

Finding point-pairs

- Given an \mathbf{a} , find a corresponding \mathbf{b} on the surface.
- Then one approach would be to search every possible triangle or surface point and then take the closest point.
- The key is to find a more efficient way to do this



Find Closest Point from Dense Cloud

- Basic approach is to divide space into regions. Suppose that we have one point \mathbf{b}_k^* that is a possible match for a point \mathbf{a}_k . The distance $\Delta^* = \|\mathbf{b}_k^* - \mathbf{a}_k\|$ obviously acts as an upper bound on the distance of the closest point to the surface.
- Given a region \mathbf{R} containing many possible points \mathbf{b}_j , if we can compute a lower bound Δ_L on the distance from \mathbf{a} to any point in \mathbf{R} , then we need only consider points inside \mathbf{R} if $\Delta_L < \Delta^*$.



Find Closest Point from Dense Cloud

- There are many ways to implement this idea
 - Simply partitioning space into many buckets
 - Octrees, KD trees, covariance trees, etc.



Approaches to closest triangle finding

1. (Simplest) Construct linear list of triangles and search sequentially for closest triangle to each point.
2. (Only slightly harder) Construct bounding spheres around each triangle and use these to reduce the number of careful checks required.
3. (Faster if have lots of points) Construct hierarchical data structure to speed search.
4. (Better but harder) Rotate each level of the tree to align with data.



FindClosestPoint(a,[p,q,r])

Many approaches. One is to solve the system

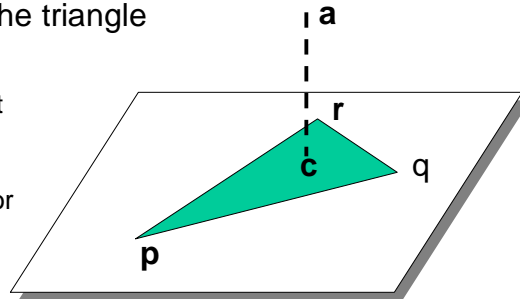
$$\mathbf{a} - \mathbf{p} \approx \lambda(\mathbf{q} - \mathbf{p}) + \mu(\mathbf{r} - \mathbf{p})$$

in a least squares sense for λ and μ . Then compute

$$\mathbf{c} = \mathbf{p} + \lambda(\mathbf{q} - \mathbf{p}) + \mu(\mathbf{r} - \mathbf{p})$$

If $\lambda \geq 0, \mu \geq 0, \lambda + \mu \leq 1$, then \mathbf{c} lies within the triangle and is the closest point. Otherwise, you need to find a point on the border of the triangle

Hint: For efficiency, work out the least squares problem explicitly. You will have to solve a 2 x 2 linear system for λ, μ

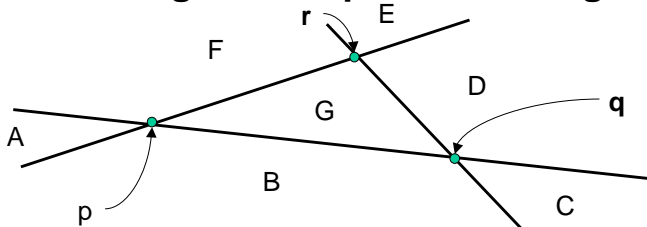


Copyright © CISST ERC, 2000

Engineering Research Center for Computer Integrated Surgical Systems and Technology



Finding closest point on triangle



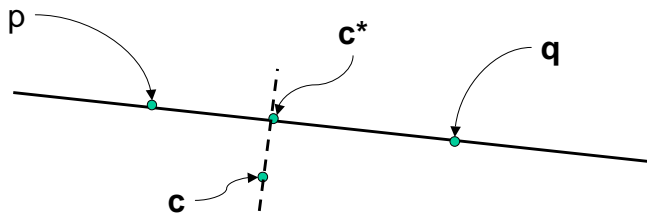
Region	$\lambda < 0$	$\mu < 0$	$\lambda + \mu > 1$	Closest point
A	Yes	Yes	No	p
B	No	Yes	No	<i>ProjectOnSegment(c,p,q)</i>
C	No	Yes	Yes	q
D	No	No	Yes	<i>ProjectOnSegment(c,q,r)</i>
E	Yes	No	Yes	r
F	Yes	No	No	<i>ProjectOnSegment(c,r,p)</i>
G	No	No	No	c

Copyright © CISST ERC, 2000

Engineering Research Center for Computer Integrated Surgical Systems and Technology



ProjectOnSegment(c,p,q)



$$\lambda = \frac{(c-p) \cdot (q-p)}{(q-p) \cdot (q-p)}$$

$$\lambda^* = \text{Max}(0, \text{Min}(\lambda, 1))$$

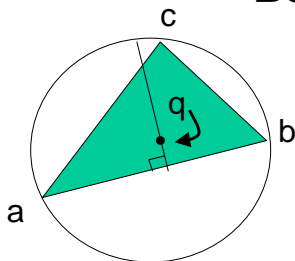
$$c^* = p + \lambda^* (q - p)$$

Copyright © CISST ERC, 2000

Engineering Research Center for Computer Integrated Surgical Systems and Technology



Bounding Sphere



Suppose you have a point \bar{p} and are trying to find the closest triangle $(\bar{a}_k, \bar{b}_k, \bar{c}_k)$ to \bar{p} . If you have already found a triangle $(\bar{a}_j, \bar{b}_j, \bar{c}_j)$ with a point \bar{r}_j on it, when do you need to check carefully for some triangle k ?

Answer: if \bar{q}_k is the center of a sphere of radius ρ_k enclosing $(\bar{a}_k, \bar{b}_k, \bar{c}_k)$, then you only need to check carefully if

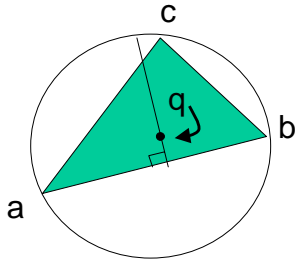
$$\|\bar{p} - \bar{q}_k\| - \rho_k < \|\bar{p} - \bar{r}_j\|.$$

Copyright © CISST ERC, 2000

Engineering Research Center for Computer Integrated Surgical Systems and Technology



Bounding Sphere



Assume edge (\vec{a}, \vec{b}) is the longest.

Then the center \vec{q} of the sphere will obey

$$(\vec{b} - \vec{q}) \cdot (\vec{b} - \vec{q}) = (\vec{a} - \vec{q}) \cdot (\vec{a} - \vec{q})$$

$$(\vec{c} - \vec{q}) \cdot (\vec{c} - \vec{q}) \leq (\vec{a} - \vec{q}) \cdot (\vec{a} - \vec{q})$$

$$(\vec{b} - \vec{a}) \times (\vec{c} - \vec{a}) \cdot (\vec{q} - \vec{a}) = 0$$

Simple approach: Try $\vec{q} = (\vec{a} + \vec{b}) / 2$.

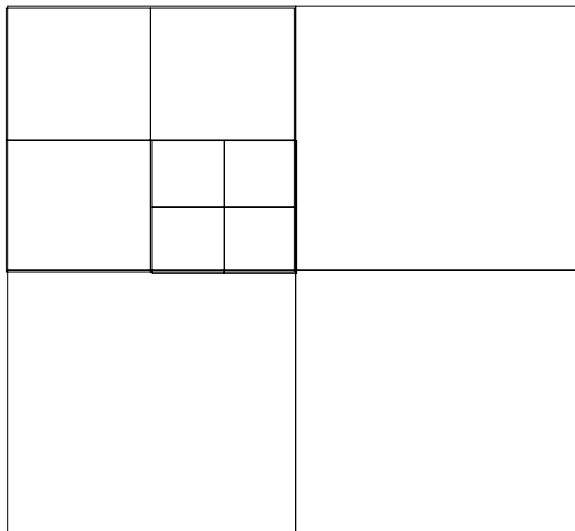
If inequality holds, then done.

Else solve the system as three equalities to get \vec{q} .

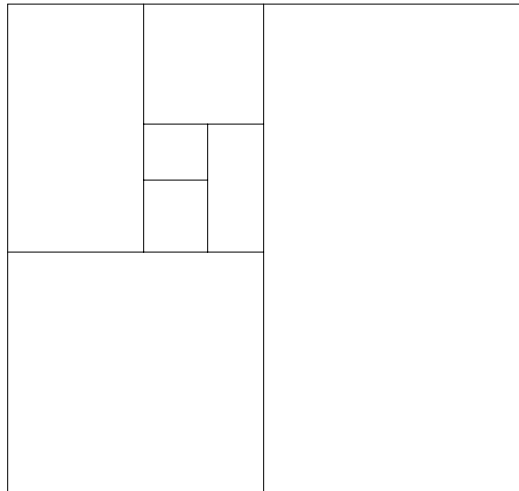
The radius $\rho = \|\vec{q} - \vec{a}\|$.



Hierarchical cellular decompositions



Hierarchical cellular decompositions



Copyright © CISST ERC, 2000

Engineering Research Center for Computer Integrated Surgical Systems and Technology



Constructing tree of bounding spheres

```
class BoundingSphere {  
    public:  
        Vec3 Center;           // Coordinates of center  
        double Radius;        // radius of sphere  
        Thing* Object;        // some reference to the thing  
                               // bounded  
};
```

Copyright © CISST ERC, 2000

Engineering Research Center for Computer Integrated Surgical Systems and Technology



Constructing octree of bounding spheres

```
class BoundingBoxTreeNode {  
    Vec3 Center;          // splitting point  
    Vec3 UB;             // corners of box  
    Vec3 LB;  
    int HaveSubtrees;  
    int nSpheres;  
    double MaxRadius;    // maximum radius of sphere in box  
    BoundingBoxTreeNode* SubTrees[2][2][2];  
    BoundingSphere** Spheres;  
    :  
    :  
    BoundingBoxTreeNode(BoundingSphere** BS, int nS);  
    ConstructSubtrees();  
    void FindClosestPoint(Vec3 v, double& bound, Vec3& closest);  
};
```



Constructing octree of bounding spheres

```
BoundingBoxTreeNode(BoundingSphere** BS, int nS)  
{ Spheres = BS; nSpheres = nS;  
  Center = Centroid(Spheres, nSpheres);  
  MaxRadius = FindMaxRadius(Spheres, nSpheres);  
  UB = FindMaxCoordinates(Spheres, nSpheres);  
  LB = FindMinCoordinates(Spheres, nSpheres);  
  ConstructSubtrees();  
};
```



Constructing octree of bounding spheres

```
ConstructSubtrees()
{ if (nSpheres<= minCount || length(UB-LB)<=minDiag)
  { HaveSubtrees=0; return; };
  HaveSubtrees = 1;
  int nnn, npn, npp, nnp, pnn, ppn, ppp, pnp;
  // number of spheres in each subtree
  SplitSort(Center, Spheres, nnn, npn, npp, nnp, pnn, ppn, ppp, pnp);
  Subtrees[0][0][0] = BoundingBoxTree(Spheres[0], nnn);
  Subtrees[0][1][0] = BoundingBoxTree(Spheres[nnn], npn);
  Subtrees[0][1][1] = BoundingBoxTree(Spheres[nnn+npn], npp);
  :
  :
}
```

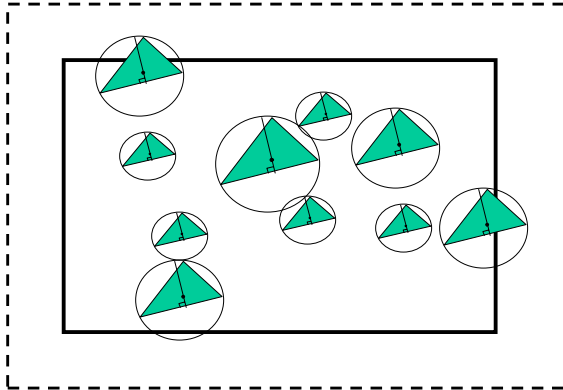


Constructing octree of bounding spheres

```
SplitSort(Vec3 SplittingPoint, BoundingSphere** Spheres,
          int& nnn, int& npn, ... ,int& pnp)
{ // reorder Spheres(...) into eight buckets according to
  // comparison of coordinates of Sphere(k)->Center
  // with coordinates of splitting point. E.g., first bucket has
  // Sphere(k)->Center.x < SplittingPoint.x
  // Sphere(k)->Center.y < SplittingPoint.y
  // Sphere(k)->Center.z < SplittingPoint.z
  // This can be done "in place" by suitable exchanges.
  // Set nnn = number of spheres with all coordinates less than
  // splitting point, etc.
}
```



Searching an octree of bounding spheres



Copyright © CISST ERC, 2000

Engineering Research Center for Computer Integrated Surgical Systems and Technology



Searching an octree of bounding spheres

```

void BoundingBoxTreeNode::FindClosestPoint
    (Vec3 v, double& bound, Vec3& closest)
{ double dist = bound + MaxRadius;
  if (v.x > UB.x+dist) return; if (v.y > UB.y+dist) return;
  .... ; if (v.z < LB.z-dist) return;
  if (HaveSubtrees)
    { Subtrees[0][0][0].FindClosestPoint(v,bound,closest);
      :
      Subtrees[1][1][1].FindClosestPoint(v,bound,closest);
    }
  else
    for (int i=0;i<nSpheres;i++)
      UpdateClosest(Spheres[i],v,bound,closest);
};
    
```

Copyright © CISST ERC, 2000

Engineering Research Center for Computer Integrated Surgical Systems and Technology



Searching an octree of bounding spheres

```
void UpdateClosest(BoundingSphere* S,  
                  Vec3 v, double& bound, Vec3& closest)  
{ double dist = v-S->Center;;  
  if (dist - S->Radius > bound) return;  
  Vec3 cp = ClosestPointTo(*S->Object,v);  
  dist = LengthOf(cp-v);  
  if (dist<bound) { bound = dist; closest=cp;};  
};
```

