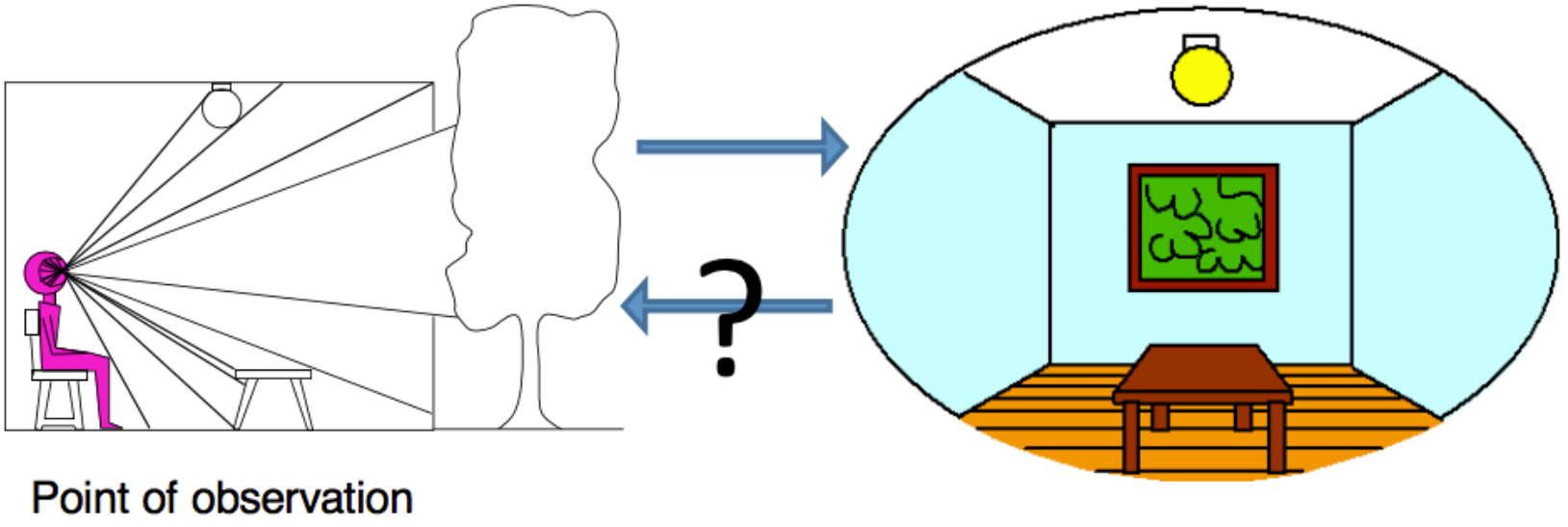
Visão Stereo

Prof. Dr. Geraldo Braz Junior

Slides baseados em nas notas de aula de Fei-Fei li

3D de imagens?

- Seria possível determinar profundidade a partir de imagens?

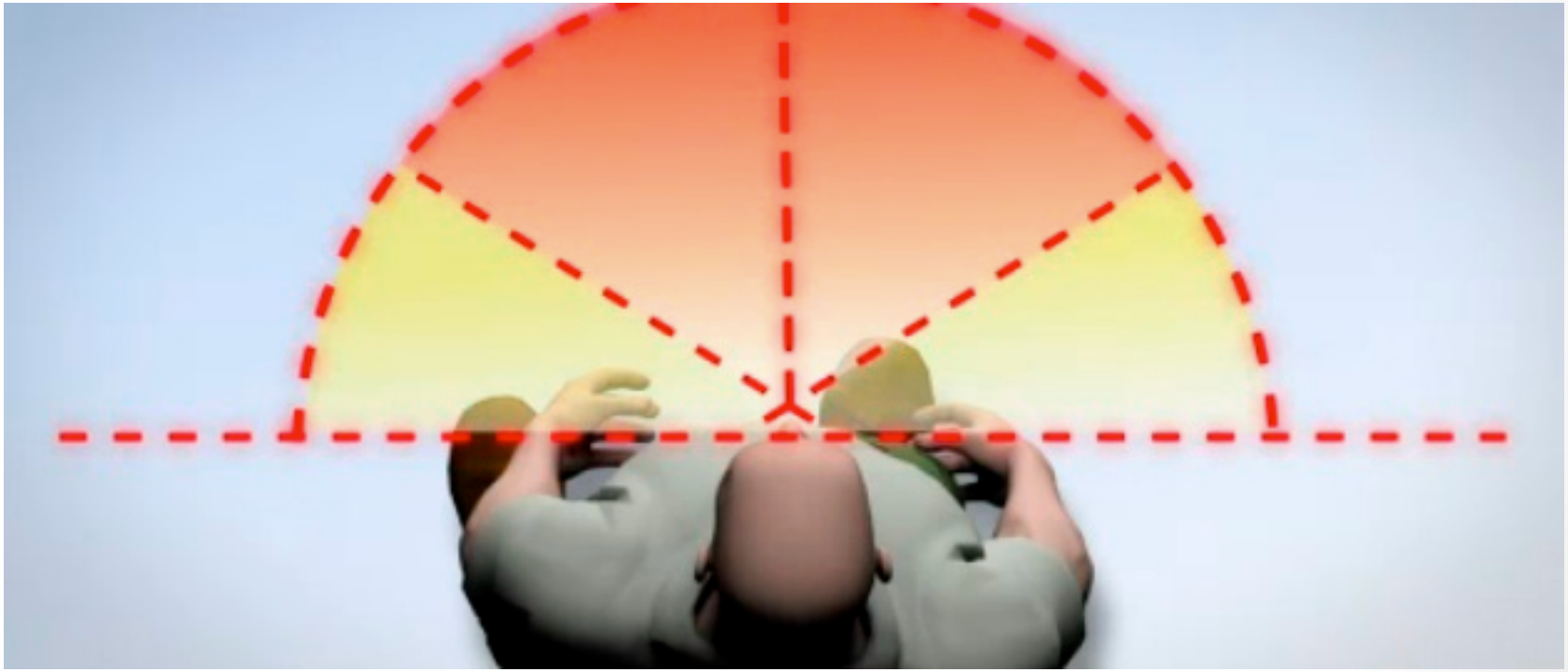


O que nos dá percepção de profundidade?

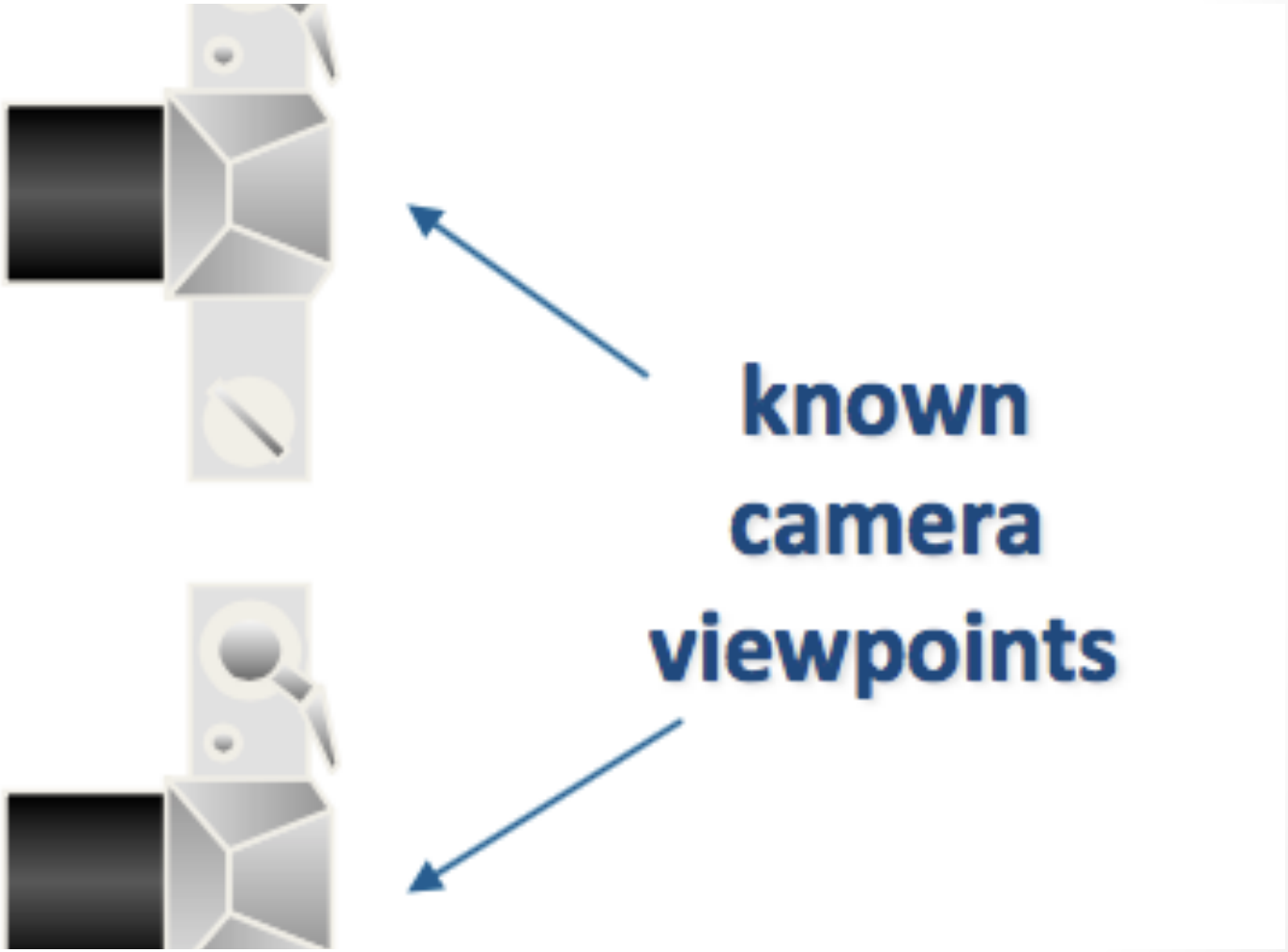
- Sombra
- Textura
- Foco
- Movimento
- Iluminação
- Silhuetas
- Luz
- Simetria

Como o nosso sistema visual funciona?

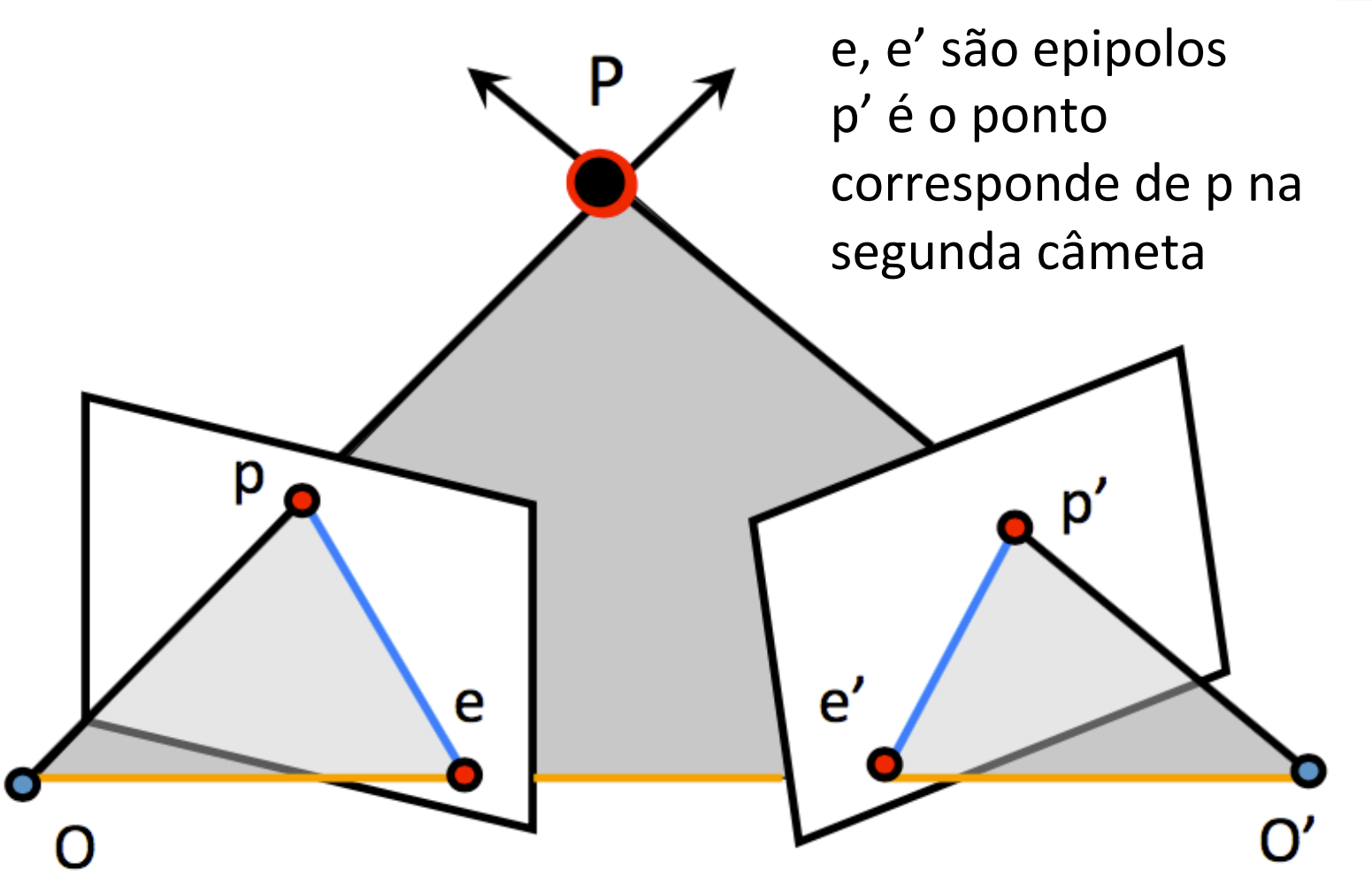
- Dois olhos é melhor do que um



2 câmeras conhecidas

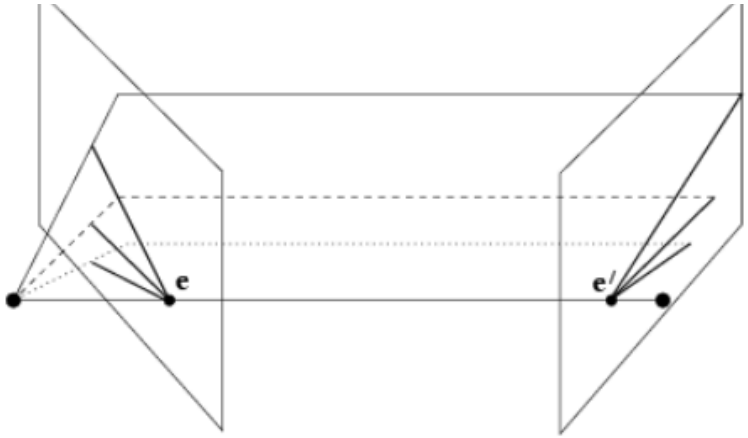


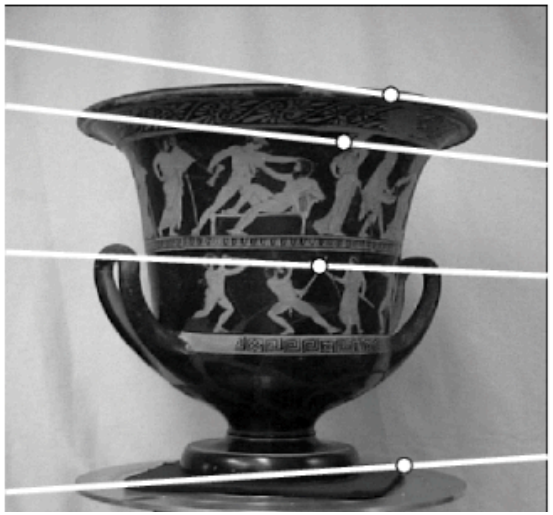
Geometria Epipolar

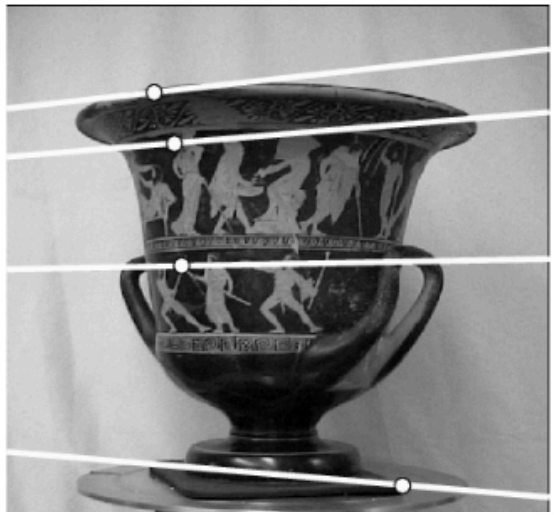


e, e’ são epipolos
p’ é o ponto corresponde de p na segunda câmeta

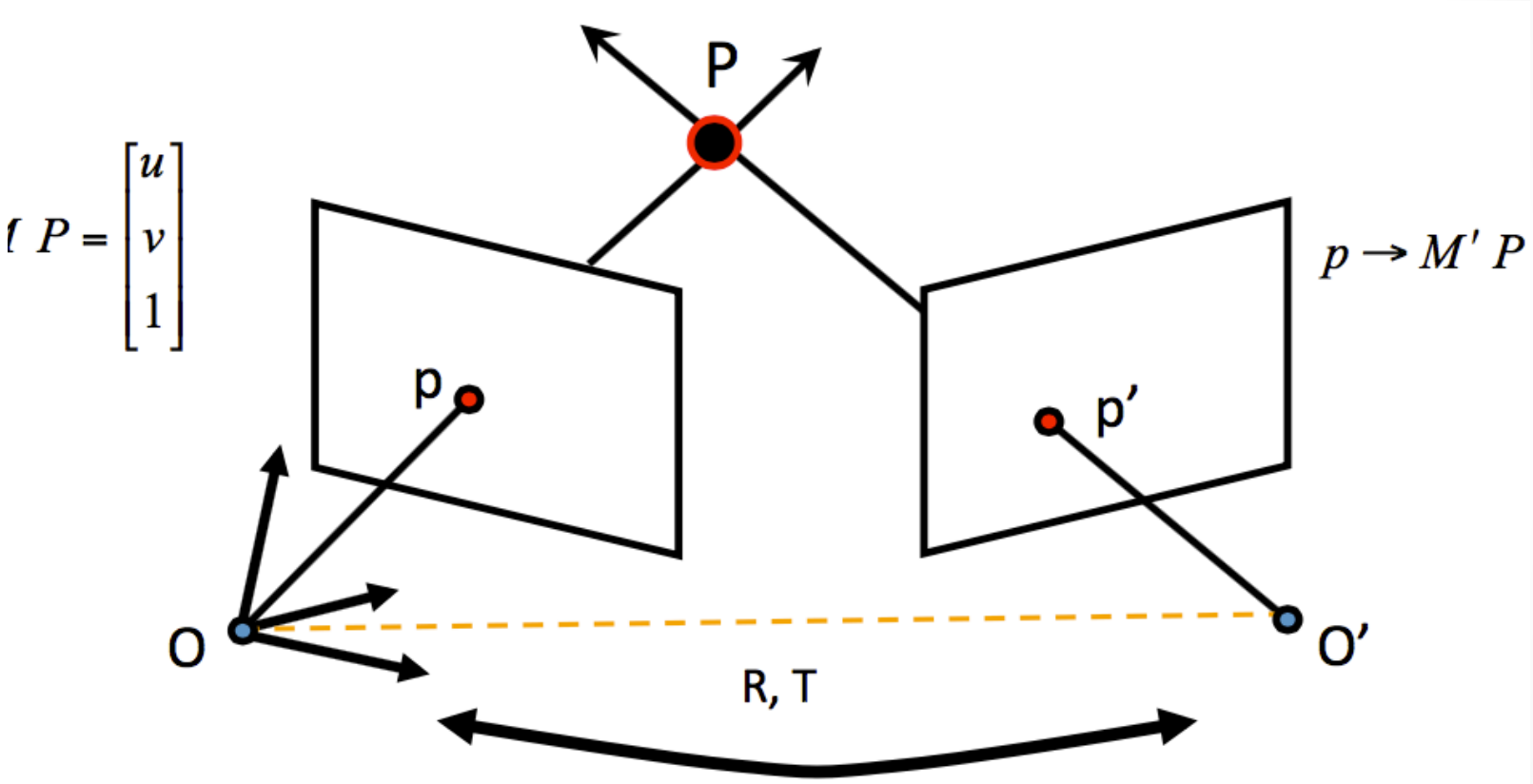
Feixe Epipolar







Restrição Epipolar



$$\ell P = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

$$p \rightarrow M' P$$

$$= K \begin{bmatrix} I & 0 \end{bmatrix}$$

$$M' = K' \begin{bmatrix} R & T \end{bmatrix}$$

Restrição Epipolar

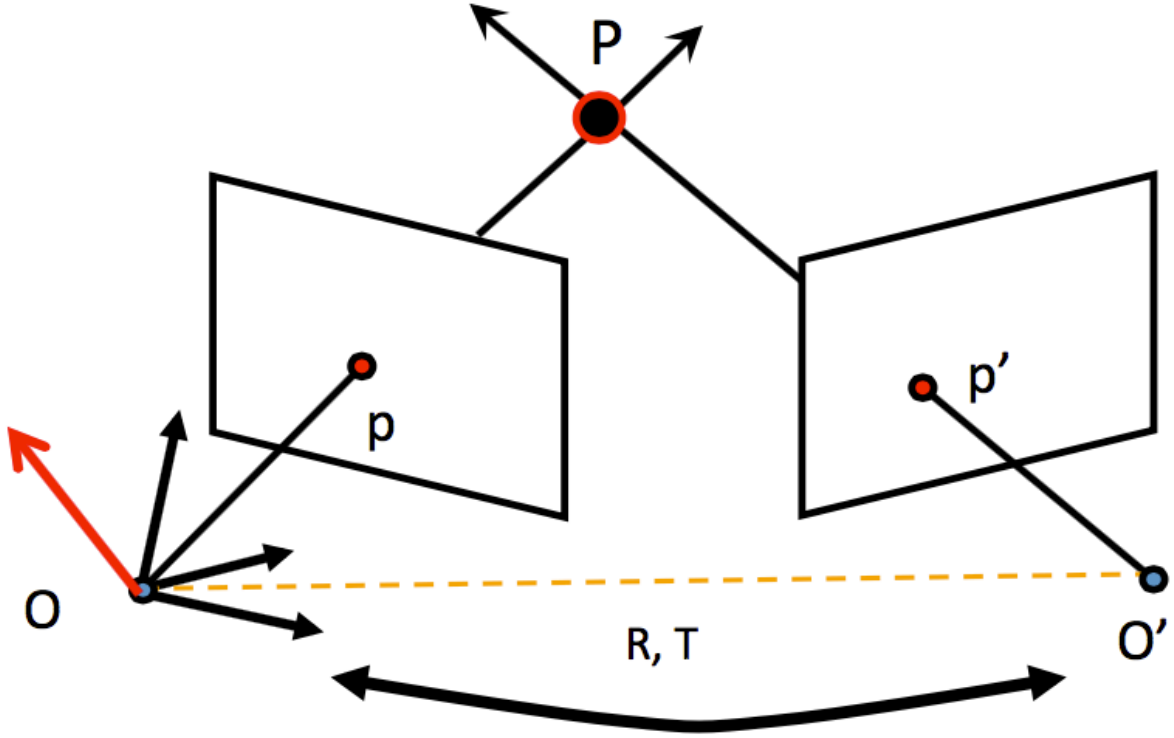
- Se K e K’ são conhecidas durante a calibração de cameras, então:

$$M = K\begin{bmatrix} I & 0 \end{bmatrix} \qquad M' = K'\begin{bmatrix} R & T \end{bmatrix}$$

K and K’ are known (calibrated cameras)

$$M = \begin{bmatrix} I & 0 \end{bmatrix} \qquad M' = \begin{bmatrix} R & T \end{bmatrix}$$

Restrição Epipolar

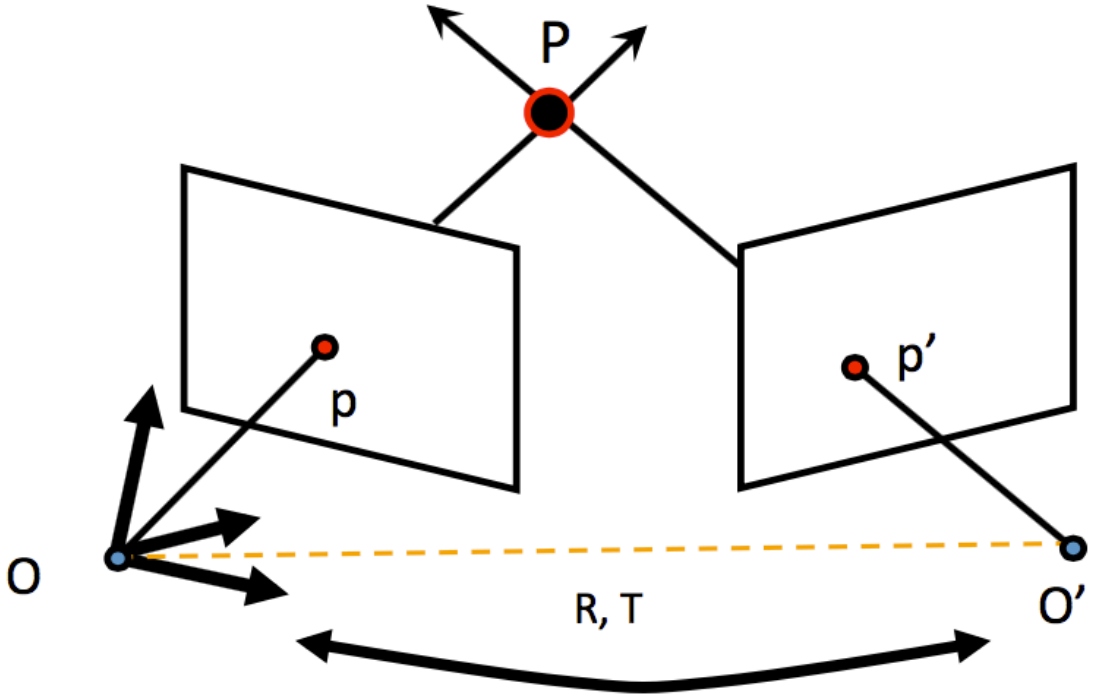


$$T \times (R\, p')$$

Perpendicular to epipolar plane

$$p^{T} \cdot \left[T \times (R\, p') \right] = 0$$

Restrição Epipolar

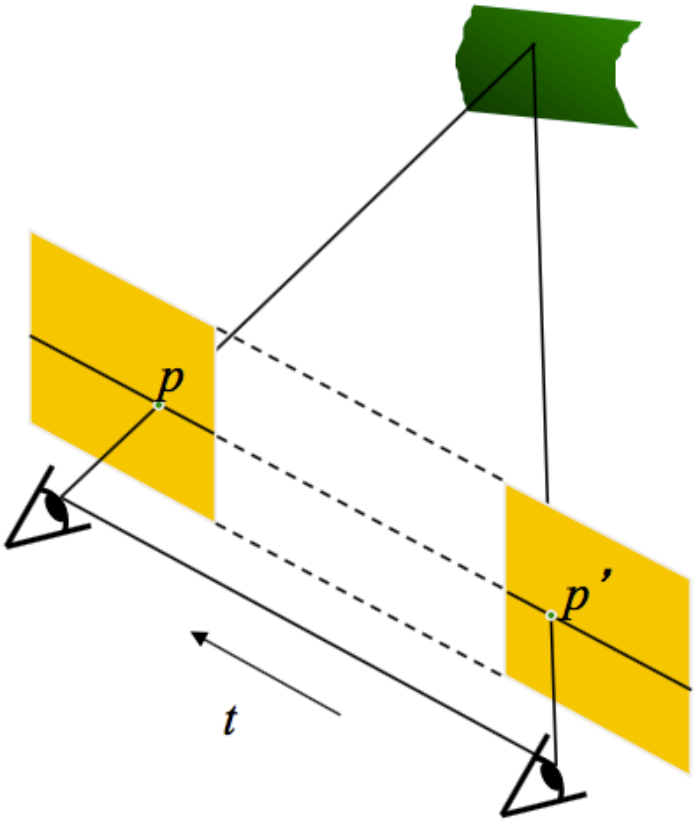


Produto vetorial como um produto escalar:
skew symmetric matrix

$$p^T \cdot [T \times (R\,p')] = 0 \rightarrow p^T \cdot [T_x] \cdot R\,p' = 0$$

(Longuet-Higgins, 1981) E = essential matrix

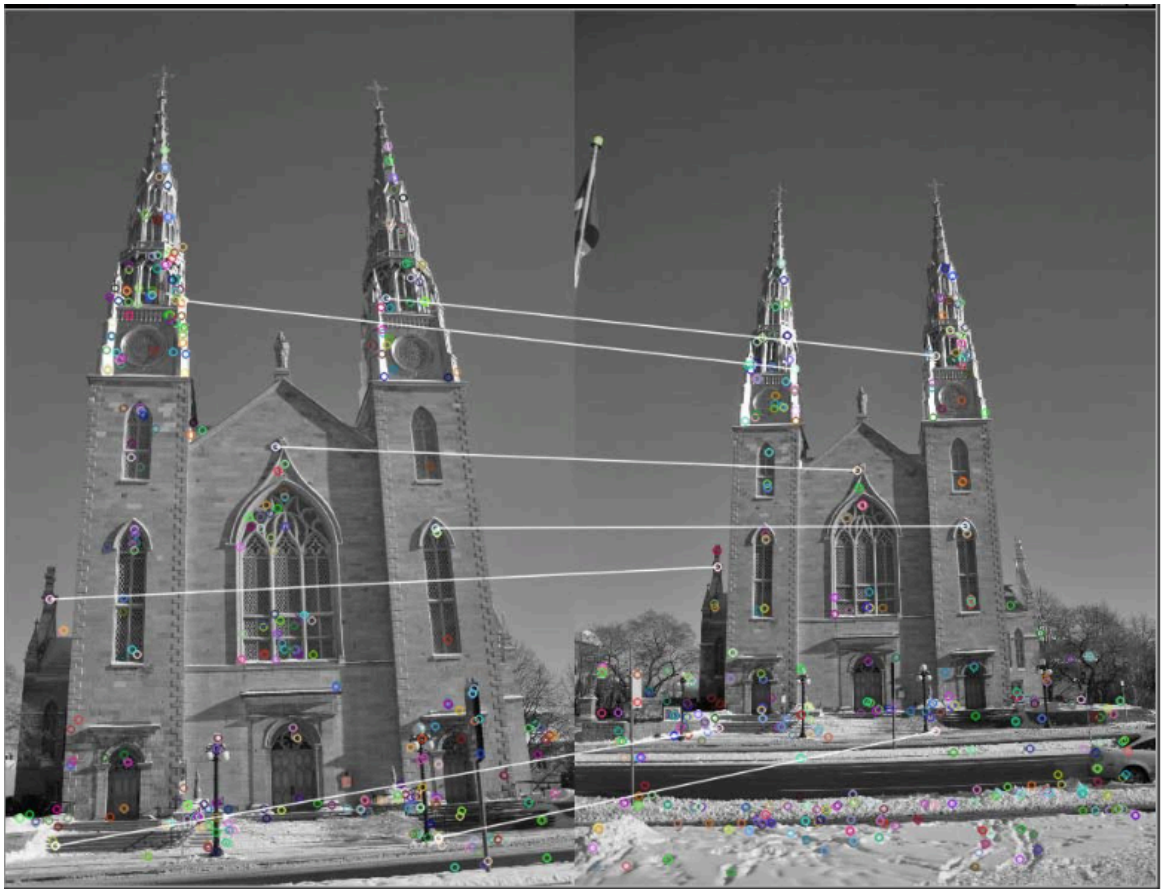
Casos Especiais: Câmeras Paralelas



$$E = [t_\times]R = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix}$$

Como descubrir a matriz fundamental E?

- São necessários pelo menos 7 pontos de matching



O processo

- Envolve encontrar bons pontos de matching (do contrário a matriz não fica razoável)
- Se houver pontos conhecidos entre as duas imagens (usando marcadores por exemplo): pode-se usar estes pontos

- Caso não exista, uma abordagem é usar detectores de cantos

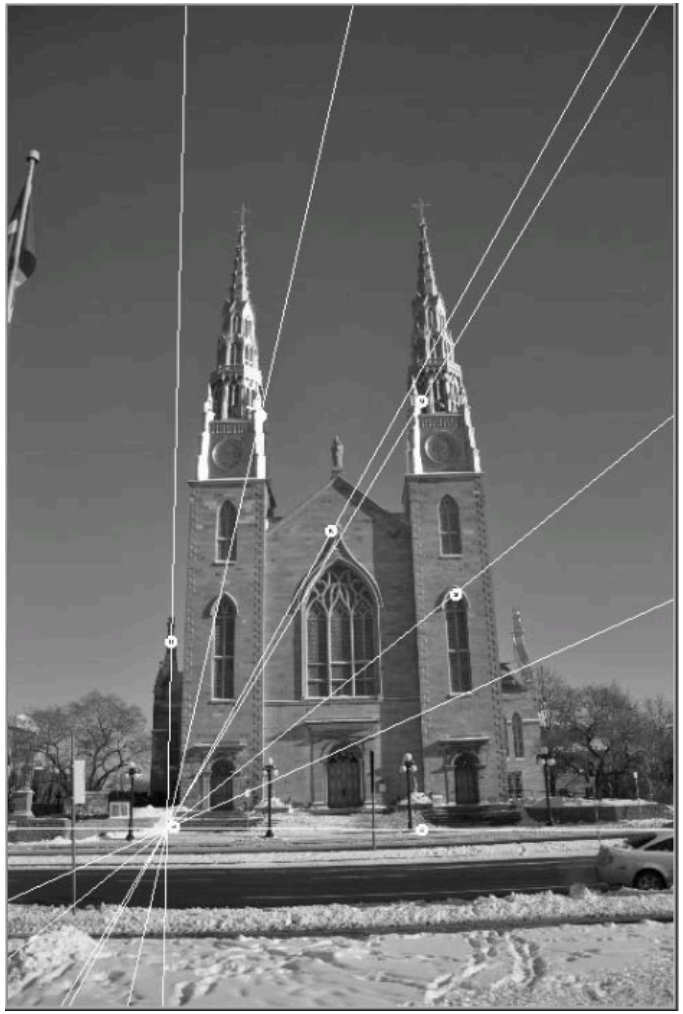
API Opencv

```
Mat fundemental =
        findFundamentalMat(
           Mat(selPoints1), // pontos da primeira imagem
           Mat(selPoints2), // pontos da segunda imagem
           CV_FM_7POINT);  // como desenha
estimar
```

*neste caso, devem ser ÓTIMOS 7 pontos

* tem que ser com marcador

opencv: computeCorrespondEpilines

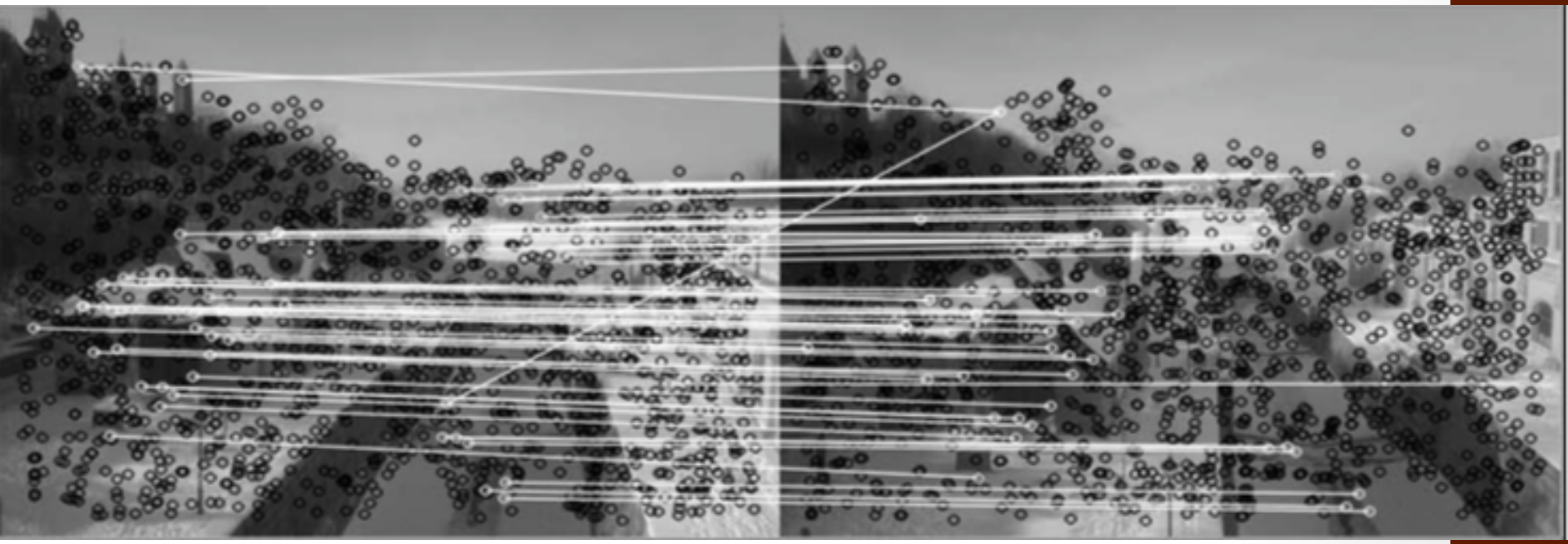


Se você não têm os 7 pontos mágicos...

- Escreva um matcher
  - Pode usar SIFT, SURF, ... qualquer um
- Faça o match, mas filtre os melhores
  - Use match com knn por exemplo

Sugestões de filtro: grau de semelhança

- Somente matches que tenham pelo menos 95% de confidência (distância entre as duas escolhas do KNN)



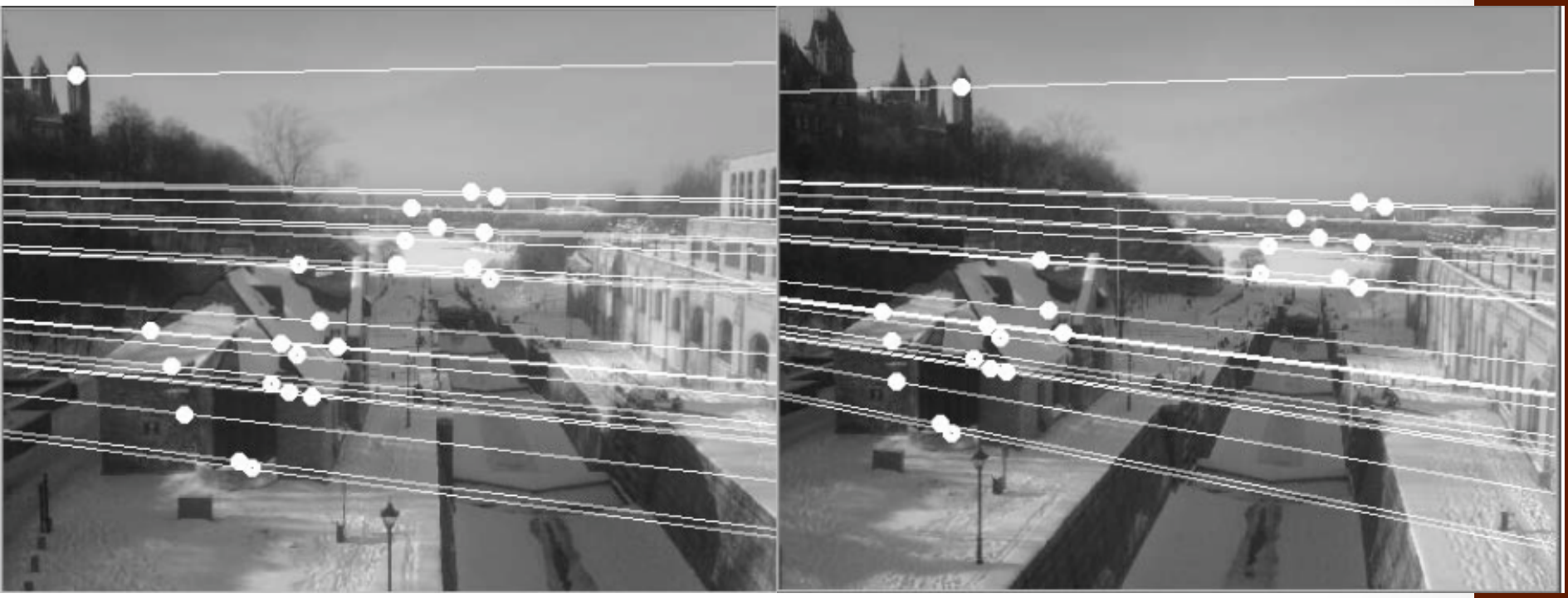
Sugestões de filtro: simetria e RANSAC

- Escolha matches que aconteçam ao mesmo tempo nas duas imagens

- Depois aplique **RANSAC** para estimação da matriz fundamental

```
Mat fundamental=
findFundamentalMat(Mat(points1), Mat(points2),
        inliers, //indica quais pontos devem ficar
        CV_FM_RANSAC, // RANSAC
        3, 0.99);  //distância no RANSAC e probabilidade
```

No fim



Vide código exemplo

E quando não existe translação?

- A matriz fundamental é simplificada e fica somente com os parâmetros de rotação

- Chamada de **Homografia**

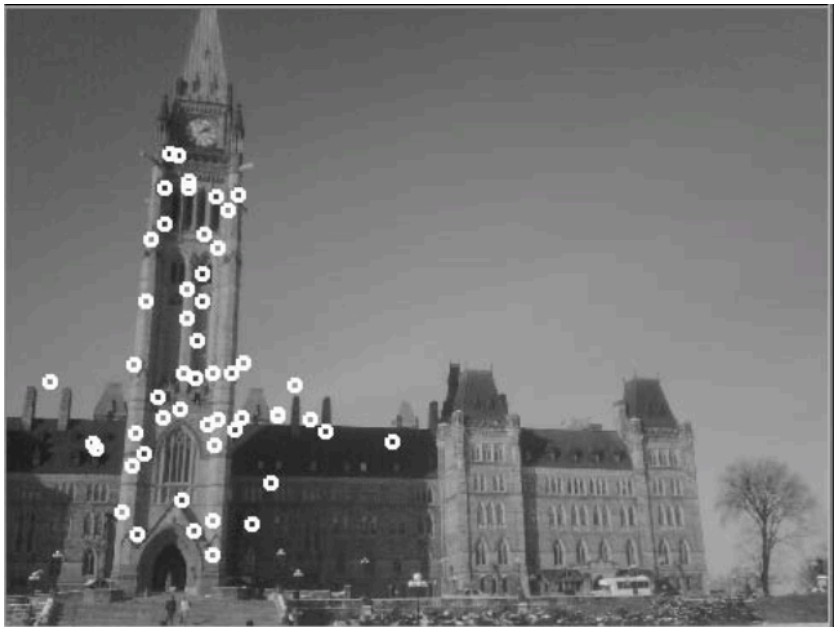
$$\begin{bmatrix} sx' \\ sy' \\ s \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

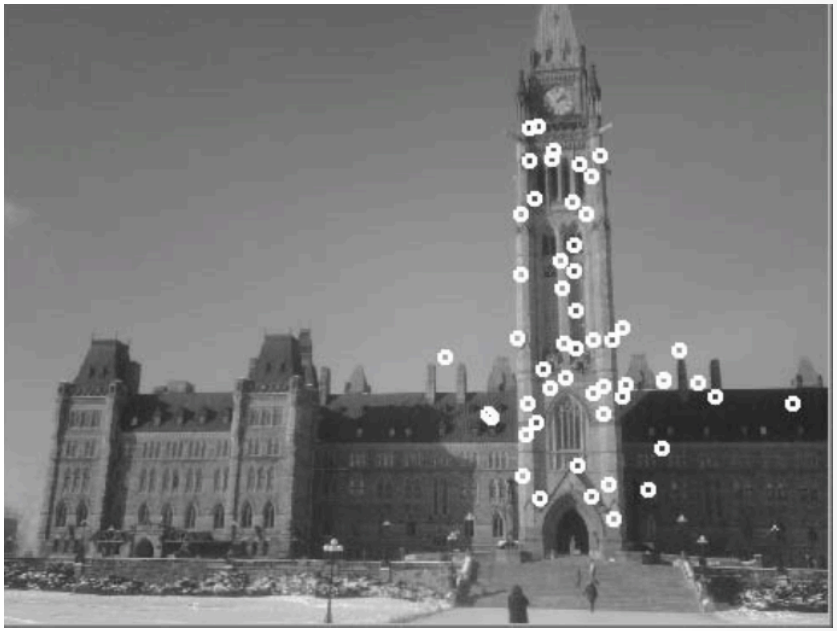
Funções bem semelhantes

```
Mat homography= findHomography(
        Mat(points1), //pontos da imagem 1
        Mat(points2), // pontos da imagem 2
        inliers,
        CV_RANSAC,
        1.0);
```

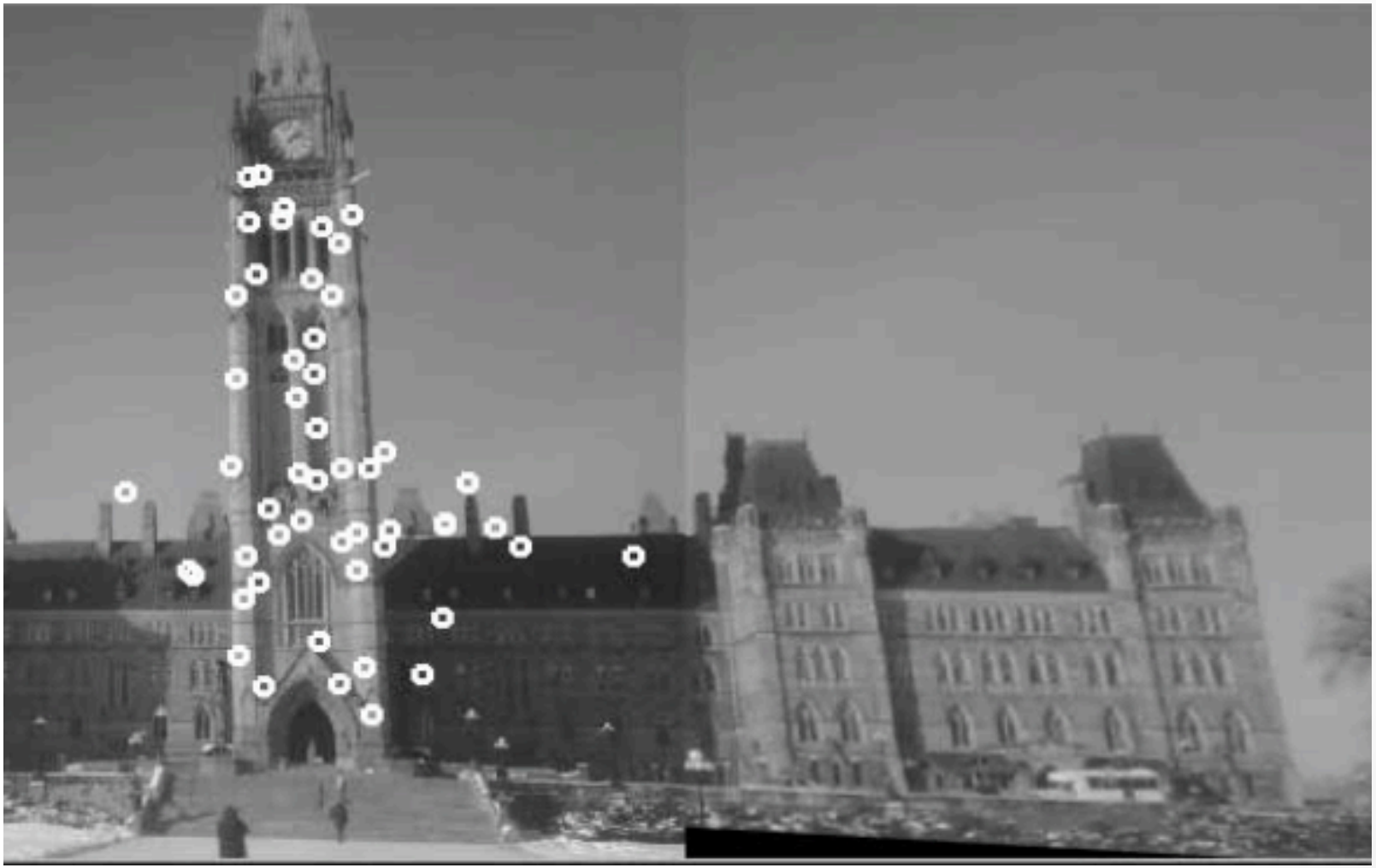
*com 4 pontos use getPerspectiveTransform

Pontos no mesmo plano





Pode-se fazer a transferência de pontos também



Transformação perspectiva

```
Mat result;
warpPerspective(image1, // imagem de entrada
        result,        // imagem de saida
        homography,     // homography
        cv::Size(2*image1.cols,image1.rows));

Mat half(result,
          Rect(0,0,image2.cols,image2.rows)); //cria uma roi

image2.copyTo(half); // copia a segunda imagem sobre a roi
```

Como seria

- para fazer um mosaico agora?
  - várias imagens de algo sob perspectivas diferentes