Simple registration and mosaics

D.A. Forsyth, UIUC

Idea:

To photograph a “spectacular big thing”:

take many photographs

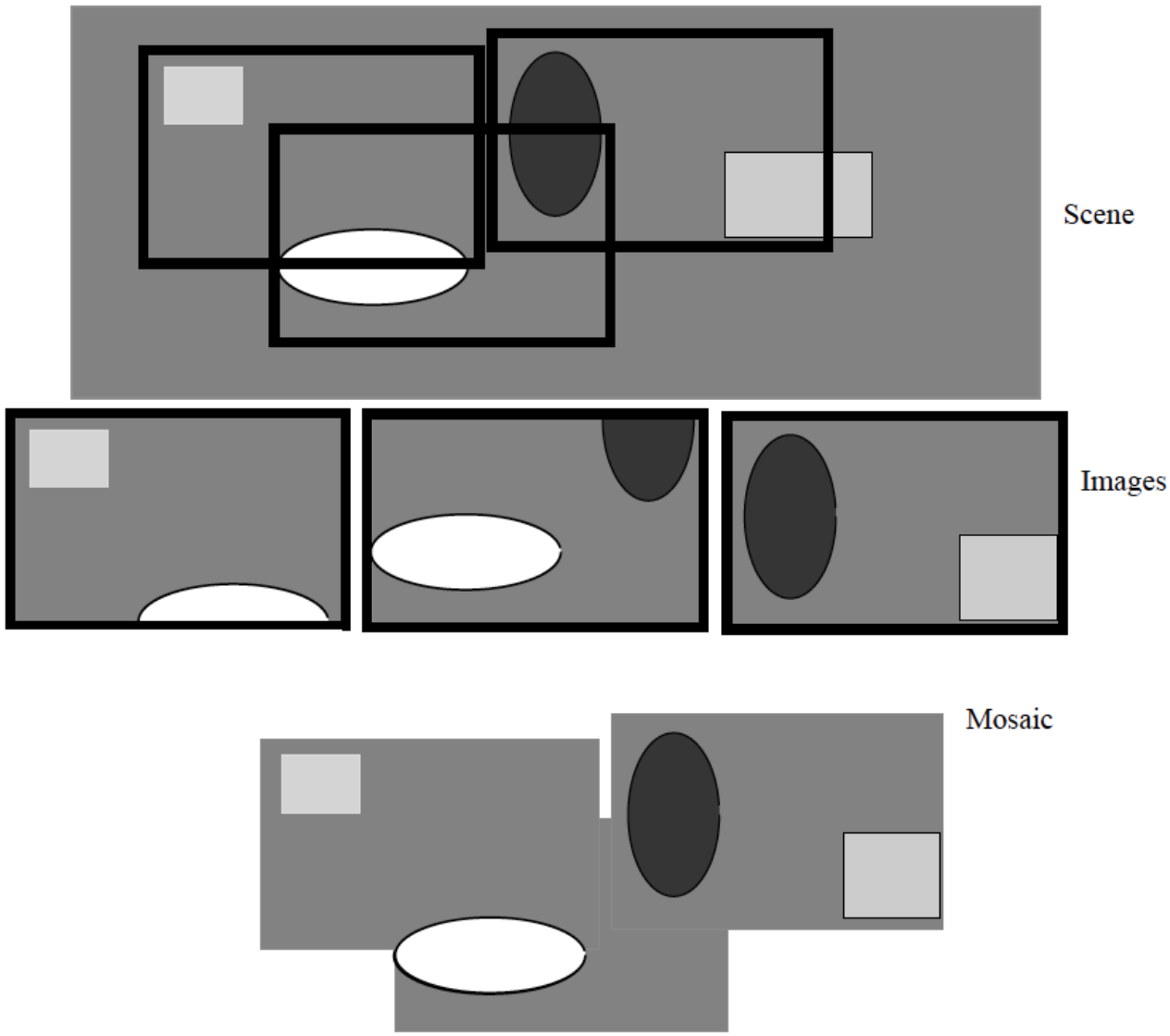
slide over one another to get overlaps right

Result:

one big picture (mosaic)

Useful representation for many purposes

Q: how to slide?


Scene
Images
Mosaic

How to register

For the moment, assume we have two images $\mathcal{A}$ and $\mathcal{B}$. These two images are overlapping views of a scene that can be aligned by a translation. These images are continuous functions of position in the image plane – they haven't been sampled yet. The fact that they can be aligned means that there is some t_x, t_y so that $\mathcal{A}(x, y) = \mathcal{B}(x - t_x, y - t_y)$ when both $x, y \in [0, 1] \times [0, 1]$ and $x + t_x, y + t_y \in [0, 1] \times [0, 1]$. Visualize this as placing $\mathcal{B}$ on top of $\mathcal{A}$, then sliding $\mathcal{B}$ by t_x, t_y; then the parts of $\mathcal{A}$ and $\mathcal{B}$ that overlap look the same. We are given $\mathcal{A}$ and $\mathcal{B}$ and must find t_x, t_y.

Idea:

compute some cost; find translation that gives min

Least squares should spring to mind. We are dealing with sampled images, and so we will search for m, n that are integers, and minimize

$$C_{\text{reg}}(m,n) = \frac{1}{N_o} \sum_{\text{overlap}} (\mathcal{A}_{ij} - \mathcal{B}_{i-m,j-n})^2$$

where overlap is the rectangle of pixel locations with meaningful values for both $\mathcal{A}$ and $\mathcal{B}$ and N_o is the number of pixels in that rectangle. It is important that C_{reg} is an average, because we need to compare overlaps of different sizes (Figure 33.2)

Important: overlaps will differ in size

Mean squared error (MSE)
Indexing ensures pixels that overlap
Are compared

Finding the translation

Simplest case:

Blank search:

choose a range of translations

compute the value for each

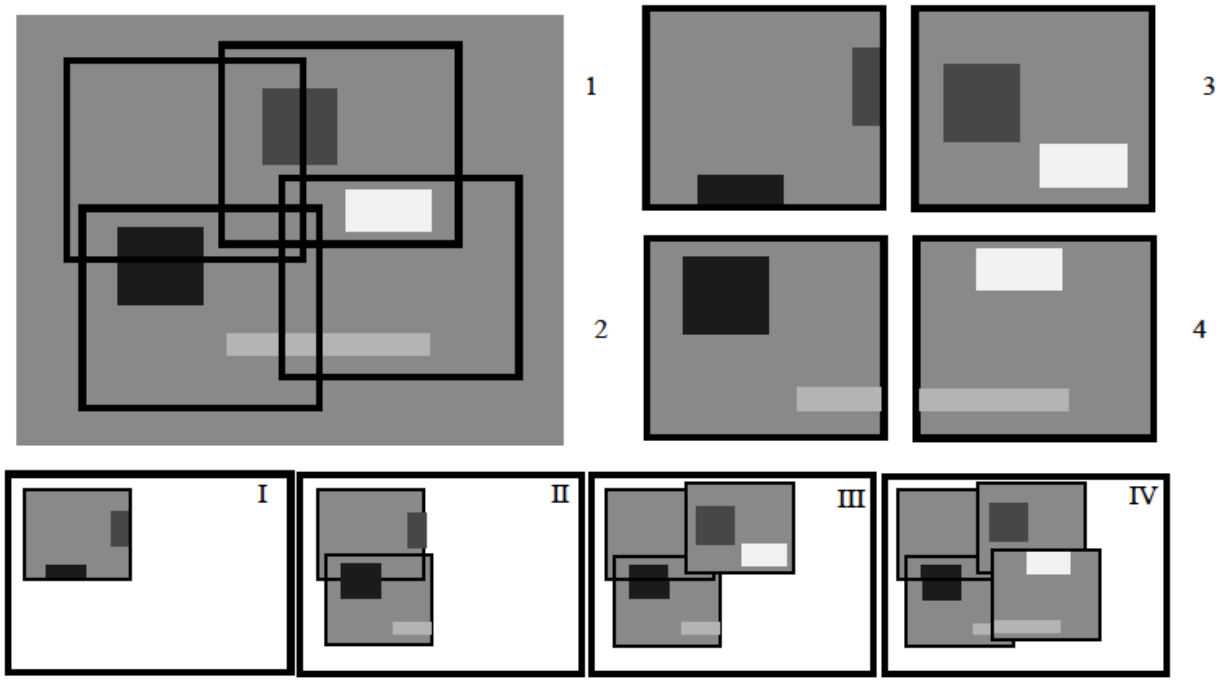
choose the best

AAARGH: slow

Insight:

overlap for smoothed and resampled images should be a good guide to true overlap - details to follow!

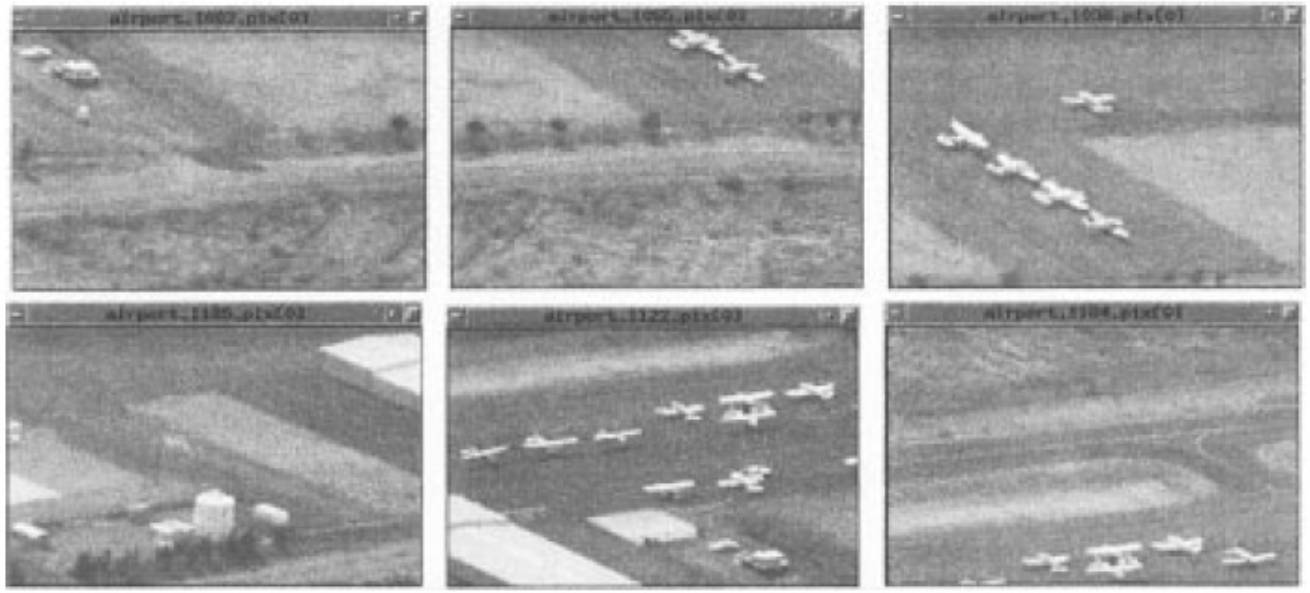
Building a simple mosaic



IGURE 4.4: **Top left** shows a simple scene with the location of four images; these re shown on the **top right**. The steps of creating a mosaic are sketched in the Roman numeral panes on the **bottom**. ~~In this case, the translation of the fourth mage with respect to the first is poorly estimated; more detail in the text.~~

AARGH – sorry!

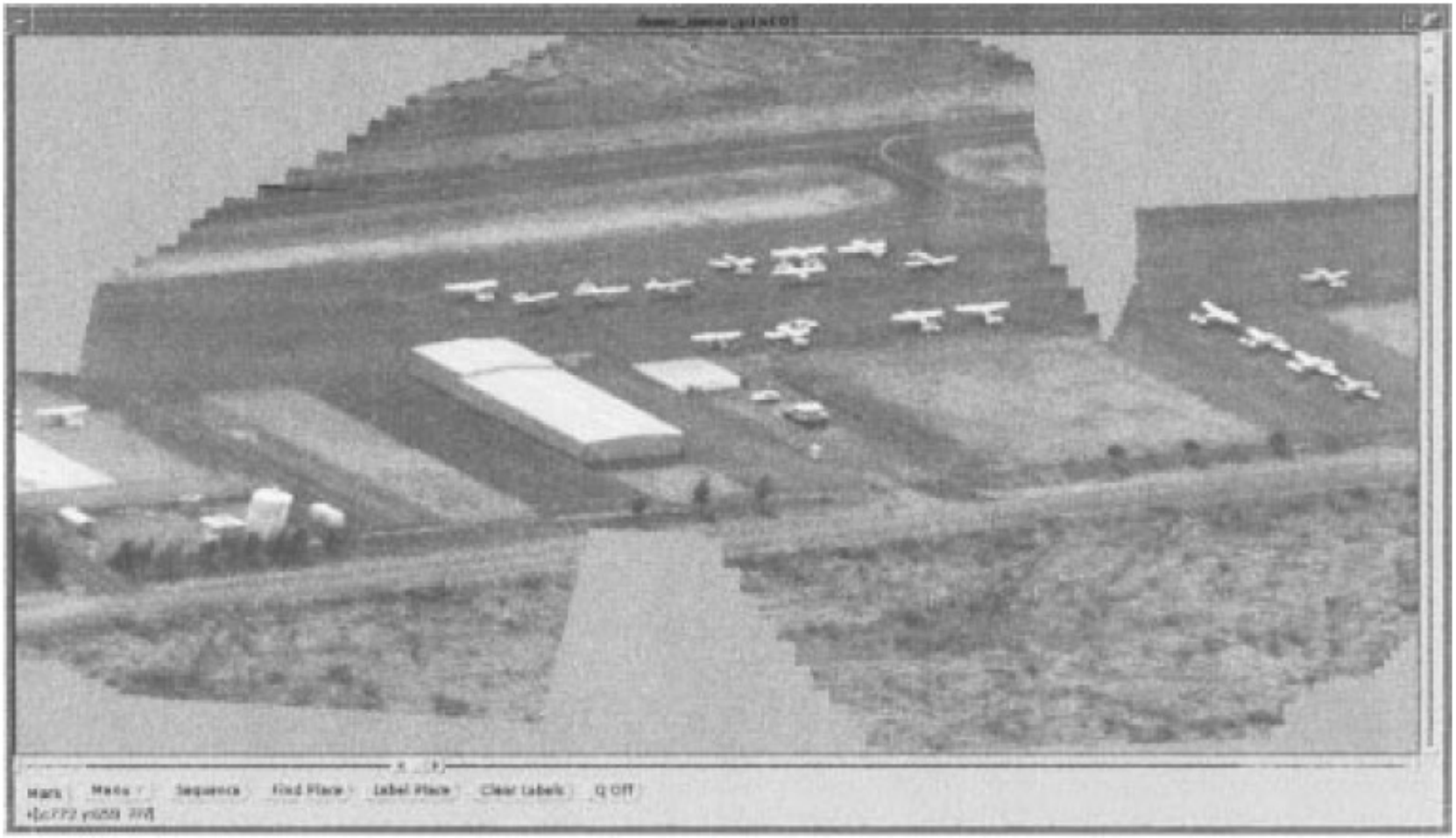
Some pictures….



(a)

Irani +Anandan, 98

Lead to a mosaic.



(b)

Irani +Anandan, 98

Application

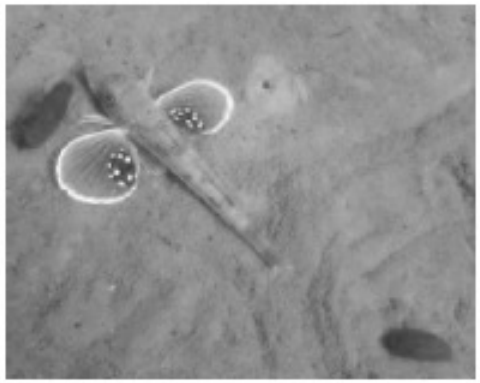
Registering color separations

- Early color photography took 3 different images
  - red filter, green filter, blue filter
- These won't overlap perfectly (slight camera shake)
- Register them by translating

Gurnard

Red

Green



Blue

BUT

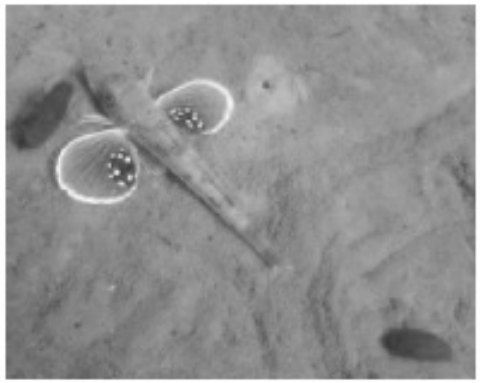
Red, Green and Blue aren't all that similar

so mean squared error may not be a great idea

Gurnard

Red

Green



Blue

Alternatives: The Cosine Distance

- The *cosine distance*, given by:

$$\sum_{\text{overlap}} \frac{(\mathcal{A}_{ij} * \mathcal{B}_{i-m,j-n})}{\sqrt{\sum_{\text{overlap}} \mathcal{A}_{ij}^2} \sqrt{\sum_{\text{overlap}} \mathcal{B}_{i-m,j-n}^2}}.$$

Annoyingly, this cost function is largest when best, even though it's called a distance. Some authors subtract this distance from one (its largest value) to fix this.

Alternatives: Correlation

- The *correlation coefficient*, given by:

$$\sum_{\text{overlap}} \frac{(\mathcal{A}_{ij} - \mu_A) * (\mathcal{B}_{i-m,j-n} - \mu_B)}{\sqrt{\sum_{\text{overlap}} \mathcal{A}_{ij}^2} \sqrt{\sum_{\text{overlap}} \mathcal{B}_{i-m,j-n}^2}}$$

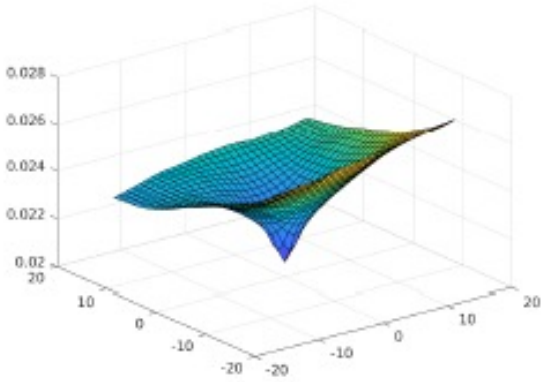
$$\text{where } \mu_A = \frac{1}{N_O} \sum_{overlap} \mathcal{A}_{ij} \text{ and}$$

$$\text{where } \mu_B = \frac{1}{N_O} \sum_{overlap} \mathcal{B}_{ij}.$$

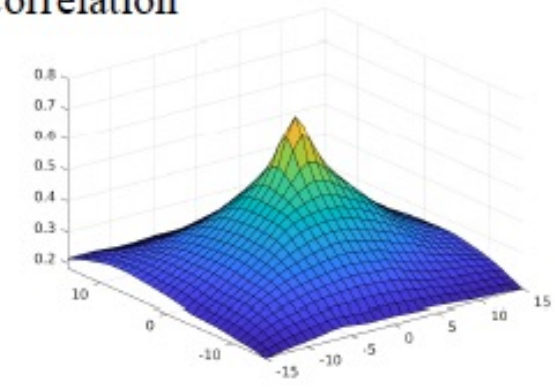
This is big for the best alignment. Notice how this corrects for the mean of the overlap in each window.

Distances

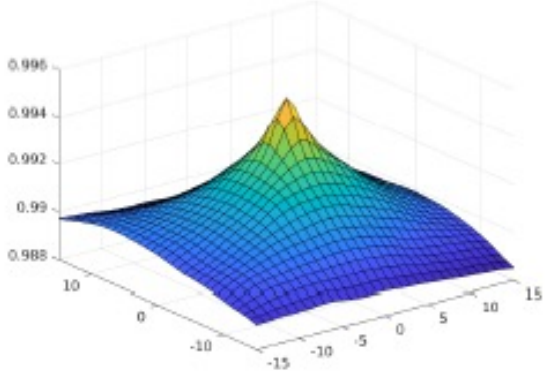
Squared error



Correlation



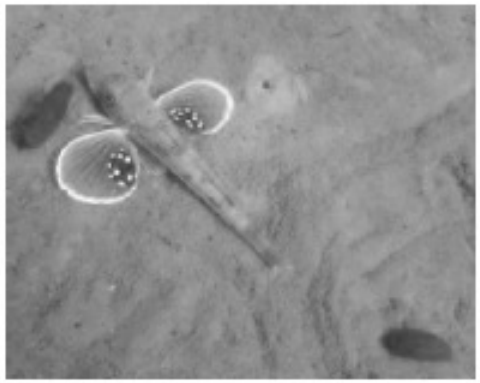
Cosine distance



Gurnard

Red

Green



Blue

Registering a zebra to itself

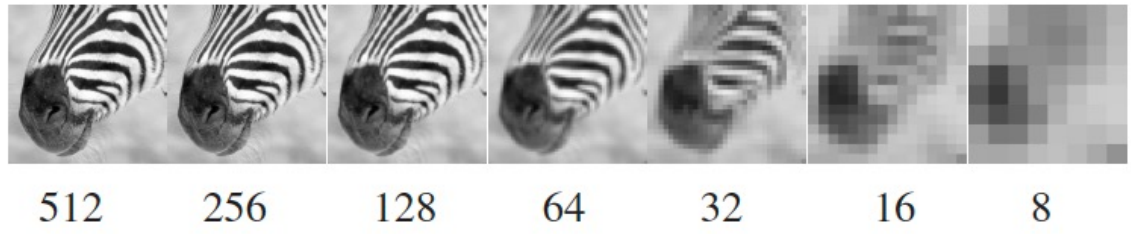
overlap for smoothed and resampled images should be a good guide to true overlap - details to follow!

The Gaussian Pyramid

Set the finest scale layer to the image
For each layer, going from next to finest to coarsest
  Obtain this layer by smoothing the next finest
  layer with a Gaussian, and then subsampling it
end

Algorithm 4.1: *Forming a Gaussian Pyramid.*

The Gaussian Pyramid



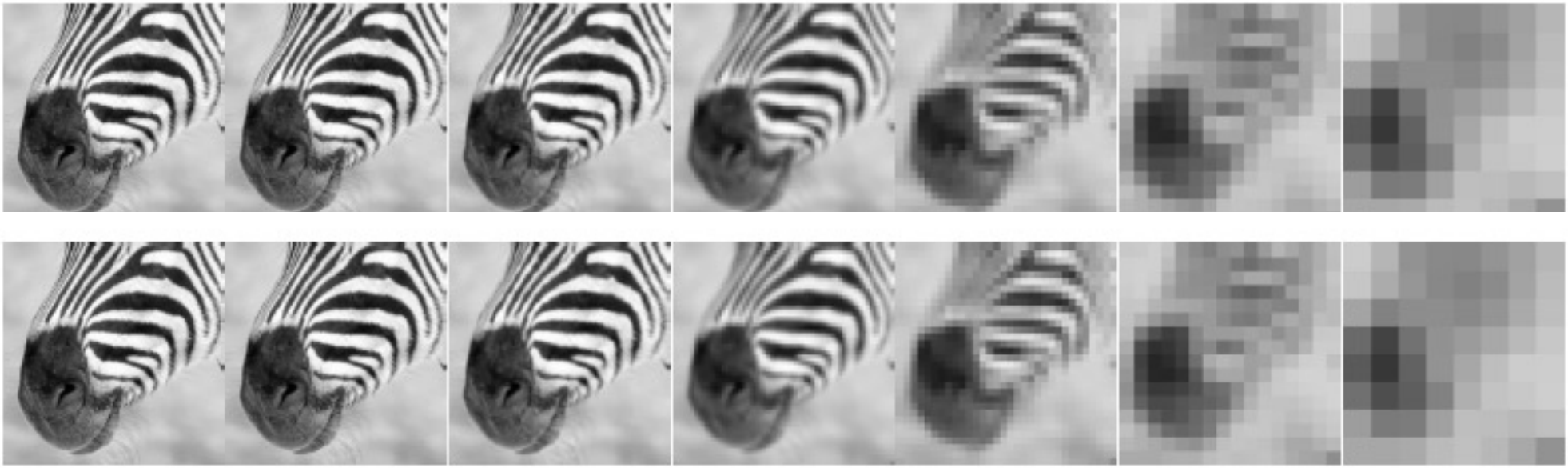
512 256 128 64 32 16 8

The Gaussian Pyramid

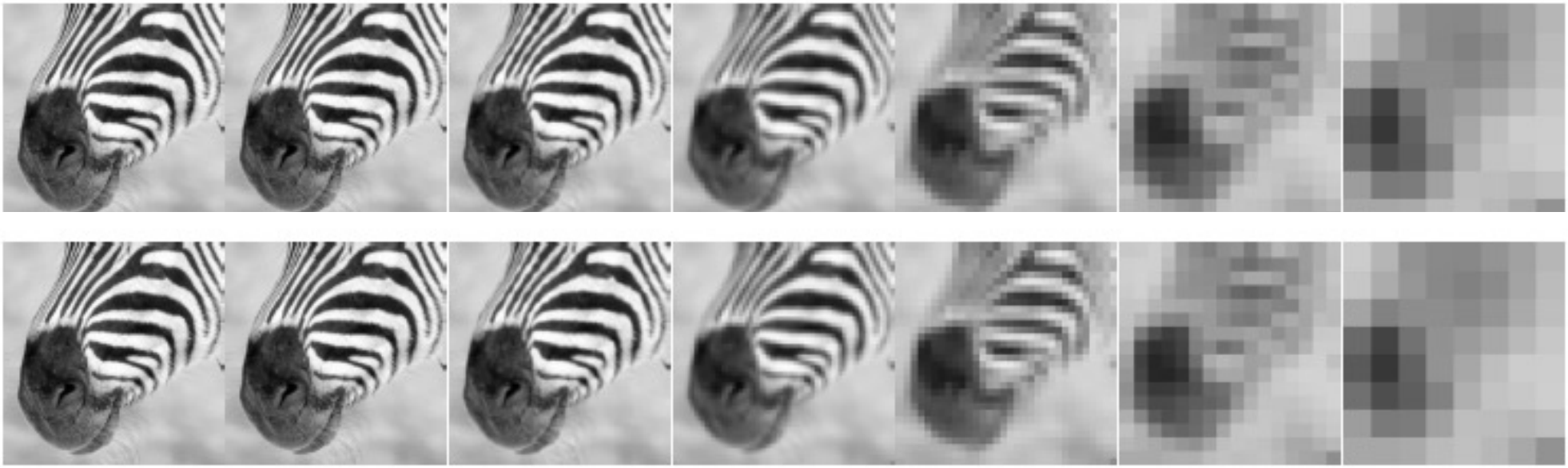
Set the finest scale layer to the image
For each layer, going from next to finest to coarsest
  Obtain this layer by smoothing the next finest
  layer with a Gaussian, and then subsampling it
end

Algorithm 4.1: *Forming a Gaussian Pyramid.*

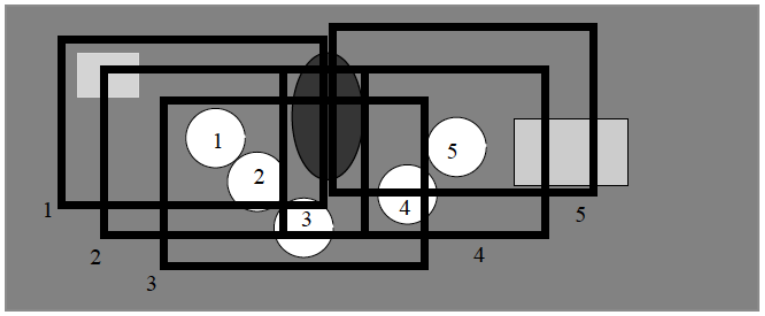
This was by 2 in example; could be others (sqrt(2) is common)



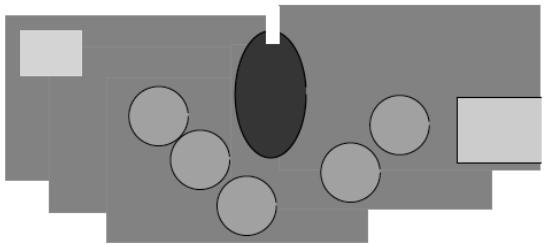
Now look at the zebra's muzzle in Figure 4.2, and think about registering this image to itself. The 8×8 version has very few pixels, and looks like a medium dark bar, darker at the muzzle end. Finding a translation to register this image to itself should be fairly straightforward, and unambiguous. Assume we find m_8, n_8.



In the 16×16 version, some stripes are visible. Registering this image to itself might be more difficult, because the stripes will create local minima of the cost function (check you follow this remark; think about what happens if you have the images registered, and then shift the muzzle perpendicular to the stripes). But if we have an estimate of the translation from the 8×8 version, we do not need to search a large range of translations to register the 16×16 version. We need to look only at four translations: $2 * m_8, 2 * n_8$; $2 * m_8 + 1, 2 * n_8$; $2 * m_8, 2 * n_8 + 1$; and $2 * m_8 + 1, 2 * n_8 + 1$. The same reasoning applies when going from the 16×16 version to the 32×32 version, and so on. This strategy is known as *coarse-to-fine search.*

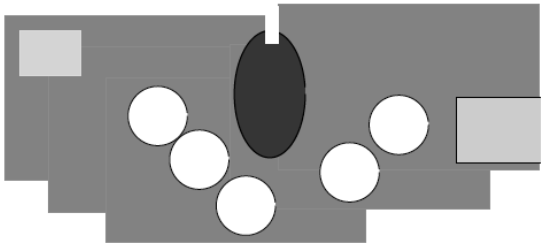


Scene, with moving circle; numbers show where it is for frame k



Registered, then take the mean

Registered, then take the median

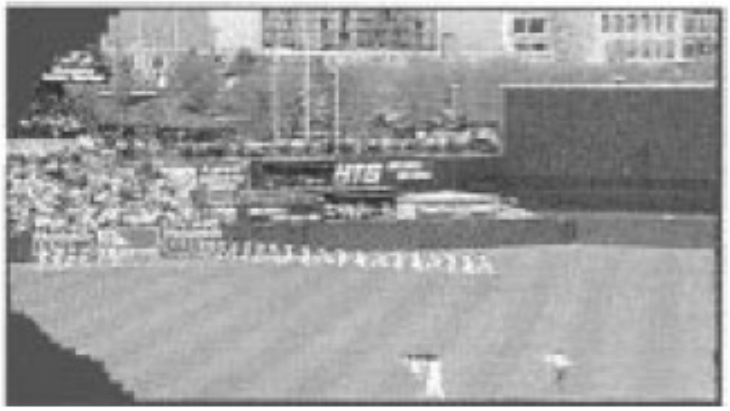


Registered, then take the pixel most different from median

Video summaries

(a)

(b)



(c)

Irani +Anandan, 98

Registration problems….

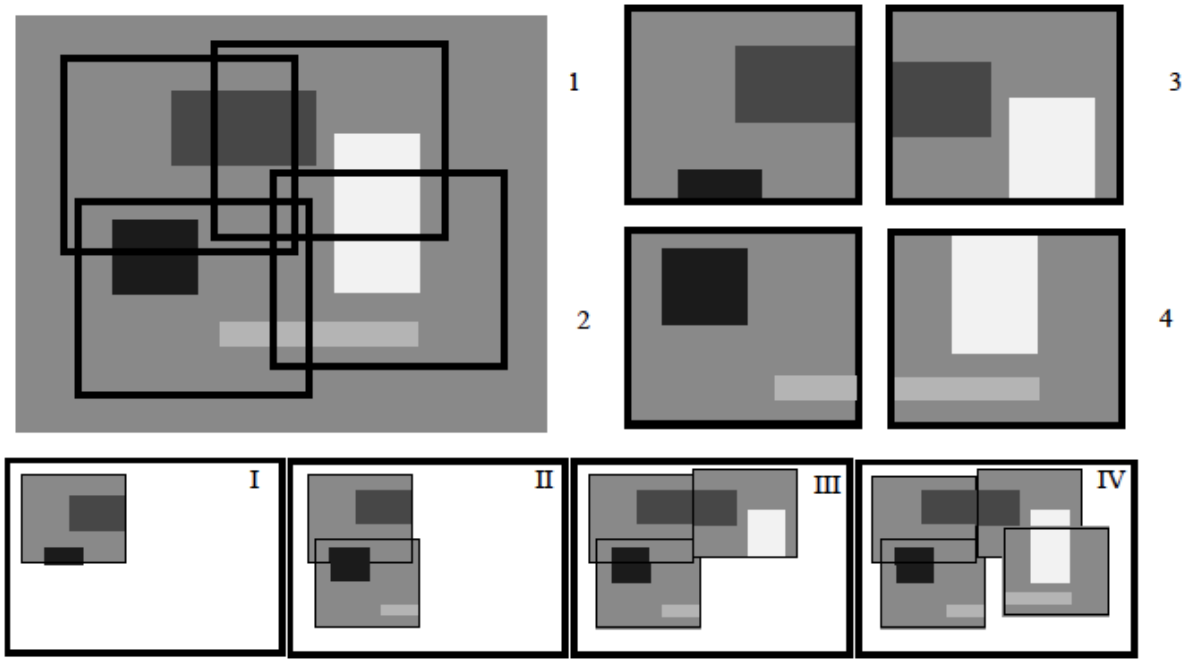
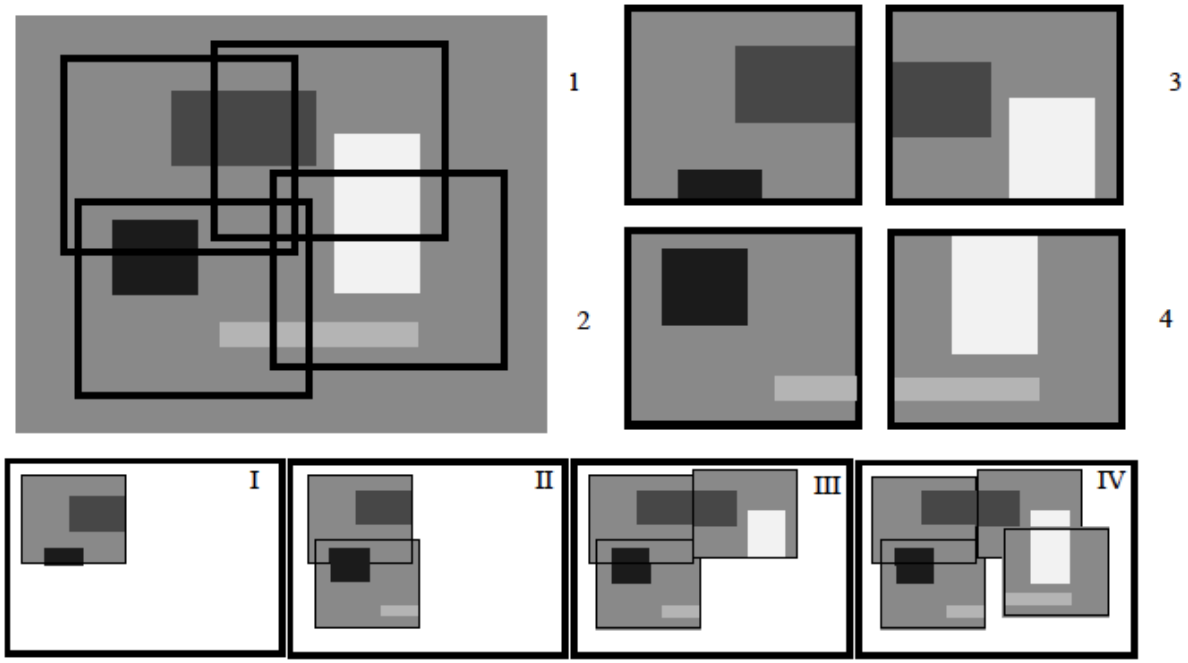


FIGURE 4.6: **Top left** *shows a simple scene with the location of four images; these are shown on the* **top right**. *The steps of creating a mosaic are sketched in the Roman numeral panes on the* **bottom**. *In this case, the translation of the fourth image with respect to the first is poorly estimated; more detail in the text.*

Registration problems….



Notice that this isn't necessary. Some pixels of $\mathcal{I}_4$ overlap $\mathcal{I}_2$. If these pixels contributed to the registration, $\mathcal{I}_4$ and $\mathcal{I}_3$ could be properly registered. This problem occurs quite generally – it is not just a result of an odd scene – and is often referred to as a failure of *loop closure*. This refers to the idea that if 2 is registered to 1, 3 is registered to 2, 4 is registered to 3, all the way up to N is registered to $N-1$, N may not be registered to 1 at all well – the loop does not close.

Rough sketch of bundle adjustment

There are several procedures to prevent or control this kind of error propagation. Start by registering the images by pairs as described. The easiest strategy is now to repeat: fix all but one image, then register that image to all the others it overlaps with. This approach can work, but may be slow, because it may take a long time for improvements to propagate across all the images. The alternative, which is better but can be onerous, is to fix one image in place, then adjust all others to register in every overlap. Associate a translation $t_{i;x}, t_{j;y}$ with each image $\mathcal{I}_i$, and set $t_{1;x} = 0, t_{1;y} = 0$. Write $\mathbf{t}$ for a vector of all these translations except $t_{1;x}, t_{1;y}$, and $O_{ij}(\mathbf{t})$ for the area of the overlap between $\mathcal{I}_i$ and $\mathcal{I}_j$. Now minimize

$$\sum_{i,j \in \text{overlapping pairs of images}} \left\{ \frac{1}{O_{ij}(\mathbf{t})} \sum_{x,y \in \text{overlap}} \left[\mathcal{I}_{i;x+t_{i;x},y+t_{i;y}} - \mathcal{I}_{j;x+t_{j;x},y+t_{j;y}} \right]^2 \right\}$$

as a function of $\mathbf{t}$. This isn't a straightforward optimization problem. If you require