Direct Linear Transformation

这里是对DLT的一些理解，顺便结合代码进行分析。

这里是一个CMU CS关于DLT的详细例子解释帮助理解:10.2 2D Alignment - DLT (cmu.edu)

参考例子设法，设一对对应点如下$\tilde{x} = (x, y, w)^T, \tilde{x}' = (x', y', w')^T$

设H矩阵如下:

$$h = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

我们令

$$\begin{aligned} \tilde{h}_1^T &= [h_1, h_2, h_3] \\ \tilde{h}_2^T &= [h_4, h_5, h_6] \\ \tilde{h}_3^T &= [h_7, h_8, h_9] \end{aligned}$$

满足如下等式

$$\tilde{x}' = h\tilde{x}$$

可以写出三个等式如下

$$\begin{aligned} x' &= h_1 x + h_2 y + h_3 w \quad 1 \\ y' &= h_4 x + h_5 y + h_6 w \quad 2 \\ w' &= h_7 x + h_8 y + h_9 w \quad 3 \end{aligned}$$

可以任意用两个等式联立生成一个等式。例如

2，3联立

$$
\begin{gathered}
y'w' = h_4 x w' + h_5 y w' + h_6 w w' \\
y'w' = h_7 y' x + h_8 y' y + h_9 y' w \\
\rightarrow -h_4 x w' - h_5 y w' - h_6 w w' + h_7 y' x + h_8 y' y + h_9 y' w = 0 \\
\rightarrow [0, 0, 0, -w'x, -w'y, -w'w, y'x, y'y, y'w] \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} \\
\rightarrow [\mathbf{0^T}, -\mathbf{w'\tilde{x}^T}, \mathbf{y'\tilde{x}^T}] \begin{bmatrix} \tilde{\mathbf{h}}_1 \\ \tilde{\mathbf{h}}_2 \\ \tilde{\mathbf{h}}_3 \end{bmatrix}
\end{gathered}
$$

这里就获得了DLT中A_i矩阵的第一行表达式，因此，两两组合就可以得到lec中的A_i (1,3获得第二行，1，2获得第三行)

最终结果如下:

$$
\underbrace{\begin{bmatrix} \mathbf{0}^\top & -\tilde{w}_i' \tilde{\mathbf{x}}_i^\top & \tilde{y}_i' \tilde{\mathbf{x}}_i^\top \\ \tilde{w}_i' \tilde{\mathbf{x}}_i^\top & \mathbf{0}^\top & -\tilde{x}_i' \tilde{\mathbf{x}}_i^\top \\ -\tilde{y}_i' \tilde{\mathbf{x}}_i^\top & \tilde{x}_i' \tilde{\mathbf{x}}_i^\top & \mathbf{0}^\top \end{bmatrix}}_{\mathbf{A}_i} \underbrace{\begin{bmatrix} \tilde{\mathbf{h}}_1 \\ \tilde{\mathbf{h}}_2 \\ \tilde{\mathbf{h}}_3 \end{bmatrix}}_{\tilde{\mathbf{h}}} = \mathbf{0}
$$

对于一组对应点，实际上有效的等式只有2组，例如1，2，因为秩是2，任意一行可以通过剩余两行进行线性组合获得。因此对于N组的对应点，我们只需要构造一个$2N \times 9$的矩阵即可。

代码如下，其实比较好理解，能够推导出上述公示后，然后取前两个等式进行组合即可。

```
##########################
#### Exercise Function ####
```

```
###########################
def get_Ai(xi_vector, xi_prime_vector):
    ''' Returns the A_i matrix discussed in the lecture for input vectors.

    Args:
        xi_vector (array): the x_i vector in homogeneous coordinates
        xi_vector_prime (array): the x_i_prime vector in homogeneous coordinates
    '''
    assert(xi_vector.shape == (3,) and xi_prime_vector.shape == (3,))

    # Insert your code here
    zero_vector = np.zeros((3,), dtype=np.float32)
    xi, yi, wi = xi_prime_vector

    Ai = np.stack([
        np.concatenate([zero_vector, -wi*xi_vector, yi*xi_vector]),
        np.concatenate([wi*xi_vector, zero_vector, -xi*xi_vector]),
    ])
    assert(Ai.shape == (2, 9))
    return Ai

###########################
#### Exercise Function ####
###########################
def get_A(points_source, points_target):
    ''' Returns the A matrix discussed in the lecture.

    Args:
        points_source (array): 3D homogeneous points from source image
        points_target (array): 3D homogeneous points from target image
    '''
    N = points_source.shape[0]

    # Insert your code here
    correspondence_pairs = zip(points_source, points_target)

    A = np.concatenate([get_Ai(p1, p2) for (p1, p2) in correspondence_pairs])
    assert(A.shape == (2*N, 9))
    return A
```

矩阵（二）：为什么Ax=0的解为最小奇异值对应的向量？_矩阵最小奇异值-CSDN博客

这个博客比较详细解释了为什么是取V对应最小的奇异值的奇异向量。

The solution to the above optimization problem is the singular vector corresponding to the smallest singular value of A (i.e., the last column of V when decomposing $A = UDV^T$. The resulting algorithm is called Direct Linear Transformation.