РАЗРАБОТКА ПРОГРАММЫ-ДЕТЕКТОРА ИЗОБРАЖЕНИЙ ВНУТРИ ФАЙЛОВЫХ КОНТЕЙНЕРОВ

ст гр. ВМКС-МЭИ-11 Шаляпин А.В.
КГТУ им. Раззакова

Целью данной работы является создание программы, способной определять наличие изображений внутри файлового контейнера

The aim of this work is to create a program that can detect the presence of images inside the container file

За основу для исследования был взят формат изображений TGA. Далее будут рассмотрены некоторые этапы разработки и теоретические сведения, на основе которых строился анализ.

Вообще детекторы базируются на принципе поиска определенных сигнатур, присущих файлам. В данном конкретном случае формат TGA имеет несколько размытую структуру, и там нет каких-то однозначно определяющих этот формат сигнатур.

Структура формата:

- Заголовок
- Изображение/карта цветов
- Зона разработчика
- Зона расширения
- Подвал

Последние три зоны были добавлены во второй версии формата. Может содержать полутоновые, палитровые и полноцветные изображения. Поддерживается сжатие RLE и добавление различных метаданных.

Разработка была разбита на 2 этапа:

1. Разработка детектора
2. Фильтрация полученных данных

Анализ данных начинается с заголовка. Для того, чтобы построить детектор изображений требуется определить правильную структуру заголовка и обозначить допустимые значения.

Рассмотрим структуру заголовка:

```
THeader = packed record
  IDLen         : Byte;
  ColorMapType  : Byte;
```

```
ImageType       : Byte;
ColorMapIndex   : Word;
ColorMapLen     : Word;
ColorMapBitCount : Byte;
ImagePositionX  : Word;
ImagePositionY  : Word;
ImageWidth      : Word;
ImageHeight     : Word;
ImageBitCount   : Byte;
ImageFlags      : Byte;
end;
```

IDLen - Длина идентификатора, который может располагаться после заголовка. Диапазон значений 0..255.

ColorMapType – Тип карты цветов (палитры). Возможные значения: 0 – нет карты цветов, 1 – есть карта цветов.

ImageType – Тип изображения. Возможные значения приведены в таблице 1.

Таблица 1. Типы поддерживаемых изображений

Значение	Описание	Используется карта цветов	Используется сжатие
0	Нет изображения	Нет	Нет
1	Изображение с палитрой	Да	Нет
2	TrueColor изображение	Нет	Нет
3	Монохромное изображение	Нет	Нет
9	Изображение с палитрой	Да	Да
10	TrueColor изображение	Нет	Да
11	Монохромное изображение	Нет	Да

ColorMapIndex - индекс первого элемента карты цветов. Не должен быть больше, чем ColorMapLen.

ColorMapLen – Количество элементов в карте цветов.

ColorMapBitCount – Количество бит, выделяемых под элемент карты цветов. Обычно используются значения 15, 16, 24, 32.

При ColorMapType = 0, три вышеуказанных значения должны равняться нулю.

ImagePositionX - Определяет абсолютную координату левого нижнего угла изображения по горизонтали.

ImagePositionY - Определяет абсолютную координату левого нижнего угла изображения по вертикали.

ImageWidth - Ширина изображения в пикселях. Диапазон значений - 1..32767.

ImageHeight - Высота изображения в пикселях. Диапазон значений - 1..32767.

ImageBitCount – Определяет количество бит на пиксель. Для детектора были выбраны значения 8, 16, 24, 32.

ImageFlags - Биты 3-0 этого поля определяют количество атрибутивных бит на пиксель. Биты 5 и 4 определяют порядок передачи пиксельных данных из файла на экран. Бит 4 устанавливается для порядка «слева направо», бит 5 — для порядка «сверху вниз». Биты 7 и 6 в целях совместимости должны быть установлены в 0.

На основе этих данных была написана простейшая функция определения файла по заголовку. Но у нее была низкая точность определения, и большое количество ложноположительных нахождений.

```
function IsValidHeader(Phdr: PHeader): Boolean;
begin
 Result:= False;
 if (Phdr^.ColorMapType in [0, 1]) then
 if (PHdr^.ImageType in [1, 2, 3, 9, 10, 11]) then
begin
  if ((PHdr.ImageFlags and $C0) <> 0) then Exit;
  if (PHdr^.ImageType in [1, 9]
)and(Phdr^.ColorMapType=0) then Exit;
  If (PHdr^.ImageType in [2, 3, 10,
11])and(PHdr^.ColorMapType=1) then Exit;
  if
(PHdr^.ImageWidth=0)or(PHdr^.ImageHeight=0)
then Exit;
  if not (Phdr^.ImageBitCount in [8, 16, 24, 32])
then Exit;
  if (PHdr^.ColorMapType=0) then begin
   if (PHdr^.ColorMapOffset<>0) then Exit;
   if (PHdr^.ColorMapLen<>0) then Exit;
   if (PHdr^.ColorMapBitCount<>0) then Exit;
  end;
  Result:= True;
 end;
end;
```

Чтобы избавиться от ложноположительных нахождений, исходя из вышеописанных данных, в эту функцию были добавлены элементы фильтрации найденных вхождений. Собственно на этапе анализа заголовка отсеиваются до 70 % ложноположительных нахождений внутри файлового контейнера.

Одним из этапов фильтрации, стала реализация алгоритма RLE и обработка им сжатых изображений (типы 9, 10, 11).

RLE (Run-Length Encoding) – это алгоритм кодирующий последовательности одинаковых символов.

Работа алгоритма заключается в том, чтобы заменить последовательность повторяющихся символов на связку длина-значение. В данной реализации алгоритма RLE в поле длина

используется 7 бит для хранения длины последовательности и 1 старший бит используется как флаг, определяющий как интерпретировать данную связку. Если этот флаг равен 1, то последовательность закодирована, если флаг равен 0, то последовательность просто скопирована как есть. А 7 бит, отведенных для длины последовательности, интерпретируются так, что 0 обозначает 1 символ в последовательности. Т.е. максимальная длина последовательности равна 128 байтам.

Т.к. алгоритм имеет довольно определенную структуру, и общий объем данных изображения должен быть равен ImageWidth*ImageHeight*BytesPerPixel, то при обработке некорректного сжатого потока произойдет ошибка, что позволяет прервать дальнейший анализ данного вхождения.

Следующим этапом фильтрации стала реализация алгоритма определения уровня шума. Существуют различные алгоритмы определения уровня шума в цифровом изображении, к примеру, блочные методы и методы на основе вейвлет-преобразования, но у каждого из них есть свои недостатки. После многочисленных поисков был выбран алгоритм гармонического анализа.

Гармонические методы[3] основаны на анализе результатов преобразования Фурье. Один из подобных методов основан на доказанном в ходе экспериментов утверждении, что гармоники сигнала близкого к шумовой модели «белый шум» имеют распределение Рэлея. При этом математическое ожидание данного распределения характеризует среднеквадратичное значение шума по формуле (1).

$$Nrms = Mr\sqrt{\frac{n*m-1}{(n*m)^2}} \quad , (1)$$

где Mr – математическое ожидание распределения, n и m – размеры рабочей области по горизонтали и вертикали. Чтобы реализовать данный метод производится двухмерное быстрое преобразование Фурье рабочей области изображения, размеры которой выбраны из условия равенства 2^i, где i – целое положительное число. Далее результаты БПФ подвергаются квантованию, и строится гистограмма. Шаг квантования q выбирается в зависимости от требуемой точности по формуле (2). Затем производится поиск моды распределения. В случае распределения Рэлея математическое ожидание M связано с модой по формуле (3).

$$q = p\sqrt{\frac{(n*m)^2}{n*m-1}} \quad , (2)$$

$$M = m_d\sqrt{\frac{\pi}{2}}, (3)$$

где p – задаваемый параметр точности (минимальный квант для Nrms), n и m - размеры рабочей области по горизонтали и вертикали, m_d – мода (наиболее часто встречающееся значение) распределения. Далее по формуле (4), полученной из формулы (1) с учётом квантования, получаем среднеквадратичное значение шума.

$$Nrms = M * p, (4)$$

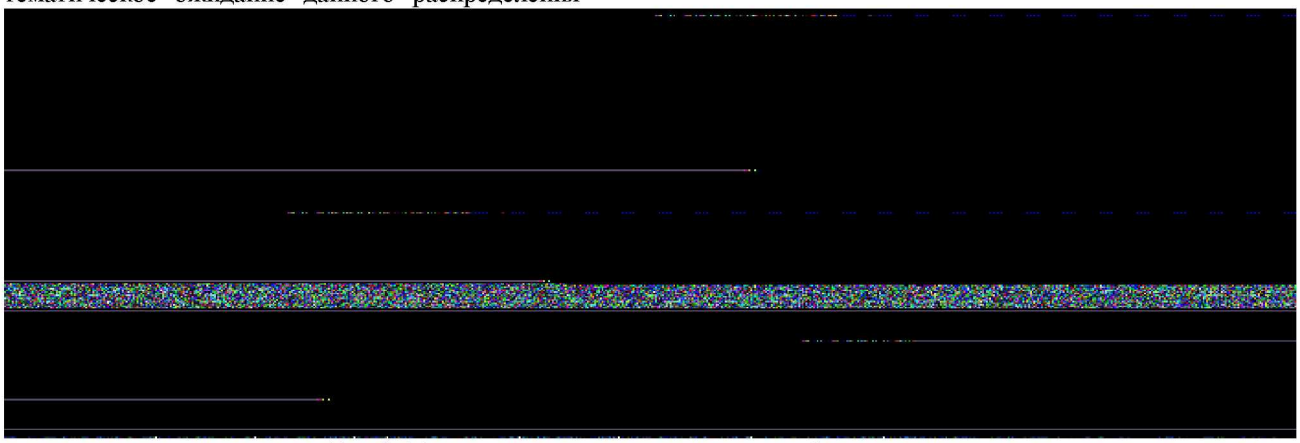

Рис 1. Пример ложноположительного изображения.

Данный алгоритм позволил с довольно высокой точностью отсеивать оставшиеся ложноположительные изображения.

Данные три этапа фильтрации позволяют отсеять до 95% ложноположительных изображений. Оставшиеся 5 % приходятся на изображения, имеющие дополнительные зоны, и для них необходимо использовать алгоритмы фильтрации данных зон.

В итоге получилась программа, способная с высокой точностью определять наличие изображений формата TGA внутри файлового контейнера. Тестирование программы проводилось как на чистых, так и на смешанных данных.

Литература

1. Wikipedia – свободная энциклопедия. http://wikipedia.org
2. Truevision Inc. – Спецификация формата Truevision TGA
3. Лапшенков Е.М – Не эталонная оценка уровня шума цифрового изображения на основе гармонического анализа. (Компьютерная оптика, 2012, том 36, № 3)