

رسالة محمد

مبانی بینایی کامپیوتر

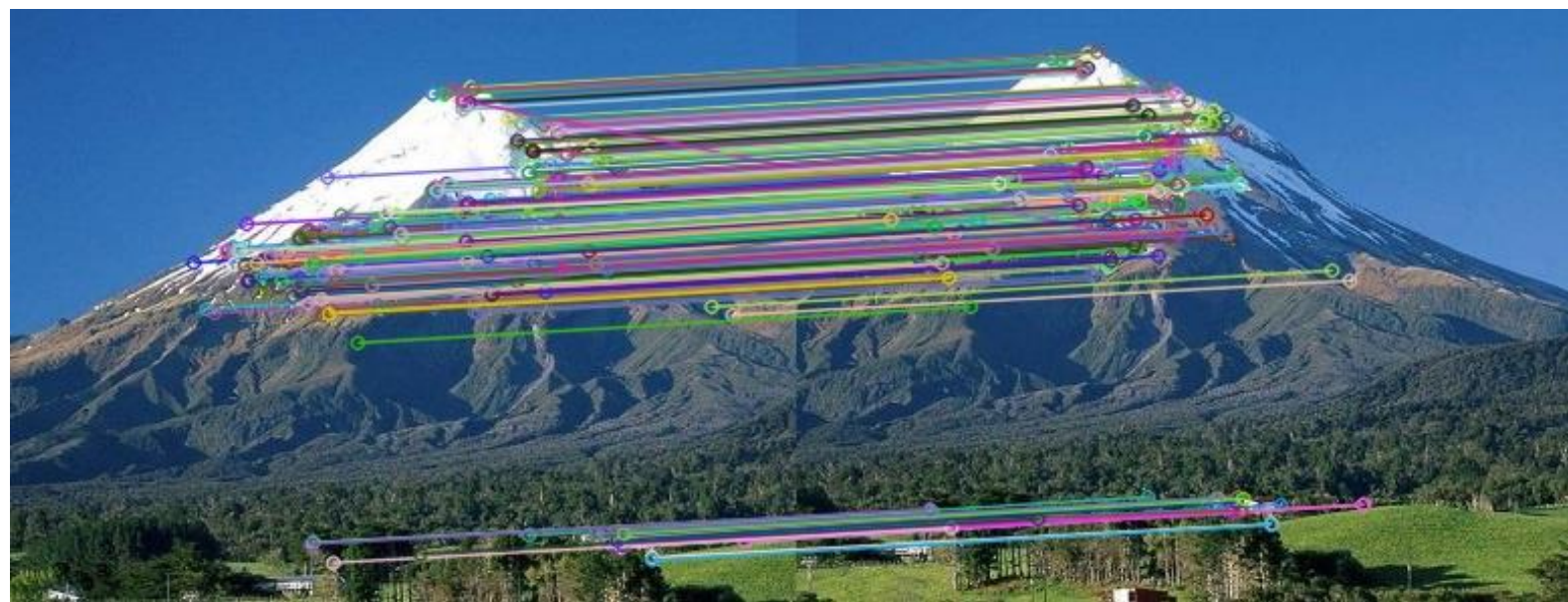
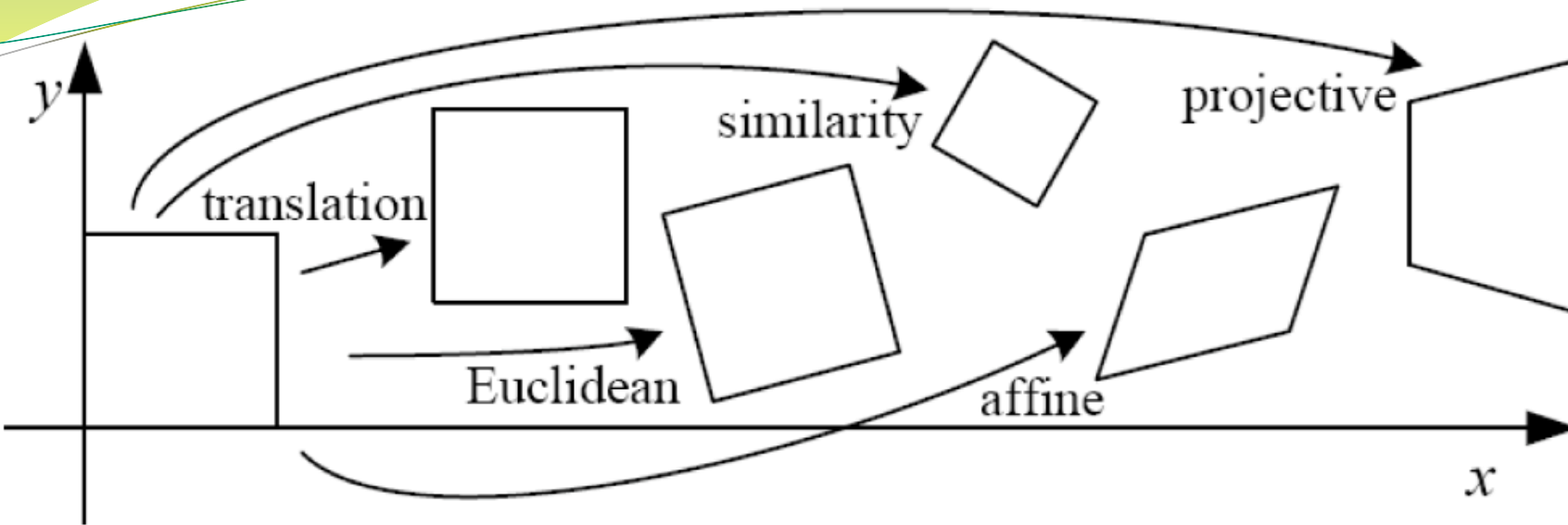
مدرس: محمدرضا محمدی

۱۴۰۱

تناظر و هم‌ترازی تصاویر

Correspondence and Image Alignment

تبدیل‌های هندسی

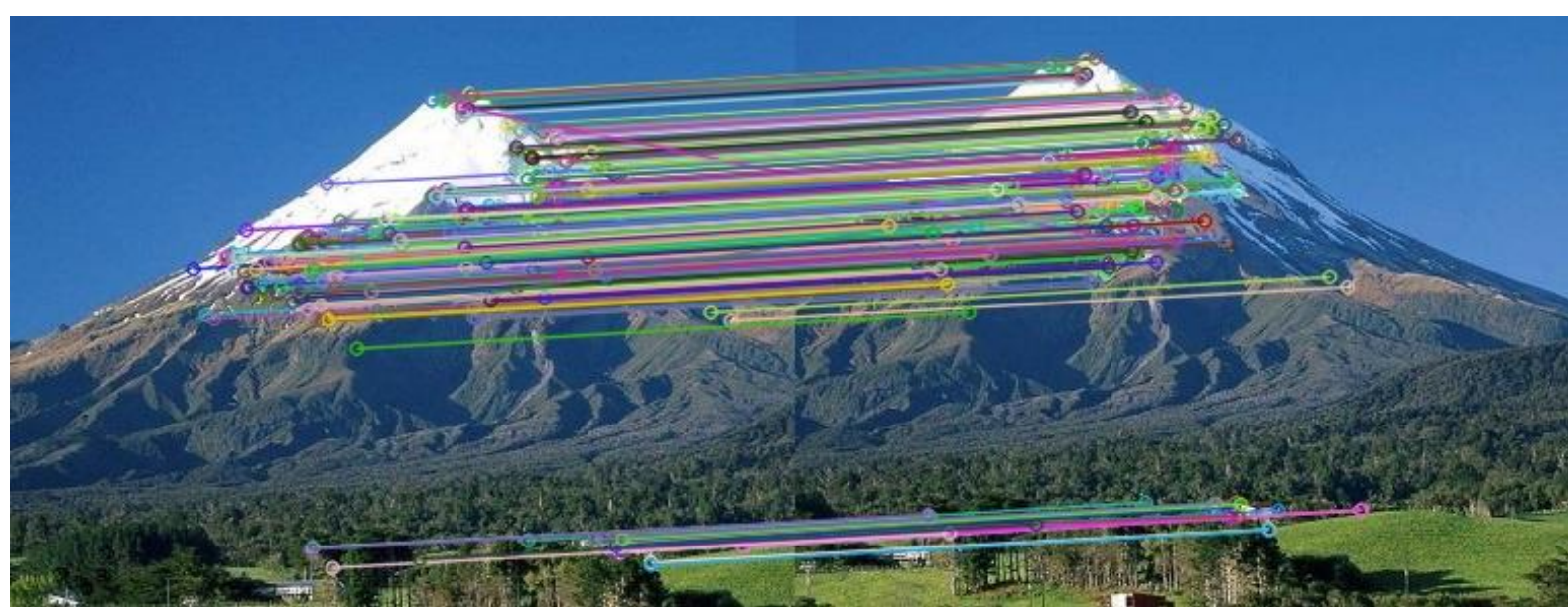


$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = T \left(\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \right)$$

انتقال

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

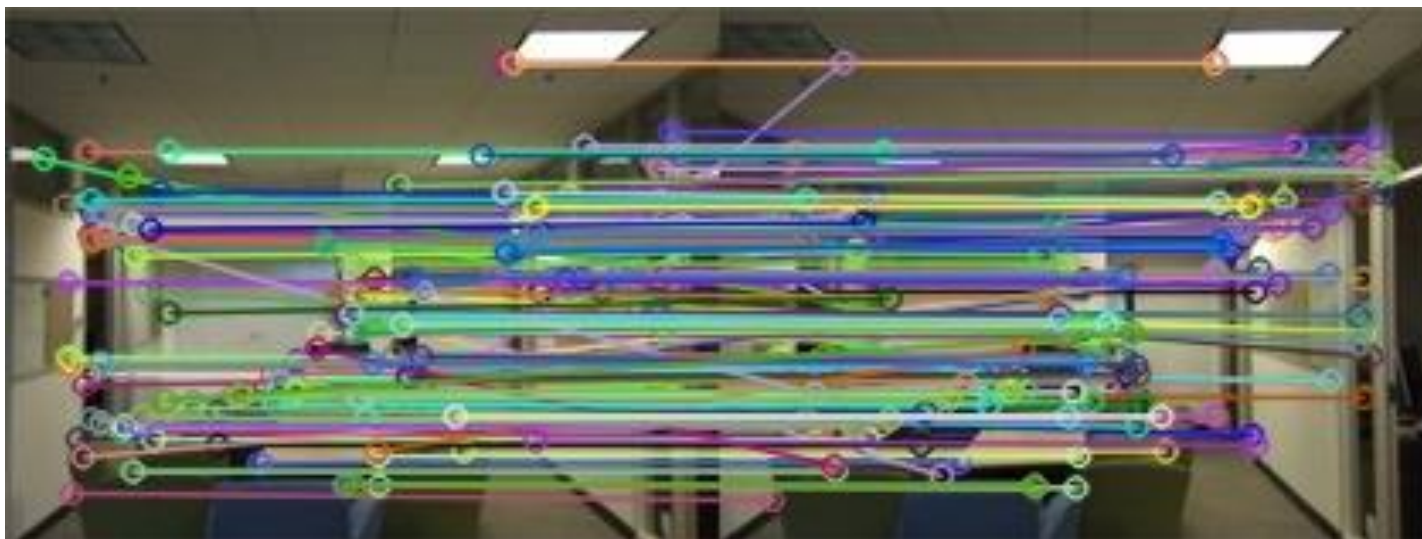
- پارامترهای مدل (x_t, y_t) بر اساس نقاط متناظر محاسبه می‌شوند
- نیازمند تنها ۱ نقطه است!
- به دلیل وجود خطا در مکان‌یابی دقیق نقاط کلیدی، می‌توان با استفاده از تعداد بیشتری نقطه به پارامترهای دقیق‌تری دست یافت



$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = T \left(\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \right)$$

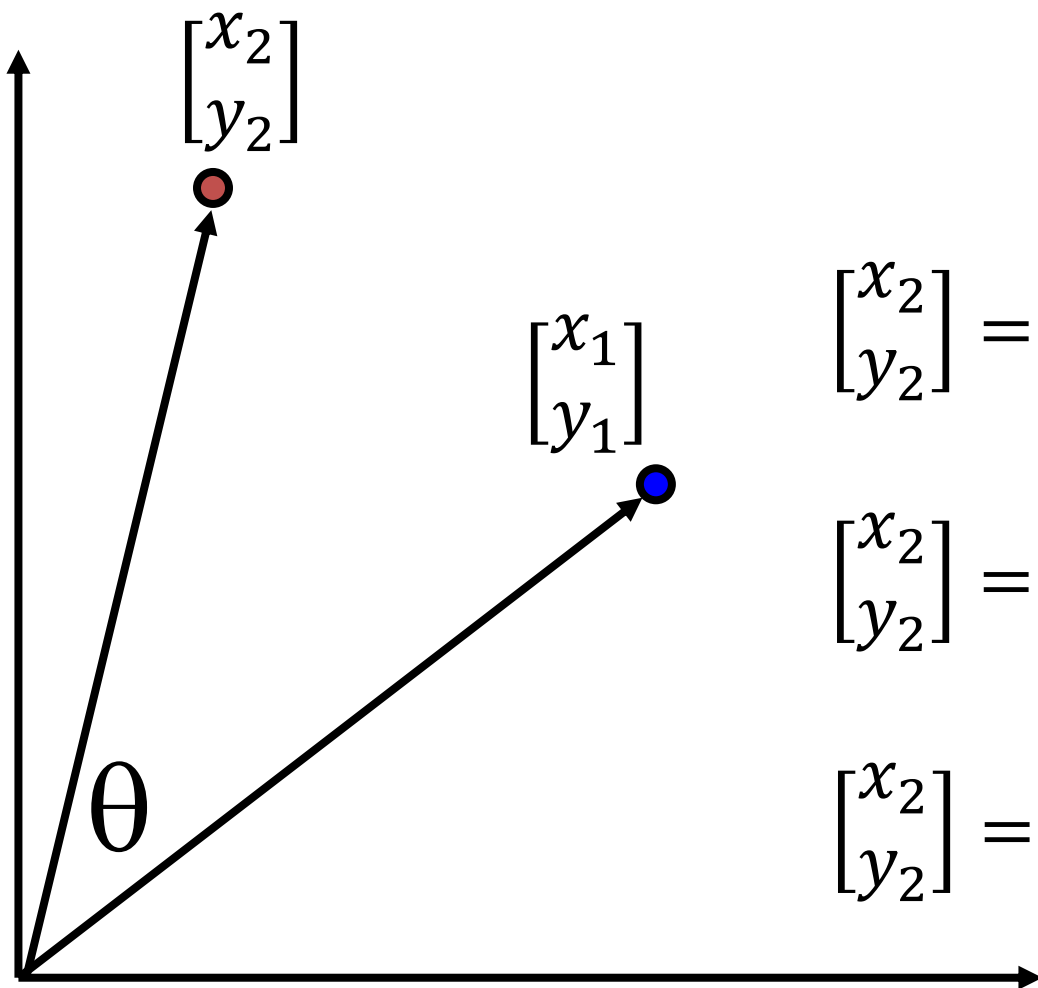
داده‌های پرت

- روش حداقل مربعات خطا حساس به داده‌های پرت است
- روش RANSAC برای بدست آوردن تابع تبدیل مقاوم نسبت به داده‌های پرت استفاده می‌شود



تبدیل Rigid

• این تبدیل تنها شامل چرخش و انتقال است



$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 \cos \theta - y_1 \sin \theta \\ x_1 \sin \theta + y_1 \cos \theta \end{bmatrix}$$

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

تبدیل Rigid

- این تبدیل تنها شامل چرخش و انتقال است

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \underbrace{\begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}}_R \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \underbrace{\begin{bmatrix} t_x \\ t_y \end{bmatrix}}_T$$

- تبدیل Rigid تنها ۳ پارامتر دارد که توسط ۲ نقطه قابل محاسبه هستند
- البته باید خطای اندازه‌گیری و داده‌های پرت را لحاظ کرد

تبدیل شباهت

- این تبدیل شامل چرخش، انتقال و مقیاس است

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = a \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

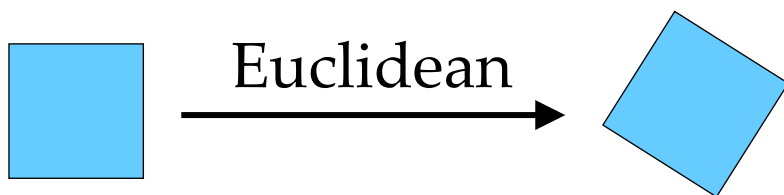
$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} a \cos \theta & -a \sin \theta & t_x \\ a \sin \theta & a \cos \theta & t_y \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

- ۴ درجه آزادی و حداقل ۲ نقطه!

تبدیل Affine

- این تبدیل شامل چرخش، انتقال، مقیاس و کجی است

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

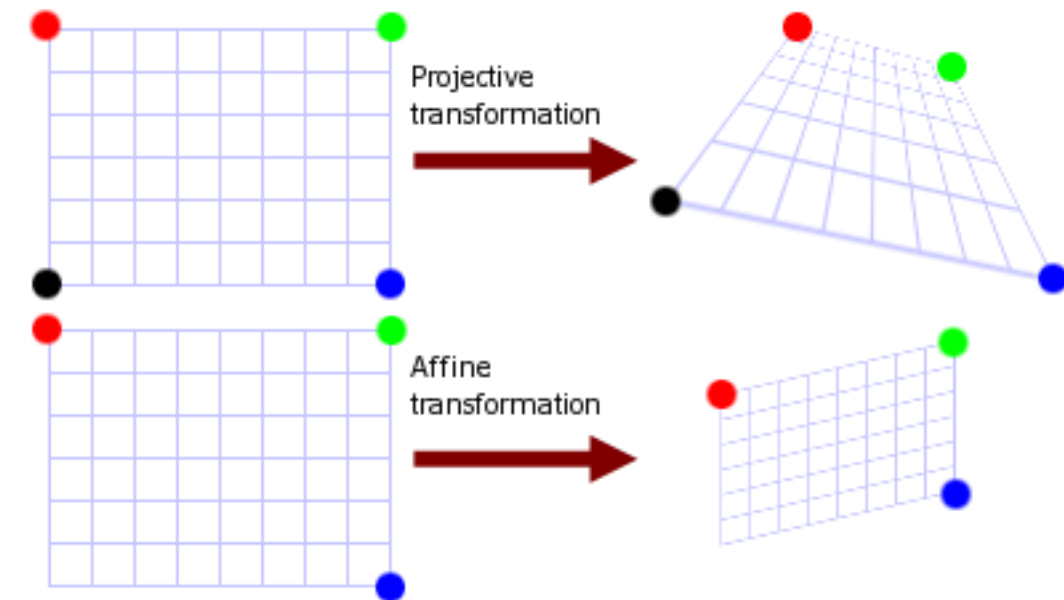


- ۶ درجه آزادی و حداقل ۳ نقطه!
- خط به خط نگاشت می شود
- خطوط موازی، موازی باقی می ماند
- نسبت ها روی یک خط حفظ می شود

تبدیل تصویری

- تبدیل‌های قبل نمی‌توانند تغییر عمق پیکسل‌ها را مدل کنند

$$s_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = H \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$



$$x_2 = \frac{h_{11}x_1 + h_{12}y_1 + h_{13}}{h_{31}x_1 + h_{32}y_1 + h_{33}}$$

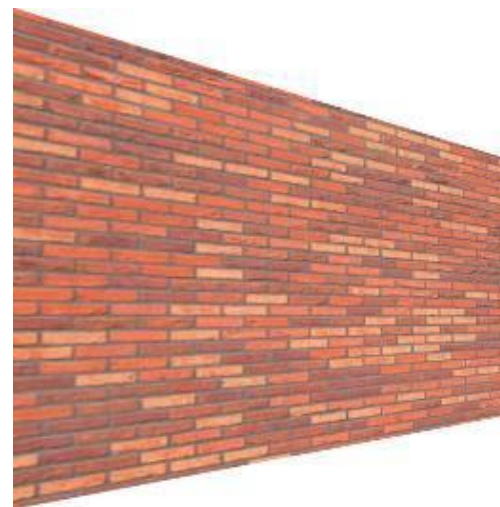
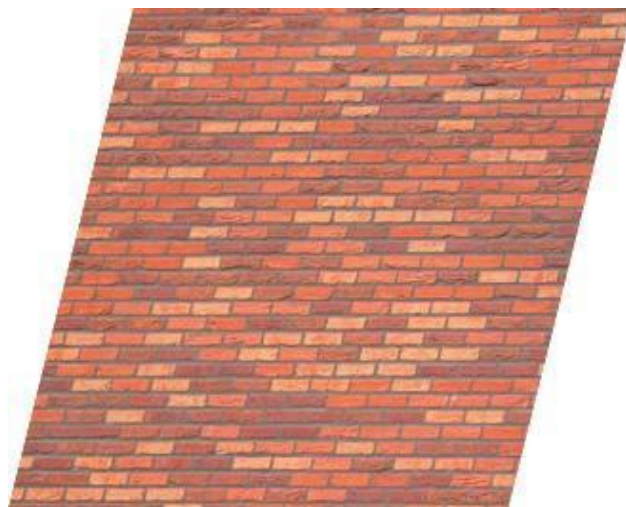
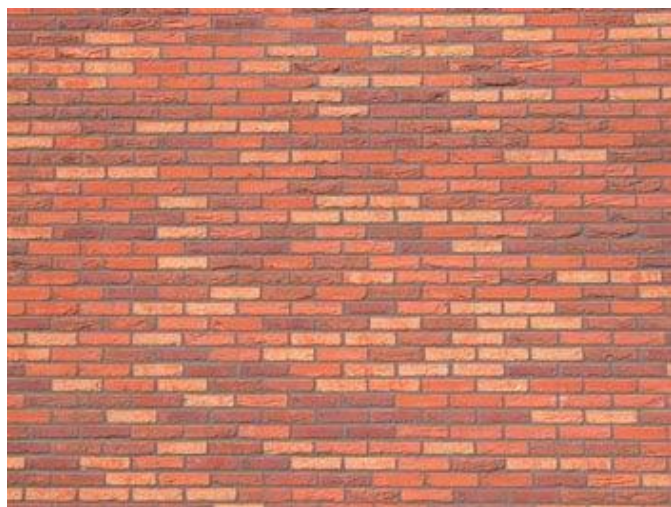
$$y_2 = \frac{h_{21}x_1 + h_{22}y_1 + h_{23}}{h_{31}x_1 + h_{32}y_1 + h_{33}}$$

تبدیل تصویری

• تبدیل تصویری، تبدیل Affine ای است که نسبت به موقعیت پیکسل در ضریب متفاوتی ضرب می شود

$$s_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

- خط به خط نگاشت می شود
- خطوط موازی لزوما موازی نمی مانند
- نسبت ها لزوما حفظ نمی شود
- ۸ درجه آزادی دارد و حداقل به ۴ نقطه نیاز دارد



توابع OpenCV

```
mat = cv2.getPerspectiveTransform(src_points, dst_points[, solveMethod])
```

```
// src_points:    Coordinates of quadrangle vertices in the source image
// dst_points:    Coordinates of the corresponding quadrangle vertices in the destination image
// mat:           Perspective transform from four pairs of the corresponding points
```

```
dst_points = cv2.perspectiveTransform(src_points, mat)
```

```
// src_points:    Input two-channel or three-channel floating-point array; each element is a 2D/3D vector to be transformed
// mat:           3x3 or 4x4 floating-point transformation matrix
// dst_points:    Output array of the same size and type as src
```

$$\text{dst}(x, y) = \text{src} \left(\frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}, \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \right)$$

```
dst = cv2.warpPerspective(src, mat, dsize[, flags[, borderMode[, borderValue]]])
```

```
// src_points:    Input image
// mat:           3x3 floating-point transformation matrix
// dsize:         Size of the output image
// flags:         Combination of interpolation methods and the optional flag WARP_INVERSE_MAP
// borderMode:    Pixel extrapolation method
// borderValue:   value used in case of a constant border; by default, it equals 0
// dst:           Output image that has the size dsize and the same type as src .
```

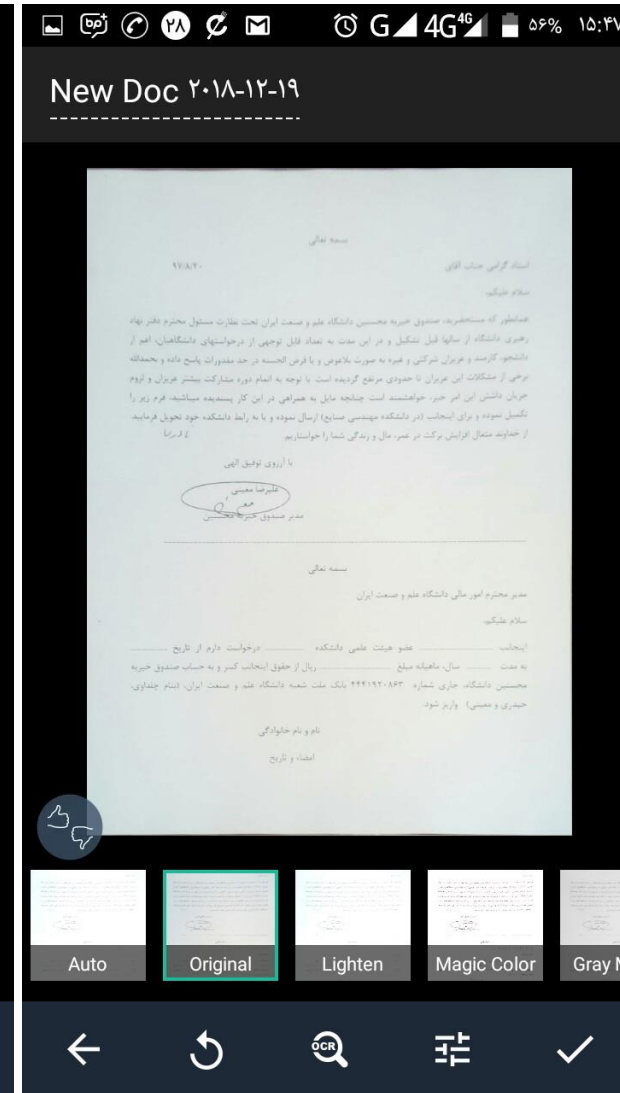
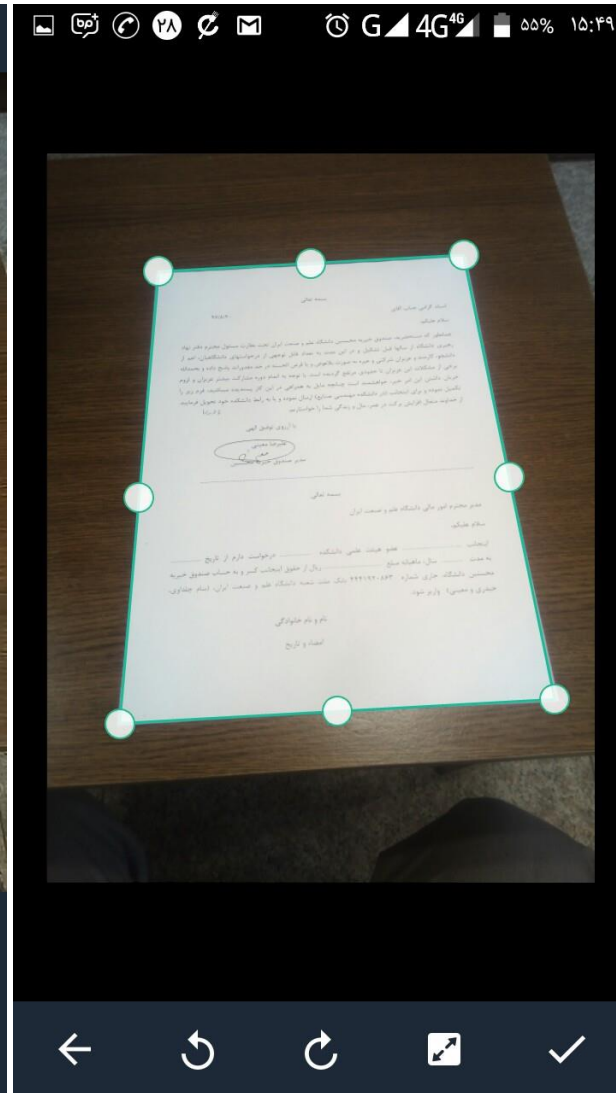
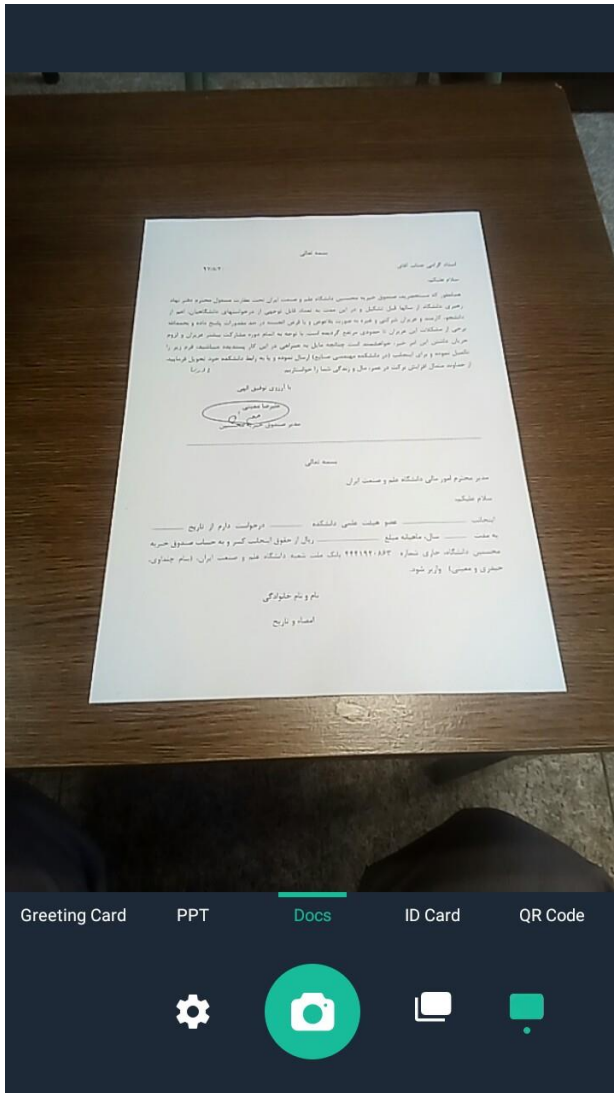
توابع OpenCV

```
mat, mask = cv2.findHomography(src_points, dst_points[, method[, ransacReprojThreshold[, maxIters[, confidence]]]])
```

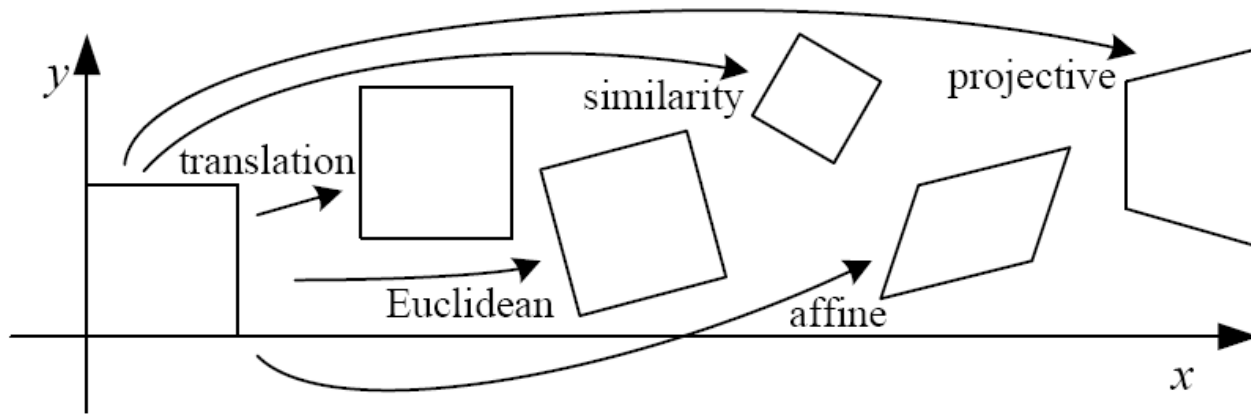
```
// src_points:      Coordinates of the points in the original plane
// dst_points:      Coordinates of the points in the target plane
// method:          Method used to compute a homography matrix (least squares, RANSAC, LMEDS, RHO)
// ransacReprojThreshold: Maximum allowed reprojection error to treat a point pair as an inlier
// maxIters:         The maximum number of RANSAC iterations
// confidence:       Confidence level, between 0 and 1
// mask:            Optional output mask set by a robust method (RANSAC or LMEDS)
// mat:             Estimated perspective transform between two planes
```






Function	Use
<code>cv::transform()</code>	Affine transform a list of points
<code>cv::warpAffine()</code>	Affine transform a whole image
<code>cv::getAffineTransform()</code>	Calculate affine matrix from points
<code>cv::getRotationMatrix2D()</code>	Calculate affine matrix to achieve rotation
<code>cv::perspectiveTransform()</code>	Perspective transform a list of points
<code>cv::warpPerspective()</code>	Perspective transform a whole image
<code>cv::getPerspectiveTransform()</code>	Fill in perspective transform matrix parameters

CamScanner



تبدیل‌های هندسی

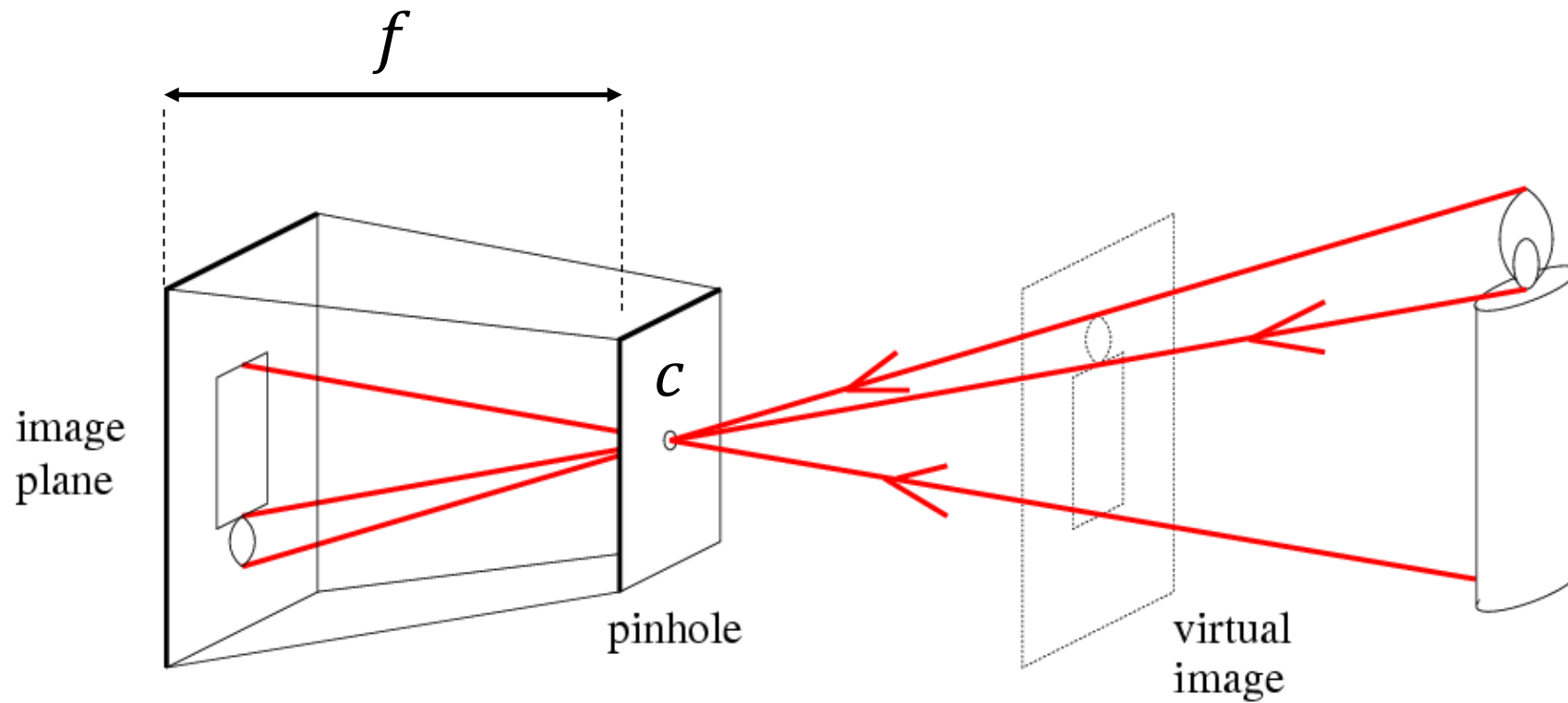


Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} I & t \end{bmatrix}_{2 \times 3}$	2	orientation + ...	
rigid (Euclidean)	$\begin{bmatrix} R & t \end{bmatrix}_{2 \times 3}$	3	lengths + ...	
similarity	$\begin{bmatrix} sR & t \end{bmatrix}_{2 \times 3}$	4	angles + ...	
affine	$\begin{bmatrix} A \end{bmatrix}_{2 \times 3}$	6	parallelism + ...	
projective	$\begin{bmatrix} \tilde{H} \end{bmatrix}_{3 \times 3}$	8	straight lines	

بینایی سه بعدی

Correspondence and Image Alignment

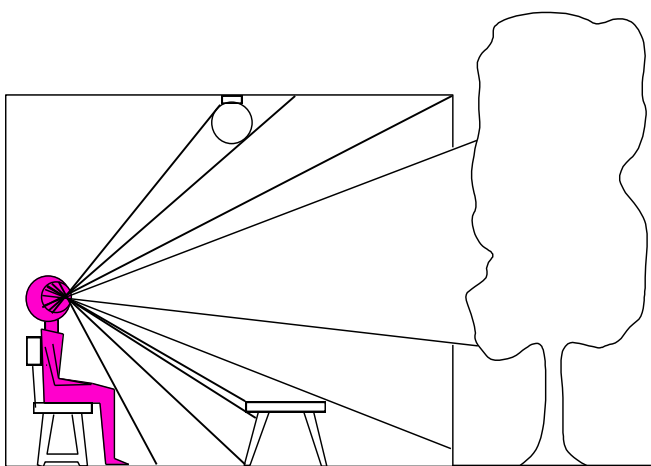
مدل دوربین Pinhole



تصویر ۲ بعدی

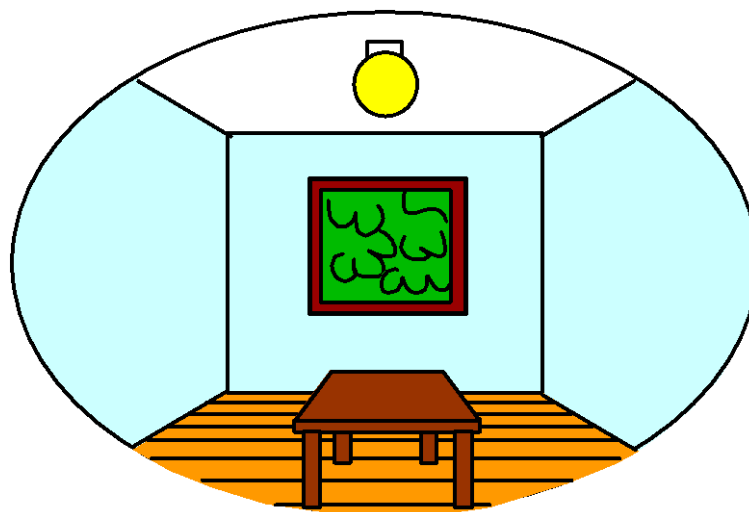
- توسط دوربین تصویربرداری، دنیای ۳ بعدی به ۲ بعد تصویر می‌شود

3D world



Point of observation

2D image



اثرات تصویربرداری

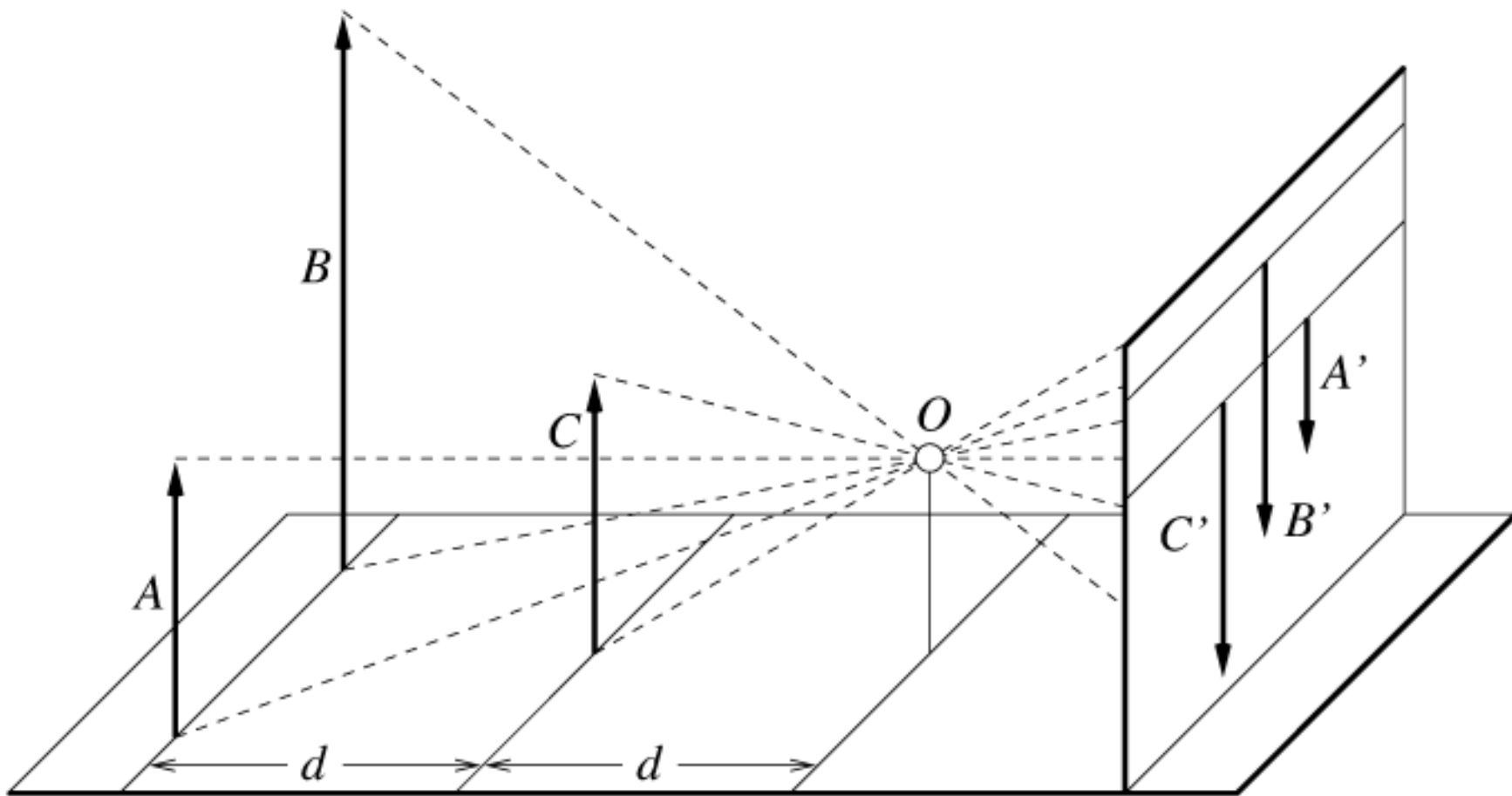


Forced Perspective Photography



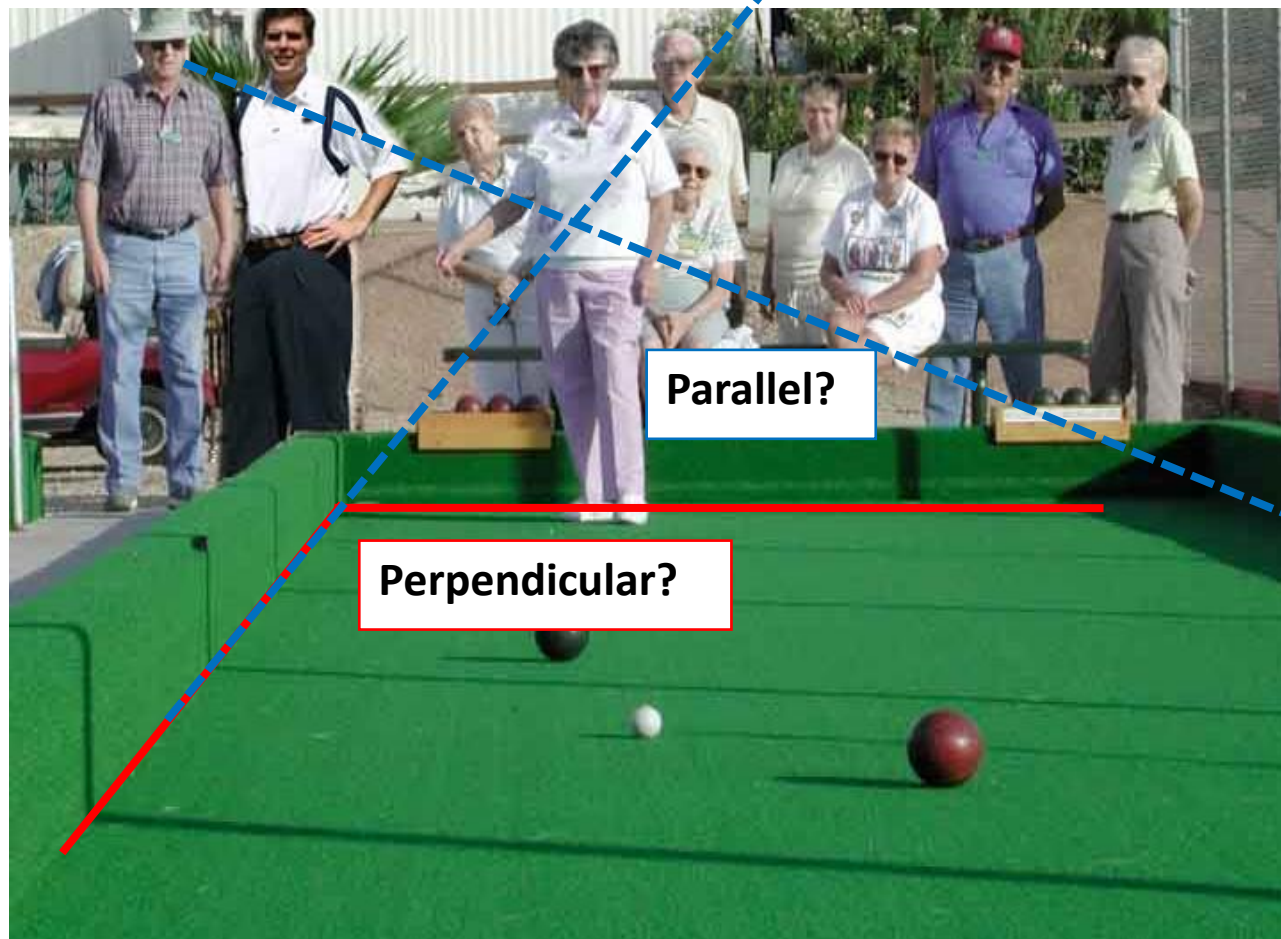
اثرات تصویربرداری

- طول اجسام حفظ نمی شود



اثرات تصویربرداری

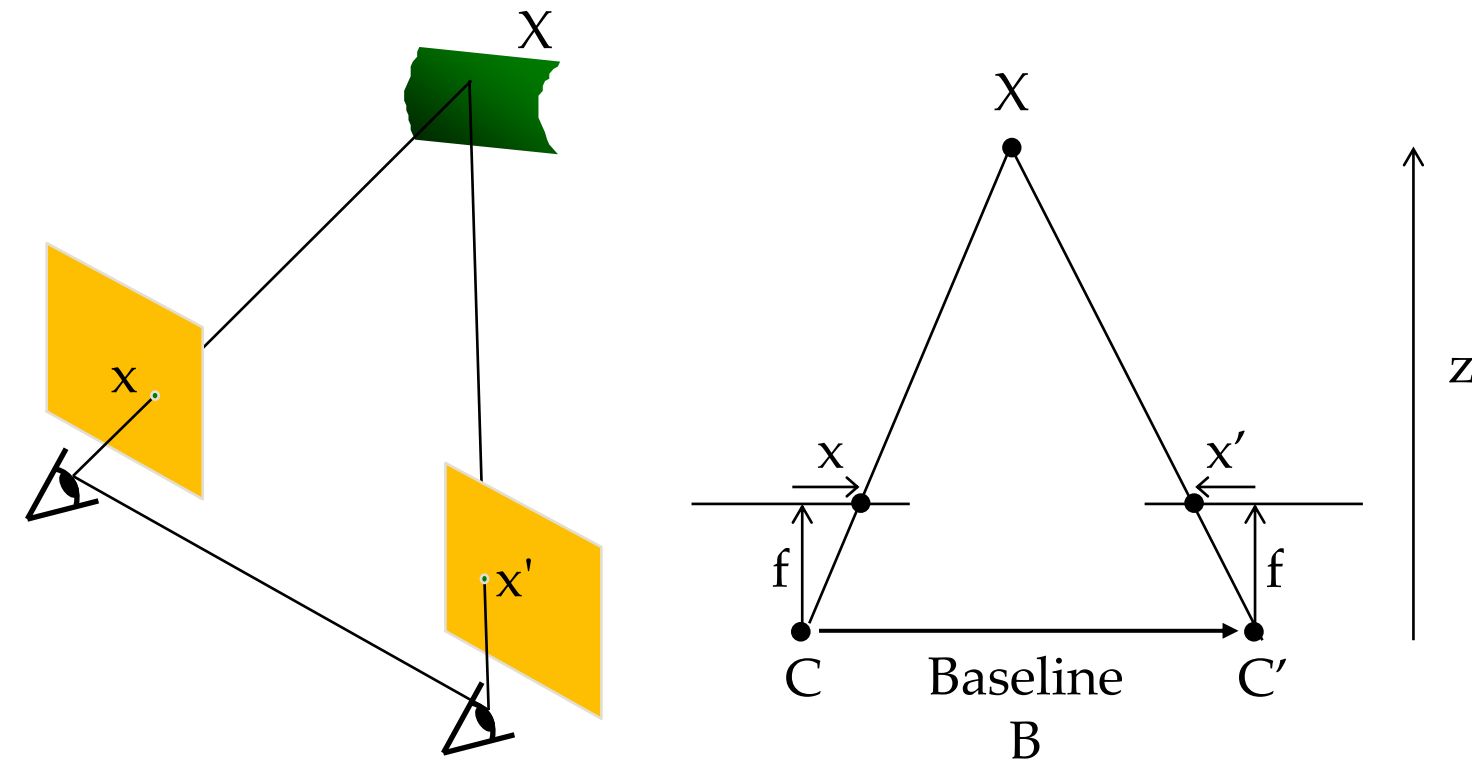
- زاویه اجسام حفظ نمی شود
- خط در تصویر نیز خط دیده می شود



تخمین عمق

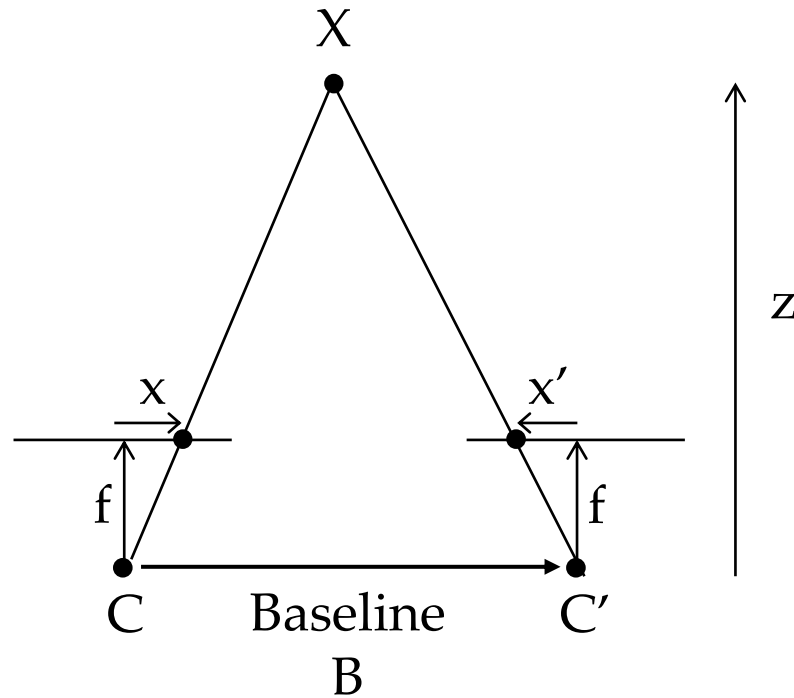
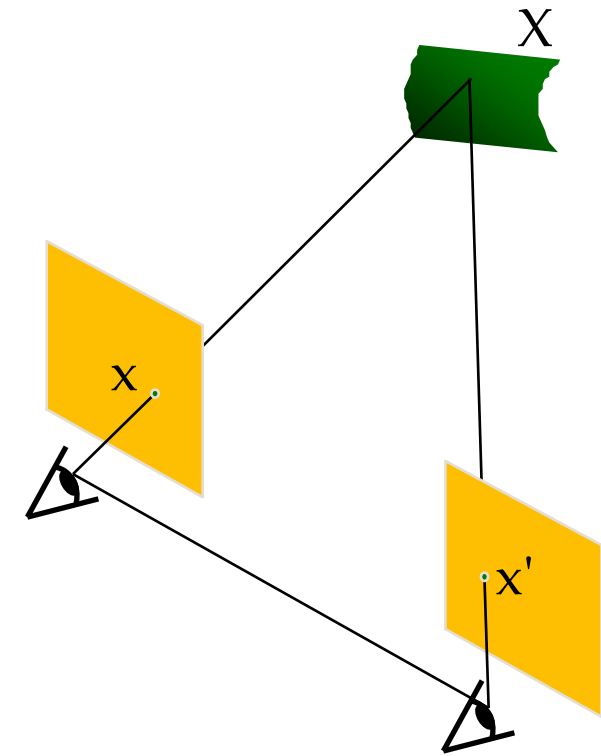
- ابزارهای مختلفی برای تخمین عمق پیکسل‌ها وجود دارد که یکی از معروفترین روش‌ها استفاده از دو دوربین است (stereo vision)

- باید موقعیت یک نقطه در دو تصویر استخراج شود و بر اساس آن، عمق مربوطه تخمین زده شود



تخمین عمق

- تناظر: چگونه می‌توان نقاط متناظر را محاسبه کرد؟
- کالیبراسیون: ارتباط میان دو دوربین چگونه قابل محاسبه است؟

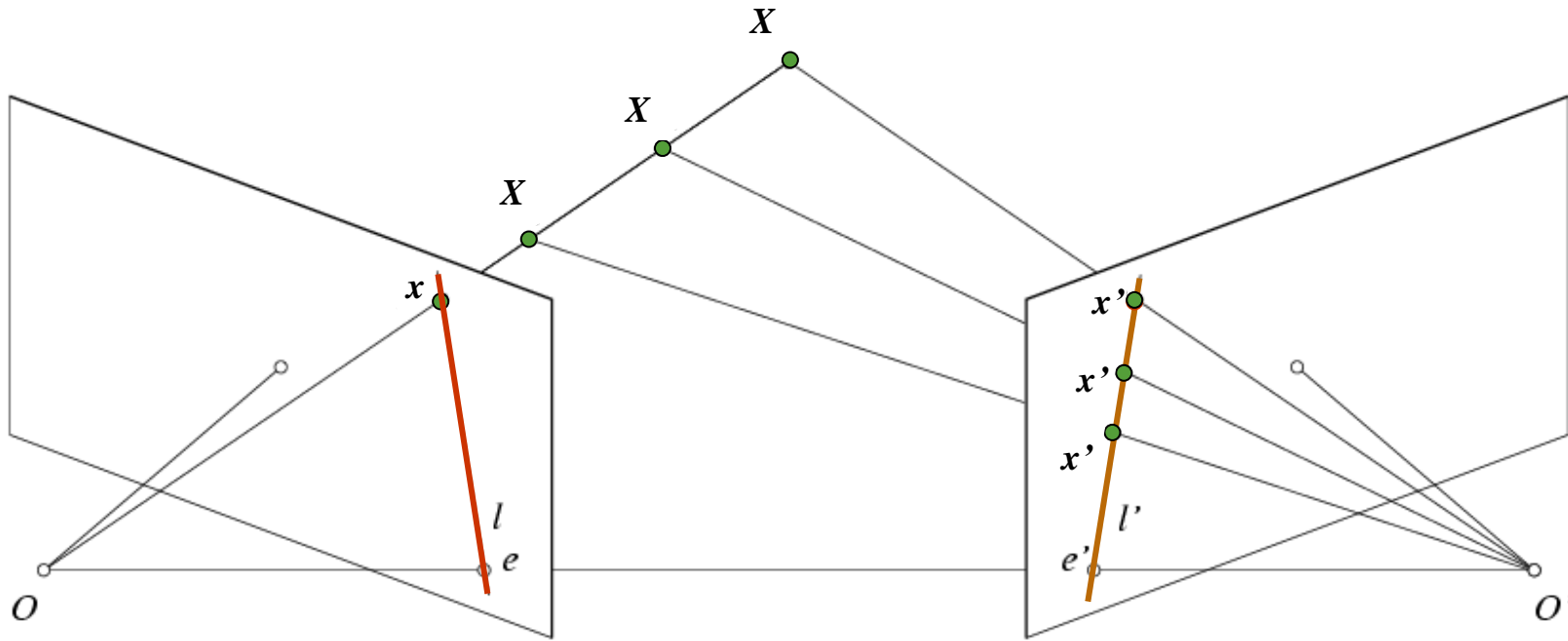


تناظریابی

- پیدا کردن نقاط متناظر عملیات ساده‌ای نیست به خصوص از آنجائیکه باید برای تمام نقاط انجام شود
- مطلوب است تا ناحیه جستجو در تصویر دوم محدود شود



هندسه Epipolar



- نقطه x در تصویر اول باید در تصویر دوم روی خط l' مشاهده شود