

15 – Restorasi Citra (Bagian 3)

IF4073 Pemrosesan Citra Digital

Oleh: Rinaldi Munir



Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung
Rinaldi Munir/IF4073-Pemrosesan Citra Digital

2024

Distorsi Geometrik

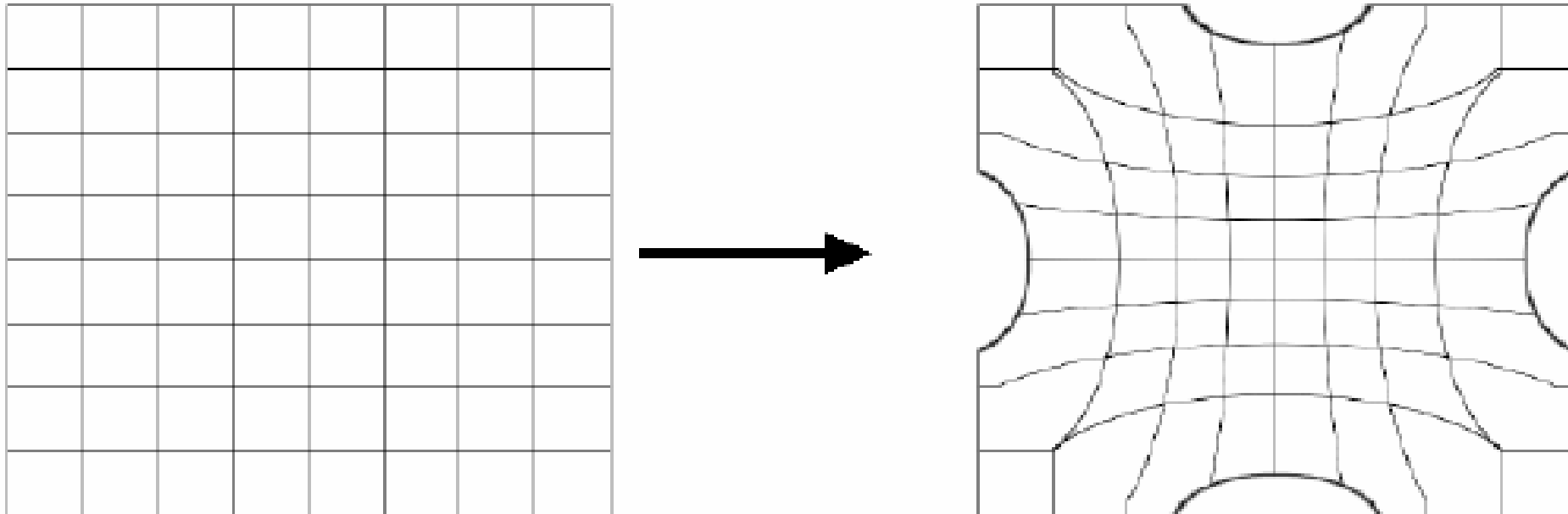


Transformasi Geometrik

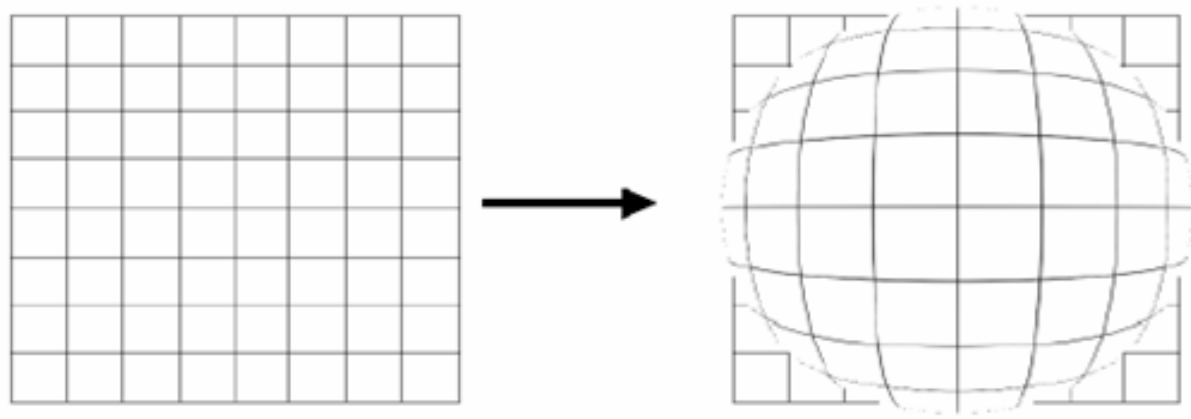
- Disebut juga “*rubber-sheet transformation*”, karena transformasi ini dipandang seperti mencetak citra ke sebuah lembar karet kemudian meregangnya (*stretching*) dengan aturan tertentu.
- Tujuan transformasi geometrik: menghilangkan distorsi geometrik yang terjadi ketika citra diakuisisi.
- Transformasi geometrik memodifikasi hubungan spasial antara pixel-pixel di dalam citra.

Contoh-contoh distorsi geometrik

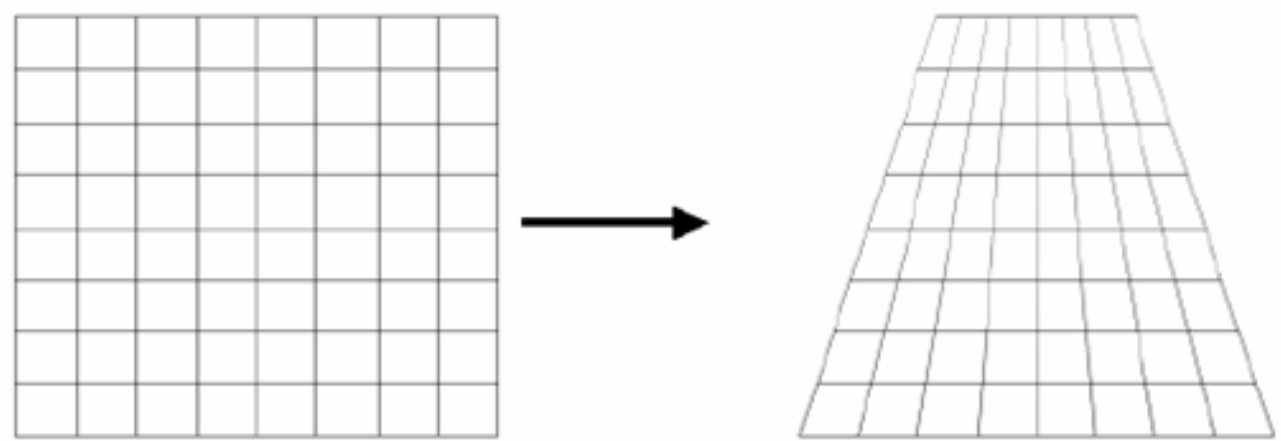
1. **Pincushion distortion** (berkaitan dengan lensa zoom)



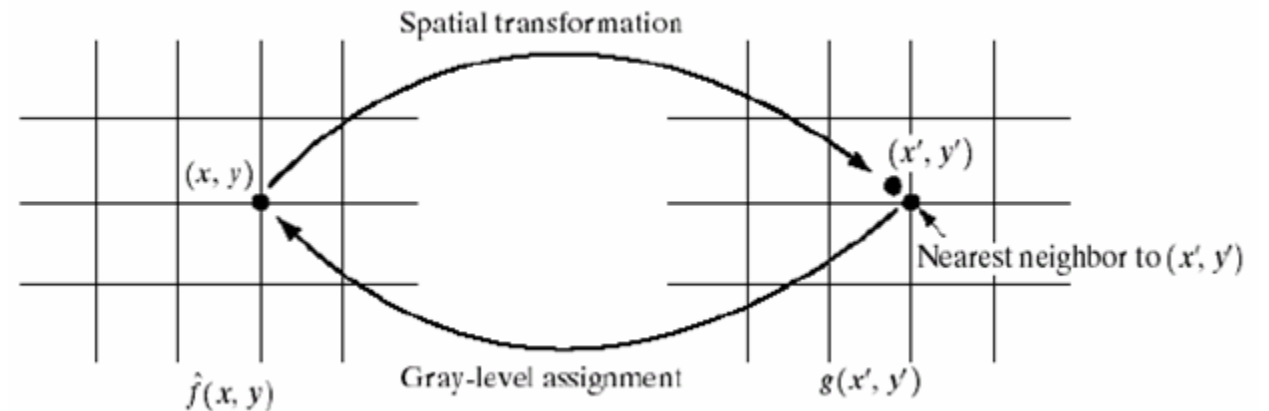
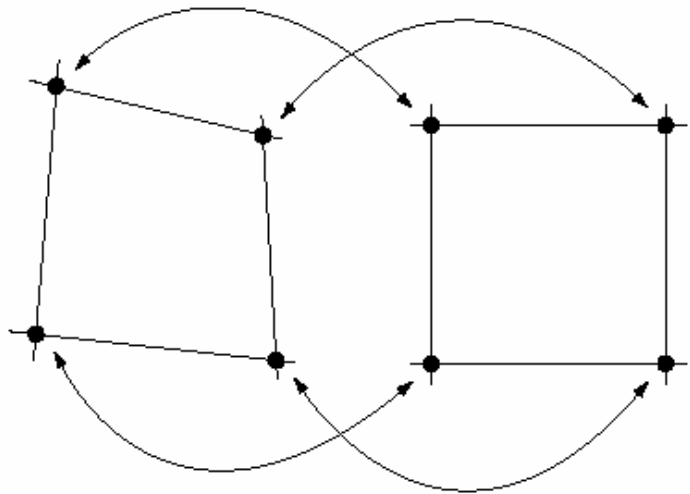
2. Barrel distortion (berkaitan dengan lensa sudut lebar)



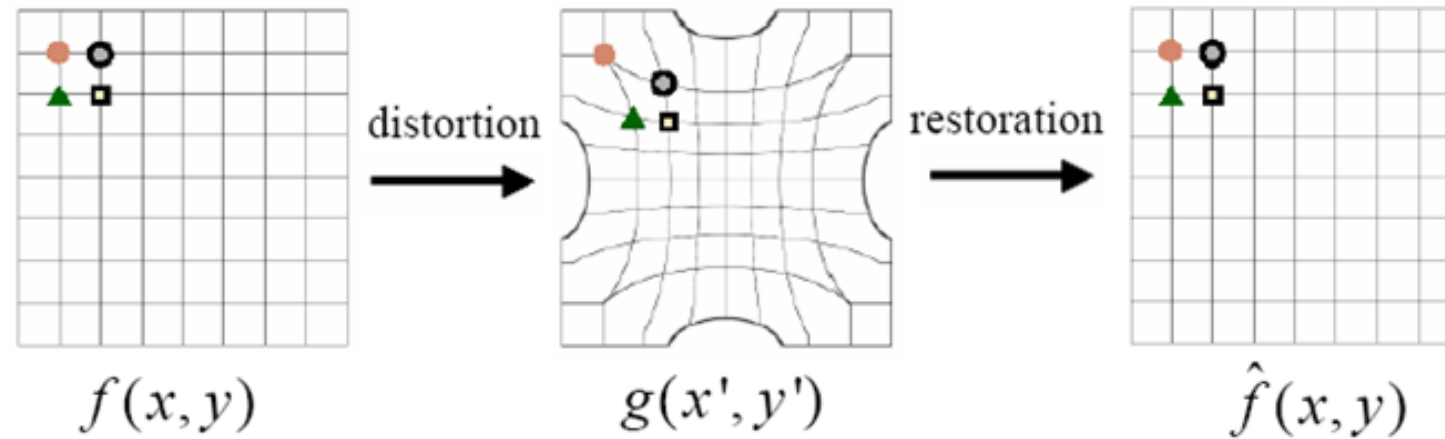
3. Perspective distortion



- Dua Langkah dalam transformasi geometrik:
 1. **Spatial transformation**: menyusun ulang kembali pixel-pixel di dalam citra
 2. **Gray-level interpolation**: mengisi (*assignment*) nilai keabuan *pixel-pixel* di dalam citra hasil transformasi spasial dari langkah 1



1. Transformasi Spasial



Restorasi: $\begin{cases} x' = r(x, y) \\ y' = s(x, y) \end{cases} \xrightarrow{r, s \text{ known}} \begin{cases} x = r'(x', y') \\ y = s'(x', y') \end{cases}$

Contoh:

$$r(x, y) = x/2 \text{ dan } s(x, y) = y/2$$

→ Menyusutkan citra menjadi setengah dalam kedua arah spasial

Sayangnya, fungsi $r(x, y)$ dan $s(x, y)$ yang menggambarkan distorsi geometrik di dalam citra umumnya tidak **diketahui**.

- **Solusi:**

- Untuk merumuskan relokasi spasial pixel dengan menggunakan titik ikat (*tiepoints*) yang sesuai
- *Tiepoints*: subset pixel yang lokasinya di dalam citra input (terdistorsi) dan citra output (dipulihkan) diketahui.

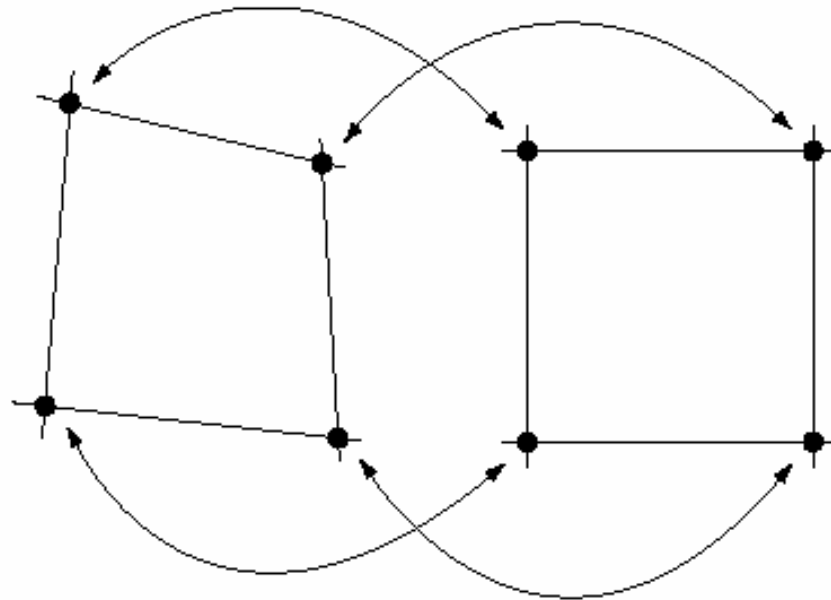


FIGURE 5.32
Corresponding
tiepoints in two
image segments.

- Misalkan proses distorsi geometrik dihampiri dengan model transformasi bilinear:

$$x' = r(x, y) = c_1x + c_2y + c_3xy + c_4$$

$$y' = s(x, y) = c_5x + c_6y + c_7xy + c_8$$

- Karena ada 8 buah koefisien yang tidak diketahui (c_1 sampai c_8), maka diperlukan 4 pasang titik *tiepoint* untuk memecahkan sistem persamaan linier. Satu pasang titik menghasilkan dua buah persamaan linier:

$$\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_8 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & x_1y_1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x_1 & y_1 & x_1y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & x_4 & y_4 & x_4y_4 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x'_1 \\ y'_1 \\ \vdots \\ y'_8 \end{bmatrix}$$

Setelah 8 koefisien tersebut diketahui, maka citra direstorasi di dalam *quadrilateral region* sebagai berikut:

- Misalkan f adalah citra orisinal (*undistorted image*), g adalah citra terdistorsi
- Misalkan kita ingin menemukan nilai citra orisinal pada titik (x_0, y_0) , yaitu kemana nilai $f_0(x_0, y_0)$ dipetakan ke dalam citra terdistorsi.
- Sulihkan (x_0, y_0) ke dalam persamaan

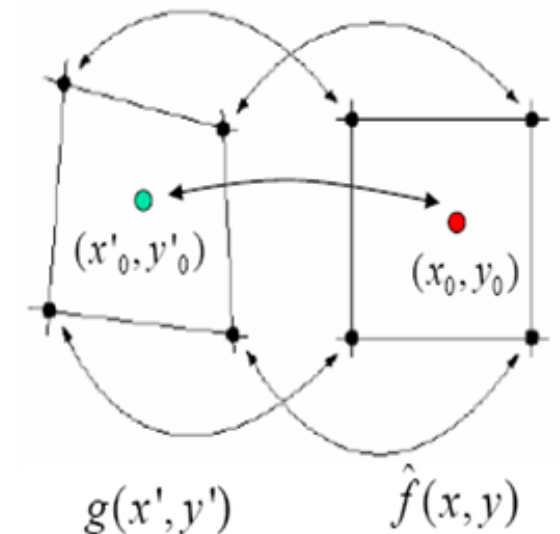
$$x' = r(x, y) = c_1x + c_2y + c_3xy + c_4$$

$$y' = s(x, y) = c_5x + c_6y + c_7xy + c_8$$

$$\blacksquare (x_0, y_0) \Rightarrow x'_0 = r(x_0, y_0), y'_0 = s(x_0, y_0)$$

diperoleh titik (x'_0, y'_0)

- Nilai titik pada citra tak-terdistorsi (*undistorted image*) yang dipetakan ke (x'_0, y'_0) adalah $g(x'_0, y'_0)$.
- Jadi, estimasi citra hasil restorasi adalah $\hat{f}(x_0, y_0) = g(x'_0, y'_0)$.
- Ulangi prosedur di atas untuk titik-titik yang lain



- Secara umum, diperlukan sejumlah *tiepoint* yang cukup untuk membangkitkan himpunan quadrilateral yang mencakup keseluruhan citra, setiap quadrilateral mempunyai 8 buah koefisiennya masing-masing.

- Masalah yang muncul dalam perhitungan transformasi spasial:

$$(x', y') = (r(x, y), s(x, y)) \leftarrow (x, y)$$

- x' dan y' berupa bilangan riil (bukan integer)
- Oleh karena itu diperlukan interpolasi nilai keabuan (*gray-level interpolation*)

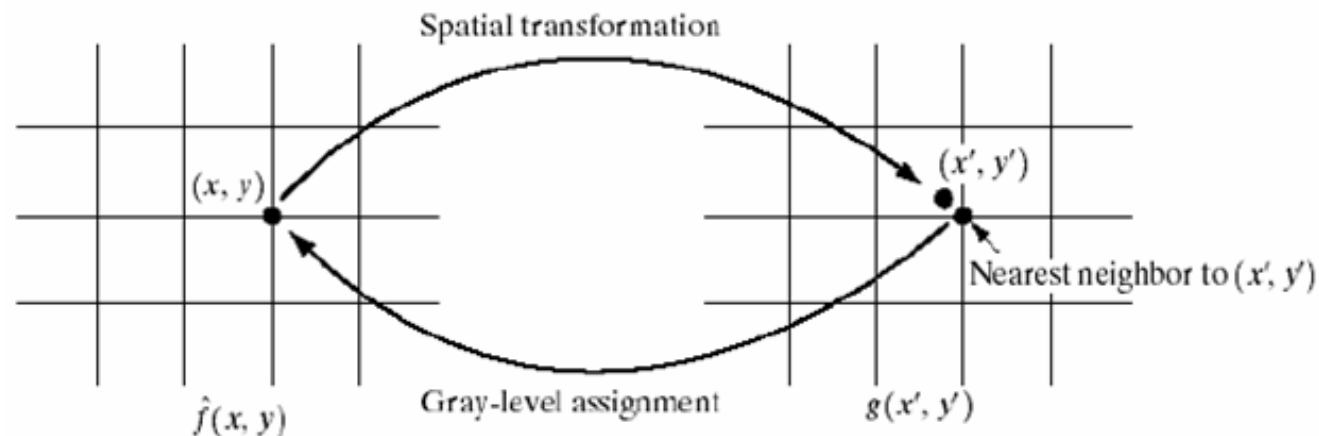


FIGURE 5.33 Gray-level interpolation based on the nearest neighbor concept.

2. Interpolasi Nilai Keabuan

Ada dua macam interpolasi nilai keabuan:

1. *Nearest neighbor interpolation*: dibulatkan ke nilai tetangga terdekat

$$\begin{cases} x' = \text{Round}\{r(x, y)\} \\ y' = \text{Round}\{s(x, y)\} \end{cases}$$

Metode ini sederhana, namun mungkin menimbulkan artefak yang tidak diinginkan, seperti *step-like edges* yang seharusnya lurus atau mulus sebelum transformasi

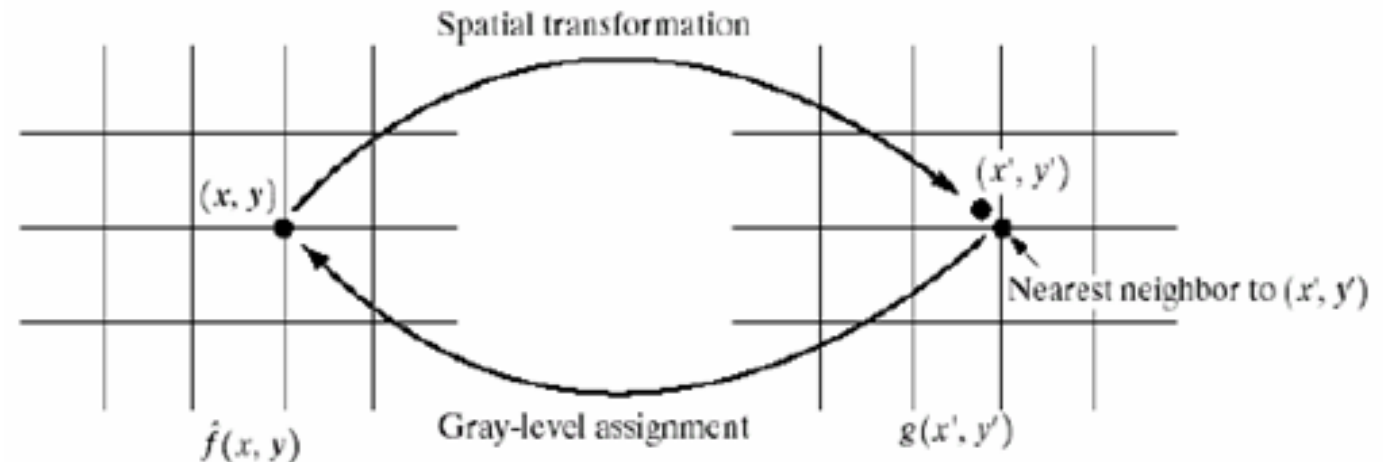
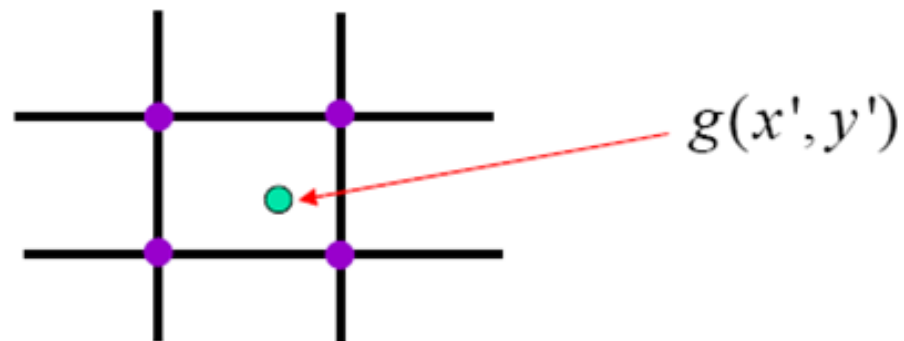


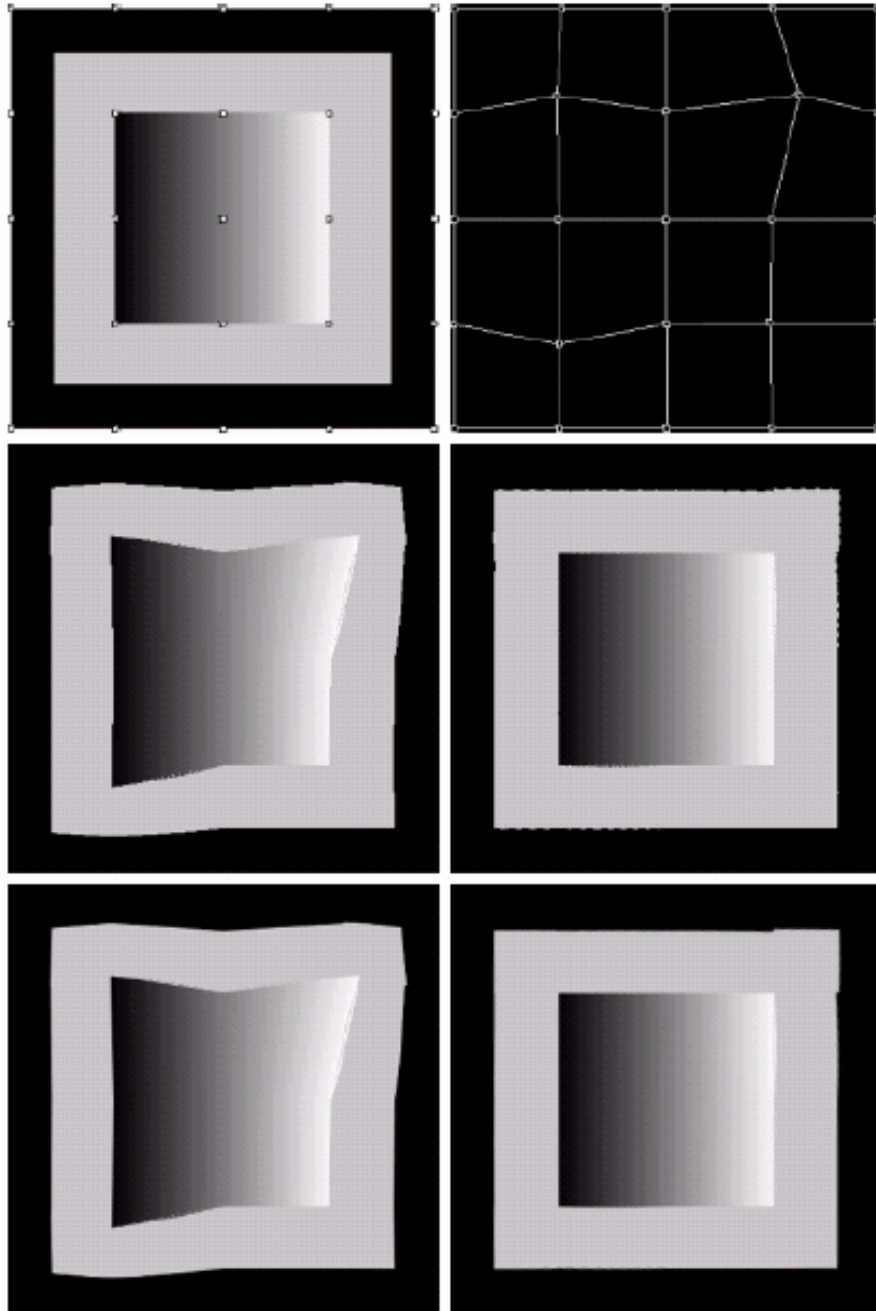
FIGURE 5.33 Gray-level interpolation based on the nearest neighbor concept.

2. *Bilinier interpolation*: menggunakan empat buah nilai keabuan tetangga terdekat untuk menginterpolasi nilai keabuan pada $\hat{g}(x, y)$

$$\hat{g}(x', y') = ax' + by' + cx'y' + d$$

Jadi, $\hat{f}(x, y) = \hat{g}(x', y')$





a	b
c	d
e	f

FIGURE 5.34 (a) Image showing tiepoints. (b) Tiepoints after geometric distortion. (c) Geometrically distorted image, using nearest neighbor interpolation. (d) Restored result. (e) Image distorted using bilinear interpolation. (f) Restored image.

- Menemukan pasangan *tiepoints* yang sesuai adalah masalah yang sulit
- Teknik yang umum digunakan:
 1. Pencocokan fitur
 - warna, histogram lokal, PCA lokal, tekstur
 2. Metode deteksi sudut
 3. Menemukan distorsi geometrik antara dua kontur
 4. Dsb

Blind Geometric Distortion Correction on Images Through Deep Learning

Xiaoyu Li¹

Bo Zhang¹

Pedro V. Sander¹

Jing Liao²

¹The Hong Kong University of Science and Technology

²City University of Hong Kong

Abstract

We propose the first general framework to automatically correct different types of geometric distortion in a single input image. Our proposed method employs convolutional neural networks (CNNs) trained by using a large synthetic distortion dataset to predict the displacement field between distorted images and corrected images. A model fitting method uses the CNN output to estimate the distortion parameters, achieving a more accurate prediction. The final corrected image is generated based on the predicted flow using an efficient, high-quality resampling method. Experimental results demonstrate that our algorithm outperforms traditional correction methods, and allows for interesting applications such as distortion transfer, distortion exaggeration, and so on.



Figure 1. Our proposed learning-based method can blindly correct images with different types of geometric distortion (first row) providing high-quality results (second row).

tions easily (see Figure 2).

Geometric distortion correction is highly desired in both photography and computer vision applications. For example, lens distortion violates the pin-hole camera model as-

Correct image for lens distortion

- Sumber: Mathwork (Matlab)

```
% Create a set of calibration images.
images = imageSet(fullfile(toolboxdir('vision'),'visiondata',...
    'calibration','fishEye'));
%%
% Detect the calibration pattern.
[imagePoints, boardSize] = detectCheckerboardPoints(images.ImageLocation);
%%
% Generate the world coordinates of the corners of the squares. The
% square size is in millimeters.
squareSize = 29;
worldPoints = generateCheckerboardPoints(boardSize, squareSize);
```

```
%%  
% Calibrate the camera.  
cameraParams = estimateCameraParameters(imagePoints,worldPoints);  
%%  
% Remove lens distortion and display the results.  
I = images.read(1);  
J1 = undistortImage(I,cameraParams);  
%%  
% Display the original and corrected images.  
figure; imshowpair(I,J1,'montage');  
title('Original Image (left) vs. Corrected Image (right)');  
  
J2 = undistortImage(I,cameraParams,'OutputView','full');  
figure; imshow(J2);  
title('Full Output View');
```

Original Image (left) vs. Corrected Image (right)



Full Output View

