

## فصل ششم

# تبدیلات مکانی

هدف : آشنایی با نحوه انجام تبدیلات مکانی مختلف

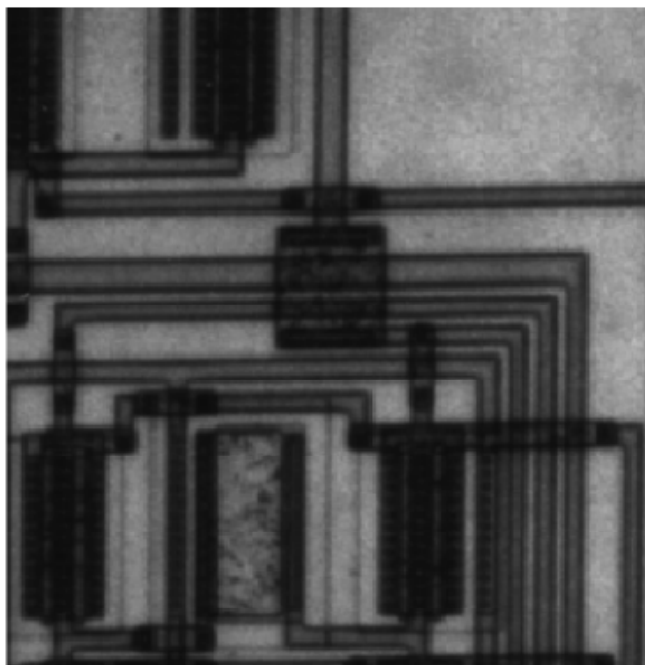
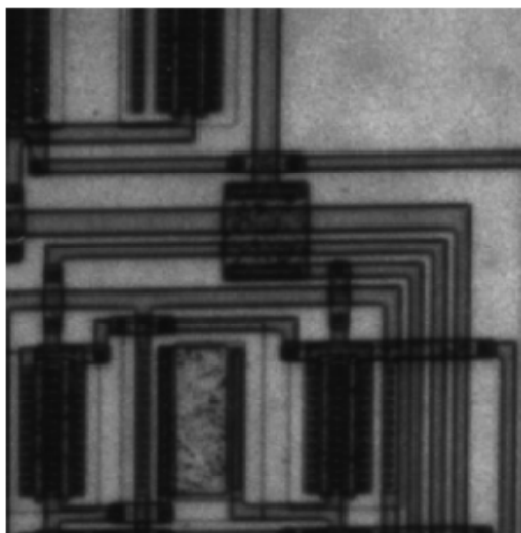
منظور از تبدیل مکانی<sup>۱</sup> تغییر دادن رابطه‌ی مکانی بین پیکسل‌های موجود در تصویر اصلی است.

### تغییر ابعاد<sup>۲</sup> تصویر

برای تغییر ابعاد یک تصویر از تابع `imresize` میتوان استفاده کرد. در این تابع باید ضریب بزرگنمایی<sup>۳</sup> را که عددی کوچکتر یا بزرگتر از ۱ است، تعیین کرد. اگر قصد بزرگتر کردن اندازه‌ی تصویر را دارید باید عددی بزرگتر از ۱ و اگر قصد کوچکتر کردن اندازه‌ی تصویر را دارید باید عددی کوچکتر از ۱ را انتخاب کنید.

مثال:

```
I = imread('circuit.tif');
J = imresize(I,1.25);
imshow(I)
figure, imshow(J)
```



به جای تعیین ضریب بزرگنمایی، می‌توانید ابعاد تصویر خروجی را تعیین کنید، اما اگر ابعاد را به درستی انتخاب نکنید تصویر خروجی دچار اعوجاج خواهد شد. اگر به جای یکی از ابعاد تصویر از NaN استفاده کنید، متلب به طور خودکار آن را طوری محاسبه می‌کند که نسبت ابعاد<sup>۴</sup> تصویر اولیه حفظ شود. مثال:

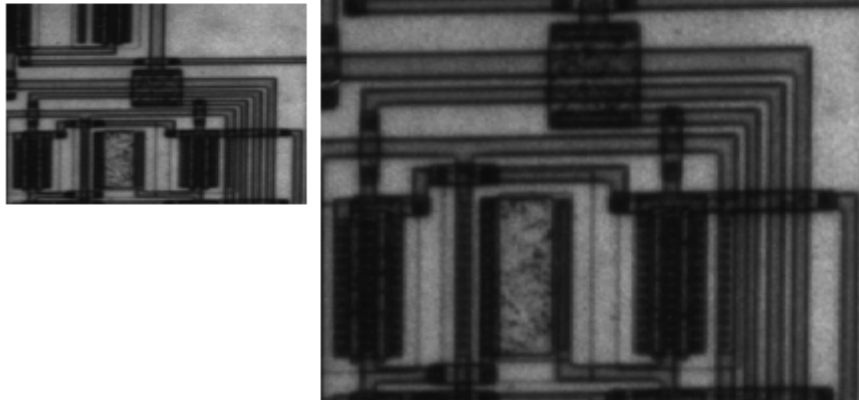
```
I = imread('circuit.tif');
J = imresize(I,[100 150]);
imshow(I)
figure, imshow(J)
```

<sup>1</sup> Spatial Transform

<sup>2</sup> Resizing

<sup>3</sup> Magnification Factor

<sup>4</sup> Aspect Ratio



برای تغییر ابعاد مورد نیاز در پردازش چند درجه‌ی تفکیک می‌توانید از تابع `impyramid` استفاده کنید.

تابع `imresize` برای افزایش ابعاد تصویر از درونیابی<sup>۵</sup> استفاده می‌کند. روش پیش فرض برای درونیابی، روش `'bicubic'` است. می‌توان از روشهای `'nearest'` و `'bilinear'` نیز استفاده کرد (برای دیدن حالت‌های مختلف به صفحه‌ی مرجع دستور `imresize` مراجعه کنید). مثال:

```
Y = imresize(X,[100 150],'bilinear')
```

### جلوگیری از پدیده همپوشانی<sup>۶</sup> (فرکانسی)

در هنگام کاهش ابعاد تصویر (به ویژه تصاویر با تباین زیاد)، تصویر حاصله دارای الگوهای پله‌ای شکل و یا الگوهای اثر ریپل (به نام `moire`) خواهد بود. برای کاهش میزان این اثر، دستور `imresize` به طور پیش‌فرض (به جز برای درونیابی نوع `'nearest'`) از فیلترهای ضدهمپوشانی استفاده می‌کند. برای غیرفعال کردن استفاده از این فیلترها، ویژگی (یا پارامتر) `Antialiasing` را برابر `false` قرار دهید.

### چرخاندن یک تصویر

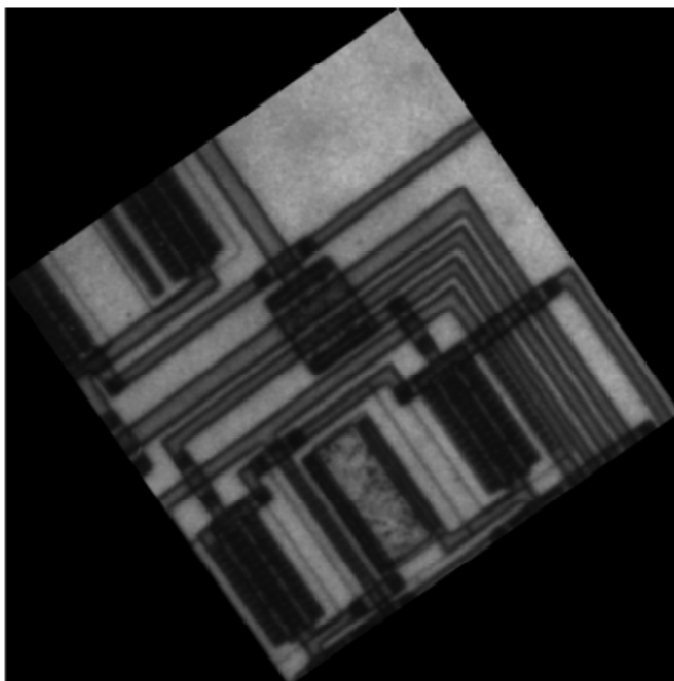
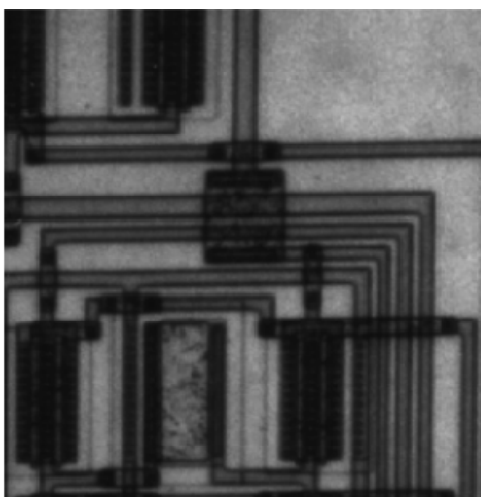
از دستور `imrotate` برای چرخاندن یک تصویر می‌توان استفاده کرد. در این دستور مقدار زاویه‌ی چرخش بر حسب درجه نیز ذکر می‌شود. اگر مقدار درجه مثبت باشد، در خلاف جهت حرکت عقربه‌های ساعت و اگر مقدار درجه منفی باشد، در جهت حرکت عقربه‌های ساعت عمل چرخش انجام می‌شود. تصویر خروجی معمولاً بزرگتر از تصویر ورودی است تا تصویر اولیه را در برگیرد. اما اگر بخواهید، می‌توانید به کمک اضافه کردن آرگومان `'crop'` ابعاد تصویر خروجی را برابر تصویر ورودی تنظیم کنید. دستور `imrotate` از درونیابی نزدیکترین همسایه (`nearest-neighbor`) برای محاسبه پیکسل‌های خروجی تصویر استفاده می‌کند. برای دیدن لیست دیگر روشهای درونیابی موجود در دستور مذکور به صفحه‌ی مرجع آن مراجعه کنید.

<sup>5</sup> Interpolation

<sup>6</sup> Aliasing

مثال:

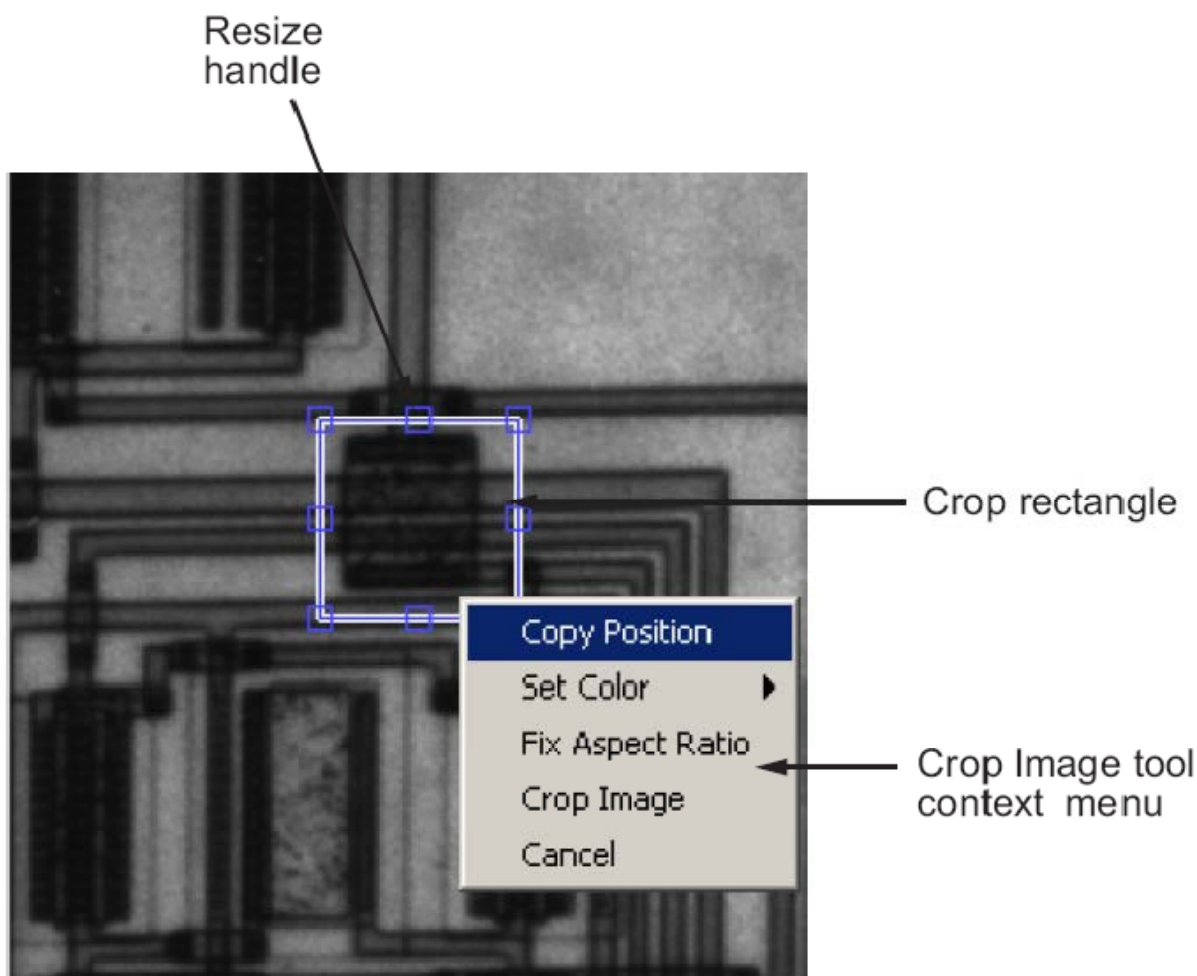
```
I = imread('circuit.tif');
J = imrotate(I,35,'bilinear');
imshow(I)
figure, imshow(J)
```



### بُرش (یا کندن تکه‌ای از) یک تصویر

برای استخراج یک ناحیه‌ی مستطیلی-شکل می‌توانید از دستور `imcrop` استفاده کنید. مشخصات ناحیه‌ی مذکور را می‌توانید به کمک ماوس و یا با برنامه‌نویسی تعیین کنید.  
مثال (استفاده از ماوس):

```
I = imread('circuit.tif')
J = imcrop(I);
```



مثال: (استفاده از برنامه نویسی)

```
I = imread('circuit.tif');
J = imcrop(I,[60 40 100 90]);
```

### انجام تبدیلات دوبعدی مکانی دلخواه

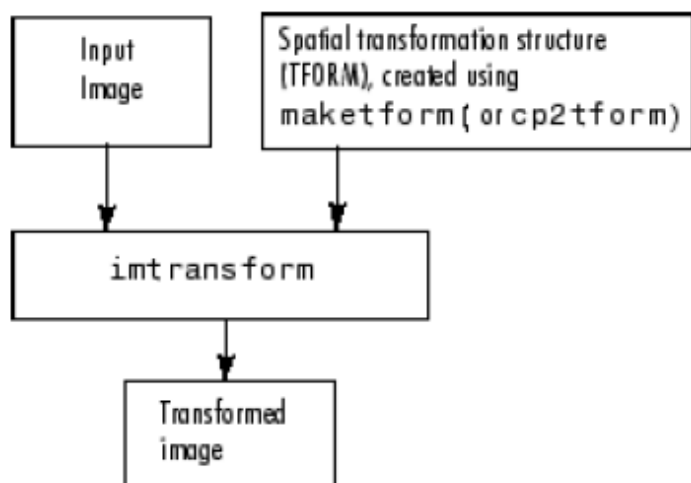
برای انجام یک تبدیل مکانی دلخواه سه مرحله باید طی شود:

۱- تعیین پارامترهای تبدیل مکانی

۲- ایجاد ساختاری به نام TFORM

۳- انجام تبدیل به کمک دستور imtransform

یک ساختار TFORM، ساختاری است که شامل تمام پارامترهای لازم برای انجام یک تبدیل دوبعدی می‌باشد. برای تولید یک ساختار TFORM می‌توان از دستور maketform استفاده کرد. شکل زیر مراحل فوق را نشان می‌دهد:



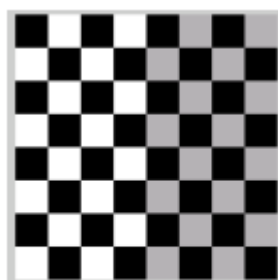
## Overview of General 2-D Spatial Transformation Process

مثال: انجام یک تبدیل مکانی دوبعدی

در این مثال، یک تبدیل ساده به نام انتقال انجام می‌شود.

گام ۱: خواندن تصویر ورودی

```
cb = checkerboard;
imshow(cb)
```



Original Image

گام ۲: تعریف تبدیل

برای تعریف بسیاری از تبدیلات استفاده از یک ماتریس مربعی به طول ۳ کافی است. شما می‌توانید با تعیین چند جفت نقطه متناظر از تصویر ورودی و خروجی نیز، تبدیل را مشخص و از دستور maketform برای ایجاد ماتریس مربعی مذکور استفاده کنید. در این مثال از ماتریس تبدیل زیر استفاده می‌شود (معادل عمل انتقال است):

```
xform = [ 1  0  0
          0  1  0
          40 40  1]
```

اولین عدد ۴۰ (از چپ) معرف میزان جابجایی در راستای افقی و دومین عدد ۴۰ معرف میزان جابجایی در راستای عمودی است.

## گام ۳: ایجاد ساختار TFORM

برای ایجاد یک ساختار TFORM می‌توان از دستور `maketform` استفاده کرد. از جمله آرگومانهای ورودی این دستور، تعیین نوع تبدیل و تعیین ماتریس (یا مجموعه‌ی جفت-نقاط) تبدیل است. نوع تبدیل شامل انواع زیر می‌تواند باشد (برای توضیحات بیشتر به صفحه‌ی مرجع مراجعه کنید):

'affine'	Affine transformation in 2-D or N-D
'projective'	Projective transformation in 2-D or N-D
'custom'	User-defined transformation that can be N-D to M-D
'box'	Independent affine transformation (scale and shift) in each dimension
'composite'	Composition of an arbitrary number of more basic transformations

در این مثال، از پارامتر نوع تبدیل 'affine' باید استفاده کنیم:

```
tform_translate = maketform('affine',xform);
```

## گام ۴: انجام تبدیل

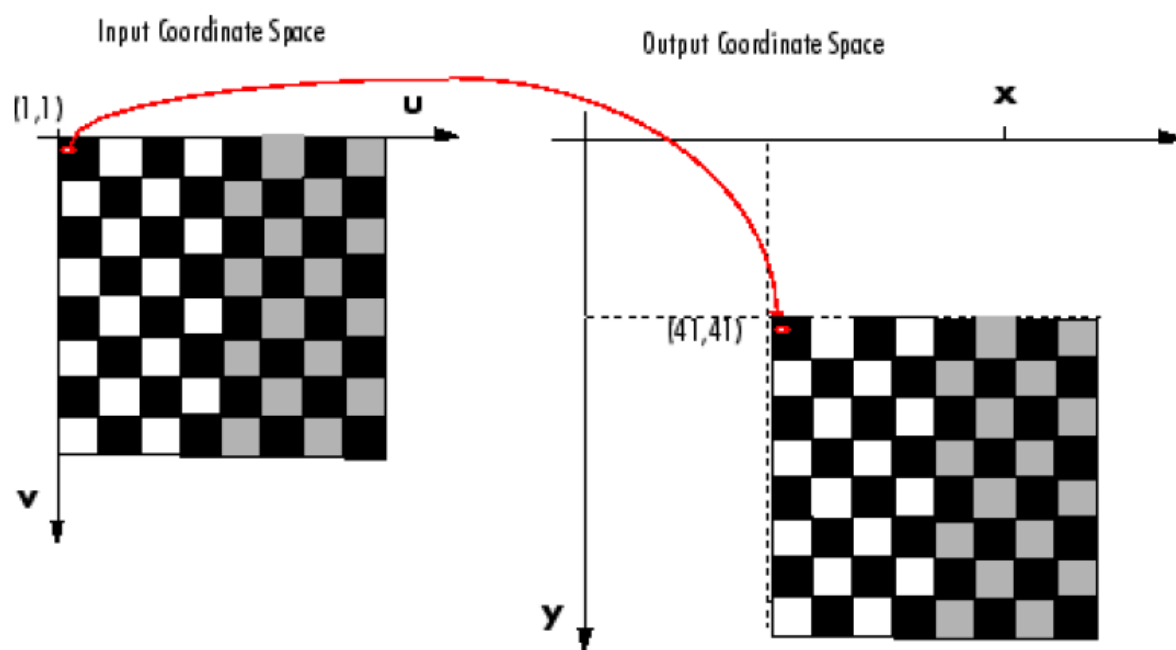
برای انجام تبدیل، از دستور `imtransform` به همراه تصویر ورودی و ماتریس تبدیل مورد نظرمان استفاده می‌کنیم.

```
[cb_trans xdata ydata]= imtransform(cb, tform_translate);
```

آرگومانهای خروجی به ترتیب شامل نتیجه‌ی تبدیل (تصویر تبدیل یافته)، و مختصات افقی (راستای  $x$ ) و مختصات عمودی (راستای  $y$ ) گوشه‌های تصویر تبدیل یافته می‌باشند.

**توجه:** در این بخش از سیستم مختصات دهی مکانی استفاده می‌شود.

شکل زیر تبدیل انجام شده در این مثال را به صورت گرافیکی نشان می‌دهد. ملاحظه می‌کنید که پیکسل واقع در مختصات  $(1,1)$  از تصویر ورودی به مختصات  $(41,41)$  در تصویر خروجی منتقل شده است.

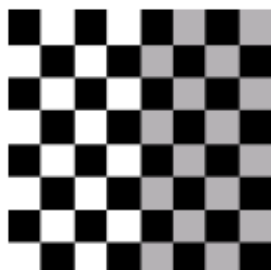


Input Image Translated

البته توجه کنید که تبدیل انجام شده، تاثیری روی مقادیر پیکسلها ندارد. اما اگر برخی تبدیلات مانند تغییر مقیاس و چرخش انجام دهیم، باید مقادیر برخی پیکسلهای جدید به روش درونیابی محاسبه شوند (برای دیدن انواع روشهای موجود برای درونیابی به صفحه‌ی مرجع دستور `imtransform` مراجعه کنید).

**گام ۵:** نمایش تصویر خروجی

```
figure, imshow(cb_trans)
```



### Translated Image

ممکن است فکر کنید که چون تصویر خروجی مشابه تصویر ورودی شده، پس عمل تبدیل تاثیری نداشته است؛ اما، واقعیت این است که تبدیل انتقال تاثیری روی مقادیر پیکسلها ندارد بلکه روی مختصات آنها تاثیرگذار است. اگر به مقادیر آرگومانهای خروجی دستور `imtransform` یعنی `xdata` و `ydata` نگاهی بیندازید، ملاحظه می‌کنید که مقادیر مختصات مکانی تغییر کرده‌اند.

```
xdata =
```

```
41    120
```

```
ydata =
```

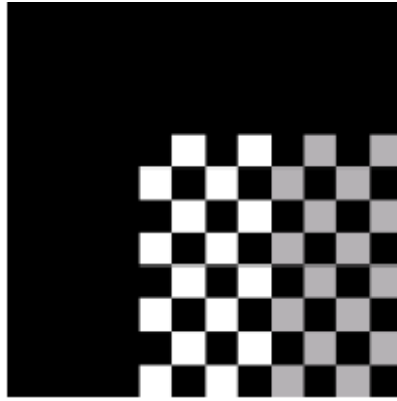
```
41    120
```

اگر می‌خواهید که تصویر خروجی نسبت به مبدا مختصات جدید نمایش داده شود، از ویژگیهای `XData` و `YData` به صورت زیر استفاده کنید:

```
cb_trans2 = imtransform(cb, tform_translate,...
    'XData', [1 (size(cb,2)+ xform(3,1))],...
    'YData', [1 (size(cb,1)+ xform(3,2))]);
figure, imshow(cb_trans2)
```

نتیجه:





توجه:

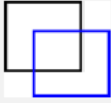
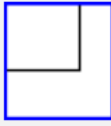

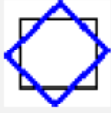
**Note** All the pixels that are now in the output image that do not correspond to locations in the input image are black. `imtransform` assigns a value, called a *fill value*, to these pixels. This example uses the default fill value but you can specify a different one — see “Specifying Fill Values” on page 6-18.

### تعریف داده‌های مربوط به تبدیل

همان طور که گفته شد دو راه برای مشخص کردن نحوه انجام تبدیل وجود دارد: یکی استفاده از ماتریس تبدیل و دیگری مشخص کردن تعدادی جفت-نقطه‌ی متناظر در تصاویر ورودی و خروجی. در هر دو حالت، باید نتیجه را به دستور `maketform` بدهیم.

### راه اول:

با استفاده از دستور `maketform` و دادن یک ماتریس تبدیل  $3 \times 3$  مناسب به آن می‌توان یک ساختار `TFORM` که مناسب برای استفاده در دستور `imtransform` باشد را ایجاد کرد. دستور `imtransform` فقط قادر به انجام تبدیلات دو بعدی است. یکی از انواع متداول تبدیلات دو بعدی، تبدیل `affine` است. برای داشتن این نوع تبدیل باید سطر آخر ماتریس تبدیل به صورت بردار ستونی `[0; 0; 1]` باشد. انواع تبدیلات `affine` و ماتریسهای تبدیل هر کدام از آنها در جدول زیر آمده است:

Affine Transform	Example	Transformation Matrix	
Translation		$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$	$t_x$ specifies the displacement along the $x$ axis $t_y$ specifies the displacement along the $y$ axis.
Scale		$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$s_x$ specifies the scale factor along the $x$ axis $s_y$ specifies the scale factor along the $y$ axis.
Shear		$\begin{bmatrix} 1 & sh_y & 0 \\ sh_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$sh_x$ specifies the shear factor along the $x$ axis $sh_y$ specifies the shear factor along the $y$ axis.
Rotation		$\begin{bmatrix} \cos(q) & \sin(q) & 0 \\ -\sin(q) & \cos(q) & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$q$ specifies the angle of rotation.

### راه دوم:

در روش دوم، مجموعه‌ای از جفت-نقاط متناظر متعلق به تصویر ورودی و تصویر خروجی را مشخص می‌کنیم و آنها را به دستور `maketform` می‌دهیم. شما باید سه جفت نقطه (یعنی در کل، ۶ نقطه) متناظر و غیر واقع بر یک خط راست از تصویر ورودی و تصویر خروجی را مشخص کنید. این سه جفت نقطه در حقیقت یک مثلث را در تصویر ورودی و در تصویر خروجی مشخص می‌کنند.

مثال:

```
in_points = [11 11;21 11; 21 21]
out_points = [51 51;61 51;61 61]
tform2 = maketform('affine',inpts,outpts)
```

### ایجاد ساختار TFORM

بعد از ایجاد داده‌های تبدیل (مطالب قسمت قبلی)، باید به کمک دستور `maketform` یک ساختار TFORM ایجاد کنید و سپس آن را به دستور `imtransform` بدهید تا تبدیل را انجام دهد. نحوه‌ی استفاده از دستور `maketform` در مثال زیر نشان داده شده است:

```
tform_translate = maketform('affine',xform)
```

آرگومان اول، نوع کلی تبدیل و آرگومان دوم نیز داده‌های تبدیل را مشخص می‌کنند. در حالت کلی دو نوع کلی affine و projective وجود دارد اما علاوه بر اینها، دستور maketform انواع دیگری را نیز پشتیبانی می‌کند. انواع کلی تبدیلات در جدول زیر آمده است:

Transformation Type	Description
'affine'	Transformation that can include translation, rotation, scaling, and shearing. Straight lines remain straight, and parallel lines remain parallel, but rectangles might become parallelograms.
'projective'	Transformation in which straight lines remain straight but parallel lines converge toward vanishing points. (The vanishing points can fall inside or outside the image -- even at infinity.)
'box'	Special case of an affine transformation where each dimension is shifted and scaled independently.
'custom'	User-defined transformation, providing the forward and/or inverse functions that are called by imtransform.
'composite'	Composition of two or more transformations.

### انجام تبدیل

حال به کمک دستور imtransform تبدیل را انجام می‌دهیم. برای نمونه، در مثالی که عمل تبدیل انتقال را انجام می‌داد، از دستور زیر استفاده شده بود:

```
cb_trans = imtransform(cb, tform_translate);
```

دستور imtransform گزینه‌های متنوع دیگری نیز دارد از جمله تعیین اندازه‌ی تصویر خروجی و تعیین مقدار مورد استفاده برای پرکردن نواحی خالی (مانند نواحی سیاه رنگ در شکل صفحه‌ی ۹). در ادامه، هر یک از این دو گزینه مورد بررسی قرار می‌گیرند.

### مشخص کردن مقدار مورد استفاده برای پرکردن نواحی خالی (*fill value*)

مقدار پیش‌فرض fill value برابر صفر (متناظر با رنگ سیاه) است. برای مقادیر (یا رنگهای) دیگر باید پارامتر FillValues را مقداردهی کنید:

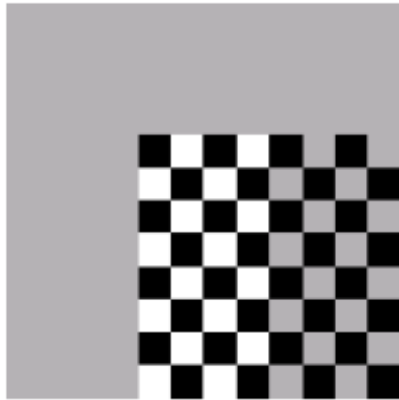
### الف - تصاویر سطح خاکستری

برای مثال، در مثال تبدیل انتقال میتوان چنین نوشت:

```
cb_fill = imtransform(cb, tform_translate,...
    'XData', [1 (size(cb,2)+xform(3,1))],...
```

```
'YData', [1 (size(cb,1)+xform(3,2))],...
'FillValues', .7);
figure, imshow(cb_fill)
```

نتیجه:



**Translated Image with Gray Fill Value**

### ب- تصاویر رنگی

می‌توان از یک عدد اسکالر یا یک بردار سه مقداره استفاده کرد. مثال:

```
rgb = imread('onion.png');
xform = [ 1 0 0
          0 1 0
          40 40 1 ]
tform_translate = maketform('affine',xform);
cb_rgb = imtransform(rgb, tform_translate,...
'XData', [1 (size(rgb,2)+xform(3,1))],...
'YData', [1 (size(rgb,1)+xform(3,2))],...
'FillValues', [187;192;57]);
figure, imshow(cb_rgb)
```



**Translated RGB Image with Color Fill Value**

اگر بخواهید تبدیلی را روی چندین تصویر ورودی (مقلاً یک آرایه‌ی ۴-بُعدی شامل ۱۰ تصویر رنگی هر یک به ابعاد  $200 \times 200 \times 3$ ) مختلف انجام دهید، برای تعیین مقدار fill value سه راه مختلف دارید. اول این که یک مقدار اسکالر را به تنهایی مشخص کنید؛ یعنی، نواحی خالی با یک رنگ خاکستری پر شود. دوم این که یک بردار سه مقدار را مشخص کنید؛ یعنی نواحی خالی را با یک رنگ مشخص پر کنید. سوم این که یک ماتریس  $3 \times 10$  (۳ سطر و ۱۰ ستون) را مشخص کنید؛ یعنی، هر تصویر از رنگ مخصوص به خود برای پر کردن نواحی خالی مربوطه استفاده می‌کند.

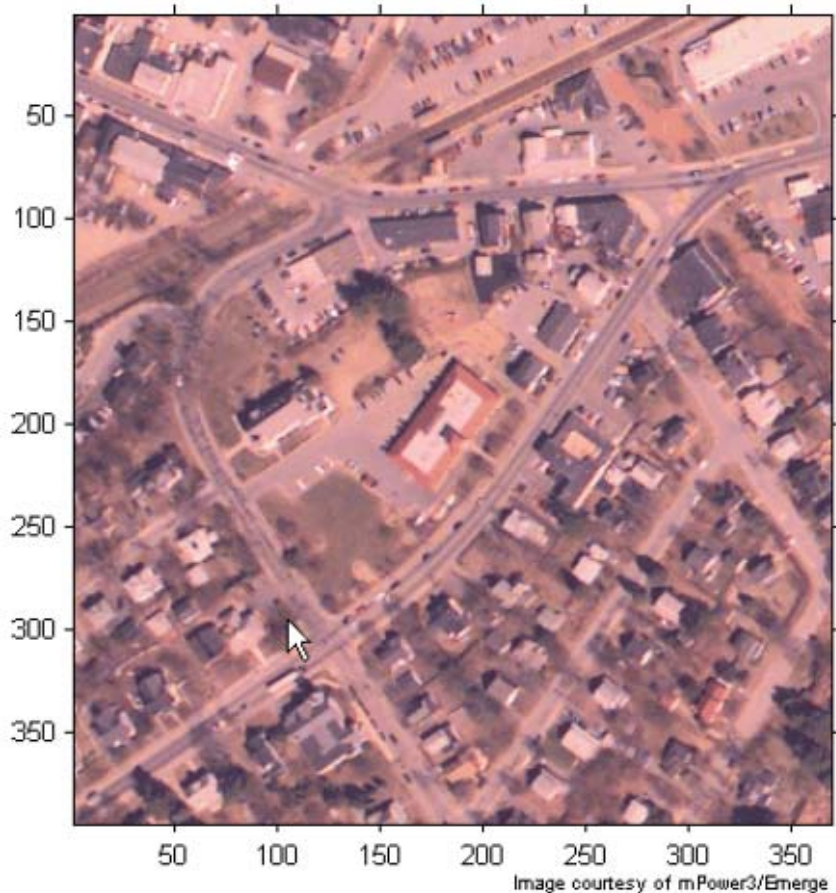
### مثال: تثبیت تصویر

گام ۱: خواندن تصاویر پایه<sup>۷</sup> و تثبیت نشده<sup>۸</sup>

```
base = imread('westconcordorthophoto.png');
unregistered = imread('westconcordaerial.png');
```

گام ۲: نمایش تصویر تثبیت نشده

```
iptsetpref('ImshowAxesVisible','on')
imshow(unregistered)
text(size(unregistered,2),size(unregistered,1)+30, ...
      'Image courtesy of mPower3/Emerge', ...
      'FontSize',7,'HorizontalAlignment','right');
```



<sup>7</sup> Base Image

<sup>8</sup> Unregistered Image

**گام ۳: ایجاد یک ساختار TFPRM**

در این مرحله با استفاده از تعدادی نقاط کنترلی<sup>۹</sup> از قبل تعیین شده یک ساختار TFPRM ایجاد می‌کنیم. این جفت نقاط کنترلی (چند نقطه مربوط به تصویر پایه و چند نقطه هم مربوط به تصویر تثبیت نشده) در یک فایل MAT ذخیره شده‌اند لذا ابتدا این فایل بارگذاری و سپس استفاده می‌شود:

```
load westconcordpoints
tform = cp2tform(input_points, base_points, 'projective');
```

دستور cp2tform از روی جفت نقاط داده شده به آن، ساختار TFORM متناظر را به ما می‌دهد.

**گام ۴: تبدیل تصویر تثبیت نشده**

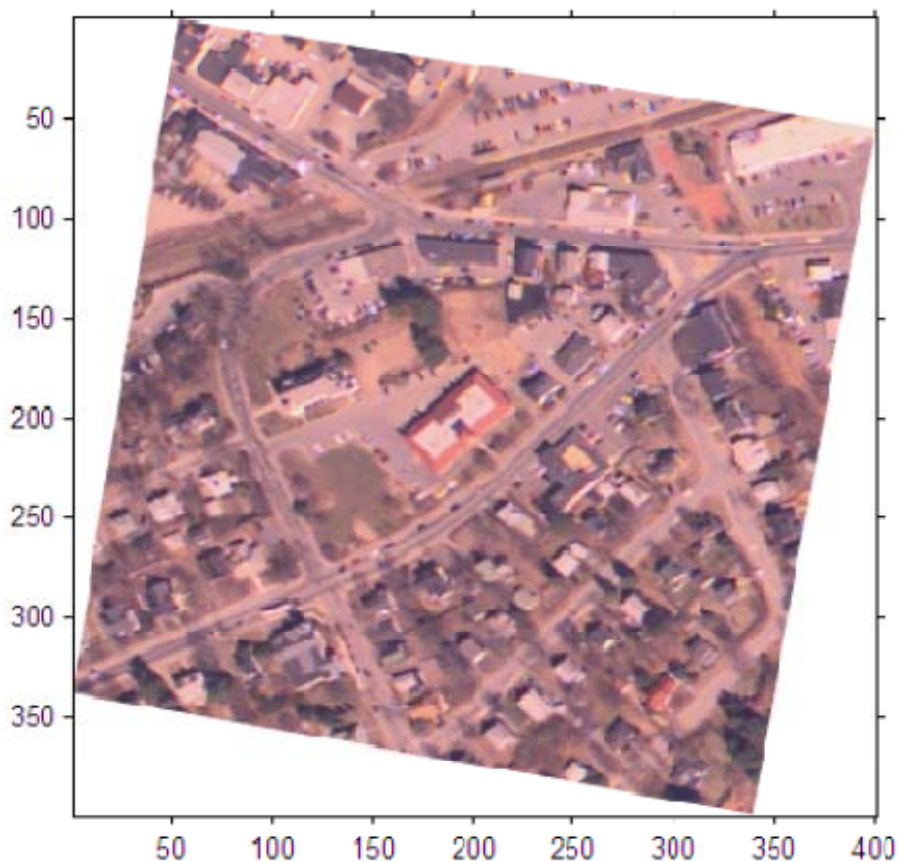
برای تثبیت تصویر تثبیت نشده با تصویر پایه، باید عمل تبدیل را روی تصویر تثبیت نشده انجام دهیم. در کد زیر، به طور اختیاری از مقدار مشخصی برای پارامتر fill value (و متناظر با رنگ سفید) استفاده شده است. در حقیقت، استفاده از پس‌زمینه سفید در هنگام قرار دادن تصویر پایه روی تصویر تثبیت شده باعث ارزیابی بهتر عمل تثبیت می‌شود.

```
registered = imtransform(unregistered, tform, 'FillValues', 255);
```

برای نمایش تصویر تثبیت شده:

```
figure; imshow(registered); hold on
```

نتیجه:



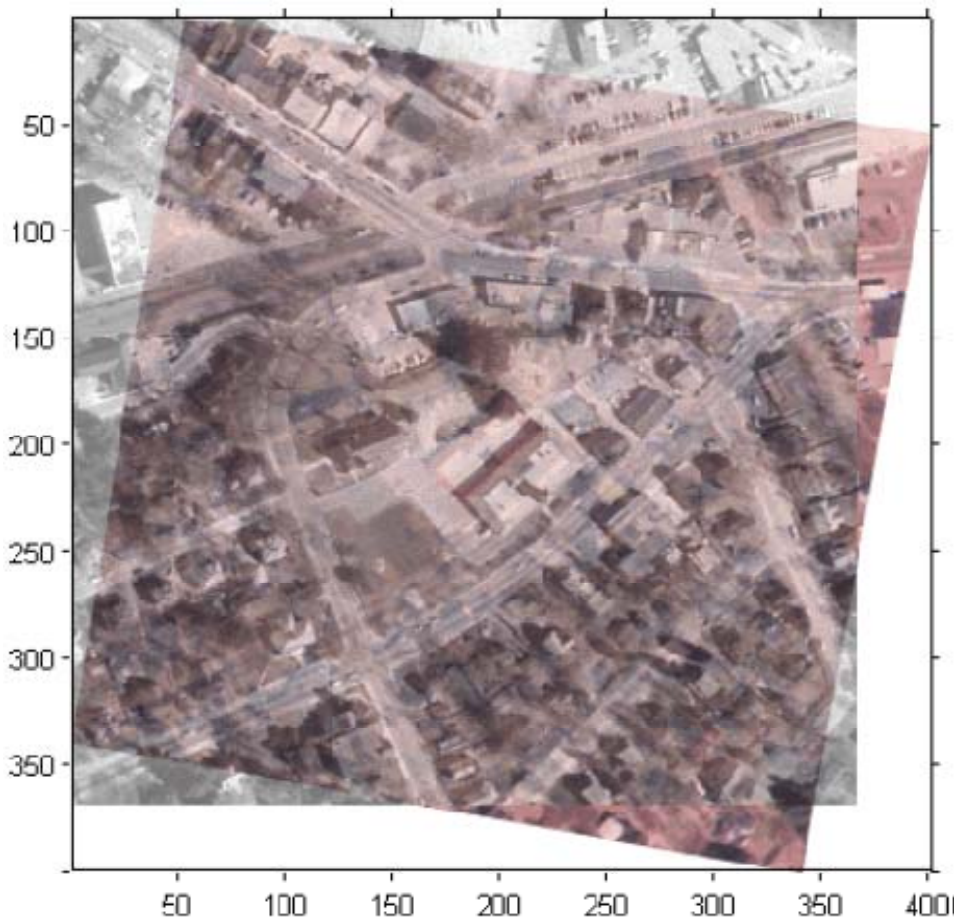
<sup>9</sup> Control Points



**گام ۵:** روی هم قرار دادن تصویر پایه و تصویر تثبیت شده

در این مرحله یک نسخه‌ی نیمه شفاف از تصویر پایه را روی تصویر تثبیت شده قرار می‌دهیم:

```
h = imshow(base, gray(256));
set(h, 'AlphaData', 0.6)
```



**Registered Image with Base Image Overlay**

ملاحظه می‌کنید که دو تصویر به خوبی روی هم قرار نگرفته‌اند. (یا به اصطلاح، تثبیت نشده<sup>۱۰</sup> به نظر می‌رسند و گرنه ما می‌دانیم که واقعاً تثبیت شده‌اند) علت این امر این است که در این مثال فرض شده است که هر دو تصویر در یک سیستم مختصات مشترک قرار دارند که البته، این طور نیست. در گامهای بعدی، دو روش برای حل این مشکل ارائه می‌شود.

**گام ۶:** استفاده از پارامترهای ورودی XData و YData

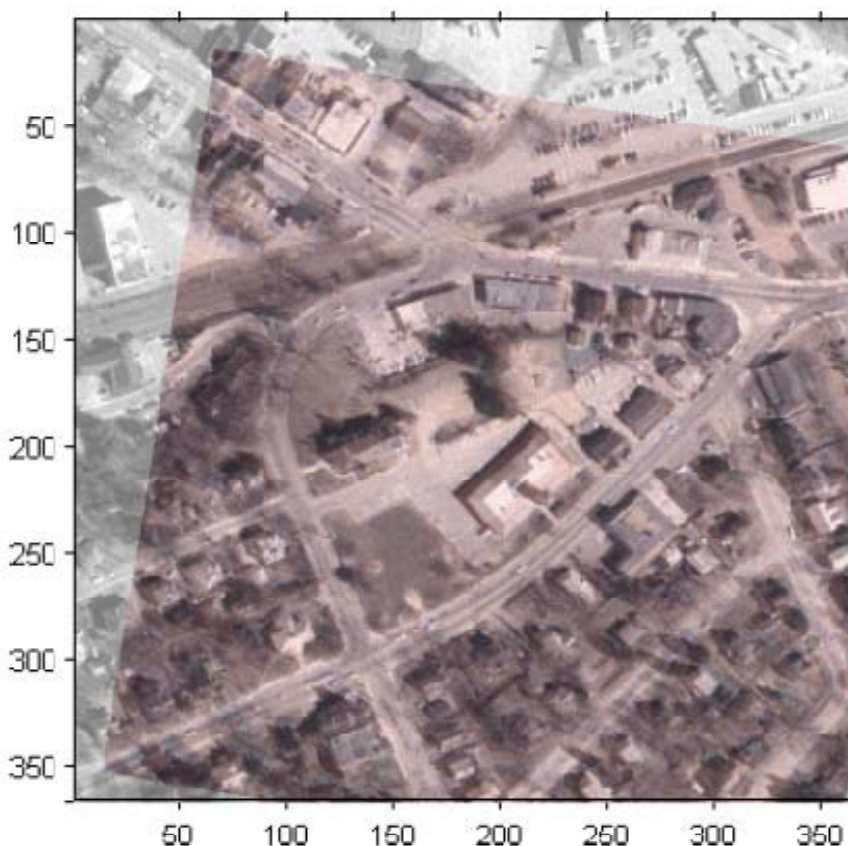
<sup>10</sup> Misregistered

یک راه برای اینکه کاری کنیم که تصاویر تثبیت شده، واقعاً تثبیت شده به نظر برسند، این است که آن نواحی‌ای از تصویر تثبیت شده را که خارج از حدود تصویر پایه است، بُرش<sup>۱۱</sup> بزنیم. برای این کار از پارامترهای 'XData' و 'YData' استفاده می‌کنیم:

```
registered1 = imtransform(unregistered,tform,'FillValues', 255,'XData',...
    [1 size(base,2)],'YData', [1 size(base,1)]);
```

حال تصویر تثبیت شده را نمایش داده و سپس نسخه‌ی نیمه شفاف‌ی از تصویر پایه را به منظور مقایسه با تصویر تثبیت شده، روی آن قرار می‌دهیم:

```
figure; imshow(registered1)
hold on
h = imshow(base, gray(256));
set(h, 'AlphaData', 0.6)
```



### Registered Image Truncated with Base Image Overlay

ملاحظه می‌کنید که عمل تثبیت شدن به خوبی نشان داده شده است اما قسمتی از تصویر تثبیت شده حذف شده است. گام بعدی راه دیگری ارائه می‌دهد.

<sup>11</sup> Truncate



گام ۷: استفاده از مقادیر خروجی `xdata` و `ydata`

یک راه دیگر این است که تصویر تثبیت شده را با تمام محدوده‌ی مختصات خود محاسبه کنیم و از گزینه‌ی اختیاری موجود در دستور `imtransform` که مختصات تصویر تبدیل یافته نسبت به تصویر اصلی (اولیه) را برمی‌گرداند، استفاده کنیم.

```
[registered2 xdata ydata] = imtransform(unregistered, tform, 'FillValues', 255);
```

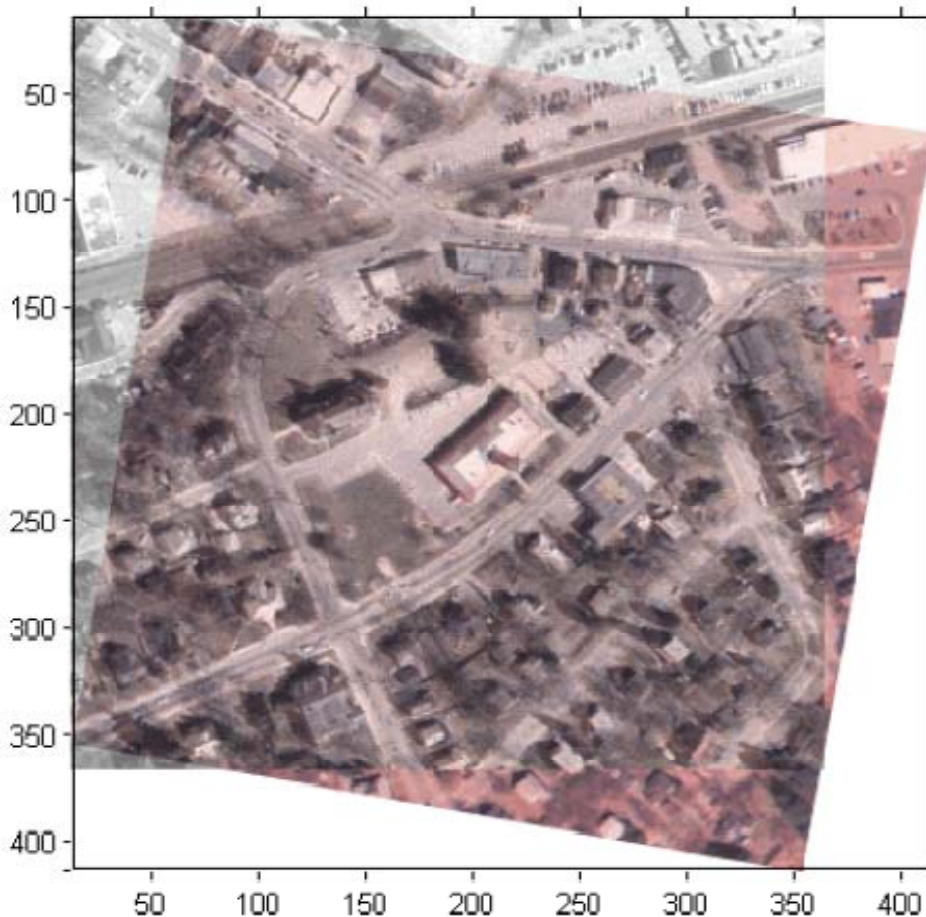
حال تصویر تثبیت شده را نمایش داده و سپس نسخه‌ی نیمه شفاف‌ی از تصویر پایه را به منظور مقایسه با تصویر تثبیت شده، روی آن قرار می‌دهیم:

```
figure; imshow(registered2, 'XData', xdata, 'YData', ydata)
```

```
hold on
```

```
h = imshow(base, gray(256));
```

```
set(h, 'AlphaData', 0.6)
```

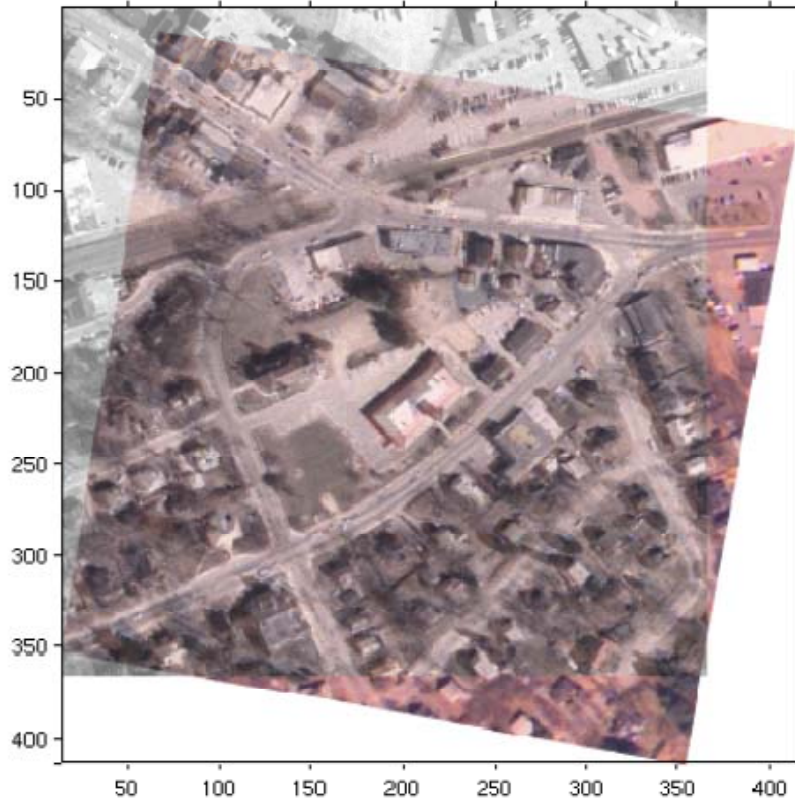


**Registered Image with Base Image Overlay**

حال محورها را به گونه‌ای تنظیم می‌کنیم که تصویر پایه به طور کامل نشان داده شود. حال با توجه به این شکل ملاحظه کنید که چگونه عمل تثبیت به خوبی نشان داده شده است و نمایش داده شده‌اند.

```
ylim = get(gca, 'YLim');
```

```
set(gca, 'YLim', [0.5 ylim(2)])
```



**Registered Image with Base Image Overlay and Adjusted Axes**

## فصل هفتم

# ثبیت تصویر

هدف : آشنایی با نحوه‌ی ثبیت انجام تصویر و انتخاب نقاط کنترلی

**تعریف تثبیت تصویر:** قرار دادن چند تصویر مختلف اما مربوط به یک صحنه روی هم طوری که اشیاء یکسان و مشترک در تصاویر روی هم بیفتند.

**اهمیت تثبیت تصویر:** تثبیت تصویر معمولاً یک مرحله‌ی مقدماتی در برخی فرآیندهای پردازش تصویر است.

از بین چند تصویری که قصد تثبیت آنها را داریم، معمولاً یکی به عنوان تصویر پایه یا تصویر مرجع<sup>۱۲</sup> انتخاب می‌شود تا دیگر تصاویر (که تصاویر ورودی<sup>۱۳</sup> نامیده می‌شوند) با آن مقایسه شوند. هدف از تثبیت تصویر انجام یک تبدیل مناسب است طوری که تصاویر ورودی به طور «مناسب» روی تصویر مرجع قرار بگیرند.

مهمترین مرحله از مراحل تثبیت تصویر، تعیین پارامترهای تبدیل مناسب است. برای تعیین پارامترهای تبدیل، شما باید یک سری جفت نقاط را در تصویر مرجع و تصویر ورودی مشخص (و در صورت لزوم تصحیح<sup>۱۴</sup>) کنید. این جفت نقاط مربوط به ویژگیها<sup>۱۵</sup> و یا محلها<sup>۱۶</sup> مشترک در دو تصویر می‌باشند. حال با مقایسه و تطابق نقاط (و یا به اصطلاح نگاشت نقطه<sup>۱۷</sup>) می‌توان تبدیل مناسب را یافت.

توجه کنید که ممکن است لازم باشد شما عملیات فوق‌الذکر را چندین بار تکرار و تبدیلهای مختلفی را بررسی و ارزیابی کنید تا به نتیجه‌ی مطلوب دست بیابید. در برخی مواقع شما باید با انجام مکرر تثبیت تصویر، ابتدا اعوجاجات (یا ناهمسانیهای) عمده<sup>۱۸</sup>، و سپس اعوجاجات جزئی و کوچک را حذف کنید. شکل زیر مراحل مختلف و متداول عملیات تثبیت تصویر را نشان می‌دهد.

---

12 Reference

13 Input Images

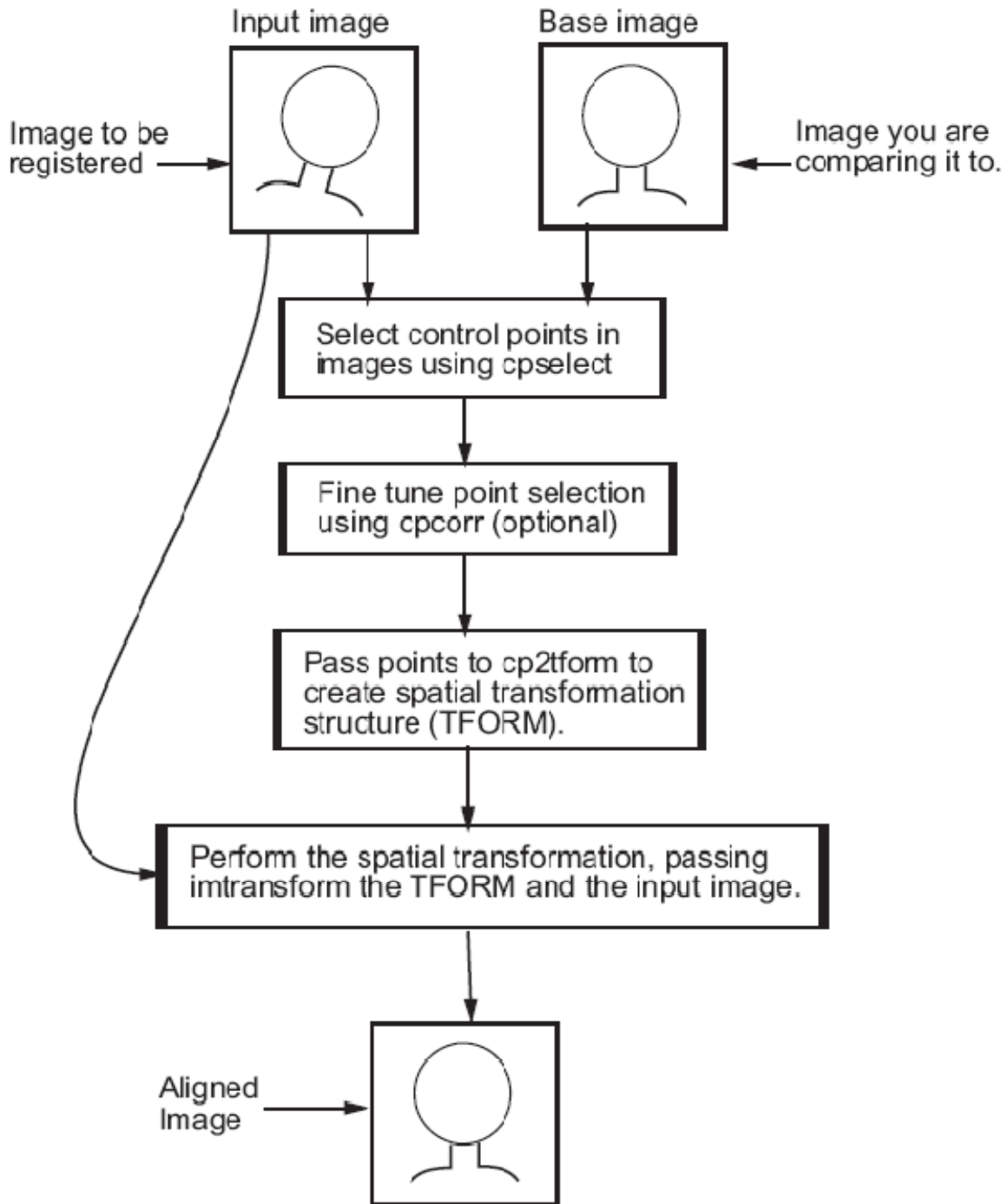
14 Fine Tune

15 Features

16 Landmarks

17 Point Mapping

18 Gross Global Distortions



### مثال: تثبیت تصویر نسبت به تصویر هوایی تصحیح شده

تصاویر عکسبرداری هوایی<sup>۱۹</sup> بدلیل کروی بودن سطح زمین شامل اعوجاجاتی در راستای X و Y می‌باشند. اگر این اعوجاجات از تصاویر عکسبرداری هوایی حذف شوند، تصویر حاصل را تصویر هوایی تصحیح شده<sup>۲۰</sup> می‌نامند. در این مثال بین یک تصویر عکسبرداری هوایی و یک تصویر هوایی تصحیح شده، عمل تثبیت انجام می‌شود.

<sup>19</sup> Aerial Photo Image

<sup>20</sup> Orthophoto Image

## گام ۱: خواندن تصاویر

در این مثال، تصویر تصحیح شده به عنوان تصویر پایه استفاده شده و westconcordorthophoto.png نام دارد که یک تصویر سطح خاکستری بوده و انواع مختلف اعوجاجات در آن حذف شده است. تصویر تثبیت نشده westconcordaerial.png نام داشته و شامل برخی انواع اعوجاجات می‌باشد.

```
orthophoto = imread('westconcordorthophoto.png');
figure, imshow(orthophoto)
unregistered = imread('westconcordaerial.png');
figure, imshow(unregistered)
```



Aerial Photo Image

Image Courtesy of mPower3/Emerge



Orthophoto Image

Image Courtesy of MassGIS

## گام ۲: انتخاب نقاط کنترلی در تصاویر

در اینجا از یک ابزار تعاملی برای انتخاب نقاط کنترلی متناظر در دو تصویر پایه و تثبیت نشده (یا تصویر ورودی) استفاده می‌کنیم. این نقاط کنترلی معمولاً نشانه‌هایی<sup>۲۱</sup> شامل تقاطع جاده‌ها و یا عوارض طبیعی زمین می‌باشند.

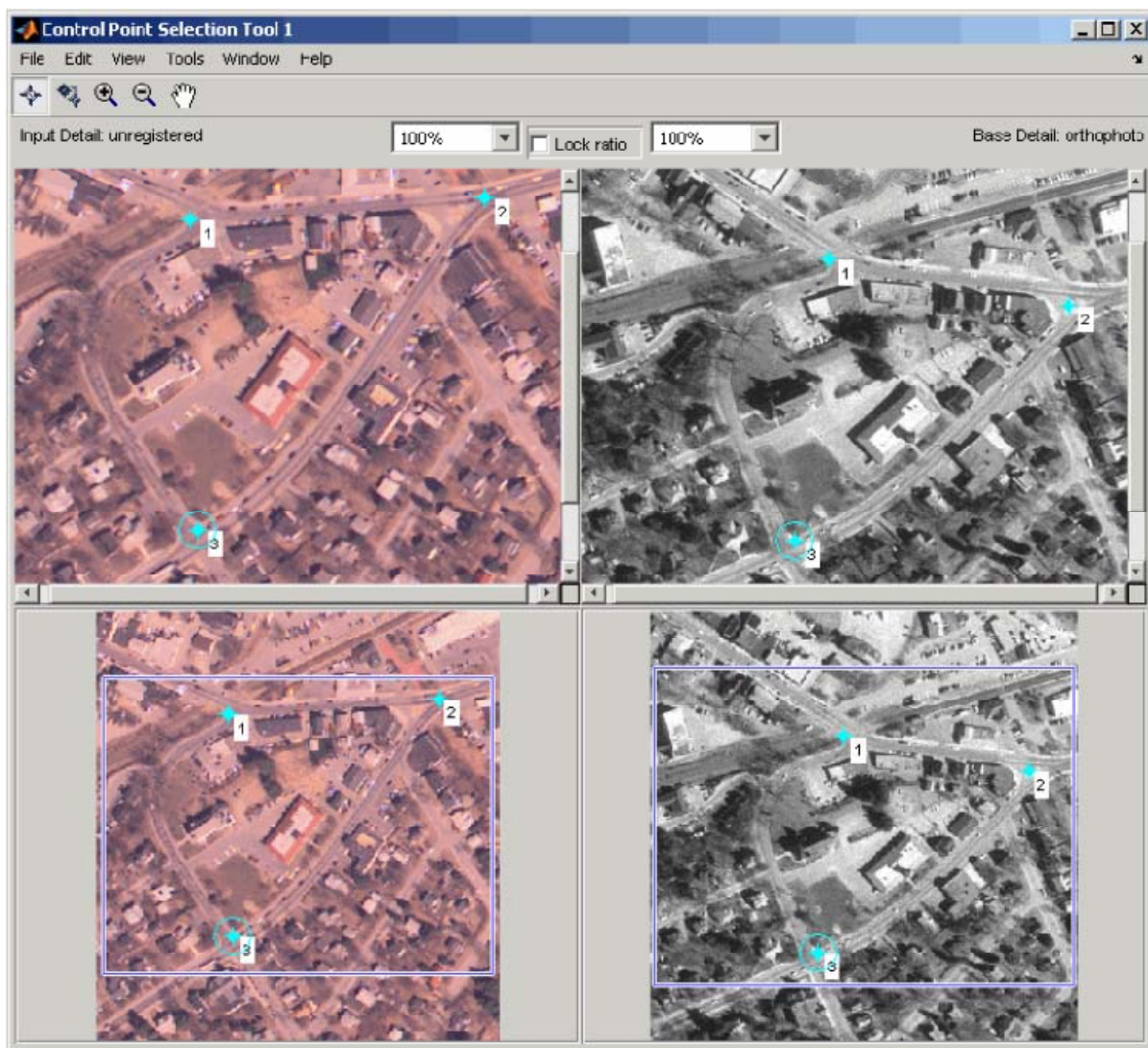
برای شروع این ابزار تعاملی به صورت زیر عمل کنید:

```
cpselect(unregistered, orthophoto)
```

حال در پنجره ظاهر شده جفت نقاط را تعیین می‌کنیم. در شکل زیر سه جفت نقطه مشخص شده است. حداقل تعداد جفت نقاط بستگی به نوع تبدیلی دارد که قصد انجام آن را دارید.

<sup>21</sup> Landmarks





گام ۳: ذخیره‌ی جفت نقاط مشخص شده در فضای کاری متلب

در پنجره‌ای که به کمک آن جفت نقاط را تعیین کرده‌اید، از منوی File گزینه‌ی Export Points to Workspace را انتخاب کنید. برای مثال، سه جفت نقطه‌ی زیر که در مختصات مکانی بیان شده‌اند، مربوط به نقاط انتخاب شده در تصویر ورودی می‌باشند. ستون سمت چپ، مقادیر مختصه‌ی X و ستون سمت راست، مقادیر مختصه‌ی Y می‌باشند:

input\_points =

118.0000	96.0000
304.0000	87.0000
358.0000	281.0000
127.0000	292.0000

گام ۴: اصلاح محل نقاط کنترلی (اختیاری)

ممکن است نقاط کنترلی که در مرحله‌ی قبل انتخاب کرده‌اید، کاملاً دقیق و منطبق بر محل‌های مورد نظرتان نبوده باشد؛ بنابراین، بهتر است محل نقاط مذکور را اصلاح کنید. یک راه انجام اصلاح، استفاده از همبستگی متقابل به کمک دستور `cpcorr` است. البته، یک شرط لازم برای استفاده از این دستور این است که دو تصویر باید نسبت به هم چرخش نداشته و نیز دارای یک مقیاس باشند. در مثالی که در حال بررسی آن هستیم، دو تصویر (یعنی تصویر پایه و تصویر ورودی) نسبت به هم چرخش دارند؛ بنابراین، در اینجا نمی‌توان از دستور `cpcorr` استفاده کرد (بعداً استفاده از آن را خواهیم دید).

#### گام ۵: تعیین نوع تبدیل و به دست آوردن پارامترهای مربوطه

در این مرحله، نقاط کنترلی را به دستور `cp2tform` می‌دهیم تا پارامترهای تبدیل مربوطه را محاسبه کند. البته در این دستور شما باید نوع اصلی تبدیل را نیز تعیین کنید. برای این کار با توجه به اعوجاجی که در تصویر مشاهده می‌کنید باید یکی از پنج نوع تبدیلی که در بخش بعدی (بخش انواع تبدیلات) می‌آید را انتخاب کنید. در ضمن توجه کنید که تصاویر ممکن است بیش از یک نوع اعوجاج داشته باشند. در مثال فعلی، مهمترین اعوجاجی که در تصویر ورودی وجود دارد، اعوجاج پرسپکتیو است؛ بنابراین، ما باید نوع تبدیل `projective` را انتخاب کنیم:

```
mytform = cp2tform(input_points, base_points, 'projective');
```

#### گام ۶: انجام تبدیل روی تصویر ورودی (یا تثبیت نشده)

حال نوبت به انجام تبدیل به کمک دستور `imtransform` است:

```
registered = imtransform(unregistered, mytform);
```

با توجه به مراحل طی کرده بودیم، در این مثال هم می‌توانید تصویر نیمه‌شفافی از تصویر تثبیت شده را روی تصویر پایه قرار داده و هر دو را (به منظور ارزیابی بهتر عمل تثبیت تصویر) با هم نمایش دهیم. نتیجه به صورت شکل زیر است:





### انواع تبدیلات

لیست تبدیلات موجود برای استفاده در دستور `cp2tform` به ترتیب پیچیدگی آنها به صورت زیر است:

- 'nonreflective similarity'
- 'affine'
- 'projective'
- 'polynomial' (Order 2, 3, or 4)
- 'piecewise linear'
- 'lwm'

برای دیدن اطلاعات بیشتر در مورد این تبدیلات و نیز دانستن حداقل تعداد نقاط کنترالی لازم برای هر کدام از آنها به صفحه‌ی مرجع دستورات `cpselect` و `cp2tform` مراجعه کنید.

The first four transformations, 'nonreflective similarity', 'affine', 'projective', and 'polynomial' are global transformations. In these transformations, a single mathematical expression applies to an entire image. The last two transformations, 'piecewise linear' and 'lwm' (local weighted mean), are local transformations. In these transformations, different mathematical expressions apply to different regions within an image. When exploring how different transformations affect the images you are working with, try the global transformations first. If these transformations are not satisfactory, try the local transformations: the piecewise linear transformation first, and then the local weighted mean transformation.

### انتخاب نقاط کنترلی

#### استفاده از همبستگی برای اصلاح نقاط کنترلی

برای اصلاح خودکار نقاط کنترلی که شما قبلاً با چشم خود آنها را انتخاب کرده‌اید، می‌توانید از دستور `cpcorr` استفاده کنید. البته همان طور که قبلاً نیز بیان شد، باید دو شرط مربوط به چرخش و مقیاس برقرار باشند.

```
input_pts_adj= cpcorr(input_points, base_points, input, base);
```

تمرین: با توجه به راهنمای زیر، در مورد نحوه‌ی عملکرد دستور `cpcorr` توضیح دهید.

```
>> help cpcorr
```

CPCORR Tune control point locations using cross-correlation.

INPUT\_POINTS = CPCORR(INPUT\_POINTS\_IN,BASE\_POINTS\_IN,INPUT,BASE) uses normalized cross-correlation to adjust each pair of control points specified in INPUT\_POINTS\_IN and BASE\_POINTS\_IN.

INPUT\_POINTS\_IN must be an M-by-2 double matrix containing the coordinates of control points in the input image. BASE\_POINTS\_IN is an M-by-2 double matrix containing the coordinates of control points in the base image.

CPCORR returns the adjusted control points in INPUT\_POINTS, a double matrix the same size as INPUT\_POINTS\_IN. If CPCORR cannot correlate a pairs of control points, INPUT\_POINTS will contain the same coordinates as INPUT\_POINTS\_IN for that pair.

CPCORR will only move the position of a control point by up to 4 pixels. Adjusted coordinates are accurate up to one tenth of a pixel. CPCORR is designed to get subpixel accuracy from the image content and coarse control point selection.

Note that the INPUT and BASE images must have the same scale for CPCORR to be effective.

CPCORR cannot adjust a point if any of the following occur:

- points are too near the edge of either image
- regions of images around points contain Inf or NaN
- region around a point in input image has zero standard deviation
- regions of images around points are poorly correlated

## فصل هشتم

# طراحی و پیاده‌سازی فیلترهای دوبعدی

هدف : آشنایی با نحوه‌ی طراحی و پیاده‌سازی فیلترهای دوبعدی

فیلتر کردن ابزاری برای اصلاح و یا بهسازی یک تصویر است. برای مثال، شما به کمک فیلتر کردن می‌توانید برخی ویژگیهای مشخص را حذف و یا تقویت کنید. آن دسته از اعمال پردازش تصویر که از طریق فیلتر کردن می‌توان آنها را انجام داد شامل نرم کردن<sup>۲۲</sup>، تیز کردن<sup>۲۳</sup>، و تقویت لبه‌ها<sup>۲۴</sup> است. عمل فیلتر کردن یک عملگر مبتنی بر همسایگی<sup>۲۵</sup> است؛ به این معنا که برای محاسبه‌ی مقدار یک پیکسل خاص  $p$  از تصویر خروجی، به مقادیر پیکسل‌های واقع در یک همسایگی خاص از پیکسل متناظر  $p$  در تصویر ورودی نیاز داریم. یکی از متداولترین انواع فیلترها، فیلتر خطی<sup>۲۶</sup> است؛ به این معنا که مقدار پیکسل  $p$  (در تصویر خروجی) برابر یک ترکیب خطی از مقادیر پیکسل‌های متعلق به همسایگی مذکور است.

### کانولوشن (یا پیچش)

خروجی هر فیلتر خطی از طریق کانولوشن تصویر ورودی با پاسخ ضربه‌ی آن فیلتر محاسبه می‌شود. ماتریس پاسخ ضربه در حکم ماتریس وزن محسوب شده و به نامهای مختلف از جمله کرنل کانولوشن<sup>۲۷</sup> و فیلتر نیز نامیده می‌شود. کرنل کانولوشن مانند کرنل همبستگی<sup>۲۸</sup> است با این تفاوت که باید به اندازه‌ی ۱۸۰ درجه چرخانده شود. کرنل همبستگی ابزاری ریاضی برای محاسبه‌ی میزان شباهت دو سیگنال یا دو ماتریس مختلف می‌باشد.

نحوه‌ی انجام فیلتر کردن در حوزه‌ی مکان (و یا زمان) این گونه است که پاسخ ضربه‌ی فیلتر طراحی شده را به اندازه‌ی ۱۸۰ درجه چرخانده و مرکز آن را روی نقاط مختلف تصویر قرار می‌دهیم. حال، مجموع حاصلضربهای دودویی عناصر متناظر از تصویر و پاسخ ضربه را محاسبه کرده و به عنوان خروجی فیلتر ثبت می‌کنیم. برای مثال، اگر فرض کنیم تصویر ورودی و پاسخ ضربه ماتریسهایی به ترتیب به صورت زیر باشند:

$$A = \begin{bmatrix} 17 & 24 & 1 & 8 & 15 \\ 23 & 5 & 7 & 14 & 16 \\ 4 & 6 & 13 & 20 & 22 \\ 10 & 12 & 19 & 21 & 3 \\ 11 & 18 & 25 & 2 & 9 \end{bmatrix}$$

$$h = \begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix}$$

آن گاه برای مثال، برای محاسبه‌ی پیکسل واقع در مختصات (2,4) از تصویر خروجی (یعنی تصویر فیلتر شده)، باید مراحل زیر را انجام داد:

(۱) ماتریس پاسخ ضربه را ۱۸۰ درجه بچرخانید.

(۲) مرکز ماتریس پاسخ ضربه را روی پیکسل تصویر واقع در مختصات (2,4) قرار دهید.

(۳) مجموع حاصلضربهای عناصر دودویی مشخص شده در شکل و رابطه‌ی زیر را محاسبه کنید:

$$1 \times 2 + 8 \times 9 + 15 \times 4 + 7 \times 7 + 14 \times 5 + 16 \times 3 + 13 \times 6 + 20 \times 1 + 22 \times 8 = 575$$

22 Smoothing

23 Sharpening

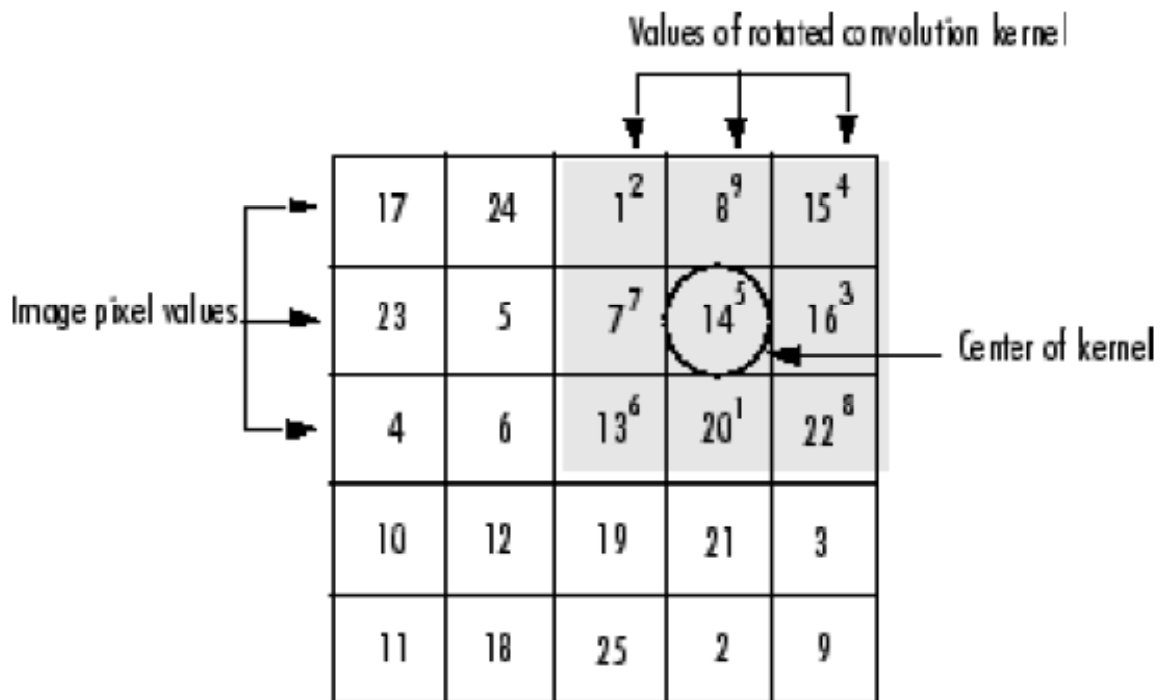
24 Edge Enhancement

25 Neighborhood Operation

26 Linear Filtering

27 Convolution Kernel

28 Correlation Kernel

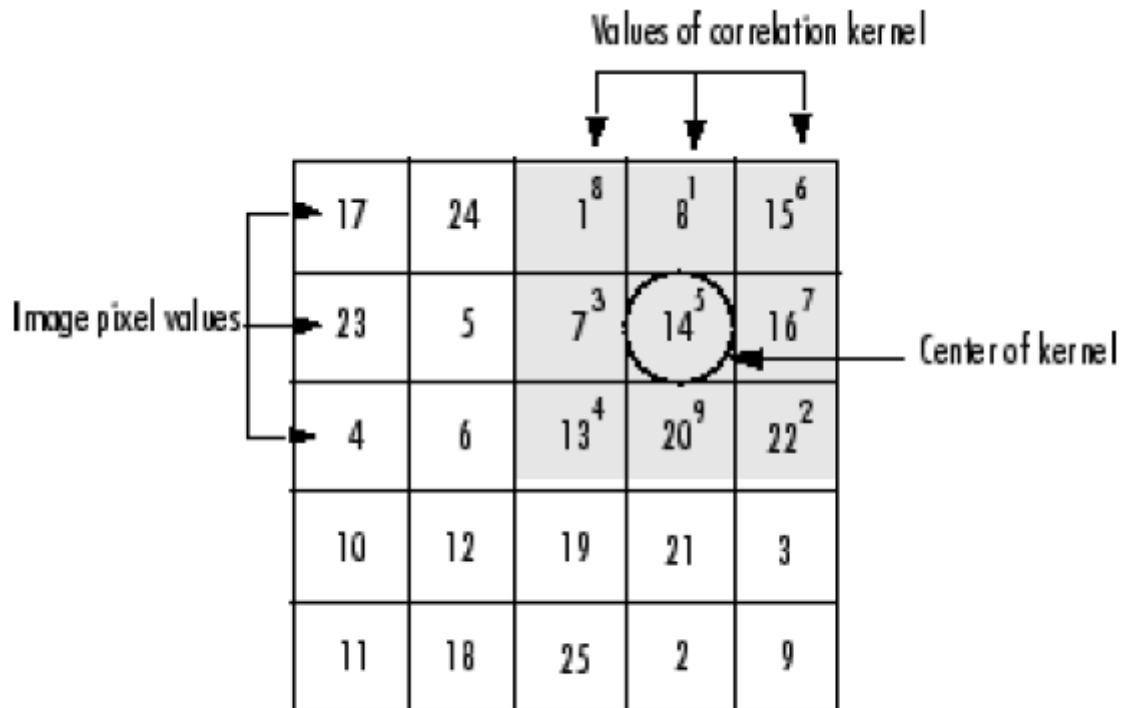


## Computing the (2,4) Output of Convolution

توجه کنید که برای محاسبه‌ی مقدار تصویر خروجی در لبه‌ها، از آنجا که در فرآیند همپوشانی و محاسبه‌ی مجموع حاصلضربها نظیری برای عناصر لبه‌ای ماتریس پاسخ ضربه در تصویر اصلی وجود ندارد، می‌توانیم این عناصر را صفر فرض کنیم. به این کار گستراندن با صفر<sup>۲۹</sup> می‌گویند.

### همبستگی

نحوه‌ی محاسبه‌ی همبستگی مانند کانولوشن است با این تفاوت که در محاسبات آن دیگر لازم نیست کرنل همبستگی را بچرخانیم. بنابراین، اگر مثال ارائه شده در بخش قبل (برای کانولوشن) را بخواهیم برای همبستگی مجدداً در نظر بگیریم، شکل اخیر به صورت زیر اصلاح می‌شود:



## Computing the (2,4) Output of Correlation

### فیلتر کردن تصاویر به کمک دستور `imfilter`

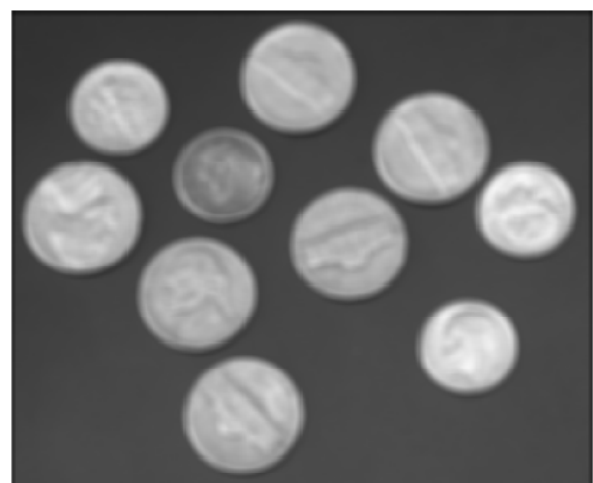
برای انجام فیلتر کردن (چه کرنل کانولوشن و چه کرنل همبستگی) می‌توان از دستور `imfilter` استفاده کرد. برای مثال، اگر بخواهیم یک فیلتر متوسط‌گیر را پیاده‌سازی کنیم، به صورت زیر می‌توان عمل کرد:

```
I = imread('coins.png');
h = ones(5,5) / 25;
I2 = imfilter(I,h);
imshow(I), title('Original Image');
figure, imshow(I2), title('Filtered Image')
```

نتیجه:



Original Image



Filtered Image

### انواع داده‌ای

نوع داده‌ای تصویر خروجی همان نوع داده‌ای تصویر ورودی است. دستور `imfilter` محاسبات خود را به صورت ممیز شناور و با دقت مضاعف انجام می‌دهد اما حاصل نهایی را با توجه به نوع تصویر خروجی و محدوده‌ی آن، بُرش می‌زند و یا این که گرد می‌کند. به دلیل این عمل بُرش، ممکن است شما بخواهید ابتدا نوع تصویر را تغییر دهید تا در نتیجه‌ی فیلتر، اطلاعات کمتری را از دست دهید. برای مثال در برنامه‌ی زیر تصویر ورودی از نوع `double` است به همین دلیل خروجی دستور `imfilter` مقادیر منفی را به خود می‌گیرد:

```
>>A = magic(5)
A =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
>>h = [-1 0 1]
h =
    -1     0     1
>>imfilter(A,h)
ans =
    24   -16   -16    14    -8
     5   -16     9     9   -14
     6     9    14     9   -20
    12     9     9   -16   -21
    18    14   -16   -16    -2
```

اما اگر تصویر ورودی را از نوع `uint8` تعریف کنیم، به دلیل عمل بُرش، خروجی فیلتر مقادیر منفی نخواهد داشت:

```
>>A = uint8(magic(5));
>>imfilter(A,h)
ans =
    24     0     0    14     0
     5     0     9     9     0
     6     9    14     9     0
    12     9     9     0     0
    18    14     0     0     0
```

ملاحظه می‌کنید که مقادیر منفی به صفر بُرش زده می‌شوند.

### گزینه‌های کانولوشن و همبستگی

دستور `imfilter` قادر به فیلتر کردن از هر دو طریق کانولوشن و همبستگی است. به طور پیش فرض عمل همبستگی انجام خواهد شد. اگر مایل به انجام کانولوشن هستید، آرگومان رشته‌ای `'conv'` را به دستور فوق اضافه کنید. برای مثال:



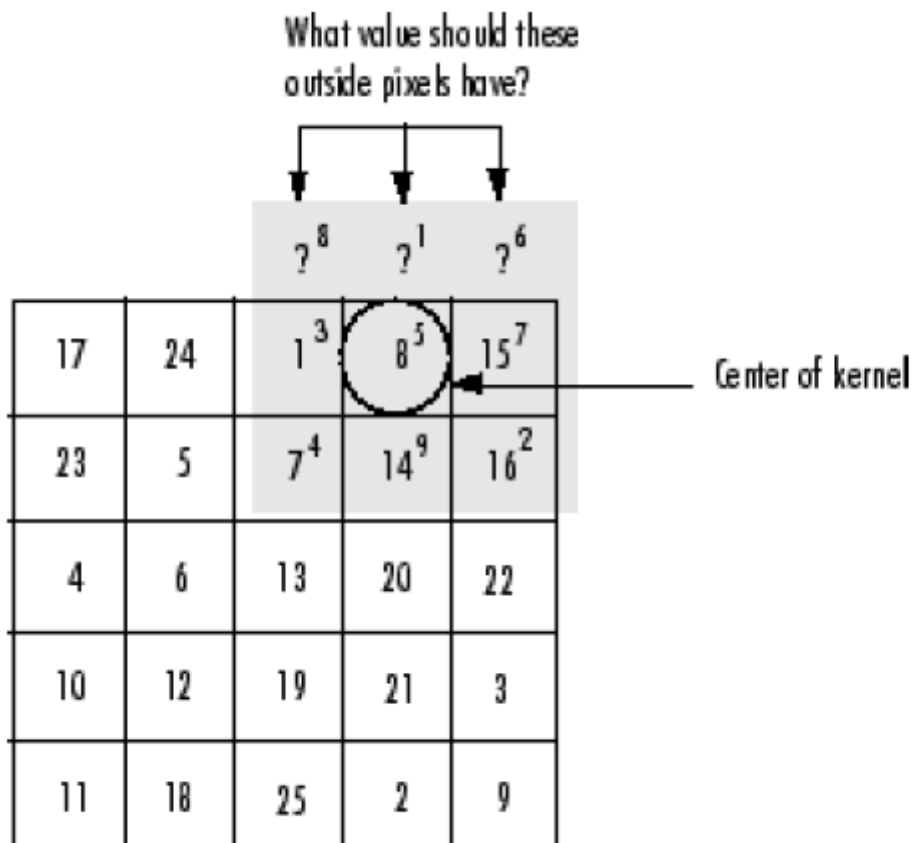
```

>>A = magic(5);
>>h = [-1 0 1]
>>imfilter(A,h) % filter using correlation
ans =
    24 -16 -16 14 -8
     5 -16  9  9 -14
     6  9 14  9 -20
    12  9  9 -16 -21
    18 14 -16 -16 -2
>>imfilter(A,h,'conv') % filter using convolution
ans =
   -24 16 16 -14 8
    -5 16 -9 -9 14
    -6 -9 -14 -9 20
   -12 -9 -9 16 21
   -18 -14 16 16 2

```

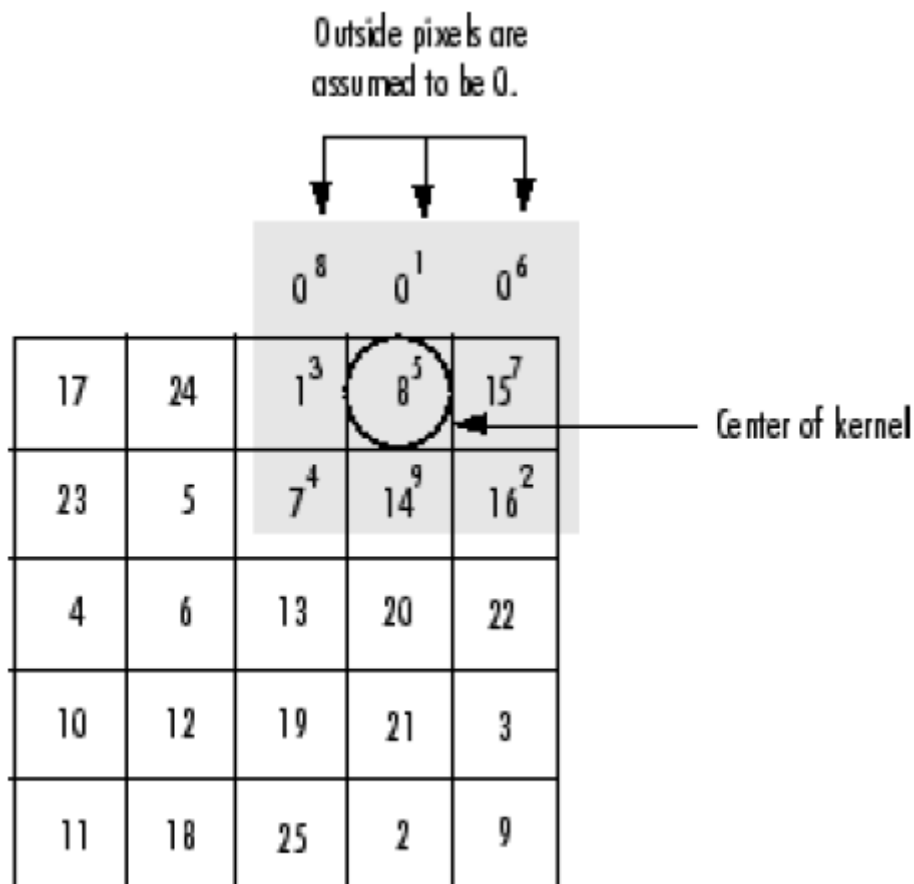
### گزینه‌های گستراندن مرز<sup>۳۰</sup>

همان طور که قبلاً نیز اشاره شد، برای محاسبه‌ی مقدار تصویر خروجی در لبه‌ها، در فرآیند همپوشانی و محاسبه‌ی مجموع حاصلضربها نظیری برای عناصر لبه‌ای ماتریس پاسخ ضربه در تصویر اصلی وجود ندارد. شکل زیر این مطلب را نشان می‌دهد:



### When the Values of the Kernel Fall Outside the Image

دستور `imfilter`، مقادیر مذکور در تصویر ورودی را برابر صفر در نظر می‌گیرد (گستراندن با صفر):



### Zero Padding of Outside Pixels

اگر از شیوه‌ی گستراندن با صفر برای فیلتر کردن استفاده کنیم، در تصویر خروجی ممکن است یک نوار سیاه رنگ مشاهده شود. این پدیده در مثال زیر نشان داده شده است:

```
I = imread('eight.tif');
h = ones(5,5) / 25;
I2 = imfilter(I,h);
imshow(I), title('Original Image');
figure, imshow(I2), title('Filtered Image with Black Border')
```

نتیجه:

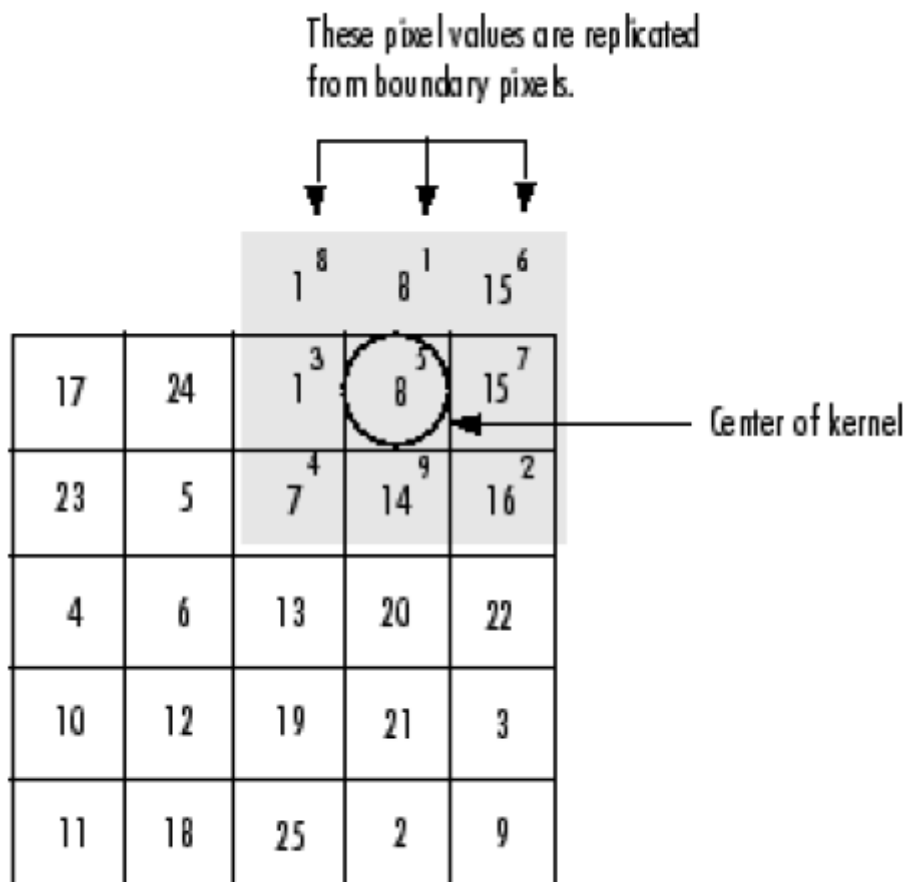


Original Image



Filtered Image with Black Border

اگر پدیده‌ی فوق برایتان نامطلوب است، می‌توانید از گزینه‌ی دیگری که در دستور `imfilter` موجود است، استفاده کنید. این گزینه، تکثیر مرز<sup>۳۱</sup> نام داشته و در حقیقت مقدار پیکسل‌های خارج از مرز را برابر مقدار نزدیکترین پیکسل داخل مرز در نظر می‌گیرد. این عمل در شکل زیر نشان داده شده است:



Replicated Boundary Pixels

<sup>31</sup> Border Replication

برای استفاده از چنین قابلیت‌هایی باید از آرگومان رشته‌ای 'replicate' استفاده کنید:

```
I3 = imfilter(I,h,'replicate');
figure, imshow(I3);
title('Filtered Image with Border Replication')
```

نتیجه:



**Filtered Image with Border Replication**

علاوه بر دو گزینه‌ی فوق‌الذکر، چند گزینه‌ی دیگر نیز برای گستراندن مرز وجود دارد از جمله 'circular' و 'symmetric'. برای توضیحات بیشتر به صفحه‌ی مرجع دستور imfilter مراجعه کنید.

### فیلتر کردن چندبعدی

برای فیلتر کردن یک تصویر سه بعدی به کمک یک فیلتر دوبعدی کافی است فیلتر مذکور را به هر یک از سه صفحه‌ی دو بعدی تشکیل دهنده‌ی تصویر اعمال کنیم. به مثال زیر توجه کنید:

۱- خواندن و نمایش تصویر رنگی ورودی

```
rgb = imread('peppers.png');
imshow(rgb);
```



۲- فیلتر کردن تصویر و نمایش نتیجه

```
h = ones(5,5)/25;  
rgb2 = imfilter(rgb,h)  
figure,imshow(rgb2)
```

نتیجه:



### دیگر توابع فیلتر کردن

تابع `filter2` برای انجام همبستگی دوبعدی، تابع `conv2` برای انجام همبستگی دوبعدی، و تابع `convn` برای انجام همبستگی چندبعدی استفاده می‌شوند. همه‌ی این توابع، ابتدا تصویر ورودی خود را به نوع `double` تبدیل می‌کنند و تصویر خروجی نیز از همین نوع است. همچنین، این توابع فقط از گستراندن با صفر پشتیبانی می‌کنند. در مقابل، دستور `imread` تصویر ورودی را به نوع `double` تبدیل نمی‌کند. همچنین، این دستور انواع دیگری از گستراندن را نیز پشتیبانی می‌کند.

### فیلتر کردن یک تصویر با انواع فیلترهای از قبل مشخص

از دستور `fspecial` می‌توان برای تولید انواع مختلفی از فیلترها در قالب کرنل کانولوشن استفاده کرد. بعد از ایجاد کرنل کانولوشن، از دستور `imfilter` برای انجام فیلتر استفاده کنید. در مثال زیر، فیلتر مخصوصی به نام فیلتر `unsharp masking` تولید و به تصویر ورودی اعمال می‌شود. اثر این فیلتر این است که اثر لبه‌ها و جزئیات تصویر را مشهودتر و برجسته‌تر می‌کند.

```
I = imread('moon.tif');
h = fspecial('unsharp');
I2 = imfilter(I,h);
imshow(I), title('Original Image')
figure, imshow(I2), title('Filtered Image')
```

نتیجه:



Image Courtesy of Michael Myers

Original Image



Filtered Image

### طراحی فیلترهای خطی در حوزه فرکانسی

تمام فیلترهای خطی طراحی شده به کمک توابع جعبه ابزار پردازش تصویر متلب از نوع FIR می‌باشند. دلیل اینکه در پردازش تصویر فیلترهای FIR بیشتر از فیلترهای IIR مورد استفاده قرار می‌گیرند، این است که:

- ۱- فیلترهای FIR را به راحتی می‌توان به صورت ماتریس‌هایی از ضرایب نمایش داد.
  - ۲- فیلترهای FIR دوبعدی، به سادگی توسعه‌یافته‌هایی از نوع یک بعدی خود هستند.
  - ۳- چندین روش مطمئن برای طراحی فیلترهای FIR وجود دارد.
  - ۴- پیاده‌سازی فیلترهای FIR راحت و آسان است.
  - ۵- فیلترهای FIR را می‌توان طوری طراحی کرد که دارای فاز خطی شوند. داشتن چنین قابلیتی باعث جلوگیری از تولید اعوجاج می‌شود.
  - ۶- فیلترهای FIR ذاتاً پایدار می‌باشند.
- در مقابل، فیلترهای IIR دارای پایداری ذاتی نبوده و در نتیجه، طراحی و پیاده‌سازی آنها مشکل است.



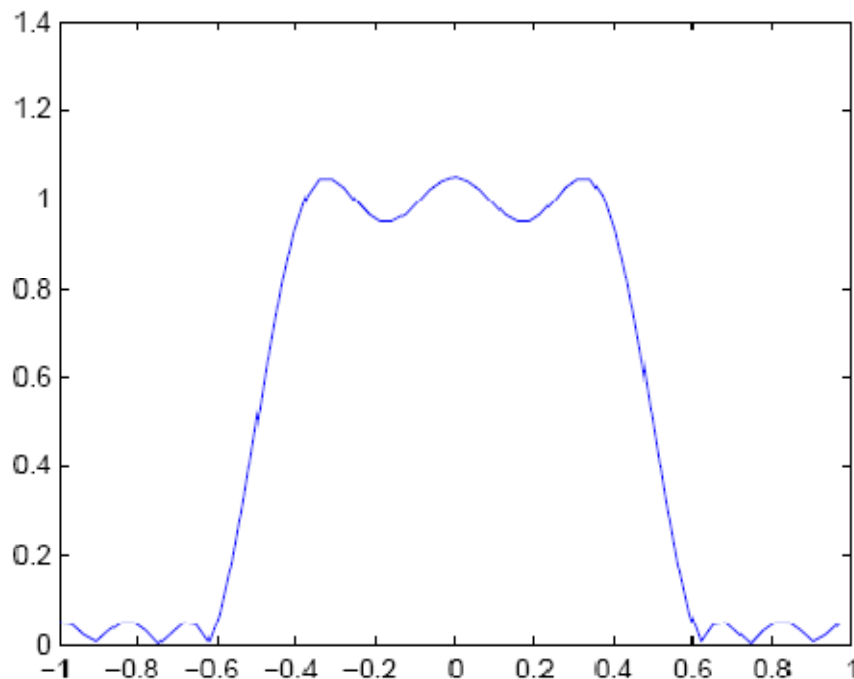
### روش تبدیل فرکانسی برای طراحی فیلتر

در روش تبدیل فرکانسی<sup>۳۲</sup>، ابتدا یک نمونه‌ی یک بعدی از فیلتر FIR طراحی شده و سپس معادل دوبعدی آن ساخته می‌شود. نمونه‌ی دوبعدی تولید شده، اکثر ویژگی‌های نمونه‌ی یک بعدی (به ویژه باند گذر<sup>۳۳</sup> و ویژگی‌های ریبِل<sup>۳۴</sup>) را حفظ می‌کند. در این روش از ماتریس تبدیل برای ساختن نمونه‌ی دوبعدی استفاده می‌شود.

دستور `ftrans2` وظیفه‌ی انجام تبدیل فرکانسی را بر عهده دارد. به طور پیش فرض، این دستور از ماتریس تبدیلی استفاده می‌کند که موجب تقارن استوانه‌ای نمونه‌ی دوبعدی می‌شود. البته، شما می‌توانید با تعریف ماتریس تبدیل مورد نظرتان، نوع تقارن را تعیین کنید.

در مثال زیر، ابتدا (به کمک دستور `remez`) یک فیلتر یک بعدی بهینه‌ی دارای ریبِل‌های مساوی<sup>۳۵</sup> طراحی شده و سپس، نمونه‌ی دوبعدی آن (به کمک دستور `ftrans2`) ساخته می‌شود.

```
b = remez(10,[0 0.4 0.6 1],[1 1 0 0]);
h = ftrans2(b);
[H,w] = freqz(b,1,64,'whole');
colormap(jet(64))
plot(w/pi-1,fftshift(abs(H)))
figure, freqz2(h,[32 32])
```



**One-Dimensional Frequency Response**

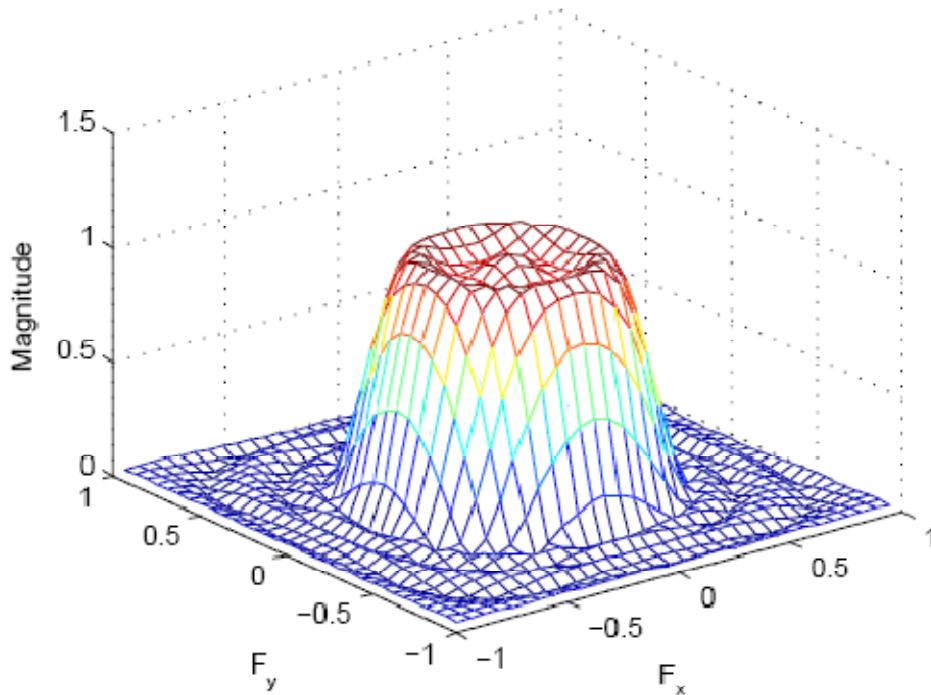
32 Frequency Transformation Method

33 Transition Band

34 Ripple Characteristics

35 Equiripple





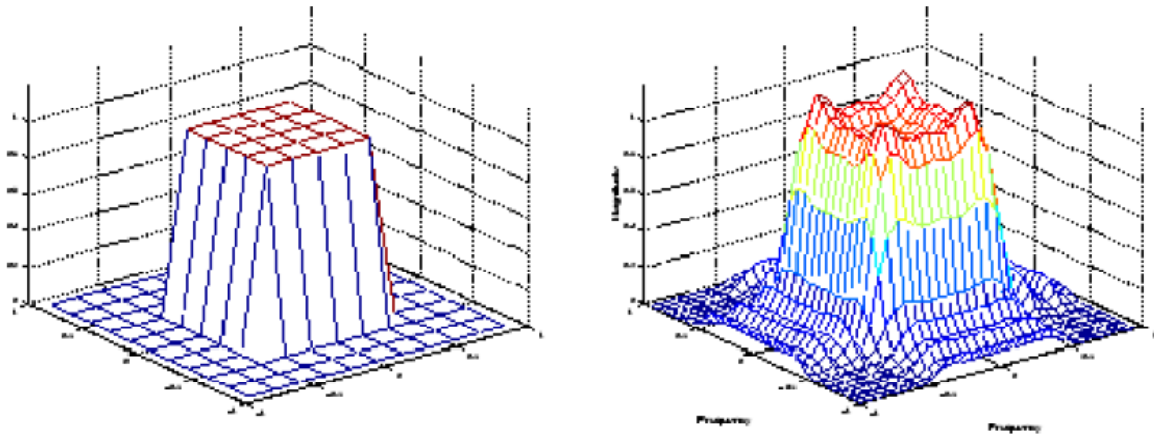
**Corresponding Two-Dimensional Frequency Response**

### روش نمونه برداری فرکانسی برای طراحی فیلتر

در این روش از روی فیلتر ایده آل و مطلوب مورد نظر ما، یک فیلتر دوبعدی طراحی و ساخته می شود. در این روش، ما باید ماتریسی از مقادیر فیلتر مطلوب (در نقاط مشخصی از محور فرکانسی) را تهیه کنیم، حال به کمک توابع متلب، فیلتری طراحی می شود پاسخ فرکانسی آن از نقاط مشخص شده بگذرد (در حقیقت، یک نوع مساله‌ی درونیابی است). معمولاً هیچ گونه محدودیتی در مورد نحوه‌ی تغییرات پاسخ فرکانسی در فواصل بین نقاط تعیین شده (در محور فرکانسی) اعمال نمی شود. در این فواصل، پاسخ مذکور معمولاً رفتاری ریپل گونه دارد. منظور از ریپل، تغییرات نوسانی حول یک مقدار ثابت است.

از تابع `fsamp2` برای طراحی فیلتر به روش نمونه برداری فرکانسی استفاده می شود. ورودی این تابع، یک ماتریس مانند `Hd` است که شامل نقاط فرکانسی و مقادیر مطلوب فیلتر در این نقاط می باشد. خروجی این تابع نیز فیلتری مانند `h` است که پاسخ فرکانسی آن از نقاط مذکور عبور می کند. برنامه‌ی زیر یک فیلتر  $11 \times 11$  طراحی کرده و سپس پاس خفرکانسی را رسم می کند. تابع `fsamp2` فیلتر را طراحی کرده و تابع `freqz2` نیز مقادیر فیلتر دوبعدی مذکور را محاسبه می کند.

```
Hd = zeros(11,11); Hd(4:8,4:8) = 1;
[f1,f2] = freqspace(11,'meshgrid');
mesh(f1,f2,Hd), axis([-1 1 -1 1 0 1.2]), colormap(jet(64))
h = fsamp2(Hd);
figure, freqz2(h,[32 32]), axis([-1 1 -1 1 0 1.2])
```



**Desired Two-Dimensional Frequency Response (left) and Actual Two-Dimensional Frequency Response (right)**

در پاسخ فرکانسی فیلتر واقعی (یا همان فیلتر طراحی شده) به وجود ریب‌ها توجه کنید. این ریب‌ها یکی از مشکلات اساسی روش طراحی نمونه‌برداری فرکانسی می‌باشند. این ریب‌ها هر جا که در پاسخ ایده‌آل گذری تیز وجود داشته باشد، رخ می‌دهند. اگر بخواهید محدوده‌ی این ریب‌ها را کاهش دهید باید از فیلتر بزرگتر (یا طولانی‌تری) استفاده کنید اما در این صورت دو مشکل وجود خواهد داشت: اول این که ارتفاع این ریب‌ها کاهش نخواهد یافت، و دوم این که حجم محاسباتی فیلتر افزایش خواهد یافت. اگر بخواهید که تقریب نرم‌تری از پاسخ ایده‌آل به دست آورید، باید از یکی از دو روش تبدیل فرکانسی (قبلاً مطرح شد) و یا روش پنجره‌گیری (در ادامه بررسی می‌شود) استفاده کنید.

### روش پنجره‌گیری

در روش پنجره‌گیری<sup>۳۶</sup>، پاسخ ضربه‌ی فیلتر ایده‌آل در یک پنجره‌ی مشخص ضرب می‌شود. فیلتری که ب‌ه‌این صورت به دست می‌آید معمولاً بهتر از روش نمونه‌برداری فرکانسی است. در جعبه‌ابزار پردازش تصویر متلب، از یکی از دو تابع `fwind1` و `fwind2` برای طراحی به روش پنجره‌گیری استفاده می‌شود. در تابع `fwind1` پنجره‌ی دو بعدی لازم برای طراحی فیلتر، از روی یک یا دو پنجره‌ی یک بعدی (که شما آن را مشخص می‌کنید) ساخته می‌شود؛ اما در تابع `fwind2` پنجره‌ی دو بعدی باید مستقیماً در اختیار این تابع قرار داده شود.

تابع `fwind1` از دو روش مختلف برای تولید پنجره‌ی دو بعدی استفاده می‌کند:

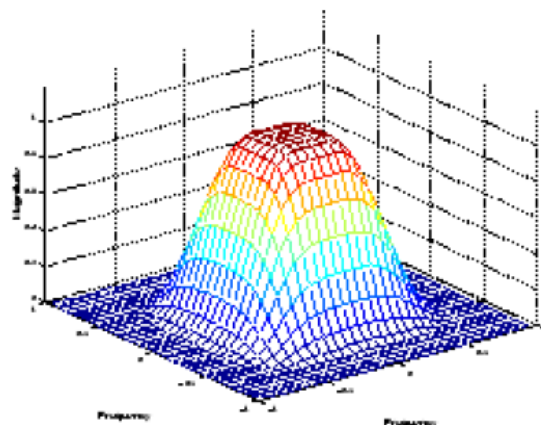
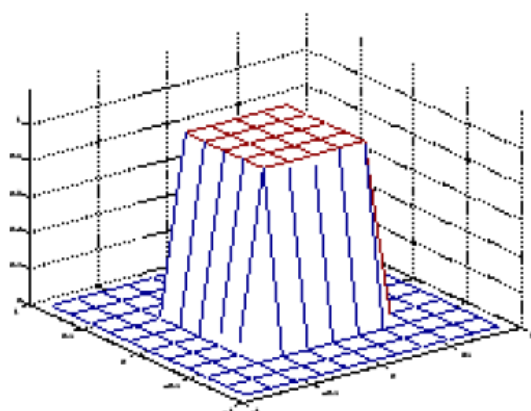
- ۱- تبدیل یک پنجره‌ی یک بعدی برای تولید یک پنجره‌ی دو بعدی که تقریباً به صورت استوانه‌ای متقارن است. این تبدیل تقریباً شبیه به عمل چرخش<sup>۳۷</sup> است.
- ۲- تولید یک پنجره‌ی مستطیلی جدایی‌پذیر از روی دو پنجره‌ی یک بعدی و به کمک ضرب خارجی آنها.

<sup>36</sup> Windowing Method

<sup>37</sup> Rotation

مثال زیر از تابع `fwind1` برای تولید یک پنجره  $11 \times 11$  از روی پاسخ فرکانسی مطلوب `Hd` استفاده می‌کند. در این مثال، از تابع `hamming` برای مشخص کردن پنجره یک بعدی همینگ (و البته ساخت نمونه‌ی دو بعدی آن) استفاده شده است.

```
Hd = zeros(11,11); Hd(4:8,4:8) = 1;
[f1,f2] = freqspace(11,'meshgrid');
mesh(f1,f2,Hd), axis([-1 1 -1 1 0 1.2]), colormap(jet(64))
h = fwind1(Hd,hamming(11));
figure, freqz2(h,[32 32]), axis([-1 1 -1 1 0 1.2])
```



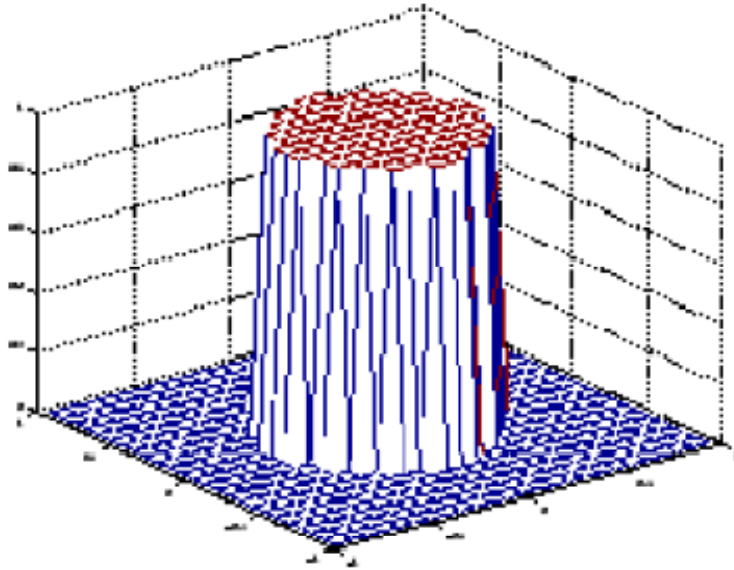
**Desired Two-Dimensional Frequency Response (left) and Actual Two-Dimensional Frequency Response (right)**

### تولید ماتریس پاسخ فرکانسی مطلوب

توابع `fwind1`، `fsamp2` و `fwind2` همگی نیاز به یک ماتریس اندازه‌ی پاسخ فرکانسی مطلوب دارند. برای تولید چنین ماتریسی می‌توانید از تابع `freqspace` استفاده کنید. این تابع نقاط فرکانسی به تعداد مشخص شده (توسط شما) را تولید می‌کند. این نقاط به طور متساوی‌الفاصله نسبت به یکدیگر قرار گرفته‌اند. برای مثال، اگر بخواهید یک فیلتر پایین‌گذر ایده‌آل با فرکانس قطع 0.5 و با تقارن استوانه‌ای را مشخص کنید، می‌توانید به صورت زیر عمل کنید:

```
[f1,f2] = freqspace(25,'meshgrid');
Hd = zeros(25,25); d = sqrt(f1.^2 + f2.^2) < 0.5;
Hd(d) = 1;
mesh(f1,f2,Hd)
```

نتیجه:



## Ideal Circular Lowpass Frequency Response

### محاسبه‌ی پاسخ فرکانسی یک فیلتر

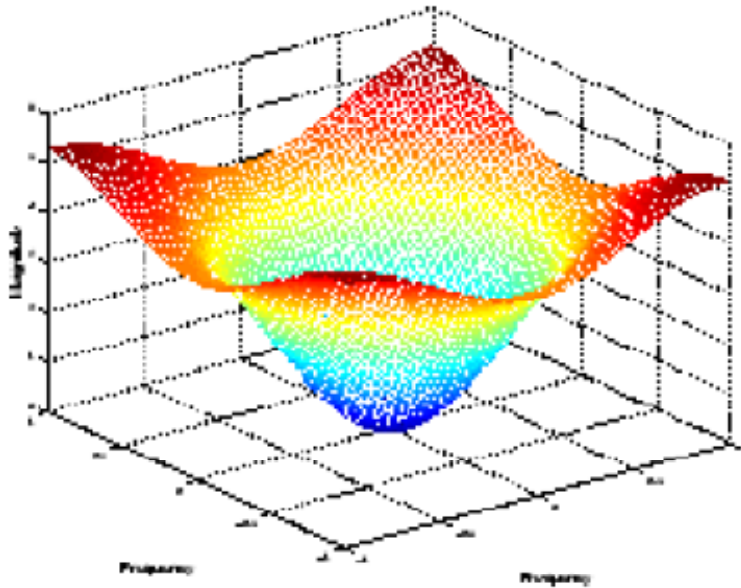
از تابع `freqz2` می‌توان برای محاسبه‌ی پاسخ فرکانسی یک فیلتر دو بعدی استفاده کرد. اگر برای این تابع از هیچ آرگومان خروجی استفاده نکنید، تابع `freqz2` پاسخ فرکانسی را به صورت یک صفحه‌ی مش رسم می‌کند. برای مثال، اگر فیلتر زیر را در نظر بگیرید:

```
h = [ 0.1667    0.6667    0.1667
      0.6667   -3.3333    0.6667
      0.1667    0.6667    0.1667];
```

آن گاه دستور زیر، یک منحنی  $64 \times 64$  نقطه‌ای از پاسخ فرکانسی را رسم می‌کند.

```
freqz2(h)
```

نتیجه:



## Frequency Response of a Two-Dimensional Filter

اگر بخواهید ماتریس پاسخ فرکانسی و نقاط فرکانسی مربوطه را داشته باشید، از آرگومانهای خروجی برای این تابع استفاده کنید:

$[H,f1,f2] = \text{freqz2}(h);$

در دستور فوق، مقادیر فرکانسهای  $f1$  و  $f2$  به صورت نرمالیزه شده برگردانده می‌شوند، طوری که عدد 1.0 متناظر با نصف فرکانس نمونه‌برداری (و یا همان  $\pi$  رادیان) است.