Cubical Marching Squares: Adaptive Feature Preserving Surface Extraction from Volume Data

Chien-Chang Ho      Fu-Che Wu      Bing-Yu Chen
Yung-Yu Chuang      Ming Ouhyoung

National Taiwan University
Overview: Marching Cubes

- The most cited paper in history of SIGGRAPH


from [www.nasa.gov](http://www.nasa.gov)
from [www.openqvis.com](http://www.openqvis.com)
from [graphics.csie.ntu.edu.tw](http://graphics.csie.ntu.edu.tw)
Overview: Problems & Goals

- Adaptive Resolution
- Consistent Topology
- Sharp Features
- Parallel Processing
Overview: Problems & Goals

- Adaptive Resolution
- Consistent Topology
- Sharp Features
- Parallel Processing
Overview: Problems & Goals

- Adaptive Resolution
- Consistent Topology
- Sharp Features
- Parallel Processing
Overview: Problems & Goals

- Adaptive Resolution
- Consistent Topology
- Sharp Features
- Parallel Processing
Overview: Problems & Goals

- Adaptive Resolution
- Consistent Topology
- Sharp Features
- Parallel Processing
Marching Cubes Table

- Using binary pattern of eight vertices
- Totally 256 cases in 15 configurations
Steps of marching cubes
Adaptive resolution

- [WILHELMS J., GELDER A. V., 1992]
- [SHU R. et al, 1995], etc.
Adaptive resolution
Crack patching
Consistent topology

- Ambiguity problems
  - [NIELSON G. M., HAMANN B. 1991]
  - [NATARAJAN B. K., 1994]
  - [CHERNYAEV E., 1995], etc.
Sharp features

- [KOBBELT L. P. et al, 2001], EMC (Extended Marching Cubes).
- [JU T. et al, 2002], DC (Dual Contouring)
Hermite data: with additional normal direction

[Diagram showing Hermite data with grid and additional normal direction highlighted with arrows and markers.]

Sharp Features
Real-time through parallel processing

• limited by inter-cell dependency
Inter-cell dependency
Problems & Goals

- Adaptive resolution
- Sharp features
- Consistent topology
- Parallel processing
Cubical Marching Squares
CMS Algorithm

1: procedure **CUBICALMARCHINGSQUARES**(*HermiteData H*)
2:    InitializeBaseGrid(*B*);
3:    for each cell *c* in *B*
4:       **SUBDIVIDECELL**(*H*, *c*);
5:    end for
6:    for each leaf face *f*
7:       **GENERATESEGMENT**(*f*);
8:    end for
9:    for each leaf cell *c*
10:       **EXTRACTSURFACE**(*c*);
11:    end for
12: end procedure

▷ initialize a coarse base grid *B*
CMS Algorithm

1: procedure CubicalMarchingSquares(HermiteData \( H \))
2:   InitializeBaseGrid(\( B \));
3:   for each cell \( c \) in \( B \)
4:     \( \text{SubdivideCell}(H, c) \);
5:   end for
6:   for each leaf face \( f \)
7:     \( \text{GenerateSegment}(f) \);
8:   end for
9:   for each leaf cell \( c \)
10:   \( \text{ExtractSurface}(c) \);
11: end for
12: end procedure
CMS Algorithm

1: procedure CubicalMarchingSquares(HermiteData $H$)
2:    InitializeBaseGrid($B$); ▷ initialize a coarse base grid $B$
3:    for each cell $c$ in $B$
4:        SubdivideCell($H$, $c$);
5:    end for
6:    for each leaf face $f$
7:        GenerateSegment($f$);
8:    end for
9:    for each leaf cell $c$
10:       ExtractSurface($c$);
11: end for
12: end procedure
CMS Algorithm

1: procedure CUBICALMARCHINGSQUARES(HermiteData \( H \))
2: InitializeBaseGrid(\( B \));
3: for each cell \( c \) in \( B \)
4: \hspace{1cm} SUBDIVIDECELL(\( H, c \));
5: end for
6: for each leaf face \( f \)
7: \hspace{1cm} GENERATESEGMENT(\( f \));
8: end for
9: for each leaf cell \( c \)
10: \hspace{1cm} EXTRACTSURFACE(\( c \));
11: end for
12: end procedure

\( \triangleright \) initialize a coarse base grid \( B \)
CMS Algorithm

1: procedure CUBICALMARCHINGSQUARES(HermiteData \( H \))
2:   InitializeBaseGrid(\( B \));  \( \triangleright \) initialize a coarse base grid \( B \)
3:   for each cell \( c \) in \( B \)
4:     SUBDIVIDECELL(\( H, c \));
5:   end for
6:   for each leaf face \( f \)
7:     GENERATSEGMENT(\( f \));
8:   end for
9:   for each leaf cell \( c \)
10:  EXTRACTSURFACE(\( c \));
11: end for
12: end procedure
Segment generation on faces for normal cases
Segment generation on faces for ambiguity cases
Analysis of face ambiguity

- Separated
  
- Joined
  
Case 3.1

Case 3.2
Segment generation in motion
Inter-cell independency
Benefits - inter-cell independency

1. Faster  
2. Parallelizable  
3. Lower Error
Algorithm – CubicalMarchingSquares

1: procedure CubicalMarchingSquares(HermiteData H)
2:   InitializeBaseGrid(B);
3:   for each cell c in B
4:     SubdivideCell(H, c);
5:   end for
6:   for each leaf face f
7:     GenerateSegment(f);
8:   end for
9:   for each leaf cell c
10:    ExtractSurface(c);
11: end for
12: end procedure

▷ initialize a coarse base grid B
CMS is crack free

• Cracks are avoided using CMS
Benefits – adaptive & crack free
1. Smooth Shape 2. Reduce 3D \(\Rightarrow\) 2D
Results & Conclusion

Solutions

Previous work & Problems
Benefits - consistent topology
1. Correct Shape  2. Lower Error
Benefits - consistent topology

1. Correct Shape  2. Lower Error
Simulation

- Available shapes
- generated randomly in a limited space
## Average geometric errors

<table>
<thead>
<tr>
<th>case</th>
<th>times</th>
<th>DC</th>
<th>EMC</th>
<th>CMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3,590,980</td>
<td>0.01473</td>
<td>0.00586</td>
<td>0.00383</td>
</tr>
<tr>
<td>2</td>
<td>1,554,028</td>
<td>0.02309</td>
<td>0.01310</td>
<td>0.01013</td>
</tr>
<tr>
<td>3</td>
<td>207,302</td>
<td>0.10027</td>
<td>0.01801</td>
<td>0.01263</td>
</tr>
<tr>
<td>4</td>
<td>30,972</td>
<td>0.23064</td>
<td>0.00601</td>
<td>0.00422</td>
</tr>
<tr>
<td>5</td>
<td>803,311</td>
<td>0.03779</td>
<td>0.02631</td>
<td>0.02011</td>
</tr>
<tr>
<td>6</td>
<td>101,875</td>
<td>0.11998</td>
<td>0.02737</td>
<td>0.02633</td>
</tr>
<tr>
<td>7</td>
<td>12,198</td>
<td>0.17139</td>
<td>0.09565</td>
<td>0.01628</td>
</tr>
<tr>
<td>8</td>
<td>109,141</td>
<td>0.03979</td>
<td>0.02831</td>
<td>0.02184</td>
</tr>
<tr>
<td>9</td>
<td>72,201</td>
<td>0.04721</td>
<td>0.03525</td>
<td>0.02492</td>
</tr>
<tr>
<td>10</td>
<td>4,237</td>
<td>0.19682</td>
<td>0.05283</td>
<td>0.04541</td>
</tr>
<tr>
<td>11</td>
<td>70,238</td>
<td>0.04789</td>
<td>0.03535</td>
<td>0.02620</td>
</tr>
<tr>
<td>12</td>
<td>30,706</td>
<td>0.09845</td>
<td>0.04559</td>
<td>0.02419</td>
</tr>
<tr>
<td>13</td>
<td>1,405</td>
<td>0.85461</td>
<td>0.92935</td>
<td>0.00100</td>
</tr>
<tr>
<td>14</td>
<td>70,238</td>
<td>0.04821</td>
<td>0.03573</td>
<td>0.02653</td>
</tr>
</tbody>
</table>
Virtual Sculptor
Remeshing
CSG & LOD
Conclusion

Marching Cubes
Topological Marching Cubes
Extended Marching Cubes
Dual Contouring
Cubical Marching Squares

Parallel Processing
Sharp Features
Consistent Topology
Adaptive Resolution
Future work

- Medical imaging by CMS
- Acceleration using GPU
Preliminary result: Acceleration using GPU

```
struct Pix
{
    float2 tex : TEXCOORD0;
};

struct Cout
{
    float4 dif : COLOR0;
};

Cout SharpFeature(const Pix In)
{
    Cout Out;
    float3 u = tex2D(SN0, In.tex).xyz;
    float3 v = tex2D(SN1, In.tex).xyz;
    float3 v0 = tex2D(SN2, In.tex).xyz;
    float3 n0 = tex2D(SN3, In.tex).xyz;
    float3 v1 = tex2D(SN4, In.tex).xyz;
    float3 n1 = tex2D(SN5, In.tex).xyz;
    float minSharpAngle = abs(dot(n0, n1));
    bool bFlag = minSharpAngle < constThetaSharp;
    if (!bFlag)
    {
        float fT = dot(n0, v0);
        float fF = dot(n1, v1);
        float fDF = fT * fF;
        float1Det = fA ^ 1 * fB ^ 1;
        if (abs(fDet) < 0.000001)
        {
            bFlag = false;
        }
        else 
        {
            float fU = (fC ^ 1 * fB ^ 1) / fDet;
            float fV = (fA ^ 1 * fC ^ 1) / fDet;
            Out.dif.xyz = fU * u + length(u) * fV * v + length(v);
            Out.dif.w = tex2D(SN0, In.tex).w;
        }
    }
    if (!bFlag)
    {
        Out.dif.xyz = (tex2D(SN2, In.tex).xyz + tex2D(SN4, In.tex).xyz) / 2;
        Out.dif.w = -1;
    }
    return Out;
```
Thanks
Applications
Cube based patterns by CMS
New patterns
• Triangle count
• Memory space requirement
• Computation
Overview: Problems & Goals

- Adaptive Resolution
- Consistent Topology
- Sharp Features
- Parallel Processing
Marching squares (2D)...
Ambiguity problem

- Face ambiguity
  - [NIELSON G. M., HAMANN B. 1991], etc.
Ambiguity problem…
Segment generation on faces

- Table look up
- Detect sharp features
- Resolve topological ambiguity
Resolving face ambiguity