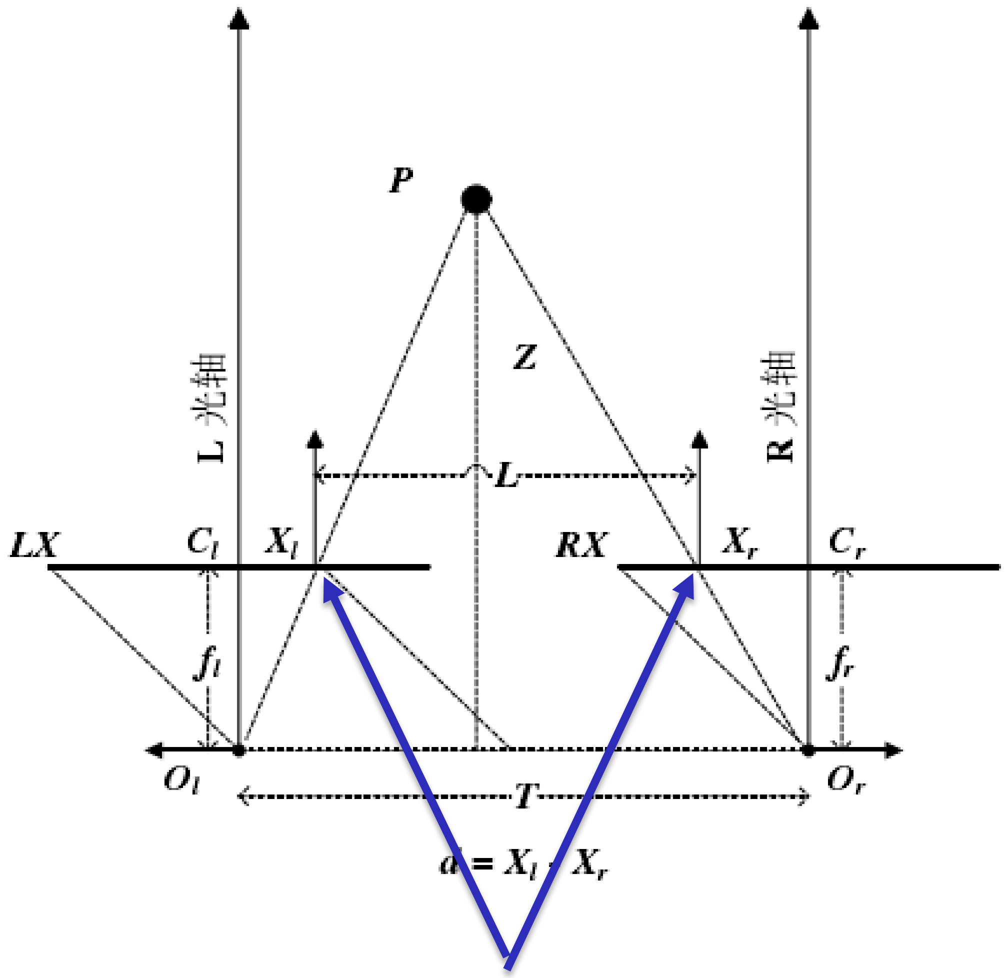
基于视觉及人工智能的三维重建

计算机图形学——原理、方法及应用（第4版），潘云鹤、童若锋、耿卫东、唐敏、童欣，高等教育出版社，2022。

内　容

- 双目重建的原理



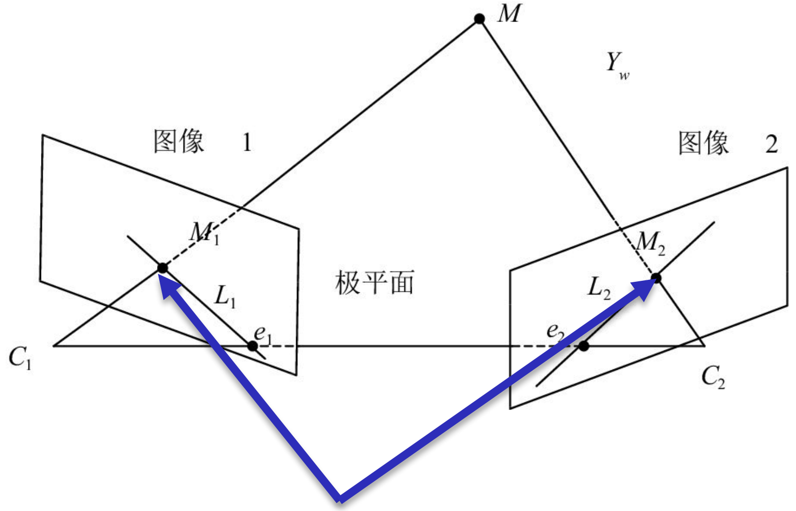
$$\frac{T-(X_l-X_l)}{Z-f}=\frac{T}{Z}$$

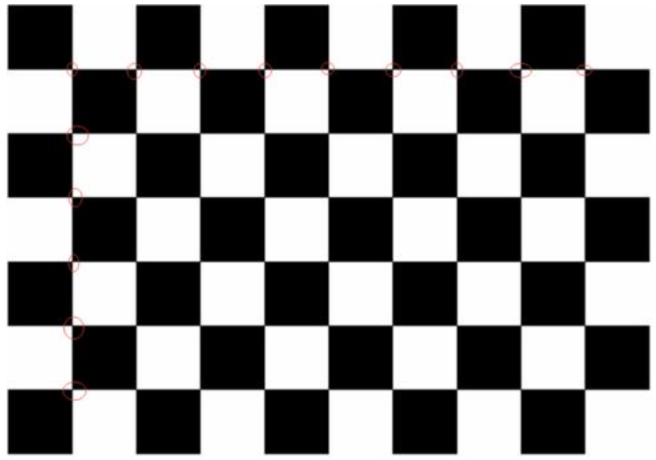
$$Z=\frac{Tf}{(X_l-X_l)}$$

基于双目视觉的三维重建基本流程:

相机标定→ 图像校正→ 立体匹配→深度计算→ 三维重建

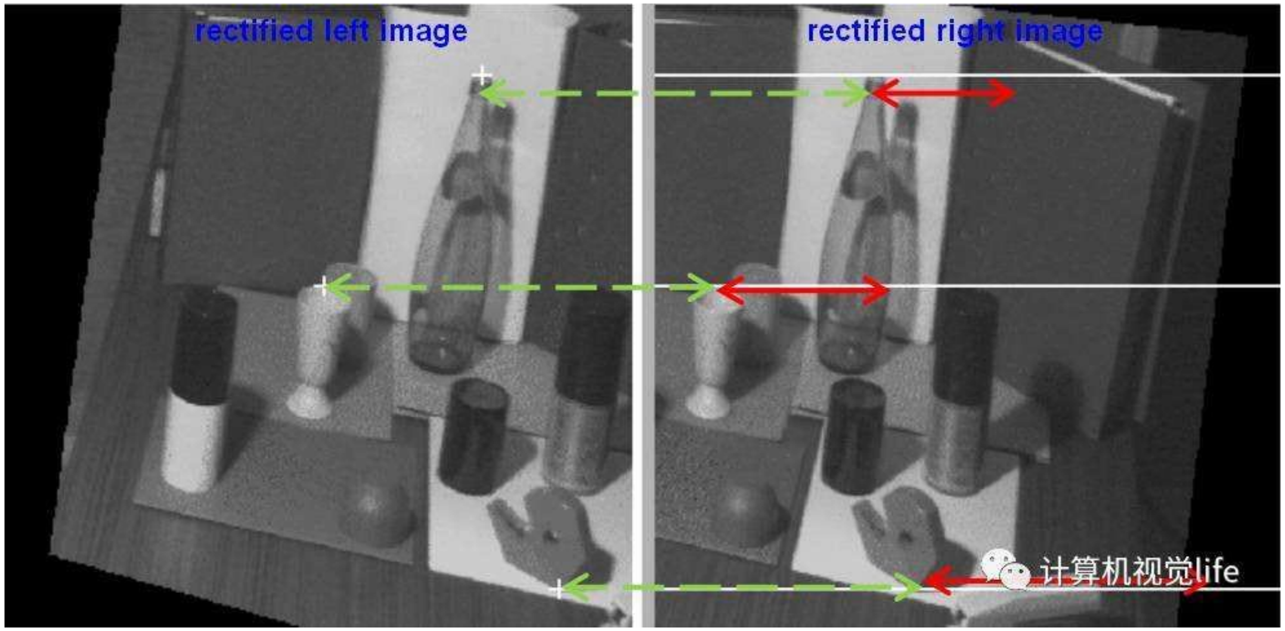
关键步骤是立体匹配





问题：如何寻找双目图像中的像素匹配?

立体匹配



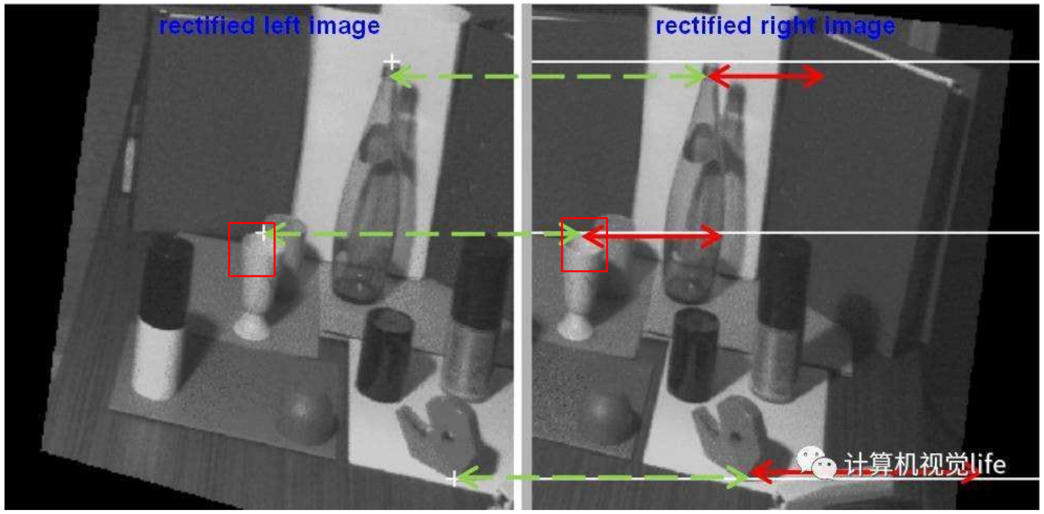
(1) 局部方法 (2) 整体方法 (3) 半整体方法

基本思路：以给定大小的图像块滑动窗口寻找匹配

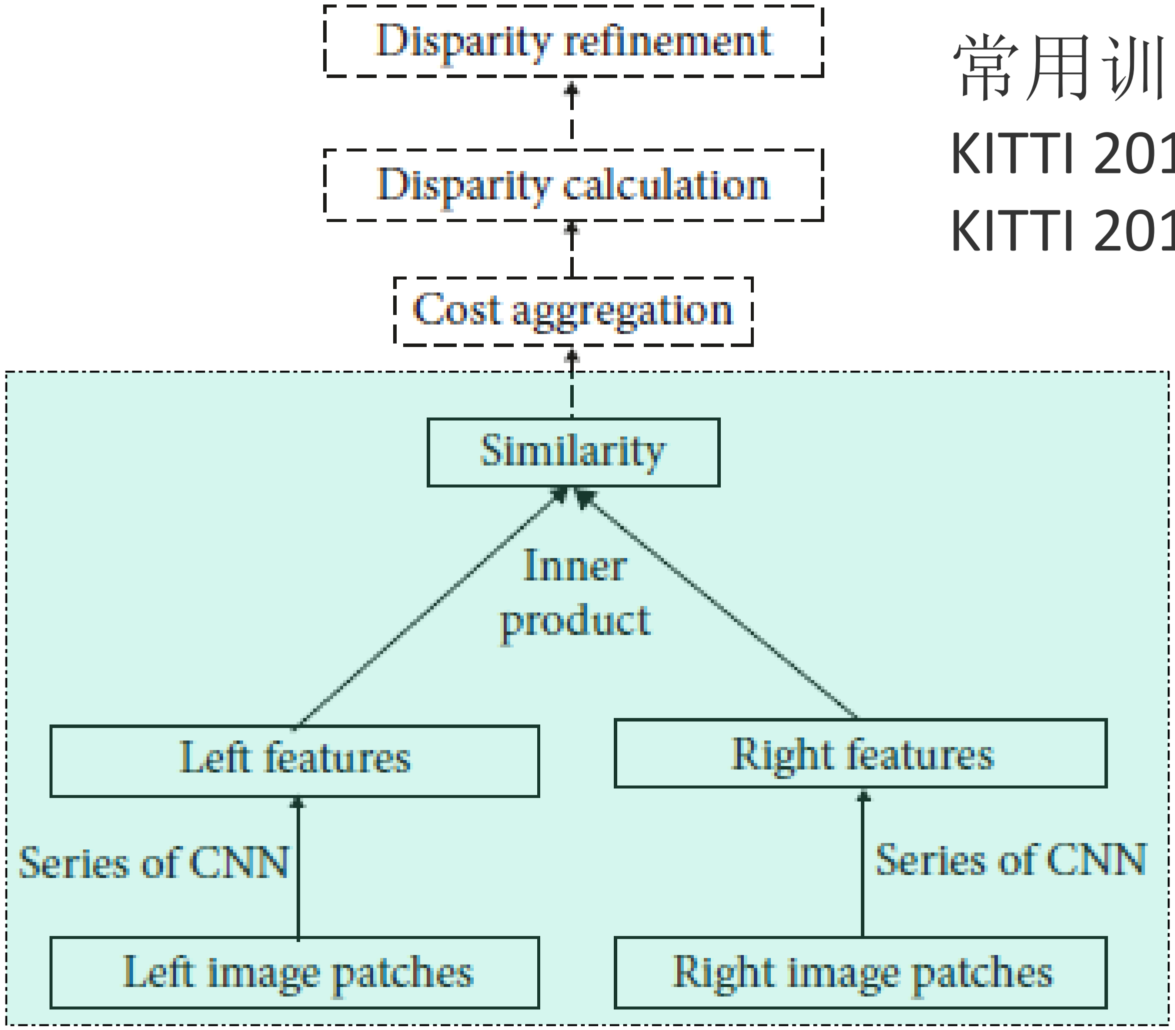
- 匹配代价计算（即两个窗口的相似度计算）
- 代价聚合
- 视差计算和修正

匹配代价计算

是一个特征表达及特征相似度度量问题，SAD（ Sum of Absolution Difference ）窗口中对应像素差绝对值的和梯度特征、纹理特征及其Census 变换都可尝试用于匹配代价计算。



基于孪生神经网络的代价计算



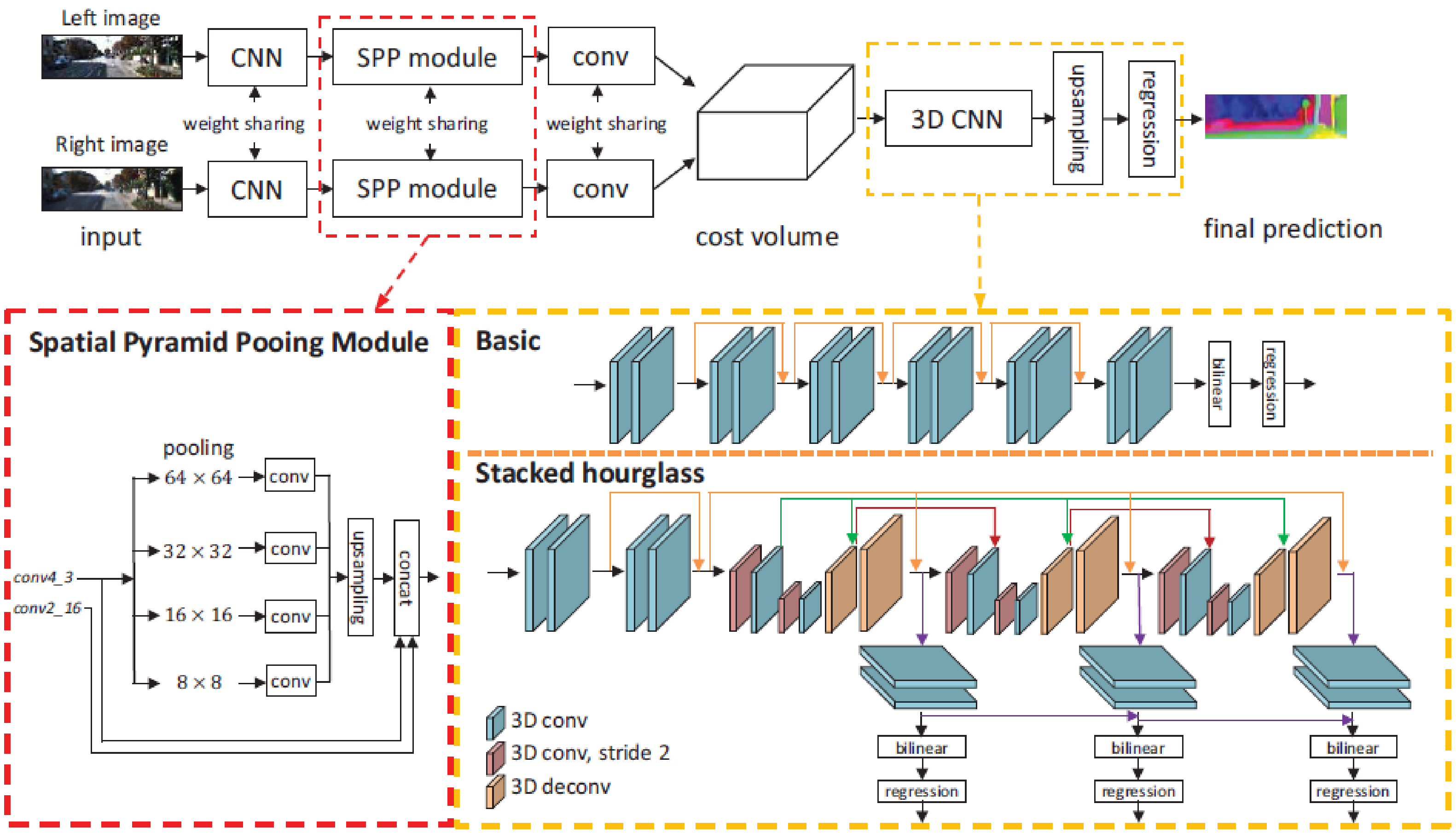
Siamese network

常用训练集

KITTI 2012 194图像对

KITTI 2015 200图像对

基于端到端神经网络的深度计算
--直接输出深度估计

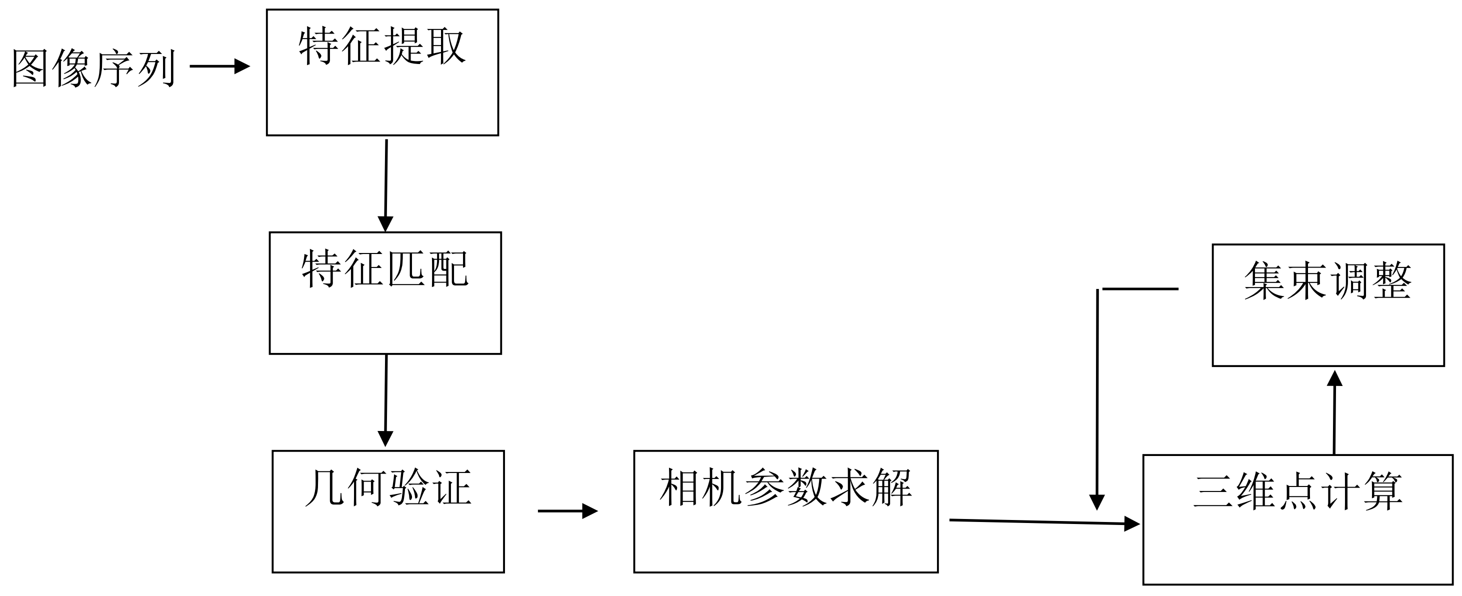


7 厘米

基于运动的三维形状重建
（structure from motion）

从运动恢复结构的基本概念及流程

按双目视觉的思路，但不借助棋盘格而是直接利用拍摄的视频对相机进行标定（称为相机的自标定）。相机自标定在准确性、鲁棒性等方面的欠缺使得需要在后续增加额外的处理。



增量式SFM基本流程

从运动恢复结构的具体步骤

- 特征提取：

包括特征点的检测及描述，目的是检测出角点等具有结构信息的点并给出能够尽可能表达出其特点的特征描述，用于与其它图像中特征点的匹配。Harris 角点检测、高斯差分检测（Difference of gaussian，DOG）等都可以用来进行特征点检测，SIFT特征由于具有尺度和旋转不变性及对光照不敏感的优点，经常用来作为特征描述。

- 特征匹配：

将具有较多重叠区域的两幅图像中提取出的特征点根据其特征描述子进行配对，特征描述子间的距离 、归一化互相关性（Normalized cross correlation，NCC）等都可以用来判断特征点的相似性。当场景中具有重复结构或对称形状时，会使得某些特征点具有非常相似的特征描述子从而造成误匹配。

从运动恢复结构的具体步骤

- 几何验证：

目的是消除特征误匹配。

由于特征描述子仅仅表达了特征点非常小的邻域内的局部信息，通过特征描述子进行的匹配也就只能是根据特征点的局部外观计算相似度。几何验证就是希望通过特征点之间的上下文信息减少和纠正误匹配。最简单的几何验证可通过特征点间的相对位置进行验证和重匹配，也可以根据特征点间的匹配构建两幅图像间的变换，对不符合变换的匹配进行删减和修正。

- 相机参数求解：

根据匹配好的特征点对相机进行标定，如果拍摄照片的相机内参未知，则我们不仅仅是求解相机的外参，而是需要求解整个基本矩阵，这就要求至少要有四对匹配点。

从运动恢复结构的具体步骤

- 三维点计算：

根据双目视觉理论求出对应的三维点。

由于特征点检测、相机自标定的误差，在图像序列中采用不同的图像会恢复出不同的三维点。解决的思路是采用优化方法使得求出的三维点在一定的度量下尽量符合不同图像对的约束。

- 集束调整：

增量式SFM的关键步骤，作为增量式方法，相机的参数由多个视图局部求取，当图像序列不断增加，相机参数和三维点的计算误差将会积累并失控。为此当图像序列增加到一定程度，需对计算好的相机参数和三维点根据已知信息统一考虑甲乙优化，以减少误差。

基于明暗的三维形状重建
(shape from shading)

基于明暗的三维形状重建 (shape from shading)

- 利用图像中物体表面的明暗变化来恢复其表面各点的相对高度或表面法向。
- 可以分为优化法、传播法、局部法和线性化方法四类。
- 较多采用的是优化法。

优化法通过最小化目标函数求解物体表面法向

$$\min_{\vec{\mathrm{n}}}\sum_{i,j}\left(\left(R\left(i,j,\vec{\mathrm{n}}(i,j)\right)-I(i,j)\right)^2+\lambda\left\|\vec{\mathrm{n}}(i,j)-\frac{1}{m}\sum\vec{\mathrm{n}}_a\right\|^2\right)$$

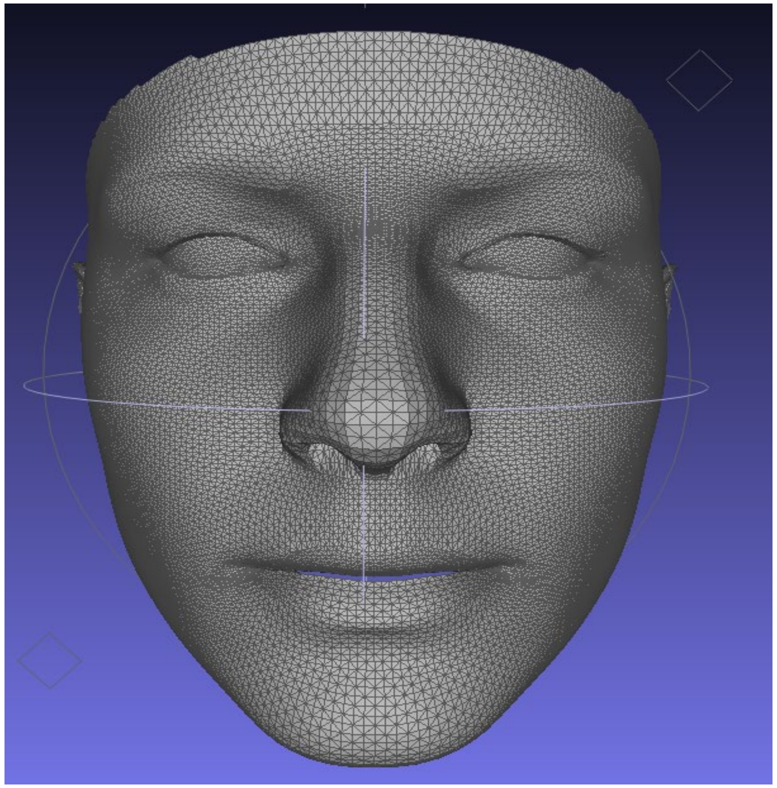
(i,j)为图像中属于需重建物体的所有像素点。$\vec{\mathrm{n}}(i,j)$ 为(i,j)处物体的法向，是我们需求取的。 $R\left(i,j,\vec{\mathrm{n}}(i,j)\right)$为$(i,j)$处根据法向、光源等绘制得到的颜色值，$I(i,j)$为输入图像的颜色值。$\left\|\vec{\mathrm{n}}(i,j)-\frac{1}{m}\sum\ \vec{\mathrm{n}}_a\right\|^2$为正则项，用于约束重建物体的光滑性，其中$\vec{\mathrm{n}}_a$指$(i,j)$周围m个邻接像素处的法向。可以通过针对梯度、高度等给出各种正则项，如$\|\nabla z\|^2$。

其中的难点是绘制$R\left(i,j,\vec{\mathrm{n}}(i,j)\right)$不仅需要$(i,j)$处的法向，还需要光源信息。而简单地将其定义为$R\left(i,j,\vec{\mathrm{n}}(i,j)\right)=\rho\vec{L}\cdot\vec{\mathrm{n}}(i,j)$（其中$\rho$为反射系数，$\vec{L}$为光源方向）难以描述自然环境下的光源。
为此，常用的方法是运用*球谐函数*（*Spherical Harmonics*， *SH*）描述光源，定义$R\left(i,j,\vec{\mathrm{n}}(i,j)\right)=\rho(i,j)\mathrm{f}\left(\vec{\mathrm{n}}(i,j)\right)$，$\mathrm{f}(\vec{\mathrm{n}})=\sum_{l=0}^{s}\sum_{m=-l}^{l}c_l^m y_l^m(\vec{\mathrm{n}})$，其中$y_l^m(\vec{\mathrm{n}})$*为球谐基。*

参数化人脸重建

参数化人脸模型

Mesh of BFM09



$$\mathbf{S} = \overline{\mathbf{S}} + \mathbf{A}_{id}\boldsymbol{\alpha}_{id} + \mathbf{A}_{exp}\boldsymbol{\alpha}_{exp},$$

$\overline{S}$:平均脸

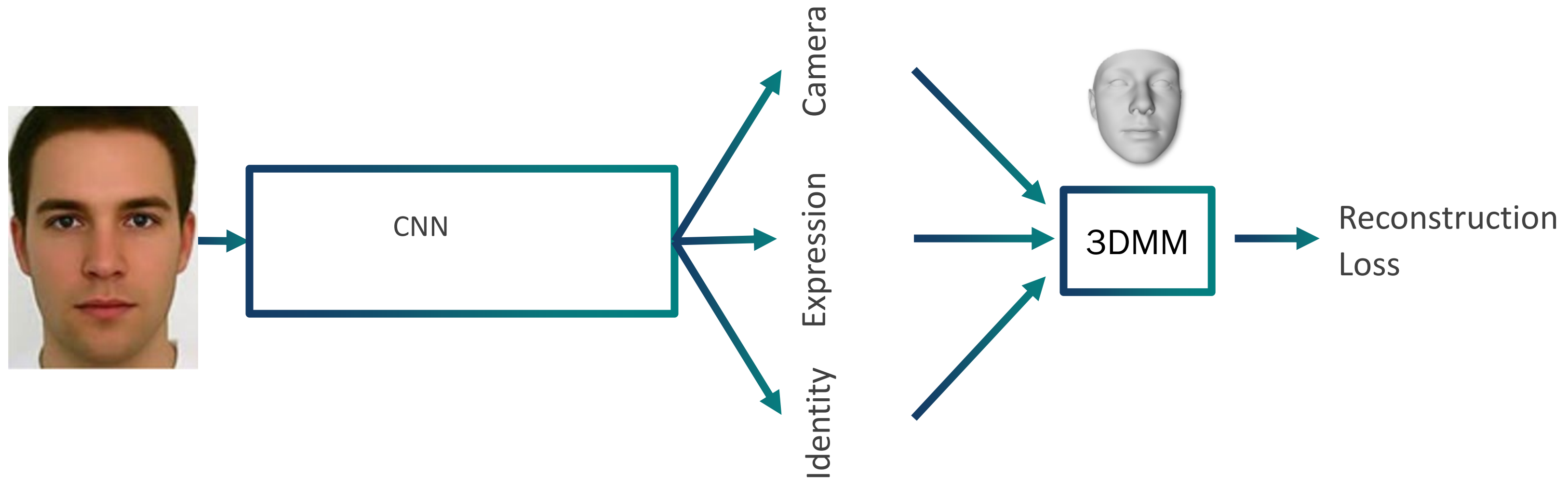
A_{id} : ID基

A_{exp}: 表情基

α_{id} : ID参数

α_{exp}: expression 参数

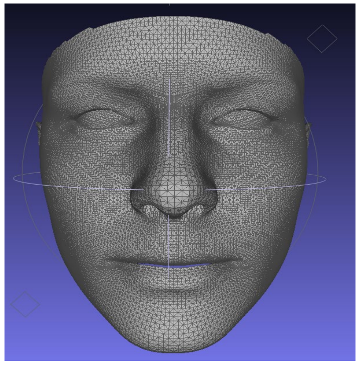
重建算法– 基础网络结构



- 输入:
  - RGB image
- 输出:
  - 3DMM 参数
    - ID(99)
    - expression(29)
  - camera参数（姿态）(7)

$$\mathbf{S} = \overline{\mathbf{S}} + \mathbf{A}_{id}\boldsymbol{\alpha}_{id} + \mathbf{A}_{exp}\boldsymbol{\alpha}_{exp},$$

Mesh of BFM09



$\overline{S}$:平均脸

A_{id} : ID基

A_{exp}: 表情基

α_{id} : ID参数

α_{exp}: expression 参数

相机参数化

- **平移旋转:**

$$\boldsymbol{V_{3d}} = \boldsymbol{R} * \left(\boldsymbol{\bar{S}} + \boldsymbol{A_{id}\alpha_{id}} + \boldsymbol{A_{exp}\alpha_{exp}}\right) + \boldsymbol{t_{3d}}$$

V_{3d}： 网格模型的3D坐标向量.
$\boldsymbol{R}$： 旋转矩阵.

- **投影:**
- **正交投影**

$$\boldsymbol{V_{2d}} = f * \mathrm{Pr} * \boldsymbol{R} * \left(\boldsymbol{\bar{S}} + \boldsymbol{A_{id}\alpha_{id}} + \boldsymbol{A_{exp}\alpha_{exp}}\right) + \boldsymbol{t_{2d}}$$

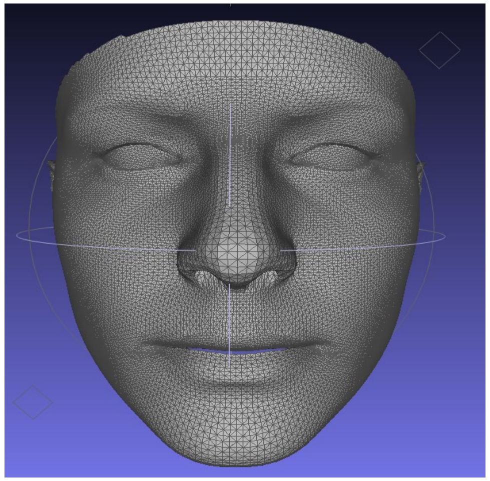
人脸重建损失函数

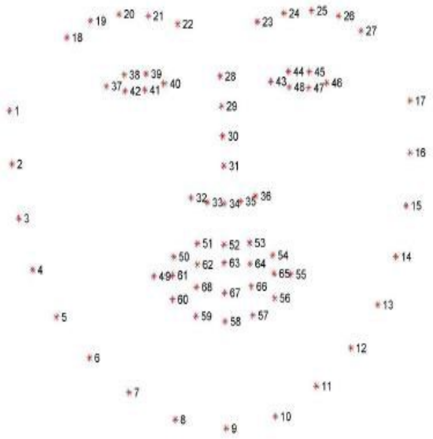
- L_{recon} **由3部分组成.**

$$L_{recon} = \left(L_{reproj} + \lambda \mathbb{I} L_{3D}\right) + \mu L_{regular}$$

- L_{reproj}：约束了ground truth landmark $\hat{V}_{2d}$ 与 网格投影点landmark V_{2d} 之间的差异.
- N：landmark数量.

$$L_{reproj} = \frac{1}{2N} ||V_{2d} - \hat{V}_{2d}||_2^2$$





人脸重建损失函数

- L_{recon} **由3部分组成.**

$$L_{recon} = \left(L_{reproj} + \lambda \mathbb{I} L_{3D}\right) + \mu L_{regular}$$

- L_{3D}为参数误差

L_{3D}:　　L_{para}　**or**　L_{3dmesh}

$$L_{para} = \frac{1}{N}\sum_{i=1}^{N}\left(\alpha_{id}^{i} - gt\alpha_{id}^{i}\right)^2 + \frac{1}{M}\sum_{i=1}^{M}\left(\beta_{exp}^{i} - gt\beta_{exp}^{i}\right)^2$$

$$\mathrm{L}_{3dmesh} = \left\|\bar{\mathrm{S}} + \sum \ \alpha_{id}A_{id} + \sum \ \beta_{exp}\mathrm{B}_{exp} - gtS\right\|$$

人脸重建损失函数

- L_{recon} **由3部分组成.**

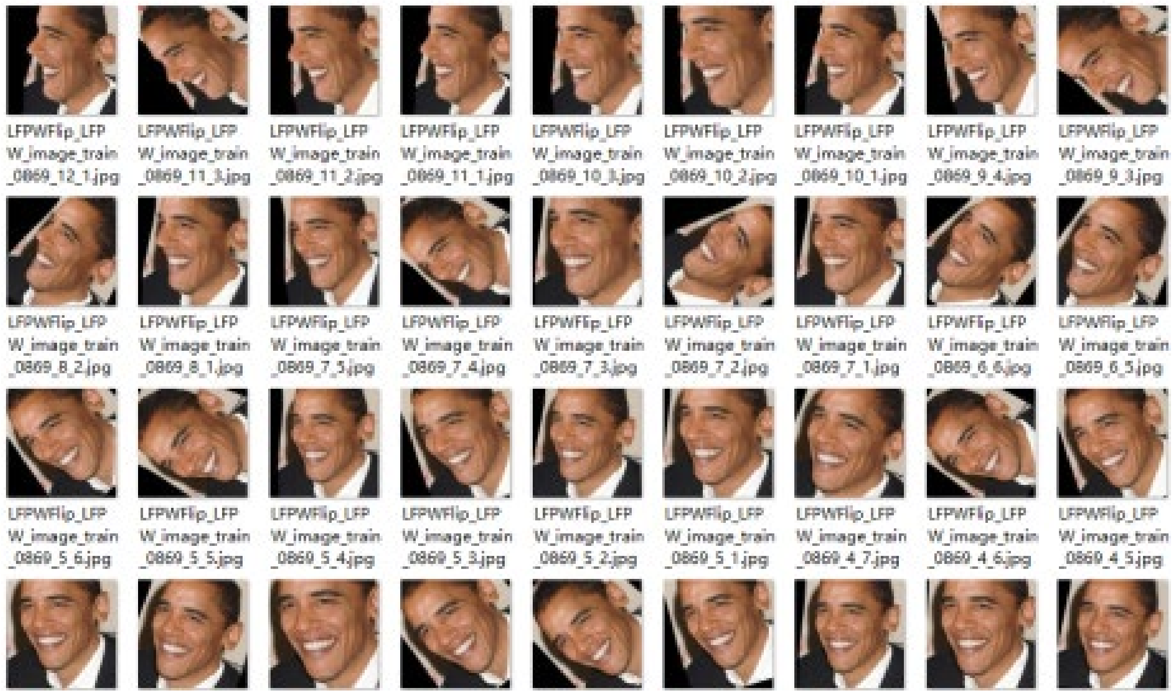
$$L_{recon} = \left(L_{reproj} + \lambda \mathbb{I} L_{3D}\right) + \mu L_{regular}$$

- 正则项

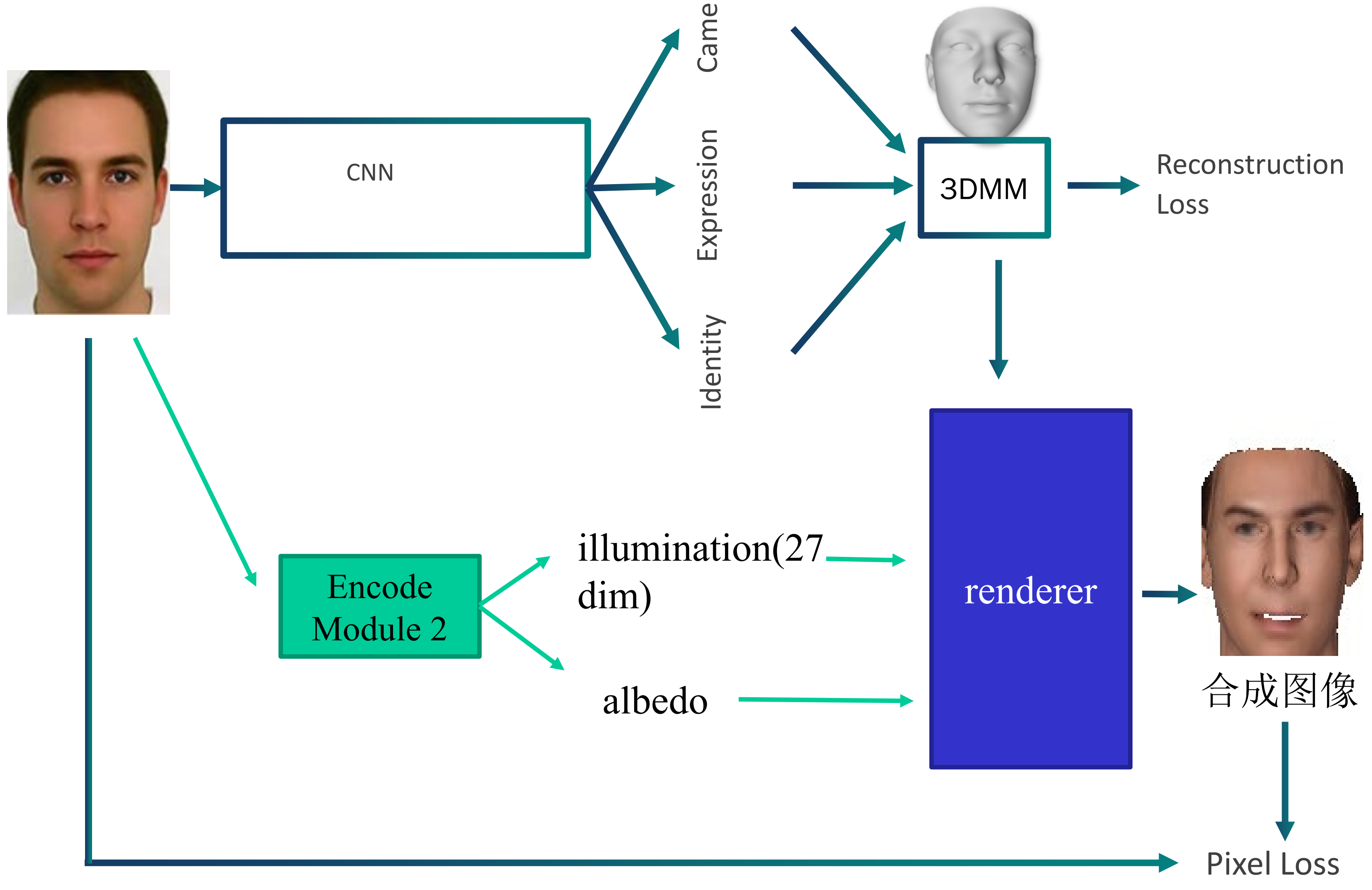
$$\begin{aligned} L_{regular} = &\left\| Mean[s_b] - \vec{0}_{99} \right\|_2^2 + \left\| Mean[e_b] - \vec{0}_{29} \right\|_2^2 \\ &+ \left\| Var[s_b] - \vec{1}_{99} \right\|_2^2 + \left\| Var[e_b] - \vec{1}_{29} \right\|_2^2 \end{aligned}$$

人脸重建训练集

- **训练集:**
  - 增广的300W数据集:
    - 3DMM 参数
    - 相机参数


LFPWFlip_LFPW_image_train_0869_12_1.jpg
LFPWFlip_LFPW_image_train_0869_11_3.jpg
LFPWFlip_LFPW_image_train_0869_11_2.jpg
LFPWFlip_LFPW_image_train_0869_11_1.jpg
LFPWFlip_LFPW_image_train_0869_10_3.jpg
LFPWFlip_LFPW_image_train_0869_10_2.jpg
LFPWFlip_LFPW_image_train_0869_10_1.jpg
LFPWFlip_LFPW_image_train_0869_9_4.jpg
LFPWFlip_LFPW_image_train_0869_9_3.jpg
LFPWFlip_LFPW_image_train_0869_8_2.jpg
LFPWFlip_LFPW_image_train_0869_8_1.jpg
LFPWFlip_LFPW_image_train_0869_7_5.jpg
LFPWFlip_LFPW_image_train_0869_7_4.jpg
LFPWFlip_LFPW_image_train_0869_7_3.jpg
LFPWFlip_LFPW_image_train_0869_7_2.jpg
LFPWFlip_LFPW_image_train_0869_7_1.jpg
LFPWFlip_LFPW_image_train_0869_6_6.jpg
LFPWFlip_LFPW_image_train_0869_6_5.jpg
LFPWFlip_LFPW_image_train_0869_5_6.jpg
LFPWFlip_LFPW_image_train_0869_5_5.jpg
LFPWFlip_LFPW_image_train_0869_5_4.jpg
LFPWFlip_LFPW_image_train_0869_5_3.jpg
LFPWFlip_LFPW_image_train_0869_5_2.jpg
LFPWFlip_LFPW_image_train_0869_5_1.jpg
LFPWFlip_LFPW_image_train_0869_4_7.jpg
LFPWFlip_LFPW_image_train_0869_4_6.jpg
LFPWFlip_LFPW_image_train_0869_4_5.jpg

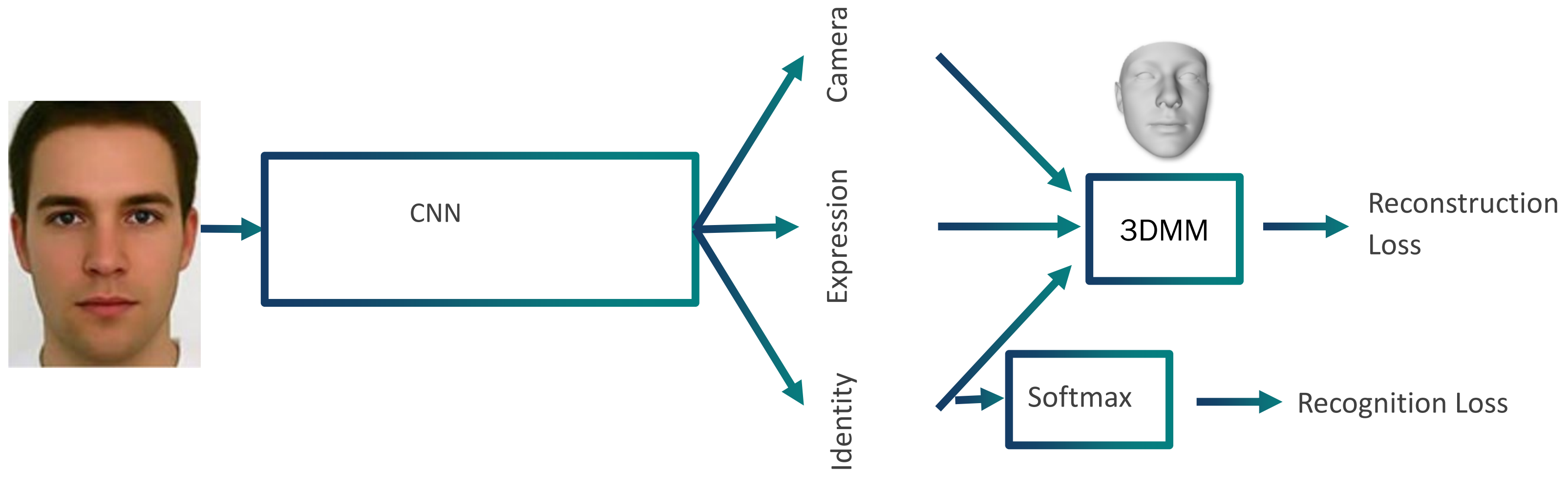
网络改进



CNN
Camera
Expression
Identity
3DMM
Reconstruction Loss
Encode Module 2
illumination(27 dim)
albedo
renderer
合成图像
Pixel Loss

网络改进

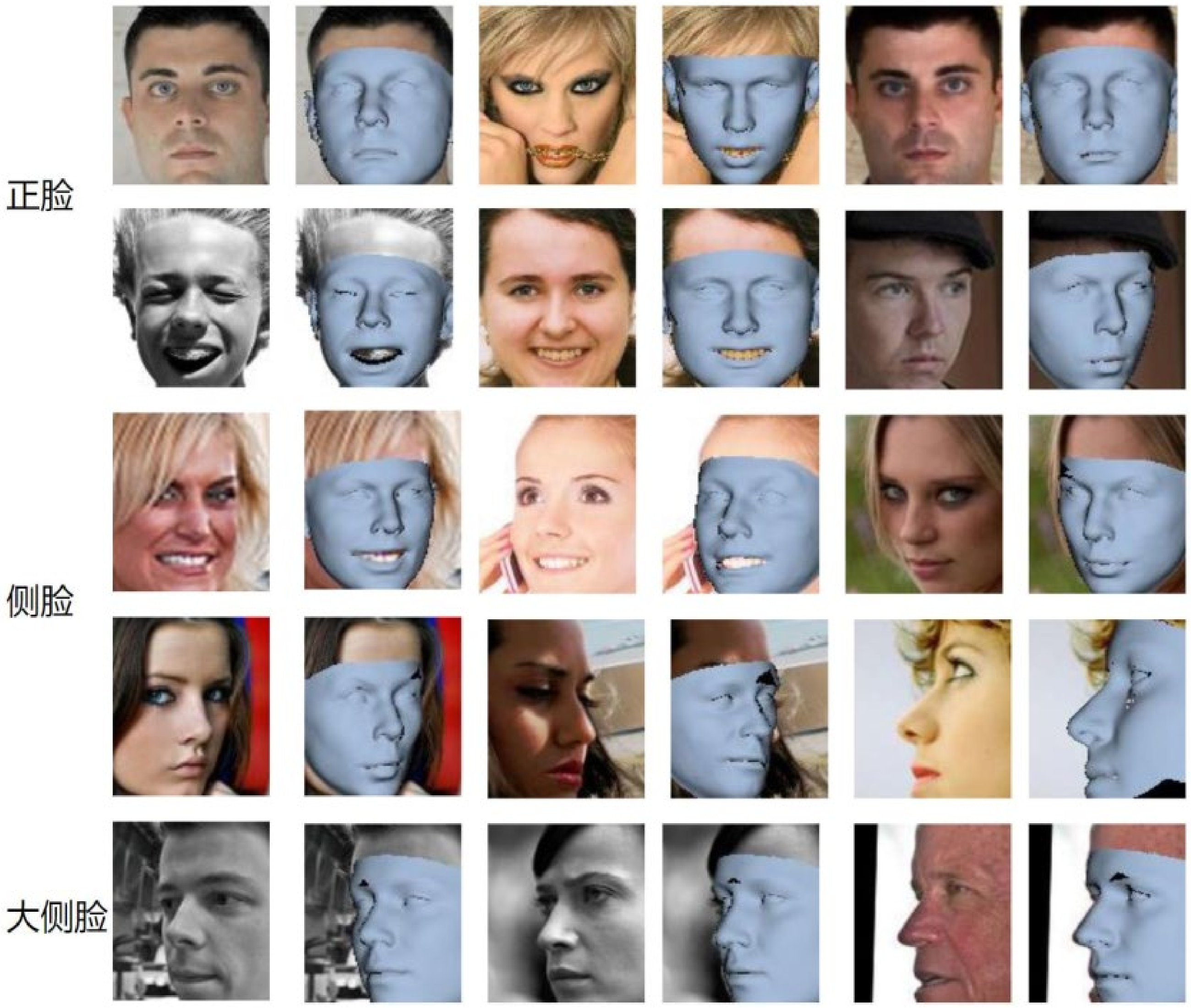
为增强人脸网格的可识别性，结合人脸识别损失函数，对网络进行改进在参数域进行聚类，提高重建人脸网格的可识别性



$$\mathrm{d} = \|S_1 - S_2\| = \|\bar{S} + A\alpha_1 - (\bar{S} + A\alpha_2)\| = \|A(\alpha_1 - \alpha_2)\|$$

$$\mathrm{d} = \|A(\alpha_1 - \alpha_2)\| = (\alpha_1 - \alpha_2)^T A^T A(\alpha_1 - \alpha_2) = (\alpha_1 - \alpha_2)^T(\alpha_1 - \alpha_2) = \|\alpha_1 - \alpha_2\|$$

重建效果



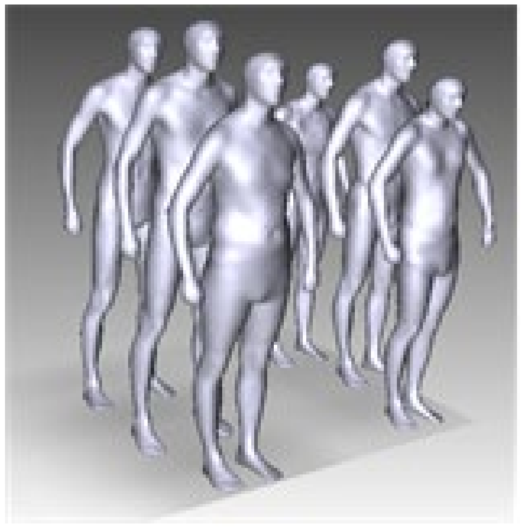
算法在 AFLW2000 数据集上的可视化结果

参数化人体重建

人体模型的参数化表达

SCAPE 是一种使用身材参数和姿态参数表示人体网格模型的参数化人体模型，其基本思路是通过对模板人体进行变形得到个性化人体网格。

SCAPE 模型有51个姿态参数，用θ表示和一个30维的身材参数，用β的表达。



人体模型的参数化表达

当人体的身材参数和姿态参数确定时，人体表面上的顶点可由下式计算。

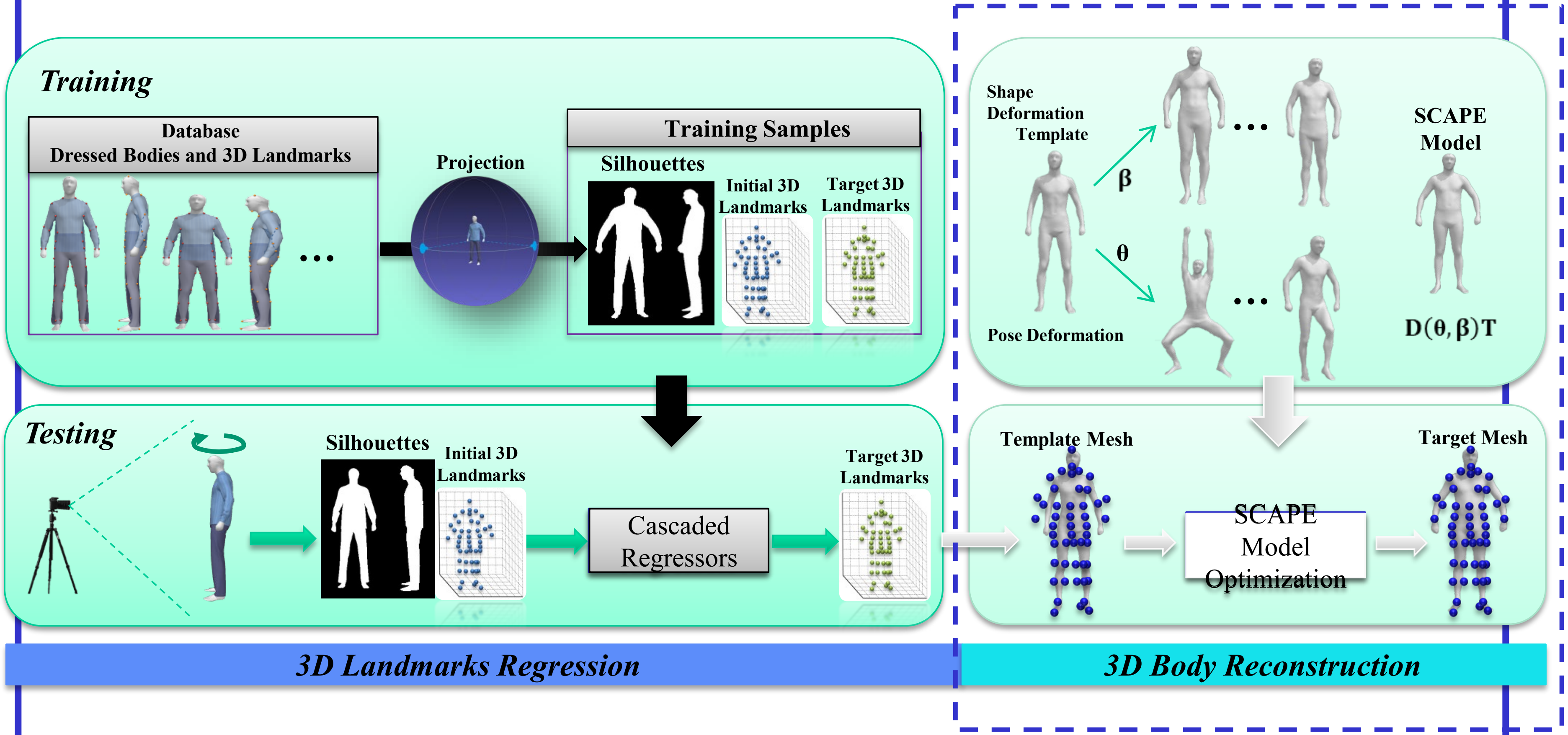
$$\underset{y1,\ldots,yn}{\arg\min} \sum_{k} \sum_{j=2,3} \left\| R_k(\theta) S_k(\beta) Q_k(\theta) \hat{e}_{j,k} - (y_{j,k} - y_{1,k}) \right\|^2$$

其中，$\hat{e}_{j,k}$为模板人体第k个三角形的边，$y_{j,k}$为目标人体第k个三角形中的顶点。

反过来，当已知人体表面顶点时，可通过下式求解下述优化问题得到人体的身材和姿态参数。

$$\underset{\theta,\beta}{\arg\min} \sum_{k} \sum_{j=2,3} \left\| R_k(\theta) S_k(\beta) Q_k(\theta) \hat{e}_{j,k} - (y_{j,k} - y_{1,k}) \right\|^2$$

人体模型的参数化重建

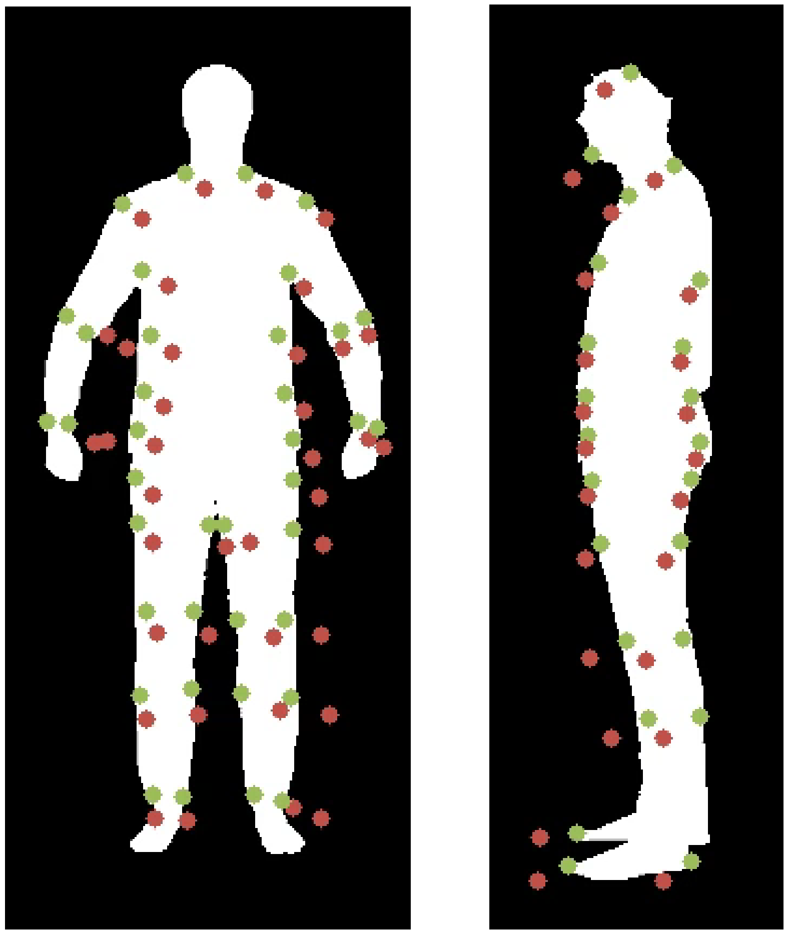


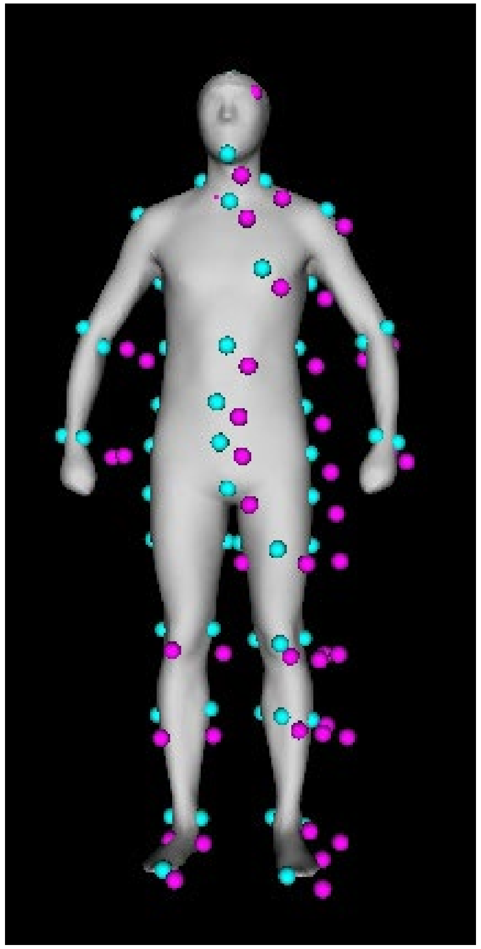
人体站立照片 → 人体关键点→ SCAPE模型参数

重建效果

3D 关键点回归

人体重建


2D Views


3D View

谢　谢!

造 型

网格

曲面造型

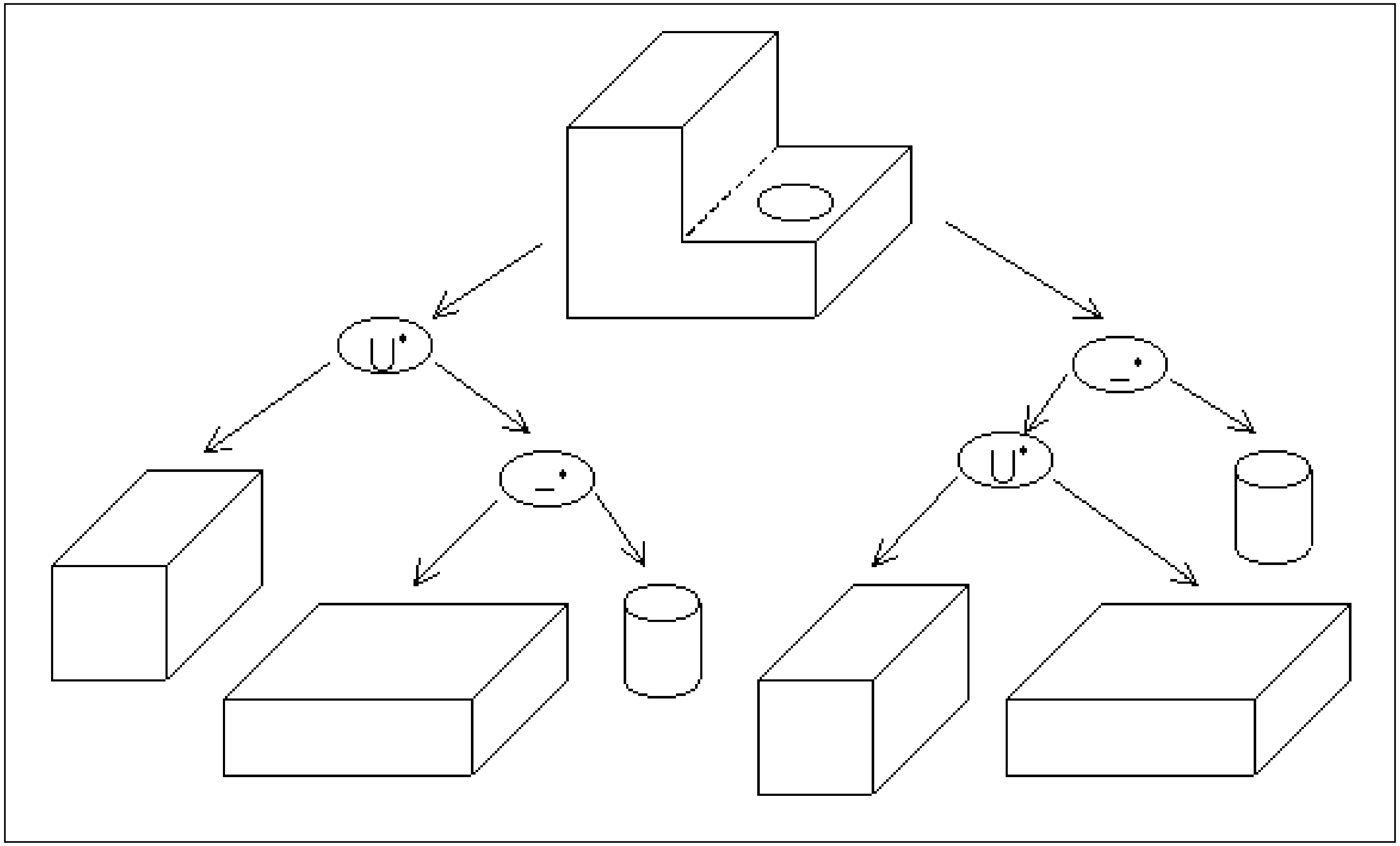
实体造型

其它方法

物体表达的基本方法

- **CSG树 Constructive Solid Geometry**
- 分解表达 Decomposition
- 边界表达 Boundary representations

- **CSG树**

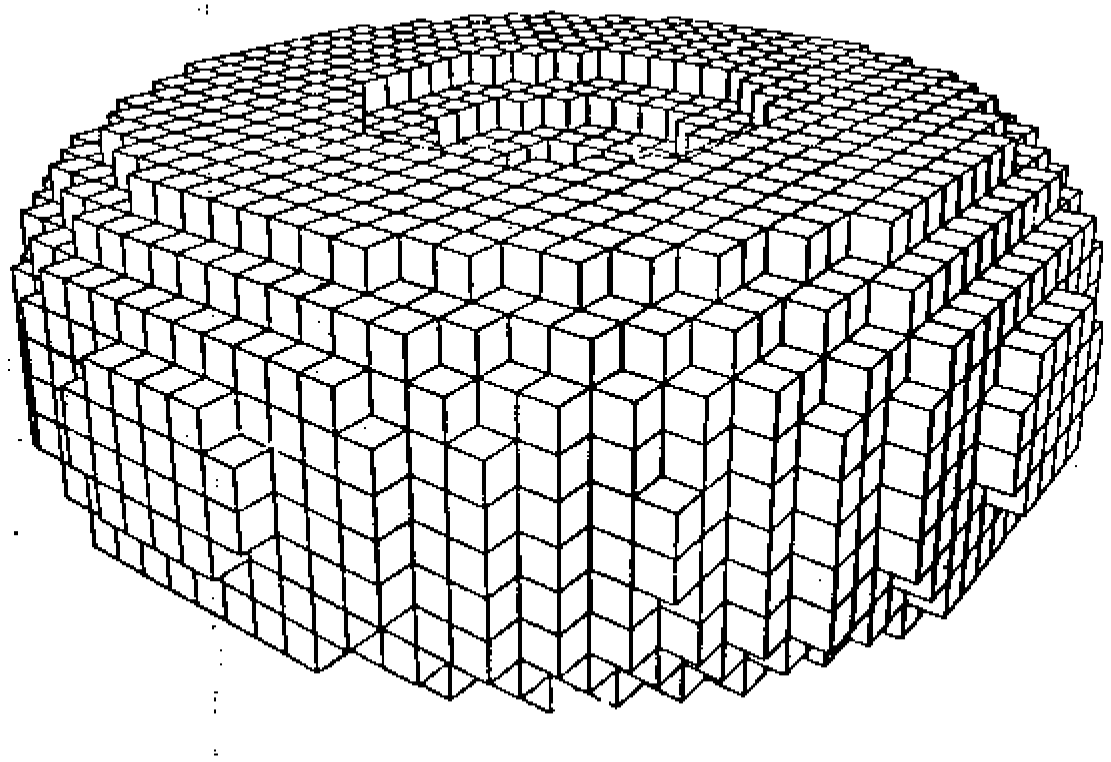


CSG 表达

- 通过布尔运算组合长方体、柱体等简单体元.
- 以树的形式存储：叶节点为基本体元，中间节点为布尔运算或变换.
- 基本体元可以是长方体、柱体、球、锥体等.
- 表达不唯一，但紧凑.
- 数据结构简单.
- 全局操作简单：树的合并、子树删除.
- 可以转化为BREP.
- 局部操作困难.
- 显示困难.

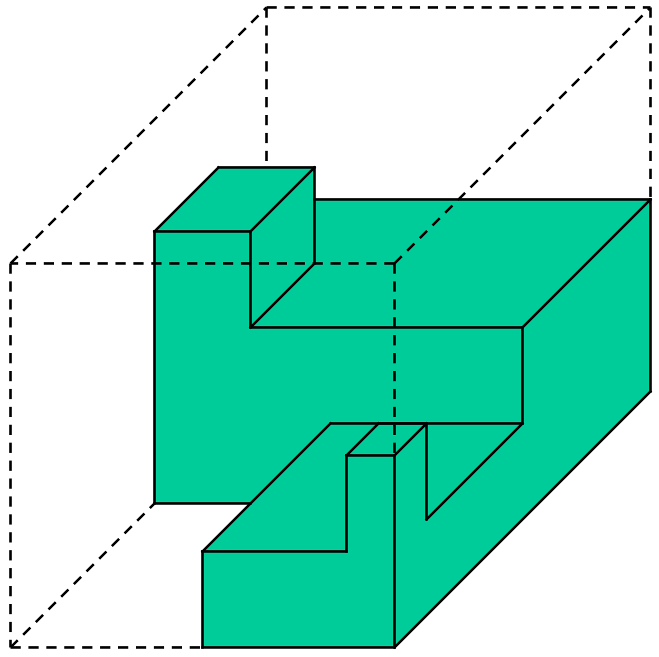
物体表达的基本方法

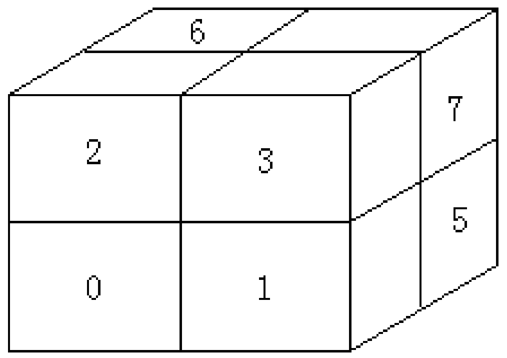
- CSG树 Constructive Solid Geometry
- **分解表达 Decomposition**
- 边界表达 Boundary representations

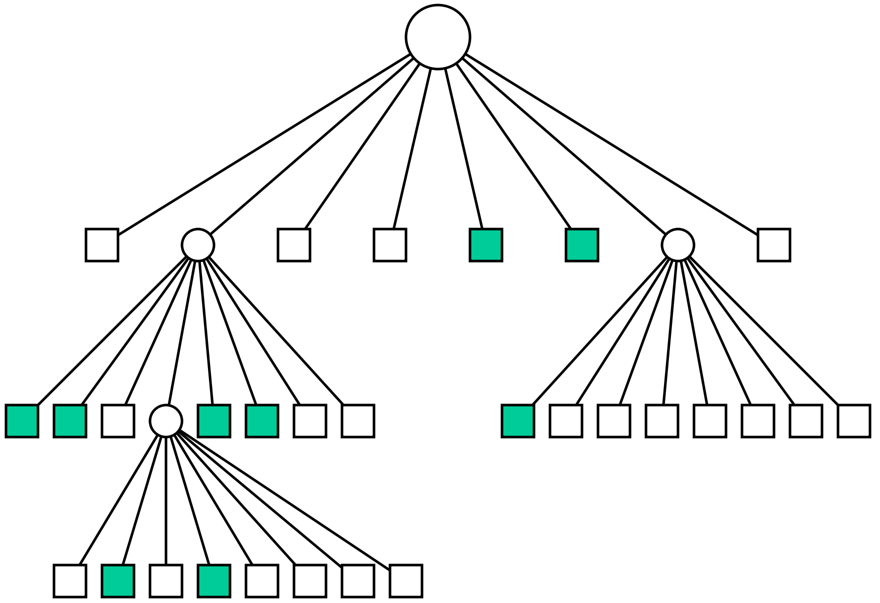


用体素表示物体，类似像素表示二值图像

八叉树表达







Primary schemes for representing solid objects

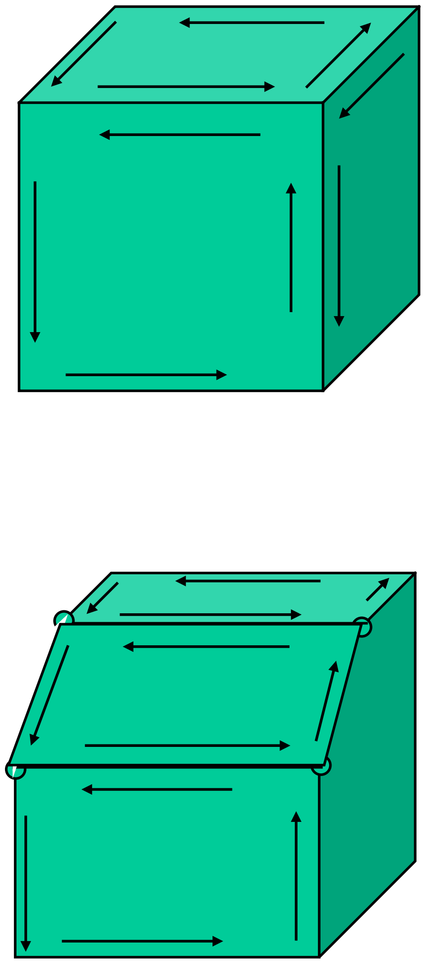
- Constructive Solid Geometry
- Decomposition
- **Boundary representations**

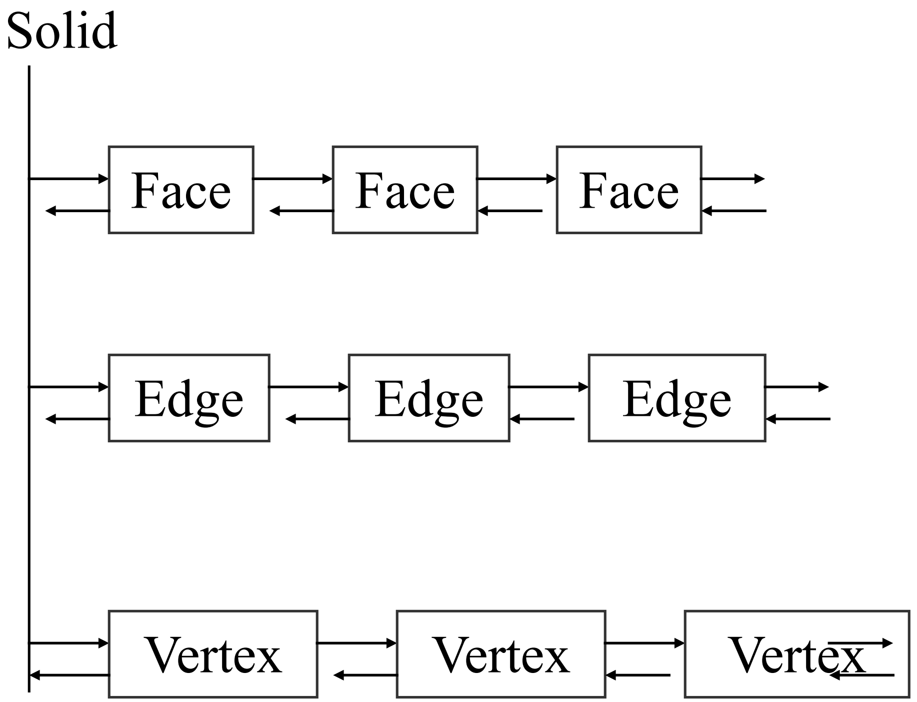
Brep: Boundary Representation

- Represent the object in terms of its surface boundaries, vertices, edges and faces.
- Some of the systems support only planar faces and curves surfaces are often approximated by planar polygons.
- Boundaries should be “2-manifold”.

  Every point has some arbitrary small neighborhood of points around it that can be considered topologically same as a disk on a plane.
- Well suited for direct manipulations.
- Advanced operations like boolean operations are plagued by numerical problems, leading to undependable results.

Brep: Boundary Representation

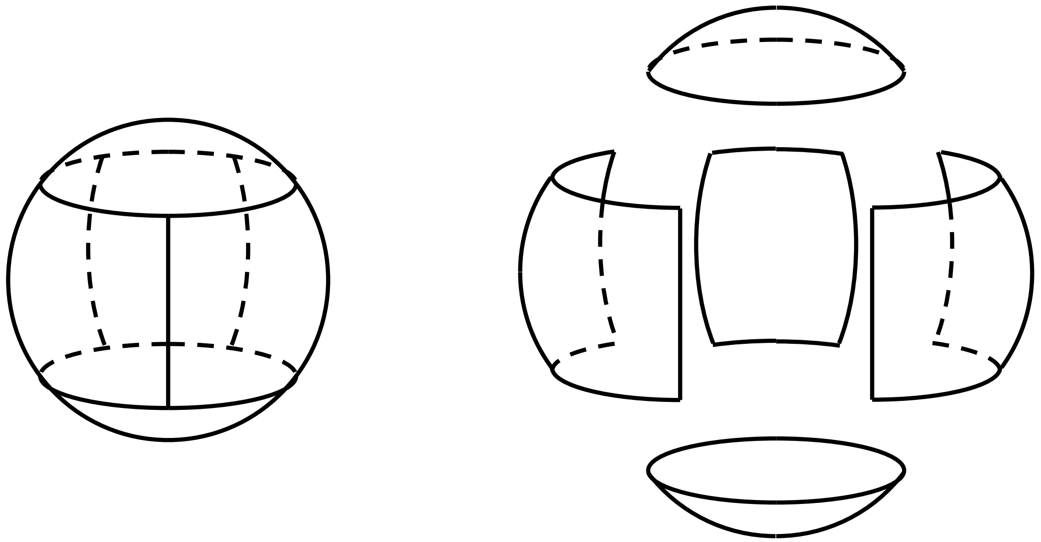




Typically two types of data is stored for each object

- Geometric data
- Topological data

Geometric data consists of basic shape definition parameters such as coefficients of the bi-cubic surface, where as Topological data includes the connectivity relationships among the geometric components.



A “Polyhedron” is a solid bounded by a set of polygons such that two and only two polygons meet at an edge and it is possible to traverse the polyhedron by crossing the edges and moving from one face to the other.

A “Simple Polyhedron” is one that can be deformed into a sphere. The relationship between vertices, edges and faces is

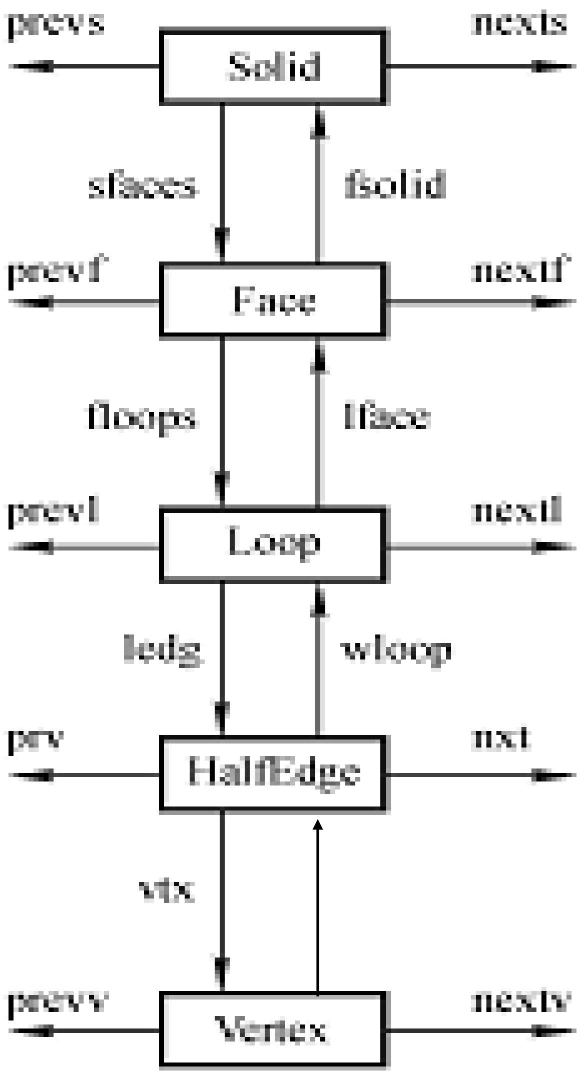
$$V - E + F = 2$$

which is known as Euler’s Formula.

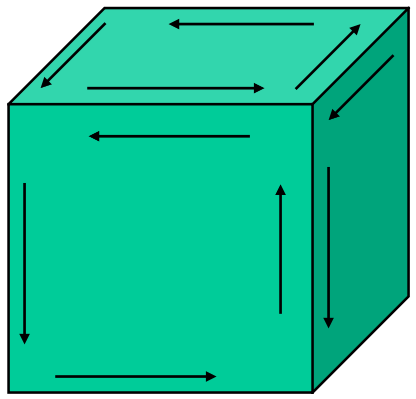
For a general polyhedra, I.e. a polyhedra that have faces with holes, the generalisation of the Euler’s formula holds

$$V - E + F - L = 2\,(S - G)$$

5 types: Solid、Face、Loop、HalfEdge、Vertex:

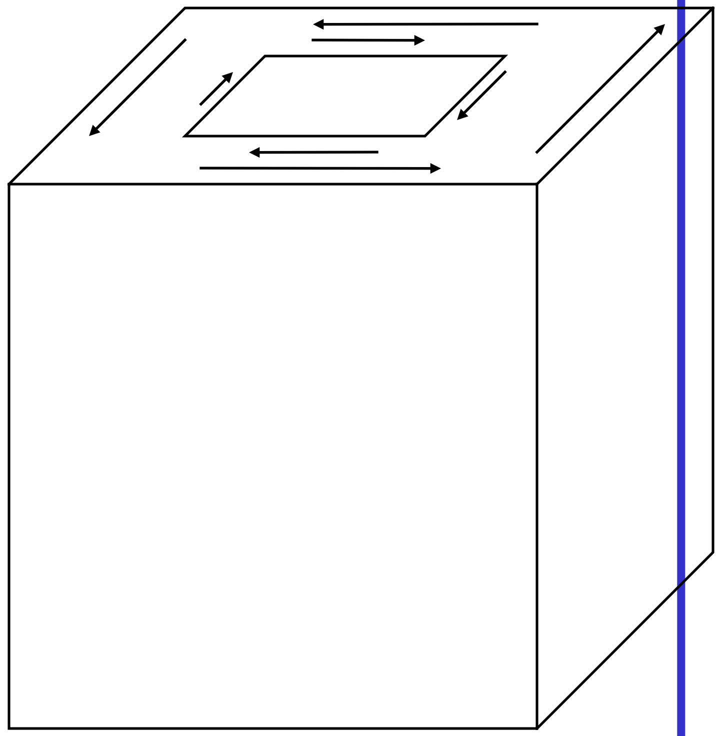


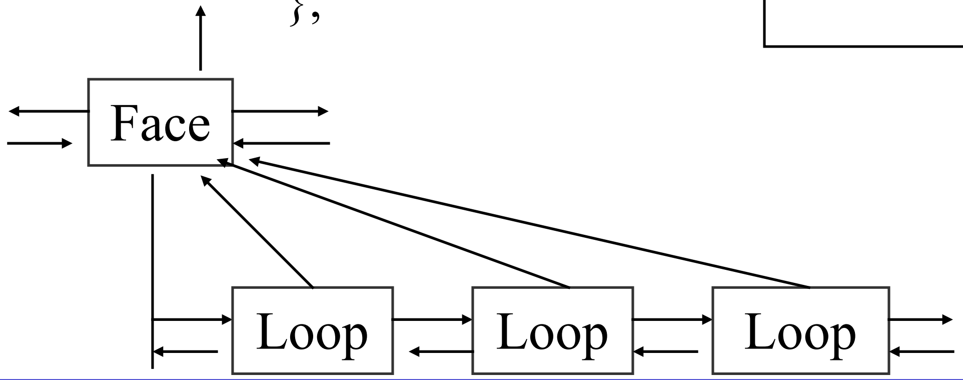
typedef struct solid   Solid;
typedef struct face   Face;
typedef struct loop   Loop;
typedef struct halfedge HalfEdge;
typedef struct vertex   Vertex;
typedef struct edge   Edge;
typedef union nodes   Node;



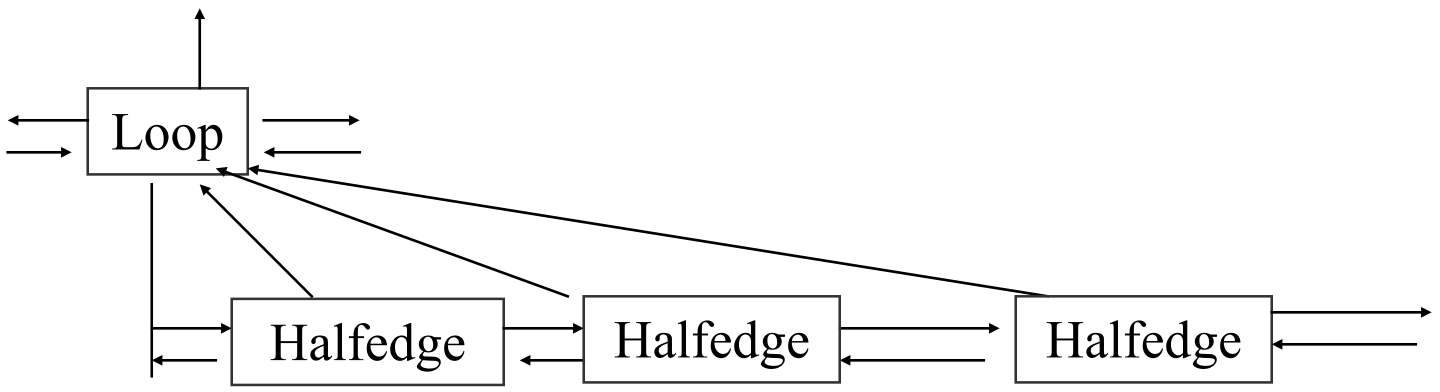
```
struct solid
{
    Id      Solidno;  /*solid identifier*/
    Face    *sfaces;  /*pointer to list faces*/
    Edge    *sedges;  /*pointer to list of vertices*/
    Vertex  *sverts;  /*pointer to list of solid*/
    Solid   *nexts;  /*pointer to next solid*/
    Solid   *prevs;  /*pointer to previous solid*/
};
```

```
struct face
{
Id  faceno;  /*face identifier*/
Solid *fsolid;  /*back pointer to solid*/
Loop *flout;  /*pointer to outer loop*/
Loop *floops; /*pointer to list of
                 loops*/
vector  feq;  /*face equation*/
SURFACE *fgeom ;
Face *nextf;  /*pointer to next face*/
Face *prevf;  /*pointer to previous
                 face*/
};
```



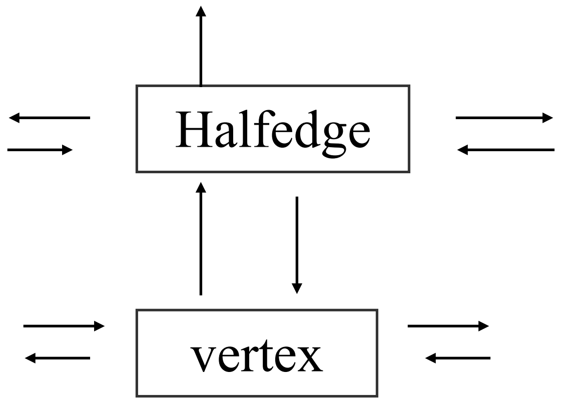


```
struct loop
{
    HalfEdge *lege;  /*pointer to ring of halfedges*/
    Face *lface;  /*back pointer to face*/
    Loop *nextl;  /*pointer to next loop*/
    Loop *prevl;  /*pointer to previous face*/
};
```



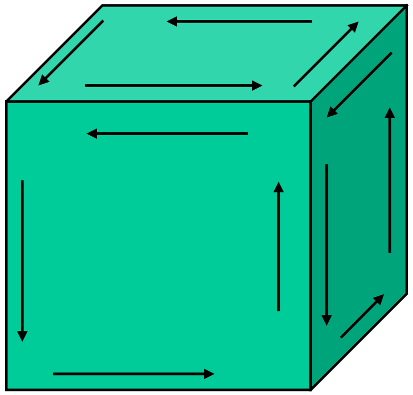
10.2.2 半边结构程序描述

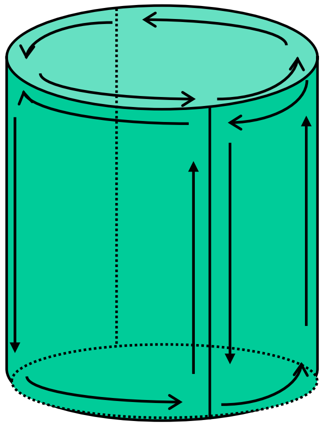
```
struct halfedge
{
Edge *edg;  /*pointer to parent edge*/
Vertex *vex;  /*pointer to starting vertex*/
Loop *wloop;  /*back pointer to loop*/
HalfEdge *nxt;  /*pointer to next halfedge*/
HalfEdge *prv;  /*pointer to previous
                            halfedge*/
};
```

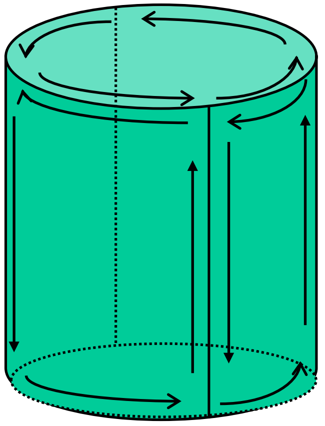


```
struct vertex
{
    Id 1 vertexno;          /*vertex identifier*/
    HalfEdge *vedge;        /*pointer to a halfedge*/
    vector vcoord;          /*vertex coordinates*/
    Vertex *nextv;          /*pointer to next vertex*/
    Vertex *prevv;          /*pointer to previous
                                   vertex*/
};
```

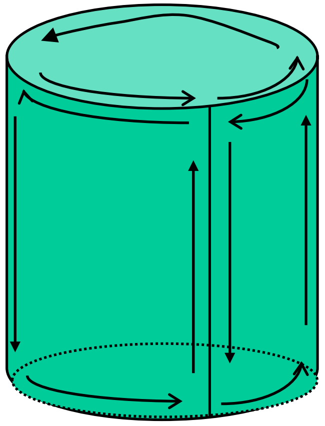
```
struct edge
{
    HalfEdge *he1;  /*pointer to right
                                  halfedge*/
     HalfEdge *he2;  /*pointer to left halfedge*/
        Edge *nexte;  /*pointer to next edge*/
      Edge *preve;  /*pointer to previous edge*/
                 Curve *cgeom ;
                         };
```





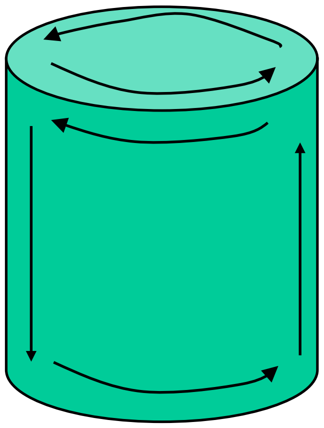


$8v-12e+6f=2$

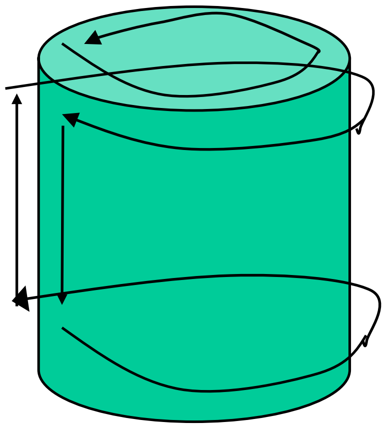


$6v-9e+5f=2$

$v-e+f=2$



$4v-6e+4f=2$



$2v-3e+3f=2$

Euler’s Operators

The creation and manipulation of typical boundary models can be encapsulated into small set of basic operations. These operators are called Euler’s Operators and preserve the topological validity.
A linear combination of the primitive Euler Operators is capable of representing any admissible transition.

M(make) K(kill) V(vertex) E(edge)F(face) S(solid) H(hole)R(ring)

$$v-e+f=2(s-h)+r$$

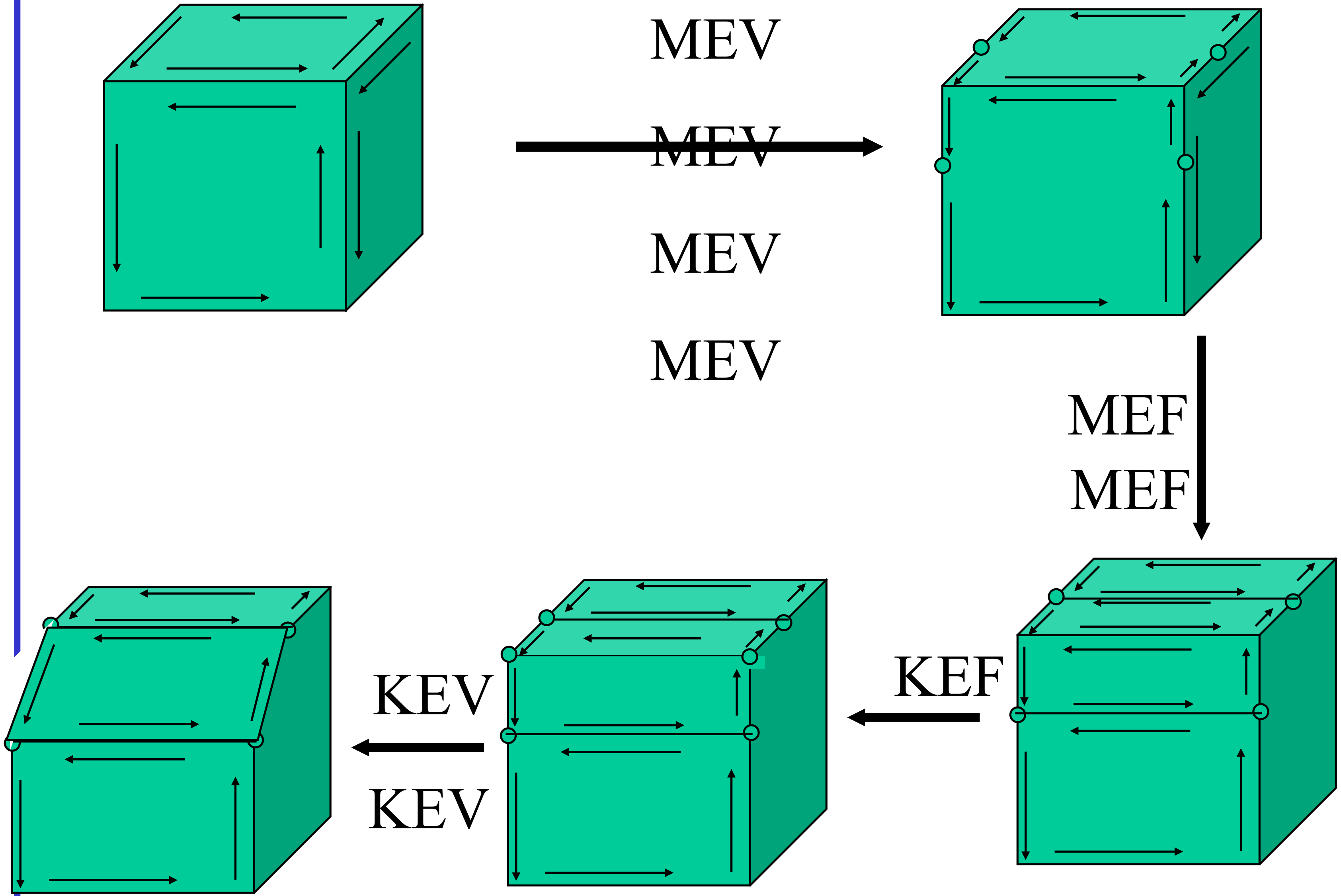
Primitive Euler Operators

- Make an edge and a vertex (MEV)
- Make a face and an edge (MEF)
- Make a Solid, a face, and a vertex, no edge and ring (MFVS)
- Make a cavity or passage and a body (KFMRH)
- Make an edge and kill a Ring (MEKR)

Each of these five operators has a corresponding complementary operator.

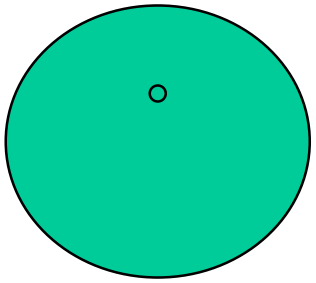
M(make) K(kill) V(vertex) E(edge) F(face) S(solid) H(hole) R(ring)

Example of local operation



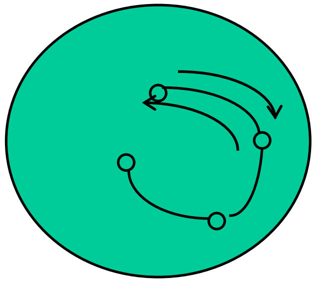
立方体的生成

1. MVFS



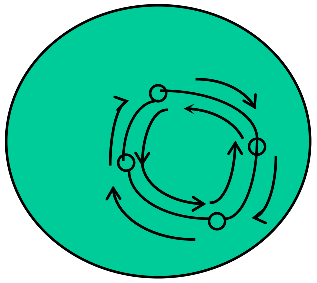
1 solid， 1 face ， 1 vertex

2. MEV
MEV
MEV

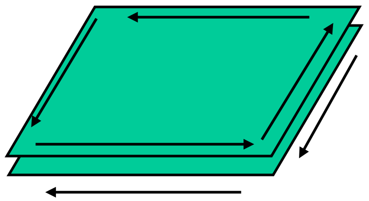


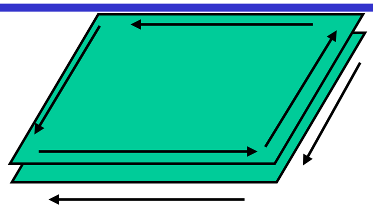
1solid， 1 face， 3 edges，
4 vertexes

3. MEF



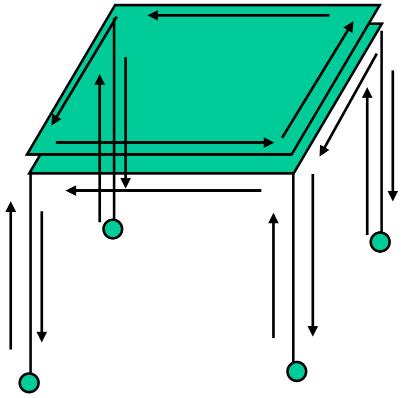
1solid， 2 faces， 4 edges
，
4 vertexes





4. MEV

MEV
MEV
MEV

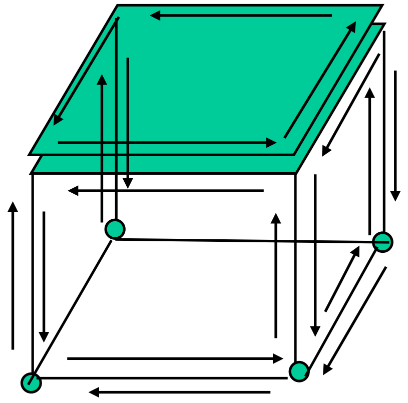


5. MEF

MEF

MEF

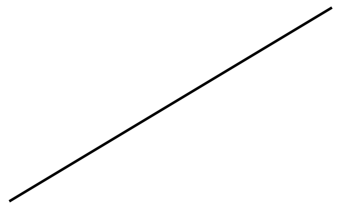
MEF



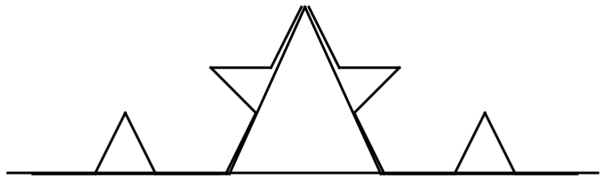
Comparison of the representation schemes

- Accuracy: Spatial partitioning and polygonal BREP are approximate representations, where as primitive instancing, CSG, & BREPs with curved surfaces are more accurate.
- Domain: Limited for sweep representation and primitive instancing. Spatial partitioning schemes can represent any solid.
- Uniqueness: Only Octree and spatial enumeration techniques guarantee uniqueness. If the primitives are carefully chosen primitive instancing can guarantee uniqueness.
- Validity: B-REPs are difficult to validate. No checking is needed for spatial occupancy enumeration and local checking for CSG and Octrees.
- Closure: Primitive instancing and sweeps are not closed under boolean operations. BREPs are, in general, closed.

Fractal



1. Helge Von Koch



2. Mandelbrot

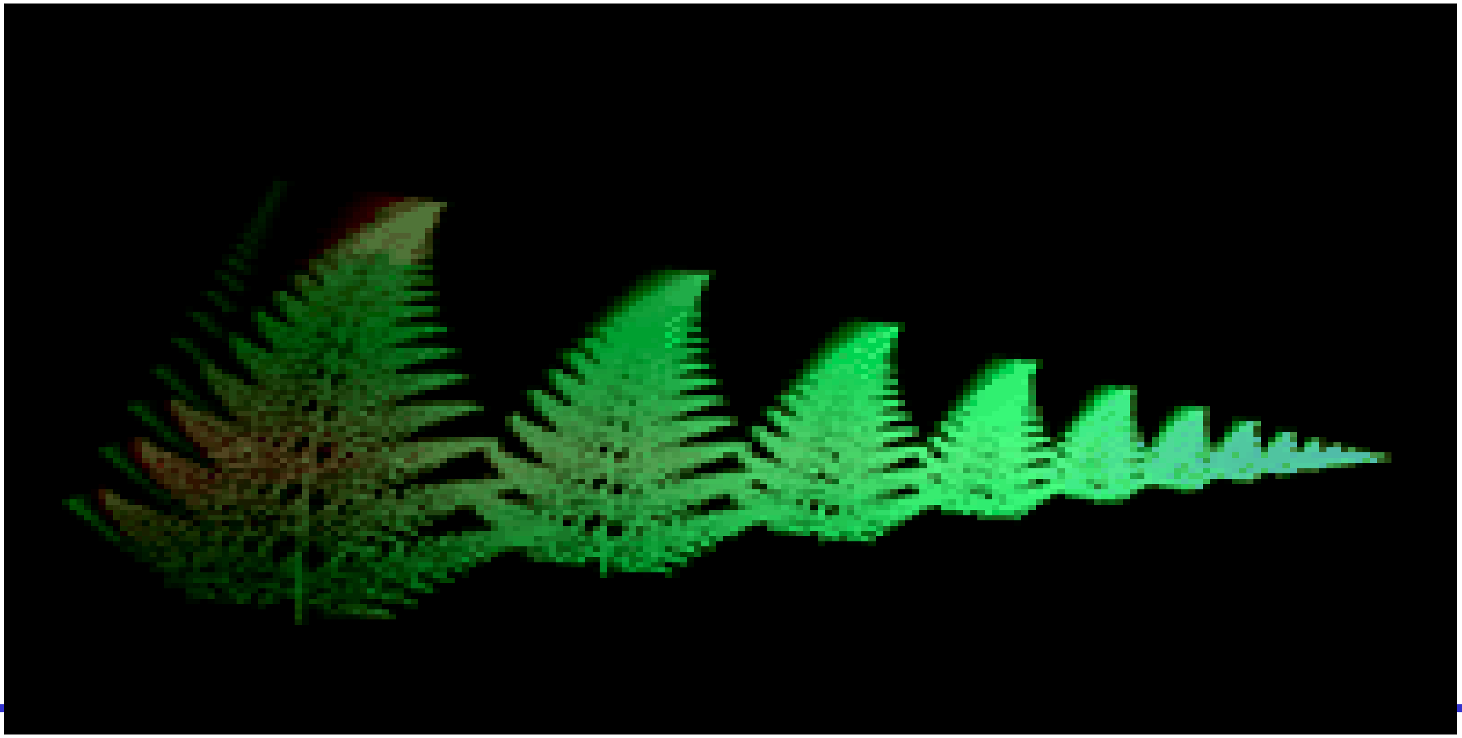
3. fractal dimension

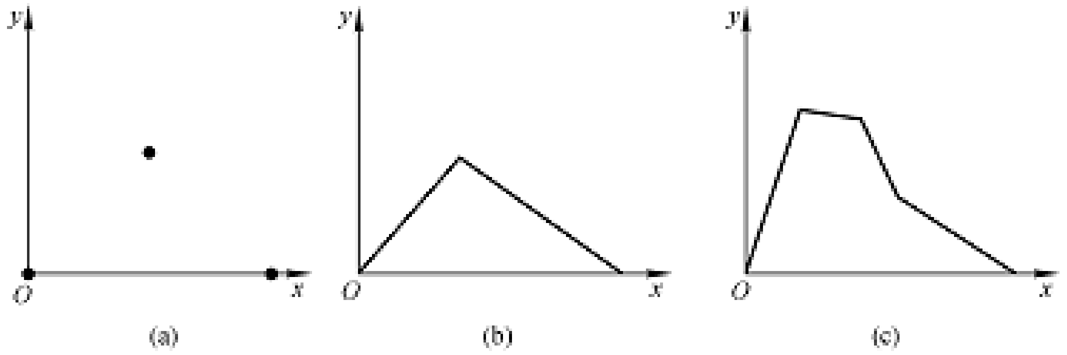
$$D = (\log N)/\log(1/S)$$

N: the number of sub-shape,

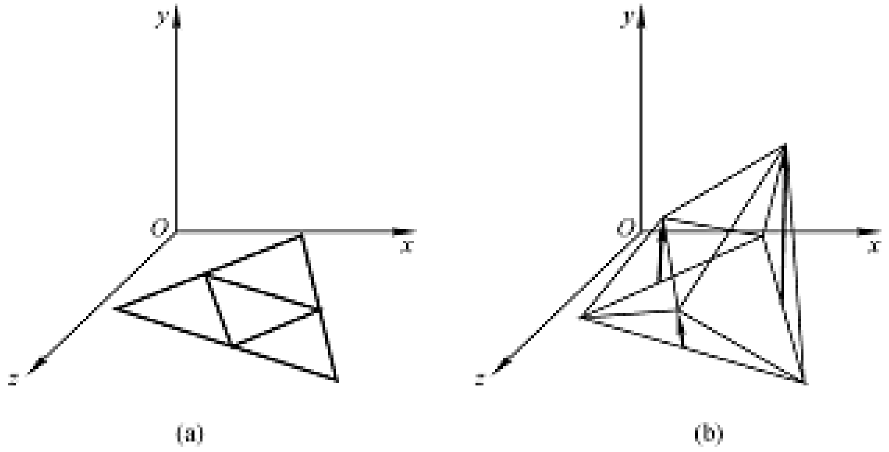
S: scale

Fr



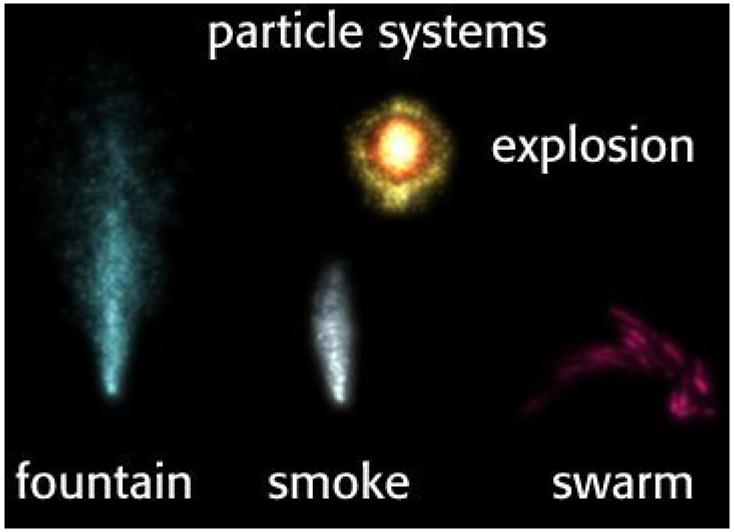


seashore



mountain

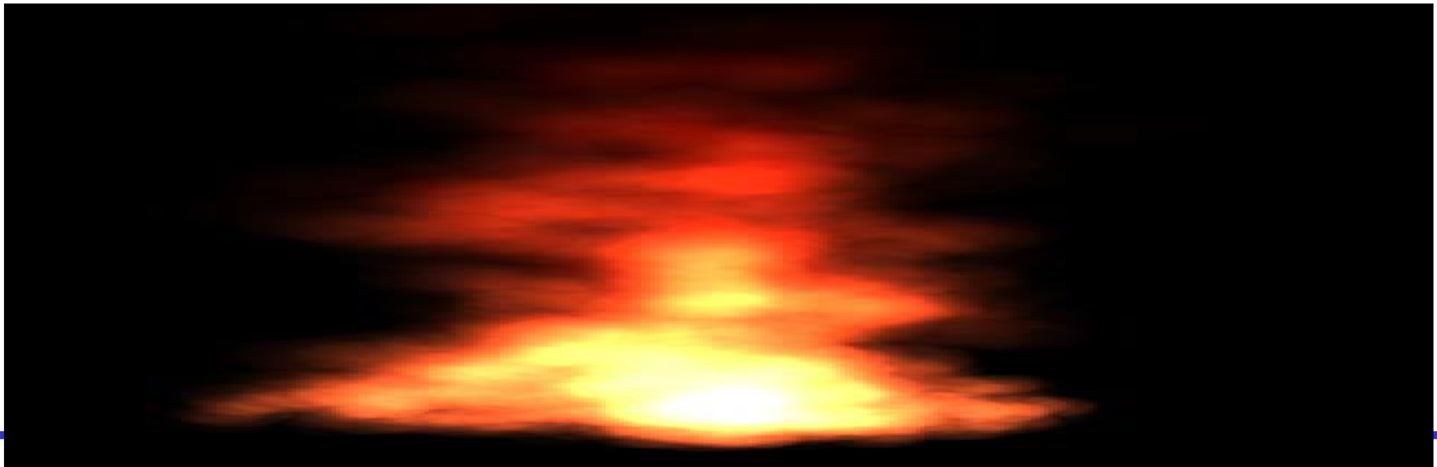
Particle Systems



Particle systems offer a solution to modeling amorphous, dynamic and fluid objects like clouds, smoke, water, explosions and fire.

Representing Objects with Particles

- An object is represented as clouds of primitive particles that define its volume rather than by polygons or patches that define its boundary.
- A particle system is dynamic, particles changing form and moving with the passage of time.
- Object is not deterministic, its shape and form are not completely specified. Instead



Basic Model of Particle Systems

1) New particles are generated into the system.
2) Each new particle is assigned its individual attributes.
3) Any particles that have existed past their prescribed lifetime are extinguished.
4) The remaining particles are moved and transformed according to their dynamic attributes.
5) An image of the particles is rendered in the frame buffer, often using special purpose algorithms.

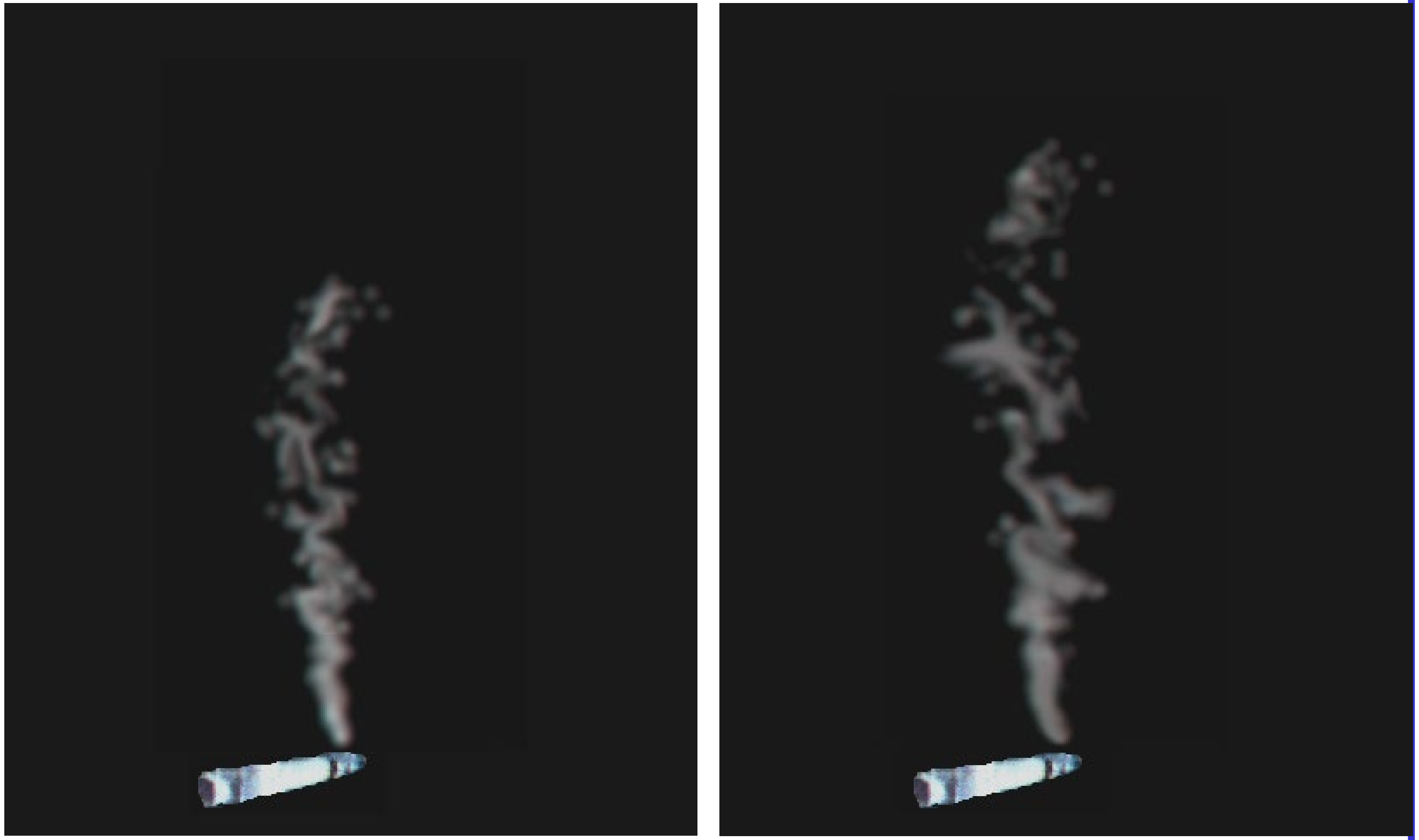
Attributes of a Particle

- Position
- Velocity
- Life Span
- Size
- Weight
- Representation
- Color
- Owner

Basic Particle System Physics

- Particle is a point in 3D space.
- Forces (e.g. gravity or wind) accelerate a particle.
- Acceleration changes velocity.
- Velocity changes position

Particle system



Thank You