Предложения по конкурсу на лучшую программу сегментации изображений

1 Что такое качество сегментации

Понятие качества сегментации в общем виде формализовать трудно. Тем не менее, попробуем предложить некоторый вариант численной оценки. Это позволит сравнивать между собой различные программы и выявлять лучшие.

Рассмотрим пару полноцветных (TrueColor) изображений, не обязательно одинакового размера.

- Из второго изображения вырежем круг фиксированного радиуса (или, более обще, фиксированную кусочно-гладкую фигуру).
- Вклеим его в первое изображение опять же в случайное место.
- Сегментируем полученное изображение.
- Некоторые сегменты могут пересекаться как с внутренностью вклеенной фигуры, так и с её внешностью. Назовем такие сегменты *ошибочными*. Часть точек ошибочного сегмента принадлежит внешности фигуры, часть – внутренности. Меньшую часть из них назовем *ошибочными* точками.
- Качество сегментации будем обозначать двумя параметрами:
    1. Средний размер сегмента (в точках).
    2. 100% – (доля *ошибочных* точек по отношению к площади фигуры, %).

Замечания.

1. Предлагаемая далее программа позволяет готовить набор картинок требуемого вида (со вклеенными кругами) и тестировать заданную программу сегментации (exe- или bat-файл). Программа должна быть доступна всем участникам. Жюри будет тестировать на своем наборе изображений, каждый участник может подготовить для отладки у себя на компьютере свой набор.
2. Представленные алгоритмы сегментации должны быть оформлены в виде exe- (или bat-) файлов и работать исключительно в режиме «командной строки». Любой диалог с пользователем («Выберите файл для сегментации») категорически запрещен, т.к. программа будет вызываться автоматически. Файл для сегментации и желаемое количество сегментов передаются как аргументы в командной строке.
3. Желательно (хотя и не обязательно), программа должна уметь сегментировать изображение на различное заданное количество сегментов, например команда

Segm_first.exe pict.bmp pict.txt 250

сегментирует картинку pict.bmp примерно на 250 сегментов записывает результат в файл pict.txt. Сравниваются программы при примерно одинаковом количестве полученных сегментов.

4. Текущая версия тестовой программы вклеивает именно круги. Чтобы не давать преимущества программам, целенаправленно ищущим окружности, круги можно будет заменить на более сложную фигуру.
5. Формат всех рассматриваемых файлов – BMP-24. Это простейший формат файла, его можно прочитать и сохранять без дополнительных библиотек, детальнее – см. далее.
6. Вклеивать круги лучше не совсем уж в случайное место. Его можно подобрать так, чтобы контраст с окрестностями был поменьше. Предлагаемая далее программа так и делает.

2 Что такое сегментация

Сегментацией изображения называется его разбиение на однородные связные области (сегменты) по некоторому признаку.

С вычислительной точки зрения сегментацией изображения размера mx*my будем называть матрицу того же размера из натуральных чисел. Каждое число обозначает номер сегмента (>0), к которому принадлежит точка. Такие матрицы сохраняются в текстовом файле вида

```
mx my комментарий
s00 s10 …
s01 s11 …
…
```

например, при сегментации файла размером 4*3:

```
4 3 ; в первой строке файла заданы размеры матрицы по x (4) и по y (3)
1 2 1 1
2 2 1 1
2 2 2 3
```

где цифры в теле матрицы означают номер сегмента (>0), к которому относится каждая точка. Сегменты предполагаются связными (точнее говоря, связными по 4-м соседним точкам) и их номера не обязаны идти подряд. Никакое упорядочение сегментов также не требуется.

3 Вспомогательные программы

3.1 segm_show.exe – показать сегментацию

Текстовые файлы с сегментацией велики и ненаглядны. Для визуального просмотра результатов сегментации предлагается программа segm_show.exe. При запуске без параметров выдает подсказку:

Show segmentation
Usage:
segm_show.exe segm.txt [pict.bmp]

Results:
stat.txt, segm.bmp [, segmA.bmp]
stat.txt - staticstics

Рисует данную сегментацию (segm.txt) в графическом виде. Например, пусть дан текстовый файл sgm_000.txt с сегментацией (показан только его верхний левый угол):

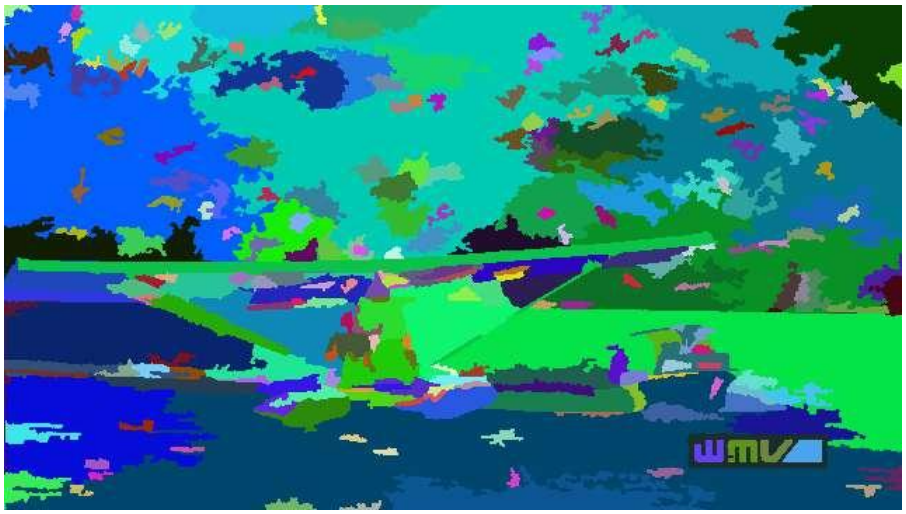
```
640  360
   169   169   169   169   169   169
   169   169   169   169   169   169
   169   169   169   169   169   169
   169   169   169   169   169   169
   169   169   169   169   169   169
   169   169   169   169   169   169
   169   169   169   169   169   169
   169   169   169   169   169   169
   169     3     3     3   169   169
   169     3     3     3     3     3
   169     3     3     3     3     3
```

Он является результатом некоторой сегментации рисунка 000012.bmp:

В результате выполнения команды:

```
segm_show.exe sgm_000.txt
```

в текущем каталоге появится графический файл segm.bmp:



Каждая точка в нем показана цветом, зависящим от номера сегмента:

- Уровень красного равен (номер сегмента mod 256). Благодаря этому мы всегда можем увидеть номер текущего сегмента (по модулю 256).
- Уровни зеленого и синего выбираются как некоторая псевдослучайная функция от номера сегмента.

Кроме того строится еще файл stat.txt, содержащий статистику по сегментам, в нашем случае это:

```
640  360 --
   0:         0 mid(   0,     0) rad(   0)
   1:     29896 mid( 327,   323) rad( 162)
   2:     27300 mid( 304,    77) rad(  94)
   3:     17997 mid(  76,   102) rad(  64)
   4:     14301 mid( 516,   249) rad(  99)
…
 236:        59 mid( 314,   259) rad(   6)
 237:        54 mid( 287,   240) rad(   3)
 238:        47 mid( 209,   211) rad(   3)
 239:        43 mid( 271,   192) rad(   4)
 240:        38 mid( 282,   200) rad(   5)
```

(Всего в сегментации – 240 сегментов с номерами от 1 до 240). В этом файле для каждого сегмента указан его размер (в точках), центр (среднее арифметическое всех точек сегмента) и радиус (среднеквадратичное расстояние точек от центра).

Если выполнить команду с двумя аргументами

```
segm_show.exe sgm_000.txt 000012.bmp
```

то будет построен еще один файл sgmA.bmp:

в котором сегменты наложены на изображение.

3.2 Тривиальный пример: сегментация на квадраты

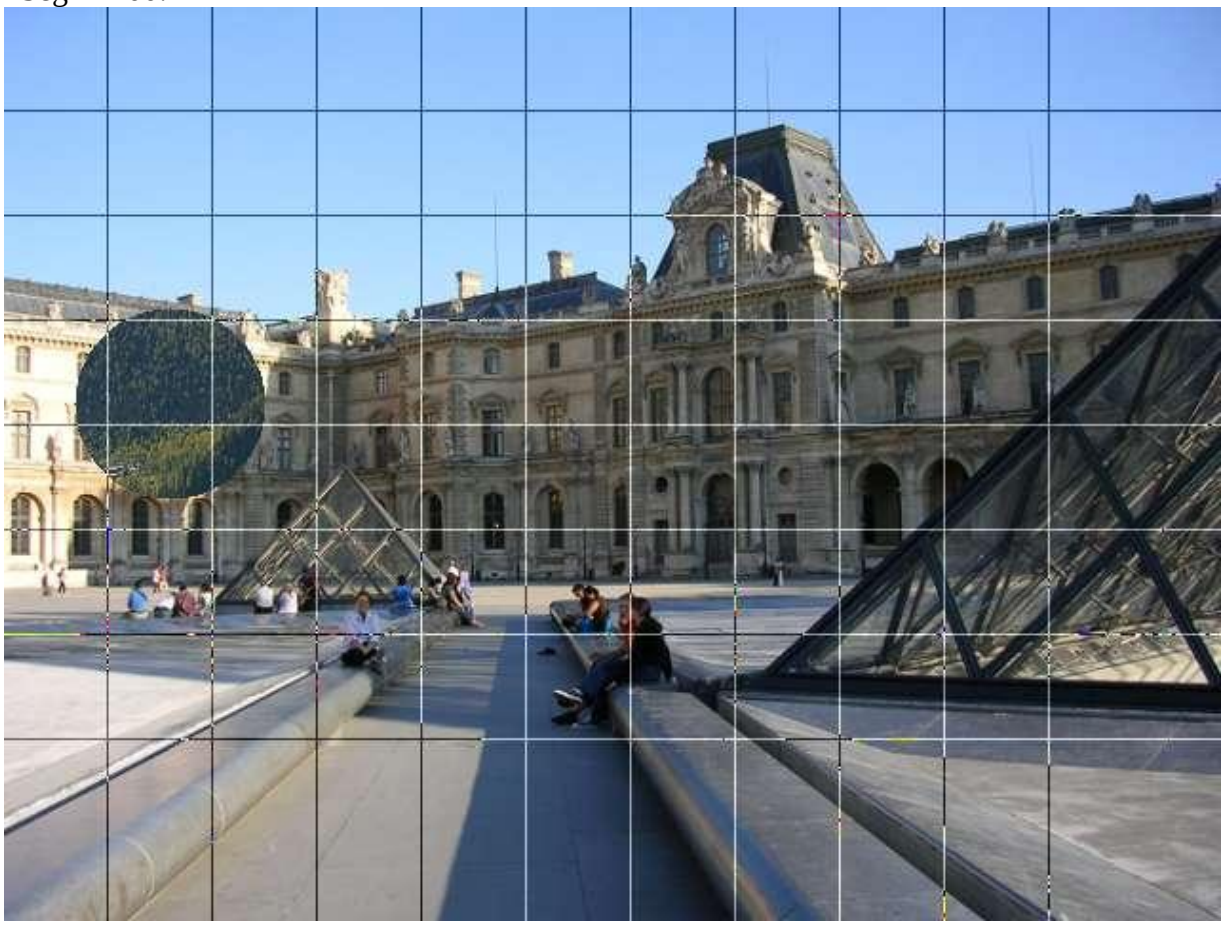
В качестве тривиального примера предлагается программа, сегментирующая изображение на квадраты: Segm_by_squares.exe,

Segmentation by squares.
Usage:
Segm_by_squares.exe pict.bmp segm.txt [nSegm=100]

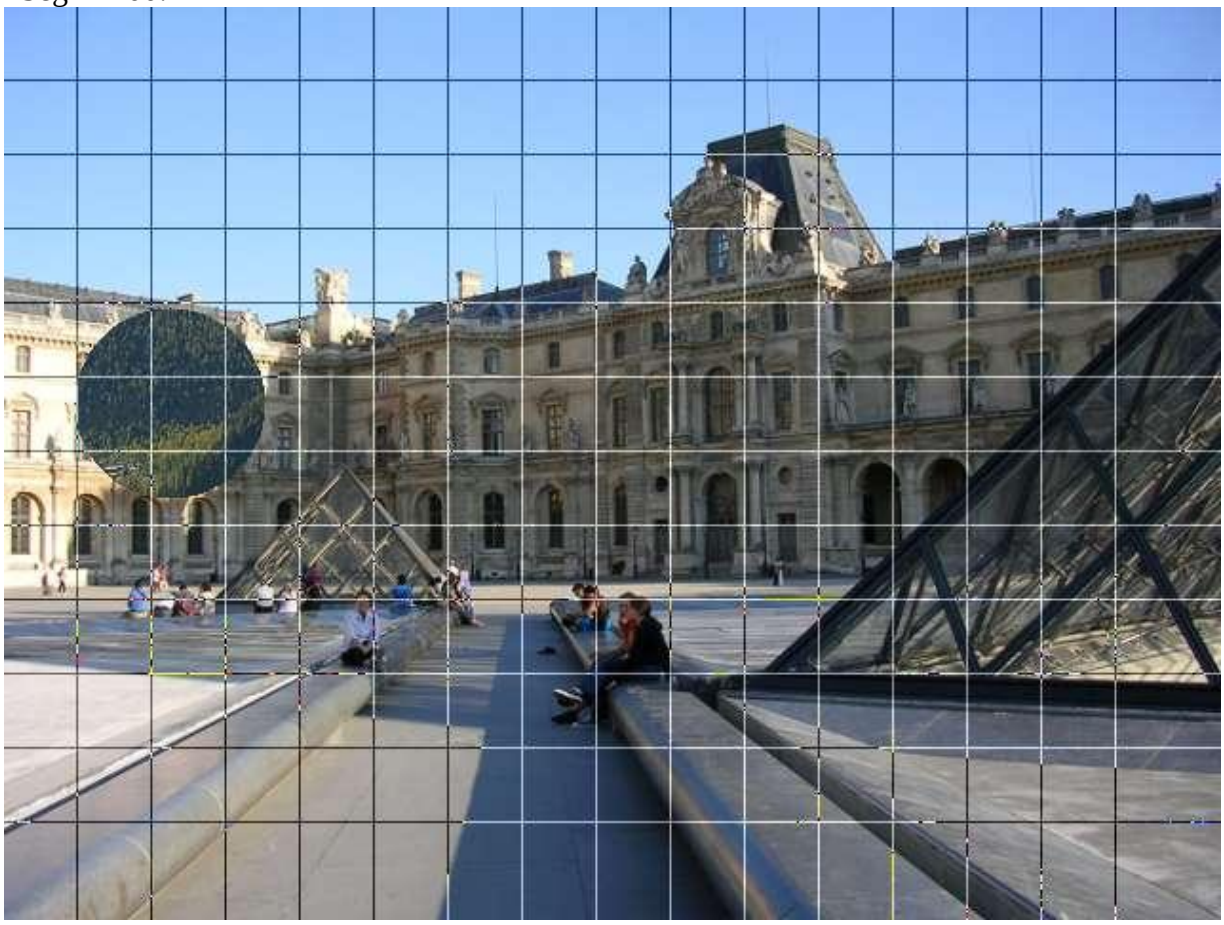
Results:
segm.txt - matrix of segmentation by squares

Пример. Рассмотрим картинку

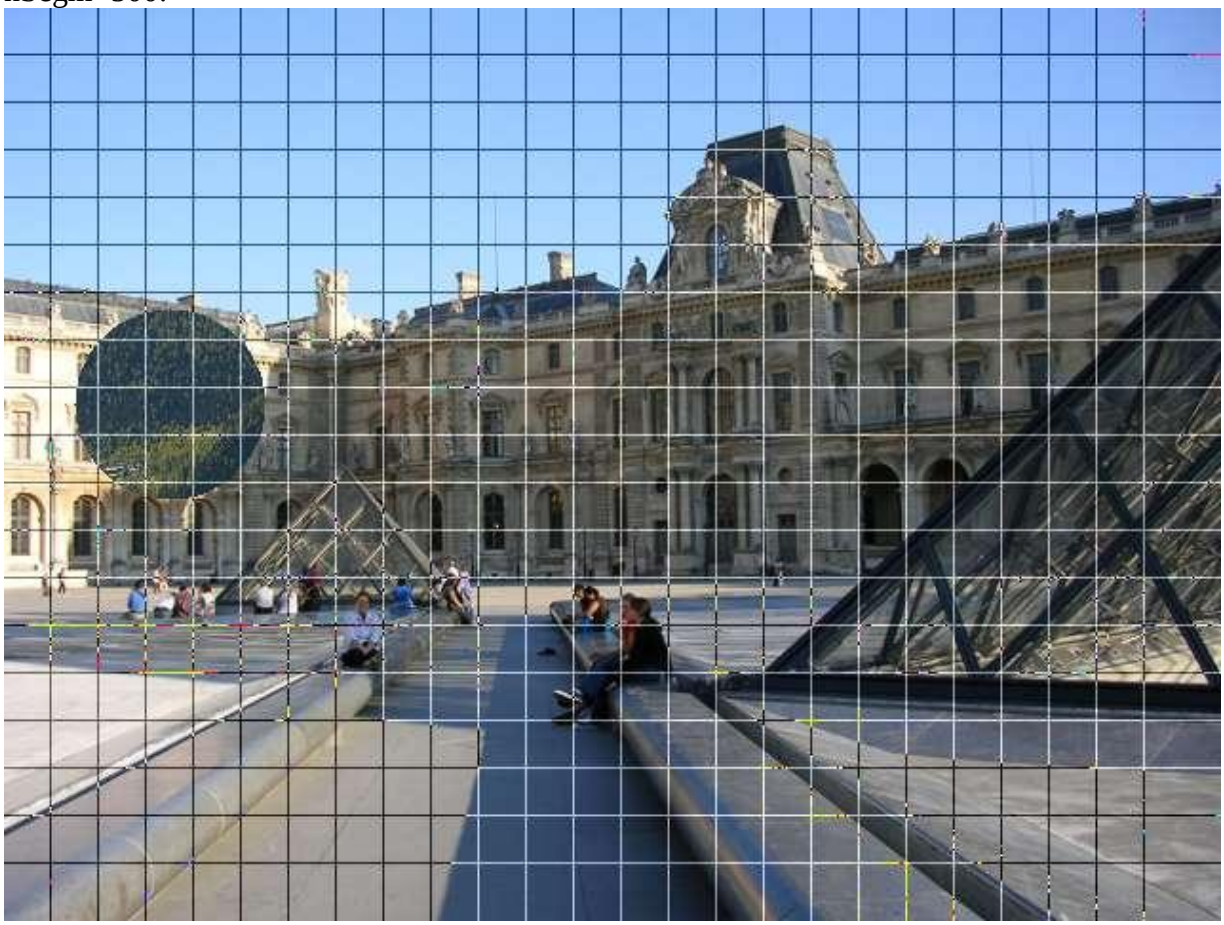
nSegm=100:



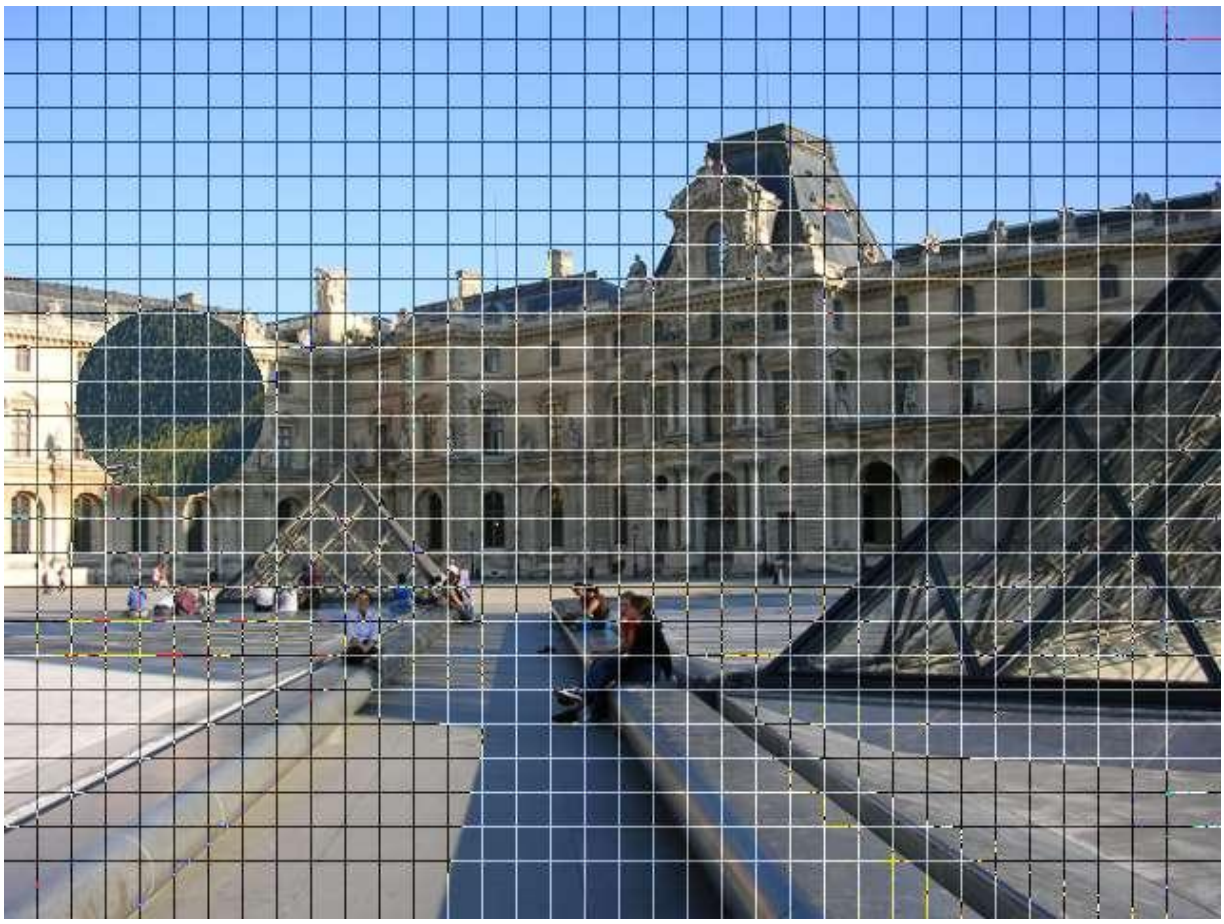
nSegm=200:



nSegm=500:



nSegm=1000:



4 Проверка качества сегментации

Рассмотрим программу Segm_test.exe, позволяющую тестировать различные программы сегментации.

Программа работает в двух режимах:

- Построение тестовых изображений
- Нахождение качества сегментации

4.1 Построение тестовых изображений

Запуск:

Segm_test.exe –c srcPath resPath cnt [mx my]

где

srcPath – каталог с исходными bmp-24-файлами, из которых строятся тестовые файлы.
resPath – выходной каталог с результатами работы.

Из исходного каталога случайно выбираются cnt пар, из второго изображения пары вырезается заданная фигура (круг пока) в случайном месте и вклеивается в случайное место в первом изображении из пары. Если заданы два дополнительных параметра

Полученное склеенное изображение записывается в каталог resPath под именем nnnn.bmp, где nnnn – номер пары. В этом же каталоге строится текстовый файл-описание, со строчками вида:

nnnn.bmp C mmmm xxxx yyyy

где

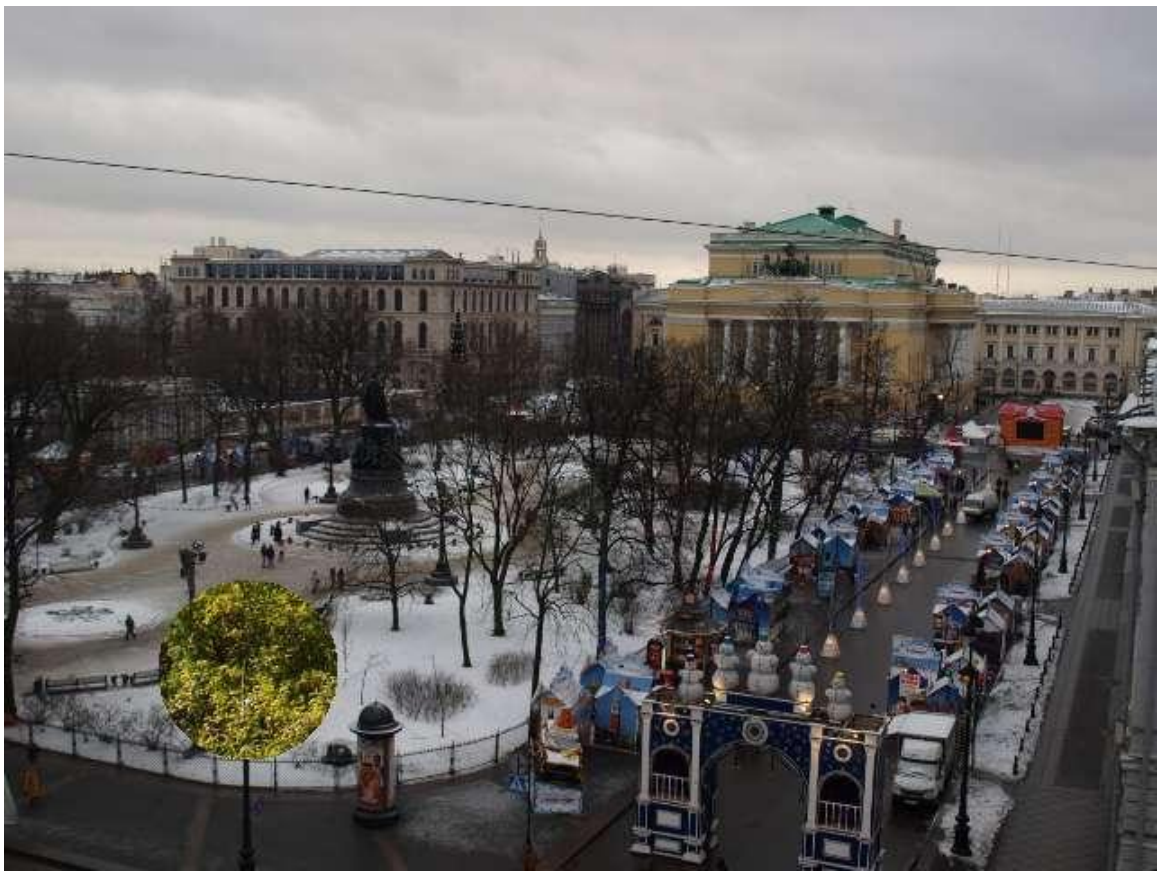
nnnn.bmp	- имя файла,
C	- признак обрабатываемой фигуры (circle),
mmmm	- количество точек в фигуре,
xxxx	- x-координата центра вклеенной фигуры на изображении,
yyyy	- y-координата центра вклеенной фигуры на изображении.

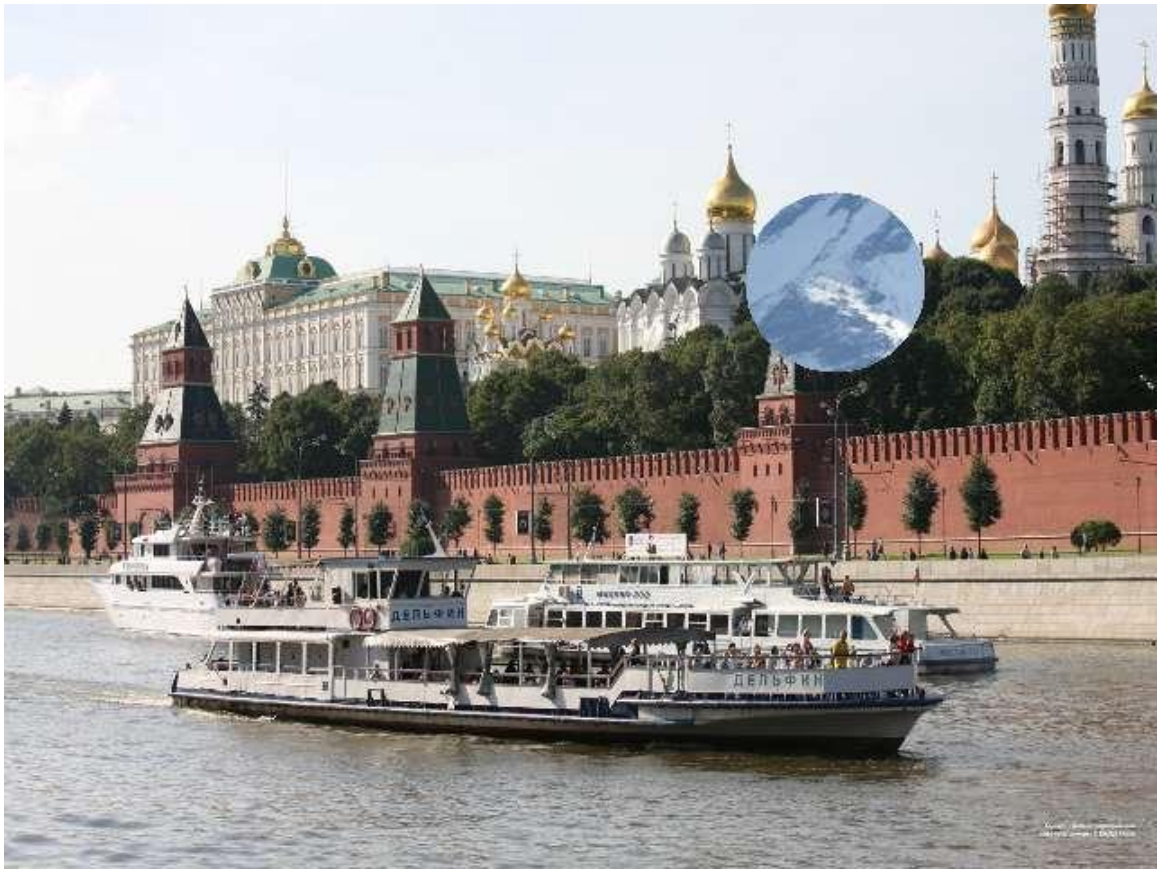
Замечание. В предложенный выше алгоритм были внесены некоторые изменения. Если его использовать напрямую, по получающиеся «врезанные» круги оказываются в среднем чересчур резкими, то есть слишком легкими для сегментации. Поэтому при выборе случайного места из которого вырезается фигура и случайного же места, куда она вставляется, отдается предпочтение точкам, у которых средняя длина цветового градиента в окрестности выше (функция goodPoint). На глаз фигуры все равно остаются очень заметными, но чуть меньше.

Вот примеры полученных изображений:


11 18


04.26.2005





и полученного файла с описаниями:

```
FileName C mmmm xxxx yyyy
0000.bmp C 7843   87  208
0001.bmp C 7843  185  334
```

```
0002.bmp C 7843  487  396
0003.bmp C 7843  381  386
0004.bmp C 7843   65  362
0005.bmp C 7843  263  292
0006.bmp C 7843  573  162
0007.bmp C 7843  181  142
0008.bmp C 7843  105  170
0009.bmp C 7843  135  368
0010.bmp C 7843  159  324
0011.bmp C 7843  105  250
0012.bmp C 7843  101  278
0013.bmp C 7843  461  154
0014.bmp C 7843   99  424
0015.bmp C 7843  357  126
0016.bmp C 7843   55  424
```
…

Качество сегментации «разбиением на квадраты» для примера, приведенного в п.2.2 такое:

nSegm	Средний размер сегмента	Количество «плохих» точек	Качество
100	3491	3975	49.32%
200	1600	3148	59.86%
500	647	2201	71.94%
1000	338	1321	83.16%

4.2 Требования к сегментирующей программе

Сегментирующая программа (exe- или bat-файл) должна воспринимать 3 аргумента (3-ий может игнорироваться):

zzz.exe src.bmp segm.txt nSegm

где

src.bmp – bmp-24-файл, подлежащий сегментации,
segm.txt – текстовый файл с результатами сегментации в формате:

mx my комментарий
s00 s10 …
s01 s11 …
…

например, при сегментации файла размером 4*3:

```
4 3 ; в первой строке файла заданы размеры матрицы по x (4) и по y (3)
1 2 1 1
2 2 1 1
2 2 2 3
```

где цифры в теле матрицы означают номер сегмента (>=0), к которому относится каждая точка. Сегменты не предполагаются связными и их номера не обязаны идти подряд. Никакое упорядочение сегментов также не требуется.

nSegm – предполагаемое количество сегментов. По возможности, программа должна ориентироваться на это число (если оно задано), если же программа не имеет такой возможности (алгоритм сегментации жестко задан), то оно может его просто игнорировать.

Если программа имеет другой интерфейс, то следует создать bat-файл, который приняв указанные выше параметры, передаст их программе в нужном ей виде.

4.3 Проверка эффективности программы

Запуск:

Segm_test.exe –t resPath progName nSegm prot.txt

где

resPath – каталог с искуственными изображениями (с вклеенной фигурой),
progName – исследуемая программа сегментации (exe- или bat-файл),
nSegm – рекомендуемое количество сегментов,
prot.txt – выходной текстовый файл с результатами работы.

Строки файла prot.txt имеют вид:

nnnn.bmp C mmmm xxxx yyyy nSegm qq.qq

где

nnnn.bmp	– имя файла,
C	– признак обрабатываемой фигуры (circle),
mmmm	– количество точек в фигуре,
xxxx	– x-координата центра вклеенной фигуры на изображении,

yyyy – y-координата центра вклеенной фигуры на изображении,
nSegm – количество сегментов,
dd.dd – качество сегментации (в %).

В конец этого же файла записывается небольшая таблица с общим итогом проверки качества сегментации: для каждого количества сегментов указывается среднее значение качества. Каждая строка таблицы имеет следующий вид:

nSegm_min nSegm_max dd.dd_mid

где

nSegm_min – минимальное количество сегментов в интервале,
nSegm_max – максимальное количество сегментов в интервале (превышает nSegm_min не более, чем на 30%),
dd.dd_mid – среднее качество сегментации для интервала (в %).

4.4 Пример: качество сегментирования на квадраты

Проверим работу на примера тривиального сегментирования – «на квадраты» (см.выше).

Строим 100 файлов размера 640*480 со вклеенным кругом диаметра 100:

Segm_test.exe -C C:\00\100Photo\bmp\ C:\00\100Photo\bmp_res\ 100 640 480

Для построенных файлов находим качество сегментации с помощью программы сегментирования на 200..5000 квадратов:

```
Segm_test.exe -T C:\00\100Photo\bmp_res\ Segm_by_squares.exe 200
Segm_test.exe -T C:\00\100Photo\bmp_res\ Segm_by_squares.exe 500
Segm_test.exe -T C:\00\100Photo\bmp_res\ Segm_by_squares.exe 1000
Segm_test.exe -T C:\00\100Photo\bmp_res\ Segm_by_squares.exe 2000
Segm_test.exe -T C:\00\100Photo\bmp_res\ Segm_by_squares.exe 5000
```

Вот полученный протокол работы:

```
N=  100; SZ= 3491.00 +-   0.00; eff= 42.43 +-11.44; prog=Segm_by_squares.exe
N=  200; SZ= 1600.00 +-   0.00; eff= 61.02 +- 7.21; prog=Segm_by_squares.exe
N=  500; SZ=  647.00 +-   0.00; eff= 75.37 +- 3.54; prog=Segm_by_squares.exe
N= 1000; SZ=  338.00 +-   0.00; eff= 82.54 +- 2.26; prog=Segm_by_squares.exe
N= 2000; SZ=  145.00 +-   0.00; eff= 88.39 +- 1.52; prog=Segm_by_squares.exe
N= 5000; SZ=   64.00 +-   0.00; eff= 92.34 +- 0.82; prog=Segm_by_squares.exe
```

Или, в табличном виде:

Запрашиваемое количество сегментов	Средний размер сегмента	Эффективности
100	3491	42.43 ±11.44
200	1600	61.02 ± 7.21
500	647	75.37 ± 3.54
1000	338	82.54 ± 2.26
2000	145	88.39 ± 1.52
5000	64	92.34 ± 0.82

5 Чтение файла вида «BMP-24»

Файлы указанного типа являются простейшими файлами из всех графических форматов. Они содержат в несжатом виде полную информацию о цветах каждой точки в формате RGB-24. Строчки в bmp-файле хранятся в обратном порядке, то есть в начале идет самая нижняя строка изображения. Внутри строки байты имеют порядок

BGR BGR …

Кроме того, длина каждой строки выровнена до кратного четырех. То есть для изображения шириной 3 точки длина строки равна 12 байт (3 точки по 3 байта, всего 9 байт, затем увеличиваем длину строки до кратного 4).

В начале bmp-файла идет заголовок следующей структуры:

```
typedef struct {
    WORD   bfType;        // 0x4d42 | 0x4349 | 0x5450
    int    bfSize;        // размер файла
    int    bfReserved;    // 0
    int    bfOffBits;     // смещение до поля данных,
                          // обычно 54 = 16 + biSize
    int    biSize;        // размер струкуры в байтах:
                          // 40(BITMAPINFOHEADER) или 108(BITMAPV4HEADER)
                          // или 124(BITMAPV5HEADER)
    int    biWidth;       // ширина в точках
    int    biHeight;      // высота в точках
    WORD   biPlanes;      // всегда должно быть 1
    WORD   biBitCount;    // 0 | 1 | 4 | 8 | 16 | 24 | 32
    int    biCompression; // BI_RGB | BI_RLE8 | BI_RLE4 |
                          // BI_BITFIELDS | BI_JPEG | BI_PNG
                          // реально используется лишь BI_RGB
    int    biSizeImage;   // Количество байт в поле данных
                          // Обычно устанавливается в 0
    int    biXPelsPerMeter;// горизонтальное разрешение, точек на дюйм
    int    biYPelsPerMeter;// вертикальное разрешение, точек на дюйм
    int    biClrUsed;     // Количество используемых цветов
                          // (если есть таблица цветов)
    int    biClrImportant; // Количество существенных цветов.
                          // Можно считать, просто 0
} BMPheader;
```

Для чтения файла можно применить такой код:

```
FILE *f = fopen( fname, "rb" );
if( !f ) { setSize(0,0); return -1; }
BMPheader bh;      // File header sizeof(BMPheader) = 56
size_t res;

// читаем заголовок
res = fread( &bh, 1, sizeof(BMPheader), f );
if( res != sizeof(BMPheader) ) { fclose(f); return -1; }

// проверяем сигнатуру
if( bh.bfType!=0x4d42 && bh.bfType!=0x4349 && bh.bfType!=0x5450 ) {
    fclose(f); return -1;
}

// проверка размера файла
fseek( f, 0, SEEK_END);
int filesize = ftell(f);
// восстановим указатель в файле:
fseek( f, sizeof(BMPheader), SEEK_SET);
// проверим условия
if( bh.bfSize != filesize ||
    bh.bfReserved != 0    ||
    bh.biPlanes   != 1    ||
   (bh.biSize!=40 && bh.biSize!=108 && bh.biSize!=124)||
    bh.bfOffBits != 14+bh.biSize ||
    bh.biWidth <1 || bh.biWidth >10000 ||
    bh.biHeight<1 || bh.biHeight>10000 ||
    bh.biBitCount    != 24 ||     // рассматриваем только полноцветные изображения
    bh.biCompression !=  0        // пока рассматриваем только несжатие изображения
) {
    fclose(f);
    setSize(0,0);
    return -1;
}
// Заголовок прочитан и проверен, тип - верный (BGR-24), размеры (mx,my) найдены
int mx_ = bh.biWidth;
int my_ = bh.biHeight;
int mx3 = (3*mx_+3)&(-4);  // Compute row width in file, including padding
                           // to 4-byte boundary
unsigned char *tmp_buf = new unsigned  char[mx3*my_]; // читаем данные
res = fread( tmp_buf, 1, mx3*my_, f);
if( (int)res != mx3*my_ ) { delete []tmp_buf; fclose(f); setSize(0,0); return -1; }
// данные прочитаны
fclose(f);
```