Matting

Anat Levin, MIT CSAIL

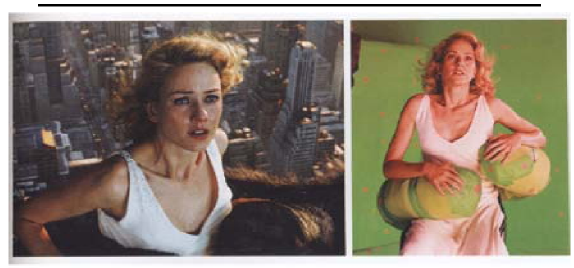
With some slides from Alexei Efros & Fredo Durand

How does Superman fly?

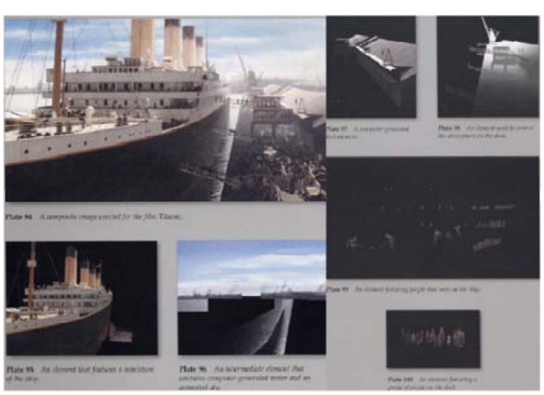
Super-human powers?
OR
Image Matting?

Motivation: compositing

Combining multiple images.
Typically, paste a foreground object onto a new background



From Cinefex



From the Art & Science of Digital Compositing

Page layout, magazine covers

Photo editing

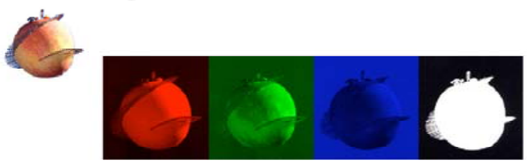
- Edit the background independently from foreground

Photo editing

- Edit the background independently from foreground

Alpha

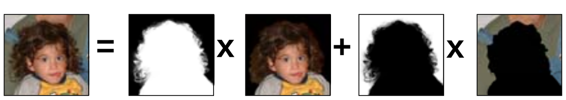
- α: 1 means opaque, 0 means transparent
- 32-bit images: R, G, B, α



Matting and compositing

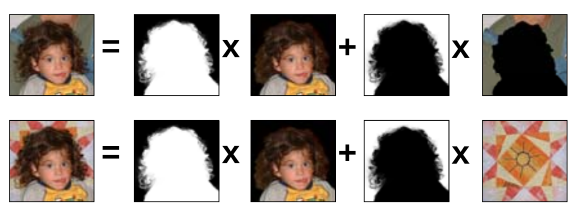
The matting equations

$$I_i = \alpha_i F_i + (1 - \alpha_i) B_i$$



Replace background

$$I_i = \alpha_i F_i + (1 - \alpha_i) B_i$$



Review: alpha channel

Add one more channel:

- Image(R,G,B,alpha) ⇐ *Sprite!*

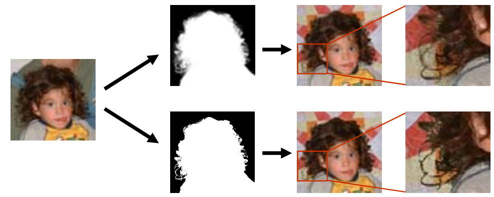
Encodes transparency (or pixel coverage):

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

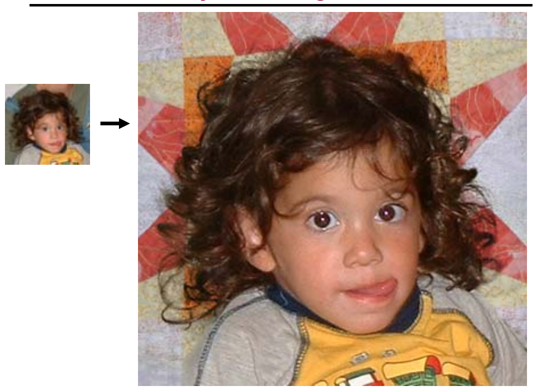
Example: alpha = 0.7

Partial coverage or semi-transparency

Why is matting hard?

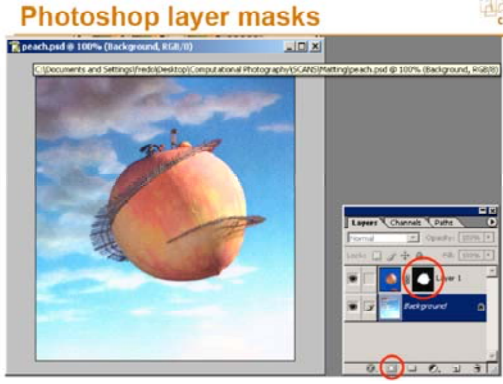


Why is matting hard?



Why is matting hard?

Photoshop layer masks



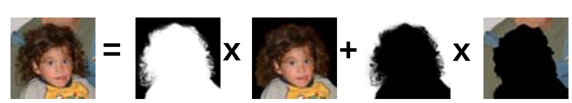
How many equations? How many unknowns?

$$I_i = \alpha_i F_i + (1-\alpha_i) B_i$$

$$I^R{}_i = \alpha_i F_i^R + (1-\alpha_i) B_i^R$$

$$I^G{}_i = \alpha_i F_i^G + (1-\alpha_i) B_i^G$$

$$I^B{}_i = \alpha_i F_i^B + (1-\alpha_i) B_i^B$$



Matting is ill posed: 7 unknowns but 3 constraints per pixel

“Pulling a Matte”

Problem Definition:

- The separation of an image I into
  - A foreground object image F,
  - a background image B,
  - and an alpha matte α
- F 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

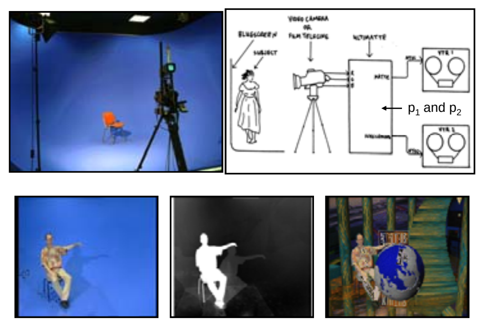
Most common form of matting in TV studios & movies

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,B_b) > threshold
  - Or use Vlahos' formula: $\alpha = 1-p_1(C_b-p_2C_g)$
- Hope that foreground object doesn't look like background
  - no blue ties!
- Why blue?
- Why uniform?

The Ultimatte



Blue screen for superman?

Solution #1: No Blue!

The matting eq: $I_i = \alpha_i F_i + (1-\alpha_i) B_i$

Background is known: $B^R = 0, B^G = 0, B^B = 1$

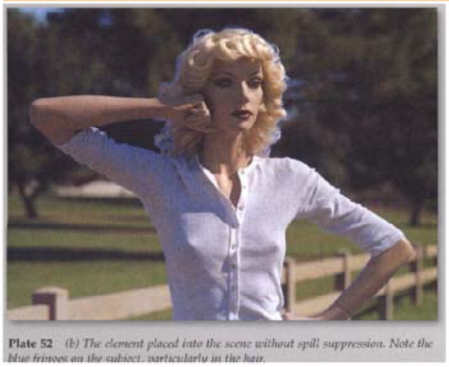
Assumption: $F^B = 0$

Now only 3 unknowns!

$$I^B = \alpha 0 + (1-\alpha)1 \quad \Rightarrow get\ \alpha$$

$$\begin{aligned} I^R{}_i &= \alpha_i F_i^R + (1-\alpha_i)0 \\ I^G{}_i &= \alpha_i F_i^G + (1-\alpha_i)0 \end{aligned} \quad \Rightarrow get\ F^R, F^G$$

Blue/Green screen matting issues



Triangulation Matting (Smith & Blinn)

Instead of reducing the number of unknowns, we could attempt increase the number of equations

One way to do this is to photograph the object of interest in front of 2 known and distinct backgrounds

How many equations?

How many unknowns?

Does the background need to be constant color?

The Algorithm

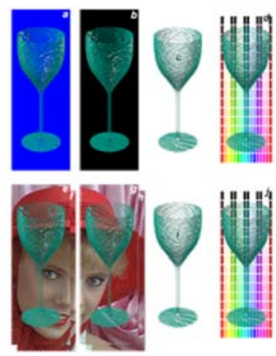
For every pixel: $(B^{R_1}, B^{G_1}, B^{B_1}),\ (B^{R_2}, B^{G_2}, B^{B_2}) - known$

$$I^{R_1} = \alpha F^R + (1-\alpha) B^{R_1}$$
$$I^{G_1} = \alpha F^G + (1-\alpha) B^{G_1}$$
$$I^{B_1} = \alpha F^B + (1-\alpha) B^{B_1}$$
$$I^{R_2} = \alpha F^R + (1-\alpha) B^{R_2}$$
$$I^{G_2} = \alpha F^G + (1-\alpha) B^{G_2}$$
$$I^{B_2} = \alpha F^B + (1-\alpha) B^{B_2}$$

Solve a system of 6 equations in 4 unknowns

Triangulation Matting Examples

From Smith & Blinn's SIGGRAPH'96 paper



More Examples

More examples

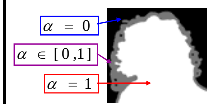
Natural image matting

The rules:

Only 1 input image is given (e.g. downloaded from the web), we have no control over the background

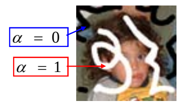
User can help, but want to minimize user work

User interfaces



The trimap interface:

- **Bayesian Matting (Chuang et al, CVPR01)**
- **Poisson Matting (Sun et al SIGGRAPH 04)**
- **Random Walk (Grady et al 05)**



Scribbles interface:

- **Wang&Cohen ICCV05**
- **Levin et al CVPR06**

Trimap based algorithms

Assumptions: the trimap is narrow.

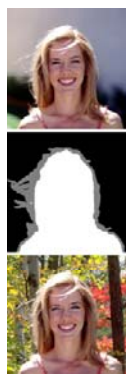
Thus we could guess F,B values in the mixed region by copying colors from neighboring F,B pixels

Given F,B solve for α

Use α to refine F,B estimate

Use F,B estimate to refine α estimate

and so on



Problems with trimap based approaches

- **Iterate between solving for F,B and solving for α**
- **Accurate trimap required**

(Replotted from Wang&Cohen)

A closed form solution to natural images matting

Anat Levin, Dani Lischinski and Yair Weiss

Presented at CVPR 2006

$$I_i = \alpha_i F_i + (1 - \alpha_i) B_i$$

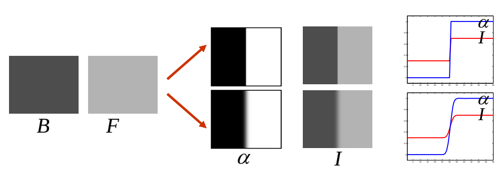
- **Analytically eliminate F,B. Obtain quadratic cost in α**
- **Provable correctness result**
- **Quantitative evaluation of results**

The matte as a linear function of intensity

Assume F,B are approximately constant in a window:

$$I_i \approx \alpha_i F + (1 - \alpha_i) B \qquad i \in w$$

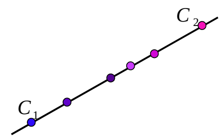
$$\alpha_i \approx a I_i + b \qquad a = \frac{1}{F - B}, b = \frac{-B}{F - B}$$



Color lines

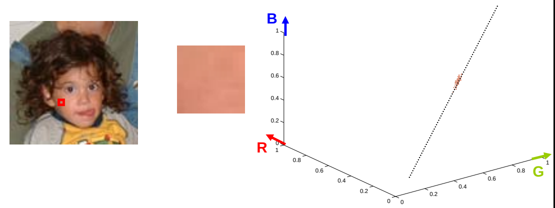
Color Line: $\left\{C_i \in R^3 \middle| C_i = \beta_i C_1 + (1-\beta_i) C_2\right\}$

(Omer&Werman 04)



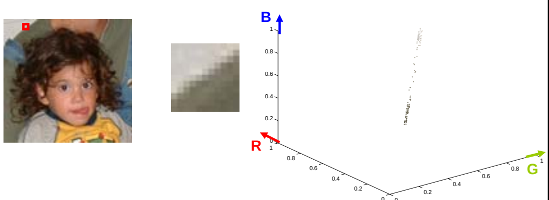
Color lines

Color Line: $\left\{C_i \in R^3 \middle| C_i = \beta_i C_1 + (1-\beta_i) C_2\right\}$



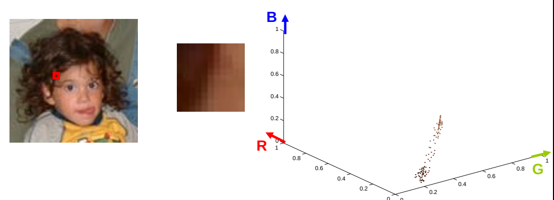
Color lines

Color Line: $\left\{C_i \in R^3 \middle| C_i = \beta_i C_1 + (1-\beta_i) C_2\right\}$



Color lines

Color Line: $\left\{C_i \in R^3 \middle| C_i = \beta_i C_1 + (1-\beta_i) C_2\right\}$



Linear model from color lines

Observation:

If the F,B colors in a local window lie on a color line, then

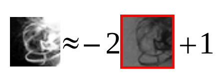
$$\alpha_i = a^R R_i + a^G G_i + a^B B_i + b \qquad \forall i \in w$$

Linear relation- 1 channel case

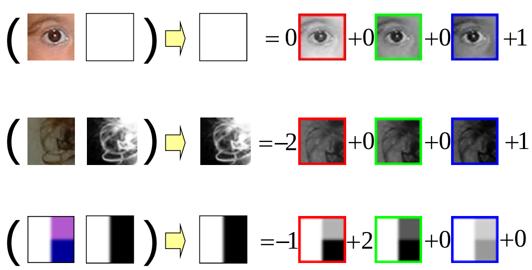
Assume F,B are approximately constant in a window:

$$I_i \approx \alpha_i F + (1-\alpha_i) B \qquad i \in w$$

$$\alpha_i \approx a I_i + b \qquad a = \frac{1}{F-B}, b = \frac{-B}{F-B}$$



Examples for linear relations

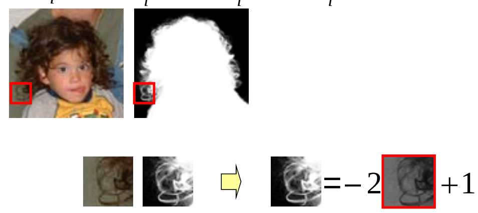


Linear model from color lines

Observation:

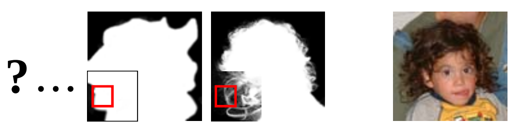
If the F,B colors in a local window lie on a color line, then

$$\alpha_i = a^R R_i + a^G G_i + a^B B_i + b \qquad \forall i \in w$$

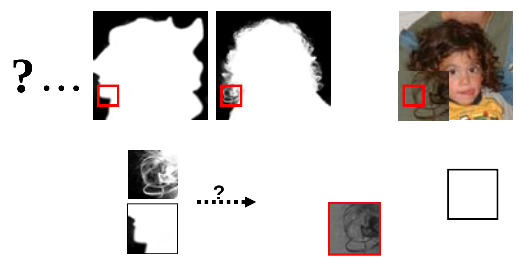


Result: **F,B can be eliminated from the matting cost**

Evaluating an α-matte



Evaluating an α-matte



Evaluating an α-matte

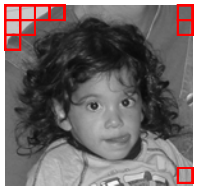
$$J(\alpha) = \sum_{w \in I} d\left(\alpha_w, Span\{R_w, G_w, B_w, 1\}\right) + \varepsilon \cdot smoothness(\alpha)$$

Evaluating an α-matte, 1 channel case

Minimize:

$$\sum_{j \in I} \sum_{i \in w_j} (a_j I_i + b_j - \alpha_i)^2$$

Evaluating an α-matte, 1 channel case



Minimize:

$$\sum_{j\in I}\sum_{i\in w_j}(a_j I_i + b_j - \alpha_i)^2$$

Theorem

F,B locally on color lines

$$J(\alpha)=\sum_{w\in I} d\left(\alpha_w, Span\{R_w,G_w,B_w,1\}\right)$$

$$=\alpha^T L\alpha$$

Where $L(i,j)$ **local function of the image**

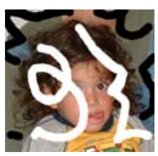
$$L(i,j)\propto \sum_{k|(i,j)\in w_k} -\left(1+(C_i-\mu_k)^T(\Sigma_k+\varepsilon I_3)^{-1}(C_j-\mu_k)\right)$$

Solving for α using linear algebra

Input:

Image+ user scribbles

$$\alpha = \arg\min \alpha^T L\alpha$$
$$s.t.\ \alpha_i = 0,\quad i\in$$
$$\alpha_i = 1,\quad i\in$$



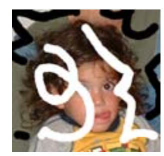
$$L(i,j)\propto \sum_{k|(i,j)\in w_k} -\left(1+(C_i-\mu_k)^T(\Sigma_k+\varepsilon I_3)^{-1}(C_j-\mu_k)\right)$$

Solving for α using linear algebra

Input:

Image+ user scribbles

$$\alpha = \arg\min \alpha^T L\alpha$$
$$s.t.\ \alpha_i = 0,\quad i\in$$
$$\alpha_i = 1,\quad i\in$$



Advantages:

- **Quadratic cost- global optimum**
- **Solve efficiently using linear algebra**
- **Provable correctness**
- **Insight from eigenvectors**

Cost minimization and the true solution

Theorem:

Given: $I = \alpha^* F^* + (1-\alpha^*)B^*$

If:

- F^*,B^* **locally on color lines**
- **Constraints consistent with** α^*

Then:

$$\alpha^* = \arg\min \alpha^T L\alpha$$
$$s.t.\ \alpha_i = 0,\quad i\in$$
$$\alpha_i = 1,\quad i\in$$

Matting and spectral segmentation

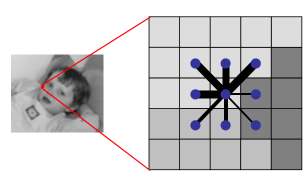
Spectral segmentation: Analyzing smallest eigenvectors of a graph Laplacian L **(E.g. Normalized Cuts, Shi&Malik 97)**

$$L = D - W$$

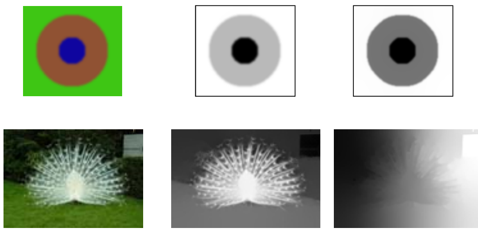
$$D(i,i)=\sum_j W(i,j)$$

$$W_{Global}(i,j)=e^{-\|C_i-C_j\|^2/\sigma^2}$$

$$W_{Matting}(i,j)\propto \sum_{k|(i,j)\in w_k}\left(1+(C_i-\mu_k)^T(\Sigma_k+\varepsilon I_3)^{-1}(C_j-\mu_k)\right)$$



Comparing eigenvectors



Input image

Matting Eigenvectors

Global-σ Eigenvectors

Eigenvectors as guides

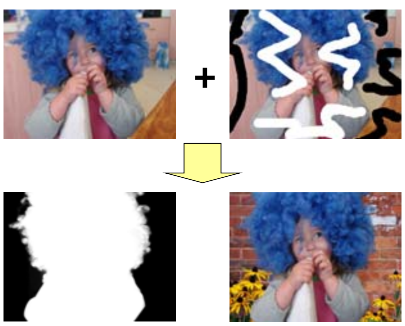
Input Image

Eigenvectors analyzing

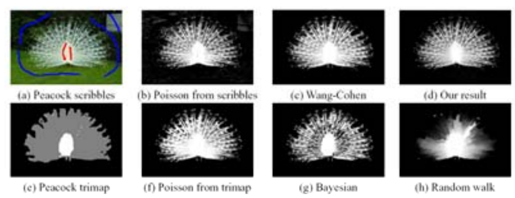
Scribbling

Matte

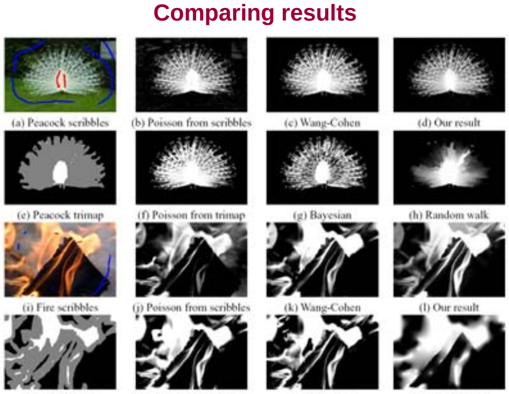
Matting results



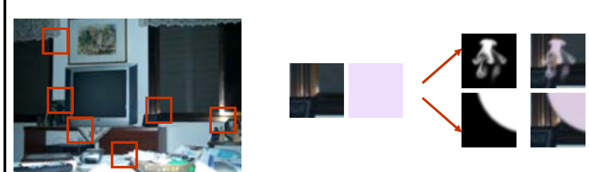
Comparing results



Comparing results



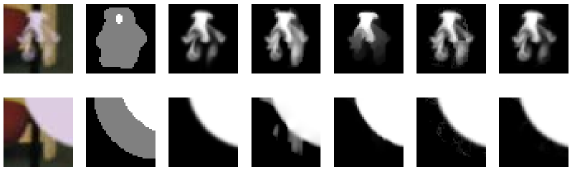
Quantitative results



Experiment Setup:

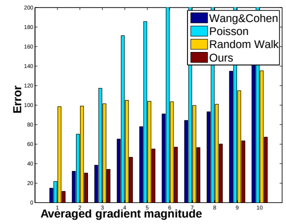
- Randomize 1000 windows from a real image
- Create 2000 test images by compositing with a constant foreground using 2 different alpha mattes
- Use a trimap to estimate mattes from the 2000 test images, using the different algorithms
- Compare errors against ground truth

Quantitative Results

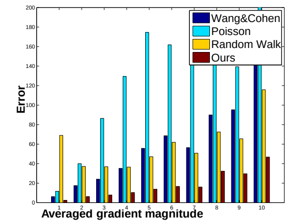


Input | Trimap | Ground Truth Alpha | Poisson | Random Walk | Wang& Cohen | Ours

Quantitative results



Smoke Matte



Circle Matte

Conclusions

- **Analytically eliminate F,B and obtain quadratic cost $\alpha^T L \alpha$. Solve efficiently using linear algebra.**
- **Provable correctness result.**
- **Connection to spectral segmentation.**
- **Quantitative evaluation.**

Code available:
http://www.cs.huji.ac.il/~alevin/matting.tar.gz

Environment Matting and Compositing

slides by Jay Hetler

Douglas E. Zongker ~ Dawn M. Werner ~ Brian Curless ~ David H. Salsin

Environment Matting Equation

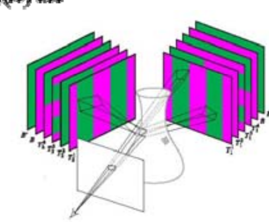
$$C = F + (1-\alpha)B + \Phi$$

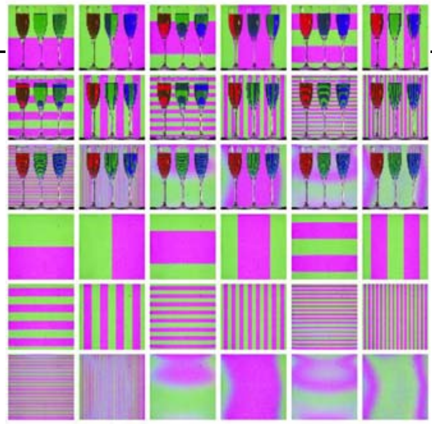
- C ~ Color
- F ~ Foreground color
- B ~ Background color
- α ~ Amount of light that passes through the foreground
- Φ ~ Contribution of light from Environment that travels through the object

Explanation of Φ

$$\Phi = \sum_{i=1}^{m} \int R_i(x)\, T_i(x)\, dx$$

R – reflectance image
T – Texture image





Environment Mattes

Performance

Calibration

Matting: 10-20 minutes extraction time for each texture map (Pentium II 400Mhz)

Compositing: 4-40 frames per second

Real-Time?

How much better is Environment Matting?

Alpha Matte **Environment Matte** **Photograph**

How much better is Environment Matting?

Alpha Matte **Environment Matte** **Photograph**

Movies!