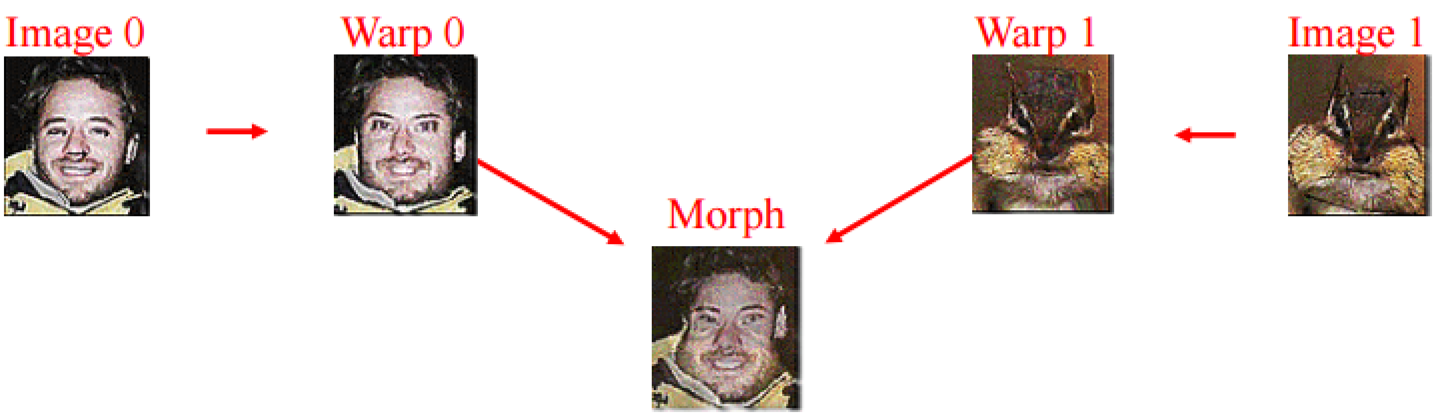
Image Morphing

A combination of generalized image warping with a cross-dissolve between pixels

Morphing involves two steps:

- Pre-warp the two images
- Cross-dissolve their colors



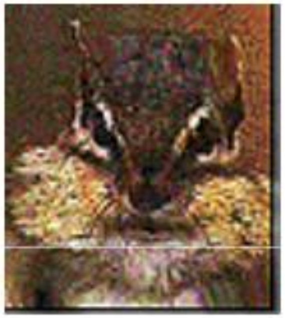
Cross-Dissolving Two Images

Cross-dissolve

A weighted combination of two images, pixel-by-pixel

Image 0

Image 1



Combination controlled by a single interpolation parameter t:

$$\text{Dest} = t \cdot (\text{Image1}) + (1-t)(\text{Image2})$$

Image Pre-Warping

Why warp first?

In order to align features that appear in both images (e.g., eyes, mouth, hair, etc). Without such an alignment, we would get a "double-image" effect!!

Image pre-warping

Re-position all pixels in the source images to avoid the "double-image" effect as much as possible

Pre-warping implemented using the Field Morphing Algorithm

Image Morphing

Both morphing steps specified by same parameter t

- Warp the two images according to t
- Cross-dissolve their colors according to t

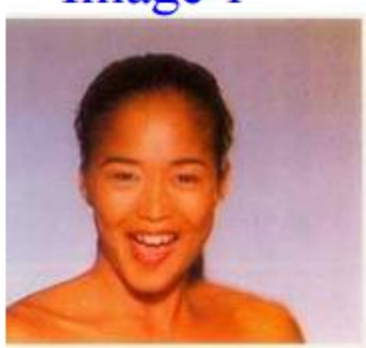
Morphing videos generated by creating a sequence of images, defined by a sequence of t-values (e.g., 0,0.1,0.2,…,0.9,1)

$(1-t) * \text{Warp0} + t * \text{Warp1}$

Morphing Example

Image 0

Image 1

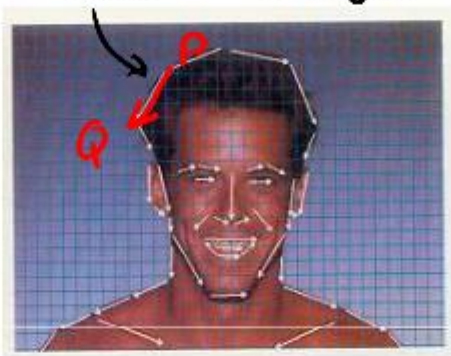


Intermediate Images

Beier-Neely Field Morphing Algorithm (1992)

Warped image computed using Field Morphing Algorithm

line in Image0



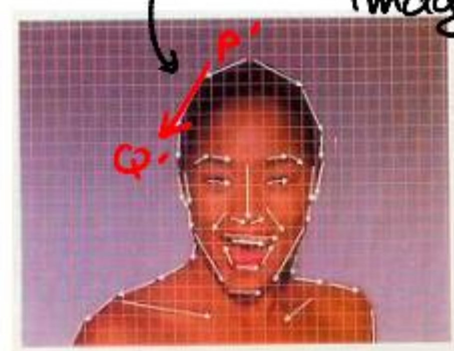
line in Intermediate image

$P(t) = (1-t)P + t \cdot P'$

$Q(t) = (1-t)Q + tQ'$

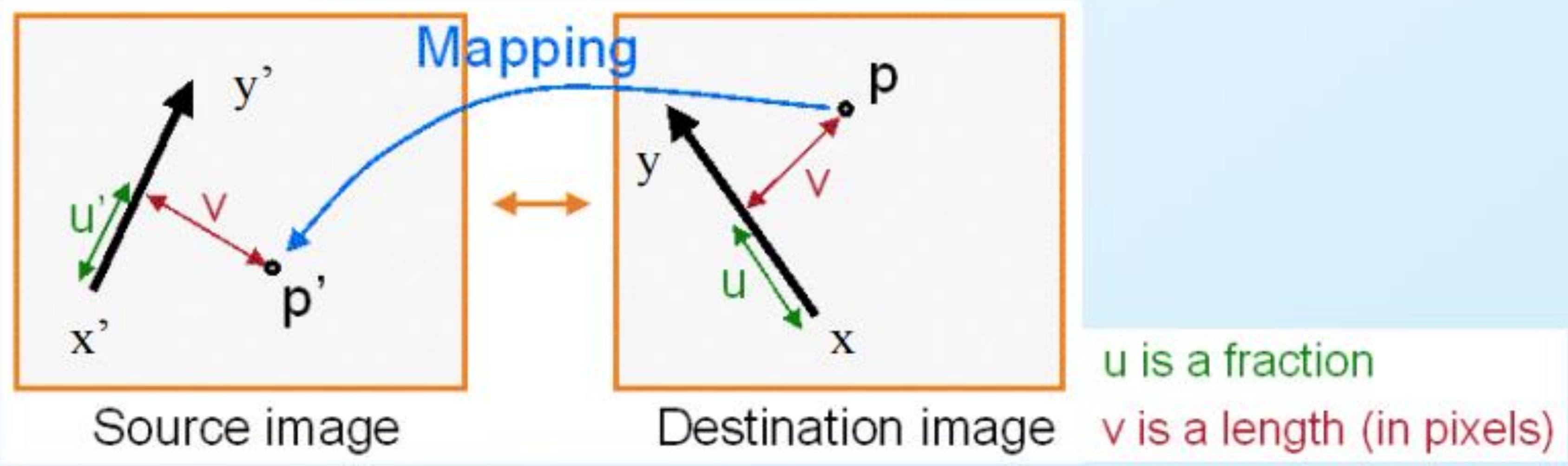
Image warp specified by interactively drawing lines in the two source images

corresponding line in Image 1



Feature-Based Warping: Beier-Neeley

- Beier & Neeley use pairs of lines to specify warp
  - Given **p** in destination image, where is **p'** in source image?



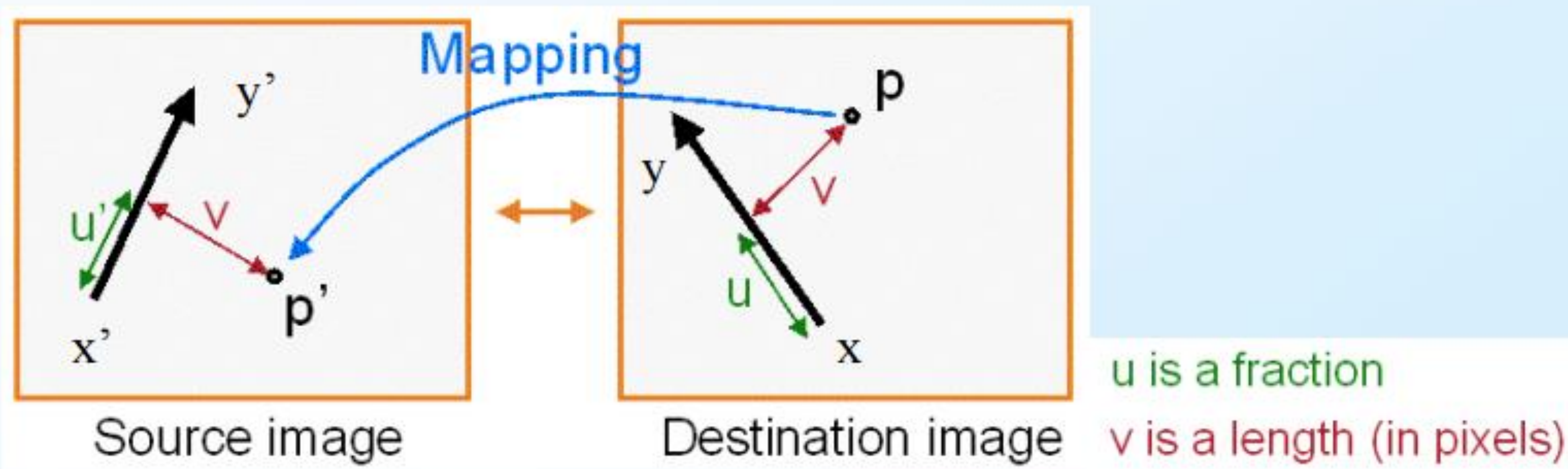
From: https://www.doc.ic.ac.uk/~dfg/graphics/graphics2008/Warping-Morphing.pdf

Feature-Based Warping: Beier-Neeley

$$u = \frac{(p-x)\cdot(y-x)}{\|y-x\|^2} \qquad v = \frac{(p-x)\cdot Perpendicular(y-x)}{\|y-x\|}$$

Correction

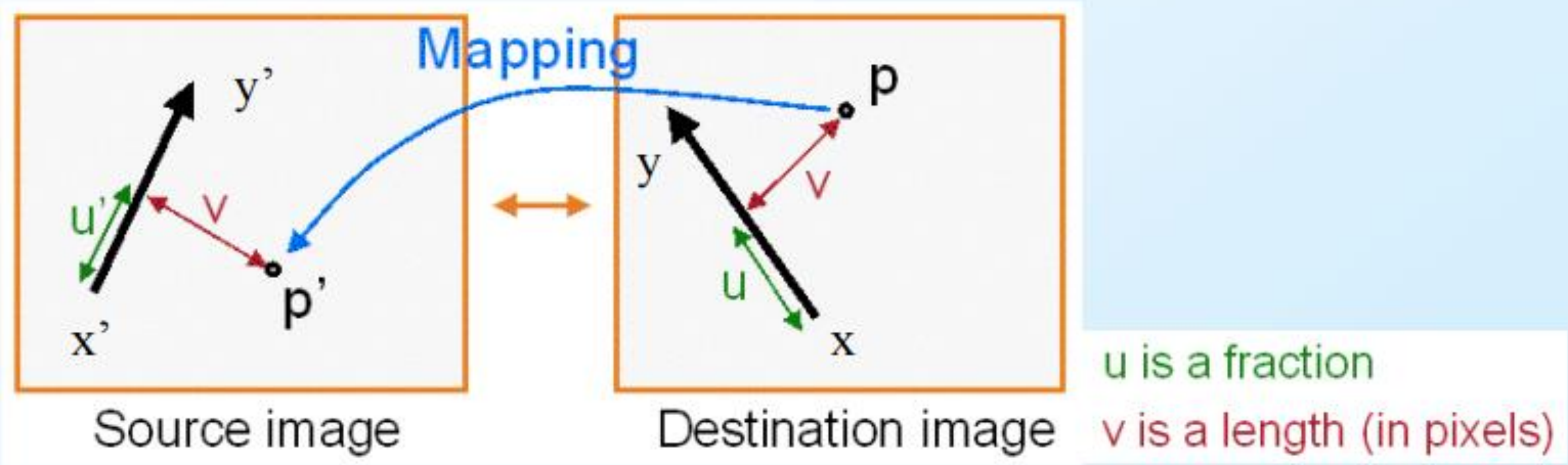
$$p' = x' + u\cdot(y'-x') + \frac{v\cdot Perpendicular(y'-x')}{\|y'-x'\|}$$



From: https://www.doc.ic.ac.uk/~dfg/graphics/graphics2008/Warping-Morphing.pdf

Feature-Based Warping: Beier-Neeley

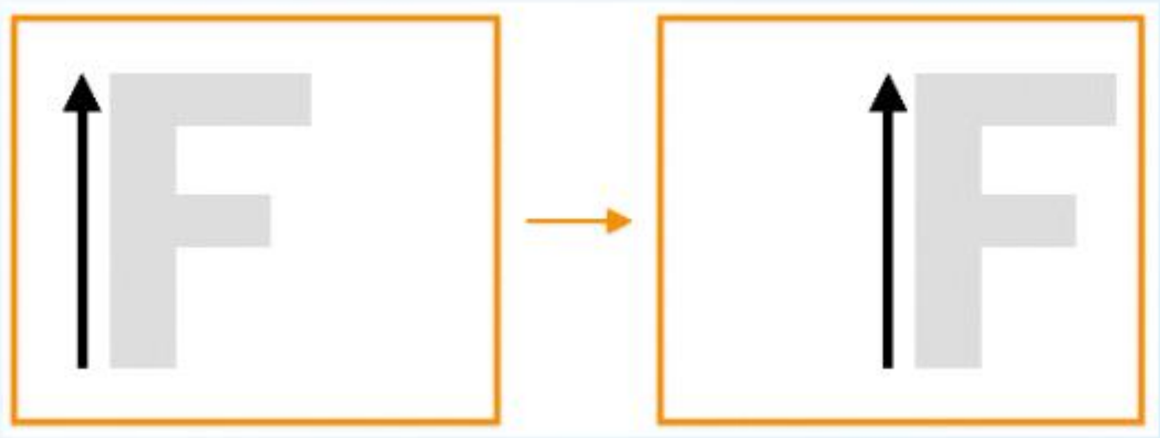
- For each pixel p in the destination image
  - find the corresponding u,v
  - find the p' in the source image for that u,v
  - destination(p) = source(p')



From: https://www.doc.ic.ac.uk/~dfg/graphics/graphics2008/Warping-Morphing.pdf

Warping with One Line Pair: Beier-Neeley

- What happens to the "F" ?

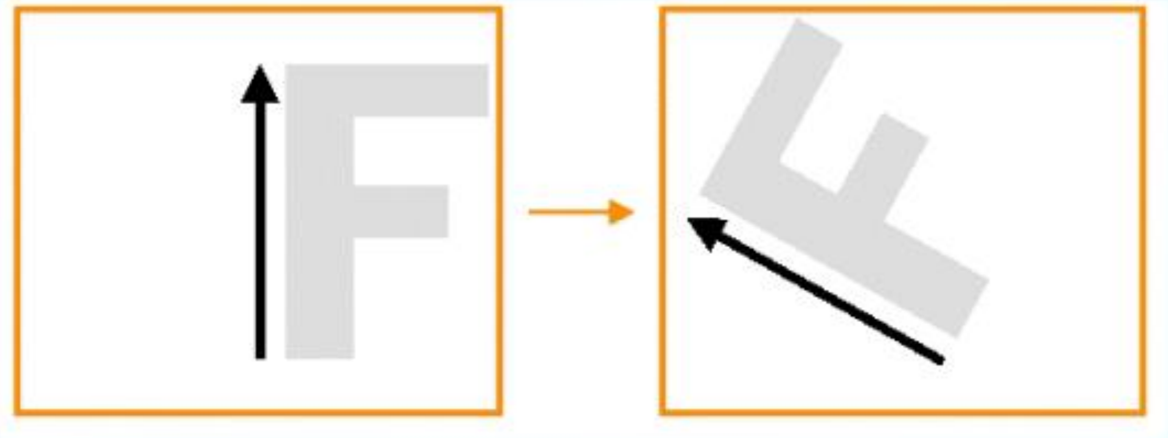


Translation !

From: https://www.doc.ic.ac.uk/~dfg/graphics/graphics2008/Warping-Morphing.pdf

Warping with One Line Pair (cont.): Beier-Neeley

- What happens to the "F" ?

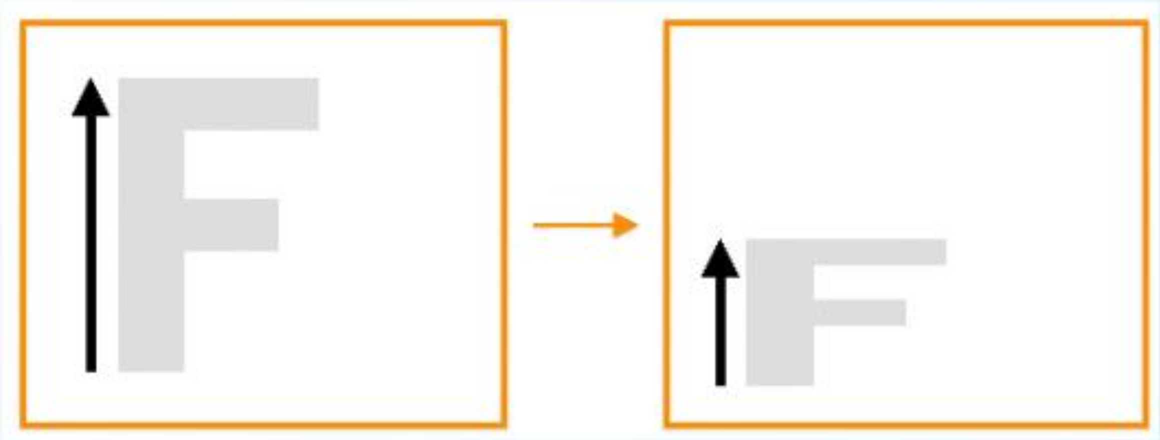


Rotation !

From: https://www.doc.ic.ac.uk/~dfg/graphics/graphics2008/Warping-Morphing.pdf

Warping with One Line Pair (cont.): Beier-Neeley

- What happens to the "F" ?

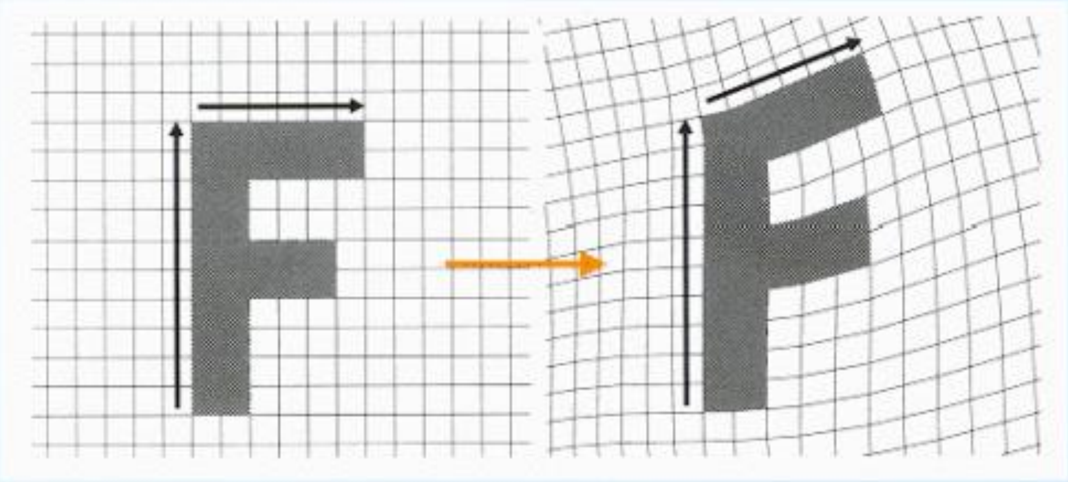


Scale !

From: https://www.doc.ic.ac.uk/~dfg/graphics/graphics2008/Warping-Morphing.pdf

Warping with Multiple Line Pairs: Beier-Neeley

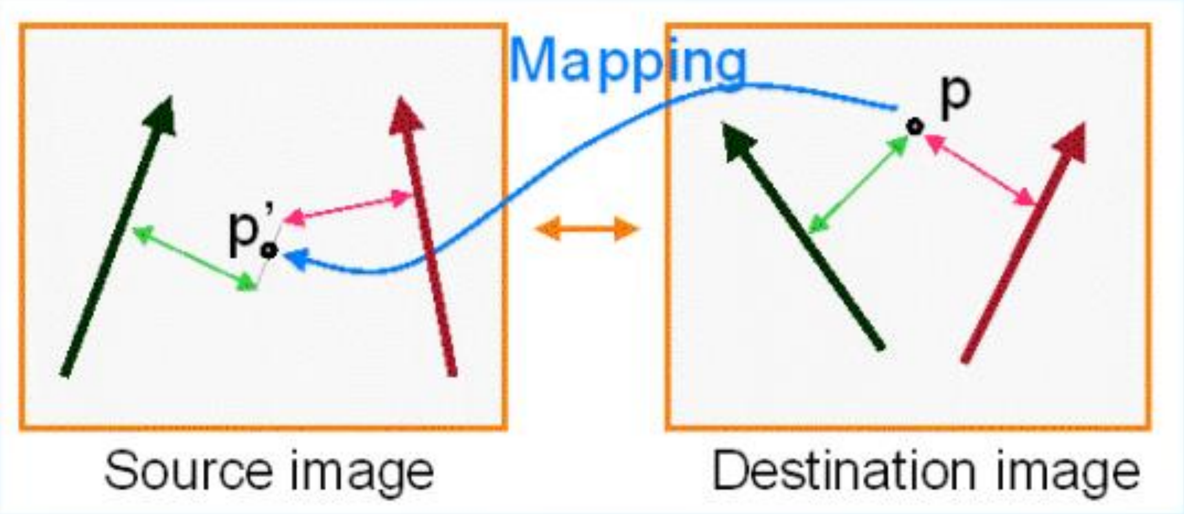
- Use weighted combination of points defined each pair of corresponding lines



From: https://www.doc.ic.ac.uk/~dfg/graphics/graphics2008/Warping-Morphing.pdf

Warping with Multiple Line Pairs: Beier-Neeley

- Use weighted combination of points defined by each pair corresponding lines



p' is a weighted average

From: https://www.doc.ic.ac.uk/~dfg/graphics/graphics2008/Warping-Morphing.pdf

Weighting Effect of Each Line Pair: Beier-Neeley

- To weight the contribution of each line pair

$$weight[i] = \left(\frac{length[i]^p}{a + dist[i]} \right)^b$$

  - where
    - length[i] is the length of L[i]
    - dist[i] is the distance from X to L[i]
    - a, b, p are constants that control the warp

From: https://www.doc.ic.ac.uk/~dfg/graphics/graphics2008/Warping-Morphing.pdf

Warping Pseudocode: Beier-Neeley

```
foreach destination pixel p do
    psum = (0, 0)
    wsum = (0, 0)
    foreach line L[i] in destination do
        p'[i] = p transformed by (L[i], L'[i])
        psum = psum + p'[i] * weight[i]
        wsum += weight[i]
    end
    p' = psum / wsum
    destination(p) = source(p')
end
```

From: https://www.doc.ic.ac.uk/~dfg/graphics/graphics2008/Warping-Morphing.pdf