Assignment5

`视觉组考核(OpenCV)`

> 前期的任务带领大家快速入门了OpenCV的冰山一角(但是**举足轻重!**)，同时也偏重应用。接下来的任务中理论的比重会更大，数学的过程大家无可避免地需要掌握。为了更系统地学习，接下来的内容会大量参考或者来源于多伦多大学**CSC 420: Introduction to Image Understanding** 。里面的全部内容希望大家都能掌握，虽然我们可能不会全部涉及。**2021年的课件已经可以访问**：http://www.cs.toronto.edu/~fidler/slides/2021Winter/CSC420/lecture1.pdf ，仅需在地址栏中修改*lecture* 后的序号

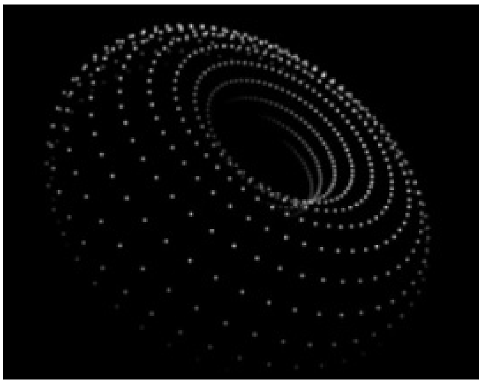
2D Image Transformations

Introduction

首先需要一些基础的线性代数知识，你需要了解矩阵，理解矩阵乘法。以下面这个视频中的内容为例：
why you NEED math for programming (or https://www.youtube.com/watch?v=sW9npZVpiMI)
具体的原理在以下两个网页中有详细介绍：

- https://www.a1k0n.net/2011/07/20/donut-math.html
- Why a Spinning Donut is Pure Math
  这个的叙述的更为详细，但是在3D perspective rendering处给出的演示结果并没有体现出其所希望实现的效果，因此这部分内容请参考第一个网页

第二个网页中给出了**矩阵，矩阵乘法**的链接。你也可以学习**麻省理工公开课 线性代数 MIT 18.06 Linear Algebra 中英双语字幕**的前3节(事实上你最好在寒假把全P学完)；以及**3Blue1Brown**的视频

小任务：实现出第一个网页中最后的动态效果：

Tips：网页中计算结果都是三维的，为了显示在屏幕上，最简单的方法就是忽略 z，仅保留x, y

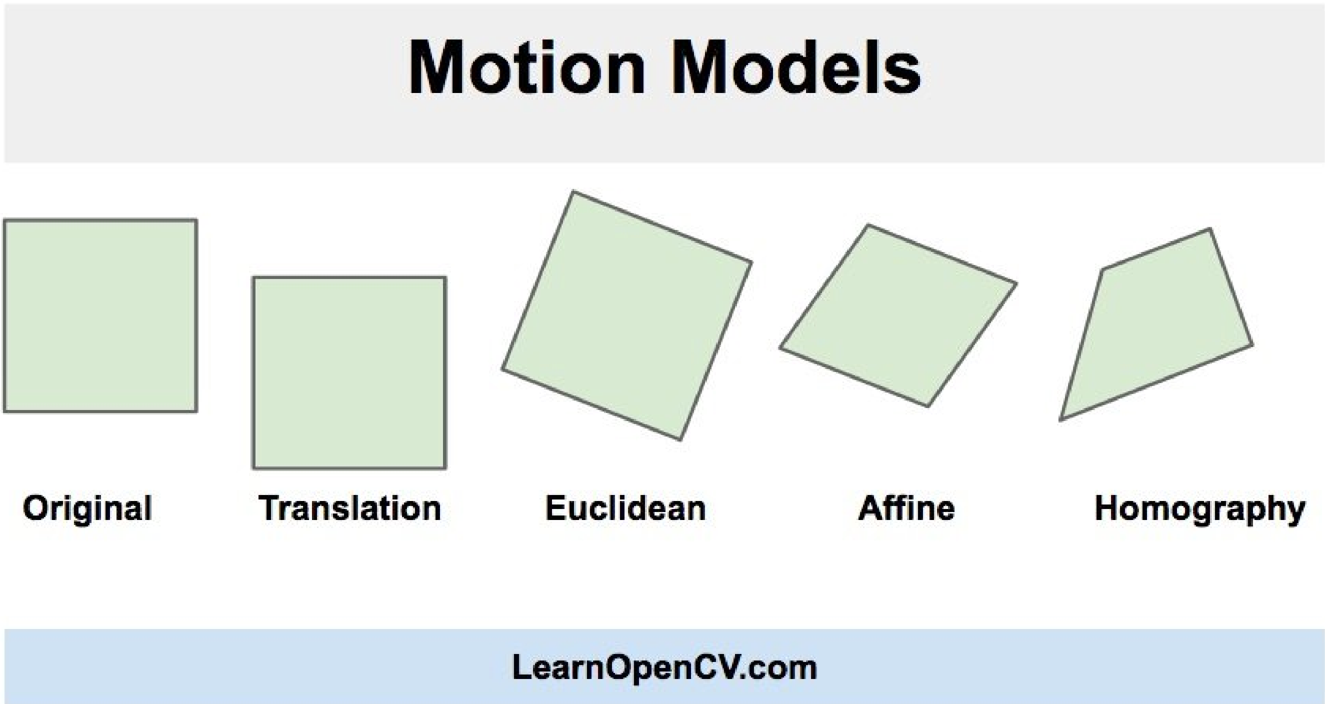
仿射变换，投影变换(Homography)

总参考：http://www.cs.toronto.edu/~fidler/slides/2021Winter/CSC420/lecture9.pdf

Affine Transformation 与 **Projective Transformation** 是重要的概念，并且要被严格区分。

Affine Transformation

An **Affine Transform encodes translation (move), scale, rotation and shear**. The image below illustrates how an affine transform can be used to change the shape of a square. Note that using an affine transform you can change the shape of a square to a parallelogram at any orientation and scale. However, **the affine transform is not flexible enough to transform a square to an arbitrary quadrilateral**. In other words, **after an affine transform parallel lines continue to be parallel**. 【1】



Affine transformations are combinations of: **Linear transformations** and **Translations**
In OpenCV an affine transform is a 2×3 matrix. The first two columns of this matrix encode rotation, scale and shear, and the last column encodes translation (i.e. shift). 【1】

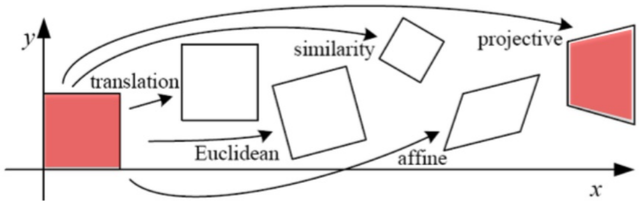
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & e \\ c & d & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Properties of affine transformations: see 【2】，page 3

> Tips：**Closed under composition**：仿射变换的组合依然是仿射变换

继续阅读【2】直至第6页(或下图)，了解 2D Image Transformations

2D Image Transformations



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

- These transformations are a nested set of groups
- Closed under composition and inverse is a member

[source: R. Szeliski]

> Tips：#DoF：自由度个数
> **These transformations are a nested set of groups**：表格中每种变换都是下一行所述变换的子集

Projective Transformation (Homography)

Projective Transformation = Homography

What is Homography ?

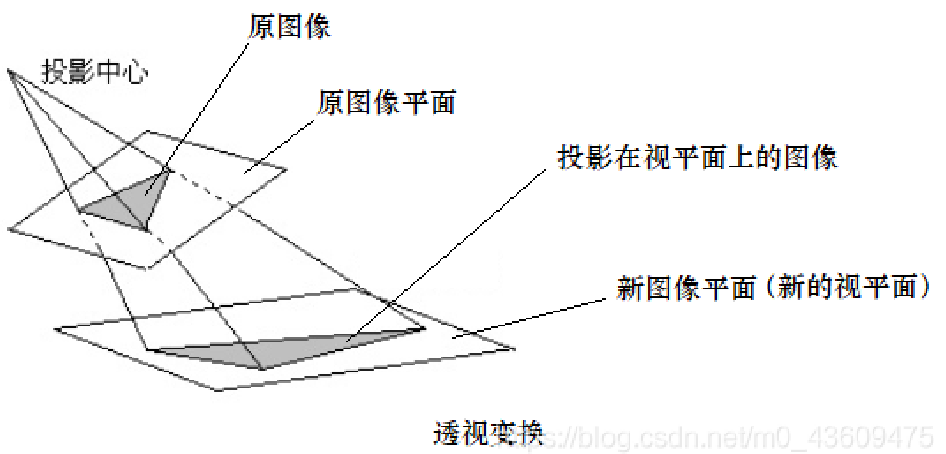
A Homography is a **transformation (a 3×3 matrix)** that maps the points in one image to the corresponding points in the other image.【3】

$$w\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Properties of projective transformations: see【2】，page 7

> Affine transformation is a special case, where g = h = 0 and i = 1

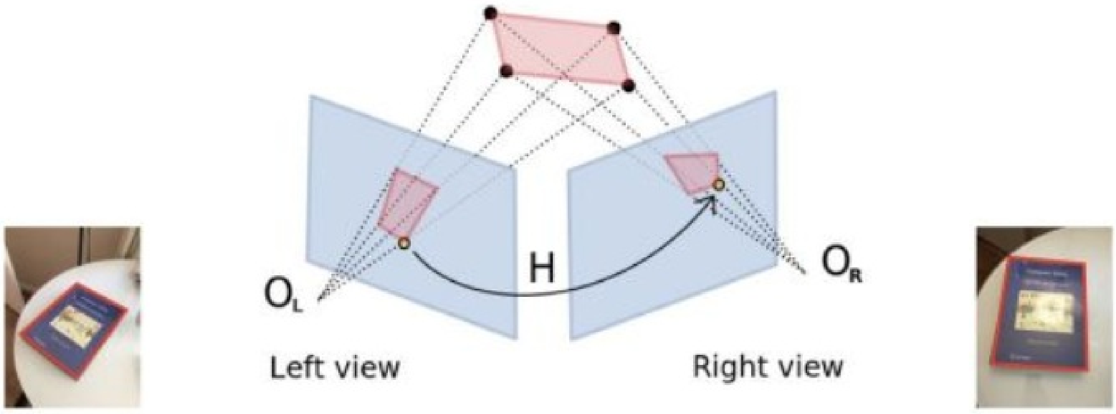
投影变换自动包含了，或者说一定情况下可以理解为**透视变换**(**Perspective Transformation**)。这点你可以从【2】page 5看出，它可以符合视觉上的透视关系。必须指出，Homography是 $2\mathrm{D} \to 2\mathrm{D}$ 的映射，而透视往往意味着三维到二维的映射，这里所说的**透视变换**是三维中两***平面***在透视关系下的对应，本质是将图像投影到一个新的视平面：



【4】

例如将照片中的**平面物体**(且必须是)转换到我们正视它时的视角：

由于需要**保证透视关系**成立，这个变换**必须是单个**透视变换矩阵所描述(forward or inverse)，因为**透视变换不满足*Closed under composition***，即透视变换的复合一般**不是**透视变换，但该复合是**Homography**，因为透视变换是**Homography**且**Homography**构成*群*(封闭)，例如下图中的变换**H**一定是**Homography**而常常不是**透视变换**：



透视变换最重要的一个性质是*连接对应点的直线*交于同一点(投影中心)，对于上图可以举出一维情形下的反例：

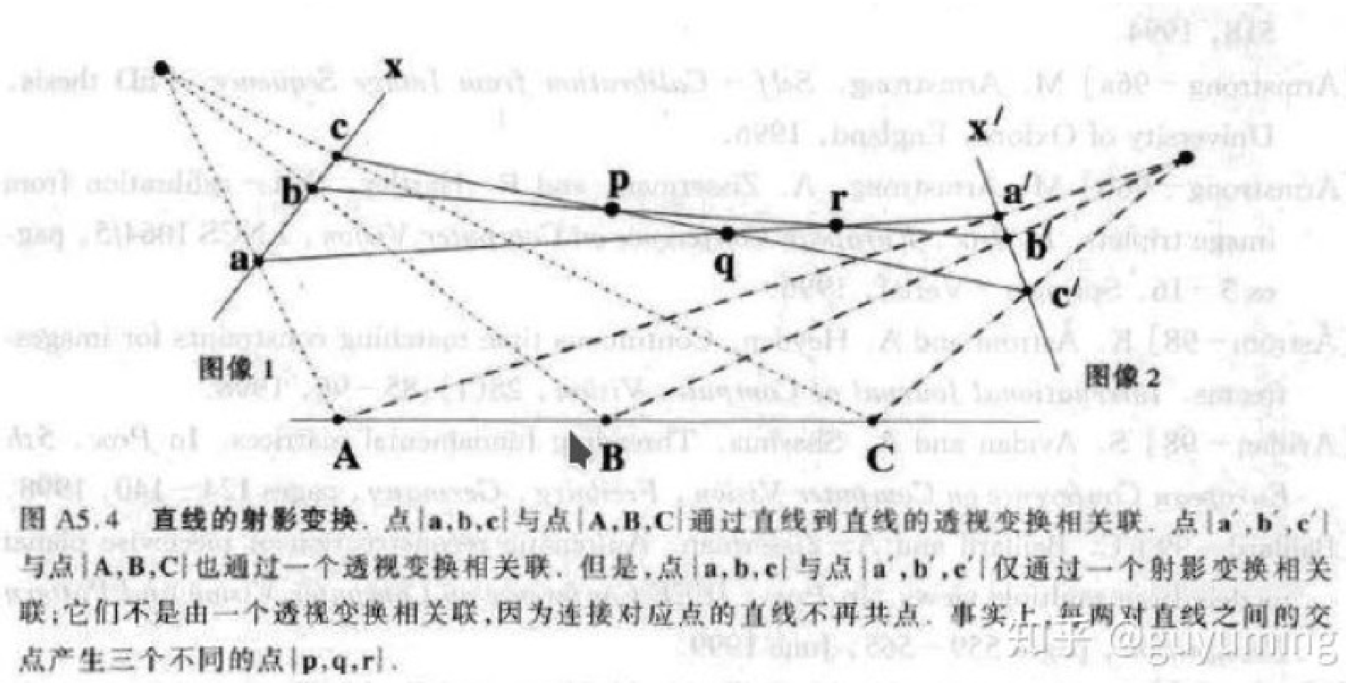


图 A5.4 直线的射影变换. 点{a,b,c}与点{A,B,C}通过直线到直线的透视变换相关联. 点{a′,b′,c′}与点{A,B,C}也通过一个透视变换相关联. 但是,点{a,b,c}与点{a′,b′,c′}仅通过一个射影变换相关联;它们不是由一个透视变换相关联,因为连接对应点的直线不再共点. 事实上,每两对直线之间的交点产生三个不同的点{p,q,r}.

参考：https://www.zhihu.com/question/266997043/answer/1540850374

而对于**普遍的三维到二维的*透视关系***，称其为：**Perspective Projection Transformation** 或 **Perspective Projection**，这将涉及到以后相机模型与成像的相关内容。

> 注意：不要混淆以上名词！由于***Projective***这个词实在难以同其物理意义分开，我们以后大多使用***Homography***以避免产生如下的抱怨：
> *It's common in computer graphics to call a plane homography a "plane projective transform." To muddy the waters further, the mapping from P3 to P2 that models a pinhole camera is also called a "projective transformation."*【5】

Applications

继续阅读【2】至第31页，了解 Homography的应用

Solving for Homographies

let (x_i, y_i) be a point on the reference image, and (x_i', y_i') its match in another image. A homography H maps (x_i, y_i) to (x_i', y_i') :

$$a\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix}\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$\because a = h_{20}x_i + h_{21}y_i + h_{22}$, we can get rid of that a on the left : see 【2】, page 33

通过基础的变换，得到：

$$\begin{aligned} h_{00}x_i + h_{01}y_i + h_{02} - x_i'(h_{20}x_i - h_{21}y_i - h_{22}) = 0 \\ h_{10}x_i + h_{11}y_i + h_{12} - y_i'(h_{20}x_i - h_{21}y_i - h_{22}) = 0 \end{aligned}$$

写成矩阵形式：

$\mathrm{A}_{2\times 9}\mathbf{h}_{9\times 1} = \mathbf{0}_{2\times 1} \xrightarrow{\text{推广}} \mathrm{A}_{2\mathrm{n}\times 9}\mathbf{h}_{9\times 1} = \mathbf{0}_{2\mathrm{n}\times 1}$: see 【2】, page 35-36

上述方程可能无解($\mathrm{h} = 0$没有任何用处)(与n有关，$n > 4$后可能常常如此，但$n = 1$时有且有无穷多解)，但可以找到一个最近似的 h，使得$||\mathrm{Ah}||$最小，即：

$$\min_{\mathrm{h}} \ (\mathrm{Ah})^\top (\mathrm{Ah})$$

$(\mathrm{Ah})^\top(\mathrm{Ah}) = \mathrm{h}^\top(\mathrm{A}^\top\mathrm{A})\mathrm{h} \geq 0$ ，显然$(\mathrm{A}^\top\mathrm{A})$是对称矩阵且positive semidefinite(or positive definite).

- $(\mathrm{A}^\top\mathrm{A})$是对称矩阵 $\rightarrow (\mathrm{A}^\top\mathrm{A}) = \mathrm{Q}\Lambda\mathrm{Q}^\top = \lambda_1\mathrm{q}_1\mathrm{q}_1^\top + \cdots + \lambda_9\mathrm{q}_9\mathrm{q}_9^\top$
- $(\mathrm{A}^\top\mathrm{A})$positive semidefinite $\rightarrow \lambda_1, \ldots, \lambda_9 \geq 0$

Solution: h = eigenvector of $\mathrm{A}^\top\mathrm{A}$ with smallest eigenvalue

- $\mathrm{h}^\top(\mathrm{A}^\top\mathrm{A})\mathrm{h} = \lambda_1\mathrm{h}^\top\mathrm{q}_1\mathrm{q}_1^\top\mathrm{h} + \cdots + \lambda_9\mathrm{h}^\top\mathrm{q}_9\mathrm{q}_9^\top\mathrm{h} = \lambda_1(\mathrm{q}_1^\top\mathrm{h})^2 + \cdots + \lambda_9(\mathrm{q}_9^\top\mathrm{h})^2 \geq 0$
- $\min_{\mathrm{h}} \mathrm{h}^\top(\mathrm{A}^\top\mathrm{A})\mathrm{h} = \min(\lambda_\mathrm{i})|_{\mathrm{h}=\mathrm{q_i}}$

最后的问题：How many matches do I need to estimate h?

> 高等几何中有定理如下：任一**二维投影对应**可由已知4对**对定点**(每一方中无三点共线)唯一确定。
> 所以**至少需要4对点**以求Homography

计算一个Homography的步骤：

- 给出至少四对对应点(可以人为给出，在有些应用中则需要自动地匹配：see【2】，page 37)
- 计算Homography

思考：https://learnopencv.com/warp-one-triangle-to-another-using-opencv-c-python/中只利用了三角形的顶点(三对对应点)，为什么？(注意他使用的函数：**getAffineTransform()**)

SUMMARY

- A homography is a mapping between projective planes
- You need at least 4 correspondences (matches) to compute it

应用案例

- Image Rotation and Translation
- https://learnopencv.com/warp-one-triangle-to-another-using-opencv-c-python/
- 【2】中偷拍试卷的例子和Panorama Stitching(page 38-39)
- https://learnopencv.com/homography-examples-using-opencv-python-c/
  - What is the difference between `findHomography()` and `getPerspectiveTransform()` ?

    > `getPerspectiveTransform` computes the transform using 4 correspondences (which is the minimum required to compute a homography/perspective transform) where as `findHomography` computes the transform even if you provide more than 4 correspondencies.
    > The `getPerspectiveTransform` is the base for `findHomography`, and it is useful in many situations where you **only have 4 points, and you know they are the correct ones**. The findHomography is usually used with sets of points **detected automatically** - you can find many of them, but with low confidence.【6】

  - 关于 `warpPerspective()` ：
    在这个案例中，使用 `findHomography` 计算Homography之后依然使用的是 `warpPerspective` ，因此严格来说，这个函数的命名是略不严谨的。

提交要求

1. 包含以下内容的zip压缩包
    - 一份学习笔记(PDF)
    - 小任务的代码(**OpenCV支持矩阵运算，禁止使用其他科学计算库**)
2. 学习笔记需要包含：
    - 简要归纳你学到的知识
    - 列举应用案例(除了多伦多大学课件中的内容)中与本讲有关的OpenCV函数
    - 小任务的思路
    - 对思考问题的回答
    - **心得体会**
3. 截止日期：**见群通知**
4. 邮件格式：
    - 邮件以 **视觉考核-姓名-年级** 的格式命名
    - zip包命名同上
5. 发送至正确的邮箱