Sistemi Multimediali

Lab project: Image/video compositing

Goal of the project

- **Experimenting with video compositing**
- **We will apply two key methods in compositing:**
  - **Alpha Blending**
  - **Chroma Keying**
- **Application to still images, but easily applicable to video frames (which are still images indeed)**

Motivation of this work

Compositing video tracks

- **Compositing is the technique which allows to combine two images into one**
- **In videos, compositing is performed frame by frame, compositing the corresponding images**

Alpha channel

- **RGB images encode each pixel using 8 bits for R, 8 for G and 8 for B → 24 bits/pixel**
- **Video frames and still images can be encoded using 32-bits/pixel: R, G, B + α**
- **α defines a «transparency» for each pixel of the image and can be exploited for combining pixels from different frames/images**
  - **α** = 1: opaque object (complete coverage)
  - **α** = 0: transparent object (no coverage)
  - 0 < **α** < 1: semi-transparent (partial coverage)
- **Two images can be combined together, according to their transparency value**

Alpha Blending

- **Alpha blending combines two images, according to the value of α, using all the pixel of the two images.**

- **To summarize, this operation implements one image «fading» to another one**

What is it useful for ???

Alpha Blending

Photo retouching


+

=


HDR Photography

Alpha blending

- **Given two RGB-α images(/frames), which can be (but not necessairily) considered:**
  - **a background image $B= (b_r, b_g, b_b, \alpha_b)$ and**
  - **a foreground image $F= (f_r, f_g, f_b, \alpha_f)$**

  With $0 \leq \alpha_b, \alpha_f \leq 1$

- **We obtain the composited (alpha blended) image C calculating each pixel as:**

$$C = (\alpha_f F + (1- \alpha_f) \alpha_b B) / \alpha'$$

With $\alpha' = \alpha_f + (1- \alpha_f) \alpha_b$

TASK 1

- **Create a class implementing Alpha blending over two images:**
  - **Receiving as input:**
    - **Image F and the value of α_f**
    - **Image B and the value of α_b**
    - **Image F and Image B should have the same size (for simplicity)**
  - **Returning as output a new image C, calculated as:**

$$C = (\alpha_f F + (1- \alpha_f) \alpha_b B) / \alpha'$$

With $\alpha' = \alpha_f + (1- \alpha_f) \alpha_b$

Chroma keying

Chroma key is the operation of superimposing an image F (foreground) over an image B (background), eliminating pixels of (around) a certain color from the foreground F

The basic technique derives from the theory of alpha blending

Chroma keying

- **Given:**
  - **a foreground image F (with blue or green key)**
  - **a background image B**

- **The final image C is obtained as:**

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

- **Where:**
  - **α= 0 for the pixels in F corresponding to the blue or green key (with a certain tolerance)**
  - **α> 0 for the other pixels**

Chroma keying

- **Different techniques assign a value α to the different pixels using different approaches**
- **Some of them work in the HSL color space**
  - **Easy and effective but conversion RGB → HSL not straightforward**
- **Other work directly on RGB color space**
  - **Easy methods do not always offer the best results**
  - **Best methods involve some calculation**

Chroma keying

- **Method 1: Petro Vlahos (1964)**

- **From experience and observation, best results achieved when the B components dominate the R and G components. The α value is calculated accordingly:**

$$\alpha = 1 - (F_b - MAX(F_r, F_g))$$

- Formerly used for analogic films; later adapted for computer programs.

- *1994: Lifetime achievement award by the Academy of Motion Picture Arts and Sciences*

Task 2

- **Create a class implementing Chroma Keying over two images, using the Vlaho method:**
  - **Receiving as input:**
    - **Image F (with green or blue key)**
    - **Image B**
    - **Image F and Image B should have the same size (for simplicity)**
  - **Returning as output a new image C, calculated as:**

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

Where the value of α for each pixel has the value:

$$\alpha = 1 - (F_b - MAX(F_r, F_g))$$

Implementation and final relation

- **After the implementation of tasks 1 and 2, try the two methods (and the optional part, if you decide to implement is, see the following) with an experiment session, showing the results of the methods on at least 5-10 experiments**

Implementation and final relation

- **Relation:**
  - **a short (5-10 pgs. about) presentation of your library and of the choices you made**

- **When everything is ready, pack the code, the relation and the test images and send them to giorgio.leonardi@uniupo.it for their evaluation and discussion**

Grade

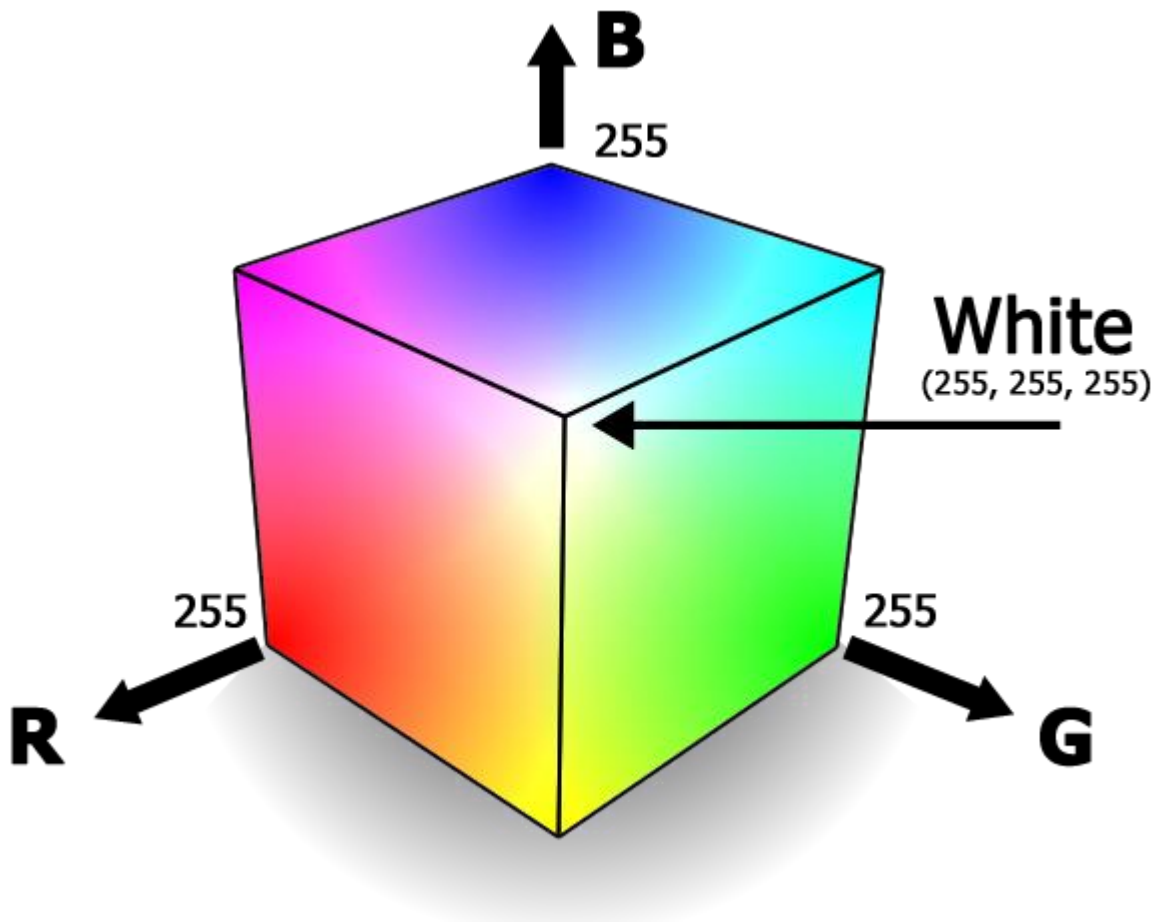
- **With the «standard» work (no optional part) and a good relation, you will obtain a maximum of 28 points (or it will confirm your theory grade, if it is > 28 EXCEPT the laude!)**

- **Add two or more points (laude!) with a correct optional part**

OPTIONAL!

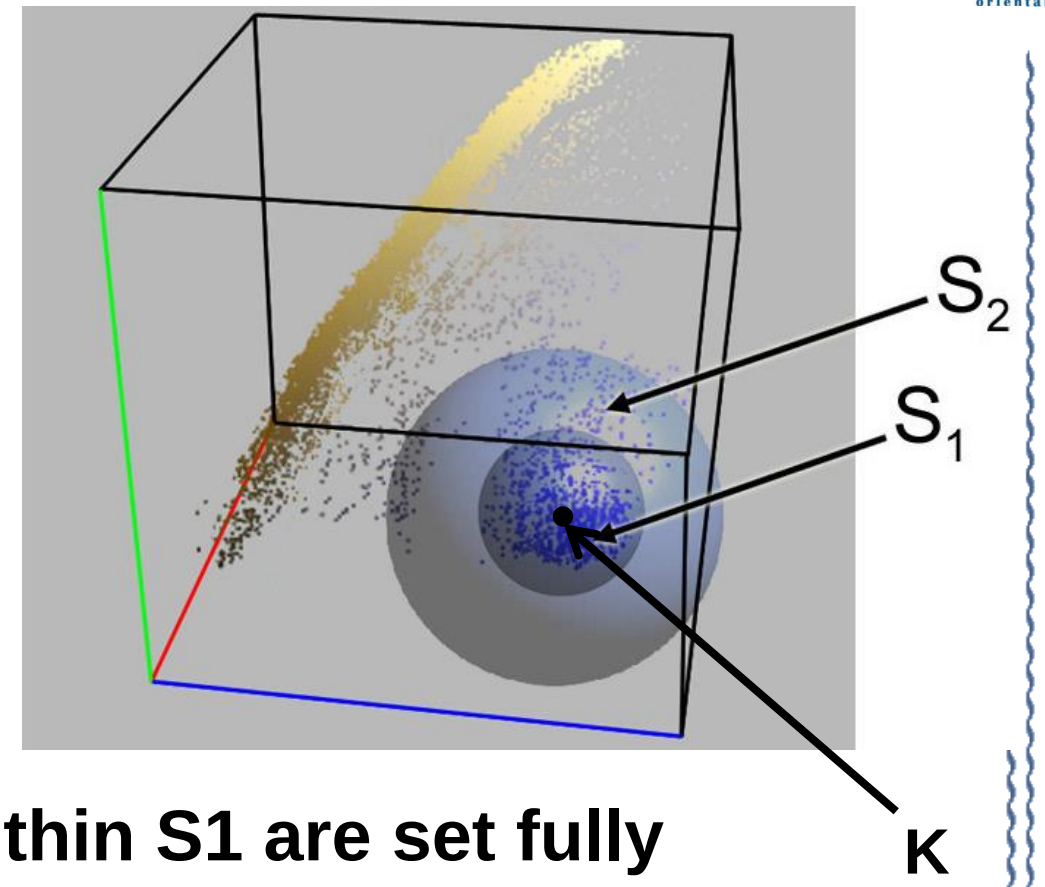
3D keying

- **AKA Primatte Keying – works in the RGB space using the RGB color cube:**



3D keying

- We define:
  - A key color K
  - A keying sphere S1
  - A tolerance (fading) sphere S2



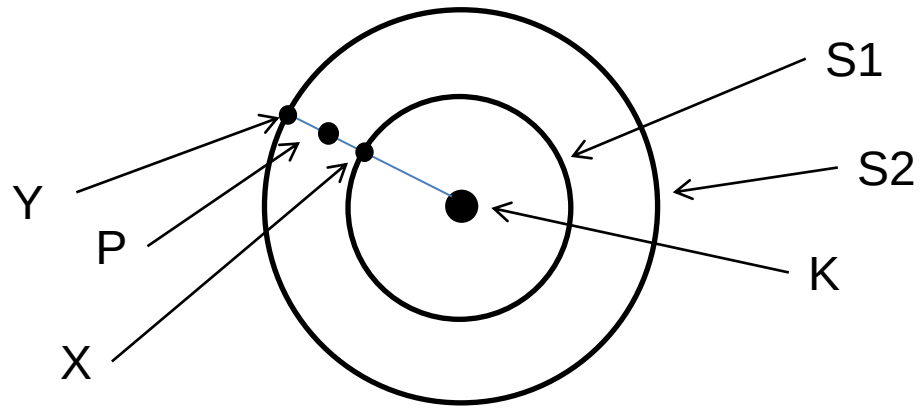
- All pixels within S1 are set fully invisible.
- The coordinates of X can be created by averaging background pixel values.

- Pixels of F, located within S1 are set fully transparent;
- pixels of F outside S2 are fully visible, and
- pixels of F within S2 but outside S1 are semi-transparent.

3D keying

- **Given:**
  - **A key color K = (R_k, G_k, B_k) as center of the spheres S1 and S2**
  - **A keying sphere S1 with radius R1**
  - **A tolerance (fading) sphere S2 with radius R2**

- **Each pixel P_f = (R_f, G_f, B_f) in the image F will be assigned with a value of α in the following way:**
  - **If P_f falls inside the sphere S1, then α= 0**
  - **If P_f falls outside the sphere S2, then α= 1**
  - **Else α = $\frac{|\overline{XP}|}{|\overline{XY}|}$**



Optional

- **Create a class implementing 3D Chroma Keying over two images, using the Vlaho method:**
  - **Receiving as input:**
    - **Image F (with green or blue key)**
    - **Image B**
    - **Image F and Image B should have the same size (for simplicity)**
    - **Key color K, radius R1 and radius R2**
  - **Returning as output a new image C, calculated as:**

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