؟ آن ترکیبی از t و R که منجر به این حالت می‌شود ترکیب صحیح ما می‌باشد لذا توانسته‌ایم که مقادیر صحیح t و R را از E استخراج کنیم.

البته فراموش نکنیم اگرچه المان‌های دوران صحیح می‌باشند، مقدار t فرضی بوده و مقیاس واقعی آن مشخص نیست بلکه در حقیقت ما فقط جهت آن را به دست می آوریم. به لحاظ مفهومی هم این موضوع صحیح است چرا که همانطور که دیدیم، E توجیه نسبی دوربین‌ها را از طریق شرط هم صفحه‌ای برقرار می کند و طبیعتا از حل آن فاصله دوربین‌ها (همان t) به دست نمی‌آید.

تمرین: برنامه‌ای بنویسید که با استفاده از E، ماتریس دوران و t را به دست آورید.

حل برنامه زیر این کار را انجام می‌دهد

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Extract motion parameters from essential matrix.
% We know that E = [tx] R, where
% [tx] = [ 0 -t3 t2; t3 0 -t1; -t2 t1 0]
%
% If we take SVD of E, we get E = U diag(1,1,0) V'
% t is the last column of U
[U,D,V] = svd(E);
W = [0 -1 0; 1 0 0; 0 0 1];


%% Now from the 4 possible combinations of R and t
Hresult_c2_c1(:,:,1) = [ U*W*V' U(:,3) ; 0 0 0 1];
Hresult_c2_c1(:,:,2) = [ U*W*V' -U(:,3) ; 0 0 0 1];
Hresult_c2_c1(:,:,3) = [ U*W'*V' U(:,3) ; 0 0 0 1];
Hresult_c2_c1(:,:,4) = [ U*W'*V' -U(:,3) ; 0 0 0 1];
% make sure each rotation component is a legal rotation matrix
for k=1:4
if det(Hresult_c2_c1(1:3,1:3,k)) < 0
Hresult_c2_c1(1:3,1:3,k) = -Hresult_c2_c1(1:3,1:3,k);
end
end
```

در ابتدا باید از E، SVD بگیریم. پس از ساختن W ماتریس پروژکشن $^{C1}_{C2}\mathbf{H}$ در چهار حالت زیر تعیین می‌کنیم:

$$t= u3 ; R=UWV^T$$
$$t= u3 ; R=UW^TV^T$$
$$t=- u3 ; R= UWV^T$$
$$t= -u3 ; R= UW^TV^T$$

البته باید در اینجا اطمینان حاصل کنیم که R بر اساس سیستم مختصات دست راست به دست آمده برای این منظور det(R) باید بزرگ‌تر از صفر باشد. لذا در صورتی ک det(R)<0 باشد آن را با R– جایگزین می‌کنیم. حال باید مختصات نقطه‌ای را با استفاده از هر کدام از حالات بالا به دست آورده و در حالتی که Z نقطه در هر دوربین بزرگ‌تر از صفر بود، R و t مربوطه، R و t مورد نظر و جواب نهایی ما می‌باشند. برای این منظور به این شکل عمل می‌کنیم.

- نقطه‌ای را در نظر می‌گیریم (برای این نقطه p1 و p2 یعنی مختصات تصویری آن معلوم می‌باشند)
- برای هر کدام از حالات ^1_2H این مراحل را طی می‌کنیم:
  - مختصات سه بعدی نقطه را در دوربین اول پیدا می‌کنیم (^1P)
  - مختصات سه بعدی نقطه را نسبت به دوربین ۲ نیز پیدا می‌کنیم یعنی: $^2P = {}^2_1H\ {}^1P$
- Z مربوط به ^1P و ^2P را چک می‌کنیم.
- حالتی که Z هر دو نقطه مثبت بود، R، t را نهایی می‌کنیم.

۹-٤ به دست آوردن مختصات زمینی نقاط (Structure)

نحوه به دست آوردن مختصات ^1P با استفاده از دو تصویر قبلاً در مبحث استریو نشان داده شد. دیدیم که برای به دست آوردن مختصات ^1P (مختصات سه بعدی یک نقطه که مختصات تصویری آن در هر دو دوربین مشخص است) را از روابط زیر تعیین می‌کنیم:

$$\mathbf{p}_1 \times \mathbf{M}_1\mathbf{P} = 0$$
$$\mathbf{p}_2 \times \mathbf{M}_2\mathbf{P} = 0$$

دستگاه فوق دارای ۴ معادله بوده و در فرم $AP=0$ نوشته می‌شود که برای حل آن اگر از A، SVD بگیریم، داریم $A = U D V^T$ که در حقیقت مجهولات ما که مختصات نقطه P در سیستم مختصات دوربین

اول (یعنی $\mathbf{P}^1$) می‌باشند، عبارتند از آخرین ستون V. البته این مقادیر باید نرمال شوند لذا المان‌های P را بر عضو چهارم تقسیم می‌کنیم.

برنامه زیر با داشتن مختصات تصویری، مختصات سه بعدی نقطه را به دست می آورد.

```
p1x = [ 0 -p1(3,1) p1(2,1); % skew symmetric matrix
p1(3,1) 0 -p1(1,1);
-p1(2,1) p1(1,1) 0 ];
p2x = [ 0 -p2(3,1) p2(2,1);
p2(3,1) 0 -p2(1,1);
-p2(2,1) p2(1,1) 0 ];
A = [ p1x * M1; p2x * M2 ];
% The solution to AP=0 is the singular vector of A corresponding
to the
% smallest singular value; that is, the last column of V in
A=UDV'
[U,D,V] = svd(A);
P = V(:,4); % get last column of V
P1est = P/P(4); % normalize
```

پس از آن باید R و t از ماتریس A استخراج شوند. در برنامه زیر چهار حالت بررسی شده و R و t به دست می آیند.

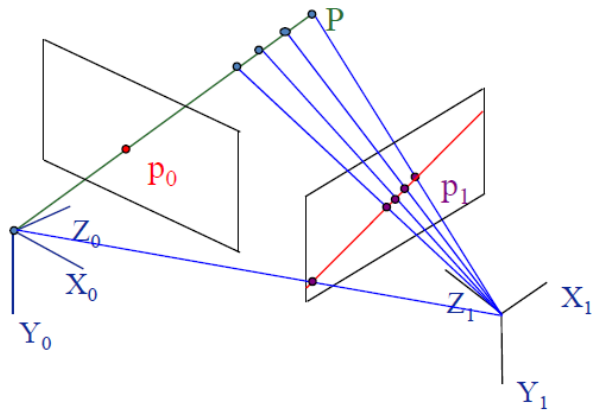
```
for i=1:4
Hresult_c1_c2 = inv(Hresult_c2_c1(:,:,i));
M2 = Hresult_c1_c2(1:3,1:4);
A = [ p1x * M1; p2x * M2 ];
% The solution to AP=0 is the singular vector of A corresponding
to the
% smallest singular value; that is, the last column of V in
A=UDV'
[U,D,V] = svd(A);
P = V(:,4); % get last column of V
P1est = P/P(4); % normalize
P2est = Hresult_c1_c2 * P1est;
if P1est(3) > 0 & P2est(3) > 0
Hest_c2_c1 = Hresult_c2_c1(:,:,i); % We've found a good solution
break; % break out of for loop; can stop searching
end
end
```

حال در صورتی که R و t معلوم باشند، می توانیم مختصات بقیه نقاط را به دست آوریم. برنامه زیر مراحل فوق را نشان می‌دهد. این برنامه twoview.m نام دارد. زمانی که المان‌های R و t به دست آمدند می‌توانیم مختصات بقیه نقاط را به دست آوریم. برنامه زیر این کار را انجام می‌دهد.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Hest_c1_c2 = inv(Hest_c2_c1);
M2est = Hest_c1_c2(1:3,:);
% Reconstruct point positions (these are good to the same scale
factor)
fprintf('Reconstructed points wrt camera1:\n');
for i=1:length(p1)
p1x = [ 0 -p1(3,i) p1(2,i);
p1(3,i) 0 -p1(1,i);
-p1(2,i) p1(1,i) 0 ];
p2x = [ 0 -p2(3,i) p2(2,i);
p2(3,i) 0 -p2(1,i);
-p2(2,i) p2(1,i) 0 ];
A = [ p1x * M1; p2x * M2est ];
[U,D,V] = svd(A);
P = V(:,4); % get last column of V
P1est(:,i) = P/P(4); % normalize
fprintf('%f %f %f\n', P1est(1,i), P1est(2,i),P1est(3,i));
end
% Show the reconstruction result in 3D
figure;
plot3(P1est(1,:),P1est(2,:),P1est(3,:),'d');
axis equal;
axis vis3d;
```

۱۰ Fundamental Matrix

دیدیم که با توجه به شکل زیر ماتریس E نقطه p_0 و متناظر آن در تصویر دوم p_1 را از طریق رابطه $\mathbf{p}_0^T \mathbf{E}\, \mathbf{p}_1 = 0$ با هم مرتبط می‌کنند که در آن p_0 و p_1 مختصات نرمال شده نقاط متناظر بوده و $\mathbf{E} = [\mathbf{t}]_x \mathbf{R}$



اما همانطور که قبلاً هم دیدیم برای اینکه بتوانیم با E کار کنیم مختصات باید نرمال شده باشند به عبارت دیگر ماتریس K یا همان ماتریس المان‌های توجیه داخلی دوربین‌ها باید معلوم باشند چرا که دیدیم $\mathbf{p} = \mathbf{K}^{-1}\mathbf{u}$ که در آن u مختصات نرمال نشده تصویری می‌باشند. بنابراین اگر K را نداشته باشیم، مجبوریم با نقاط نرمال نشده کار کنیم. حال ببینیم این کار را چگونه می‌توانیم انجام دهیم.

در رابطه $\mathbf{p}_0^T \mathbf{E}\, \mathbf{p}_1 = 0$ می‌دانیم که $\mathbf{p}_1 = \mathbf{K}^{-1}\mathbf{u}_1$ و نیز $\mathbf{p}_0^T = (\mathbf{K}^{-1}\mathbf{u}_0)^T = \mathbf{u}_0^T\mathbf{K}^{-T}$ بنابراین از قرار دادن این مقادیر در رابطه فوق داریم: $\mathbf{u}_0^T\mathbf{K}^{-T}\,\mathbf{E}\,\mathbf{K}^{-1}\mathbf{u}_1 = 0$ یا: $\mathbf{u}_0^T\,\mathbf{F}\,\mathbf{u}_1 = 0$ در این رابطه $\mathbf{F} = \mathbf{K}^{-T}\,\mathbf{E}\,\mathbf{K}^{-1}$ ماتریس Fundamental یا اساسی است که بر مبنای مختصات نرمال نشده یا همان مختصات پیکسلی تعیین می‌شود. البته باز با استفاده از آن می‌توان خطوط اپی پولار را تشکیل داد. بنابراین دیده می‌شود در صورتی که از مختصات پیکسلی نقاط یعنی همان مختصات قرائت شده تصویری استفاده کنیم ماتریسی که به دست می‌آید ماتریس F است که طبیعتاً با توجه به اینکه ما مختصات تصحیح نشده را استفاده کرده‌ایم، مدل سه‌بعدی حاصله دارای اعوجاج خواهد بود. اما اگر از مختصات تصحیح شده (یا همان نرمال شده) استفاده کنیم، ماتریس حاصل ماتریس E و شکل سه‌بعدی حاصله شکل تصحیح شده می‌باشد.

۱۰-۱ حل F:

برای به دست آوردن اعضاء F به همان صورت E می‌توانیم عمل کنیم.ا داریم:

$$(x0\ y0\ 1)\begin{pmatrix} F11 & F12 & F13 \\ F21 & F22 & F23 \\ F31 & F32 & F33 \end{pmatrix} = 0$$

که در آن مختصات x_0 و y_0 همان مختصات قرائت شده تصویری نقاط متناظر می‌باشند. برای حل F کافی است با هدف تعیین المان‌های F، معادله فوق را بازنویسی و فرم Ax=0 در آوریم که در آن x همان اعضای ماتریس F می‌باشند. برای این منظور داریم:

$$(x0x1\quad x0y1\quad x0\quad y0x1\quad y0y1\quad y0\quad x1\quad y1\quad 1)\begin{pmatrix} F11 \\ F12 \\ \dots \\ F33 \end{pmatrix} = 0$$

معادله فوق نشان دهنده یک سیستم هموژن است که می‌توان با استفاده از SVD به صورتی که قبلاً دیدیم آن را حل کرده در اینجا شبیه به حالت قبل از Preconditionig و Postconditining برای بهبود نتایج استفاده کنیم.

تمرین: برنامه ای بنویسید که ماتریس فاندامنتال را حساب کند

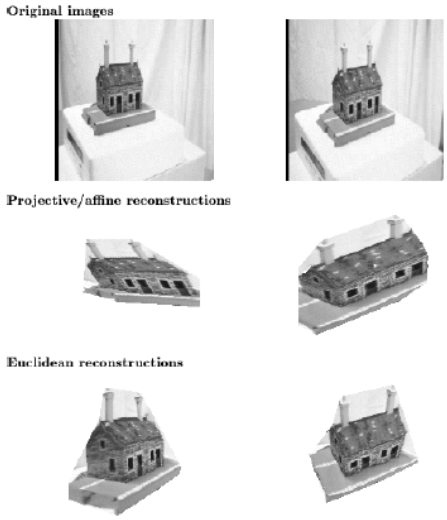
```
% Calculate the Fundamental matrix.
clear all
close all
K = [ 300 0 150; % intrinsic camera parameters
0 300 150;
0 0 1];
% These are the points in image 1
u1 = [
61.4195 102.1798 150.0000 68.3768 106.2098 150.0000 74.3208
109.6134 150.0000 176.0870 196.1538 174.0000 192.8571 172.2222
190.0000;
124.4290 136.1955 150.0000 167.2490 181.1490 197.2377 203.8325
219.1146 236.6025 127.4080 110.0296 170.7846 150.0000 207.7350
184.6410;
1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 ];
% These are the corresponding points in image 2
u2 = [
45.5272 63.4568 86.9447 61.6620 80.2653 104.1468 75.7981 94.7507
118.6606 135.5451 176.3357 147.5739 184.5258 157.8633 191.6139;
126.5989 136.7293 150.0000 165.9997 180.2731 198.5963 200.5196
217.7991 239.5982 125.7710 105.4355 172.3407 150.0000 212.1766
188.5687;
1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
1.0000 1.0000 1.0000 1.0000 1.0000 1.0000 ];
I1 = imread('I1.tif');
```

```
I2 = imread('I2.tif');
% Display points on the images for visualization
imshow(I1, []);
for i=1:length(u1)
x = round(u1(1,i)); y = round(u1(2,i));
rectangle('Position', [x-4 y-4 8 8], 'EdgeColor', 'r');
text(x+4, y+4, sprintf('%d', i), 'Color', 'r');
end
figure, imshow(I2, []);
for i=1:length(u2)
x = round(u2(1,i)); y = round(u2(2,i));
rectangle('Position', [x-4 y-4 8 8], 'EdgeColor', 'r');
text(x+4, y+4, sprintf('%d', i), 'Color', 'r');
end
% Get unnormalized image points
p1 = u1;
p2 = u2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Scale and translate image points so that the centroid of
% the points is at the origin, and the average distance of the
points to the
% origin is equal to sqrt(2).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
xn = p1(1:2,:); % xn is a 2xN matrix
N = size(xn,2);
t = (1/N) * sum(xn,2); % this is the (x,y) centroid of the
points
xnc = xn - t*ones(1,N); % center the points; xnc is a 2xN matrix
dc = sqrt(sum(xnc.^2)); % dist of each new point to 0,0; dc is
1xN vector
davg = (1/N)*sum(dc); % average distance to the origin
s = sqrt(2)/davg; % the scale factor, so that avg dist is
sqrt(2)
T1 = [s*eye(2), -s*t ; 0 0 1];
p1s = T1 * p1;
xn = p2(1:2,:); % xn is a 2xN matrix
N = size(xn,2);
t = (1/N) * sum(xn,2); % this is the (x,y) centroid of the
points
xnc = xn - t*ones(1,N); % center the points; xnc is a 2xN matrix
dc = sqrt(sum(xnc.^2)); % dist of each new point to 0,0; dc is
1xN vector
davg = (1/N)*sum(dc); % average distance to the origin
s = sqrt(2)/davg; % the scale factor, so that avg dist is
sqrt(2)
T2 = [s*eye(2), -s*t ; 0 0 1];
p2s = T2 * p2;
% Compute fundamental matrix F from point correspondences.
% We know that p1s' F p2s = 0, where p1s,p2s are the scaled
image coords.
% We write out the equations in the unknowns F(i,j)
% A x = 0
A = [p1s(1,:)'.*p2s(1,:)' p1s(1,:)'.*p2s(2,:)' p1s(1,:)' ...
p1s(2,:)'.*p2s(1,:)' p1s(2,:)'.*p2s(2,:)' p1s(2,:)' ...
p2s(1,:)' p2s(2,:)' ones(length(p1s),1)];
% The solution to Ax=0 is the singular vector of A corresponding
to the
```

```
% smallest singular value; that is, the last column of V in
A=UDV'
[U,D,V] = svd(A);
x = V(:,size(V,2)); % get last column of V
% Put unknowns into a 3x3 matrix. Transpose because Matlab's
"reshape"
% uses the order F11 F21 F31 F12 ...
Fscale = reshape(x,3,3)';
% Force rank=2
[U,D,V] = svd(Fscale);
Fscale = U*diag([D(1,1) D(2,2) 0])*V';
% Undo scaling
F = T1' * Fscale * T2;
disp('Calculated fundamental matrix:');
disp(F);
save F
```

برنامه ای که در بالا آمده است fundamental.m است نام دارد که عین برنامه essential.m است با این تفاوت که در قسمت ورود مختصات تصویری آن‌ها را دیگر نرمال نمی‌کند. برنامه دیگر draw fundamental.m است که خطوط اپی پولار را ترسیم می‌کند.

نهایتاً شبیه به حالت E، می‌توان با استفاده از F نیز مختصات سه‌بعدی نقاط را محاسبه نمود با این تفاوت که شکل عارضه الزاماً در سیستم اقلیدسی نیست یعنی خطوط متقاطع الزاماً متقاطع به دست نمی‌آیند تفاوت بین اشکال به دست آمده در E، F در شکل زیر دیده می‌شود.



تصاویر اصلی، تصاویر باز سازی شده توسط ماتریس فاندامنتال و ضروری (به ترتیب از بالا به پایین)

۱۱ Ransac (Random Sample Consensus)

در بخش‌های قبل دیدیم که برای حل ماتریس‌های E، F و نیز به دست آوردن مختصات سه بعدی نقاط نیازمند تناظر یابی نقاط در تصاویر مختلف هستیم. در پروسه‌هایی که این عملیات (تناظر یابی) توسط اپراتور انجام شود، دقت بالا بوده و می‌توان با اطمینان از نقاط به دست آمده برای حل معادلات استفاده کرد. فقط کافی است با یک حد آستانه معقول نقاط اشتباه را کشف و از محاسبات خارج نمود. اما در کاربردهای ماشین بینایی می‌دانیم که معمولاً هدف آنست که پردازش‌ها، از جمله استخراج و تناظر یابی نقاط به صورت اتوماتیک صورت گیرد. همانطور که قبلاً هم اشاره کرده بودیم بحث تناظر یابی اتوماتیک نقاط با توجه به پیچیدگی‌های عارضه جزء مسائل دشوار و پر ابهام در بینایی ماشین می‌باشد. برای حل این موضوع حداقل کاربرد که باید انجام داد آنست که نقاط اشتباه را کشف و از چرخه محاسبات خارج سازیم. روش RANSAC یکی از چنین تکنیک‌هایی است که با استفاده از آن می‌توان نقاط که به اشتباه تناظر یابی شده‌اند را کشف و حذف نمود. در این روش مجموعه‌ای از نقاط تناظر یابی شده به صورت رندوم انتخاب می‌شوند و با استفاده آن‌ها و رابطه $u_0^T.F.u_1$ ماتریس فاندا منتال محاسبه می‌شود. باقیمانده نقاط دیگر استخراج شده از طریق $e = u_0^T.F.u_1$ محاسبه شده و در صورتی که تعداد نقاط مورد قبول (inliers) از یک عدد مشخص بالاتر بود، محاسبات متوقف و ماتریس نهایی F با استفاده از تمام نقاط مورد قبول مجدداً حساب می‌شود. در این صورت تمامی نقاطی که در رابطه $u_0^T.F.u_1 = 0$ صدق کنند (یا باقیمانده آن‌ها از یک حد آستانه t کوچکتر باشد)، به عنوان نقاط صحیح تشخیص داده می‌شوند.

تمرین: برنامه‌ای بنویسید که نقاط را از یک تصویر به صورت اتوماتیک استخراج نموده در تصویر دوم تناظریابی کرد و پس از کشف نقاط صحیح با استفاده از تکنیک Ransac، ماتریس فاندا منتال را تشکیل دهد.

حل: برنامه زیر مراحل بالا را انجام می‌دهد.

```
% Now match these points to another image
I2 = double(imread('test012.jpg'));
THRESH = 0.8; % correlation score threshold
% For each corner point found above, we will extract a template
subimage of
% size NxN centered on that point, and try to match it to the second
image.
nPts = 0; % actual number of matching points found
for i=1:maxPts
x = cols(indices(i)); % Location of corner point
y = rows(indices(i));
% Get the template from the first image, surrounding this point
```

```
M = floor(N/2);
if x<=M || x>size(I1,2)-M continue; end
if y<=M || y>size(I1,1)-M continue; end
T = I1(y-M:y+M, x-M:x+M);
C = normxcorr2(T,I2); % Do normalized cross correlation
% The scores image C is bigger than I, by M rows and M columns along
% the sides and the top and bottom. So when we find the location of
% the peak score, we should subtract M from the indices.
cmax = max(C(:));
if cmax < THRESH continue; end
[yp xp] = find(C==cmax);
yp = yp-M;
xp = xp-M;
fprintf('Point %d matches with score=%f\n', i, cmax);
nPts = nPts + 1;
x1(1,nPts) = x;
x1(2,nPts) = y;
x1(3,nPts) = 1;
x2(1,nPts) = xp;
x2(2,nPts) = yp;
x2(3,nPts) = 1;
End
% Draw matched points
figure, imshow(I1, []);
for i=1:nPts
x = x1(1,i);
y = x1(2,i);
rectangle('Position', [x-N/2 y-N/2 N N], 'EdgeColor', 'y');
text(x+N/2,y+N/2, sprintf('%d', i), 'Color', 'w'); % label with id
number
end
figure, imshow(I2, []);
for i=1:nPts
x = x2(1,i);
y = x2(2,i);
rectangle('Position', [x-N/2 y-N/2 N N], 'EdgeColor', 'y');
text(x+N/2,y+N/2, sprintf('%d', i), 'Color', 'w'); % label with id
number
end

% Grab 8 matching points at random from x1,x2
v = randperm(length(x1));
disp('Choosing points:'); disp(v(1:8));
p1 = x1(:,v(1:8));
p2 = x2(:,v(1:8));
figure, imshow(I1, []);
for i=1:8
x = p1(1,i);
y = p1(2,i);
rectangle('Position', [x-N/2 y-N/2 N N], 'EdgeColor', 'y');
text(x+N/2,y+N/2, sprintf('%d', v(i)), 'Color', 'w'); % label
end
figure, imshow(I2, []);
for i=1:8
x = p2(1,i);
y = p2(2,i);
rectangle('Position', [x-N/2 y-N/2 N N], 'EdgeColor', 'y');
```

```
text(x+N/2,y+N/2, sprintf('%d', v(i)), 'Color', 'w'); % label
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Scale and translate image points so that the centroid of
% the points is at the origin, and the average distance of the points to the
% origin is equal to sqrt(2).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
xn = p1(1:2,:); % xn is a 2xN matrix
N = size(xn,2);
t = (1/N) * sum(xn,2); % this is the (x,y) centroid of the points
xnc = xn - t*ones(1,N); % center the points; xnc is a 2xN matrix
dc = sqrt(sum(xnc.^2)); % dist of each new point to 0,0; dc is 1xN vector
davg = (1/N)*sum(dc); % average distance to the origin
s = sqrt(2)/davg; % the scale factor, so that avg dist is sqrt(2)
T1 = [s*eye(2), -s*t ; 0 0 1];
p1s = T1 * p1;
xn = p2(1:2,:); % xn is a 2xN matrix
N = size(xn,2);
t = (1/N) * sum(xn,2); % this is the (x,y) centroid of the points
xnc = xn - t*ones(1,N); % center the points; xnc is a 2xN matrix
dc = sqrt(sum(xnc.^2)); % dist of each new point to 0,0; dc is 1xN vector
davg = (1/N)*sum(dc); % average distance to the origin
s = sqrt(2)/davg; % the scale factor, so that avg dist is sqrt(2)
T2 = [s*eye(2), -s*t ; 0 0 1];
p2s = T2 * p2;
% Compute fundamental matrix F from point correspondences.
% We know that p1s' F p2s = 0, where p1s,p2s are the scaled image coords.
% We write out the equations in the unknowns F(i,j)
% A x = 0
A = [p1s(1,:)'.*p2s(1,:)' p1s(1,:)'.*p2s(2,:)' p1s(1,:)' ...
p1s(2,:)'.*p2s(1,:)' p1s(2,:)'.*p2s(2,:)' p1s(2,:)' ...
p2s(1,:)' p2s(2,:)' ones(length(p1s),1)];
% The solution to Ax=0 is the singular vector of A corresponding to the
% smallest singular value; that is, the last column of V in A=UDV'
[U,D,V] = svd(A);
x = V(:,size(V,2)); % get last column of V
% Put unknowns into a 3x3 matrix. Transpose because Matlab's "reshape"
% uses the order F11 F21 F31 F12 ...
Fscale = reshape(x,3,3)';
% Force rank=2
[U,D,V] = svd(Fscale);
Escale = U*diag([D(1,1) D(2,2) 0])*V';
% Undo scaling
F = T1' * Fscale * T2;
disp('Calculated fundamental matrix:');
disp(F);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Check error residuals for all matching pairs of points
figure
for i=1:length(x2)
```

```
subplot(1,2,1), imshow(I1,[]);
% The product l=F*p2 is the equation of the epipolar line
corresponding
% to p2, in the first image. Here, l=(a,b,c), and the equation of the
% line is ax + by + c = 0.
l = F * x2(:,i);
% Let's find two points on this line. First set x=1 and solve
% for y, then set x=L and solve for y.
L = size(I1,2);
pLine0 = [ 1; (-l(3)-l(1)*(1))/l(2); 1];
pLine1 = [ L; (-l(3)-l(1)*(L))/l(2); 1];
line([pLine0(1) pLine1(1)], [pLine0(2) pLine1(2)], 'Color', 'r');
rectangle('Position', [x1(1,i)-4 x1(2,i)-4 8 8], 'EdgeColor', 'r');
text(x1(1,i)+4, x1(2,i)+4, sprintf('%d', i), 'Color', 'r');
subplot(1,2,2), imshow(I2,[]);
rectangle('Position', [x2(1,i)-4 x2(2,i)-4 8 8], 'EdgeColor', 'g');
text(x2(1,i)+4, x2(2,i)+4, sprintf('%d', i), 'Color', 'g');
% Calculate residual error. The product p1'*F*p2 should = 0. The
% difference is the residual.
res = x1(:,i)' * F * x2(:,i);
fprintf('Residual is %f\n', res);
pause%(0.25);
end
```

در این برنامه بخش اول نقاط را از تصویر اول به صورت اتوماتیک استخراج می‌کند. پس از آن حول این نقاط یک پنجره در نظر گرفته و با استفاده از تکنیک cross-carrelation نقاط متناظر را در تصویر بعدی تعیین می‌کند. سپس با استفاده از الگوریتم RANSAC نقاط را پالایش و F را نهایتاً با لحاظ preconditioning و postconditioning حساب می‌کند. در این مرحله برای اینکه با استفاده از ransac نقاط را پالایش کند به این شرح عمل می شود:

۱. تعداد S نقطه را به صورت رندوم انتخاب و با استفاده از آن‌ها مدل را تشکیل بده. (در اینجا مدل F است. لذا، F حساب می‌شود)

۲. با استفاده از مدل تشکیل شده (در اینجا $u_0^T.F.u_1$) تمامی نقاطی که باقیمانده آن‌ها در حد قابل قبول است را مشخص کن. طول این مجموعه آماری را Si می‌نامیم.

۳. در صورتی که سایز Si (یعنی همان تعداد نقاط inliers یا همان نقاط مورد قبول) از یک حد آستانه T بزرگتر بود عملیات را متوقف و مدل نهایی (همان ماتریس F در اینجا) را با استفاده از نقاط S محاسبه کن.

بعد از تعداد N تکرار نمونه آماری S با بزرگترین Si انتخاب و مدل با استفاده از آن به صورت نهایی تعیین می‌شود.

۱۱-۱ تعداد نمونه‌های مورد نیاز در روش RANSAC

سوالی که وجود دارد آنست که برای اینکه روش ransac جواب دهد، چند بار تکرار باید انجام شود.