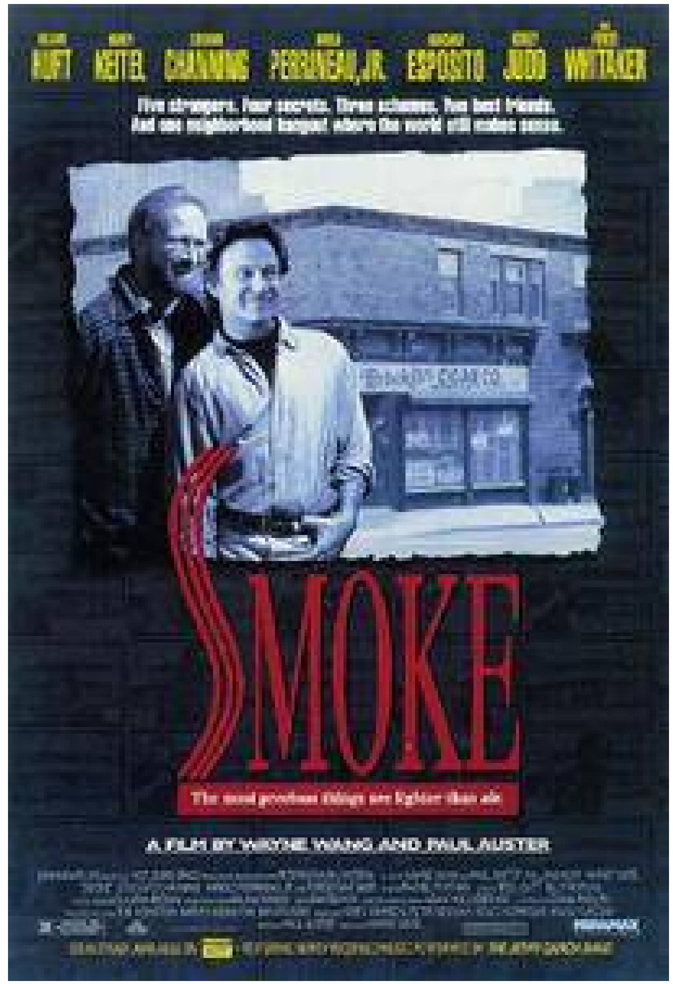
Background Subtraction and Matting

© Yuri Bonder

15-463: Computational Photography
Alexei Efros, CMU, Fall 2005

“Smoke” (1996), the “photo album scene”



Moving in Time

Moving only in time, while not moving in space, has many advantages

- No need to find correspondences
- Can look at how each ray changes over time
- In science, always good to change just one variable at a time

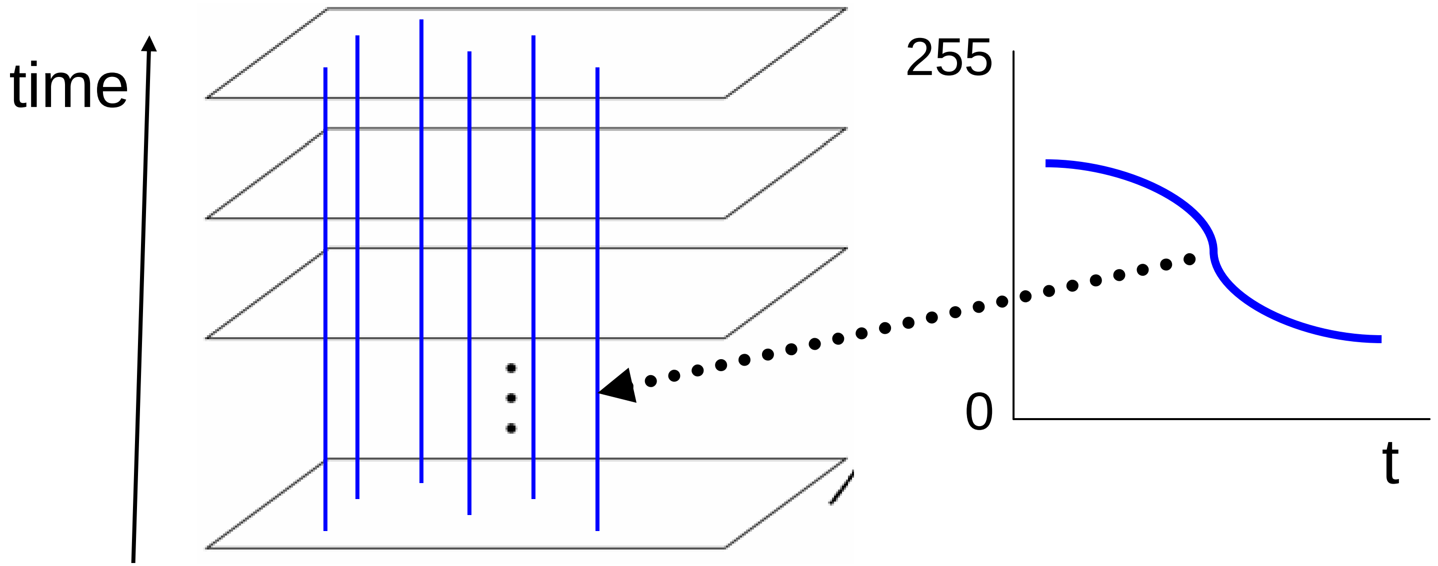
This approach has always interested artists (e.g. Monet)



Modern surveillance video camera is a great source of information

- There are now many such WebCams now, some running for several years!

Image Stack



As can look at video data as a spatio-temporal volume

- If camera is stationary, each line through time corresponds to a single ray in space
- We can look at how each ray behaves
- What are interesting things to ask?

Example

Getting the right pixels

Average image

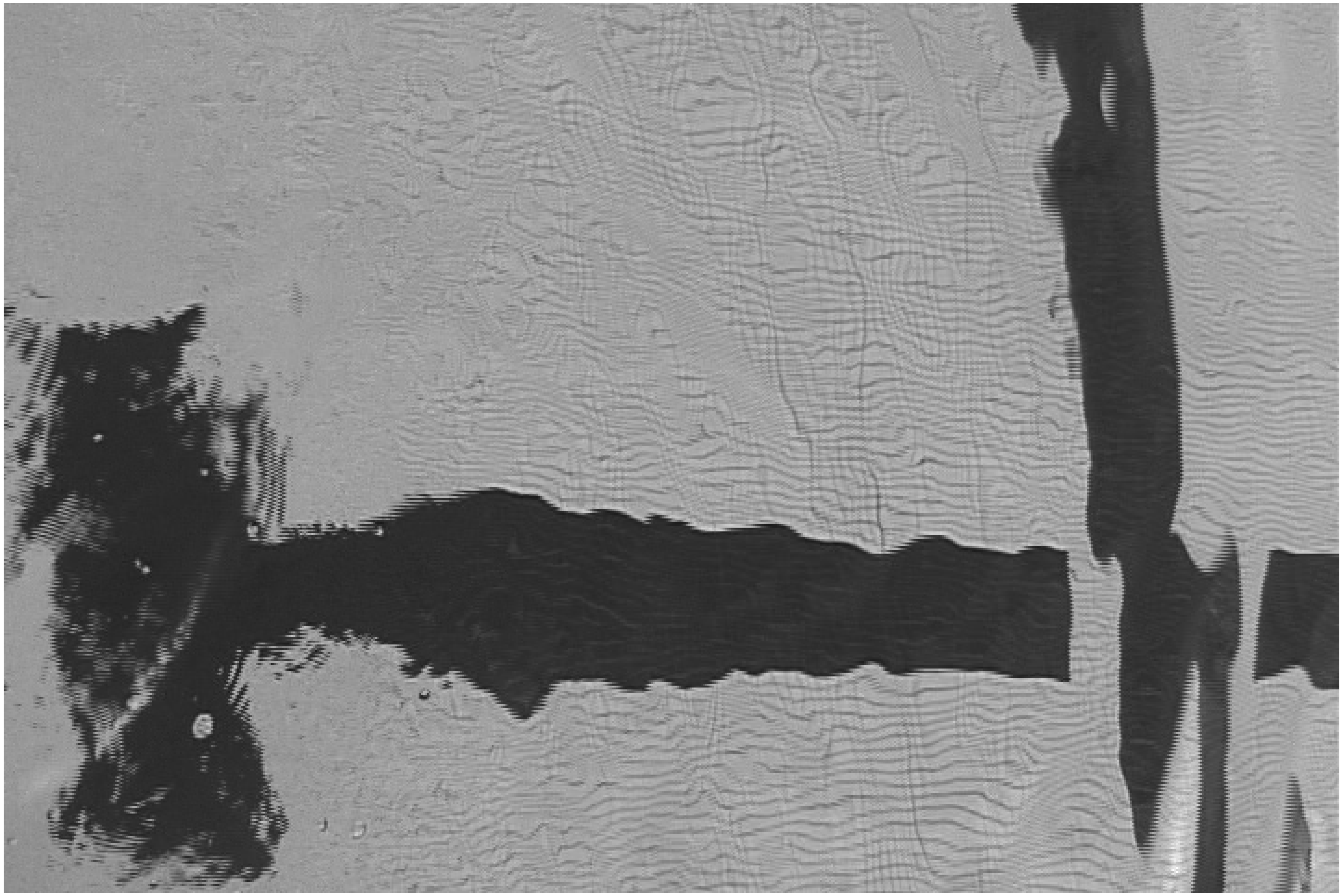
Median Image

Webcams

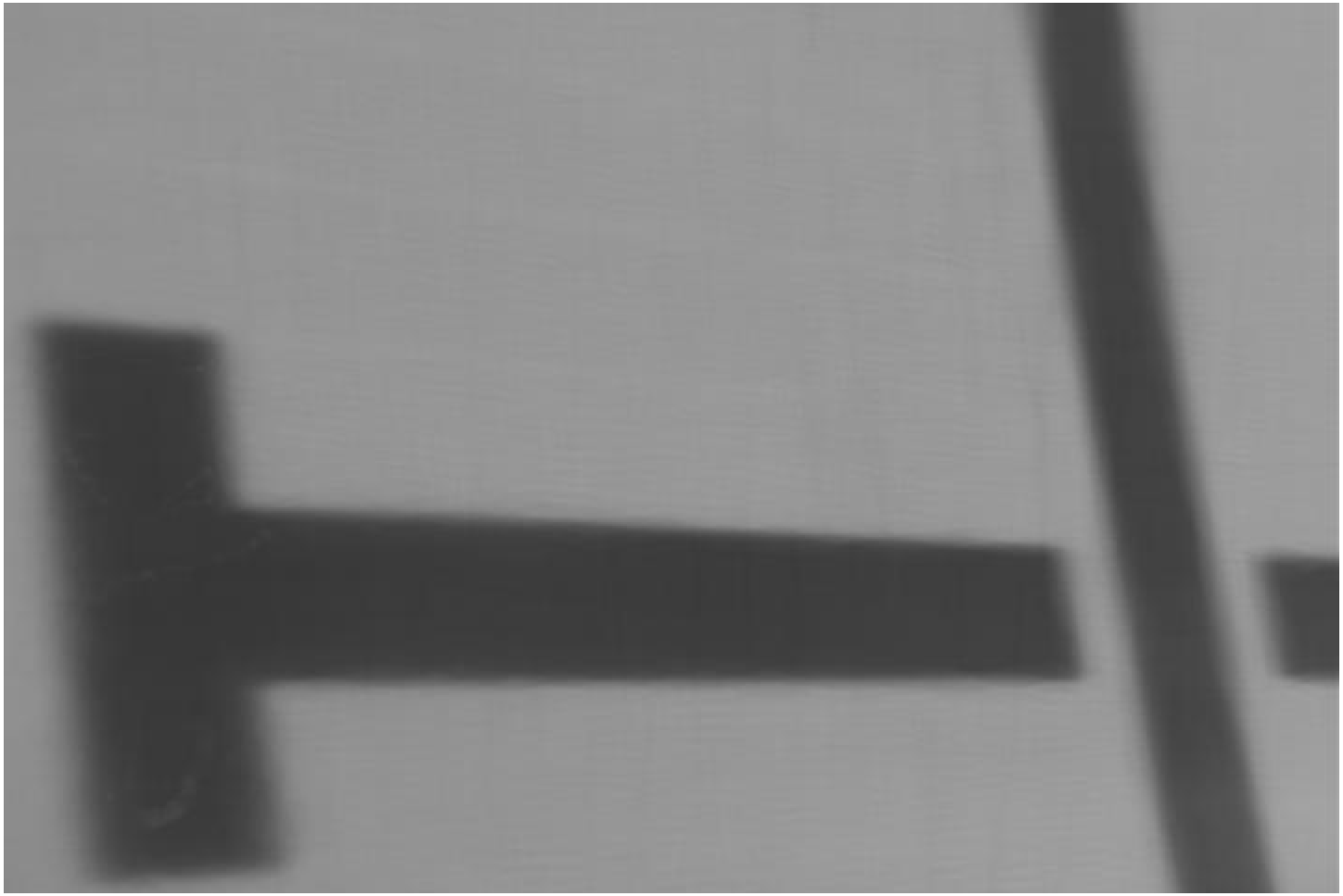
Lots of cool potential projects

- PCA, weather morphing, weather extrapolation, etc.

Input Video

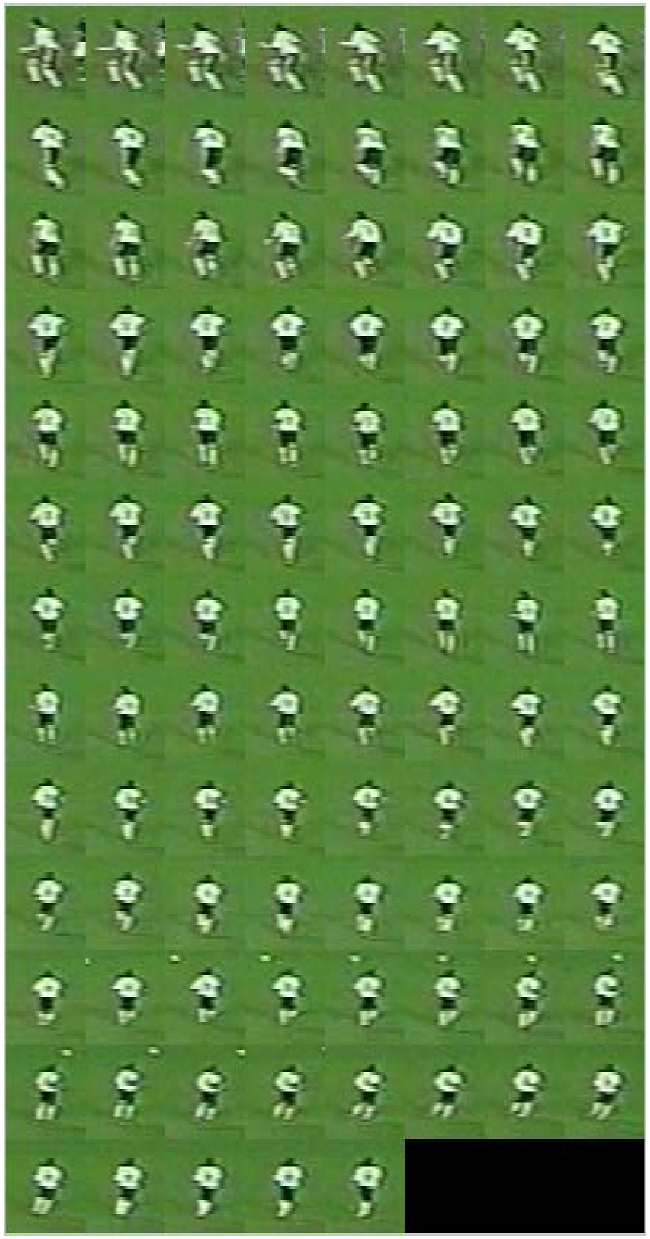


Average Image



What is happening?

Figure-centric Representation



Context-based Image Correction

Input sequence

3 closest frames

median images

Average/Median Image

What can we do with this?

Background Subtraction

-

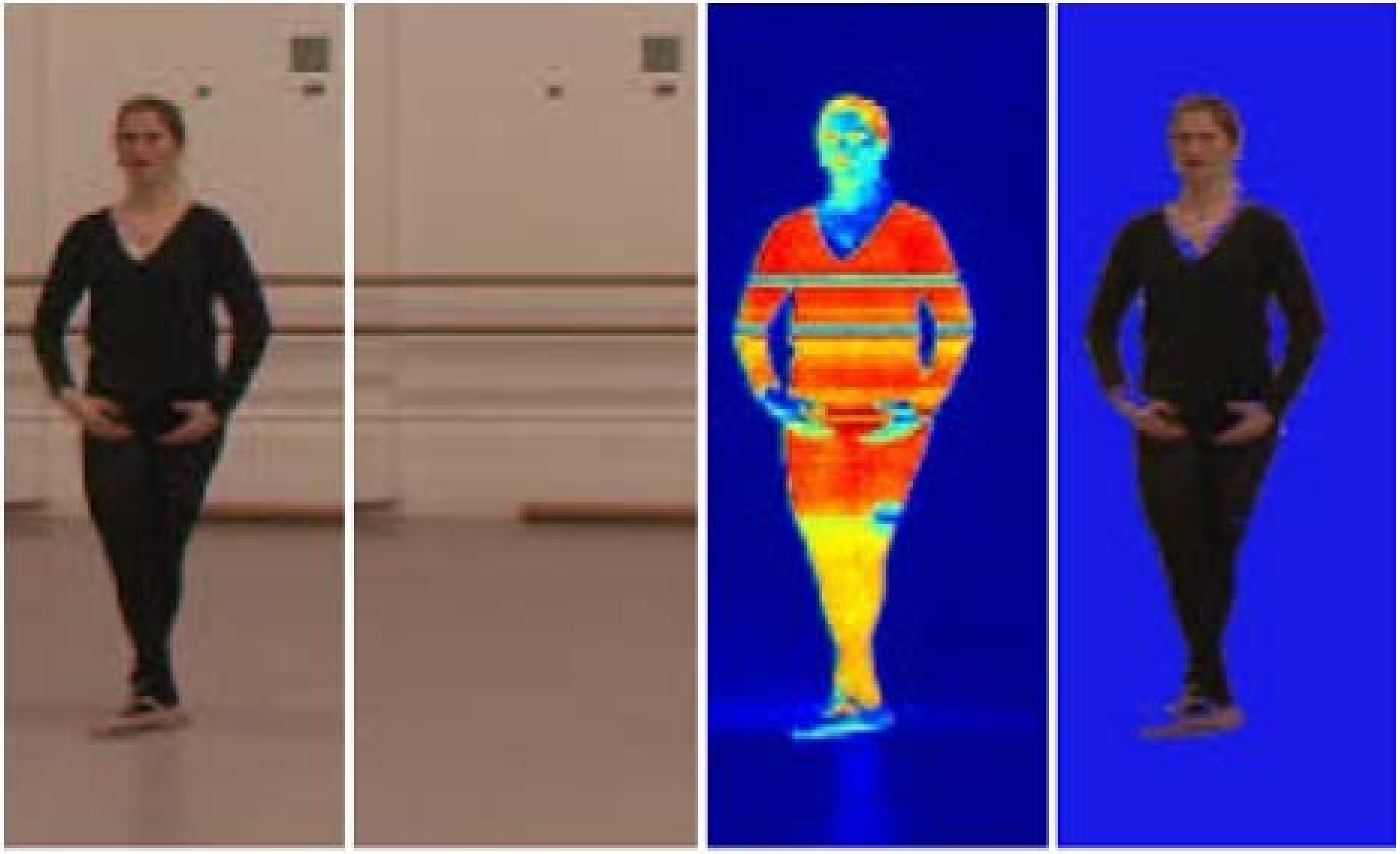
=

Crowd Synthesis (with Pooja Nath)

1. Do background subtraction in each frame
2. Find and record "blobs"
3. For synthesis, randomly sample the blobs, taking care not to overlap them

Background Subtraction

A largely unsolved problem…



One video frame | Estimated background | Difference Image | Thresholded Foreground on blue

How does Superman fly?

Super-human powers?

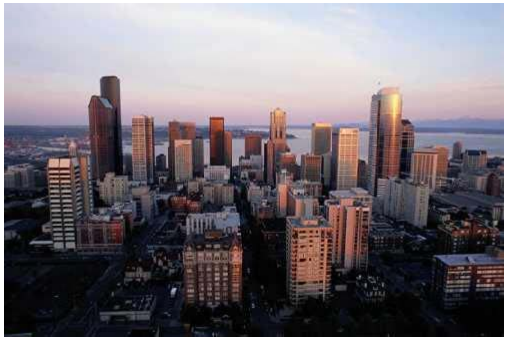
OR

Image Matting and Compositing?

Image Compositing

Compositing Procedure

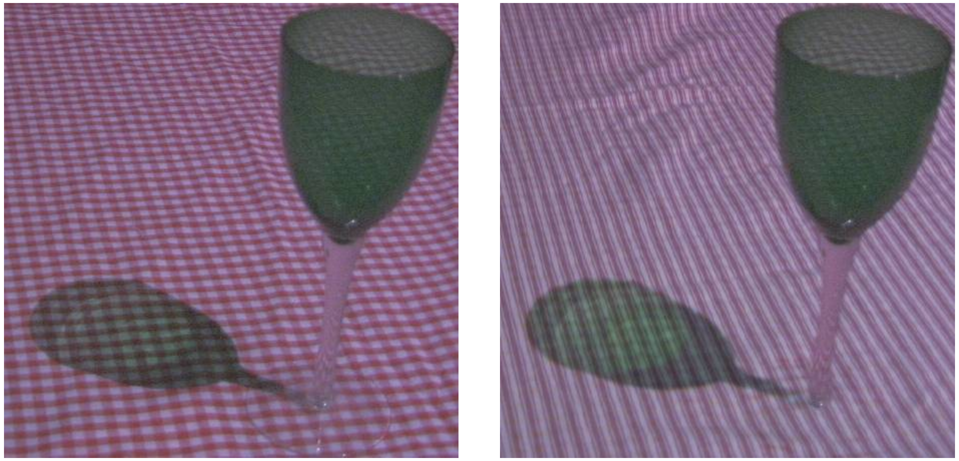
1. Extract Sprites (e.g using *Intelligent Scissors* in Photoshop)



2. Blend them into the composite (in the right order)

Composite by
David Dewey

Compositing: Two Issues



Semi-transparent objects

Pixels too large

Solution: alpha channel

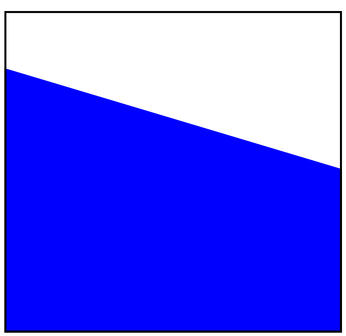
Add one more channel:

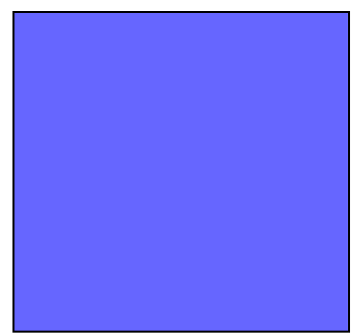
- Image(R,G,B,alpha) ⇐ *Sprite!*

Encodes transparency (or pixel coverage):

- Alpha = 1: opaque object (complete coverage)
- Alpha = 0: transparent object (no coverage)
- 0<Alpha<1: semi-transparent (partial coverage)

Example: alpha = 0.7





Partial coverage or semi-transparency

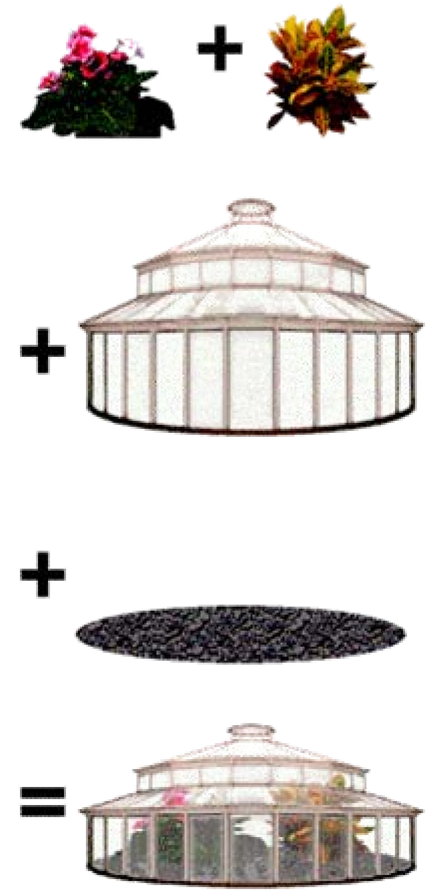
Multiple Alpha Blending

So far we assumed that one image (background) is opaque.

If blending semi-transparent sprites (the “A over B” operation):

$$I_{comp} = \alpha_a I_a + (1-\alpha_a)\alpha_b I_b$$

$$\alpha_{comp} = \alpha_a + (1-\alpha_a)\alpha_b$$



Note: sometimes alpha is premultiplied: im(αR,αG,αB,α):

$$I_{comp} = I_a + (1-\alpha_a) I_b$$

(same for alpha!)

“Pulling a Matte”

Problem Definition:

- The separation of an image C into
  - A foreground object image C_o,
  - a background image C_b,
  - and an alpha matte α
- C_o and α can then be used to composite the foreground object into a different image

Hard problem

- Even if alpha is binary, this is hard to do automatically (background subtraction problem)
- For movies/TV, manual segmentation of each frame is infeasible
- Need to make a simplifying assumption…

Blue Screen

Blue Screen matting

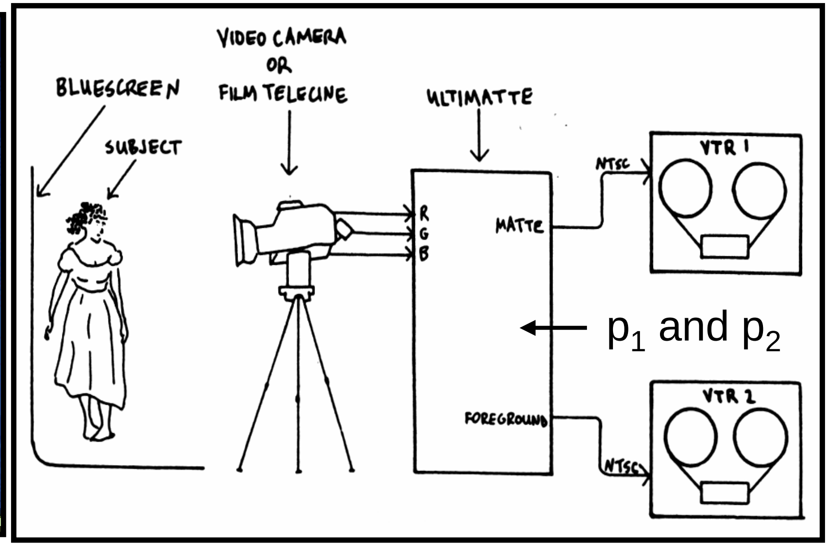
Most common form of matting in TV studios & movies

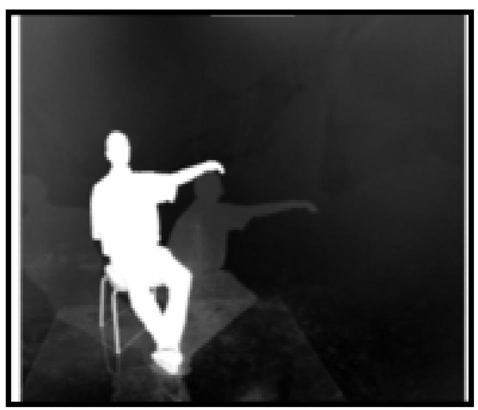
Petros Vlahos invented blue screen matting in the 50s. His Ultimatte® is still the most popular equipment. He won an Oscar for lifetime achievement.

A form of background subtraction:

- Need a known background
- Compute alpha as SSD(C,Cb) > threshold
  - Or use Vlahos' formula: $\alpha = 1-p_1(B-p_2G)$
- Hope that foreground object doesn't look like background
  - no blue ties!
- Why blue?
- Why uniform?

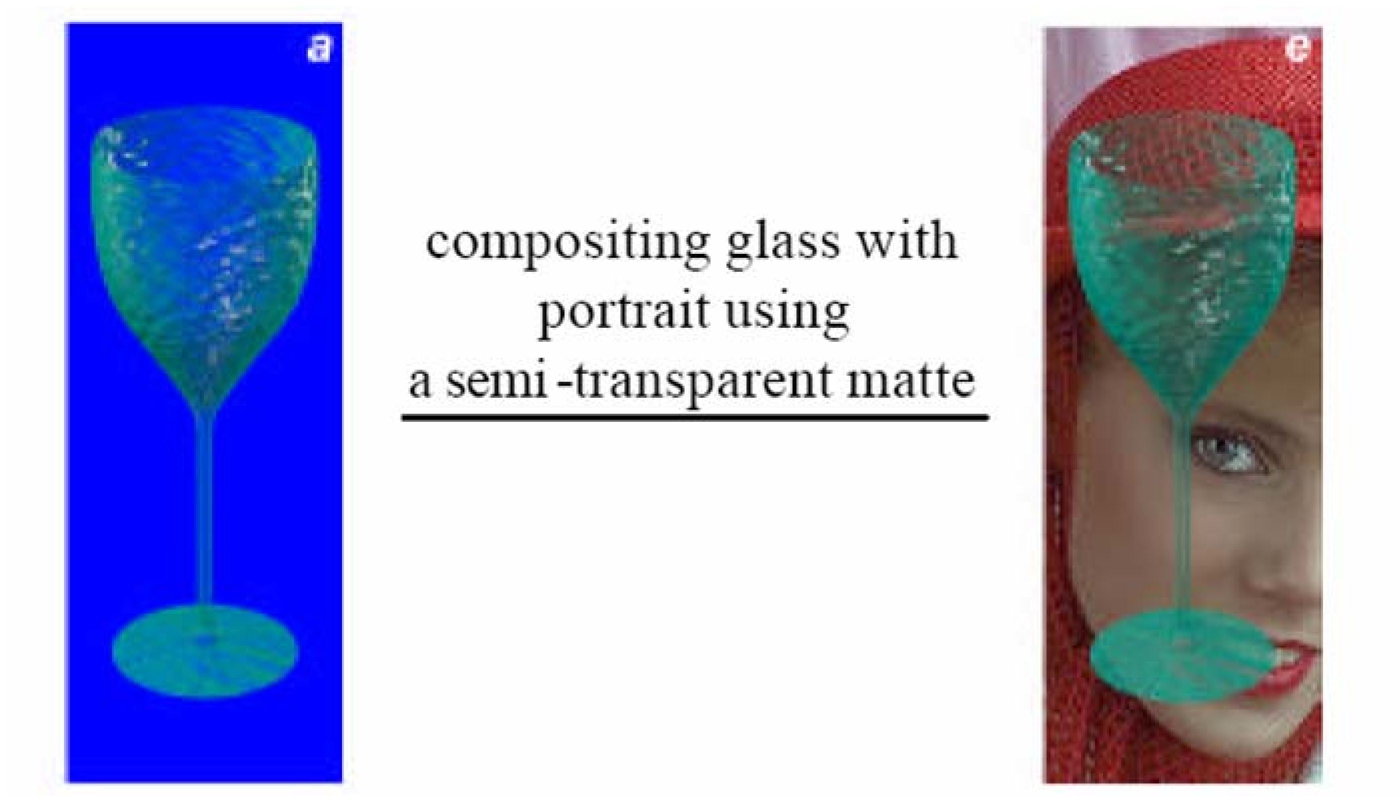
The Ultimatte





Blue screen for superman?

Semi-transparent mattes



What we really want is to obtain a true alpha matte, which involves semi-transparency

- Alpha between 0 and 1

Matting Problem: Mathematical Definition

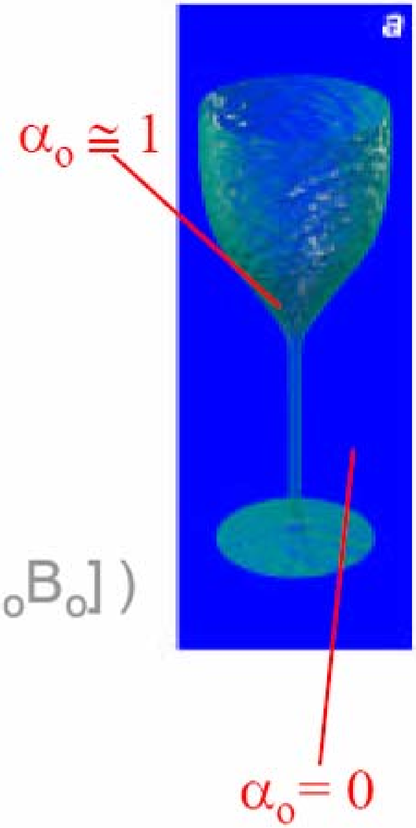
For every pixel in the composite image,

<u>given</u>

- backing color $C_k = [R_k \;\; G_k \;\; B_k]$, and
- composite pixel color $C = [R \;\; G \;\; B]$

<u>compute</u>

- foreground pixel color
  $C_o = [R_o \;\; G_o \;\; B_o \;\; \alpha_o]$ ($= [\alpha_o R_o \; \alpha_o G_o \; \alpha_o B_o]$)
  such that



The matting equation

$$C = C_o + (1 - \alpha_o)\, C_k$$

Why is general matting hard?

Matting Equation:

$$C = C_o + (1 - \alpha_o) C_k$$

Solution #1: No Blue!

Matting Equation:

$$C = C_o + (1 - \alpha_o)\, C_k$$

- If we know that the foreground contains no blue, we have $B_o = 0$
- This leaves us with 3 equations and 3 unknowns, which has exactly one solution

$$R = \alpha_o R_o + (1 - \alpha_o)\, R_k \quad \leftarrow \text{3. Solve for } R_o$$
$$G = \alpha_o G_o + (1 - \alpha_o)\, G_k \quad \leftarrow \text{2. Solve for } G_o$$
$$B = B_k - \alpha_o B_k \quad \leftarrow \text{1. Solve for } \alpha_o$$

Main difficulty:

Solution #2: Gray or Flesh

Matting Equation:

$$C = C_o + (1 - \alpha_o) C_k$$

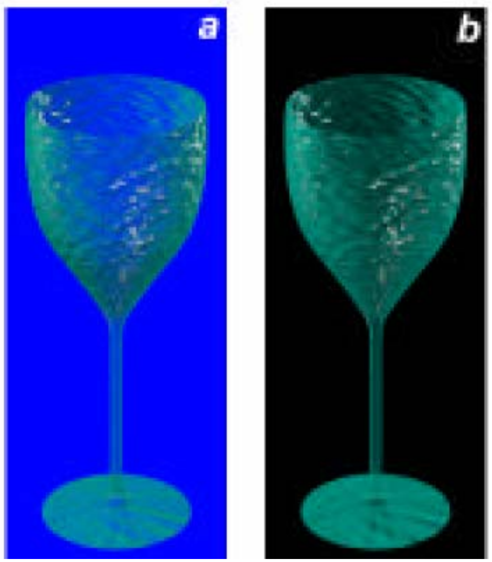
- If we know that the foreground contains gray, that means that $R_o = B_o = G_o$
- This leaves us with 3 equations and 2 unknowns

Triangulation Matting (Smith & Blinn)

Matting Equation:

$$C = C_o + (1 - \alpha_o)\, C_k$$

- Instead of reducing the number of unknowns, we could attempt to increase the number of equations
- One way to do this is to photograph an object of interest in front of two known but distinct backgrounds



How many equations?

How many unknowns?

Does the background need to constant color?

The Algorithm

For every pixel p in the composite image,

given

- backing color $C_{k1} = [R_{k1}\ \ G_{k1}\ \ B_{k1}]$ at p,
- backing color $C_{k2} = [R_{k2}\ \ G_{k2}\ \ B_{k2}]$ at p,
- composite pixel color $C_1 = [R_1\ \ G_1\ \ B_1]$ at p, and
- composite pixel color $C_2 = [R_2\ \ G_2\ \ B_2]$ at p,

solve the system of 6 equations

$$R_1 = \alpha_o R_o + (1 - \alpha_o) R_{k1} \quad R_2 = \alpha_o R_o + (1 - \alpha_o) R_{k2}$$
$$G_1 = \alpha_o G_o + (1 - \alpha_o) G_{k1} \quad G_2 = \alpha_o G_o + (1 - \alpha_o) G_{k2}$$
$$B_1 = \alpha_o B_o + (1 - \alpha_o) B_{k1} \quad B_2 = \alpha_o B_o + (1 - \alpha_o) B_{k2}$$

for unknowns R_o, G_o, B_o, α_o

Triangulation Matting Examples

From Smith & Blinn's SIGGRAPH'96 paper

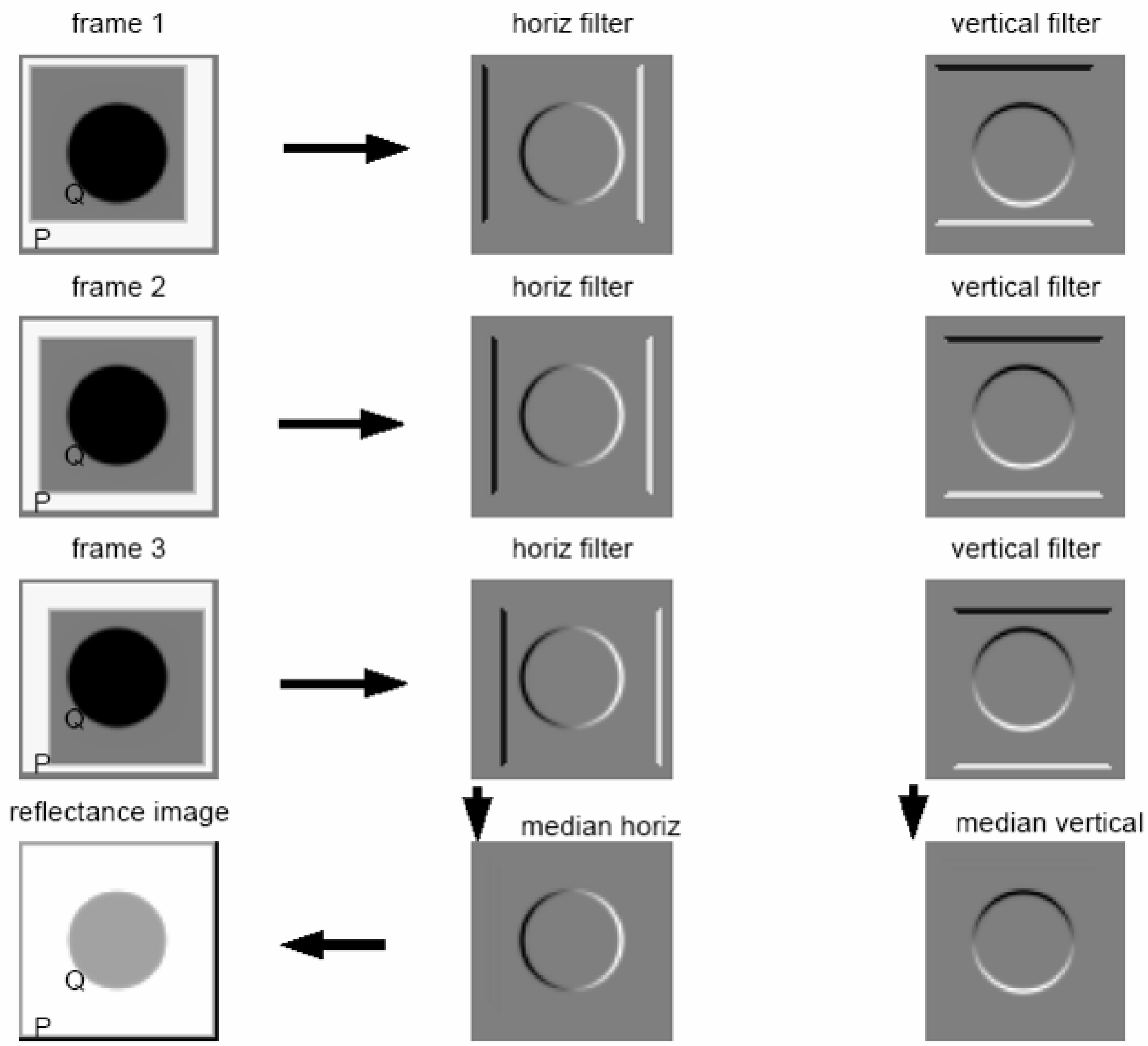
More Examples

More examples

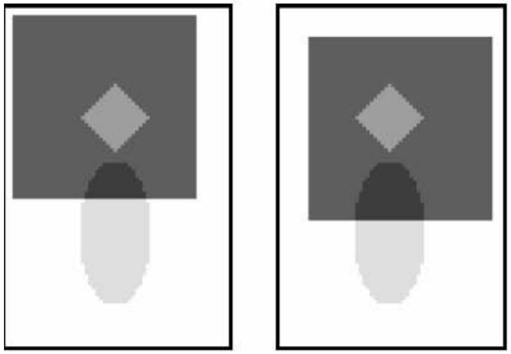
Removing Shadows (Weiss, 2001)

How does one detect (subtract away) shadows?

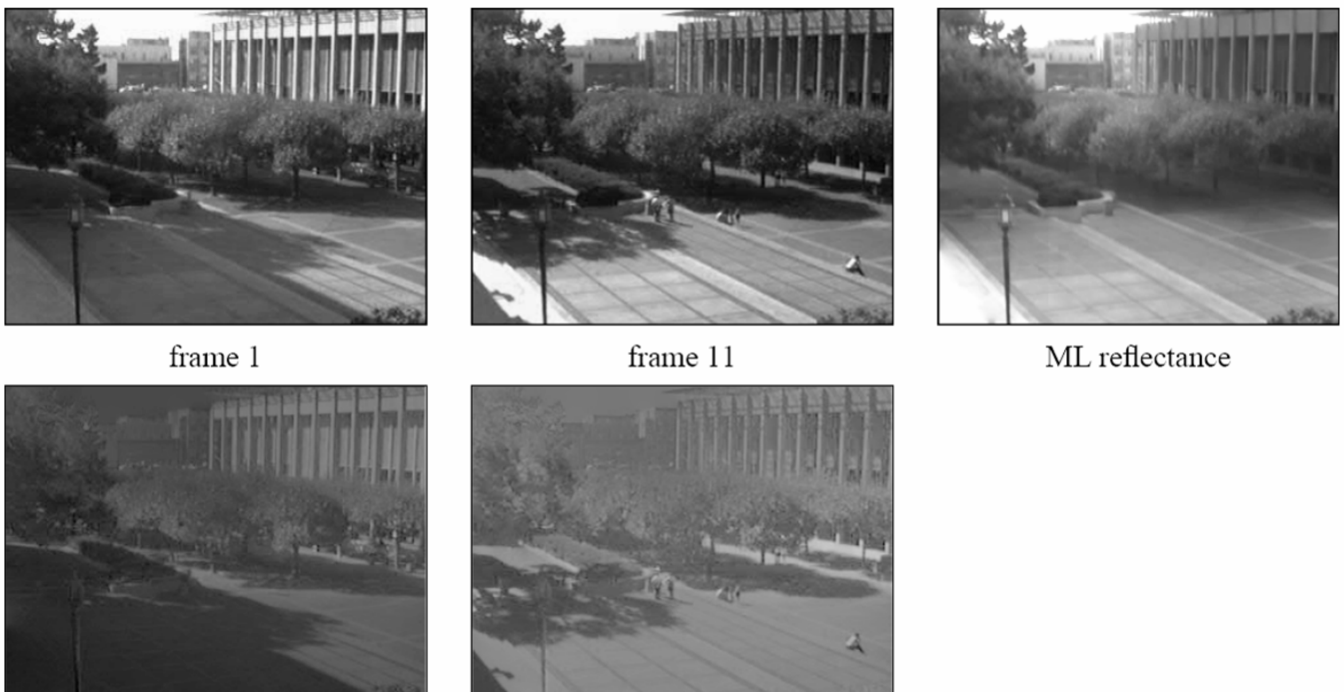
Averaging Derivatives



Recovering Shadows



first frame — last frame



frame 1 — frame 11 — ML reflectance

ML illumination 1 — ML illumination 2

Compositing with Shadows