初步思路：3D mesh→ mesh decomposition为大量super-patch→ 两两merge形成hierarchical segmentation

1 super-patch

对应2D image的superpixel，3D mesh decomposition的算法有很多，但是能够得到源代码的工作几乎没有。不过，受到近年来比较受欢迎的SLIC superpixel算法[1]的启发，根据[5]的工作，我们提出一个利用K-means的super-patch算法。对于任意一个triangular mesh model，标准化后，我们将其看作一个graphical model, 用G(V,E)：每个节点V_i代表一个face，相互连接的节点表示相互相邻的面，每一条Edge上定义$Distance(V_i, V_j)$为两个face的“距离”：

$$Distance(V_i, V_j) = a \cdot (1 - cos^2(\alpha)) + b \cdot Phy\_Dist(V_i, V_j)$$

其中α是两个面之间的dihedral angle，$Phy_Dist(V_i, V_j)$是两个面的重心到相邻edge的中点的距离之和。权重a, b保证了这个距离在[0,1]之间。具体的选取由下段描述的training决定。

1.1 Training

受Berkeley的segementation dataset[3]的影响，2009年Princeton发布了3D segmentation的benchmark[2]，400个model中每个模型都由若干位志愿者做出了分割，做为ground truth。任取200个模型作为training set，剩下的其中100个模型作为test set，另外100个模型作为validation set，把training set中的每一对相邻的face提取出来，计算$((1 - cos^2(\alpha)), Phy_Dist(V_i, V_j))$，定义$Distance_{groundtruth}$为$V_i, V_j$同属不同segment中的概率（在每个模型中13个ground truth中label不同的概率）。接下来就可以通过一个简单的logistic regression 来将a, b确定。

1.2 K-means clustering

当distance被定义好后，对于任意模型，每一对相邻的(V_i, V_j)之间的distance就可以算出来了。再定义任意两个不相邻的face 之间的距离为

$$Distance(V_i, V_j) = min_{V_3 \neq V_1, V_2}(Distance(V_1, V_3) + Distance(V_3, V_2))$$

在计算的时候可以运用寻找最短路径的Shortest Path Faster Algorithm(SPFA)算法，接下来就可以开始做clustering了。由于我们定义的距离函数很简单，而且对初始的over-segmentation的精度没有特别严格的要求（仅仅是想让每个3d 模型中的patch个数相同），所以我们取k为一个比较大的值。(k=2000)

Initialize Cluster centers C_k by randomly choosing k faces
set label $l(i) = -1$ for each face i
set distance $d(i) = \infty$ for each face i
set residual error $E = \infty$
while E won't change **do**
  for Each cluster center C_k **do**
    for each face i **do**
      compute $D = Distance(C_k, i)$
      if $D < d(i)$ **then**
        set $d(i) = D$
        set $l(i) = k$
      end if
    end for
  end for
  Compute new cluster centers(move to the closest face centers)
  Compute residual error E(distance between previous centers and recomputed centers)
end while

super patch的代码（包括training）在3d_cascaded_seg/super_patch中。

2 Super-Patch Merging

2.1 Boundary Recall Measurement

受[1]中二维图片measurement的启发，我们定义三维mesh的boundary recall如下：

$$BR_G(S) = \frac{\sum_{p \in \partial G} \mathbb{1}(\min_{q \in \partial S} distance(p, q) < \epsilon)}{\partial G} \tag{1}$$

其中G为ground truth segmentation, S 为machine segmentation, ∂表示分割的边缘(boundary)。对于3D mesh，定义两两vertex之间的edge作为boundary，这样一个segmentation的边缘就由很多线段组成。将每条线段类比为2D中的pixel，2D中两个pixel之间的距离可以直接计算，而计算两个线段的距离的时候，我们利用[6]中的方法（helper/DistBetween2LineSegment_mex.mex）

这个measurement数值越大说明真实的边缘检测出来的比例越多。

★我写的计算这个measurement的代码复杂度是$O(\partial G * \partial S)$，因此计算速度极慢，不知道能不能写一个更快的算法，因为速度原因在cascade training的时候也收到了很大的影响。

2.2 other measurements

其他的measurement包括rand index(有代码), Cut Discrepancy, Hamming Distance等等。其中rand index是很多3D segmentation paper中都使用的度量。这个度量的计算写出的代码运行可以很快(randindex.cpp)。

2.3 Cascaded algorithm for super-patch agglomeration

这部分的思路主要借鉴[4]。将每个super-patch对应于文中的super-pixel, 其余的思路是类似的。★ 所有算法和[4]中框架一样。

对于一个3D Model I_i, 我们将其分为k_i个super-patch$R_i = \{R_{i,1}, ..., R_{i,k_i}\}$，再令$N(R_i)$为所有的相邻区域。每一个区域对$(p,q) \in N(R_i)$ 都由一个表示特征的向量$\phi_{p,q}(I_i, R_i)$组成。具体使用的特征在section 2.3.3中会介绍。

我们同时也有ground truth的数据，它们将作为训练样本。这些真实的分割数据是以带标签的3D模型表示的，由于我们不假定任何类别信息，所以我们仅仅需要得到的信息是：任意一对像素是否属于同一区域。

利用模型I的真实分割的数据，我们可以给$N(R_i)$中的区域对加上标签：任给一个真实分割图G_i，我们将$R_{i,p}$标定为G_i中与它重合最大的区域的标签。接下来，如果$R_{i,p}$与$R_{i,q}$的标签是相同的，则定义$y^i_{pq} = 1$，否则$y^i_{pq} = 0$。如果对于一张图有很多的真实分割，则定义y^i_{pq}为针对所有真实分割定义标签的平均。这样y^i_{pq}的值就在0和1之间，度量了人们认为$R_{i,p}$和$R_{i,q}$相同的概率，同时也反映了区域之间边缘的强度。

将训练样本$I_1, ..., I_N$中区域对的特征和他们的标签放在一起，我们得到$\{< \phi_i, y_i >\}$，这样就是一个预测问题：给定一个区域I和初始分割R，估计相邻区域聚合的条件后验概率为$Pg(p,q;I,R) \triangleq= P(y_{pq} = 1|\phi_{pq}(I,R))$。如果$Pg(p,q;I,R) > \frac{1}{2}$，则我们必须聚合$R_p$和$R_q$，并且将它们之间的边缘去掉，否则我们将保留边缘。

虽然我们可以直接一次性地预测所有的区域对，但我们寻求一个比较简单的贪婪聚合算法。在每一个循环中，我们将拥有最大的Pg的区域对聚合，直到没有任何区域对$Pg > \frac{1}{2}$为止。由于我们是从一个比较精确的分割将其聚合为一个比较粗糙的分割，我们会保证存在一个比较有意义的分割R'，需要注意的是，在聚合之前的更新操作只会影响相邻区域，而不会影响其他区域

2.3.1 training

我们用一个logistic回归模型来表示Pg

$$Pg(p,q;I,R) = 1/(1 + exp{-w^T \phi_{pq}(I,R)})$$

这个模型一般来说是用最小化log-loss方法进行训练的，然而，在这里

简单地利用log-loss进行训练会失败，因为它会对称地惩罚两种错误:过分和低估Pg的大小。给定一个典型的图像I，一般来说它有1000个superpixels，仅仅有一小部分$N(R)$中的区域对为true negative，因为大部分相邻的superpixels应当被聚合在一起。

我们的解决办法同样是修改损失方程：false positive（错误地聚合的区域）的cost前面乘一个系数α，进一步地，我们还在损失方程前面乘上边缘的长度L_i。这个尺度变换反映了“擦除”长的边缘比短的边缘惩罚的多：

$$\mathbf{w}^*(\alpha) \;=\; \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \Big[y_i \log Pg(\boldsymbol{\phi}_i; \mathbf{w}) + \alpha(1-y_i) \log(1 - Pg(\boldsymbol{\phi}_i; \mathbf{w})) \Big] \quad (2)$$

(2)中的带权损失函数是凸的，因此可以像通常的logistic回归模型那样优化。

我们对于α的选择是由下面的想法产生的：我们想要尽可能地保持最初分割的Boundary Recall，同时还要尽可能多地聚合区域。其中Boundary Recall $REC(R, G)$定义为每张图中真实分割G中的边缘里被检测出的比例的平均。

一般地，Recall的值相对于α是递增的。假定最初的分割中的recall 是r，则我们希望找到一个α的值，它尽可能地小，并且使得recall 的值不会降低到$r - \rho$之下，其中ρ为一个很小的数值。这样就启发我们寻找一个优化α的有效算法：取一个0 到一个很大的数的区间包含α，接下来使用二分查找方法。我们利用两个图像数据集的子集：I_{train}来选取α，而I_{tune}用刚刚得到的α来评价Recall 的值。

2.3.2 cascaded merging

当模型用带权的损失函数(2)进行训练的时候，聚合的过程一般来说会很早就停止了，这样就会留下很多没有聚合的区域s。原因在于我们的模型在聚合的时候十分的“小心”，同时由于新聚合的区域中的特征的分布和之前训练中用到的特征不同，很小的区域中无效的特征可能会在更大的区域中变得有效。这引导我们产生一个简单的想法：对聚合的区域重新提取特

征，重新在更大的区域上进行训练。这样的第二个模型会聚合更多的分割区域，不过它仍然会停止。我们则会继续训练第三个模型，以此类推。

训练这个级联模型的过程和其他级联模型的训练方法是类似的，最大的区别在于，我们使用的是非对称的损失函数，而不是手动地调试参数。这样我们就可以训练一个更深的级联。

总而言之：整个算法从super-patch开始，通过一系列阶段进行增殖。在每个阶段中，一些区域被阶段中学习到的模型进行聚合，接下来的阶段用粗糙一些的分割作为输入。当聚合过程停止的时候，我们可以向后遍历，并记录每一步的分割过程。这样，我们就可以用两种方法控制分割的尺度：或者输入想要得到的分割个数，或者输入分割阶段的次数。

★ 这一部分由于要依赖boundary recall的计算(见[4])所以实现起来十分慢。这一部分是目前主要的瓶颈。

2.3.3 features between neighboring superpixels

这里所说的features都是定义在相邻两个patch $R_{i,q}$上的。

- Bounding Box

  先计算每个patch i的bounding box b_i, 在$R_{i,q}$中，定义$b_i \cap b_q$为两个box的相交部分c_{iq}，取$\phi_1(i,q) = \max(\frac{volume(b_i)}{volume(c_{iq})}, \frac{volume(b_q)}{volume(c_{iq})})$，此数值在0，1之间，越接近1 说明二者的包含关系越明显。（可以将bounding box换为凸包，更精确）

- Average Norm Vector

  每个mesh i上的法向量为n_i，对于每个patch j，定义其平均法向量为所有mesh上的法向量的加权和，权重与每个mesh的面积成正比。

- Others: to be continued..

2.4 future work

- 更多的features。

- 改进的super-patch算法：如一开始在选取k个点的时候将其均匀分布在模型中
- merging前对模型做出初步分析，定义不同的merging模型

附录

整个project均在F盘上进行操作。

A Mesh Segmentation Data

- Princeton's segmentation dataset 存放在F:/MeshsegBenchmark-1.0 中
- data/{train, test, val}.txt 分别为training, test, validation set
- data/off 中存放了所有3维模型，共380个，每10个为一个类别（人，椅子，杯子，等等。。）
- data/seg 中存放了各个算法的分割结果，其中Benchmark为ground truth, super_patch 中存放了super-patch 算法的结果，每个模型2000个分割。

这些是所有需要用到的东西。。

B codes

代码保存在F:/github/3d_cascaded_seg 其中的readme文件介绍了各个目录的组织情况，全部代码在这里托管https://github.com/luvegood/3d_cascaded_seg 请clone这里的代码，台式机里的代码不一定是最新的。同时，代码还依赖其他的toolbox：geom3d(计算bounding box等等都要

用到), toolbox_graph(3D 图结构的时候利用), minFunc(learning的时候利用)，均在F:/github中。

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels. *École Polytechnique Fédéral de Lausssanne (EPFL), Tech. Rep*, 149300, 2010.

[2] Xiaobai Chen, Aleksey Golovinskiy, and Thomas Funkhouser. A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3), August 2009.

[3] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[4] Zhile Ren and Gregory Shakhnarovich. Image segmentation by cascaded region agglomeration. June 2013.

[5] Shymon Shlafman, Ayellet Tal, and Sagi Katz. Metamorphosis of polyhedral surfaces using decomposition. In *Computer Graphics Forum*, volume 21, pages 219–228. Wiley Online Library, 2003.

[6] Dan Sunday. Distance between 3d lines and segments. http://geomalgorithms.com/a07-_distance.html.