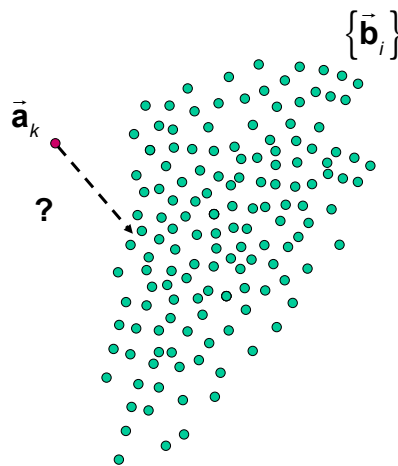


Finding point-pairs

- Given an \mathbf{a} , find a corresponding \mathbf{b} on the surface.
- 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



Suppose surface is represented by dense cloud of points



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 $\Delta_L < \Delta^*$.

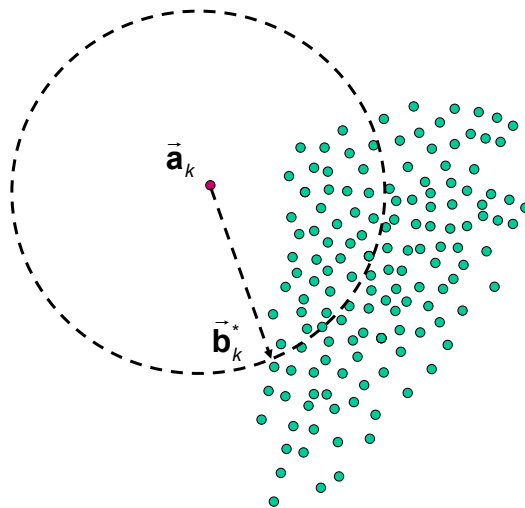
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



3

Given a match, is there anything closer?



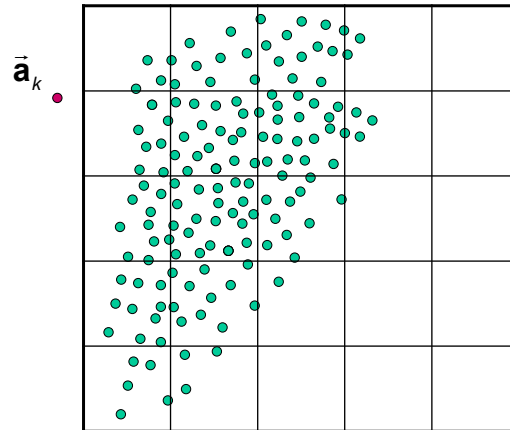
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



4

Divide cloud into cells



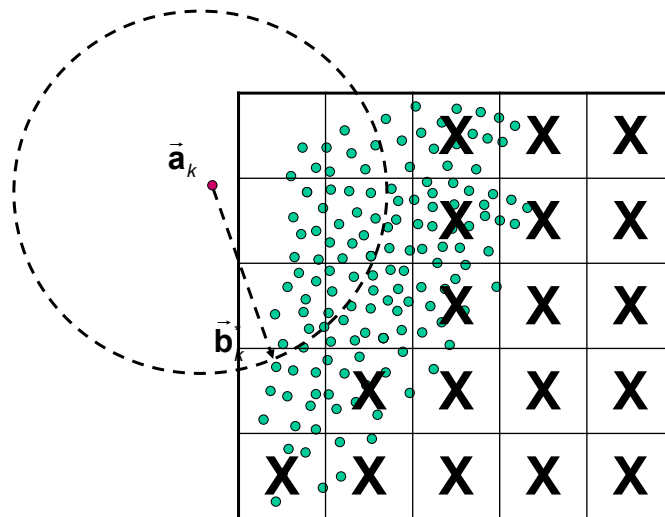
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



5

Can exclude everything outside circle



Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



6

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.

Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



7

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 or bounding boxes 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.

Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



8

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

Many approaches. One is to solve the system

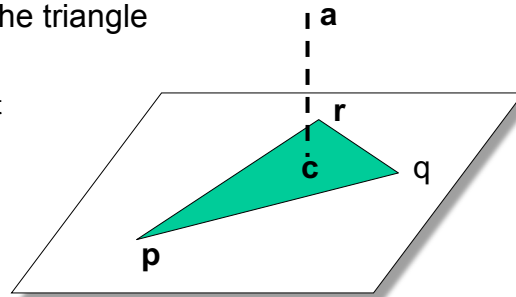
$$\mathbf{a} - \mathbf{p} \approx \lambda(\mathbf{q} - \mathbf{p}) + \mu(\mathbf{r} - \mathbf{p})$$

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 for λ, μ



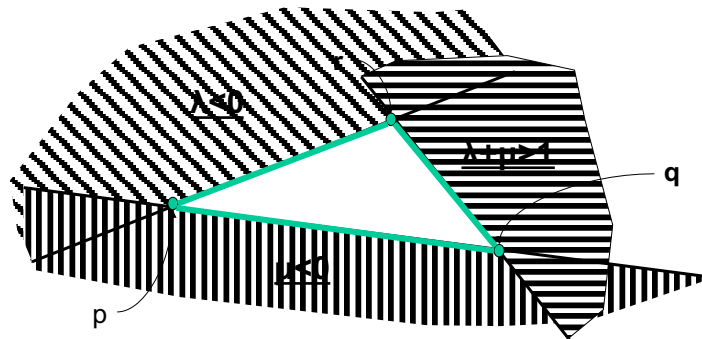
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



9

Finding closest point on triangle



Region	Closest point
$\lambda < 0$	$ProjectOnSegment(\mathbf{c}, \mathbf{r}, \mathbf{p})$
$\mu < 0$	$ProjectOnSegment(\mathbf{c}, \mathbf{p}, \mathbf{q})$
$\lambda + \mu > 1$	$ProjectOnSegment(\mathbf{c}, \mathbf{q}, \mathbf{r})$

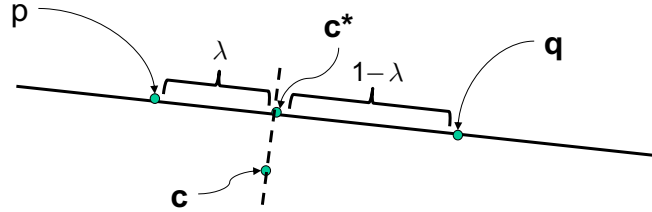
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



10

ProjectOnSegment(c,p,q)



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

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

$$\mathbf{c}^* = \mathbf{p} + \lambda^{(seg)} \times (\mathbf{q} - \mathbf{p})$$

Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



11

Simple Search with Bounding Boxes

// Triangle i has corners $[\vec{p}_i, \vec{q}_i, \vec{r}_i]$

// Bounding box lower = $\vec{L}_i = [L_{xi}, L_{yi}, L_{zi}]^T$; upper = $\vec{U}_i = [U_{xi}, U_{yi}, U_{zi}]^T$

$bound = \infty$

for $i = 1$ to N do

{ if $(L_{xi} - bound \leq a_x \leq U_{xi} + bound)$ and $(L_{yi} - bound \leq a_y \leq U_{yi} + bound)$
and $(L_{zi} - bound \leq a_z \leq U_{zi} + bound)$ then

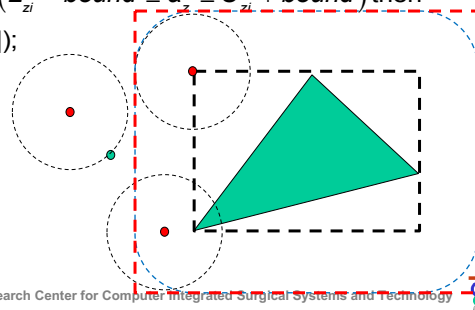
{ $\vec{h} = \text{FindClosestPoint}(\vec{a}, [\vec{p}_i, \vec{q}_i, \vec{r}_i])$;

if $\|\vec{h} - \vec{a}\| < bound$ then

{ $\vec{c} = \vec{h}$; $bound = \|\vec{h} - \vec{a}\|$ };

};

};



Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



13

Simple Search with Bounding Spheres

```
// Triangle  $i$  has corners  $[\vec{p}_i, \vec{q}_i, \vec{r}_i]$ 
// Surrounding sphere  $i$  has radius  $\rho_i$  center  $\vec{q}_i$ 
bound =  $\infty$ ;
for  $i=1$  to  $N$  do
{ if  $\|\vec{q}_i - \vec{a}\| - \rho_i \leq bound$  then
  {  $\vec{h} = \text{FindClosestPoint}(\vec{a}, [\vec{p}_i, \vec{q}_i, \vec{r}_i])$ ;
    if  $\|\vec{h} - \vec{a}\| < bound$  then
      {  $\vec{c} = \vec{h}$ ;  $bound = \|\vec{h} - \vec{a}\|$ ; };
  };
};
```

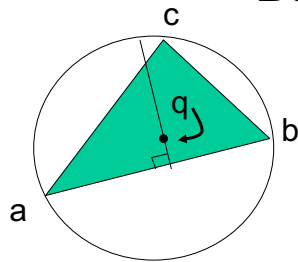
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



14

Bounding Sphere



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

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

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

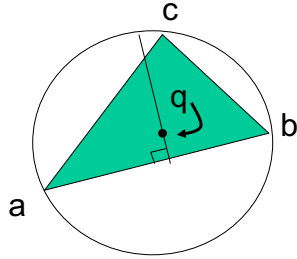
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



15

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 to get \vec{q} (next page).

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



Bounding Sphere

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

Compute $\vec{f} = (\vec{a} + \vec{b}) / 2$.

Define

$$\vec{u} = \vec{a} - \vec{f}; \vec{v} = \vec{c} - \vec{f}$$

$$\vec{d} = (\vec{u} \times \vec{v}) \times \vec{u}$$

Then the sphere center \vec{q} lies somewhere along the line

$$\vec{q} = \vec{f} + \lambda \vec{d}$$

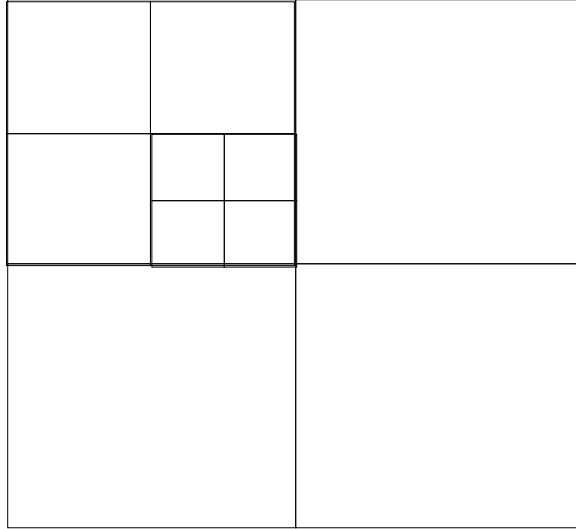
with $(\lambda \vec{d} - \vec{v})^2 \leq (\lambda \vec{d} - \vec{u})^2$. Simplifying gives us

$$\lambda \geq \frac{\vec{v}^2 - \vec{u}^2}{2\vec{d} \cdot (\vec{v} - \vec{u})} = \gamma$$

If $\gamma \leq 0$, then just pick $\lambda=0$. Else pick $\lambda=\gamma$.



Hierarchical cellular decompositions



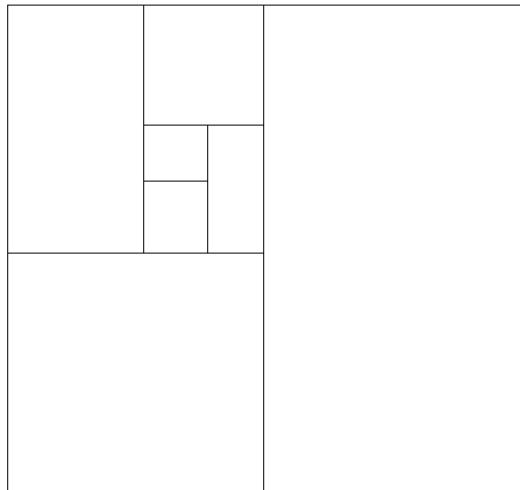
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



18

Hierarchical cellular decompositions



Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



19

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 Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



20

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);  
};
```

Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



21

Constructing octree of bounding spheres

```

BoundingBoxTreeNode(BoundingSphere** BS, int nS)
{
    Spheres = BS; nSpheres = nS;
    Center = Centroid(Spheres, nSpheres);
    // This will be the splitting point
    // Centroid is efficient to compute
    // But other choices are possible
    MaxRadius = FindMaxRadius(Spheres, nSpheres);
    UB = FindMaxCoordinates(Spheres, nSpheres);
    LB = FindMinCoordinates(Spheres, nSpheres);
    ConstructSubtrees();
};

```

Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



22

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);
    :
    :
}

```

Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



23

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.
}

```

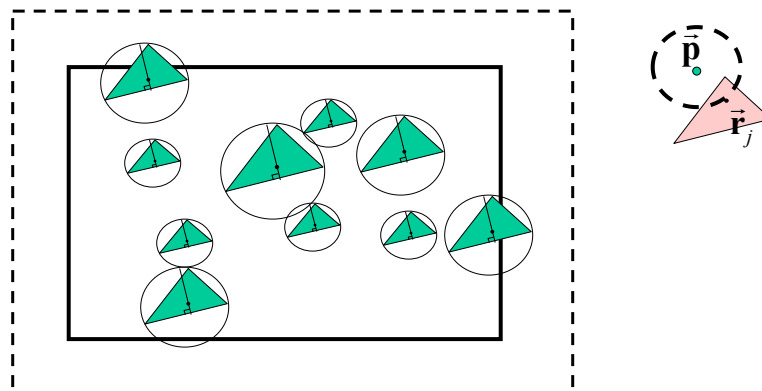
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



24

Searching an octree of bounding spheres



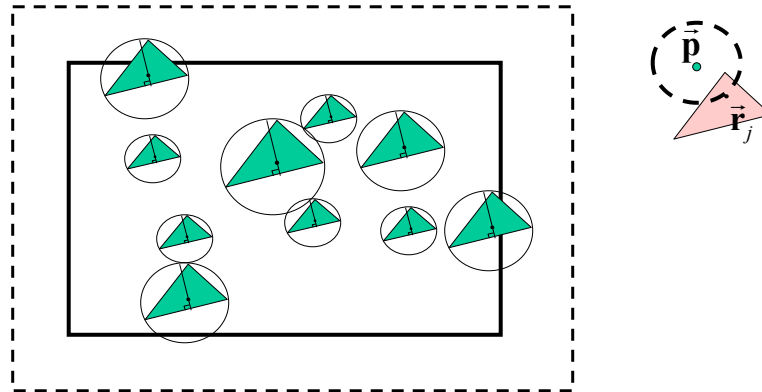
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



25

Searching an octree of bounding spheres



Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



26

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 Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



27

Searching an octree of bounding spheres

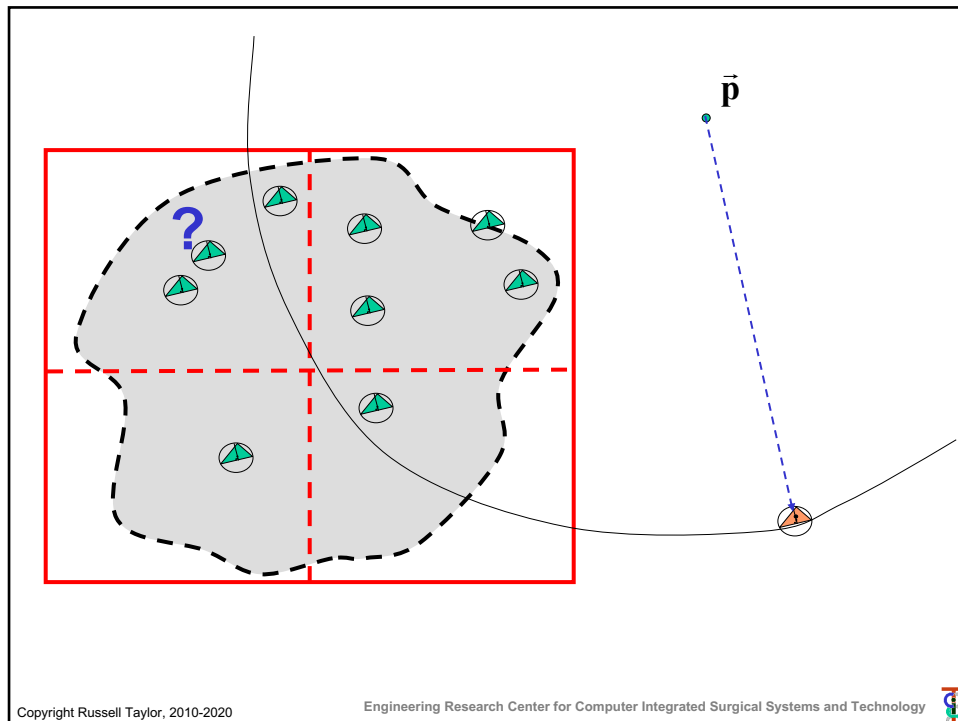
```
void UpdateClosest(BoundingSphere* S,  
                  Vec3 v, double& bound, Vec3& closest)  
{ double dist = v-S->Center;;  
  if (dist - S->Radius > bound) return;  
  Vec3 cp = ClosestPointTo(*S->Object,v);  
  dist = LengthOf(cp-v);  
  if (dist<bound) { bound = dist; closest=cp;};  
};
```

Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



28

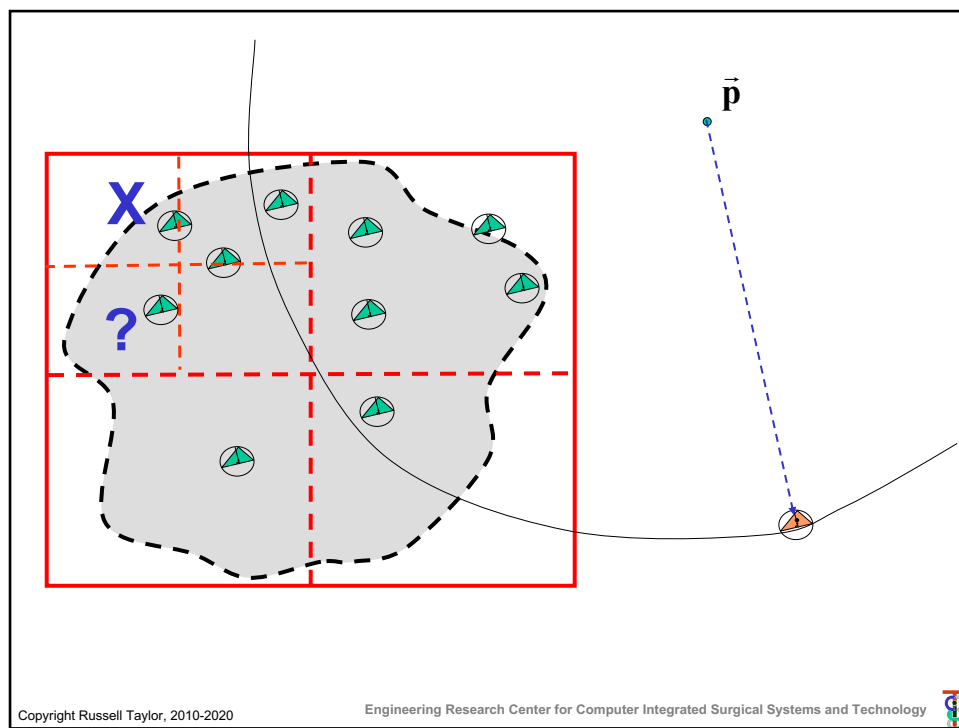


Copyright Russell Taylor, 2010-2020

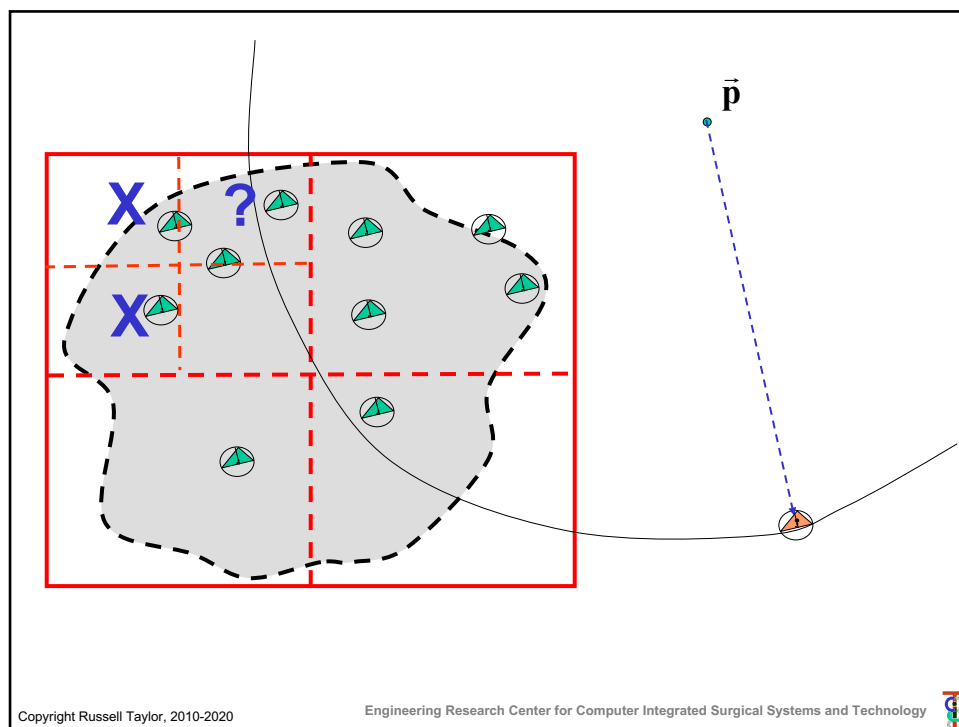
Engineering Research Center for Computer Integrated Surgical Systems and Technology



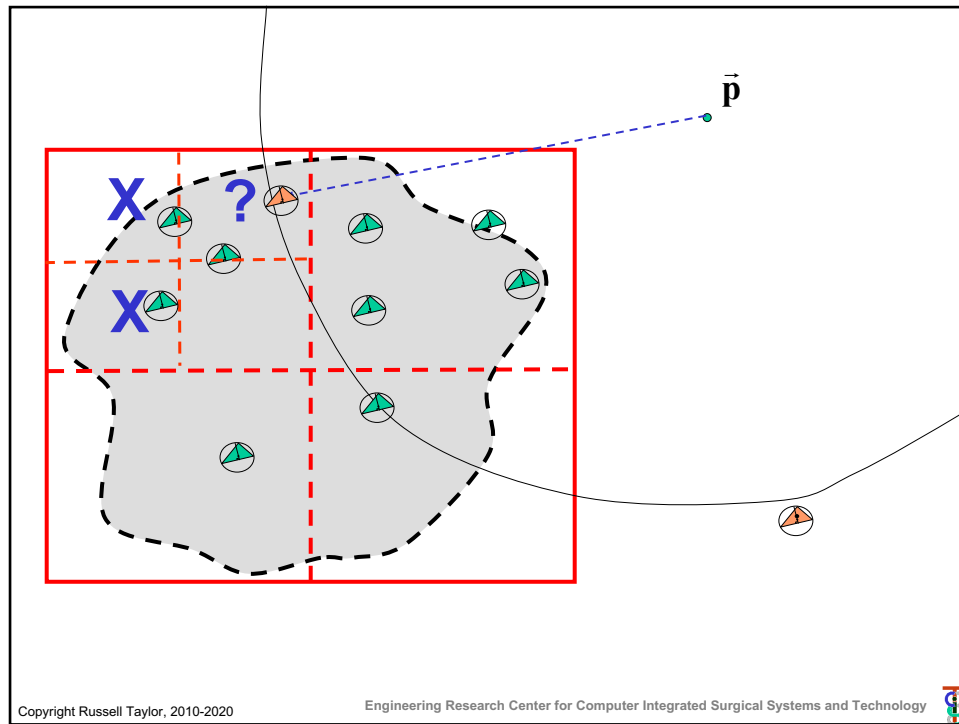
29



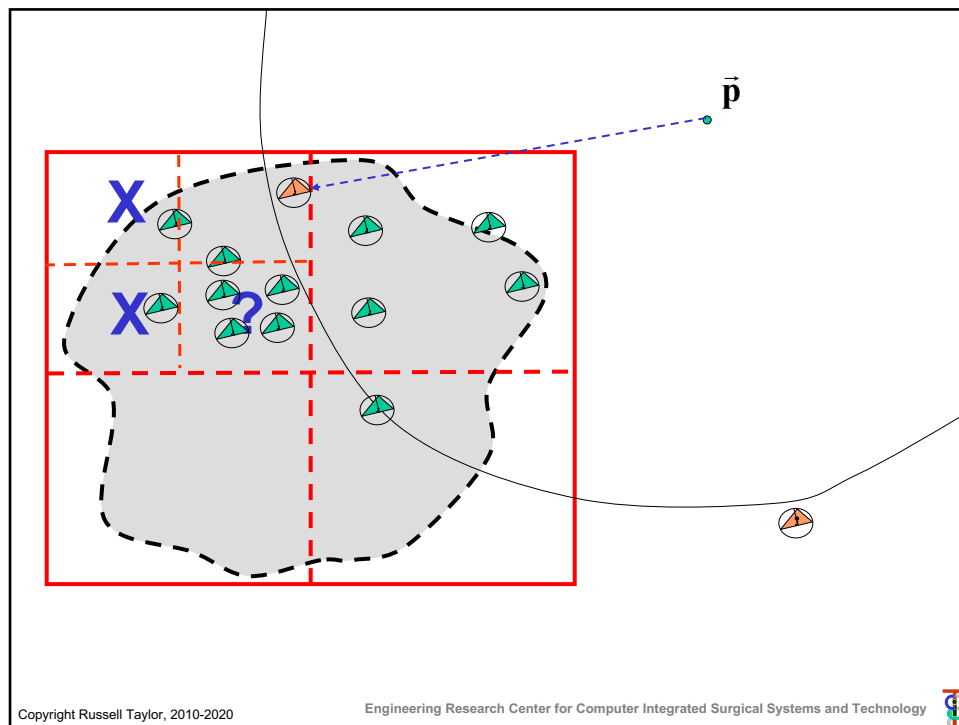
30



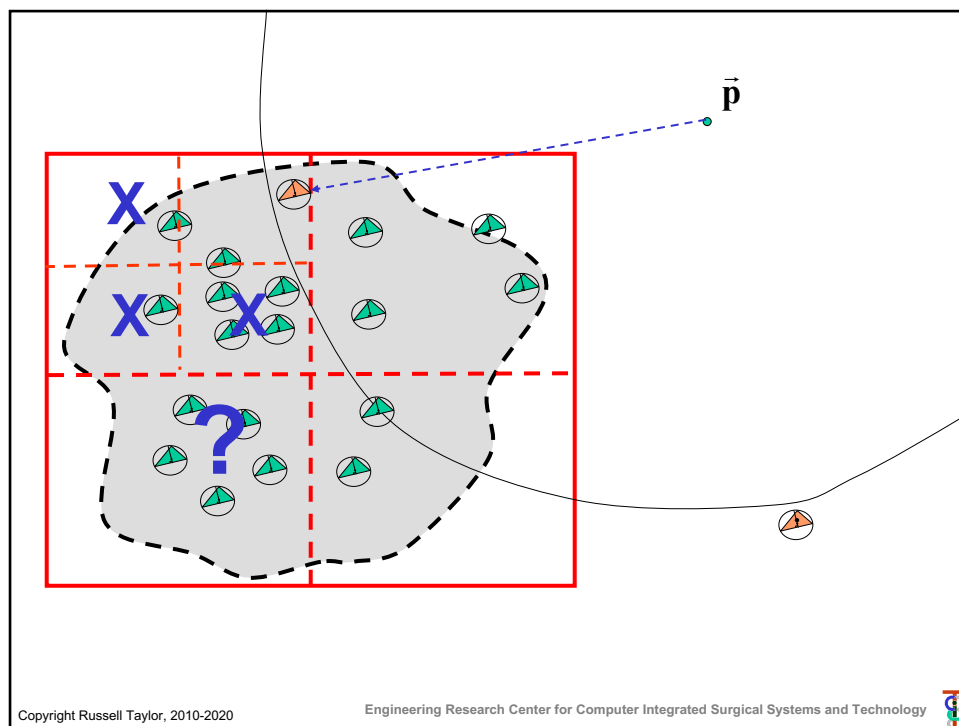
31



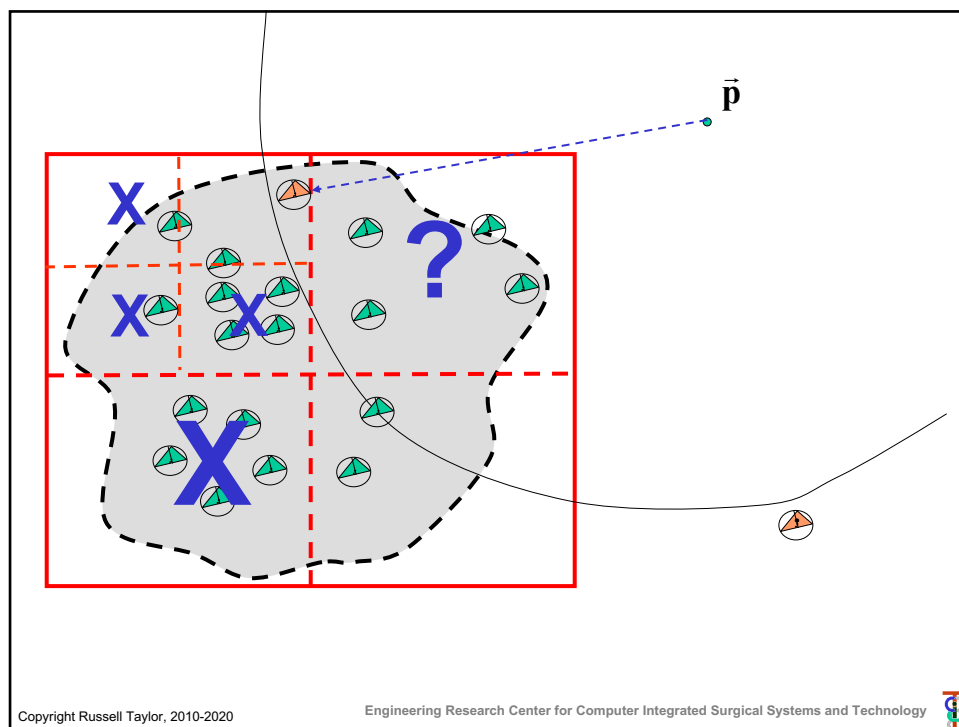
32



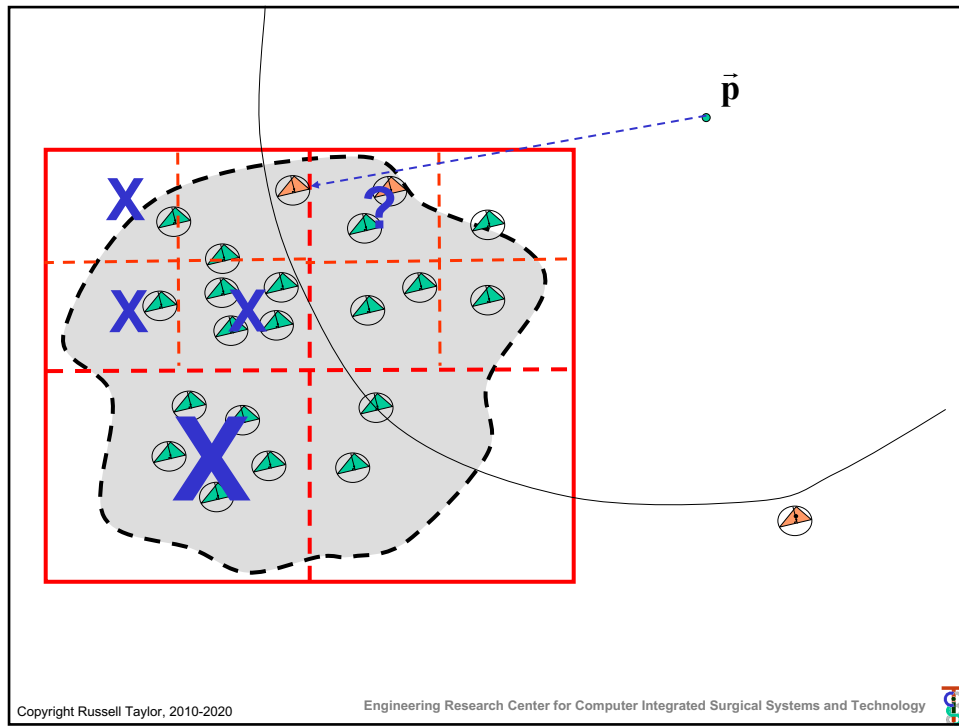
33



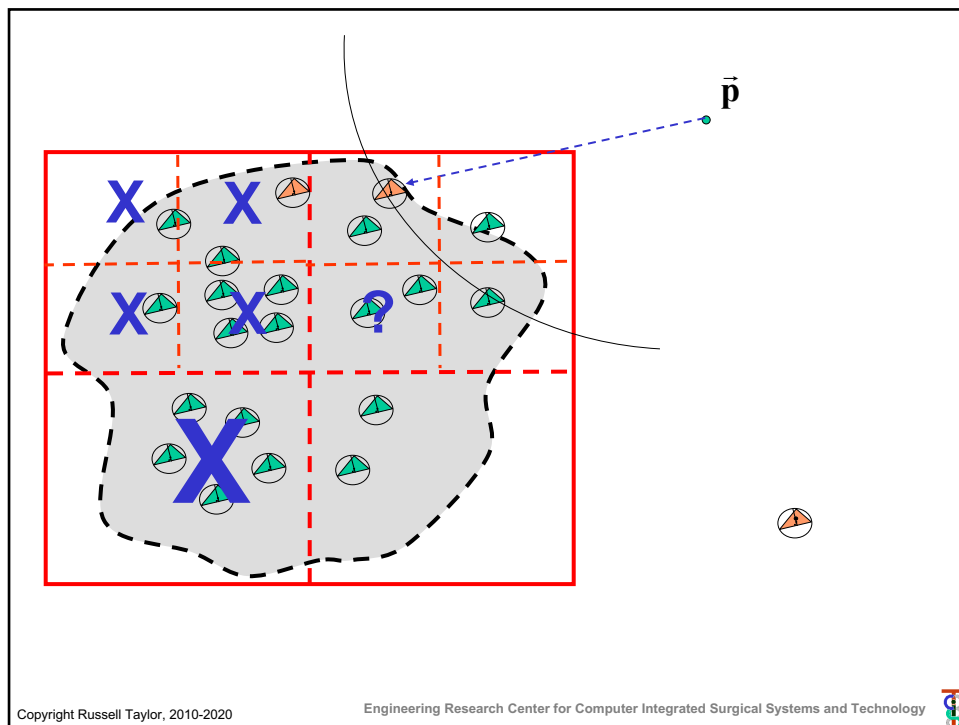
34



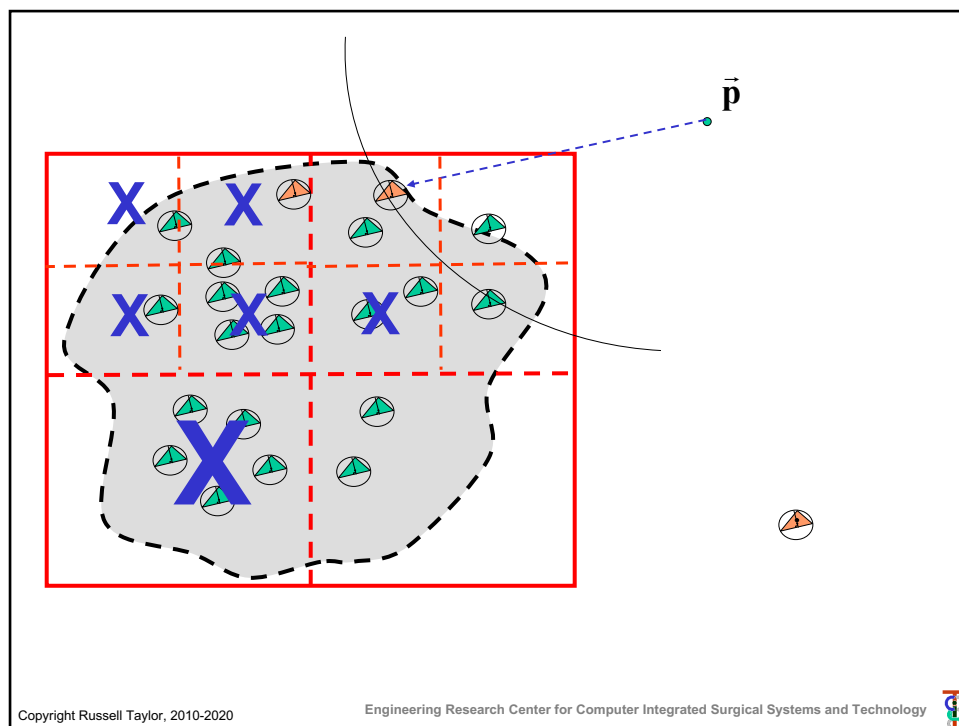
35



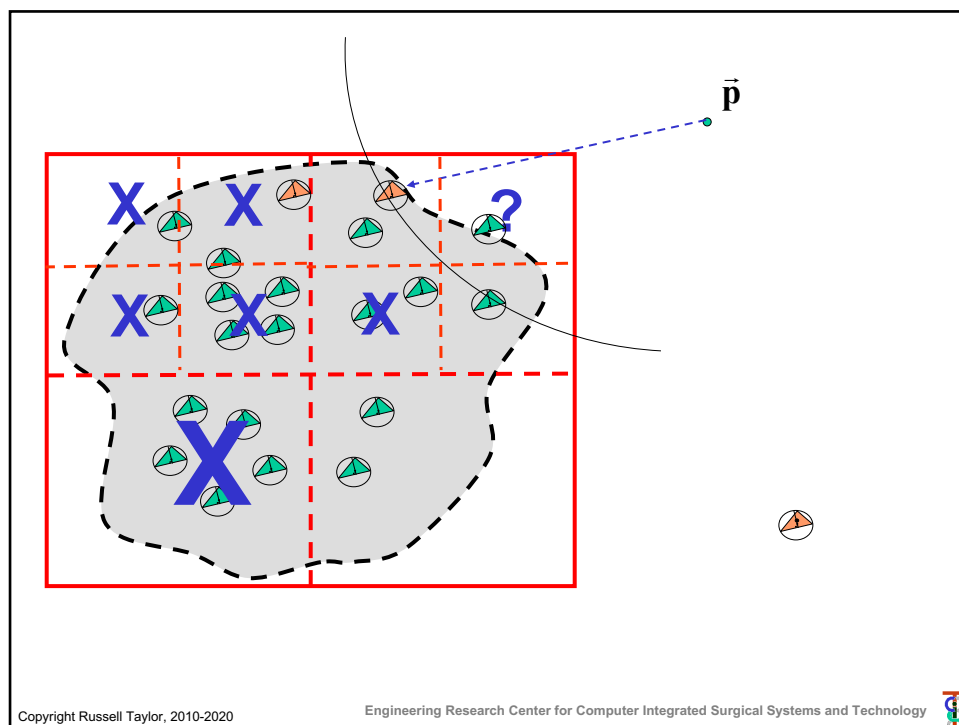
36



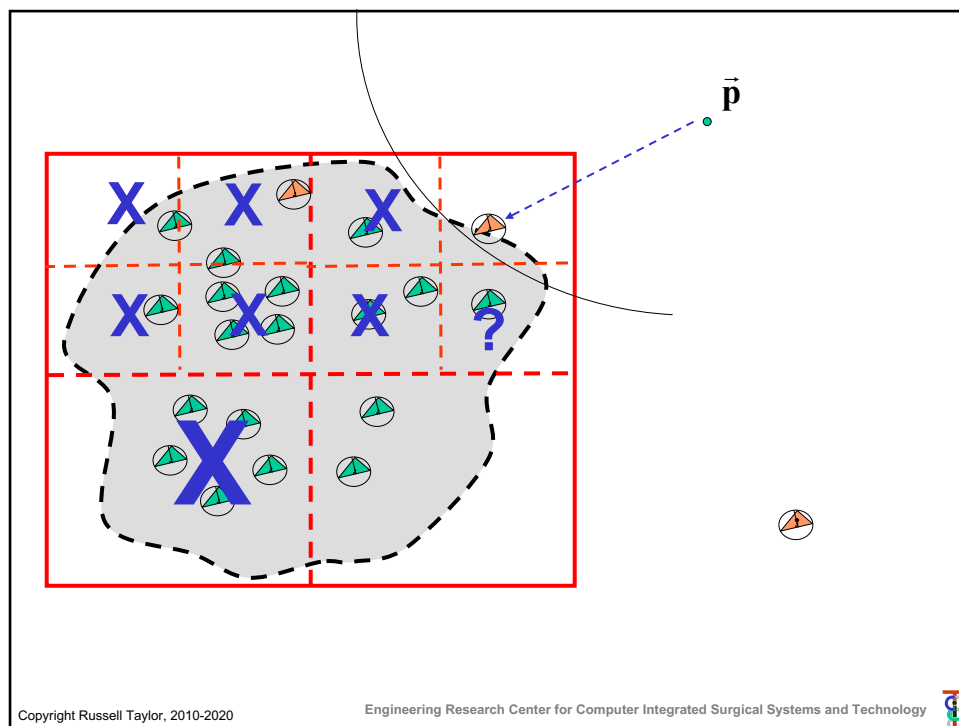
37



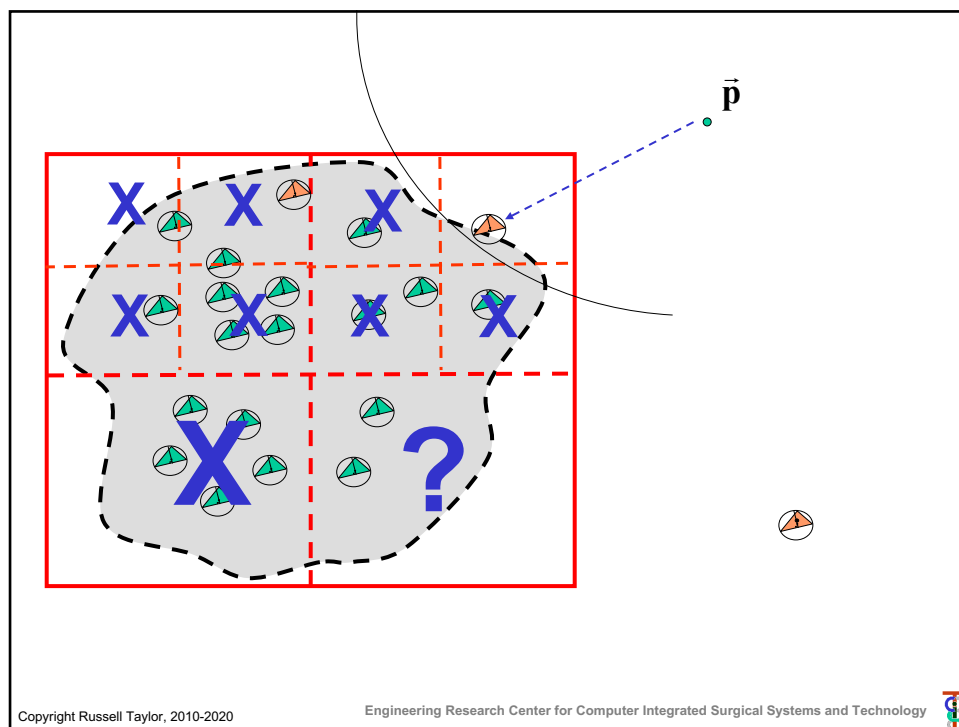
38



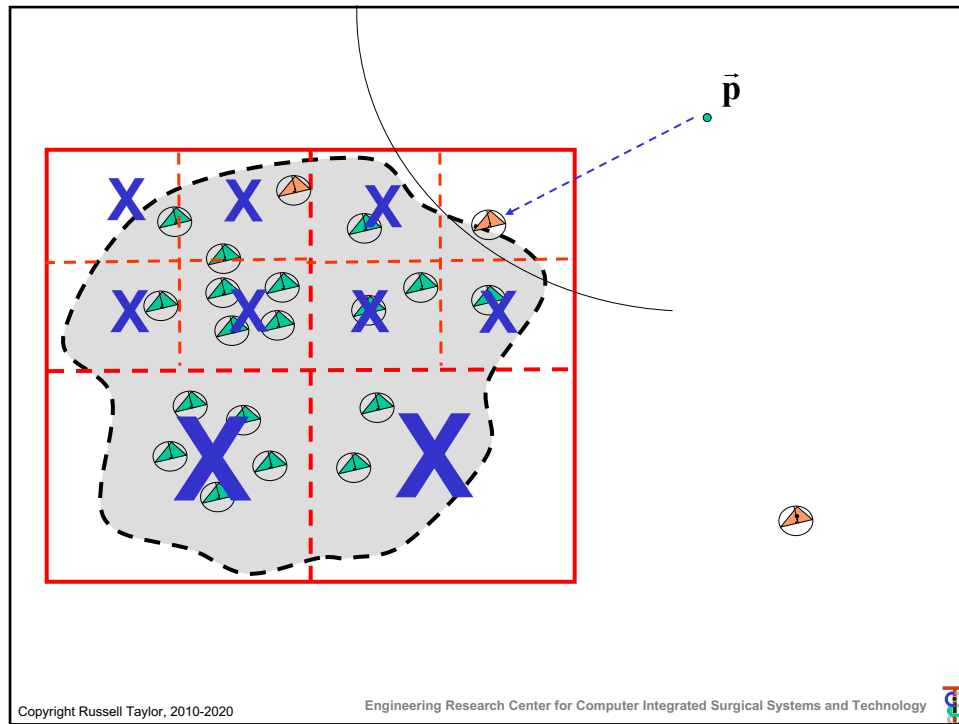
39



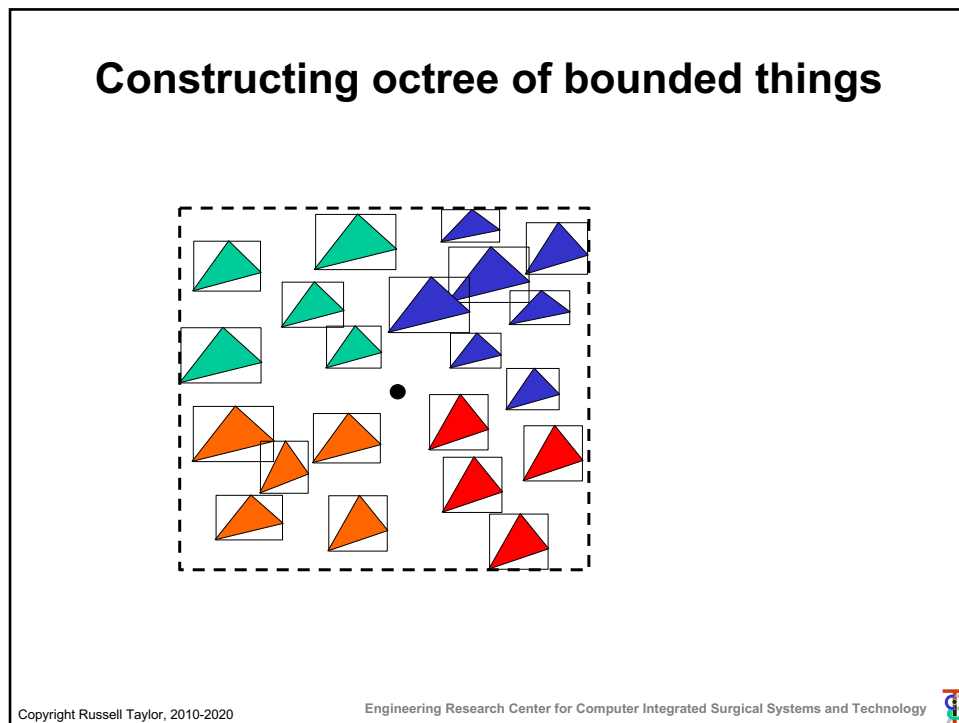
40



41

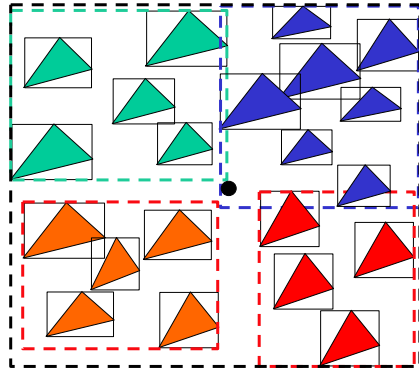


42



43

Constructing octree of bounded things



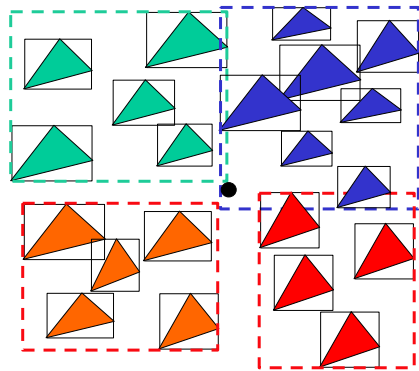
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



44

Constructing octree of bounded things



Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



45

Constructing octree of bounded things

```
class BoundingBoxTreeNode {
    Vec3 Center;          // splitting point
    Vec3 UB;              // corners of box
    Vec3 LB;
    int HaveSubtrees;
    int nThings;
    BoundingBoxTreeNode* SubTrees[2][2][2];
    Thing** Things;
    :
    :
    BoundingBoxTreeNode(Thing** BS, int nS);
    ConstructSubtrees();
    void FindClosestPoint(Vec3 v, double& bound, Vec3& closest);
};
```

Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



46

Properties of “Things”

```
Class Thing
{ public:
    :
    vec3 SortPoint();
        // returns a point that can be used to sort the object
    vec3 ClosestPointTo(vec3 p);
        // returns point in this thing closest to p
    [vec3,vec3] EnlargeBounds(frame F,vec3 LB, vec3 UB);
        // Given frame F, and corners LB and UB of bounding box
        // around some other things, returns a the corners of a bounding
        // box that includes this Thing2 as well,
        // where Thing2=F.Inverse()*this thing
    [vec3,vec3] BoundingBox(F);
        { return EnlargeBounds(F,[∞, ∞, ∞],[-∞,-∞,-∞]);};

    int MayBeInBounds(Frame F, vec3 LB, vec3 UB);
        // returns 1 if any part of this F.Inverse()*this thing could be
        // in the bounding box with corners LB and UB
}
```

Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



47

Triangle Things

```

Class Triangle : public Thing
{vec3 Corners[3]; // vertices of triangle
:
vec3 SortPoint() { return Mean(Corners); } // or use Corner[0]
[vec3,vec3] EnlargeBounds(frame F,vec3 LB, vec3 UB)
{ vec3 FiC[3]=F.inverse()*Corners;
  for (int l=0;l<3;l++)
    { LB.x = min(LB.x,FiC[l].x); UB.x = max(UB.x,FiC[l].x);
      LB.y = min(LB.y,FiC[l].y); UB.y = max(UB.y,FiC[l].y);
      LB.z = min(LB.z,FiC[l].z); UB.z = max(UB.z,FiC[l].z);
    };
  return [LB, UB];
};
[vec3,vec3] BoundingBox(F)
{ return EnlargeBounds(F,[ $\infty$ ,  $\infty$ ,  $\infty$ ],[ $-\infty$ , $-\infty$ , $-\infty$ ]);};
int MayBeInBounds(Frame F, vec3 LB, vec3 UB)
{ vec3 FiC[3]=F.inverse()*Corners;
  for (int k=0;k<3; k++) if (InBounds(FiC[k],LB,UB)) return 1;
  return 0;}
}

```

Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



48

Constructing octree of bounded things

```

BoundingBoxTreeNode(Thing** BS, int nS)
{ Things = BS; nThings = nS;
  UB = FindMaxCoordinates(Things,nThings);
  LB = FindMinCoordinates(Things,nThings);
  Center = LB+(UB-LB)/2.0; // Splitting point
                          // Not necessarily the best
                          // Alternatives would be centroid or
                          // the median of the SortPoint()'s.

  ConstructSubtrees();
};

```

Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



49

Constructing octree of bounded things

```

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

```

Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



50

Constructing octree of bounded things

```

SplitSort(Vec3& SplittingPoint, BoundingThing** Things,
          int& nnn, int& npn, ... ,int& pnp)
{ // reorder Spheres(...) into eight buckets according to
  // comparison of coordinates of Thing(k)->SortPoint()
  // with coordinates of splitting point. E.g., first bucket has
  //   Thing(k)->Center.x < SplittingPoint.x
  //   Thing(k)->Center.y < SplittingPoint.y
  //   Thing(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.

  // If desired, may be modified to simultaneously find a good
  // value for SplittingPoint (e.g., median)
}

```

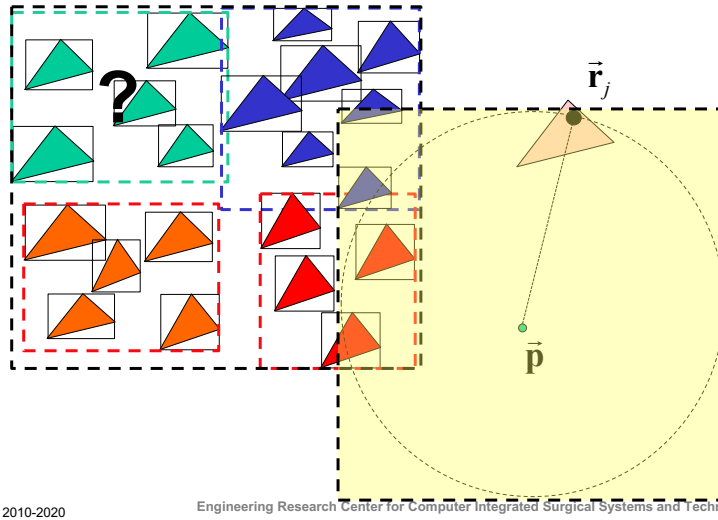
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



51

Searching octree of bounded things



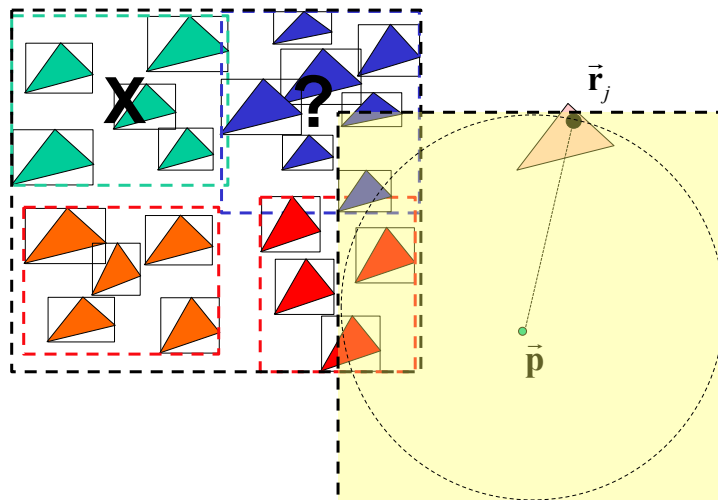
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



52

Searching octree of bounded things



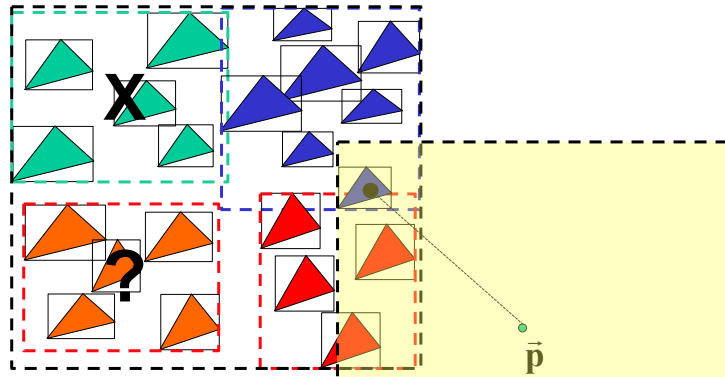
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



53

Searching octree of bounded things



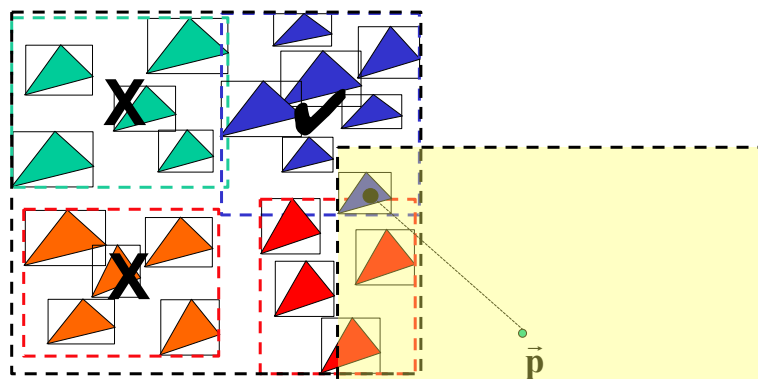
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



54

Searching octree of bounded things



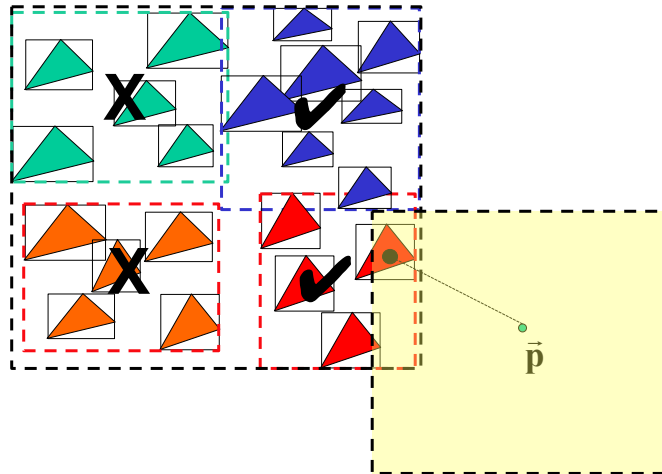
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



55

Searching octree of bounded things



Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



56

Searching octree of bounded things

```
void BoundingBoxTreeNode::FindClosestPoint
    (Vec3 v, double& bound, Vec3& closest)
{
    if ((v.x > UB.x+bound) || (v.x<LB.x-bound)) return;
    if ((v.y > UB.y+bound) || (v.y<LB.y-bound)) return;
    if ((v.z > UB.z+bound) || (v.z<LB.z-bound)) 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<nThings;i++)
        UpdateClosest(Things[i],v,bound,closest);
};
```

Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



57

Searching octree of bounded things

```
void UpdateClosest(Thing* Thing,  
                  Vec3 v, double& bound, Vec3& closest)  
{ Vec3 cp = Thing->ClosestPointTo(v);  
  dist = LengthOf(cp-v);  
  if (dist<bound) { bound = dist; closest=cp;};  
};
```

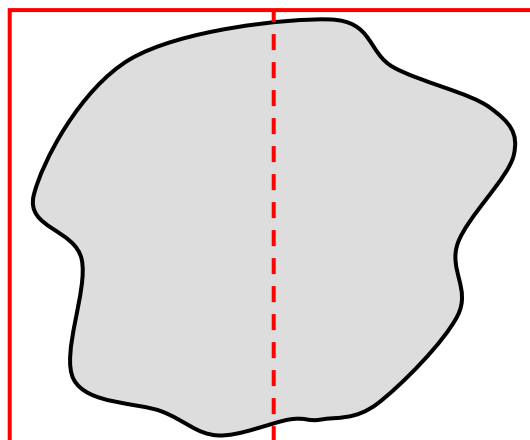
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



58

Covariance Trees*



* Referred to by my former student Seth Billings as Principal Direction Trees

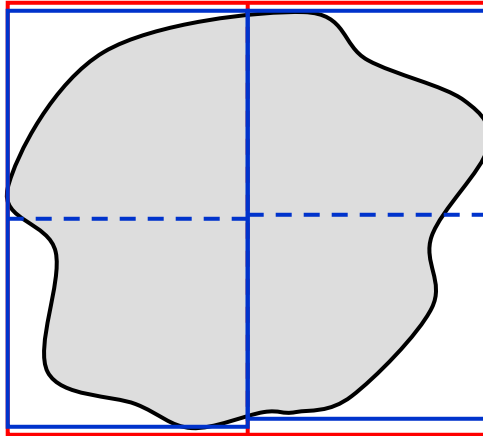
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



59

Covariance Trees



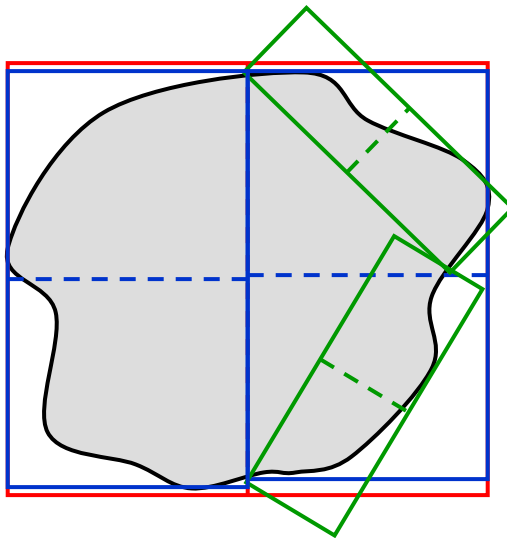
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



60

Covariance Trees



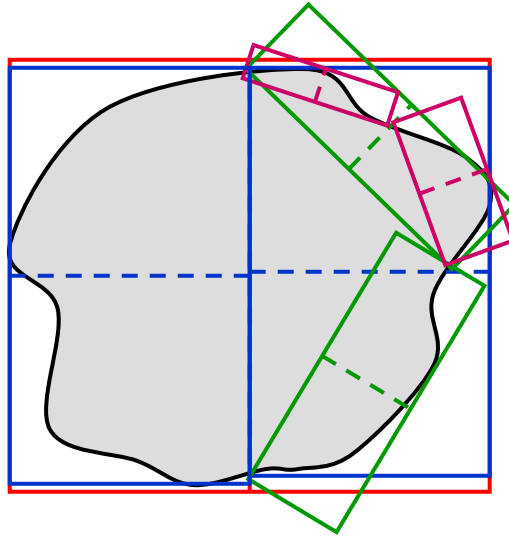
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



61

Covariance Trees



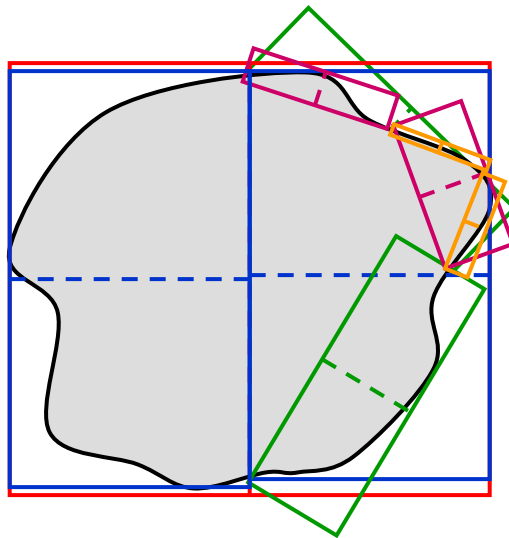
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



62

Covariance Trees



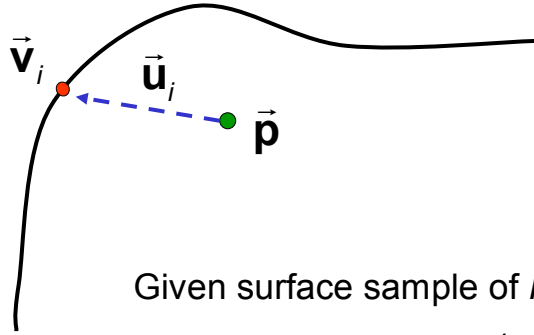
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



63

Covariance Tree Construction



Given surface sample of N points $\{\vec{v}_i\}$

Compute centroid $\vec{p} = \frac{1}{N} \sum_{i=1}^N \vec{v}_i$

Compute residual vectors $\vec{u}_i = \vec{v}_i - \vec{p}$

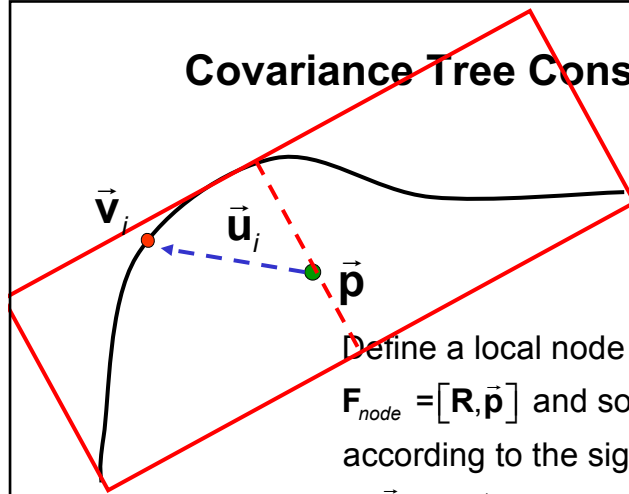
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



64

Covariance Tree Construction



Define a local node coordinate system

$\mathbf{F}_{node} = [\mathbf{R}, \vec{p}]$ and sort the surface points according to the sign of the x component of $\vec{b}_i = \mathbf{R}^{-1} \bullet \vec{u}_i$. Compute bounding box

$$\vec{b}^{\min} \leq \mathbf{R}^{-1} \bullet \vec{u}_i \leq \vec{b}^{\max}$$

Assign these points to "left" and "right" subtree nodes.

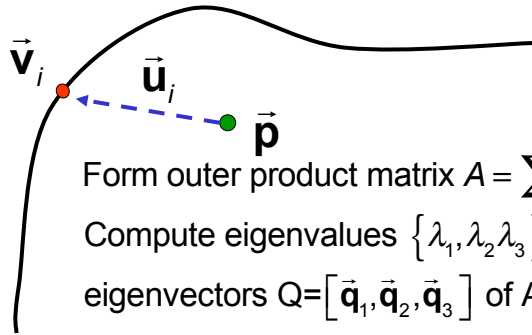
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



65

Covariance Tree Construction



Form outer product matrix $A = \sum_i \vec{u}_i \vec{u}_i^T$

Compute eigenvalues $\{\lambda_1, \lambda_2, \lambda_3\}$ and

eigenvectors $Q = [\vec{q}_1, \vec{q}_2, \vec{q}_3]$ of A

Find a rotation \mathbf{R} such that \mathbf{R}_x is the eigenvector corresponding to the largest eigenvalue.

(Depending on algorithm used, Q will be a rotation matrix, so all you may have to do is rotate Q)

Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



66

Constructing Cov Tree of Objects

```
class CovTreeNode {
    Frame F;           // splitting point
    Vec3 UB;           // corners of box
    Vec3 LB;
    int HaveSubtrees;
    int nThings;
    CovTreeNode* SubTrees[2];
    Thing** Things;
    :
    :
    CovTreeNode(Thing** Ts, int nT);
    ConstructSubtrees();
    void FindClosestPoint(Vec3 v, double& bound, Vec3& closest);
};
```

Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



67

Constructing Cov Tree of Things

```

CovTreeNode(Thing** Ts, int nT)
{
    Things = Ts; nThings = nT;
    F = ComputeCovFrame(Things,nThings);
    [UB,LB] = FindBoundingBox(F,Things,nThings);
    ConstructSubtrees();
};

[vec3 UB,vec3 LB]=FindBoundingBox(F,Things,nThings)
{
    UB = LB = F.inverse()*(Things[0]->SortPoint());
    for (int k=0;k<nThings;k++)
        { [LB,UB] = Things[k]->EnlargeBounds(F,UB,UB);
        };
    return [UB,LB];
};

```

Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



68

Constructing Cov Tree of Things

```

Frame F = FindCovFrame(Thing** Ts, int nT)
{
    [vec3 Points, int nP] = ExtractPoints(Ts,nT);
    // may extract nT sort points or perhaps
    // all corner points if things are triangles
    return FindCovFrame(Points,nP);
};

Frame F = FindCovFrame(vec3* Ps, int nP)
{
    vec3 C = Centroid(Ps,nP);
    Matrix A = 0;
    for (i=0;i<nP;i++) A+=OuterProduct(Ps[i],Ps[i]);
    R = CorrespondingRotationMatrix(A); // see notes
    return Frame(R,C);
};

```

Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



69

Constructing Cov Tree of Things

```
ConstructSubtrees()
{ if (nThings<= minCount || length(UB-LB)<=minDiag)
  { HaveSubtrees=0; return; };
  HaveSubtrees = 1;
  int nSplit;
  nSplit = SplitSort(F,things);
  Subtrees[0] = CovarianceTreeNode(Things[0],nSplit);
  Subtrees[1] = CovarianceTreeNode(Things[nSplit],nThings-nSplit);
}
```

Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



70

Constructing Cov Tree of Things

```
Int nSplit = SplitSort(Frame F, Thing** Ts,int nT)
{ // find an integer nSplit and reorder Things(...) so that
  //   F.inverse()*(Thing[k]->SortPoint()).x <0 if and only if k<nSplit
  // This can be done "in place" by suitable exchanges.
  return nSplit;
}
```

Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



71

Covariance tree search

Given

- node with associated \mathbf{F}_{node} and surface sample points \vec{s}_i .
- sample point \vec{a} , previous closest point \vec{c} , $dist = \|\vec{a} - \vec{c}\|$

Transform \vec{a} into local coordinate system $\vec{b} = \mathbf{F}_{node}^{-1}\vec{a}$

Check to see if the point \vec{b} is inside an enlarged bounding box $\vec{b}^{min} - dist \leq \vec{b} \leq \vec{b}^{max} + dist$. If not, then quit.

Otherwise, if no subnodes, do exhaustive search for closest.

Otherwise, search left and right subtrees.



Searching a Covariance Tree of Things

```
void CovarianceTreeNode::FindClosestPoint
    (Vec3 v, double& bound, Vec3& closest)
{
    vLocal=F.Inverse()*v; // transform v to local coordinate system
    if (vLocal.x > UB.x+bound) return;
    if (vLocal.y > UB.y+bound) return;
    // similar checks on remaining bounds go here .... ;
    if (vLocal.z < LB.z-bound) return;
    if (HaveSubtrees)
    {
        Subtrees[0].FindClosestPoint(v,bound,closest);
        Subtrees[1].FindClosestPoint(v,bound,closest);
    }
    else
    {
        for (int i=0;i<nThings;i++)
            UpdateClosest(Things[i],v,bound,closest);
    }
};
```



Searching a Covariance Tree of Things

```
void UpdateClosest(Thing* T, Vec3 v, double& bound, Vec3& closest)
{ // here can include filter if have a bounding sphere to check
  Vec3 cp = T->ClosestPointTo(v);
  dist = LengthOf(cp-v);
  if (dist<bound) { bound = dist; closest=cp;};
};
```

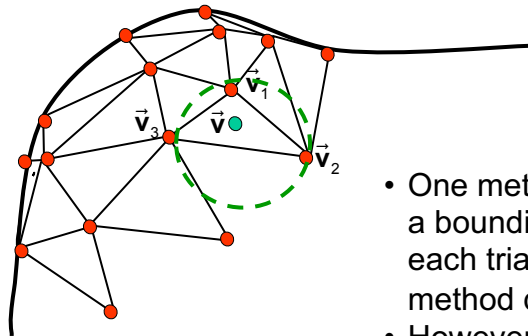
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



74

Covariance Trees for Triangle Meshes



- One method is simply to place a bounding sphere around each triangle, and then use the method discussed previously
- However, this may be inconvenient if the mesh is deforming

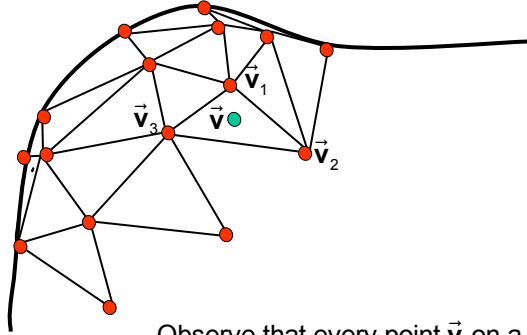
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



75

Covariance Trees for Triangle Meshes



Observe that every point \vec{v} on a triangle $[\vec{v}_1, \vec{v}_2, \vec{v}_3]$ can be expressed as a convex linear combination $\vec{v} = \lambda_1 \vec{v}_1 + \lambda_2 \vec{v}_2 + \lambda_3 \vec{v}_3$ with $\lambda_1 + \lambda_2 + \lambda_3 = 1$. Therefore, if $[\vec{v}_1, \vec{v}_2, \vec{v}_3]$ are in some bounding box, then \vec{v} will also be in that bounding box

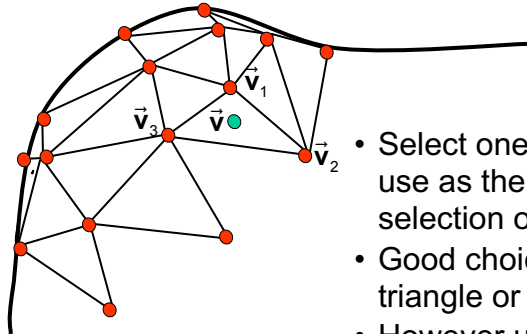
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



76

Covariance Trees for Triangle Meshes



- Select one point on the triangle to use as the “sort” point for selection of left/right subtrees.
- Good choices are centroid of triangle or just one of the vertices.
- However use all vertices of each triangle in determining the size of bounding boxes.
- Note this would work equally well for octrees.

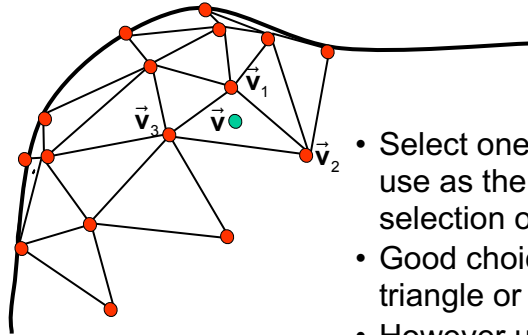
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



77

Covariance Trees for Triangle Meshes



- Select one point on the triangle to use as the “sort” point for selection of left/right subtrees.
- Good choices are centroid of triangle or just one of the vertices.
- However use all vertices of each triangle in determining the size of bounding boxes.
- Note this would work equally well for octrees.

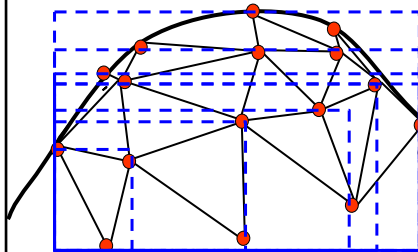
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



78

Covariance Trees for Triangle Meshes



- Select one point on the triangle to use as the “sort” point for selection of left/right subtrees.
- Good choices are centroid of triangle or just one of the vertices.
- However use all vertices of each triangle in determining the size of bounding boxes.
- Note this would work equally well for octrees.

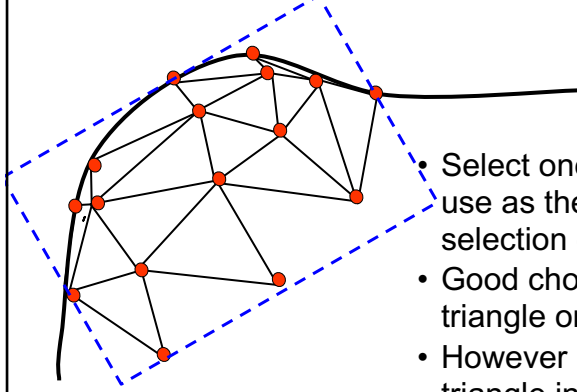
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



79

Covariance Trees for Triangle Meshes



- Select one point on the triangle to use as the “sort” point for selection of left/right subtrees.
- Good choices are centroid of triangle or just one of the vertices.
- However use all vertices of each triangle in determining the size of bounding boxes.
- Note this would work equally well for octrees.

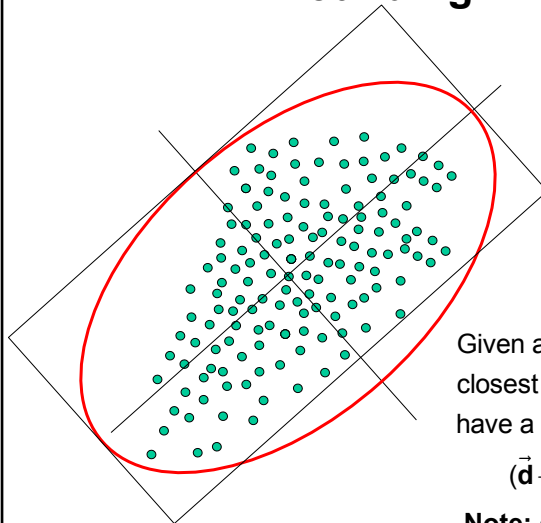
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



80

An Alternative to Bounding Boxes: Bounding Ellipsoids



Compute

$$\bar{\mathbf{p}}_c = \frac{1}{N} \sum_N \bar{\mathbf{v}}_i$$

$$\bar{\mathbf{u}}_i = \bar{\mathbf{v}}_i - \bar{\mathbf{p}}_c$$

$$\mathbf{A} = \sum_i \bar{\mathbf{u}}_i \bar{\mathbf{u}}_i^T = \mathbf{Q} \mathbf{\Lambda} \mathbf{Q}^T$$

$$\mathbf{\Lambda} = \text{diag}(\bar{\lambda})$$

$$\rho^2 = \max_i \bar{\mathbf{u}}_i^T \mathbf{A} \bar{\mathbf{u}}_i$$

Given a search point $\vec{\mathbf{d}}$ and previous closest distance δ , the ellipsoid may have a closer point if

$$(\vec{\mathbf{d}} - \vec{\mathbf{p}}_c)^T \mathbf{A} (\vec{\mathbf{d}} - \vec{\mathbf{p}}_c) < \rho^2 + (\delta \max_k \lambda_k)^2$$

Note: can probably get a tighter bound, but this will work

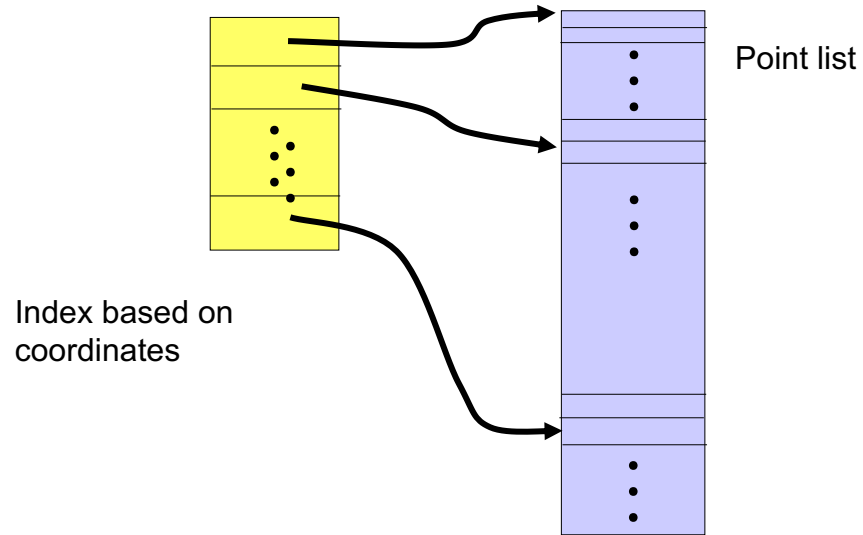
Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology



81

Simple spatial sort



Copyright Russell Taylor, 2010-2020

Engineering Research Center for Computer Integrated Surgical Systems and Technology

