CAD/CAM (21-342)

Advanced Manufacturing Laboratory
Department of Industrial Engineering
Sharif University of Technology

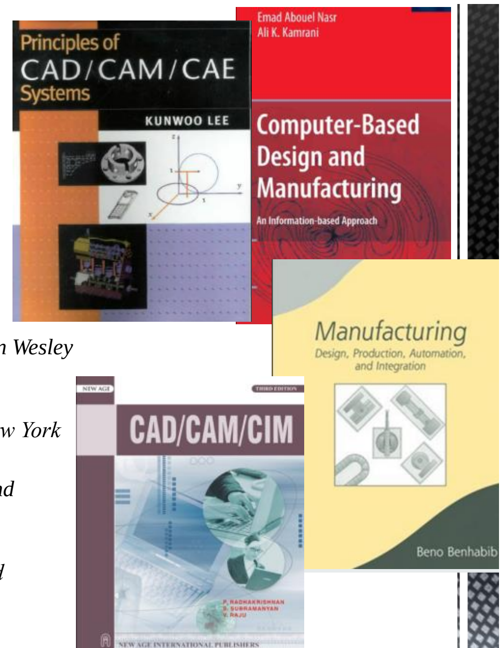
Session # 7

Course Description

- Instructor
  - Omid Fatahi Valilai, Ph.D. Industrial Engineering Department, Sharif University of Technology
  - Email: FValilai@sharif.edu , Tel: 6616-5706
  - Website: Sharif.edu/~fvalilai
- Class time
  - Saturday- Monday 10:30-12:00
- Course evaluation
  - Mid-term (25%)
  - Final exam (40%)
  - Quiz (5%)
  - Exercise (30%)

Course Description *(Continued ...)*

- *Mid-term session:*
  - *Monday: 8th Ordibehesht 1393, 10:30 ~ 12:30*
- *Final Exam:*
  - *Saturday: 24th Khordad 1393, 15:00 ~ 17:30*
- *Reference:*
  - *Lee, Kunwoo; "Principles of CAD/CAM/CAE systems", 1999, Addsion Wesley*
  - *Abouel Nasr, Emad; Kamrani, Ali K.; "Computer-Based Design and Manufacturing: An Information-Based Approach", 2007, Springer, New York*
  - *Benhabib, Beno; "Manufacturing: Design, Production, CAD/CAM, and Integration", 2003, Marcel Dekker Inc, New York*
  - *Radhakrishnan, P.; Subramanian, S.; Raju, V.; "CAD/CAM/CIM", 3rd edition, 2005, New age international (P) limited publishers, New York*



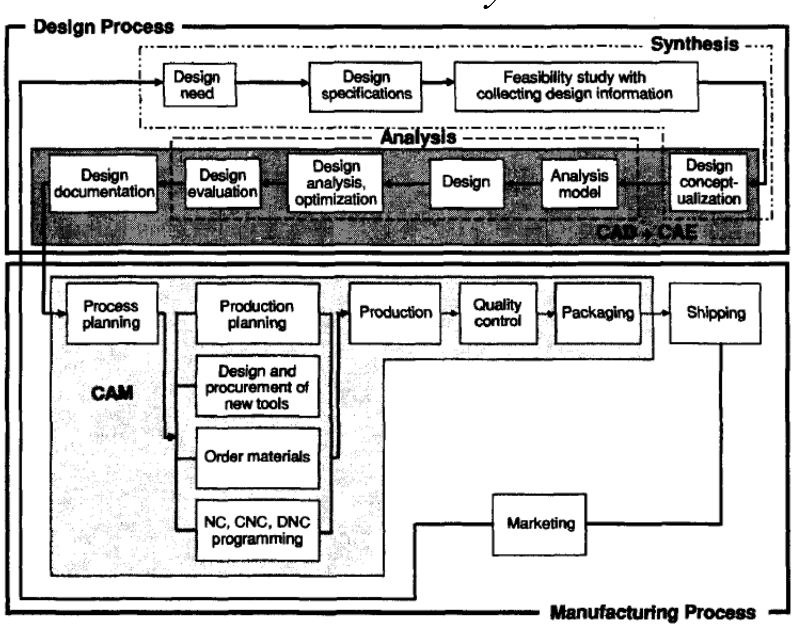
Course Description *(Continued..)*

- *Contents:*
  - *Introduction to CAD/CAM/CAE systems* *(5 sessions)*
  - *Components of CAD/CAM/CAE systems* *(2 sessions)*
  - *Geometric modeling systems* *(3 sessions)*
  - *Optimization in CAD* *(5 sessions)*
  - *Rapid prototyping and manufacturing* *(3 sessions)*
  - *Virtual engineering* *(2 sessions)*
  - *Product Life Cycle Cost Model* *(2 sessions)*
  - *Computer-Based Design and Features/Methodologies of Feature Representations* *(5 sessions)*
  - *Feature-Based Process Planning and Techniques* *(3 sessions)*
  - *Collaborative Engineering* *(2 sessions)*

Course Description *(Continued..)*

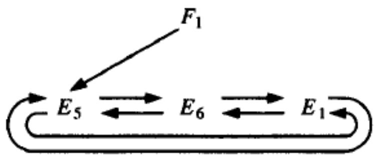
- *Contents:*
  - *Geometric modeling systems* *(3 sessions)*
    - *Wireframe modeling systems*
    - *Surface modeling systems*
    - *Solid modeling systems*
    - *Non-manifold modeling systems*
    - *Assembly modeling systems*

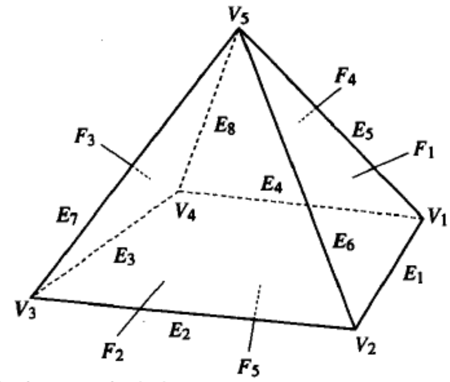
Introduction to CAD/CAM/CAE systems



Geometric modeling systems

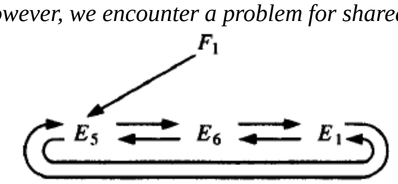
- *Data Structures*
  - *Half-Edge data structure*
    - *A remedy for variable size of face table, a list of edges for each face can be sored in a doubly linked list.*

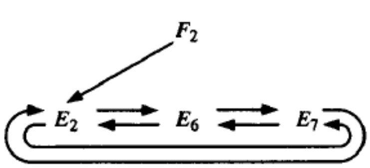


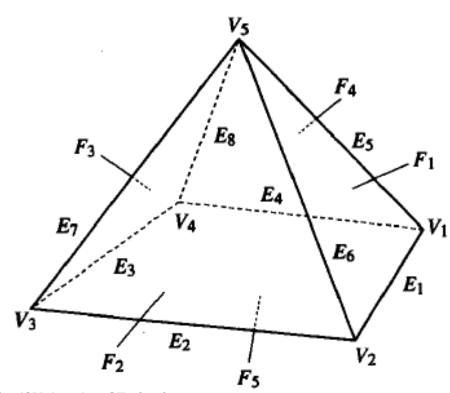


Geometric modeling systems

- *Data Structures*
  - *Half-Edge data structure*
    - *However, we encounter a problem for shared edges:*

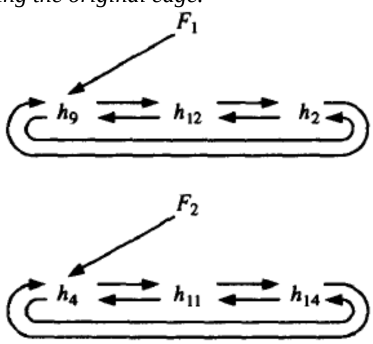


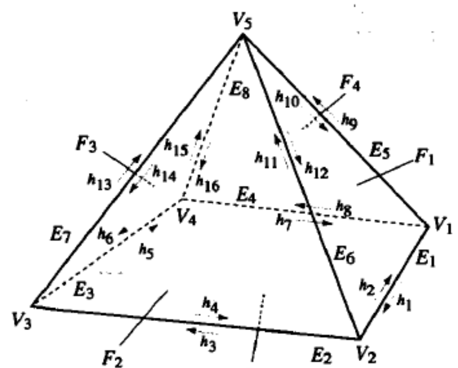




Geometric modeling systems

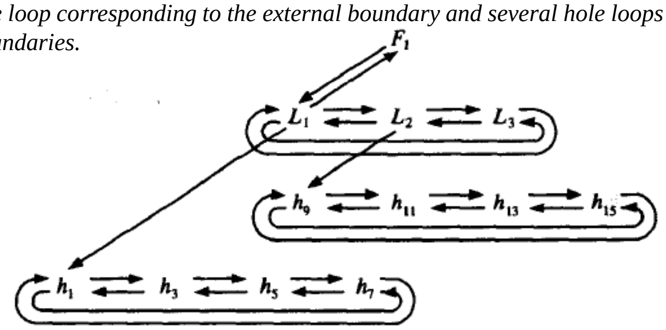
- *Data Structures*
  - *Half-Edge data structure*
    - *We can solve this problem by splitting each edge into halves and using them separately for two faces sharing the original edge.*





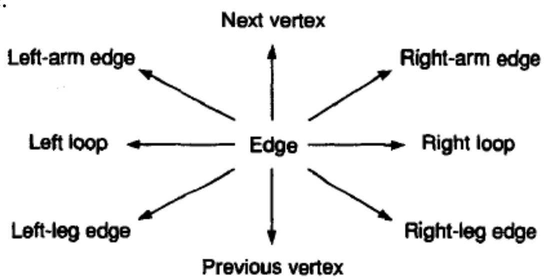
Geometric modeling systems

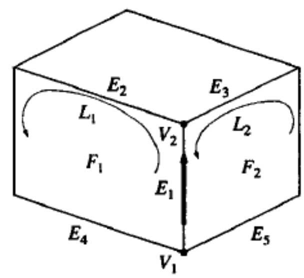
- *Data Structures*
  - *Half-Edge data structure*
    - *Loops can be used to take care of faces having inner holes without adding redundant bridge edges.*
    - *Any face is bounded by one loop corresponding to the external boundary and several hole loops corresponding internal boundaries.*



Geometric modeling systems

- *Data Structures*
  - *Winged-Edge data structure*
    - *In this data structures , the edges play the major role in contrast with the faces in half-edged data structures*
    - *Each edge stores the faces sharing the edge, the neighboring edges sharing the any of the vertices of the edge.*



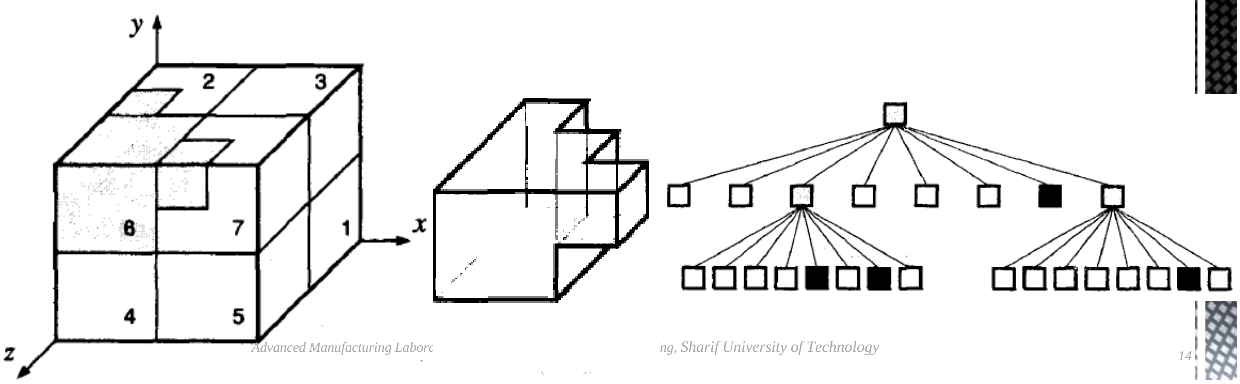


Geometric modeling systems

- *Data Structures*
  - *Decomposition model structure*
    - *A solid model can be described approximately as an aggregate of simple solids such as cubes.*
    - *Typical decomposition models and the data structures for storing them include:*
      - *Voxel representation*
      - *Octree representation*
      - *Cell decomposition*

Geometric modeling systems

- *Data Structures*
  - *Decomposition model structure*
    - *Octree representation*
    - *It represents a solid as an aggregate of hexahedra but it reduces the memory requirement considerably dividing the space differently.*
    - *Octants can be represented as the nodes of a tree , this tree is called an Octree.*



Geometric modeling systems

- *Data Structures*
  - *Decomposition model structure*
    - *Octree representation*

```
struct      octreeroot
{
            float               xmin, ymin, zmin;        /*space of interest*/
            float               xmax, ymax, zmax;
            struct    octree    *root                    /*root of the tree*/
};

struct      octree
{
            char                code;/*BLACK, WHITE, GREY*/
            struct    octree    *oct[8];    /*pointers to octants, present if GREY*/
};
```

Geometric modeling systems

- Data Structures
  - Decomposition model structure
    - Octree representation

```
make_tree( p, t, depth)
primitive   *p;       /* p = the primitive to be modeled */
octree      *t;       /* t = node of the octree, initially
                         the initial tree with one grey node */
int         depth;    /* initially max. depth of the recursion */
{
            int       i;
            switch( classify( p, t ) )
            {
                      case WHITE:
                            t–> code = WHITE;
                            break;
                      case BLACK:
                            t-> code = BLACK;
                            break;
                      case GREY:
                            if (depth == 0 )
                            {
                                   t–> code = BLACK;
                            }
                            else
                            {
                                   subdivide( t );
                                   for( i = 0; i < 8; i++ )
                                          make_tree( p, t–> oct[i], depth–1 );
                            }
                            break;
                      }
            }
}
```


QUIZ

```
struct operator {
      int     op_type,          /* union, intersection or difference operator */
              L_type;           /* left node type: 0=operator, 1=primitive */
              R_type            /* right node type: 0=operator, 1=primitive */
      void    *L_ptr;           /* left node */
              *R_ptr;           /* right node */
              *p_ptr;           /* parent node */
}

struct primitive {
      int     prim_type;                 /* type of primitive */
      double  pos_x, pos_y, pos_z;       /* position of instance */
      double  ori_x, ori_y, ori_z;       /* orientation of instance */
      void    *attribute;                /* the value of dimensions of the primitive */
}
```