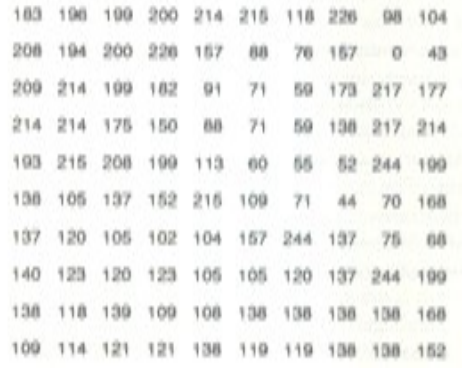
INF555

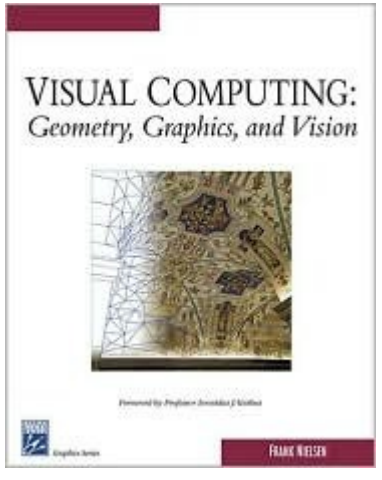
Fundamentals of 3D



Lecture 1:
Abstract Data Structures

Stacks, queues and hashing with...
...applications in visual computing

Mercredi 14 Septembre 2011



INF555: Fundamentals of 3D

- L1. Abstract data structures
- L2. Transformations and coordinate pipelines
- L3. Advanced transformations
- L4. Images and colors
- L5. Meshes
- L6. Animation
- L7. Randomized algorithms
- L8. High-dimensions for 3D
- L9. Robustness

→ Examen 20 min oral en Décembre

Débouchés de la filière Image (4A)

··Formation complementaire intégrée (FCI) avec l'··École des Ponts ParisTech
Master II ··Mathématiques / Vision / Apprentissage (MVA)

Master II à l'··École Polytechnique Fédérale de Lausanne (EPFL), Suisse :
Master II d'informatique/ Master II de robotique et systèmes autonomes
··Télécom ParisTech (ENST) :
+ Master ··spécialité imagerie du vivant (IMA)
+ Master ··Intelligence Artificielle et Décision (IAD) à UPMC

Cadre du Corps de l'Armement (filière recherche) :
··Royal Institute of Technology (KTH, Stockholm, Suéde.)

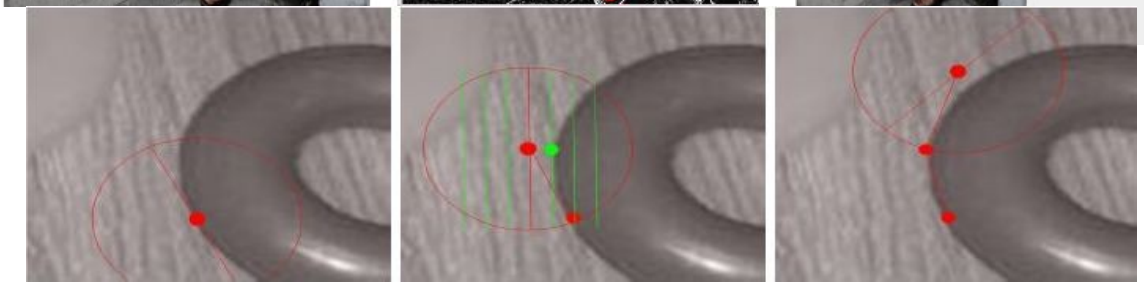
Master à l'étranger (notamment aux Etats-Unis et au Japon) :
Master à Columbia University, USA en infographie
Master en Electrical and Computer Engineering, Cornell University, USA

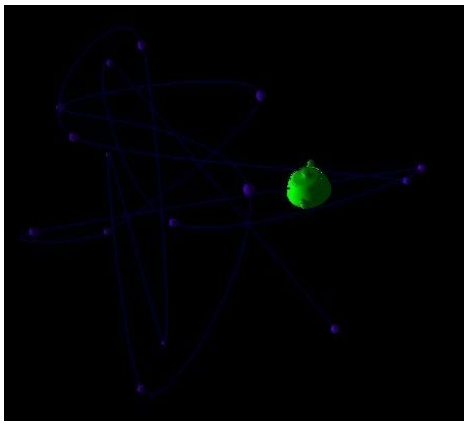
Master II ··Master Parisien de Recherche en Informatique (MPRI)
Master II ··Mathématiques / Vision / Apprentissage (MVA)
··ENSIMAG
··ENSTA
Masters européens:
Master ··ETH Zürich ··TU Munchen, Computer Science Dept.
Master in ··Biomedical Engineering à Oxford
· Masters internationaux : États-Unis / Japon
· etc.

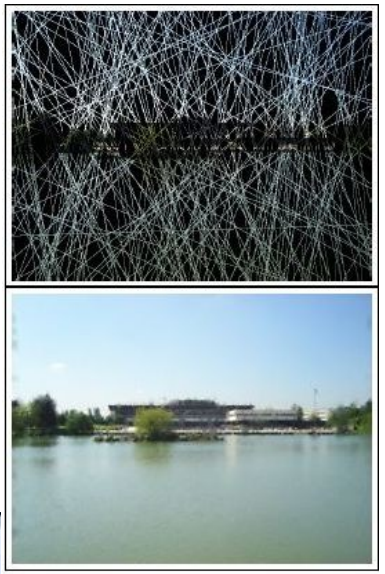
http://www.lix.polytechnique.fr/~nielsen/INF555/devenir.html

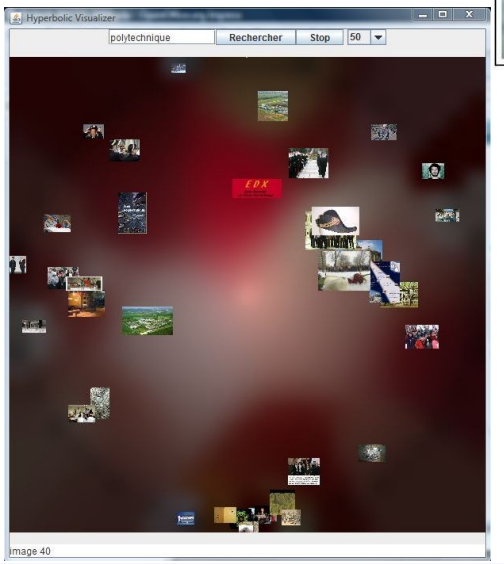
Notations et projets

- 30% rendus des programmes TDs
- 20% controle de connaissance par questions orales (lors de la soutenance du projet)
- 50% soutenance de projet







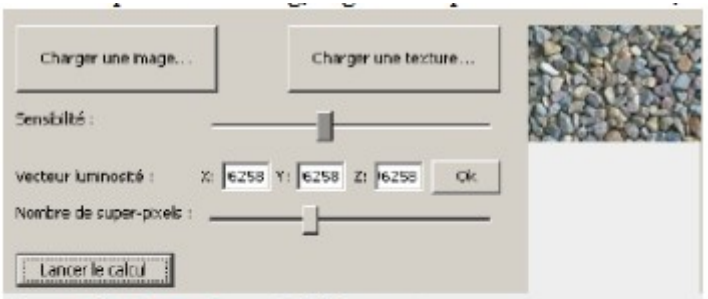


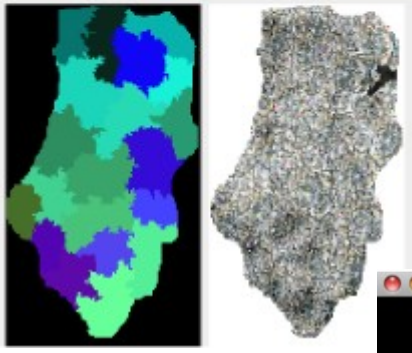
http://www.lix.polytechnique.fr/Labo/Frank.Nielsen/INF555/HallFamePj2008/HF2008-Pj.html

http://www.lix.polytechnique.fr/~nielsen/INF555/index.html

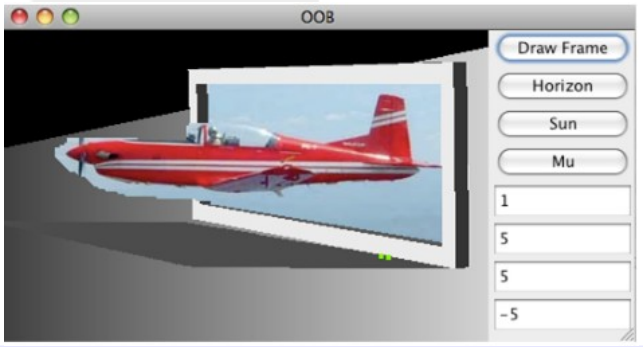
Dehazing
(anti-brouillard)

Stippling
(pointillisme)

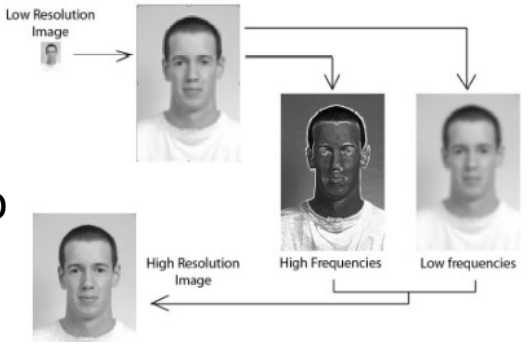




TextureShop
(retexturé les photos)



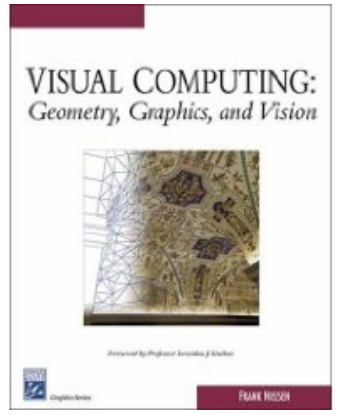
Out of bound photo
(effet photo 3D)



Face recognition/compression
(algorithmique du visage)

http://www.lix.polytechnique.fr/Labo/Frank.Nielsen/INF555/HallFamePj2009/

Reference textbook



Disponible à la bibliothèque en 5 exemplaires
Allez l'emprûnter !

Web supplements

Click on the chapter links below to access the C++/OpenGL(R) source codes.

Foreword by Prof. L. J. Guibas

1. Overview
2. Abstract Data Structures
3. Coordinate Pipelines
4. Images
5. Meshes
6. Animation
7. Randomization
8. Higher Dimensions for 3D
9. Robustness

Bibliography in PDF

View all source codes at once

Note for compiling programs.

Click on the chapter links on the left frame to access C++ programs and web supplements or here to get all source listings.

All GLUT OpenGL(R) source codes are now available with screenshots on this web page.

If you wish to receive further information concerning major updates of this site, you can register your email address :

Your E-mail: Subscribe to Visual Computing mailing list

Visual Computing: Geometry, Graphics, and Vision.

Frank Nielsen, Charles River Media / Thomson Delmar Learning, August 2005.

ISBN: 1-58450-427-7, Retail price 59.95 USD (currently discounted at Amazon.com at 35 USD, as of August 23rd 2005)

Hard cover, xiv+560 pages, 8-page color insert, 50+ C++ source codes (some in OpenGL®)

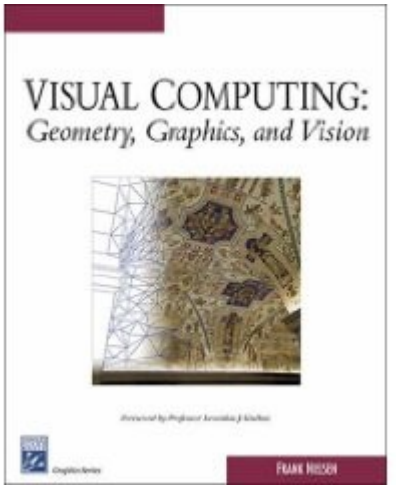
Rationale: [Information Tip Sheet]

Visual Computing: Geometry, Graphics, and Vision is a concise introduction to common notions, methodologies, data structures and algorithmic techniques arising in the mature fields of computer graphics, computer vision, and computational geometry. The central goal of the book is to provide a global and unified view of the rich interdisciplinary visual computing field that encompasses traditional computer graphics, computer vision, and computational geometry. The book is targeted at undergraduate students, and gaming or graphics professionals. Lectures in computer graphics/vision may find this textbook complementary and valuable. The book aims at broadening and fostering readers' knowledge of essential 3D techniques by providing a sizeable overall picture and describing essential concepts. Throughout the book, appropriate real world applications are covered to illustrate

Internet | Mode protégé : désactivé 100%

http://www.sonycsl.co.jp/person/nielsen/visualcomputing/

Abstract Data Structures



- Concepts encapsulated into an **interface**
- Multiple implementations (various trade-offs)

→ Good for code reuse (APIs):
→ Actual implementation is hidden

Outline of Abstract DS

- Abstract data-structures in Java
- Area Flood filling (← stacks and ← queues)
- Any segment pair intersection (← dictionary)
- Image segmentation (← union-find disjoint elements)
- Object recognition (← geometric hashing)

I. Abstract Data Structures: Stack

Interface (method prototypes):

isEmpty
push(Element)
pop (or pull)

For example, two implementations:

- Array
- Linked list

```
public static void main(String
[] args)
{
    Stack myStack=new Stack();
    int i;
    for(i=0;i<10;i++)
        myStack.Push(i);

    for(i=0;i<15;i++)

System.out.println(myStack.Pull(
));

}
```

Stack (array)

```
class StackArray
{
int nbmax;
int index;
int [ ] array;

// Constructors
StackArray(int n)
{
this.nbmax=n;
array=new int[nbmax]; index=-1;
System.out.println("Succesfully created a
stack array object...");
}

// Methods: INTERFACE
void Push(int element)
{
if (index<nbmax-1)
   array[++index]=element; }

int Pull()
{
if (index>=0 ) return array[index--];
else return -1;
}
```

Stack (linked list)

```
class Stack
{
List list;

Stack()
{
list=null;
}

void Push(int el)
{
if (list!=null)

list=list.insertHead(e
l);
       else
       list=new
List(el,null);
}

int Pull()
{int val;
if (list!=null)
    {val=list.element;
       list=list.next;}
       else val=-1;

return val;
}
}
```

```
class List
{
int element;
List next;

// Constructor
List(int el, List
tail)
{
this.element=el;
this.next=tail;
}

List
insertHead(int
el)
{
return new
List(el,this);
}

}
```

Java documentation of interfaces

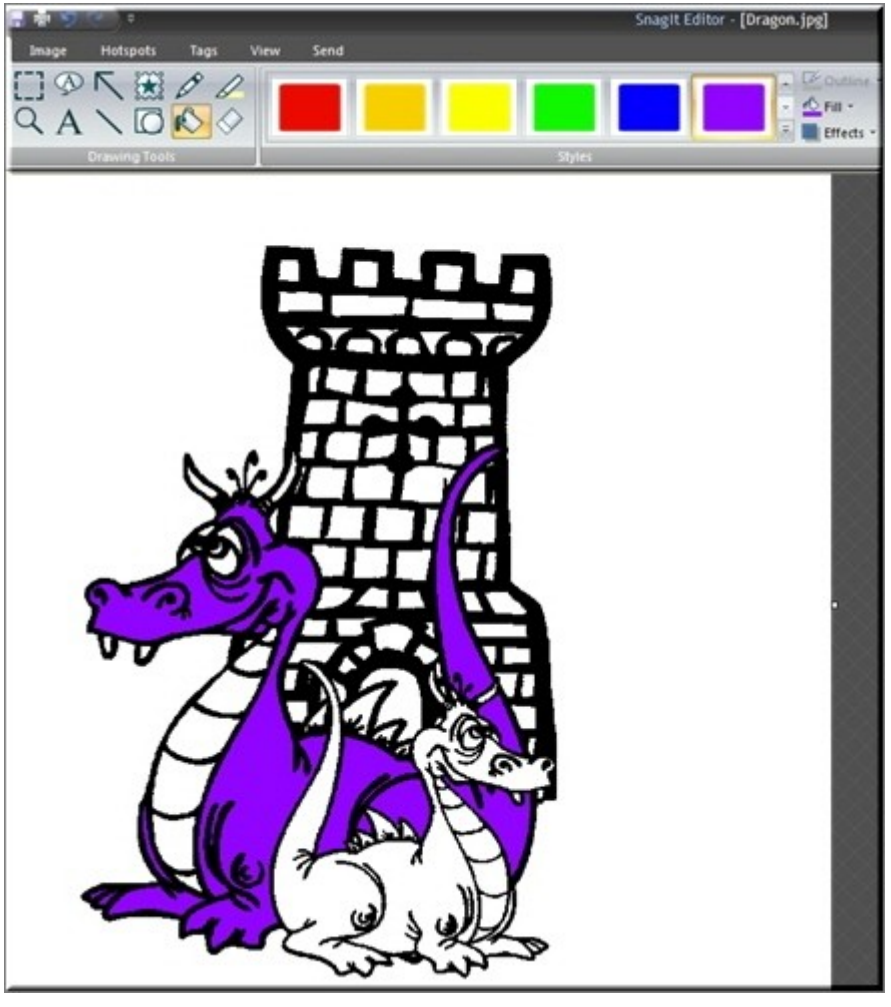
http://java.sun.com/j2se/1.4.2/docs/api/java/util/Stack.html

Method Summary	
boolean	empty() Tests if this stack is empty.
Object	peek() Looks at the object at the top of this stack without removing it from the stack.
Object	pop() Removes the object at the top of this stack and returns that object as the value of this function.
Object	push(Object item) Pushes an item onto the top of this stack.
int	search(Object o) Returns the 1-based position where an object is on this stack.

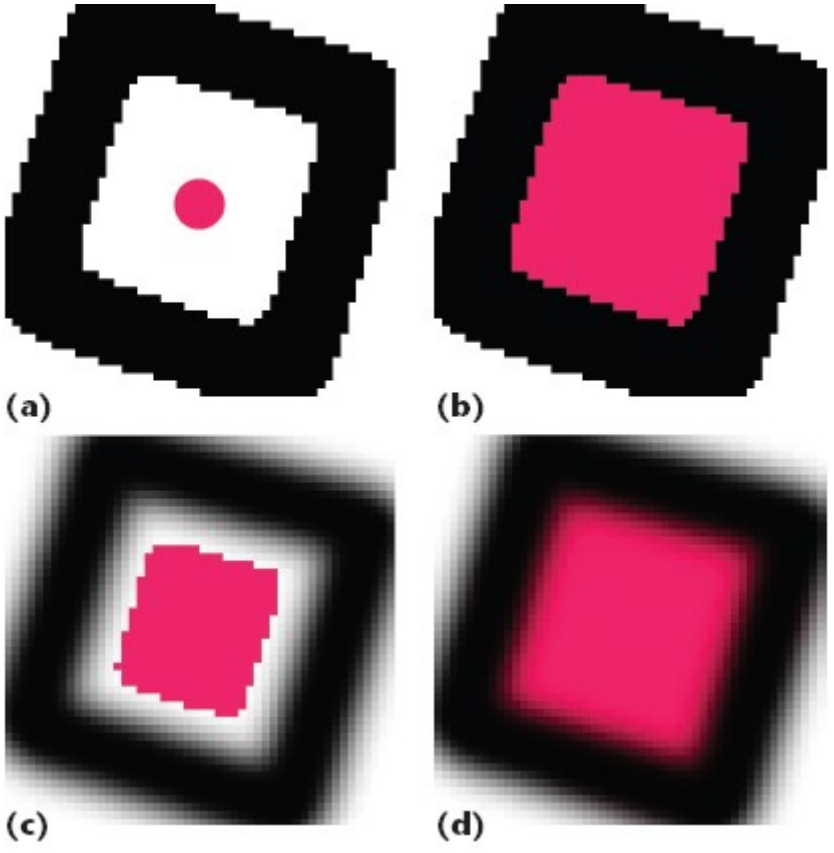
Commentez vos programmes avec JavaDoc (sous Eclipse)
ou Doxygen (incluant des formules latex par exemple):
http://www.stack.nl/~dimitri/doxygen/

II. Computer Graphics: Area Flood Filling

Basic coloring application

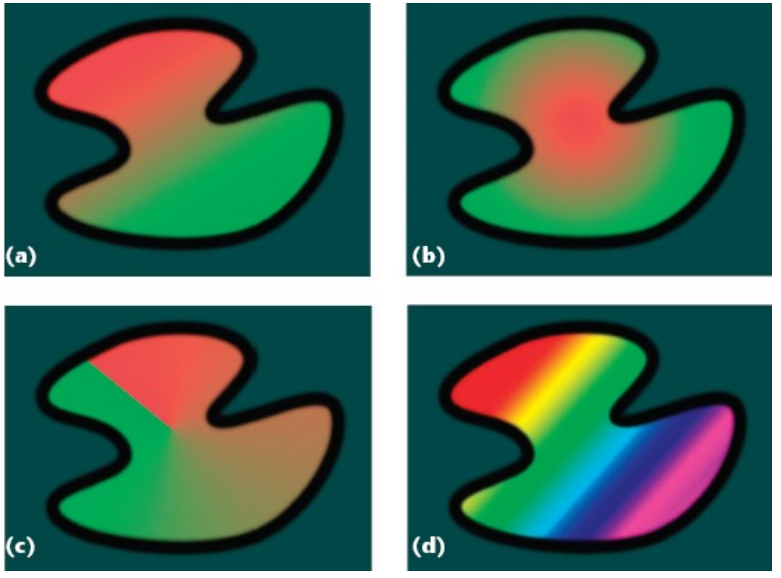


Area Flood Filling



Tint filling

Tint fill+pattern fill



Gradient filling

Application: Vectorization of Cartoon Animation

→ Rendering to multi-size displays
→ Interaction with image semantic

Vectorizing Cartoon Animations
July/August 2009 (vol. 15 no. 4)
pp. 618-629

(a) Composition of two clips

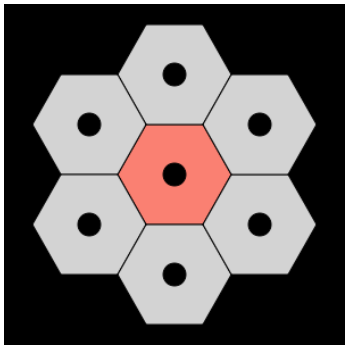
(b) Object Editing

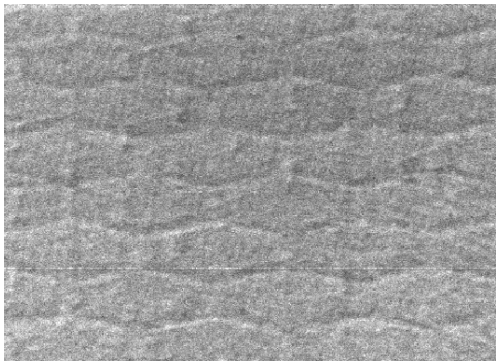
Fig. 10. Editing: (a) taking a character from one scene and putting him in another, (b) changing the shape of the character's ears

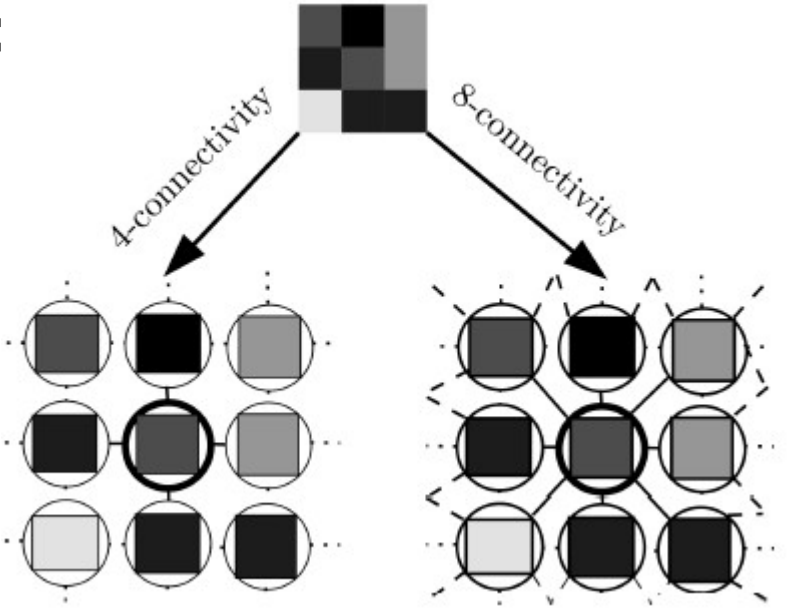
Area Flood Filling

Images as large graphs (million pixels)

Image (pixel) connectivity: 4- or 8-connectivity
(but also hexagonal, etc.)







Example of flood filling (cartoon animation, etc.)

http://www-graphics.stanford.edu/~levoy/sands_award.html

Area Flood Filling: Recursive Algorithm

```
FloodFillRecursive(I, p, C_B, C_F)
1.  I[p] ← C_F
2.  for q ∈ C_4(p)
3.      do
4.          if I[q] = C_B
5.              then FloodFillRecursive(I, q, C_B, C_F)
```

Use an abstract data-structure: Stack

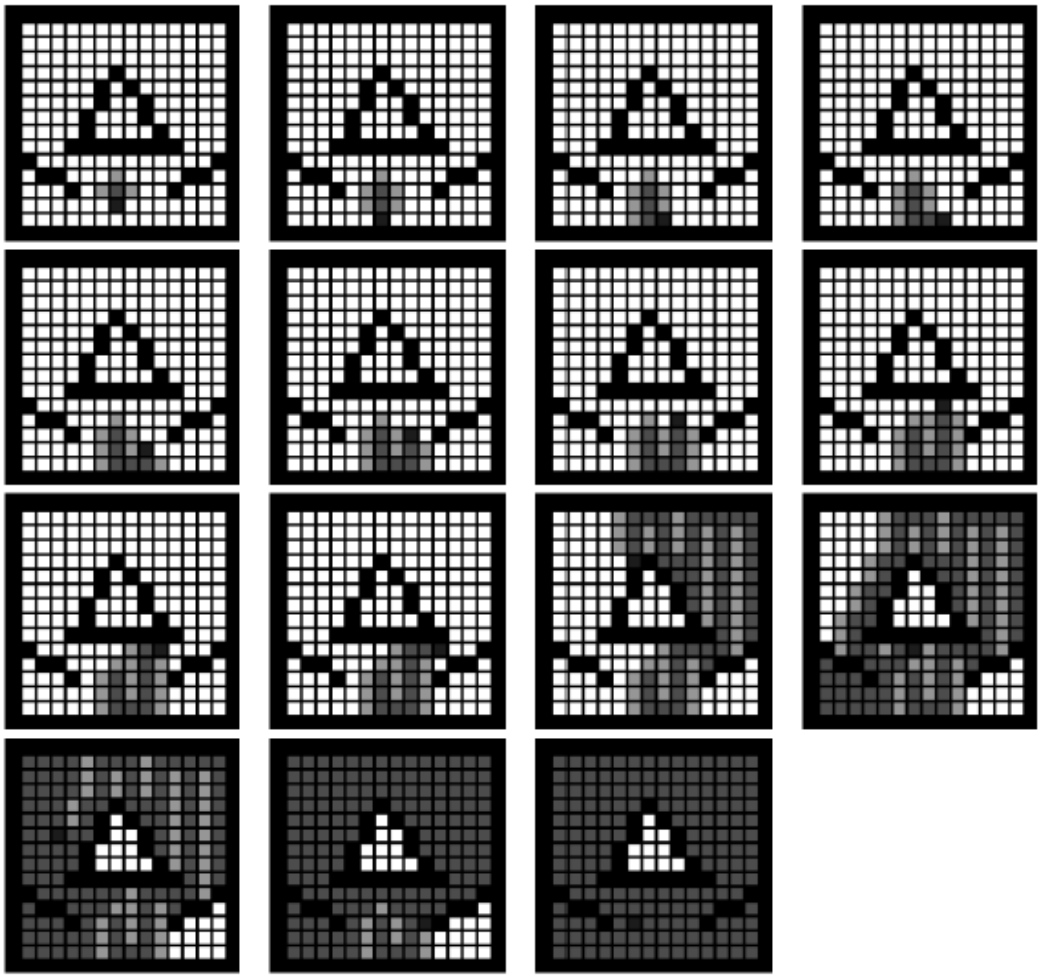
- $\text{new}() \rightarrow \emptyset$ returns an empty stack,
- $\text{pop}(\text{push}(\mathsf{S}, e)) \rightarrow \mathsf{S}$,
- $\text{top}(\text{push}(\mathsf{S}, e)) \rightarrow e$.

- $\text{isEmpty}(\text{new}()) \rightarrow \text{true}$,
- $\text{isEmpty}(\text{push}(\mathsf{S}, e)) \rightarrow \text{false}$,
- $\text{isEmpty}(\text{pop}(\text{push}(\mathsf{S}, e)) \rightarrow \text{isEmpty}(\mathsf{S})$.

Area flood filling using a stack

Handle recursion **explicitly** with the stack

```
FLOODFILLSTACK(I, p, C_B, C_F)
 1.  ◁ S is a stack data structure ▷
 2.  S ← ∅
 3.  I[p] ← C_F
 4.  S.push(p)
 5.  while S ≠ ∅
 6.      do
 7.          p ← S.pop()
 8.          for q ∈ C_4(p)
 9.              do
10.                  ◁ 4-neighborhood pixels ▷
11.                  if I[q] = C_B
12.                     then I[q] = C_F
13.                          S.push(q)
```



Depth first order

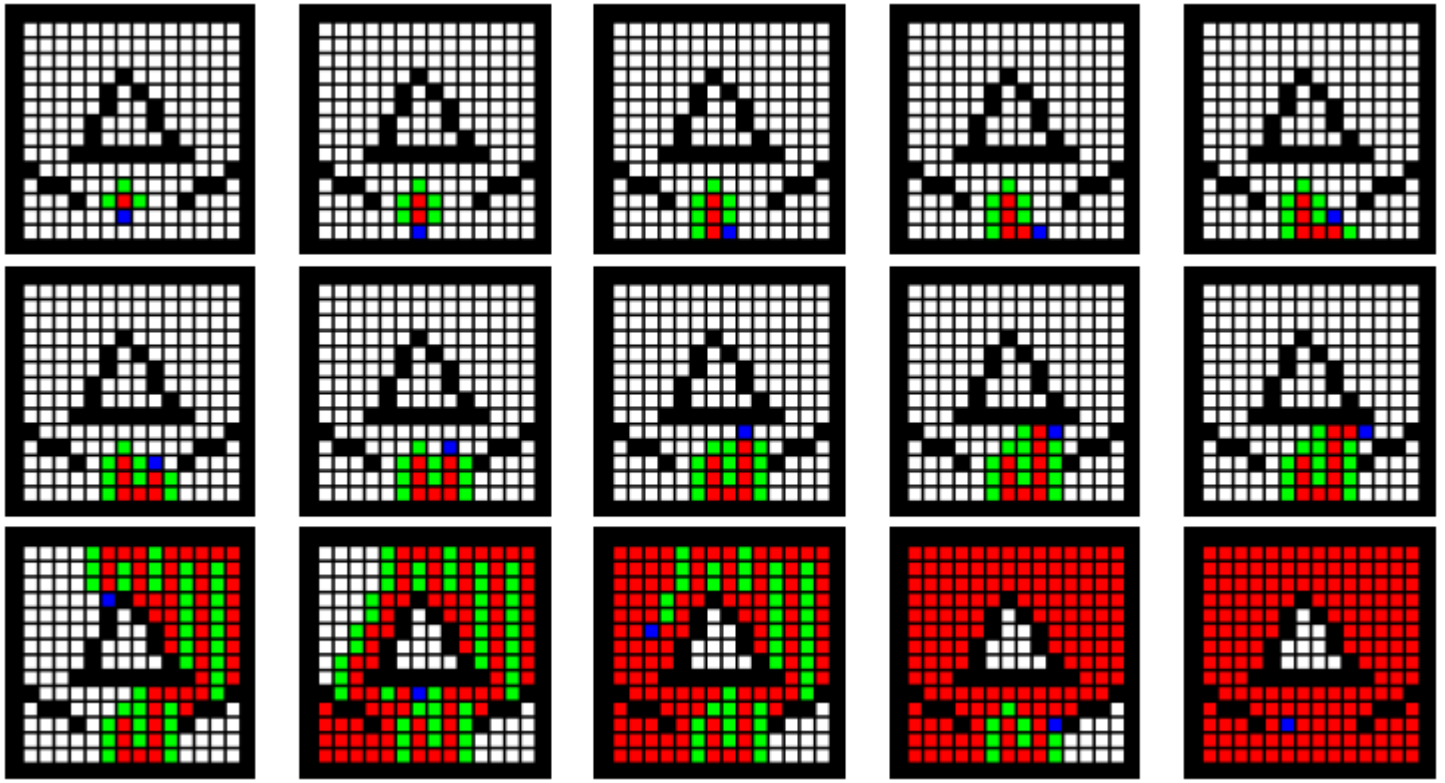
Area flood filling using a queue

Properties of abstract queues:

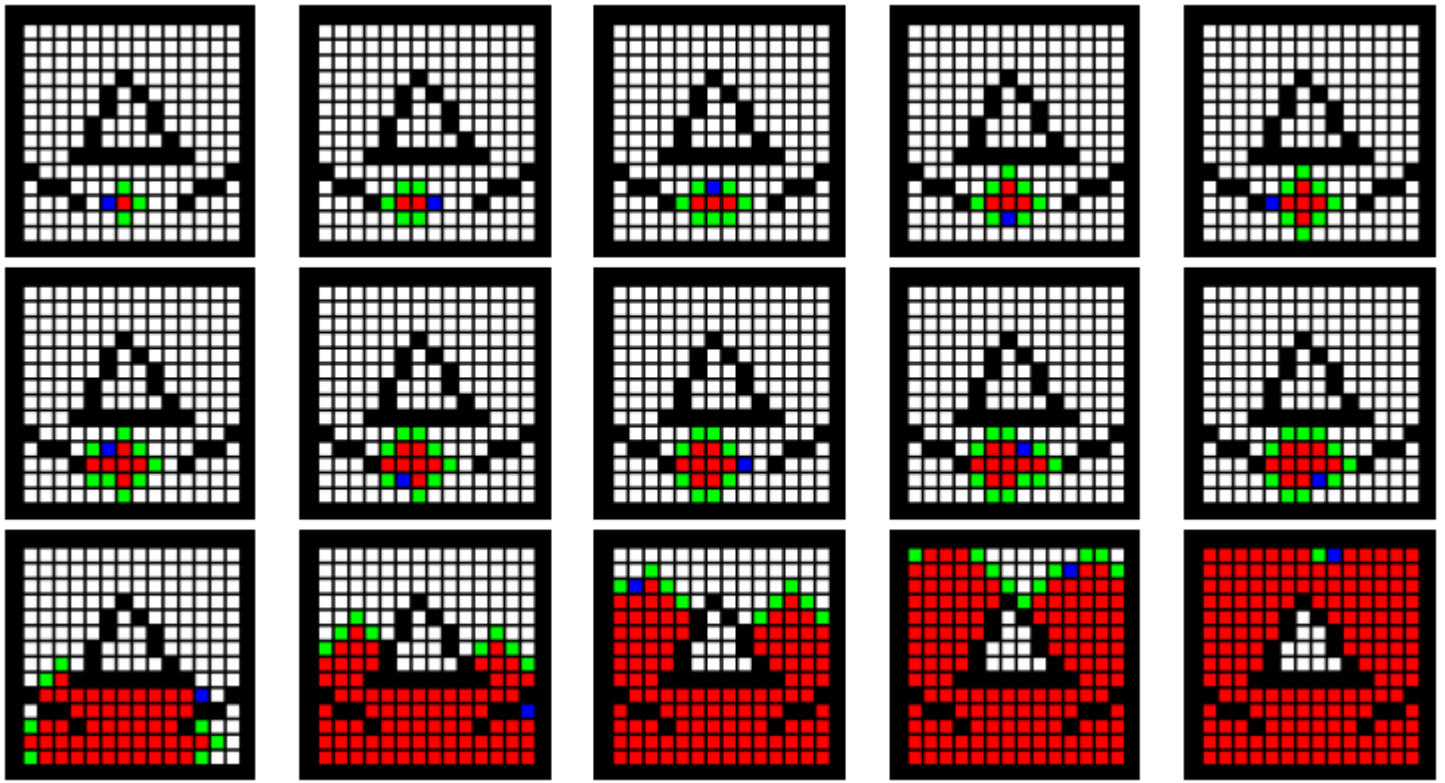
- $\text{new}() \rightarrow \emptyset$ returns an empty queue,
- $\text{head}(\text{enqueue}(\text{new}(), e)) \rightarrow e$,
- $\text{dequeue}(\text{enqueue}(\text{new}(), e)) \rightarrow \text{new}()$,
- $\text{head}(\text{enqueue}(\text{enqueue}(\mathsf{Q}, e), f)) \rightarrow \text{head}(\text{enqueue}(\mathsf{Q}, e))$ (FIFO definition),
- $\text{dequeue}(\text{enqueue}(\text{enqueue}(\mathsf{Q}, f), e)) \rightarrow \text{enqueue}(\text{dequeue}(\text{enqueue}(\mathsf{Q}, f)), e)$.

```
FLOODFILLQUEUE(I, p, C_B, C_F)
 1.  ◁ Create empty queue Q ▷
 2.  Q ← ∅
 3.  I[p] ← C_F
 4.  Q.enqueue(p)
 5.  while Q ≠ ∅
 6.      do p ← Q.head()
 7.         for q ∈ C_4(p)
 8.             do if I[q] = C_B
 9.                    then I[q] ← C_F
10.                         Q.enqueue(q)
11.         Q.dequeue()
```

Breadth first order

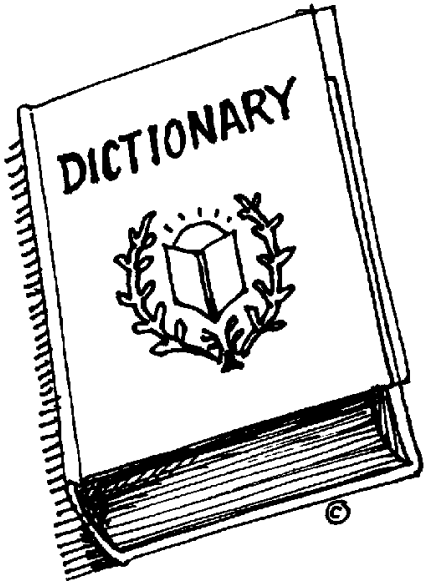


Stack
(depth search)



Queue
(breadth search)

3. Abstract dictionaries



Store items indexed by keys
Axiomatic semantic:

- $\text{new}() \rightarrow \emptyset$ returns an empty dictionary,
- $\text{find}(\text{insert}(\mathsf{D}, k, v), k) \rightarrow v$,
- $\text{find}(\text{insert}(\mathsf{D}, j, v), k) \rightarrow \text{find}(\mathsf{D}, k)$ if $k \neq j$,
- $\text{delete}(\text{new}(), k) \rightarrow \text{new}()$,
- $\text{delete}(\text{insert}(\mathsf{D}, k, v), k) \rightarrow \text{delete}(\mathsf{D}, k)$,
- $\text{delete}(\text{insert}(\mathsf{D}, j, v), k) \rightarrow \text{insert}(\text{delete}(\mathsf{D}, k), j, v)$ if $k \neq j$.

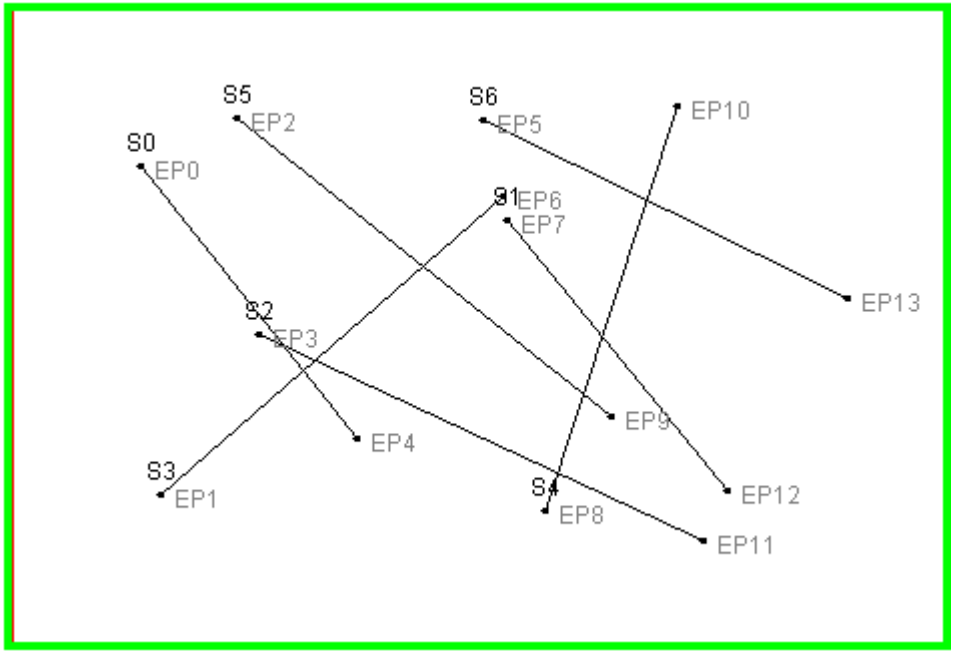
Unordered (collection) or *ordered* dictionaries
(→ augmented dictionaries)

IV. Dictionaries:
Detecting Any Segment Pair Intersection

Naive quadratic time algorithm:

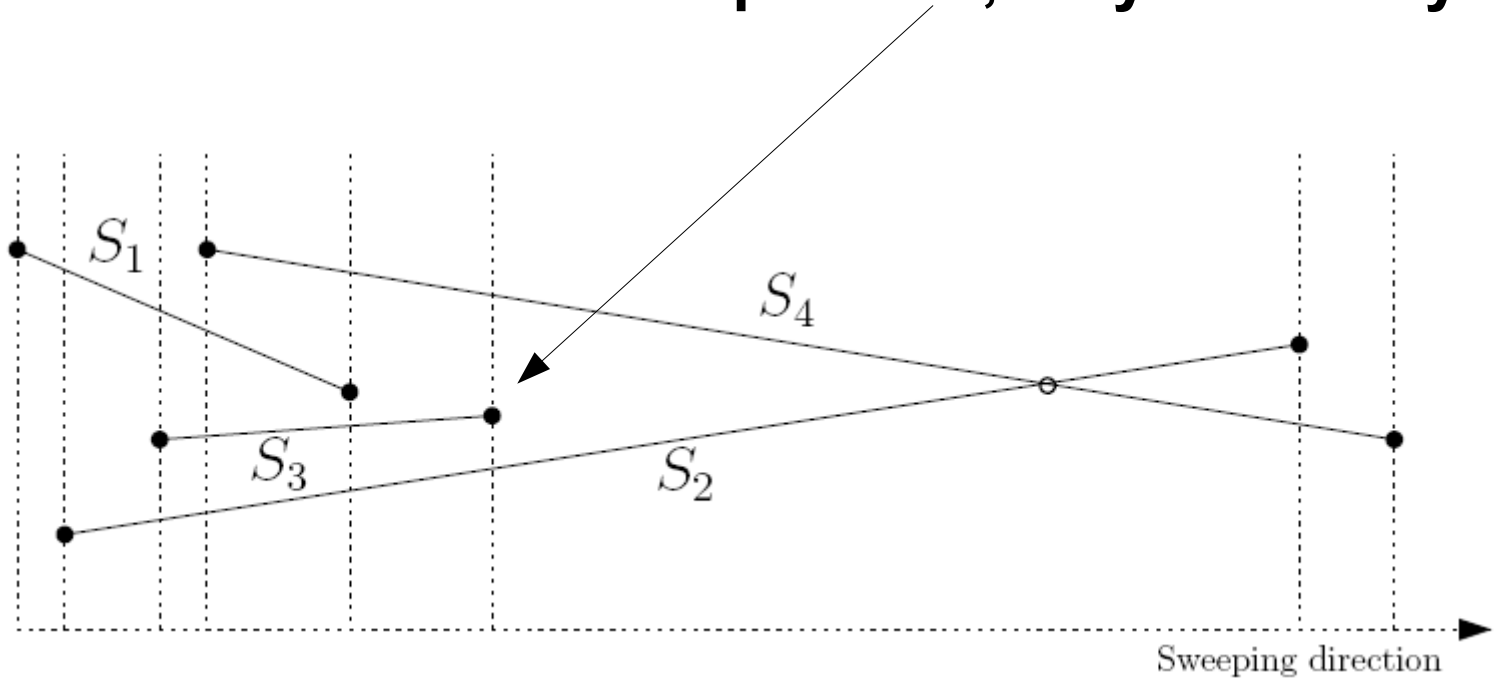
NAIVESEGMENTINTERSECTION($\mathcal{S} = \{S_1 = \{A_1, B_1\}, ..., S_n = \{A_n, B_n\}\}$)

1. ◁ Test pairwise segments for possible intersections ▷
2. $s \leftarrow 0$;
3. **for** $i \leftarrow 1$ **to** n
4. **do for** $j \leftarrow i+1$ **to** n
5. **do if** INTERSECTSEGMENT(S_i, S_j)
6. **then** $s = s+1$
7. REPORTINTERSECTIONSEGMENTPAIR(S_i, S_j)
8. **return** s



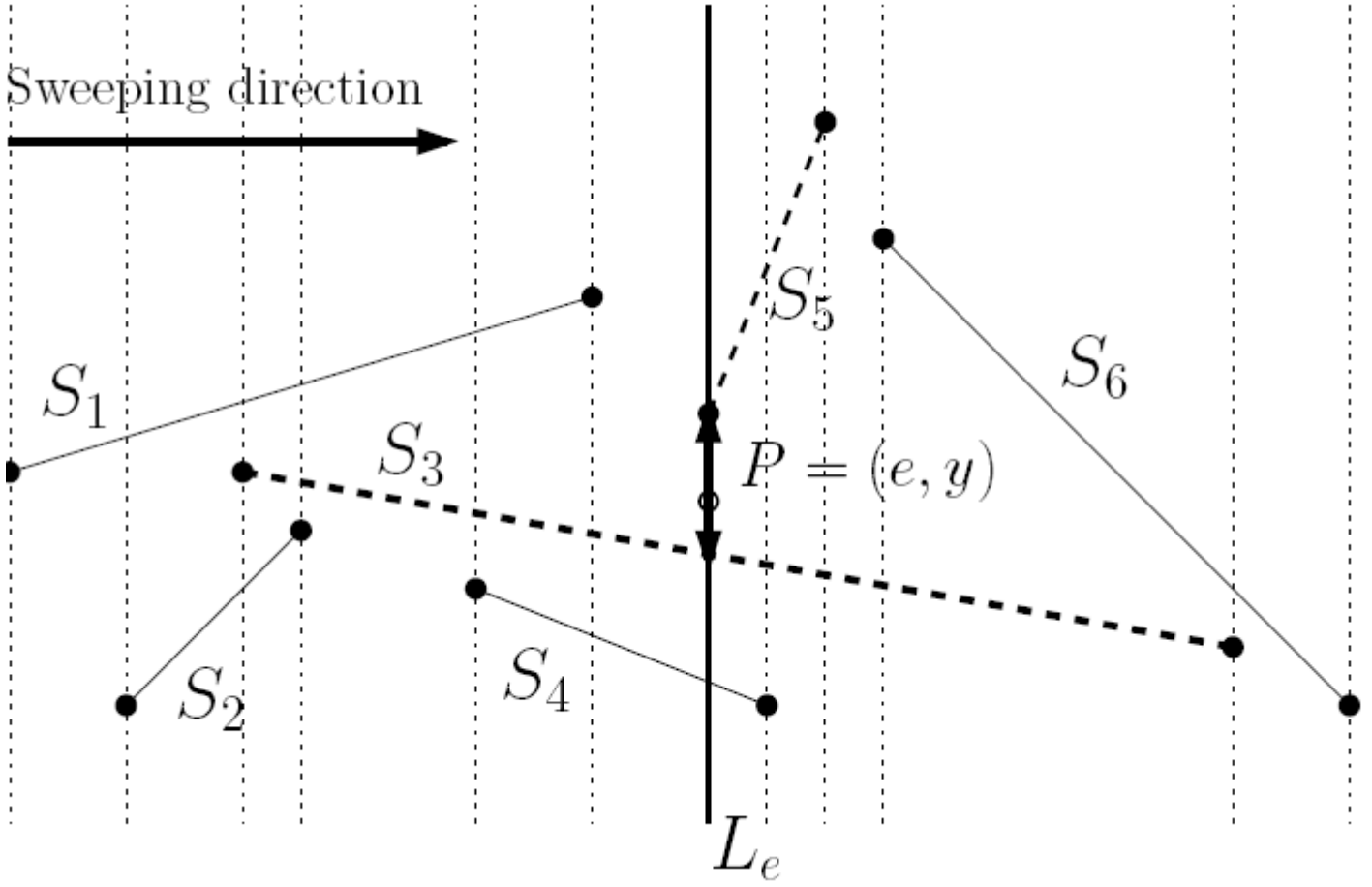
A structural property:
→ Sweeping combinatorial events

S2 intersects S4: At some position, they become y-adjacent



Sweeping line: combinatorial algorithm: 2n extremities only
→ 2n events so swap from left to right
(A preliminary computational geometric algorithm)

Ordered dictionary



For a given query (e,y) report the line segment immediately:

- above and
- below (e,y)

Dynamic insertion/deletion of line segment extremities

Implementation using binary trees (red-black, AVL, etc.)

Shamos-Hoey intersection detection algorithm

Does a pair of segment intersect? Yes or No?

```
DetectSegmentIntersection(S = {S_1 = {A_1, B_1}, ..., S_n = {A_n, B_n}})
 1.  ◁ Test whether there exists an intersection between any pair of segments of S ▷
 2.  E ← {A_i.x, B_i.x | i ∈ [[1, n]]}
 3.  T ← SortIncreasingOrder(E)
 4.  ◁ Initialize an empty dictionary D ▷
 5.  D ← ∅
 6.  for i ← 1 to 2n
 7.      do
 8.          ◁ Let S be the segment supporting endpoint x-coordinate T[i] ▷
 9.          switch
10.              case LeftEndpoint(T[i]) :
11.                   D.insert(S)
12.                   if Intersect(D.above(S), S) OR Intersect(D.below(S), S)
13.                      then return true
14.              case RightEndpoint(T[i]) :
15.                   D.delete(S)
16.                   if Intersect(D.above(S), D.below(S))
17.                      then return true
18.  return false
```

Priority queues:

- $\text{new}() \rightarrow \emptyset$ returns an empty priority queue Q,
- $\text{head}(\text{insert}(\text{new}(), v)) = v$,
- $\text{deleteHead}(\text{insert}(\text{new}(), v)) = \text{new}()$,
- $\text{head}(\text{insert}(\text{insert}(\mathsf{Q}, w), v)) =$

$$\begin{cases} v & \text{if priority}(v) < \text{priority}(\text{head}(\text{insert}(\mathsf{Q}, w))), \\ \text{head}(\text{insert}(\mathsf{Q}, w)) & \text{otherwise.} \end{cases},$$

- $\text{deleteHead}(\text{insert}(\text{insert}(\mathsf{Q}, w), v)) =$

$$\begin{cases} \text{insert}(\mathsf{Q}, w) & \text{if priority}(v) < \text{priority}(\text{head}(\text{insert}(\mathsf{Q}, w))), \\ \text{insert}(v, \text{deleteHead}(\text{insert}(\mathsf{Q}, w))) & \text{otherwise.} \end{cases}$$

(INF311: Implementation using implicit heaps stored in an array)

For the line sweep: 3 kinds of events:

- type 1: Begin extremity
- type 2: End extremity
- type 3: Intersection point

Bentley-Ottman algorithm (extend Shamos-Hoey)

Complexity: O((n+I) log n)

Initialization:

FINDSEGMENTINTERSECTIONS($\mathcal{S} = \{\{A_1, B_1\}, ..., \{A_n, B_n\}\}$)

1. ◁ Bentley-Ottman's algorithm for reporting all intersection points ▷
2. ◁ Q is a priority queue ▷
3. $k \leftarrow 0$
4. $\mathsf{Q} \leftarrow \emptyset$
5. ◁ Insert the $2n$ line segment endpoints ▷
6. ◁ sorted according to their x-abscissae ▷
7. **for** $i \leftarrow 1$ **to** n
8. **do**
9. Q.insert(A_i, i)
10. Q.insert(B_i, i)
11. ◁ While there are remaining events: ▷
12. ◁ Handle events and update the priority queue accordingly ▷
13. **while** $\mathsf{Q} \neq \emptyset$
14. **do**
15. $P \leftarrow$ Q.head()
16. ◁ HandleEventPoint is described thereafter (see next page). ▷
17. HANDLEEVENTPOINT(P)

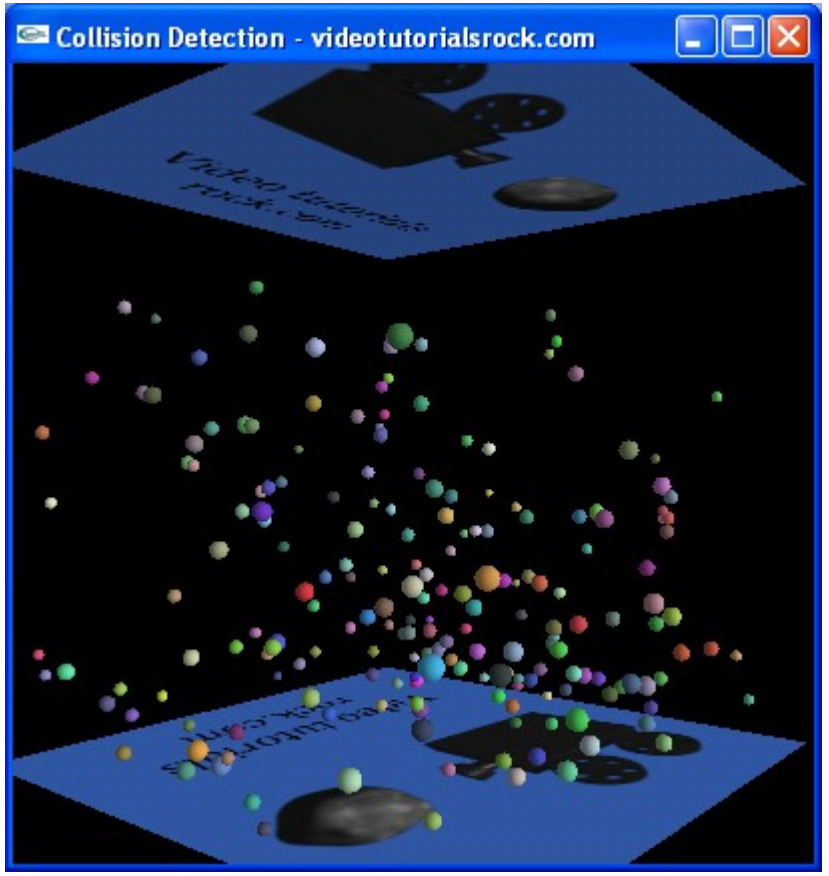
Bentley-Ottman algorithm: Handling events

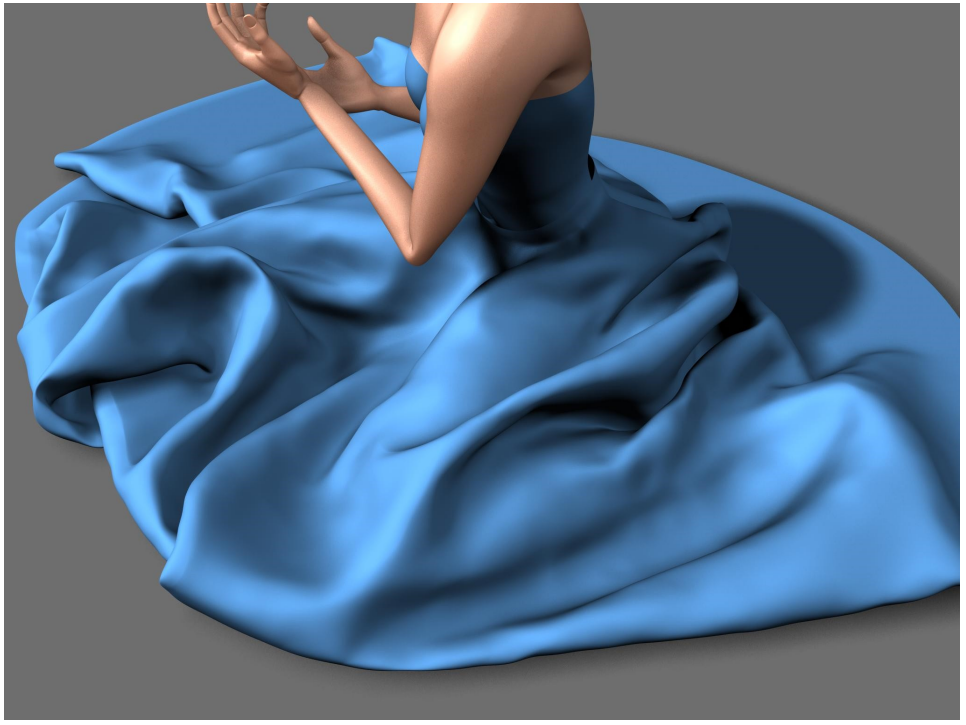
```
HandleEventPoint(P)
 1.  ◁ D is a dictionary data structure ▷
 2.  ◁ Q is a priority queue not storing duplicates (general position assumpti
 3.  ◁ x_{S1S2} represents the x-coordinate of intersection point S1 ∩ S2 ▷
 4.  ◁ Let S denote the segment which contains endpoint P ▷
 5.  switch
 6.    case LeftEndPoint(P) :
 7.          D.insert(S)
 8.          S1 ← D.above(S)
 9.          S2 ← D.below(S)
10.          if S1 ∩ S ≠ ∅
11.            then k ← k + 1
12.                 Q.insert(x_{S1S})
13.          if S2 ∩ S ≠ ∅
14.            then k ← k + 1
15.                 Q.insert(x_{S2S})
16.    case RightEndPoint(P) :
17.          S1 ← D.above(S)
18.          S2 ← D.below(S)
19.          ◁ Avoid counting an already detected intersection point ▷
20.          if S1 ∩ S2 ≠ ∅ and x_{S1S2} ≥ x_P
21.            then k ← k + 1
22.                 Q.insert(x_{S1S2})
23.                 D.delete(S)
24.    case IntersectionPoint(P) :
25.          ◁ S1 and S2 are the segments such that P = S1 ∩ S2 ▷
26.          S3 ← D.above(S1)
27.          S4 ← D.below(S2)
28.          if S3 ∩ S2 ≠ ∅
29.            then k ← k + 1
30.                 Q.insert(x_{S3S2})
31.          if S1 ∩ S4 ≠ ∅
32.            then k ← k + 1
33.                 Q.insert(x_{S1S4})
34.          SwapDictionaryItems(S1, S2, D)
```

Interesting case:
Swap manually order
at intersection points

Collision detection for 3D games

http://en.wikipedia.org/wiki/Bentley%E2%80%93Ottmann_algorithm





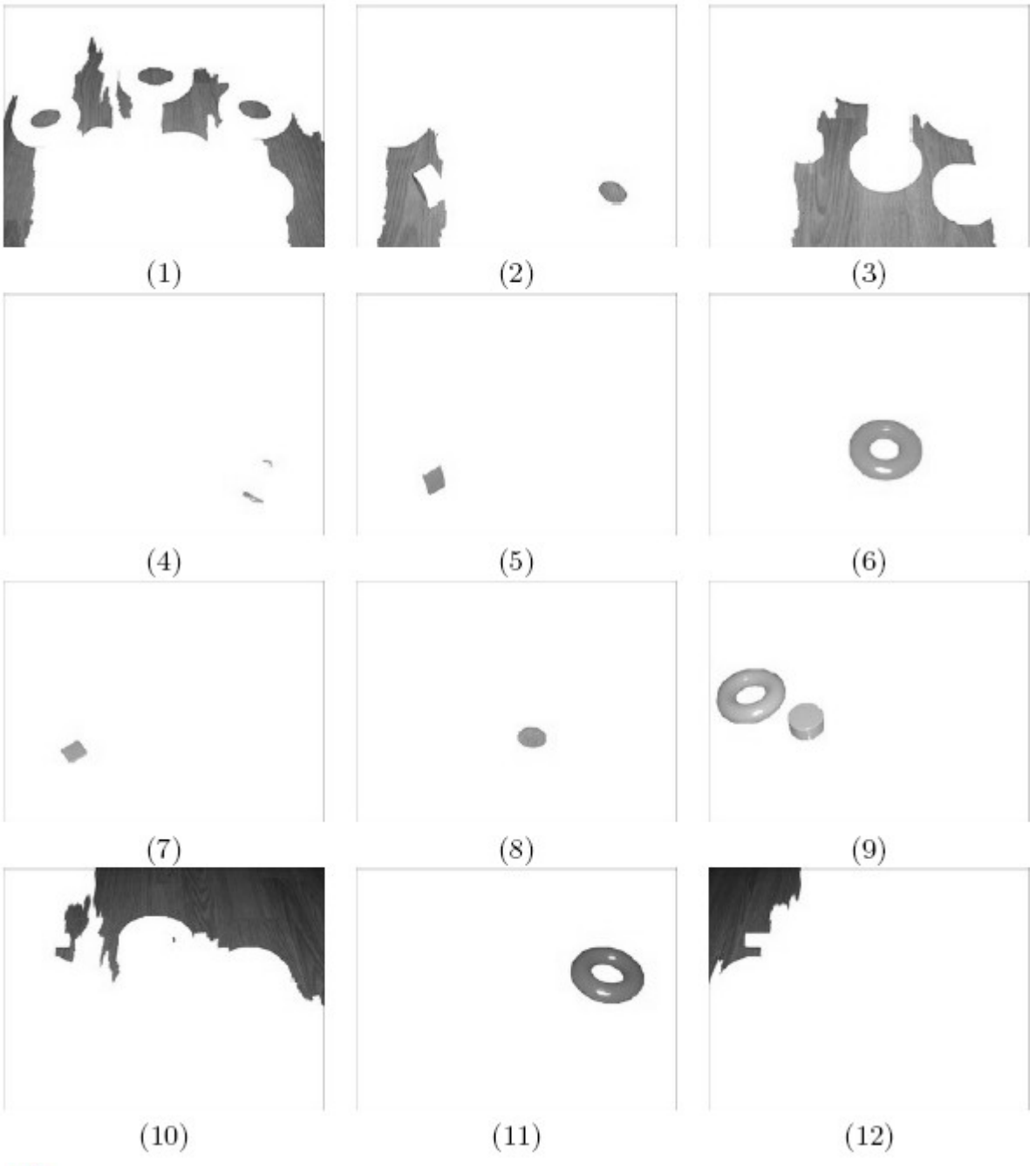
IV Union-Find abstract data-structures

Manage disjoint sets very efficiently under union

Example: Image segmentation (let a robot see!)

```
RegionMerging(I)
 1.  ◁ Test merging predicates on sorted C4 adjacent pixel pairs ▷
 2.  P ← {Pl = (pl, ql, Cl = |I(ql) − I(pl)|) | such that ql ∈ C4(pl)}
 3.  ◁ Sort in increasing order according to Cl ▷
 4.  Sort(P)
 5.  for i ← 1 to |P|
 6.      do
 7.          if region(pl) ≠ region(ql)
 8.             then ◁ MergePredicate(r1, r2)=PerceptualUnit(r1 ∪ r2) ▷
 9.                  if MergePredicate(region(pl), region(ql))
10.                     then Merge(region(pl), region(ql))
```

Union-Find abstract data-structures



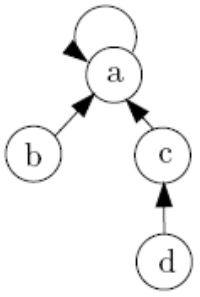
Each region is a set
Initially, each pixel defines a region

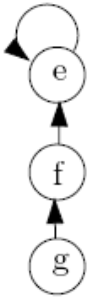
Merging regions=merging disjoint sets

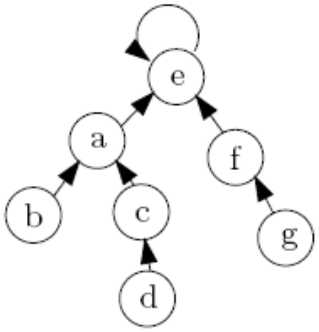
For a given element,
find the set that contains it

Union-Find abstract data-structures

MAKESET(x)
1. parent(x) $\leftarrow x$
2. rank(x) $\leftarrow 0$





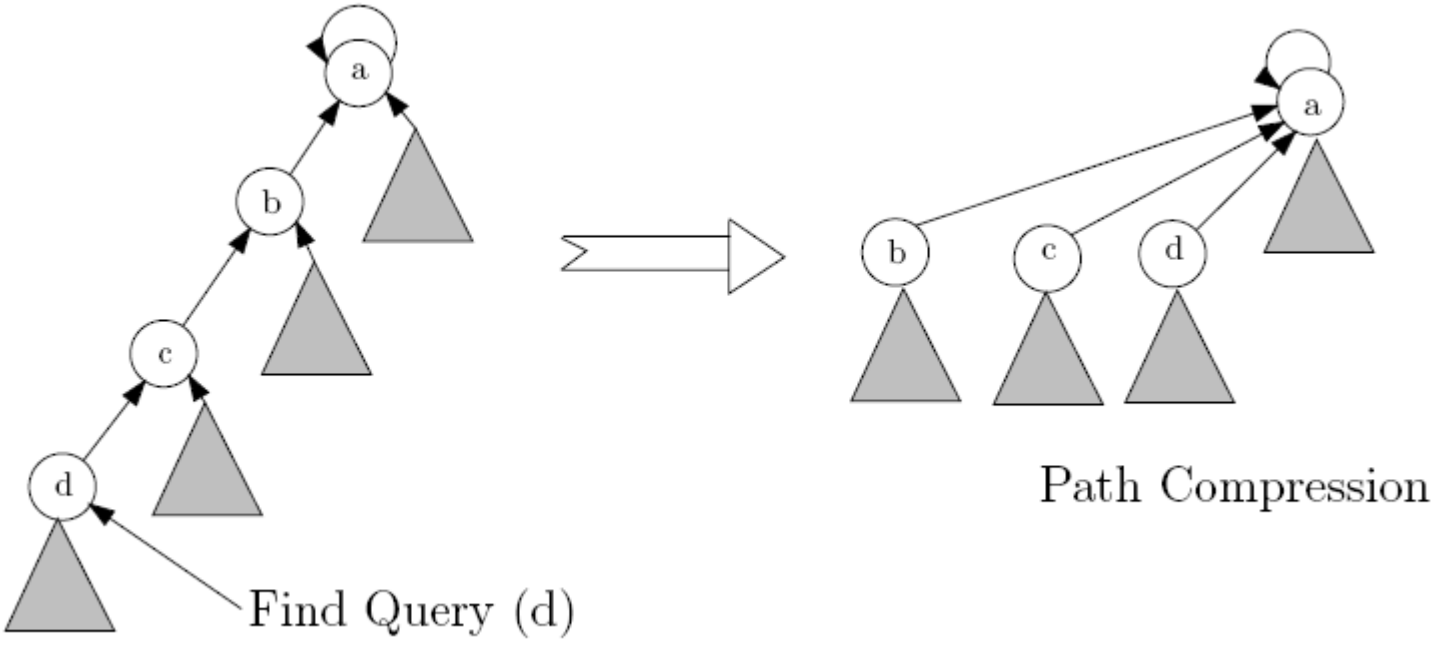


$\mathcal{S}_1 = \{a, b, c, d\}$ $\mathcal{S}_2 = \{e, f, g\}$ $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$

FIND(x)
1. ◁ Walk to the leader element ▷
2. **while** $x \neq$ parent(x)
3. **do** $x \leftarrow$ parent(x)
4. **return** x

UNION(x, y)
1. ◁ Union by rank and path compression ▷
2. $x_r \leftarrow$ Find(x)
3. $y_r \leftarrow$ Find(y)
4. **if** rank(x_r) > rank(y_r)
5. **then** parent(x_r) $\leftarrow y_r$
6. **else** parent(y_r) $\leftarrow x_r$
7. **if** rank(x_r) = rank(y_r)
8. **then** rank(x_r) $\leftarrow$ rank(x_r) + 1

Path compression: Almost linear-size data-structure (inverse of Ackermann function)

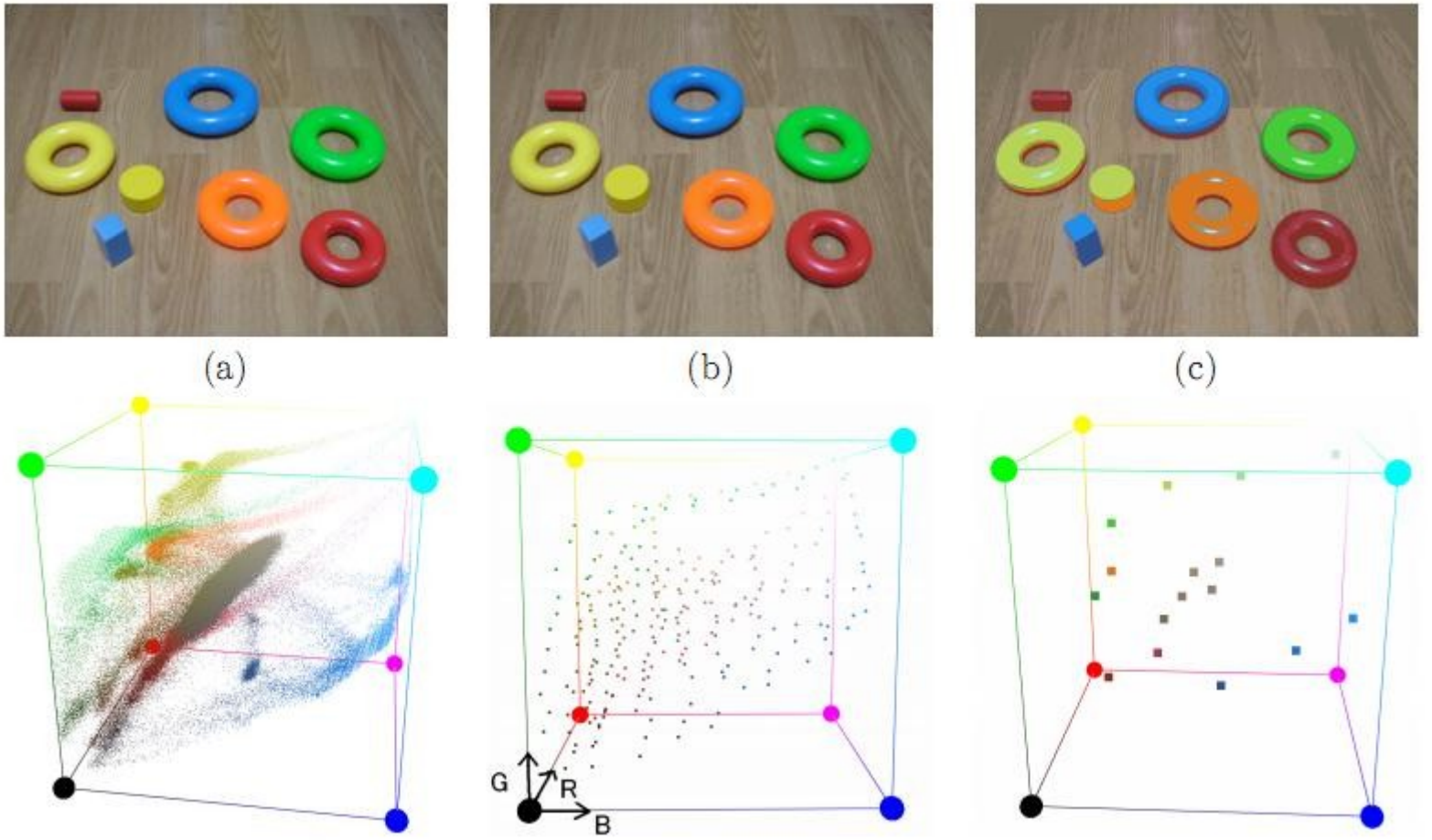


Union find DS for labelling connected components

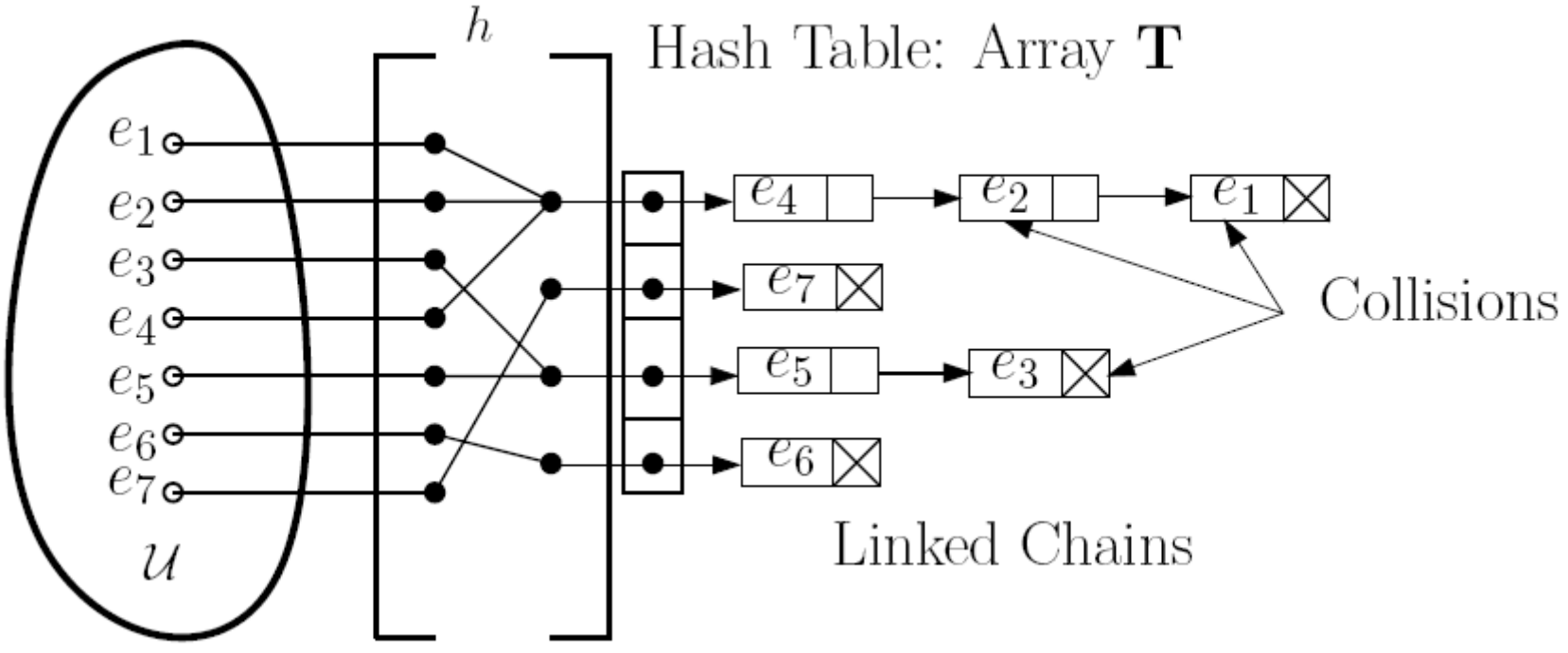
Much easier problem than segmentation

Disjoint set data-structure

Segmentation, clustering and tone mapping



V Hashing



Hashing and collision (INF3x1 du Tronc Commun):

- Open address
- Linked chains

V. Geometric hashing: Preprocessing (for a point set)

For each feature pair of points:

- Define a local coordinate basis on this pair
- Compute and quantize all other feature points in this coordinate basis
- Record (model, basis) in a hash table

← Quadratic time preprocessing

http://en.wikipedia.org/wiki/Geometric_hashing

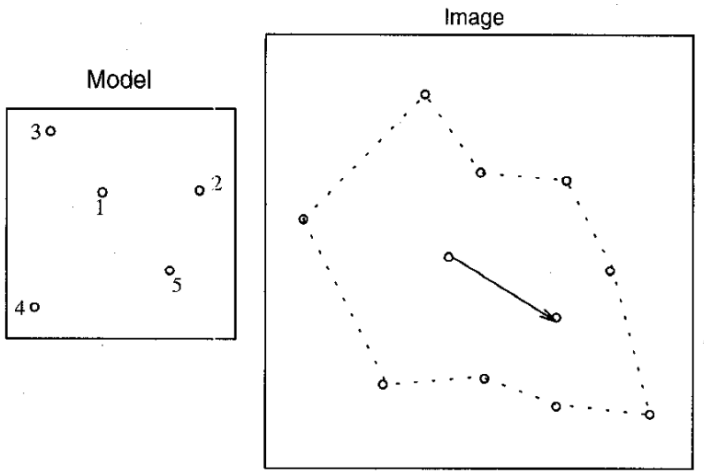
http://www.seas.gwu.edu/~simhaweb/cs151/lectures/module4/module4.html

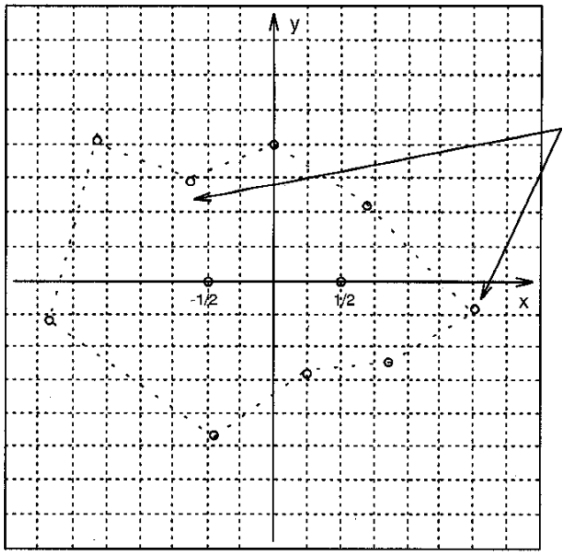
Geometric hashing: Recognition

1. Pick arbitrary ordered pair of scene:
Compute the other points using this pair as a frame basis

2. For all the transformed points, vote all records (model, basis) appear in the corresponding entry in the hash table, and histogram them
(= frequency histogram)
3. Matching candidates: (model, basis) pairs with large number of votes.

4. Recover the transformation that results in the best least-squares match between all corresponding feature points
Transform the features, and verify against the input image features
(if fails, repeat this procedure with another pair...)





Geometric hashing (2D rigid motion)

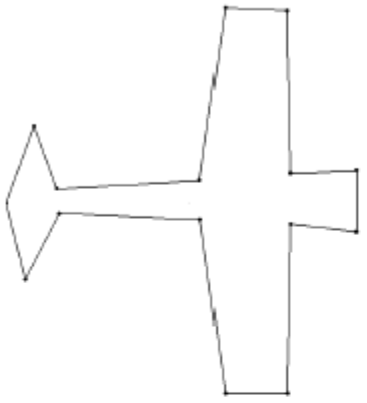
AddToHashTable($\mathbf{H}, M_i, C_i$)

1. ◁ Hash table ▷
2. ◁ C_i denotes the reference frame induced by $C_i = \{\mathbf{p}_{i,1}, \mathbf{p}_{i,2}\}$ ▷
3. **for** $j \leftarrow 1$ **to** m_i
4. **do**
5. ◁ Decompose $P_{i,j}$ onto the local frame ▷
6. ◁ Solve for λ_1 and λ_2: $(\lambda_1, \lambda_2) = C_{C_i}(P_{i,j})$ ▷
7. $\mathbf{p_{i,j}} = \frac{\mathbf{p}_{i,1}\mathbf{p}_{i,2}}{2} + \lambda_1 \mathbf{p}_{i,1} + \lambda_2 \mathbf{p}_{i,2}$
8. ◁ Add model i is the hash bin of $P_{i,j}$ ▷
9. $\mathbf{H}[\lambda_1][\lambda_2] = i;$

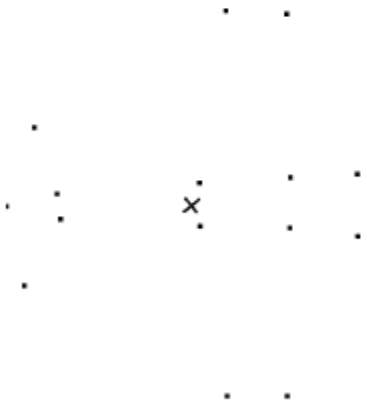
$$\begin{bmatrix} x_{i,j} \\ y_{i,j} \end{bmatrix} - \begin{bmatrix} \frac{x_{i,1}+x_{i,2}}{2} \\ \frac{y_{i,1}+y_{i,2}}{2} \end{bmatrix} = \underbrace{\begin{bmatrix} x_{i,1} & x_{i,2} \\ y_{i,1} & y_{i,2} \end{bmatrix}}_{\text{Reference frame}} \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix},$$

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} x_{i,1} & x_{i,2} \\ y_{i,1} & y_{i,2} \end{bmatrix}^{-1} \begin{bmatrix} x_{i,j} - \frac{x_{i,1}+x_{i,2}}{2} \\ y_{i,j} - \frac{y_{i,1}+y_{i,2}}{2} \end{bmatrix}.$$

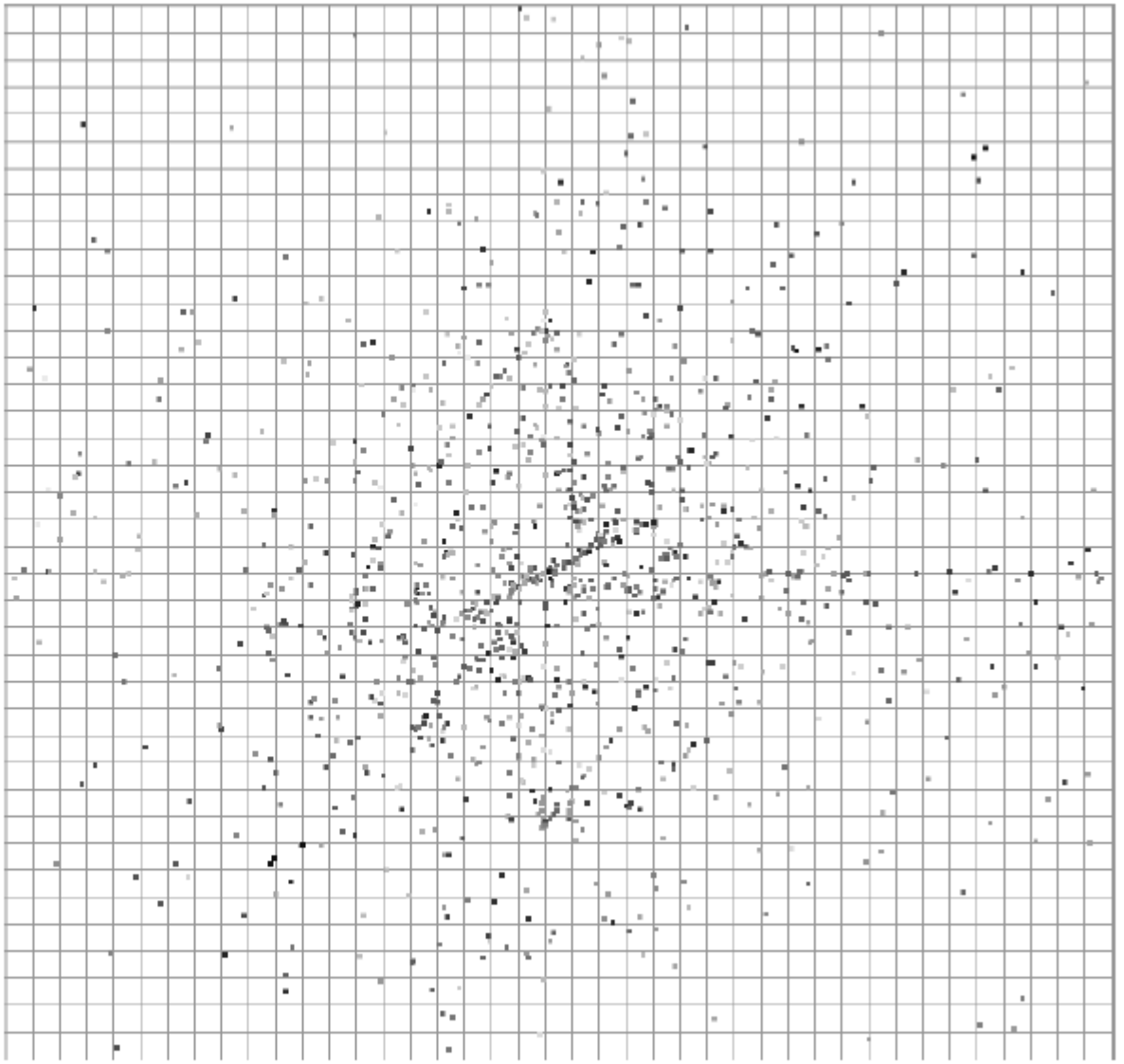
Geometric hashing



(a)

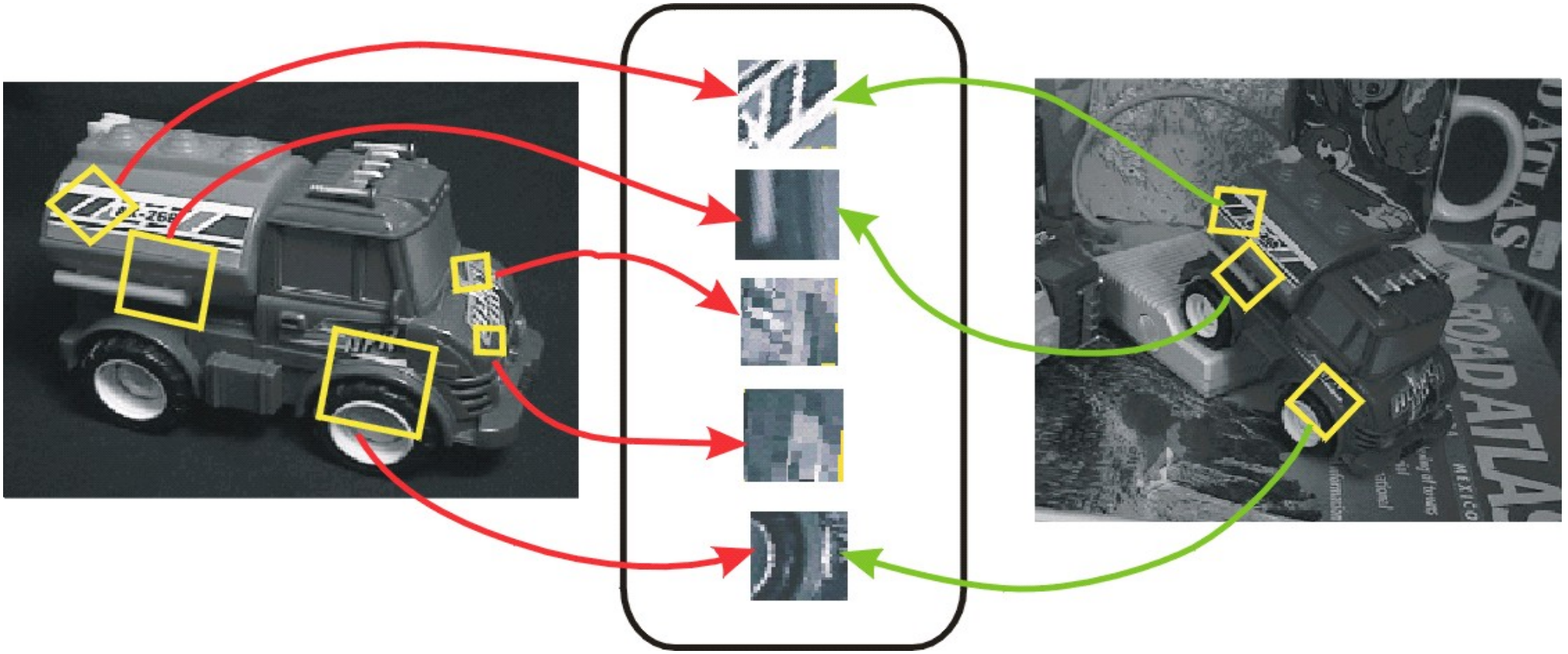


(b)



(c)

Geometric hashing: Object recognition



Object Recognition from Local Scale-Invariant Features (SIFT)

Scalable recognition

Further readings

http://www.lix.polytechnique.fr/~nielsen/INF555/index.html

- Wolfson, H.J. & Rigoutsos, I (1997).
Geometric Hashing: An Overview.
IEEE Computational Science and Engineering, 4(4), 10-21.

- Andrew S. Glassner: Fill 'Er Up!
IEEE Computer Graphics and Applications 21(1): 78-85 (2001)

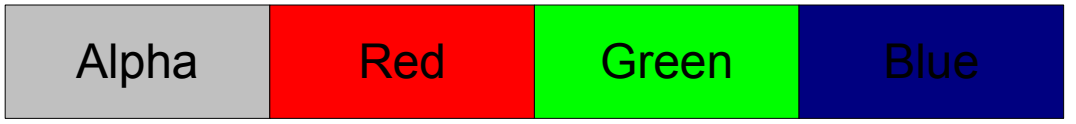
Java I/O Image

Several APIs

Can use also processing.org

```
public Image(String filename) {
     java.awt.Image img = Toolkit.getDefaultToolkit().getImage(filename);

     if (img != null) {
         PixelGrabber pg = new PixelGrabber(img, 0, 0, -1, -1, true);
         try {
             pg.grabPixels();
         } catch (InterruptedException e) {
         }
         raster = (int[]) pg.getPixels();
         height = pg.getHeight();
         width = pg.getWidth();

     } else {
         throw new IllegalArgumentException("Error opening file "
          + filename);
     }
 }
```

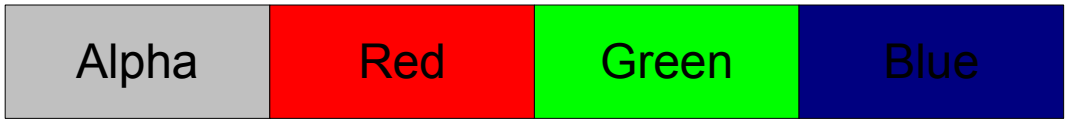
Java I/O Image




```
public int getBlue(int x, int y) {
      if (x < 0 || x >= width || y < 0 || y >= height) {
          throw new IllegalArgumentException();
      }
      return (raster[y * width + x] & 0xFF);
  }

  public int getGreen(int x, int y) {
      if (x < 0 || x >= width || y < 0 || y >= height) {
          throw new IllegalArgumentException();
      }
      return ((raster[y * width + x] & 0xFF00) >> 8);
  }

  public int getRed(int x, int y) {
      if (x < 0 || x >= width || y < 0 || y >= height) {
          throw new IllegalArgumentException();
      }
      return ((raster[y * width + x] & 0xFF0000) >> 16);

}
```

Opacity is 255 (0xFF)

Java I/O Image




```
public void setGreen(int x, int y,  int value) {
        if (x < 0 || x >= width || y < 0 || y >= height) {
            throw new IllegalArgumentException();
        }
       raster[y * width + x]   = raster[y * width + x] | ((value&0xff)<<8);
    }


    public void setRed(int x, int y,  int value) {
        if (x < 0 || x >= width || y < 0 || y >= height) {
            throw new IllegalArgumentException();
        }
        raster[y * width + x]   = raster[y * width + x] | ((value&0xff)<<16);
    }

    public void setBlue(int x, int y,  int value ) {
        if (x < 0 || x >= width || y < 0 || y >= height) {
            throw new IllegalArgumentException();
        }
        raster[y * width + x]   = raster[y * width + x] | (value&0xff);
    }
```

PPM Image

```
public void writeBinaryPPM(String filename)
    {
        String line;StringTokenizer st;
        int i;
        try {
            DataOutputStream out =new DataOutputStream(
                    new BufferedOutputStream(new
FileOutputStream(filename)));
            out.writeBytes("P6\n");
            out.writeBytes("# INF555 Ecole Polytechnique\n");
            out.writeBytes(width+" "+height+"\n255\n");
                    for (int y = 0; y < height; y++) {
                        for (int x = 0; x < width; x++) {
                            int r,g,b;
                            r=getRed(x,y);
                            g=getGreen(x,y);
                            b=getBlue(x,y);

                            out.writeByte((byte)r);
                            out.writeByte((byte)g);
                            out.writeByte((byte)b);
                        }
                    }
            out.close();
        } catch(IOException e) {}
```