

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ имени академика С.П. КОРОЛЕВА (национальный
исследовательский университет)»

ФАКУЛЬТЕТ ИНФОРМАТИКИ

КАФЕДРА ТЕХНИЧЕСКОЙ КИБЕРНЕТИКИ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ЛАБОРАТОРНОЙ РАБОТЕ

ТЕМА: «Пороговая обработка изображения методом ОТСУ»

Выполнили студенты

Проценко В.И.
Булдыгин Е.Ю.

Группа

6128

27 марта 2013 г.

Содержание

1. Пороговая бинаризация методом Оtsu	2
2. Настройка оборудования	3
2.1. Настройка одного узла	3
2.2. Настройка двухузлового кластера	7
3. Описание проекта	12
3.1. Описание программы	12
3.2. Описание build.xml	14
3.3. Компиляция и запуск под Windows	17
3.4. Компиляция и запуск под Linux	19
3.5. Установка ant под Linux	19
4. Описание среды	20
4.1. Классификация версий Apache Hadoop	20
4.2. Версии использованного программного обеспечения	20
4.3. Известные проблемы	20
Приложение А Исходный код функций	21

1. Пороговая бинаризация методом Отсу

Бинарные изображения можно получать из полутоновых изображений посредством пороговой бинаризации. При выполнении этой операции часть пикселей выбирается в качестве пикселей переднего плана, представляющих объекты интереса, а остальные — в качестве фоновых пикселей. Зная распределение значений яркости на данном изображении, некоторые значения можно выбрать в качестве порогов, разделяющих пиксели на группы. В простейшем случае выбирается одно пороговое значение t . Все пиксели с яркостью больше или равной t становятся белыми, а остальные — чёрными.

Для автоматического выбора порога бинаризации было разработано много различных методов. В данной работе реализован метод Отсу. Выбор порога в этом методе основан на минимизации внутригрупповой дисперсии двух групп пикселей, разделяемых оператором пороговой бинаризации. Отсу показал, что минимум внутригрупповой дисперсии обеспечивает максимум межгрупповой дисперсии (квадрат разности между средними значениями групп). Интерпретируя гистограмму как частоту появления конкретного значения яркости, можно записать две величины:

$$p_1(t) = \frac{1}{|H|} \sum_{i=0}^t H_i, \quad p_2(t) = 1 - p_1(t) = \frac{1}{|H|} \sum_{i=t+1}^{255} H_i, \quad |H| = \sum_{i=0}^{255} H_i,$$

где $p_i(t)$ — вероятность класса, получаемая при бинаризации с порогом t . Значение порога в методе Отсу определяется как:

$$t = \arg \max p_1(t)(1 - p_1(t))(\mu_1(t) - \mu_2(t))^2,$$
$$\mu_1(t) = \frac{1}{\sum_{i=0}^t H_i} \sum_{i=0}^t iH_i, \quad \mu_2(t) = \frac{1}{\sum_{i=t+1}^{255} H_i} \sum_{i=t+1}^{255} iH_i.$$

Соответственно алгоритм заключается в простом переборе возможных пороговых значений t с выбором наилучшего, при этом все необходимые величины могут вычисляться рекуррентно.

Таким образом, для реализации пороговой обработки методом Отсу необходимо провести вычисления в три этапа:

- 1) сбор гистограммы яркости;
- 2) расчёт порогового значения t ;
- 3) пороговая бинаризация с полученным порогом.

2. Настройка оборудования

2.1. Настройка одного узла

Для настройки одиночного узла не требуется прилагать огромных усилий. Подробное описание действий по устройству кластера на базе Hadoop можно найти по ссылкам:

<http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/>

<http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/>

Дальнейшее описание представляет собой краткий обзор этих статей. Для установки Hadoop необходимо:

- 1) Компьютер с Ubuntu 12.04 LTS;
- 2) Дистрибутив Hadoop 1.0.4;
- 3) Свежая версия Java SDK (в данной работе был использован Java SE 1.7.0_17);
- 4) Настроенная SSH.

В случае установки Hadoop на чистую систему, необходимо установить Java SDK с сайта Oracle, так как предустановленный OpenJDK может не поддерживать некоторые элементы Hadoop. Для начала удаляется OpenJDK и всё что с ним связано:

```
$ sudo apt-get purge openjdk*
```

Необходимый Java SDK можно скачать с сайта (для Ubuntu выбираем .tar.gz):

<http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html>

Далее распаковываем скачанный архив и перемещаем содержимое в каталог /usr/lib/jvm/:

```
$ tar -xvf $HOME/Downloads/jdk-7u4-linux-x64.tar.gz
$ mv $HOME/jdk1.7.0_04/ /usr/lib/jvm/
```

Hadoop использует протокол SSH для управления узлами, поэтому необходимо сначала установить SSH-сервер:

```
$ sudo apt-get install ssh
```

Для пользователя создаётся ключ для SSH-соединения (в данном случае пользователь hduser), который соответствует пустому паролю:

```
$ su - hduser
$ ssh-keygen -t rsa -P ""
```

Затем открываем доступ к локальной машине с созданным ключом:

```
$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

Для проверки запускаем соединение с localhost, при этом он будет помещён в список известных хостов hduser:

```
$ ssh localhost
```

Следующим шагом является отключение IPv6. Для этого открываем файл /etc/sysctl.conf и добавляем в конец файла строки:

```
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

После сохранения, можно проверить результат командой:

```
$ cat /proc/sys/net/ipv6/conf/all/disable_ipv6
```

Если всё было правильно сделано, то увидим 1.

Теперь можно приступить к установке Hadoop. Для этого скачиваем дистрибутив с официального сайта (<http://hadoop.apache.org/>) и извлекаем содержимое в симпатичный нам каталог (например, в /usr/local/hadoop), при этом не стоит забывать о правах доступа (владельцем должен быть hduser и группа hadoop). Процесс можно выполнить командами:

```
$ sudo tar xzf $HOME/Downloads/hadoop-1.0.4.tar.gz
$ sudo mv $HOME/hadoop-1.0.4 /usr/local/hadoop
$ sudo chown -R hduser:hadoop /usr/local/hadoop
```

После этого необходимо обновить \$HOME/.bashrc в соответствии со следующими строками:

```
# Установка переменной, связанной с Hadoop
export HADOOP_HOME=/usr/local/hadoop

# Установка местоположения Java SDK
export JAVA_HOME=/usr/lib/jvm/java-6-sun

# Некоторые удобные псевдонимы и функции
# для работы Hadoop, связанной с командами
unalias fs &> /dev/null
alias fs="hadoop fs"
unalias hls &> /dev/null
alias hls="fs -ls"

# If you have LZO compression enabled in your
# Hadoop cluster and compress job outputs with
# LZOP (not covered in this tutorial):
# C      onveniently inspect an LZOP compressed
# file from the command
# line; run via:
#
# $ lzohead /hdfs/path/to/lzop/compressed/file.lzo
#
# Requires installed 'lzop' command.
#
lzohead () {
    hadoop fs -cat $1 | lzop -dc | head -1000 | less
}

# Добавляем Hadoop bin/ в PATH
export PATH=$PATH:$HADOOP_HOME/bin
```

Теперь можно заняться конфигурированием одиночного узла. Для начала открываем файл /usr/local/hadoop/conf/hadoop-env.sh и заменяем строчку вида

```
# export JAVA_HOME=/usr/lib/j2sdk1.5-sun
```

на строчку с соответствующим указанием директории с Java SDK:

```
export JAVA_HOME=/usr/lib/jvm/java1.7.0_17
```

Можно создать отдельный каталог для служебного использования распределённой файловой системой (в нашем случае это будет /app/hadoop/tmp, не забываем при этом про права доступа):

```
$ sudo mkdir -p /app/hadoop/tmp
$ sudo chown hduser:hadoop /app/hadoop/tmp
```

Открываем conf/core-site.xml в директории с Hadoop и между тегами <configuration> ... </configuration> добавляем следующие строки:

```
<property>
  <name>hadoop.tmp.dir</name>
  <value>/app/hadoop/tmp</value>
  <description>
    A base for other temporary directories.
  </description>
</property>

<property>
  <name>fs.default.name</name>
  <value>hdfs://localhost:54310</value>
  <description>The name of the default file system.
    A URI whose scheme and authority determine the
    FileSystem implementation. The uri's scheme
    determines the config property (fs.SCHEME.impl)
    naming the FileSystem implementation class.
    The uri's authority is used to determine the host,
    port, etc. for a filesystem.
  </description>
</property>
```

В файле conf/mapred-site.xml между тегами <configuration> ... </configuration> добавляем:

```
<property>
  <name>mapred.job.tracker</name>
  <value>localhost:54311</value>
  <description>The host and port that the MapReduce
    job tracker runs at. If "local", then jobs are
    run in-process as a single map and reduce task.
  </description>
</property>
```

В файле conf/hdfs-site.xml между тегами <configuration> ... </configuration> добавляем (пока что у нас только один узел — фактор репликации равен единице):

```
<property>
  <name>dfs.replication</name>
  <value>1</value>
  <description>Default block replication.
    The actual number of replications can
    be specified when the file is created.
    The default is used if replication is
    not specified in create time.
  </description>
</property>
```

Теперь всё готово. Перед запуском форматируем HDFS командой из терминала:

```
$ /usr/local/hadoop/bin/hadoop namenode -format
```

Если всё сконфигурировано правильно, то в терминале появятся примерно такие сообщения:

```
hduser@eugeneold-desktop:~$ /usr/local/hadoop/bin/hadoop namenode -format
Warning: $HADOOP_HOME is deprecated.

13/03/26 07:18:07 INFO namenode.NameNode: STARTUP_MSG:
/*****
STARTUP_MSG: Starting NameNode
STARTUP_MSG: host = eugeneold-desktop/127.0.1.1
STARTUP_MSG: args = [-format]
STARTUP_MSG: version = 1.0.4
STARTUP_MSG: build = https://svn.apache.org/repos/asf/hadoop/common/branches/b
ranch-1.0 -r 1393290; compiled by 'hortonfo' on Wed Oct 3 05:13:58 UTC 2012
*****/
Re-format filesystem in /app/hadoop/tmp/dfs/name ? (Y or N) Y
13/03/26 07:18:14 INFO util.GSet: VM type = 32-bit
13/03/26 07:18:14 INFO util.GSet: 2% max memory = 19.33375 MB
13/03/26 07:18:14 INFO util.GSet: capacity = 2^22 = 4194304 entries
13/03/26 07:18:14 INFO util.GSet: recommended=4194304, actual=4194304
13/03/26 07:18:15 INFO namenode.FSNamesystem: fsOwner=hduser
13/03/26 07:18:15 INFO namenode.FSNamesystem: supergroup=supergroup
13/03/26 07:18:15 INFO namenode.FSNamesystem: isPermissionEnabled=true
13/03/26 07:18:15 INFO namenode.FSNamesystem: dfs.block.invalidate.limit=100
13/03/26 07:18:15 INFO namenode.FSNamesystem: isAccessTokenEnabled=false accessK
eyUpdateInterval=0 min(s), accessTokenLifetime=0 min(s)
13/03/26 07:18:15 INFO namenode.NameNode: Caching file names occurring more than
10 times
13/03/26 07:18:15 INFO common.Storage: Image file of size 112 saved in 0 seconds
.
13/03/26 07:18:16 INFO common.Storage: Storage directory /app/hadoop/tmp/dfs/nam
e has been successfully formatted.
13/03/26 07:18:16 INFO namenode.NameNode: SHUTDOWN_MSG:
/*****
SHUTDOWN_MSG: Shutting down NameNode at eugeneold-desktop/127.0.1.1
*****/
hduser@eugeneold-desktop:~$
```

Для запуска одноузлового кластера используется скрипт start-all.sh:

```
hduser@eugeneold-desktop:~$ /usr/local/hadoop/bin/start-all.sh
Warning: $HADOOP_HOME is deprecated.

starting namenode, logging to /usr/local/hadoop/libexec/./logs/hadoop-hduser-na
menode-eugeneold-desktop.out
localhost: starting datanode, logging to /usr/local/hadoop/libexec/./logs/hadoo
p-hduser-datanode-eugeneold-desktop.out
localhost: starting secondarynamenode, logging to /usr/local/hadoop/libexec/./l
ogs/hadoop-hduser-secondarynamenode-eugeneold-desktop.out
starting jobtracker, logging to /usr/local/hadoop/libexec/./logs/hadoop-hduser-
jobtracker-eugeneold-desktop.out
localhost: starting tasktracker, logging to /usr/local/hadoop/libexec/./logs/ha
doop-hduser-tasktracker-eugeneold-desktop.out
```

Остановка кластера осуществляется скриптом stop-all.sh:

```
hduser@eugeneold-desktop:~$ /usr/local/hadoop/bin/stop-all.sh
Warning: $HADOOP_HOME is deprecated.

stopping jobtracker
localhost: stopping tasktracker
stopping namenode
localhost: no datanode to stop
localhost: no secondarynamenode to stop
hduser@eugeneold-desktop:~$
```

2.2. Настройка двухузлового кластера

Если в локальной сети есть два одноузловых кластера, настроенных по предыдущему пункту, то для объединения их в один двухузловой. Если компьютеры не являются членами локальной сети, но есть доступ в Интернет, то можно воспользоваться средствами построения VPN, например Hamachi.

Для удобства работы вместо ip-адресов можно использовать символьные значения, для этого добавляем в файл /etc/hosts следующие строки (на обеих машинах):

```
192.168.0.1    master
192.168.0.2    slave
```

Примечание. Важно, чтобы имя компьютера совпадало с введёнными сокращениями. В ходе установки было выяснено, что помимо перечисленных строчек, в hosts slave узла необходимо также добавить пару «ip-адрес название удалённой машины», так как hadoop не мог разрешить имя компьютера и сопоставить ему адрес master узла.

Для корректного функционирования Hadoop необходимо обеспечить SSH-соединение без ввода пароля. Для этого выполняем команду копирования ключа (создан в предыдущем разделе):

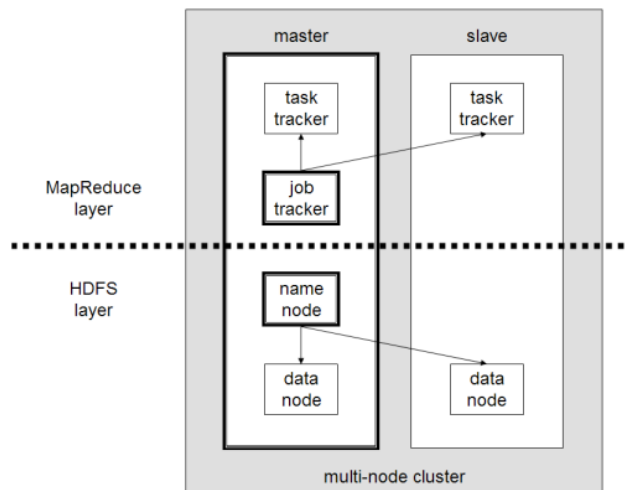
```
hduser@master:~$ ssh-copy-id -i $HOME/.ssh/id_rsa.pub hduser@slave
```

После этого проверяем соединения master-to-master и master-to-slave, при запросе подтверждаем продолжение соединения:

```
hduser@master:~$ ssh master
The authenticity of host 'master (192.168.0.1)' can't be established.
RSA key fingerprint is 3b:21:b3:c0:21:5c:7c:54:2f:1e:2d:96:79:eb:7f:95.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'master' (RSA) to the list of known hosts.
Linux master 2.6.20-16-386 #2 Thu Jun 7 20:16:13 UTC 2007 i686

hduser@master:~$ ssh slave
The authenticity of host 'slave (192.168.0.2)' can't be established.
RSA key fingerprint is 74:d7:61:86:db:86:8f:31:90:9c:68:b0:13:88:52:72.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'slave' (RSA) to the list of known hosts.
```

На рисунке представлено различие между master и slave. Помимо этого, на master предполагается запуск скриптов start-dfs.sh и start-mapred.sh, которые запускают необходимых демонов во вторичных узлах.



Для настройки узла-мастера необходимо добавить master в /conf/masters и пару строчек в /conf/slaves:

```
master
slave
```

На всех узлах производим следующие модификации /conf/*-site. Как и прежде, между тегами <configuration> ... </configuration> нужно обновить содержимое в соответствии со следующим текстом:

1) Для conf/core-site.xml на всех машинах:

```
<property>
  <name>fs.default.name</name>
  <value>hdfs://master:54310</value>
  <description>The name of the default file system.
  A URI whose scheme and authority determine
  the FileSystem implementation. The uri's scheme
  determines the config property (fs.SCHEME.impl)
  naming the FileSystem implementation class.
  The uri's authority is used to determine the host,
  port, etc. for a filesystem.
</description>
</property>
```

2) Для conf/mapred-site.xml на всех машинах:

```
<property>
  <name>mapred.job.tracker</name>
  <value>master:54311</value>
  <description>The host and port that the MapReduce
  job tracker runs at. If "local", then jobs are
  run in-process as a single map and reduce task.
</description>
</property>
```

3) Для conf/hdfs-site.xml на всех машинах:

```
<property>
  <name>dfs.replication</name>
```

```
<value>2</value>
<description>Default block replication.
The actual number of replications can be specified
when the file is created. The default is used if
replication is not specified in create time.
</description>
</property>
```

Теперь всё настроено, можно форматировать HDFS и запускать кластер. На master для форматирования файловой системы вводим команду:

```
hduser@master$ /usr/local/hadoop/bin/hadoop namenode -format
```

Запуск кластера проводится в два этапа:

- 1) Запуск файловой системы:

```
hduser@master$ /usr/local/hadoop/bin/start-dfs.sh
```

- 2) Запуск фоновых программ MapReduce (JobTracker на мастере и TaskTracker на обоих узлах):

```
hduser@master$ /usr/local/hadoop/bin/start-mapred.sh
```

Успешность запуска можно проверить по соответствующим логам в logs/ в директории с Hadoop или же запустить jps:

```
hduser@master$ $JAVA_HOME/bin/jps
16017 Jps
14799 NameNode
15686 TaskTracker
14880 DataNode
15596 JobTracker
14977 SecondaryNameNode
```

На компьютере slave:

```
hduser@slave$ $JAVA_HOME/bin/jps
15183 DataNode
15897 TaskTracker
16284 Jps
```

Остановка кластера также происходит в два этапа:

- 1) Остановка фоновых программ MapReduce:

```
hduser@master$ /usr/local/hadoop/bin/stop-mapred.sh
```

- 2) Остановка файловой системы:

```
hduser@master$ /usr/local/hadoop/bin/stop-dfs.sh
```

Если сделать всё правильно, то получим примерный вывод (как легко заметить, вторичный узел в данном запуске называется `old_slave`, если всё было сделано по инструкции, то вторичный узел должен иметь название `slave`):

```
hduser@euegene-laptop:~$ /usr/local/hadoop/bin/start-dfs.sh
Warning: $HADOOP_HOME is deprecated.

starting namenode, logging to /usr/local/hadoop/libexec/../logs/hadoop-hduser-na
menode-euegene-laptop.out
master: starting datanode, logging to /usr/local/hadoop/libexec/../logs/hadoop-h
duser-datanode-euegene-laptop.out
old_slave: starting datanode, logging to /usr/local/hadoop/libexec/../logs/hadoo
p-hduser-datanode-eugeneold-desktop.out
master: starting secondarynamenode, logging to /usr/local/hadoop/libexec/../logs
/hadoop-hduser-secondarynamenode-euegene-laptop.out
hduser@euegene-laptop:~$ /usr/local/hadoop/bin/start-mapred.sh
Warning: $HADOOP_HOME is deprecated.

starting jobtracker, logging to /usr/local/hadoop/libexec/../logs/hadoop-hduser-
jobtracker-euegene-laptop.out
master: starting tasktracker, logging to /usr/local/hadoop/libexec/../logs/hadoo
p-hduser-tasktracker-euegene-laptop.out
old_slave: starting tasktracker, logging to /usr/local/hadoop/libexec/../logs/ha
doo-hduser-tasktracker-eugeneold-desktop.out
```

```
hduser@euegene-laptop:~$ /usr/local/hadoop/bin/stop-mapred.sh
Warning: $HADOOP_HOME is deprecated.

stopping jobtracker
master: stopping tasktracker
old_slave: stopping tasktracker
hduser@euegene-laptop:~$ /usr/local/hadoop/bin/stop-dfs.sh
Warning: $HADOOP_HOME is deprecated.

stopping namenode
master: stopping datanode
old_slave: stopping datanode
master: stopping secondarynamenode
hduser@euegene-laptop:~$
```

3. Описание проекта

Структура проекта

```
/src
/lib
/input
build.xml
buildrun.sh
run.sh
```

Проект поддерживает компиляцию под следующие архитектуры: linux i386, linux amd64, windows x86, windows x64. Скомпилированные версии тестировались в Windows 7 x64 с библиотекой Karmasphere: Hadoop MapReduce 0.20.203 и в Ubuntu (см. Версии использованного программного обеспечения, раздел 4.2) с версией Hadoop 1.0.4.

Для компиляции необходимо установить ant и находясь в папке с проектом запустить его. Например, проект находится в linux в директории /develop/java/project, запускаем консоль, в ней вводим две команды: "cd /develop/java/project "ant".

3.1. Описание программы

Листинги программы приведены в конце отчёта. За основу был взят код открытого проекта <http://code.google.com/p/hadoop-computer-vision/>, а именно код примера Histogram.java.

Программа состоит из двух фаз. Нахождение порогового значения и применения порога к изображению. На вход алгоритма подаются все изображения находящиеся в заданной папке.

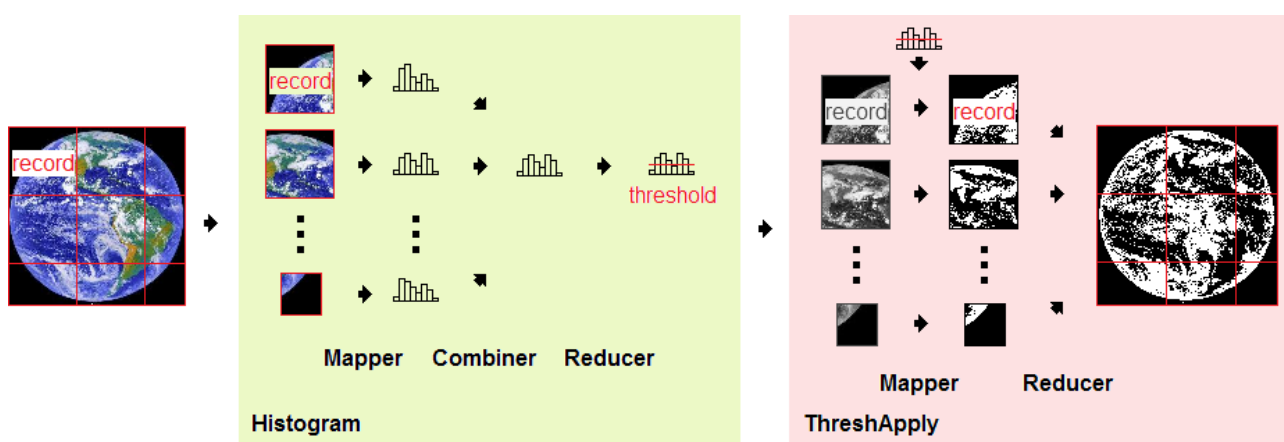


Рис. 1 — Схема работы алгоритма. Обработка одного изображения.

Первая фаза пробегает по изображениям. Метод `getSplits()` задаёт отображение: одному изображению соответствует один split, файлы являются не делимыми для `InputReader`. `ImageRecordReader` делит полученное изображение прямоугольными окнами на более мелкие изображения, теперь представляющие из себя единичные записи.

Мар класс подсчитывает частичные гистограммы для каждой записи.

Входные параметры `map()`:

Ключ — имя файла,

Значение — объект типа `Image`, часть исходного изображения, соответствующая ключу.

Выходные параметры `map()`:

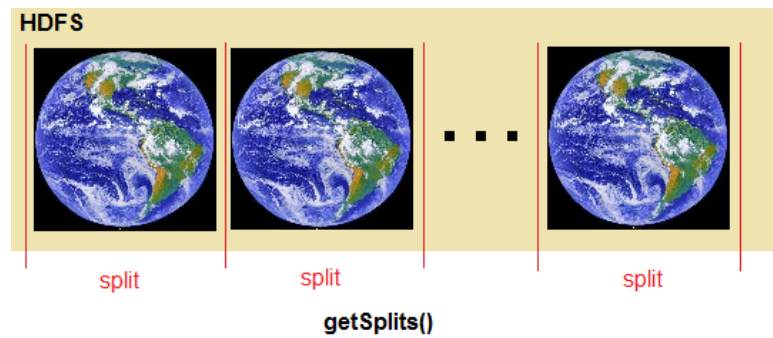


Рис. 2 — Декомпозиция входного потока изображений. Метод getSplits().

Ключ — имя файла,

Значение — объект типа LongArrayWritable, частичная гистограмма.

Combiner класс подсчитывает итоговую гистограмму для всех значений ключа. Входные параметры reduce():

Ключ — имя файла,

Значение — список значений типа LongArrayWritable, частичные гистограммы.

Выходные параметры reduce():

Ключ — имя файла,

Значение — объект типа LongArrayWritable, итоговая гистограмма.

Reducer класс подсчитывает итоговую гистограмму для всех значений ключа. Входные параметры reduce():

Ключ — имя файла,

Значение — объект типа LongArrayWritable, итоговая гистограмма.

Выходные параметры reduce():

Ключ — имя файла,

Значение — текстовое значение подсчитанного порога.

Вторая фаза на вход получает те же изображения, что и первая фаза. Путь к каталогу с подсчитанными пороговыми значениями передаётся через конфигурацию, класс Configuration, вызовом

```
conf.setStrings("mapreduce.imagerecordreader.threshpath", args[1])
```

где args[1] — временная папка. Декомозиция входного потока изображений такая же, как и в первой фазе.

Map класс находит пороговое значение для данного ключа—имени файла и применяет его к текущей записи.

Входные параметры map():

Ключ — имя файла,

Значение — объект типа Image, часть исходного изображения, соответствующая ключу.

Выходные параметры map():

Ключ — имя файла,

Значение — объект типа Image, обработанная часть исходного изображения.

Reducer класс объединяет части данного ключа в итоговое крупное изображение.

Входные параметры reduce():

Ключ — имя файла,

Значение — объект типа Image, обработанная часть исходного изображения.

Выходные параметры reduce():

Ключ — имя файла,

Значение — объект типа Image, обработанное исходное изображение.

3.2. Описание build.xml

Рассмотрим build скрипт.

```
<project default="endpoint"
        name="Create Runnable Jar for Project NewSandCastle"
        basedir=". ">
```

Описываем создание проекта "Create Runnable Jar for Project NewSandCastle текущий каталог - каталог расположения build.xml. Построение будет ждать завершения элемента с названием "endpoint" и всех зависимых элементов.

```
<property name="src" location="src"/>
<property name="build" location="build"/>
<property name="lib" location="lib"/>
<property name="jarname" location="MethodOzu.jar"/>
```

Устанавливаем необходимые переменные.

```
<condition property="isWindows86">
  <and>
    <os family="windows"/>
    <or>
      <equals arg1="{os.arch}" arg2="i386"/>
      <equals arg1="{os.arch}" arg2="x86"/>
    </or>
  </and>
</condition>
<condition property="isWindows64">
  <and>
    <os family="windows"/>
    <or>
      <equals arg1="{os.arch}" arg2="amd64"/>
      <equals arg1="{os.arch}" arg2="x86_64"/>
    </or>
  </and>
```

```

</condition>

<condition property="isLinux86">
  <and>
    <os family="unix"/>
    <or>
      <equals arg1="${os.arch}" arg2="i386"/>
      <equals arg1="${os.arch}" arg2="x86"/>
    </or>
  </and>
</condition>
<condition property="isLinux64">
  <and>
    <os family="unix"/>
    <or>
      <equals arg1="${os.arch}" arg2="amd64"/>
      <equals arg1="${os.arch}" arg2="x86_64"/>
    </or>
  </and>
</condition>

```

Определяем целевую платформу. `<os family="unix"/>` возвратит true если текущая система unix. `<condition property="isLinux64">` установит переменную isLinux64 в true, соответственно возвращаемому значению вложенного выражения.

```

<target name="init">
  <tstamp/>
  <mkdir dir="${build}"/>
</target>

```

Элемент построения "init". `<tstamp/>` - текущее время. `<mkdir dir="${build}"/>` создание папки для сохранения скомпилированных файлов.

```

<target name="compile" depends="init"
  description="compile the source">
  <javac srcdir="${src}"
    destdir="${build}">
    <classpath>
      <fileset dir="${lib}">
        <include name="**/*.jar" />
      </fileset>
    </classpath>
  </javac>
</target>

```

Элемент построения "compile". `depends="init"` выполняется после завершения "init". В `javac` устанавливаются каталоги с исходным кодом, каталог для скомпилированных классов соответственно ранее установленным переменным `src` и `build`. Также указывается переменная `classpath`, указывающая на каталог с библиотеками.

```

<target if="isLinux86" name="create_jar_linux86"
  depends="compile">
  <delete file="${jarname}"/>
  <jar destfile="${jarname}"
    filesetmanifest="mergewithoutmain">
    <manifest>

```



```

        <attribute name="Main-Class"
            value="mainpackage.MainClass"/>
    </manifest>
    <zipfileset excludes="META-INF/maven/**"
        src="${lib}/javacv-bin/javacpp.jar"/>
    <zipfileset excludes="META-INF/maven/**"
        src="${lib}/javacv-bin/javacv.jar"/>
    <zipfileset excludes="META-INF/maven/**"
        src="${lib}/jaivacv-bin/javacv-linux-x86.jar"/>
    <zipfileset excludes="META-INF/maven/**"
        src="${lib}/javacv-cppjars/opencv-2.4.4-linux-x86.jar"/>
    <fileset dir="${build}"/>
</jar>
</target>

```

Один из четырёх элементов построения, собирающий скомпилированные файлы в jar. Он также объединяет полученный jar с необходимыми библиотеками.

```

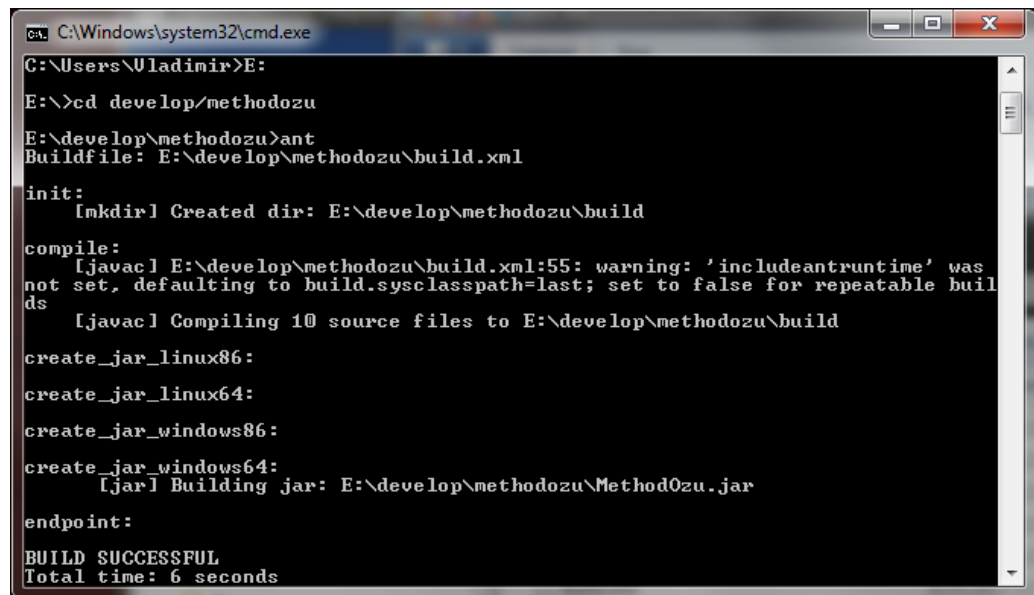
<target name="endpoint" depends="create_jar_linux86 ,
                                create_jar_linux64 ,
                                create_jar_windows86 ,
                                create_jar_windows64"/>

```

Конечная цель проекта построения. Построение на нём завершается.

3.3. Компиляция и запуск под Windows

Чтобы скомпилировать и запустить проект необходимо установить Eclipse, Karmasphere, Apache ant.



```
C:\Windows\system32\cmd.exe
G:\Users\Uladimir>E:
E:\>cd develop\methodozu
E:\develop\methodozu>ant
Buildfile: E:\develop\methodozu\build.xml

init:
[mkdir] Created dir: E:\develop\methodozu\build

compile:
[javac] E:\develop\methodozu\build.xml:55: warning: 'includeantruntime' was
not set, defaulting to build.sysclasspath=last; set to false for repeatable buil
ds
[javac] Compiling 10 source files to E:\develop\methodozu\build

create_jar_linux86:

create_jar_linux64:

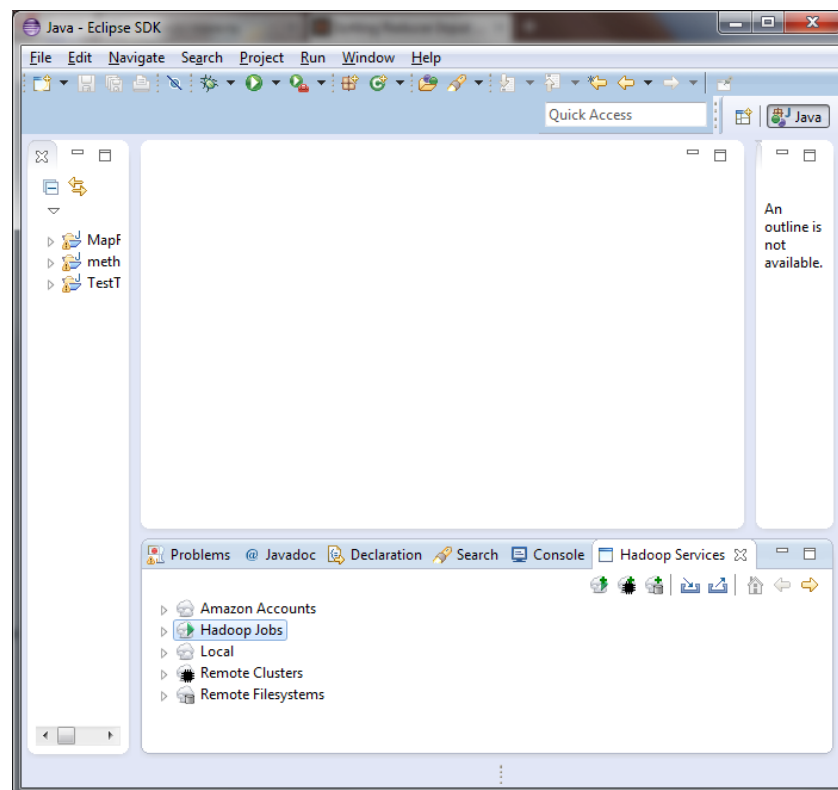
create_jar_windows86:

create_jar_windows64:
[jar] Building jar: E:\develop\methodozu\MethodOzu.jar

endpoint:

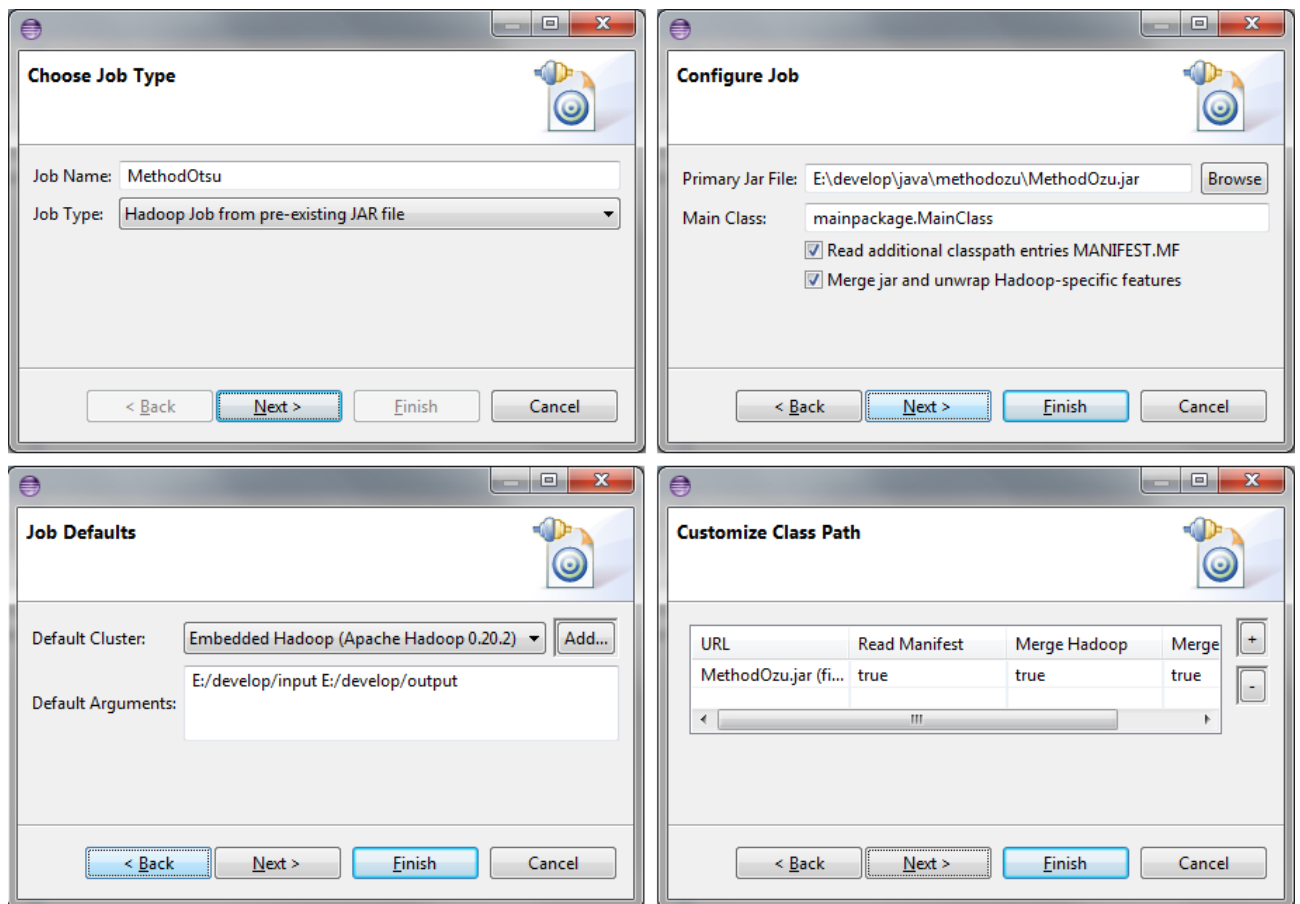
BUILD SUCCESSFUL
Total time: 6 seconds
```

После завершения подготовительных операций в Eclipse должна быть доступна вкладка Hadoop Services.



Вызвав окно консоли и перейдя в папку с проектом запускаем компиляцию командой ant. Результатом будет каталог /build и jar файл "MethodOzu.jar".

Далее работаем с eclipse. В Hadoop Services создаём New Job, выбрав правой клавишей мыши Hadoop Jobs. Следуем инструкциям.



Перед запуском необходимо предварительно убедиться, что существует указанный каталог /input и не существует /output и /tmp (может остаться в случае частичного завершения работы программы).

После запуска обработанные изображения будут сохраняться в указанный /output.



3.4. Компиляция и запуск под Linux

Чтобы скомпилировать и запустить проект необходимо установить Apache ant, настроить Apache Hadoop (см. Настройка оборудования, раздел 2).

Копируем изображения в папку /input. Убедившись что сервер запущен, а \$HADOOP_HOME указывает на каталог установленного сервера (echo \$HADOOP_HOME), запускаем один из скриптов buildrun.sh (компиляция и запуск), либо run.sh (запуск).

Результат будет находиться в hdfs в каталоге /user/hduser/method-otsu-out.

3.5. Установка ant под Linux

```
sudo apt-get update  
sudo apt-get install ant
```

4. Описание среды

4.1. Классификация версий Apache Hadoop

1.0.X — текущая стабильная версия, 1.0 release

1.1.X — текущая бета версия, 1.1 release

2.X.X — текущая альфа версия

0.23.X — то же, что и 2.X.X без NN HA.

0.22.X — модули безопасности

0.20.203.X — старая legacy стабильная версия

0.20.X — старая legacy версия

4.2. Версии использованного программного обеспечения

Apache Ant(TM) version 1.9.0 compiled on March 5 2013,

Hadoop 1.0.4,

Karmasphere Hadoop MapReduce 0.20.203,

Ubuntu 11.10, Ubuntu 12.04 i386,

Windows 7 x64,

Eclipse 4.2.2,

javacv-0.4 (javacv-0.4-bin.zip, javacv-0.4-cppjars.zip).

4.3. Известные проблемы

- 1) Запуск в Karmasphere. Если в пути к /input (и /output) том диска указан заглавной буквой, например E:/develop/input, а не e:/develop/input, то временный каталог /tmp не удалится.

ПРИЛОЖЕНИЕ А

Исходный код функций

MainClass.java

```
1 package mainpackage;
2
3
4
5 import org.apache.hadoop.conf.Configuration;
6 import org.apache.hadoop.util.ToolRunner;
7 import org.codehaus.jackson.map.util.ArrayBuilders;
8
9 // Program consists of two phases. Calculating histogram and applying
   threshold.
10 //
11 // First phase
12 // is run over images. Method getSplits works that split=image, files are
   not splittable.
13 // ImageRecordReader divides source image by rectangular subwindows, so
   this image part is a record.
14 //
15 // Map calculates partial histograms.
16 // Map output:
17 //   key format – <filename>
18 //   value – list of histograms of image parts.
19 //
20 // Reduce collects combined pairs of key defining source image file and
   list of histograms of all it's parts.
21 // Each reduce method calculates threshold according to result histogram.
   This threshold would be written to tempdir path
22 // in following format:
23 //   <filename> <threshold> \newline
24 //
25 //
26 // Second phase
27 // similarly is run over the same images. except they read as grayscale
   images(configuration parameter isColor == 0).
28 // Directory with files containing calculated thresholds is passed to
   algorithm
29 // via Configuration: conf.setStrings("mapreduce.imagerecordreader.
   threshpath", args[1]); , where args[1] is a tempdir,
30 // output path for first map reduce job.
31 // Division on parts remains unchanged.
32 //
33 // Map finds a threshold for given image filename in all files were with
   thresholds and applies threshold.
34 //
35 // Reduce collects parts and for each key stitch all parts together.
36 public class MainClass {
37     // main args: <input path> <output path>
38     public static void main(String[] args) throws Exception {
```

```

39     String tempdir = args[0].substring(0, args[0].length()-1);
40     tempdir = tempdir.substring(0, tempdir.lastIndexOf('/') + 1) + "tmp";
41     String[] newargs = new String[3];
42     newargs[0] = args[0];
43     newargs[1] = tempdir;
44     newargs[2] = args[1];
45     args[1] = tempdir;
46     int res = ToolRunner.run(new Configuration(), new Histogram(), args);
47     int res2 = ToolRunner.run(new Configuration(), new ThreshApply(),
        newargs);
48     System.exit(res);
49 }
50 }

```

Histogram.java

```

1 package mainpackage;
2
3 import java.io.IOException;
4 import java.nio.ByteBuffer;
5 import java.util.Iterator;
6
7 import org.apache.hadoop.conf.Configuration;
8 import org.apache.hadoop.conf.Configured;
9 import org.apache.hadoop.fs.Path;
10 import org.apache.hadoop.io.LongWritable;
11 import org.apache.hadoop.io.Text;
12 import org.apache.hadoop.io.Writable;
13 import org.apache.hadoop.mapreduce.Job;
14 import org.apache.hadoop.mapreduce.Mapper;
15 import org.apache.hadoop.mapreduce.Reducer;
16 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
17 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
18 import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
19 import org.apache.hadoop.util.Tool;
20 import org.apache.hadoop.util.ToolRunner;
21
22 import static com.googlecode.javacv.cpp.opencv_core.*;
23 import static com.googlecode.javacv.cpp.opencv_imgproc.*;
24
25 import edu.vt.io.Image;
26 import edu.vt.io.*;
27 import edu.vt.input.ImageInputFormat;
28
29 public class Histogram extends Configured implements Tool {
30     public static class Map extends
31         Mapper<Text, Image, Text, LongArrayWritable> {
32         private final static LongWritable one = new LongWritable(1);
33
34         @Override
35         public void map(Text key, Image value, Context context)
36             throws IOException, InterruptedException {
37

```

```

38     // Convert to gray scale image
39     IplImage im1 = value.getImage();
40     IplImage im2 = cvCreateImage(cvSize(im1.width(), im1.height()),
41         IPL_DEPTH_8U, 1);
42     cvCvtColor(im1, im2, CV_BGR2GRAY);
43
44     // Initialize histogram array
45     LongWritable[] histogram = new LongWritable[256];
46     for (int i = 0; i < histogram.length; i++) {
47         histogram[i] = new LongWritable();
48     }
49
50     // Compute histogram
51     ByteBuffer buffer = im2.getByteBuffer();
52     while (buffer.hasRemaining()) {
53         int val = buffer.get() + 128;
54         histogram[val].set(histogram[val].get() + 1);
55     }
56
57     context.write(key, new LongArrayWritable(histogram));
58 }
59 }
60
61 public static class Combine extends
62     Reducer<Text, LongArrayWritable, Text, LongArrayWritable> {
63
64     @Override
65     public void reduce(Text key, Iterable<LongArrayWritable> values,
66         Context context) throws IOException, InterruptedException {
67
68         // Initialize histogram array
69         LongWritable[] histogram = new LongWritable[256];
70         for (int i = 0; i < histogram.length; i++) {
71             histogram[i] = new LongWritable();
72         }
73
74         // Sum the parts
75         Iterator<LongArrayWritable> it = values.iterator();
76         while (it.hasNext()) {
77             LongWritable[] part = (LongWritable[]) it.next().toArray();
78             for (int i = 0; i < histogram.length; i++) {
79                 histogram[i].set(histogram[i].get() + part[i].get());
80             }
81         }
82
83         context.write(key, new LongArrayWritable(histogram));
84     }
85 }
86
87 public static class Reduce extends
88     Reducer<Text, LongArrayWritable, Text, Text> {
89
90     @Override

```



```

91     public void reduce(Text key, Iterable<LongArrayWritable> values,
92         Context context) throws IOException, InterruptedException {
93         Iterator<LongArrayWritable> it = values.iterator();
94         LongWritable[] histogram = (LongWritable[]) it.next().toArray();
95
96         int m = 0;
97         int n = 0;
98         for (int t = 0; t < 256; t++) {
99             m += t * histogram[t].get();
100            n += histogram[t].get();
101        }
102
103        float maxSigma = -1;
104        int threshold = 0;
105
106        int alpha1 = 0;
107        int beta1 = 0;
108
109        for (int t = 0; t < 256; t++) {
110            alpha1 += t * histogram[t].get();
111            beta1 += histogram[t].get();
112
113
114            float w1 = (float) beta1 / n;
115
116            float a = (float) alpha1 / beta1 - (float) (m - alpha1)
117                / (n - beta1);
118
119
120            float sigma = w1 * (1 - w1) * a * a;
121
122            if (sigma > maxSigma) {
123                maxSigma = sigma;
124                threshold = t;
125            }
126        }
127
128        context.write(key, new Text(String.valueOf(threshold)));
129    }
130 }
131
132 public int run(String[] args) throws Exception {
133     // Set various configuration settings
134     Configuration conf = getConf();
135     conf.setInt("mapreduce.imagerecordreader.windowsizepercent", 23);
136     conf.setInt("mapreduce.imagerecordreader.windowoverlappercent", 0);
137
138     // Create job
139     Job job = new Job(conf);
140
141     // Specify various job-specific parameters
142     job.setJarByClass(Histogram.class);
143     job.setJobName("Histogram");

```

```

144
145     job.setOutputKeyClass( Text.class );
146     job.setOutputValueClass( Text.class );
147     job.setMapOutputKeyClass( Text.class );
148     job.setMapOutputValueClass( LongArrayWritable.class );
149
150     job.setMapperClass( Map.class );
151     job.setCombinerClass( Combine.class );
152     job.setReducerClass( Reduce.class );
153
154     job.setInputFormatClass( ImageInputFormat.class );
155     job.setOutputFormatClass( TextOutputFormat.class );
156
157     // Set input and output paths
158     FileInputFormat.addInputPath( job, new Path( args[0] ) );
159     FileOutputFormat.setOutputPath( job, new Path( args[1] ) );
160
161     return job.waitForCompletion( true ) ? 0 : 1;
162 }
163 }

```

ThreshApply.java

```

1 package mainpackage;
2
3 import java.io.BufferedReader;
4 import java.io.File;
5 import java.io.FileReader;
6 import java.io.IOException;
7 import java.io.InputStreamReader;
8 import java.nio.ByteBuffer;
9 import java.util.ArrayList;
10 import java.util.Iterator;
11 import java.util.List;
12 import java.util.StringTokenizer;
13
14 import org.apache.commons.lang.StringUtils;
15 import org.apache.hadoop.conf.Configuration;
16 import org.apache.hadoop.conf.Configured;
17 import org.apache.hadoop.fs.Path;
18 import org.apache.hadoop.io.LongWritable;
19 import org.apache.hadoop.io.Text;
20 import org.apache.hadoop.io.Writable;
21 import org.apache.hadoop.mapred.InvalidInputException;
22 import org.apache.hadoop.mapreduce.Job;
23 import org.apache.hadoop.mapreduce.Mapper;
24 import org.apache.hadoop.mapreduce.Reducer;
25 import org.apache.hadoop.mapreduce.Reducer.Context;
26 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
27 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
28 import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
29 import org.apache.hadoop.fs.FileSystem;
30 import org.apache.hadoop.fs.FileStatus;

```

```

31 import org.apache.hadoop.util.Tool;
32 import org.apache.hadoop.util.ToolRunner;
33 import java.util.logging.Logger;
34 import java.util.logging.Level;
35
36 import com.googlecode.javacv.cpp.opencv_core.CvRect;
37 import com.googlecode.javacv.cpp.opencv_core.IplImage;
38
39 import static com.googlecode.javacv.cpp.opencv_core.*;
40 import static com.googlecode.javacv.cpp.opencv_imgproc.*;
41
42 import edu.vt.io.Image;
43 import edu.vt.io.LongArrayWritable;
44 import edu.vt.io.WindowInfo;
45 import edu.vt.input.ImageInputFormat;
46 import edu.vt.output.ImageOutputFormat;
47
48 public class ThreshApply extends Configured implements Tool {
49     public static class Map extends Mapper<Text, Image, Text, Image> {
50
51         private final static LongWritable one = new LongWritable(1);
52         private static Logger logger = Logger.getLogger(ThreshApply.Map.class
53             .getName());
54         private boolean debug = false;
55
56         @Override
57         public void map(Text key, Image value, Context context)
58             throws IOException, InterruptedException {
59             Configuration conf = context.getConfiguration();
60             debug = conf.getBoolean("mapreduce.debug", true);
61
62             if (debug)
63                 logger.log(java.util.logging.Level.INFO,
64                     "VLPR ***** MAP " + key.toString());
65
66             String width = "", height = "", record = "";
67             int threshold = -1;
68
69             // image parameters record format: <filename> threshold
70             record = getImageParameters(key, conf);
71             StringTokenizer tok = new StringTokenizer(record);
72             tok.nextToken();
73             threshold = Integer.valueOf(tok.nextToken());
74
75             if (debug)
76                 logger.log(java.util.logging.Level.INFO, "width " + width
77                     + " height " + height);
78
79             // Getting image piece.
80             IplImage imgray = value.getImage();
81
82             cvThreshold(imgray, imgray, threshold, 255, CV_THRESH_BINARY);
83

```

```

84     context.write(key, new Image(imgray, value.getWindow()));
85 }
86
87 private String getImageParameters(Text key, Configuration conf)
88     throws IOException {
89     List<IOException> errors = new ArrayList<IOException>();
90     // Threshold preliminaries
91     Path threshdir = new Path(
92         conf.get("mapreduce.imagerecordreader.threshpath"));
93
94     if (debug)
95         logger.log(java.util.logging.Level.INFO, "VLPR getImageParameters
96             threshdir " + threshdir.toString());
97
98     // Scan directory with calculated thresholds. Look through files
99     // and
100    // try to find threshold
101    // corresponding to source image file name.
102    FileSystem fs = threshdir.getFileSystem(conf);
103    FileStatus[] matches = fs.globStatus(threshdir);
104    if (matches == null) {
105        errors.add(new IOException("Input path does not exist: " +
106            threshdir.toString()));
107    } else if (matches.length == 0) {
108        errors.add(new IOException("Input Pattern " + threshdir.toString()
109            () + " matches 0 files"));
110    } else {
111        if (matches.length != 1) {
112            errors.add(new IOException("More than 1 directory for " +
113                threshdir.toString()));
114        }
115
116        FileStatus globStat = matches[0];
117        for (FileStatus stat : fs.listStatus(globStat.getPath())) {
118            if (!stat.isDir()) {
119                BufferedReader br = new BufferedReader(new InputStreamReader(
120                    fs.open(stat.getPath())));
121                String line;
122
123                if (debug)
124                    logger.log(java.util.logging.Level.INFO, "VLPR
125                        getImageParameters stat " + stat.getPath().toString());
126
127                while ((line = br.readLine()) != null) {
128                    if (debug) {
129                        logger.log(java.util.logging.Level.INFO, "VLPR
130                            getImageParameters key " + key.toString());
131                        logger.log(java.util.logging.Level.INFO, "VLPR
132                            getImageParameters key " + line);
133                    }
134
135                    if (StringUtils.contains(line, key.toString())) {
136                        return line;

```

```

127         }
128     }
129 }
130 }
131 }
132
133     if (!errors.isEmpty()) {
134         throw new InvalidInputException(errors);
135     } else {
136         errors.add(new IOException("No record for image key " + key.
137             toString()));
138         throw new InvalidInputException(errors);
139     }
140 }
141
142 public static class Reduce extends Reducer<Text, Image, Text, Image> {
143
144     private static Logger logger = Logger
145         .getLogger(ThreshApply.Reduce.class.getName());
146     private boolean debug = false;
147     private int currentSplit;
148
149     @Override
150     public void reduce(Text key, Iterable<Image> values, Context context)
151         throws IOException, InterruptedException {
152         debug = context.getConfiguration().getBoolean("mapreduce.debug",
153             true);
154
155         if (debug)
156             logger.log(java.util.logging.Level.INFO,
157                 "VLPR ***** REDUCE key: "
158                 + key.toString());
159         Image image;
160         IplImage bigimage = null;
161         IplImage imagepart = null;
162
163         boolean first = true;
164         Iterator it = values.iterator();
165         while (it.hasNext()) {
166             image = (Image)it.next();
167             imagepart = image.getImage();
168             WindowInfo window = image.getWindow();
169             if (first) {
170                 logger.log(java.util.logging.Level.INFO, "VLPR before cvCreate
171                     parentWidth:" + window.getParentWidth() + " parenHeight:" +
172                     window.getParentHeight());
173                 bigimage = cvCreateImage(new CvSize(window.getParentWidth(),
174                     window.getParentHeight()), IPL_DEPTH_8U, 1);
175                 first = false;
176             }
177
178             logger.log(java.util.logging.Level.INFO, "VLPR imagechannels " +

```

```

        imagepart.nChannels() + " imagedepth " + imagepart.depth());
176 CvRect roi = window.computeROI();
177
178     if (debug) {
179         logger.log(java.util.logging.Level.INFO, "VLPR imagechannels "
            + imagepart.nChannels() + " imagedepth " + imagepart.depth()
            );
180         logger.log(java.util.logging.Level.INFO, "VLPR width " + image
            .getWidth());
181         logger.log(java.util.logging.Level.INFO, "VLPR height " + image
            .getHeight());
182         logger.log(java.util.logging.Level.INFO, "VLPR roi w: " + roi.
            width() + " h: " + roi.height() + " x: " + roi.x() + " y:" +
            roi.y());
183     }
184
185     cvSetImageROI(bigimage, roi);
186
187     // copy sub-image
188     cvCopy(imagepart, bigimage, null);
189     cvResetImageROI(bigimage);
190 }
191 context.write(key, new Image(bigimage));
192 }
193 }
194
195 public int run(String[] args) throws Exception {
196     // Set various configuration settings
197     Configuration conf = getConf();
198     conf.setInt("mapreduce.imagerecordreader.windowsizepercent", 23);
199     conf.setInt("mapreduce.imagerecordreader.windowoverlappercent", 0);
200     conf.setStrings("mapreduce.imagerecordreader.threshpath", args[1]);
201     conf.setBoolean("mapreduce.debug", true);
202     conf.setInt("mapreduce.imagerecordreader.iscolor", 0);
203
204     // Create job
205     Job job = new Job(conf);
206
207     // Specify various job-specific parameters
208     job.setJarByClass(ThreshApply.class);
209     job.setJobName("ThreshApply");
210
211     job.setOutputKeyClass(Text.class);
212     job.setOutputValueClass(Image.class);
213
214     job.setMapperClass(Map.class);
215     job.setReducerClass(Reduce.class);
216     // job.setNumReduceTasks(0);
217
218     job.setInputFormatClass(ImageInputFormat.class);
219     job.setOutputFormatClass(ImageOutputFormat.class);
220
221     // Set input and output paths

```

```

222     FileInputFormat.addInputPath(job, new Path(args[0]));
223     FileOutputFormat.setOutputPath(job, new Path(args[2]));
224
225     int ok = job.waitForCompletion(true) ? 0 : 1;
226     // Optional
227     Path tmppath = new Path(args[1]);
228     tmppath.getFileSystem(conf).delete(tmppath, true);
229     return ok;
230 }
231 }

```

ImageInputFormat.java

```

1 package edu.vt.input;
2
3 import java.io.IOException;
4
5 import org.apache.hadoop.fs.Path;
6 import org.apache.hadoop.io.Text;
7 import org.apache.hadoop.mapreduce.InputSplit;
8 import org.apache.hadoop.mapreduce.JobContext;
9 import org.apache.hadoop.mapreduce.RecordReader;
10 import org.apache.hadoop.mapreduce.TaskAttemptContext;
11 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
12
13 import edu.vt.io.Image;
14
15 public class ImageInputFormat extends FileInputFormat<Text, Image> {
16
17     @Override
18     public RecordReader<Text, Image> createRecordReader(InputSplit split,
19         TaskAttemptContext context) throws IOException,
20         InterruptedException {
21         return new ImageRecordReader();
22     }
23
24     @Override
25     protected boolean isSplittable(JobContext context, Path file) {
26         return false;
27     }
28 }

```

ImageRecordReader.java

```

1 package edu.vt.input;
2
3 import java.io.FileReader;
4 import java.io.IOException;
5 import java.io.File;
6 import java.io.BufferedReader;
7 import java.util.logging.Level;
8
9 import org.apache.commons.lang.StringUtils;

```

```

10 import org.apache.commons.logging.Log;
11 import org.apache.commons.logging.LogFactory;
12
13 import org.apache.hadoop.conf.Configuration;
14 import org.apache.hadoop.fs.FSDataInputStream;
15 import org.apache.hadoop.fs.FileSystem;
16 import org.apache.hadoop.fs.Path;
17 import org.apache.hadoop.io.Text;
18 import org.apache.hadoop.mapreduce.InputSplit;
19 import org.apache.hadoop.mapreduce.RecordReader;
20 import org.apache.hadoop.mapreduce.TaskAttemptContext;
21 import org.apache.hadoop.mapreduce.lib.input.FileSplit;
22
23 import com.googlecode.javacpp.BytePointer;
24
25 import edu.vt.io.Image;
26 import edu.vt.io.WindowInfo;
27
28 import static com.googlecode.javacv.cpp.opencv_core.*;
29 import static com.googlecode.javacv.cpp.opencv_highgui.*;
30
31 public class ImageRecordReader extends RecordReader<Text, Image> {
32
33     private static final Log LOG = LogFactory.getLog(ImageRecordReader.
34         class);
35     // Image information
36     private String fileName = null;
37     private Image image = null;
38
39     // Key/Value pair
40     private Text key = null;
41     private Image value = null;
42
43     // Configuration parameters
44     // By default use percentage for splitting
45     boolean byPixel = false;
46     int sizePercent = 0;
47     int sizePixel = 0;
48     int borderPixel = 0;
49     int iscolor = -1;
50
51     // splits based on configuration parameters
52     int totalXSplits = 0;
53     int totalYSplits = 0;
54     int xSplitPixels = 0;
55     int ySplitPixels = 0;
56
57     // Current split
58     int currentSplit = 0;
59
60     @Override
61     public void close() throws IOException {

```



```

62     }
63
64     @Override
65     public Text getCurrentKey() throws IOException, InterruptedException {
66
67         return key;
68     }
69
70     @Override
71     public Image getCurrentValue() throws IOException, InterruptedException
72     {
73         return value;
74     }
75
76     @Override
77     public float getProgress() throws IOException, InterruptedException {
78
79         return (float) (totalXSplits * totalYSplits) / (float) currentSplit;
80     }
81
82     @Override
83     public void initialize(InputSplit genericSplit, TaskAttemptContext
84         context)
85         throws IOException, InterruptedException {
86         // Get file split
87         FileSplit split = (FileSplit) genericSplit;
88         Configuration conf = context.getConfiguration();
89
90         // Read configuration parameters
91         getConfig(conf);
92
93         // Open the file
94         Path file = split.getPath();
95         FileSystem fs = file.getFileSystem(conf);
96         FSDataInputStream fileIn = fs.open(split.getPath());
97
98         // Read file and decode image
99         byte[] b = new byte[fileIn.available()];
100         fileIn.readFully(b);
101         image = new Image(cvDecodeImage(
102             cvMat(1, b.length, CV_8UC1, new BytePointer(b)), iscolor));
103
104         // Get filename to use as key
105         fileName = split.getPath().getName().toString();
106
107         // Calculate the number of splits
108         calculateSplitInfo();
109         currentSplit = 0;
110
111     }
112

```

```

113 // Was modified
114 @Override
115 public boolean nextKeyValue() throws IOException, InterruptedException
116 {
117     if (currentSplit < (totalXSplits * totalYSplits) && fileName != null)
118     {
119         key = new Text(fileName);
120         if (totalXSplits * totalYSplits == 1) {
121             value = image;
122         } else {
123             value = getSubWindow();
124         }
125         currentSplit += 1;
126         return true;
127     }
128     return false;
129 }
130
131 private Image getSubWindow() {
132     WindowInfo window = createWindow();
133     CvRect roi = window.computeROI();
134
135     // sets the ROI
136     IplImage img1 = image.getImage();
137     cvSetImageROI(img1, roi);
138
139     // create destination image
140     IplImage img2 = cvCreateImage(cvSize(roi.width(), roi.height()),
141         img1.depth(), img1.nChannels());
142
143     // copy sub-image
144     cvCopy(img1, img2, null);
145
146     // reset the ROI
147     cvResetImageROI(img1);
148
149     return new Image(img2, window);
150 }
151
152 private void getConfig(Configuration conf) {
153     // Ensure that value is not negative
154     borderPixel = conf.getInt("mapreduce.imagerecordreader.borderPixel",
155         0);
156     if (borderPixel < 0) {
157         borderPixel = 0;
158     }
159
160     // Ensure that percentage is between 0 and 100
161     sizePercent = conf.getInt(

```

```

163         "mapreduce.imagerecordreader.windowsizepercent", 100);
164     if (sizePercent < 0 || sizePercent > 100) {
165         sizePercent = 100;
166     }
167
168     // Ensure that value is not negative
169     sizePixel = conf.getInt("mapreduce.imagerecordreader.windowsizepixel"
170
171         , Integer.MAX_VALUE);
172     if (sizePixel < 0) {
173         sizePixel = 0;
174     }
175
176     iscolor = conf.getInt("mapreduce.imagerecordreader.iscolor", -1);
177
178     byPixel = conf.getBoolean("mapreduce.imagerecordreader.windowbypixel"
179
180         , false);
181 }
182
183 private WindowInfo createWindow() {
184     WindowInfo window = new WindowInfo();
185
186     // Get current window
187     int x = currentSplit % totalXSplits;
188     int y = currentSplit / totalYSplits;
189
190     int width = xSplitPixels;
191     int height = ySplitPixels;
192
193     // Deal with partial windows
194     if (x * xSplitPixels + width > image.getWidth()) {
195         width = image.getWidth() - x * xSplitPixels;
196     }
197     if (y * ySplitPixels + height > image.getHeight()) {
198         height = image.getHeight() - y * ySplitPixels;
199     }
200
201     window.setParentInfo(x * xSplitPixels, y * ySplitPixels,
202         image.getHeight(), image.getWidth());
203     window.setWindowSize(height, width);
204
205     // Calculate borders
206     int top = 0;
207     int bottom = 0;
208     int left = 0;
209     int right = 0;
210
211     if (window.getParentXOffset() > borderPixel) {
212         left = borderPixel;
213     }
214     if (window.getParentYOffset() > borderPixel) {
215         top = borderPixel;

```

```

214     }
215     if (window.getParentXOffset() + borderPixel + window.getWidth() <
        window
216         .getParentWidth()) {
217         right = borderPixel;
218     }
219     if (window.getParentYOffset() + borderPixel + window.getHeight() <
        window
220         .getParentHeight()) {
221         bottom = borderPixel;
222     }
223
224     window.setBorder(top, bottom, left, right);
225     return window;
226 }
227
228 private void calculateSplitInfo() {
229     if (byPixel) {
230         xSplitPixels = sizePixel;
231         ySplitPixels = sizePixel;
232         totalXSplits = (int) Math.ceil(image.getWidth()
233             / Math.min(xSplitPixels, image.getWidth()));
234         totalYSplits = (int) Math.ceil(image.getHeight()
235             / Math.min(ySplitPixels, image.getHeight()));
236     } else {
237         xSplitPixels = (int) (image.getWidth() * (sizePercent / 100.0));
238         ySplitPixels = (int) (image.getHeight() * (sizePercent / 100.0));
239         totalXSplits = (int) Math.ceil(image.getWidth()
240             / (double) Math.min(xSplitPixels, image.getWidth()));
241         totalYSplits = (int) Math.ceil(image.getHeight()
242             / (double) Math.min(ySplitPixels, image.getHeight()));
243     }
244 }
245 }

```

LongArrayWritable.java

```

1 package edu.vt.io;
2
3 import org.apache.hadoop.io.ArrayWritable;
4 import org.apache.hadoop.io.LongWritable;
5
6 public class LongArrayWritable extends ArrayWritable {
7     public LongArrayWritable() {
8         super(LongWritable.class);
9     }
10
11     public LongArrayWritable(LongWritable[] values) {
12         super(LongWritable.class, values);
13     }
14
15     public String toString(){
16         String [] strings = toStrings();

```

```

17     String str = "(" + strings.length + ")[";
18     for (int i = 0; i < strings.length; i++) {
19         str += strings[i] + " ";
20     }
21     str += "]";
22     return str;
23 }
24 }

```

Image.java

```

1 package edu.vt.io;
2
3 import java.io.DataInput;
4 import java.io.DataOutput;
5 import java.io.IOException;
6 import java.nio.ByteBuffer;
7
8 import org.apache.commons.logging.Log;
9 import org.apache.commons.logging.LogFactory;
10
11 import org.apache.hadoop.io.Writable;
12 import org.apache.hadoop.io.WritableUtils;
13
14 import com.googlecode.javacpp.BytePointer;
15 import com.googlecode.javacv.cpp.opencv_core.IplImage;
16
17 import static com.googlecode.javacv.cpp.opencv_core.*;
18 import static com.googlecode.javacv.cpp.opencv_highgui.*;
19
20 public class Image implements Writable {
21
22     private static final Log LOG = LogFactory.getLog(Image.class);
23
24     // IPL image
25     private IplImage image = null;
26     private WindowInfo window = null;
27
28     public Image() {
29     }
30
31     // Create Image from IplImage
32     public Image(IplImage image) {
33         this.image = image;
34         this.window = new WindowInfo();
35     }
36
37     // Create empty Image
38     public Image(int height, int width, int depth, int nChannels){
39         this.image = cvCreateImage(cvSize(width, height), depth, nChannels);
40         this.window = new WindowInfo();
41     }
42

```

```

43 // Create Image from IplImage and IplROI
44 public Image(IplImage image, WindowInfo window) {
45     this.image = image;
46     this.window = window;
47 }
48
49 public IplImage getImage() {
50     return image;
51 }
52
53 // get window where image came from
54 public WindowInfo getWindow() {
55     return window;
56 }
57
58 // Pixel depth in bits
59 // PL_DEPTH_8U – Unsigned 8-bit integer
60 // IPL_DEPTH_8S – Signed 8-bit integer
61 // IPL_DEPTH_16U – Unsigned 16-bit integer
62 // IPL_DEPTH_16S – Signed 16-bit integer
63 // IPL_DEPTH_32S – Signed 32-bit integer
64 // IPL_DEPTH_32F – Single-precision floating point
65 // IPL_DEPTH_64F – Double-precision floating point
66 public int getDepth() {
67     return image.depth();
68 }
69
70 // Number of channels.
71 public int getNumChannel() {
72     return image.nChannels();
73 }
74
75 // Image height in pixels
76 public int getHeight() {
77     return image.height();
78 }
79
80 // Image width in pixels
81 public int getWidth() {
82     return image.width();
83 }
84
85 // The size of an aligned image row, in bytes
86 public int getWidthStep() {
87     return image.widthStep();
88 }
89
90 // Image data size in bytes.
91 public int getImageSize() {
92     return image.imageSize();
93 }
94
95 // Copies one image into current image using

```

```

96 // information contained in WindowInfo struct
97 public void insertImage(Image sourceImage){
98     IplImage img1 = this.image;
99     IplImage img2 = sourceImage.getImage();
100     WindowInfo win = sourceImage.getWindow();
101
102     // set the ROI on destination image
103     if(win.isParentInfoValid()){
104         cvSetImageROI(img1, cvRect(win.getParentXOffset(), win.
            getParentYOffset(), win.getWidth(), win.getHeight()));
105     }
106     // set the ROI on source image
107     if(win.isBorderValid()){
108         cvSetImageROI(img2, cvRect(win.getBorderLeft(), win.getBorderTop(),
            win.getWidth(), win.getHeight()));
109     }
110
111     // copy sub-image
112     cvCopy(img2, img1, null);
113
114     // reset the ROI
115     cvResetImageROI(img1);
116 }
117
118 @Override
119 public void readFields(DataInput in) throws IOException {
120     // Read image information
121     int height = WritableUtils.readVInt(in);
122     int width = WritableUtils.readVInt(in);
123     int depth = WritableUtils.readVInt(in);
124     int nChannels = WritableUtils.readVInt(in);
125     int imageSize = WritableUtils.readVInt(in);
126
127     // Read window information
128     int windowXOffest = WritableUtils.readVInt(in);
129     int windowYOffest = WritableUtils.readVInt(in);
130     int windowHeight = WritableUtils.readVInt(in);
131     int windowWidth = WritableUtils.readVInt(in);
132
133     int top = WritableUtils.readVInt(in);
134     int bottom = WritableUtils.readVInt(in);
135     int left = WritableUtils.readVInt(in);
136     int right = WritableUtils.readVInt(in);
137
138     int h = WritableUtils.readVInt(in);
139     int w = WritableUtils.readVInt(in);
140
141     window = new WindowInfo();
142     window.setParentInfo(windowXOffest, windowYOffest, windowHeight,
        windowWidth);
143     window.setBorder(top, bottom, left, right);
144     window.setWindowSize(h, w);
145

```

```

146     // Read image bytes
147     byte[] bytes = new byte[imageSize];
148     in.readFully(bytes, 0, imageSize);
149
150     image = cvCreateImage(cvSize(width, height), depth, nChannels);
151     image.imageData(new BytePointer(bytes));
152 }
153
154 @Override
155 public void write(DataOutput out) throws IOException {
156     // Write image information
157     WritableUtils.writeVInt(out, image.height());
158     WritableUtils.writeVInt(out, image.width());
159     WritableUtils.writeVInt(out, image.depth());
160     WritableUtils.writeVInt(out, image.nChannels());
161     WritableUtils.writeVInt(out, image.imageSize());
162
163     // Write window information
164     WritableUtils.writeVInt(out, window.getParentXOffset());
165     WritableUtils.writeVInt(out, window.getParentYOffset());
166     WritableUtils.writeVInt(out, window.getParentHeight());
167     WritableUtils.writeVInt(out, window.getParentWidth());
168
169     WritableUtils.writeVInt(out, window.getBorderTop());
170     WritableUtils.writeVInt(out, window.getBorderBottom());
171     WritableUtils.writeVInt(out, window.getBorderLeft());
172     WritableUtils.writeVInt(out, window.getBorderRight());
173
174     WritableUtils.writeVInt(out, window.getHeight());
175     WritableUtils.writeVInt(out, window.getWidth());
176
177     // Write image bytes
178     ByteBuffer buffer = image.getByteBuffer();
179     while (buffer.hasRemaining()) {
180         out.write(buffer.get());
181     }
182 }
183
184 }

```

ImageOutputFormat.java

```

1 package edu.vt.output;
2
3 import java.io.IOException;
4
5 import org.apache.hadoop.conf.Configuration;
6 import org.apache.hadoop.fs.FileSystem;
7 import org.apache.hadoop.fs.Path;
8 import org.apache.hadoop.io.Text;
9 import org.apache.hadoop.mapreduce.RecordWriter;
10 import org.apache.hadoop.mapreduce.TaskAttemptContext;
11 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

```



```

12
13 import edu.vt.io.Image;
14
15 public class ImageOutputFormat extends FileOutputFormat<Text, Image> {
16
17     @Override
18     public RecordWriter<Text, Image> getRecordWriter(TaskAttemptContext job
19         )
20         throws IOException, InterruptedException {
21
22         Configuration conf = job.getConfiguration();
23         Path outputPath = getOutputPath(job);
24         FileSystem fs = outputPath.getFileSystem(conf);
25
26         return new ImageRecordWriter(outputPath, fs);
27 }

```

ImageRecordWriter.java

```

1 package edu.vt.output;
2
3 import java.io.IOException;
4 import java.nio.ByteBuffer;
5
6 import org.apache.hadoop.fs.FSDataOutputStream;
7 import org.apache.hadoop.fs.FileSystem;
8 import org.apache.hadoop.fs.Path;
9 import org.apache.hadoop.io.Text;
10 import org.apache.hadoop.mapreduce.RecordWriter;
11 import org.apache.hadoop.mapreduce.TaskAttemptContext;
12
13 import edu.vt.io.Image;
14
15 import static com.googlecode.javacv.cpp.opencv_core.*;
16 import static com.googlecode.javacv.cpp.opencv_highgui.*;
17
18 public class ImageRecordWriter extends RecordWriter<Text, Image> {
19
20     private Path outputPath = null;
21     private FileSystem fs = null;
22
23     ImageRecordWriter(Path outputPath, FileSystem fs){
24         this.outputPath = outputPath;
25         this.fs = fs;
26     }
27
28     @Override
29     public void close(TaskAttemptContext context) throws IOException,
30         InterruptedException {
31
32     }
33

```

```

34  @Override
35  public void write(Text key, Image value) throws IOException,
36      InterruptedException {
37
38      // An optional 0-terminated list of JPG parameter pairs <param id,
        value>
39      int jpeg_params[] = { CV_IMWRITE_JPEG_QUALITY, 80, 0 };
40
41      // Get file name and extension
42      String fileName = key.toString();
43      String ext = getFileExt(fileName);
44
45      // Encode image into a single-row matrix of CV_8UC1
46      // Use file extension to determine compression
47      CvMat imageBuffer = cvEncodeImage(ext, value.getImage());
48
49      // Create output file
50      Path filePath = new Path(outputPath, fileName);
51      FSDataOutputStream fileStream = fs.create(filePath, false);
52
53      // Write the image to file
54      ByteBuffer buffer = imageBuffer.getByteBuffer();
55      while(buffer.hasRemaining()){
56          fileStream.write(buffer.get());
57      }
58
59      // Close the file stream
60      fileStream.close();
61  }
62
63  public String getFileExt(String fileName){
64      int idx = fileName.lastIndexOf('.');
65      if(idx == -1){
66          return null;
67      }
68
69      return fileName.substring(idx, fileName.length());
70  }
71
72 }

```

WindowInfo.java

```

1  package edu.vt.io;
2
3  import static com.googlecode.javacv.cpp.opencv_core.CvRect;
4
5  import com.googlecode.javacv.cpp.opencv_core.CvRect;
6
7  public class WindowInfo {
8
9      // Size of parent image
10     private int parentWidth = -1;

```

```

11  private int parentHeight = -1;
12
13  // Location of window in parent image
14  private int parentXOffset = -1;
15  private int parentYOffset = -1;
16
17  private int width = -1;
18  private int height = -1;
19
20  // Amount of border
21  private int borderTop = -1;
22  private int borderBottom = -1;
23  private int borderLeft = -1;
24  private int borderRight = -1;
25
26  public WindowInfo() {}
27
28  public void setParentInfo(int parentXOffset, int parentYOffset, int
    parentHeight, int parentWidth){
29      this.parentWidth = parentWidth;
30      this.parentHeight = parentHeight;
31      this.parentXOffset = parentXOffset;
32      this.parentYOffset = parentYOffset;
33  }
34
35  public boolean isParentInfoValid(){
36      if (parentWidth < 0 || parentHeight < 0 || parentXOffset < 0 ||
    parentYOffset < 0){
37          return false;
38      }
39
40      return true;
41  }
42
43  public void setBorder(int borderTop, int borderBottom, int borderLeft,
    int borderRight){
44      this.borderTop = borderTop;
45      this.borderBottom = borderBottom;
46      this.borderLeft = borderLeft;
47      this.borderRight = borderRight;
48  }
49
50  public boolean isBorderValid(){
51      if (borderTop < 0 || borderBottom < 0 || borderLeft < 0 ||
    borderRight < 0){
52          return false;
53      }
54
55      return true;
56  }
57
58  public void setWindowSize(int height, int width){
59      this.height = height;

```

```

60     this.width = width;
61 }
62
63 public boolean isWindowSizeValid(){
64     if (height < 0 || width < 0 ){
65         return false;
66     }
67
68     return true;
69 }
70
71 public CvRect computeROI(){
72     int newX = parentXOffset - borderLeft;
73     int newY = parentYOffset - borderTop;
74     int newWidth = width + borderLeft + borderRight;
75     int newHeight = height + borderTop + borderBottom;
76     return cvRect(newX, newY, newWidth, newHeight);
77 }
78
79 public int getWidth() {
80     return width;
81 }
82
83 public int getHeight() {
84     return height;
85 }
86
87 public int getParentWidth() {
88     return parentWidth;
89 }
90
91 public int getParentHeight() {
92     return parentHeight;
93 }
94
95 public int getParentXOffset() {
96     return parentXOffset;
97 }
98
99 public int getParentYOffset() {
100     return parentYOffset;
101 }
102 public int getBorderTop() {
103     return borderTop;
104 }
105
106 public int getBorderBottom() {
107     return borderBottom;
108 }
109
110 public int getBorderLeft() {
111     return borderLeft;
112 }

```

```
113
114     public int getBorderRight() {
115         return borderRight;
116     }
117 }
```