**Нижегородский государственный университет им. Н.И. Лобачевского**

***Разработка мультимедийных приложений с использованием библиотек OpenCV и IPP***

# Лабораторная работа
# Bag-of-words методы классификации изображений

*При поддержке компании Intel*

Половинкин А.Н.,
кафедра математического обеспечения ЭВМ

# Содержание

- Цели и задачи работы
- Обзор возможностей модуля features2d библиотеки OpenCV:
  - детекторы ключевых точек
  - дескрипторы ключевых точек
  - обучение словаря для методов класса bag-of-words
  - вычисление признакового описания изображений в методах класса bag-of-words
- Обзор возможностей модуля ml библиотеки OpenCV:
  - алгоритм обучения с учителем «случайный лес»
- Решение задачи классификации изображений с двумя категориями

# Цели работы

- Изучить bag-of-words подход к классификации изображений с использованием реализаций соответствующих функций библиотеки компьютерного зрения OpenCV.

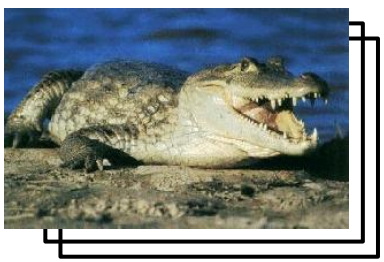
# Задачи работы

- Изучить принцип работы bag-of-words подхода для классификации изображений:
  - детектирование ключевых точек;
  - вычисление дескрипторов ключевых точек;
  - обучение словаря;
  - построение признакового описания изображения;
  - применение алгоритмов обучения с учителем.
- Рассмотреть прототипы функций, реализующих перечисленные операции в библиотеке OpenCV.
- Разработать прототип программной реализации bag-of-words подхода для классификации изображений.

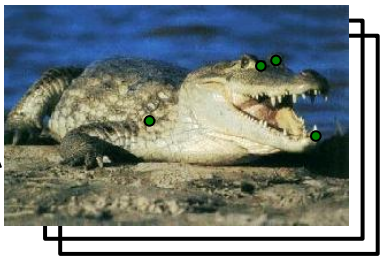
# Тестовая инфраструктура

| | |
|---|---|
| Операционная система | Microsoft Windows 7 |
| Среда разработки | Microsoft Visual Studio 2010 |
| Библиотеки OpenCV | Версия 2.4.3 |

# Bag-of-words методы классификации изображений

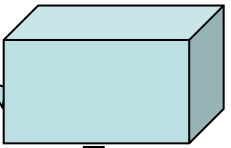


выделение ключевых точек

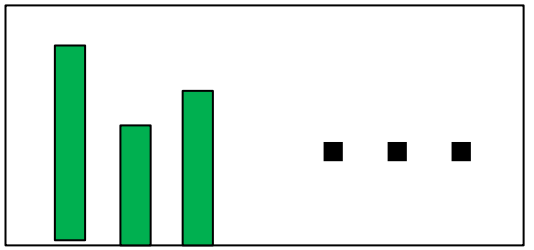


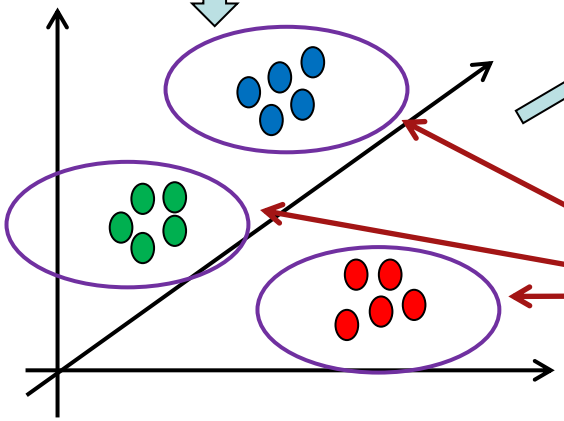
вычисление дескрипторов ключевых точек

$$\begin{pmatrix} f_{11}, \dots, f_{1n} \\ \dots \\ f_{m1}, \dots, f_{mn} \end{pmatrix}$$



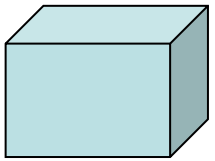
признаковое описание изображения





Обучение словаря

“слова”



Алгоритм обучения с учителем

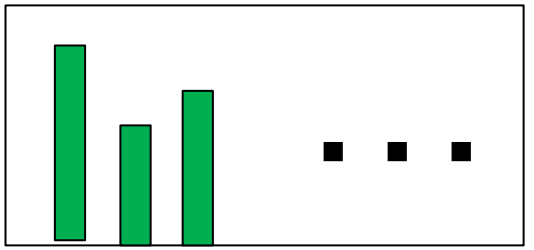
# ОБЗОР ВОЗМОЖНОСТЕЙ МОДУЛЯ FEATURES2D БИБЛИОТЕКИ OPENCV

# Bag-of-words методы классификации изображений

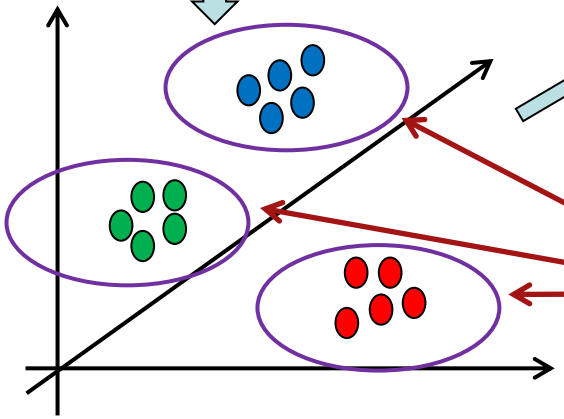

вычисление дескрипторов ключевых точек

$$\begin{pmatrix} f_{11}, \dots, f_{1n} \\ \dots \\ f_{m1}, \dots, f_{mn} \end{pmatrix}$$

признаковое описание изображения



Обучение словаря



“слова”

Алгоритм обучения с учителем

# Детектирование ключевых точек. Функции OpenCV

```
static Ptr<FeatureDetector> FeatureDetector::create(const
string& detectorType)
```

***detectorType*** – тип детектора ключевых точек ("FAST" , "STAR" , "SIFT", "SURF", "ORB", "MSER", "GFTT", "HARRIS", "Dense", "SimpleBlob")

```
void FeatureDetector::detect(const Mat& image,
vector<KeyPoint>& keypoints, const Mat& mask=Mat() )
```

***image*** – исходное изображение

***keypoints*** (выходной параметр) – массив детектированных ключевых точек

***mask*** – маска, определяющая область изображения, в которой должен выполняться поиск особых точек (данная область соответствует ненулевым элементам матрицы).

# Пример использования

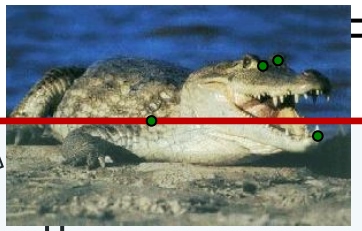
```
...
Mat img = imread(fileName);
initModule_nonfree();
Ptr<FeatureDetector> featureDetector = FeatureDetector::create("SIFT");
vector<KeyPoint> keypoints;
featureDetector->detect(img, keypoints);
...
```

При использовании детекторов SIFT и SURF необходимо дополнительно:

- подключить заголовочный файл ***opencv2/nonfree/nonfree.hpp***;
- подключить библиотеку ***opencv_nonfree242(d).lib***
- вызвать функцию инициализации модуля: ***initModule_nonfree()***;

# Bag-of-words методы классификации изображений

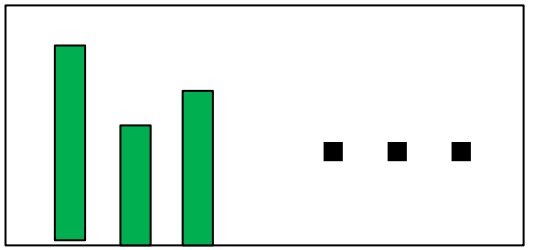
выделение ключевых точек



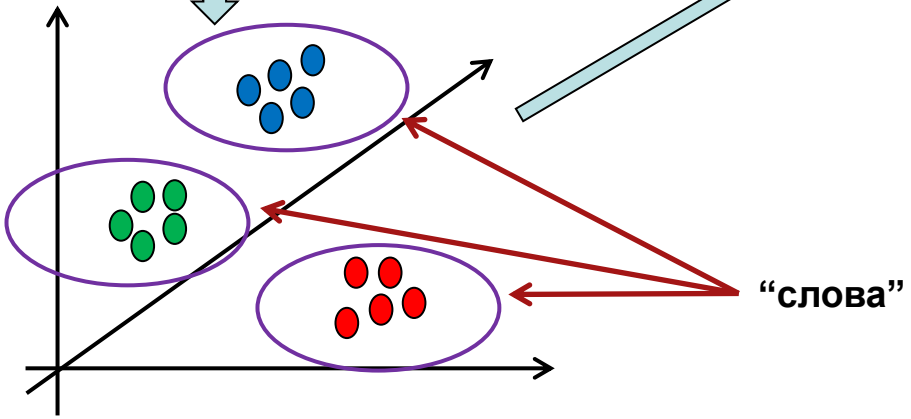
вычисление дескрипторов ключевых точек

$$\begin{pmatrix} f_{11}, \dots, f_{1n} \\ \dots \\ f_{m1}, \dots, f_{mn} \end{pmatrix}$$

признаковое описание изображения



Обучение словаря



“слова”

Алгоритм обучения с учителем

# Вычисление дескрипторов ключевых точек. Функции OpenCV

```
static Ptr<DescriptorExtractor>
DescriptorExtractor::create(const string& descriptorType)
```

***descriptorType*** – тип дескриптора ключевых точек ("SIFT", "SURF", "ORB", “BRIEF")

```
void DescriptorExtractor::compute(const Mat& image,
vector<KeyPoint>& keypoints, Mat& descriptors)
```

***image*** – исходное изображение

***keypoints*** – массив ключевых точек, в которых необходимо вычислить дескрипторы

***descriptors*** (выходной параметр) – вычисленные дескрипторы особых точек

# Пример использования

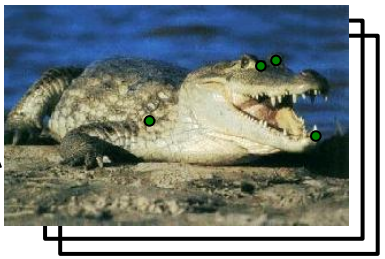
```
...
Mat img = imread(fileName);
initModule_nonfree();
Ptr<FeatureDetector> featureDetector = FeatureDetector::create("SIFT");
vector<KeyPoint> keypoints;
featureDetector->detect(img, keypoints);
Ptr<DescriptorExtractor> dExtractor = DescriptorExtractor::create("SIFT");
Mat descriptors;
dExtractor->compute(img, keypoints, descriptors);
...
```

При использовании детекторов SIFT и SURF необходимо :

- подключить заголовочный файл ***opencv2/nonfree/nonfree.hpp***;
- подключить библиотеку ***opencv_nonfree242(d).lib***
- вызвать функцию инициализации модуля: ***initModule_nonfree()***;

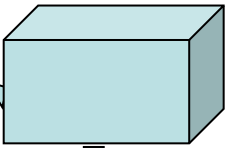
# Bag-of-words методы классификации изображений

выделение ключевых точек

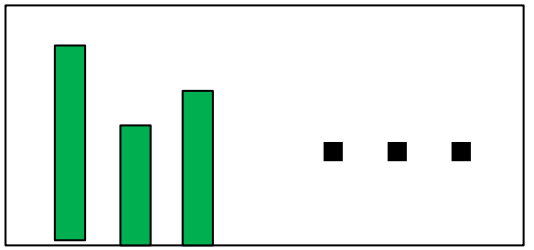


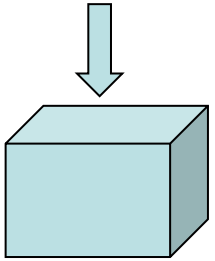
вычисление дескрипторов ключевых точек

$$\begin{pmatrix} f_{11}, \dots, f_{1n} \\ \dots \\ f_{m1}, \dots, f_{mn} \end{pmatrix}$$

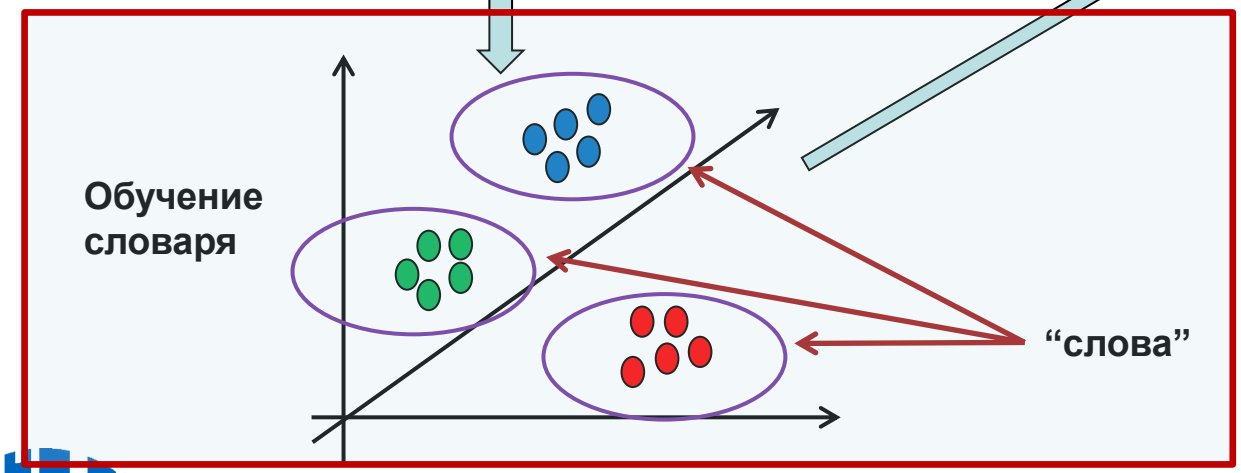


признаковое описание изображения





Алгоритм обучения с учителем



# Обучение словаря. Функции OpenCV (1)

```
BOWKMeansTrainer::BOWKMeansTrainer(int clusterCount,
const TermCriteria& termcrit=TermCriteria(), int
attempts=3, int flags=KMEANS_PP_CENTERS)
```

***clusterCount*** – число «слов» в словаре дескрипторов ключевых точек (число кластеров)

***termcrit*** – критерий остановки работы алгоритма кластеризации (максимальное число итераций или(и) по точности (центры кластеров на текущей итерации алгоритма сдвигаются на расстояние не более чем criteria.epsilon)

***attempts*** – число различным начальных инициализаций центроидов кластеров (будет выбран вариант разбиения объектов на кластеры с наилучшим значением меры компактности кластеров)

***flags*** – способ начальной инициализации центроидов

- KMEANS_RANDOM_CENTERS –инициализация случайным образом;
- KMEANS_PP_CENTERS – использовать метод Arthur и Vassilvitskii

# Обучение словаря. Функции OpenCV (2)

`void BOWTrainer::add(const Mat& descriptors)` – добавляет набор объектов к тренировочному множеству (множеству, используемому при построении словаря дескрипторов)

***descriptors*** – набор дескрипторов, добавляемых к тренировочному множеству

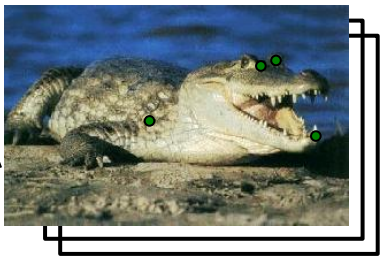
`Mat BOWTrainer::cluster()` – запускает алгоритм кластеризации, возвращает набор центроидов построенных кластеров (словарь дескрипторов)

# Пример использования

```
// descriptors – предвычисленный набор дескрипторов изображений,
//                      используемый при построении словаря
// std::vector<Mat> descriptors;
int vocSize = 100;
BOWKMeansTrainer bowTrainer(vocSize);
for (size_t i = 0; i < descriptors.size(); i++)
{
        bowTrainer.add(imgDescriptors);
}
Mat voc = bowTrainer.cluster();
```

# Bag-of-words методы классификации изображений

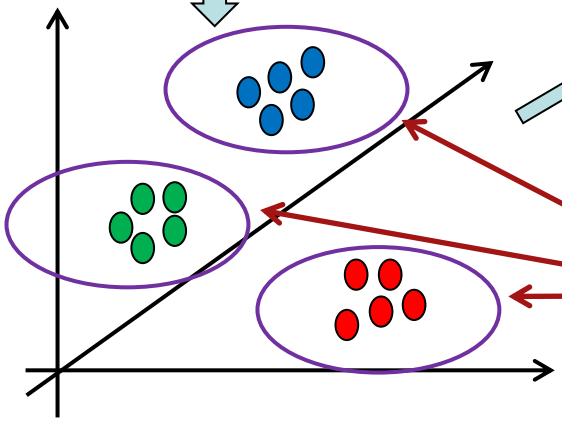
выделение ключевых точек



вычисление дескрипторов ключевых точек

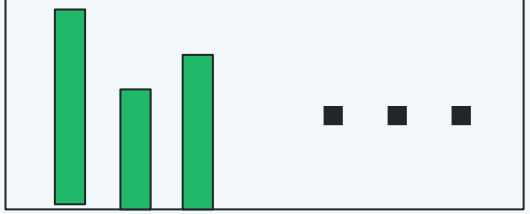
$$\begin{pmatrix} f_{11}, \dots, f_{1n} \\ \dots \\ f_{m1}, \dots, f_{mn} \end{pmatrix}$$

Обучение словаря



“слова”

признаковое описание изображения



Алгоритм обучения с учителем

# Вычисление признаковых описаний изображений. Функции OpenCV

**`class BOWImgDescriptorExtractor`** – класс, вычисляющий признаковое описание изображения. Вычисление состоит из следующих этапов:

- детектирование ключевых точек на изображении и вычисление их дескрипторов
- нахождение для каждого вычисленного дескриптора ближайшего к нему центроида кластера
- вычисление признакового описания изображения в виде нормированной гистограммы (i-ый бин гистограммы соответствует числу вхождений i-го слова из словаря в описание изображения (число дескрипторов, отнесенных к i-му кластеру)

# Вычисление признаковых описаний изображений. Функции OpenCV

```
BOWImgDescriptorExtractor::BOWImgDescriptorExtractor
(const Ptr<DescriptorExtractor>& dextractor, const
Ptr<DescriptorMatcher>& dmatcher)
```

***dextractor*** – алгоритм, вычисляющий дескрипторы в ключевых точках

***dmatcher*** – алгоритм, используемый для нахождения ближайшего центроида к текущему дескриптору ключевой точки

```
void BOWImgDescriptorExtractor::setVocabulary(const Mat&
vocabulary)
```
– устанавливает используемый словарь дескрипторов

***vocabulary*** – словарь дескрипторов (каждая строка соответствует центроиду кластера)

# Вычисление признаковых описаний изображений. Функции OpenCV

```
void BOWImgDescriptorExtractor::compute(const Mat& image,
vector<KeyPoint>& keypoints, Mat& imgDescriptor,
vector<vector<int>>* pointIdxsOfClusters=0, Mat*
descriptors=0 )
```

***image*** – исходное изображение

***keypoints*** – ключевые точки, в которых вычисляются дескрипторы

***imgDescriptor*** (выходной параметр) – вычисленное признаковое описание изображения в виде нормированной гистограммы частот встречаемости слов

***pointIdxsOfClusters*** (выходной параметр) – набор индексов дескрипторов, относящихся к тому или иному кластеру

***descriptors*** (выходной параметр) – вычисленные значения дескрипторов в ключевых точках

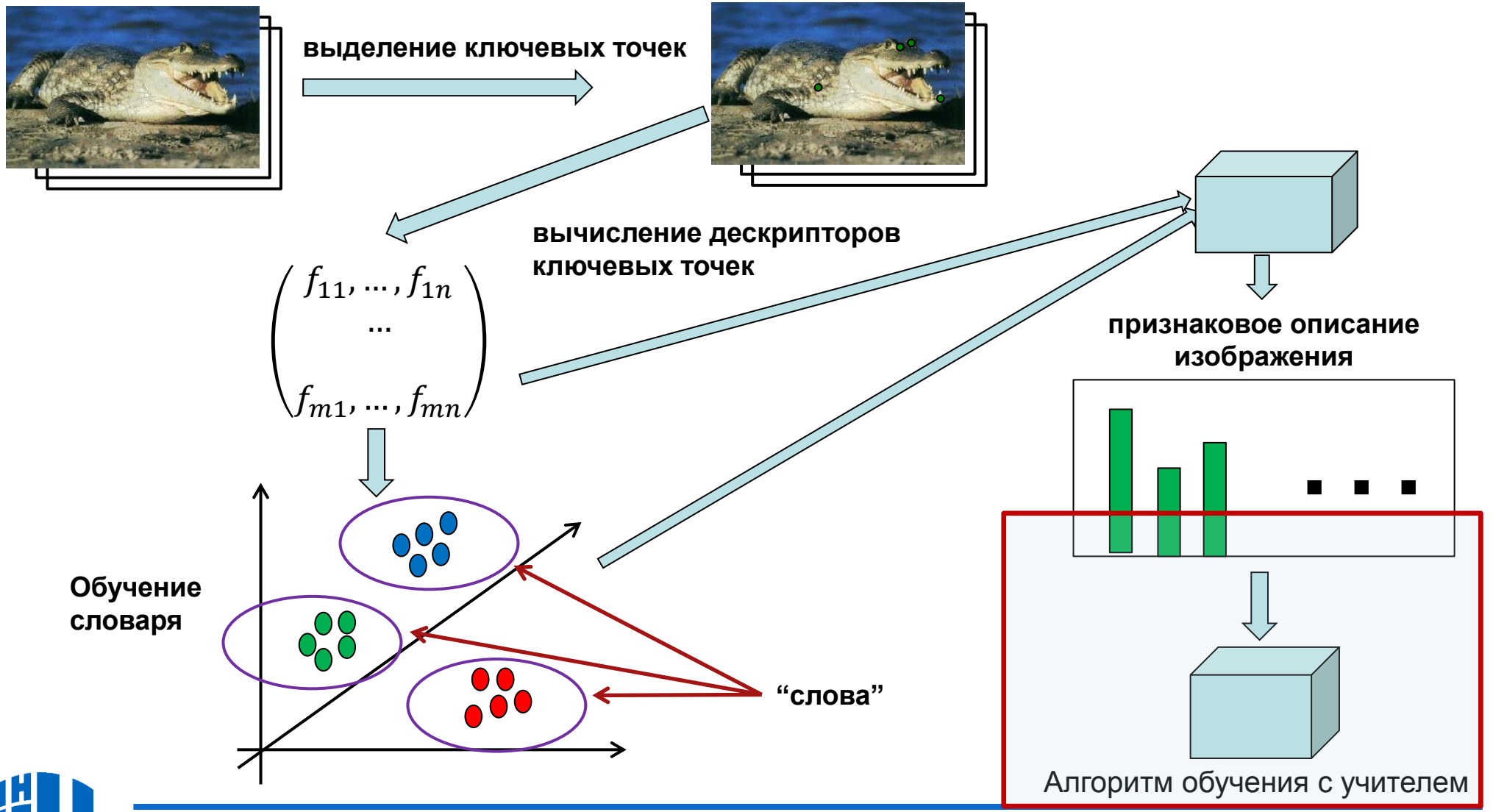
# Пример использования

```
...
// voc – предвычисленный словарь
// Mat voc;
Ptr<FeatureDetector> featureDetector = FeatureDetector::create(“SIFT”);
Ptr<DescriptorExtractor> dExtractor = DescriptorExtractor::create(“SIFT”);
Ptr<DescriptorMatcher> descriptorsMatcher =
       DescriptorMatcher::create("BruteForce");
Ptr<BOWImgDescriptorExtractor> bowExtractor = new
       BOWImgDescriptorExtractor( dExtractor, descriptorsMatcher);
bowExtractor->setVocabulary(voc);

Mat img = imread(fileName);
vector<KeyPoint> keypoints;
featureDetector->detect(img, keypoints);
Mat imgFeatures;
bowExtractor->compute(img, keypoints, imgFeatures);
...
```

# ОБЗОР ВОЗМОЖНОСТЕЙ МОДУЛЯ ML БИБЛИОТЕКИ OPENCV

# Bag-of-words методы классификации изображений



# Случайный лес. Функции OpenCV

**`CvRTParams::CvRTParams()`** – создает структуру, содержащую параметры обучения алгоритма «случайный лес» по умолчанию

```
bool CvRTrees::train(const Mat& trainData, int tflag,
const Mat& responses, const Mat& varIdx=Mat(), const Mat&
sampleIdx=Mat(), const Mat& varType=Mat(), const Mat&
missingDataMask=Mat(), CvRTParams params=CvRTParams() )
```

***trainData*** – обучающая выборка (набор векторов признаков)

***tflag*** – способ хранения данных в матрице *trainData*

- CV_ROW_SAMPLE – хранение по строкам
- CV_COL_SAMPLE – хранение по столбцам

***responses*** – обучающая выборка (набор ответов)

***varIdx*** –маска, описывающая, какие переменные должны использоваться для построения модели

# Случайный лес. Функции OpenCV

```
bool CvRTrees::train(const Mat& trainData, int tflag,
const Mat& responses, const Mat& varIdx=Mat(), const Mat&
sampleIdx=Mat(), const Mat& varType=Mat(), const Mat&
missingDataMask=Mat(), CvRTParams params=CvRTParams() )
```

***sampleIdx*** – маска, описывающая, какие объекты из обучающей выборки должны использоваться для построения модели

***varType*** – описание типов входных переменных и ответа (матрица с числом элементов, равным числу признаков + 1; элементы типа CV_8UC1, каждый элемент принимает значения; CV_VAR_ORDERED или CV_VAR_CATEGORICAL)

***missingDataMask*** – маска, описывающая переменные с отсутствующими значениями

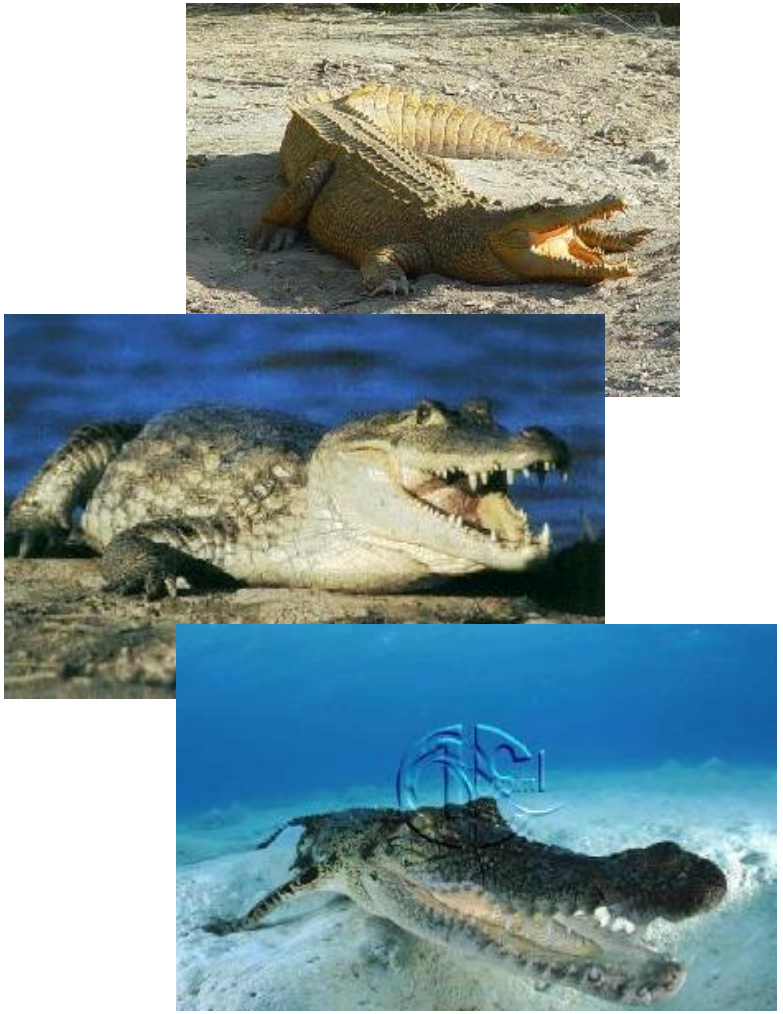
***params*** – параметры обучения алгоритма «случайный лес»

# Пример использования

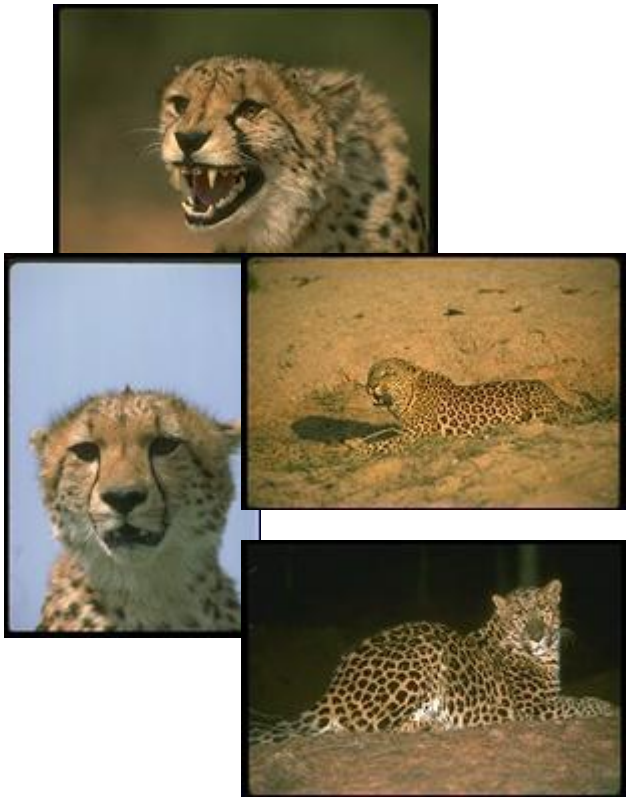
```
...
// Mat trainData;
// Mat responses;
CvRTrees* randomForest = new CvRTrees();
CvRTParams params = CvRTParams();
//устанавливаем выход исключительно по числу итераций
params.term_crit.type = CV_TERMCRIT_ITER;
params.term_crit.max_iter = 200;
int numFeatures = trainData.cols;
Mat varType(1, numFeatures + 1, CV_8UC1);
for (int i = 0; i < numFeatures; i++)
{
        // задаем тип признаков: вещественные
        varType.at<unsigned char>(i) = CV_VAR_ORDERED;
}
// задаем тип ответа: категориальный
varType.at<unsigned char>(numFeatures) = CV_VAR_CATEGORICAL;
randomForest->train(trainData, CV_ROW_SAMPLE, trainResponses, Mat(), Mat(),
varType, Mat(), params);
...
```

# ПРОТОТИП ПРОГРАММНОЙ РЕАЛИЗАЦИИ BAG-OF-WORDS ПОДХОДА ДЛЯ КЛАССИФИКАЦИИ ИЗОБРАЖЕНИЙ

# Постановка задачи

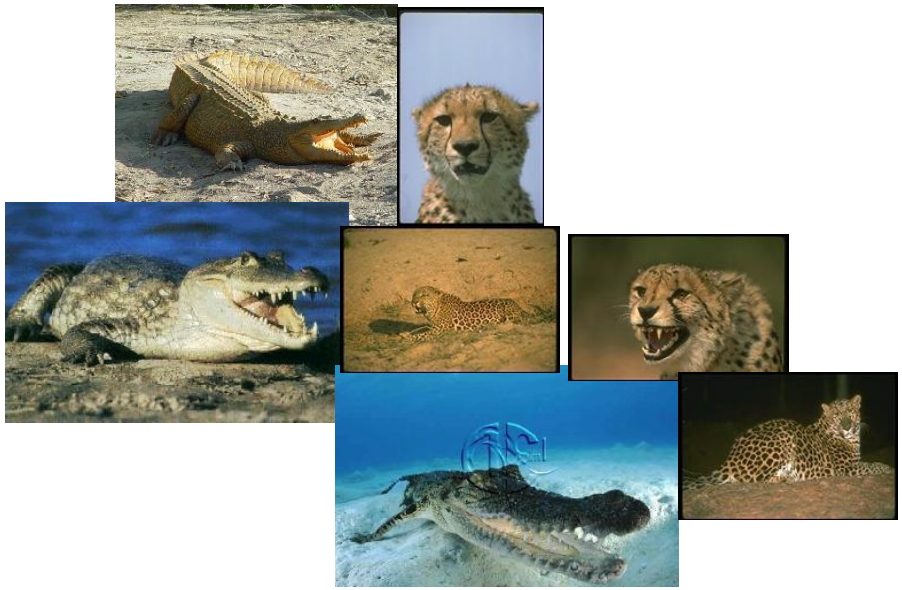


?



# Описание исходных данных

- 2 директории: ***crocodiles*** и ***leopard***, каждая из которых содержит по 50 изображений. Источник данных: Caltech-101 [http://www.vision.caltech.edu/Image_Datasets/Caltech101/]



# Постановка задачи

- Функциональность разрабатываемой программы:
  - разбиение множества изображений на обучающую и тестовую выборки;

– для тренировочной выборки:
  - нахождение  ключевых точек и вычисление их дескрипторов;
  - обучение словаря;
  - построение признакового описания изображений;
  - формирование обучающей выборки и обучение классификатора «случайный лес».

– для тестовой выборки:
  - нахождение  ключевых точек и вычисление их дескрипторов;
  - построение признакового описания изображений;
  - предсказание категории изображения с использованием обученного классификатора и вычисленных признаков;

# Входные параметры программы

- **string folder1** – путь к директории, содержащей изображения, относящиеся к первой категории
- **string folder2** – путь к директории, содержащей изображения, относящиеся ко второй категории
- **string detectorType** – тип детектора ключевых точек
- **string descriptorType** – тип дескриптора ключевых точек
- **int vocSize** – размер словаря
- **double trainProportion** – доля изображений, относящихся к обучающей выборке

# Вспомогательные функции

**`void GetFilesInFolder(const string& dirPath, std::vector<string>& filesList)`** – заполняет массив filesList списком всех *.jpg файлов из директории dirPath

**`void InitRandomBoolVector(vector<bool>& mask, double prob)`** – заполняет массив mask случайными булевскими значениями (true с вероятностью, равной prob)

# Основные функции

```
Mat TrainVocabulary(const std::vector<string>& filesList,
const std::vector<bool>& isVoc, const
Ptr<FeatureDetector>& keypointsDetector, const
Ptr<DescriptorExtractor>& descriptorsExtractor, int
vocSize)
```

***filesList*** – список файлов

***isVoc*** – маска, описывающая набор изображений из filesList, используемых для построения словаря

***keypointsDetector*** – детектор ключевых точек

***descriptorsExtractor*** – алгоритм, используемый для вычисления дескрипторов ключевых точек

***vocSize*** – размер словаря

# Основные функции (1)

```
Mat TrainVocabulary(const std::vector<string>& filesList,
const std::vector<bool>& isVoc, const
Ptr<FeatureDetector>& keypointsDetector, const
Ptr<DescriptorExtractor>& descriptorsExtractor, int
vocSize)
```

***filesList*** – список файлов

***isVoc*** – маска, описывающая набор изображений из filesList, используемых для построения словаря

***keypointsDetector*** – детектор ключевых точек

***descriptorsExtractor*** – алгоритм, используемый для вычисления дескрипторов ключевых точек

***vocSize*** – размер словаря

# Основные функции (2)

```
Mat ExtractFeaturesFromImage(Ptr<FeatureDetector>
keypointsDetector, Ptr<BOWImgDescriptorExtractor>
bowExtractor, const string& fileName)
```

***keypointsDetector*** – детектор ключевых точек

***bowExtractor*** – алгоритм, используемый для вычисления признакового описания изображения

***fileName*** – входной файл

# Основные функции (3)

```
void ExtractTrainData(const std::vector<string>&
filesList, const std::vector<bool>& isTrain, const Mat&
responses, const Ptr<FeatureDetector>& keypointsDetector,
const Ptr<BOWImgDescriptorExtractor>& bowExtractor, Mat&
trainData, Mat& trainResponses)
```

***filesList*** – список файлов

***isTrain*** – маска, описывающая набор изображений из filesList, используемых для обучения классификатора «случайный лес»

***responses*** – ответы (категории) для файлов из filesList

***keypointsDetector*** – детектор ключевых точек

***bowExtractor*** – алгоритм, используемый для вычисления признакового описания изображения

***trainData*** (выходной параметр) – матрица, содержащая признаковые описания изображений из обучающей выборки

***trainResponses*** (выходной параметр) – матрица, содержащая ответы (категории) для изображений из обучающей выборки

# Основные функции (4)

**`Ptr<CvRTrees> TrainClassifier(const Mat& trainData, const Mat& trainResponses)`** – возвращает обученный классификатор «случайный лес»

***trainData*** (выходной параметр) – матрица, содержащая признаковые описания изображений из обучающей выборки

***trainResponses*** (выходной параметр) – матрица, содержащая ответы (категории) для изображений из обучающей выборки

# Основные функции (5)

```
float Predict(const Ptr<FeatureDetector>
keypointsDetector, const Ptr<BOWImgDescriptorExtractor>
bowExtractor, const Ptr<CvRTrees> classifier, const
string& fileName)
```
– возвращает предсказанную категорию для изображения

***keypointsDetector*** – детектор ключевых точек

***bowExtractor*** – алгоритм, используемый для вычисления признакового описания изображения

***classifier*** – обученный классификатор («случайный лес»)

***fileName*** – входной файл

# Основные функции (6)

```
Mat PredictOnTestData(const std::vector<string>&
filesList, const std::vector<bool>& isTrain, const
Ptr<FeatureDetector> keypointsDetector, const
Ptr<BOWImgDescriptorExtractor> bowExtractor, const
Ptr<CvRTrees> classifier)
```
– возвращает набор предсказанных значений для тестовой выборки

***filesList*** – список файлов

***isTrain*** – маска, описывающая набор изображений из filesList, используемых для обучения

***keypointsDetector*** – детектор ключевых точек

***bowExtractor*** – алгоритм, используемый для вычисления признакового описания изображения

***classifier*** – обученный классификатор («случайный лес»)

# Основные функции (7)

**`Mat GetTestResponses(const Mat& responses, const vector<bool>& isTrain)`** – возвращает набор ответов (категорий) для изображений из тестовой выборки

***responses*** – ответы (категории) для всех изображений

***isTrain*** – маска, описывающая набор изображений, используемых для обучения

**`float CalculateMisclassificationError(Mat& responses, Mat& predictions)`** – возвращает ошибку классификации

***responses*** – правильные ответы (категории) для всех изображений

***predictions*** – предсказанные ответы (категории) для всех изображений

# Структура проекта

- **04_BowImageClassification.sln** – решение:
  - **BowImageClassification** – основное приложение

- Значения входных параметров:
  - folder1 – путь к папке crocodiles
  - folder2 – путь к папке Leopard
  - detectorType – “SIFT”
  - descriptorType – “SIFT”
  - vocSize – 25
  - trainProportion – 0.5

# Задания для самостоятельной работы

- Добавьте в разработанное приложение вывод информации о изображениях из тестовой выборки, которые были неправильно классифицированы.
- Добавьте в разработанное приложение возможность использования в качестве используемого классификатора машины опорных векторов с ядром типа Radial Basis Function. Сравните результаты с ранее реализованным подходом.

# Задания для самостоятельной работы

- Выполните исследование зависимости ошибки классификации от используемых параметров (типа используемых детекторов и дескрипторов ключевых точек, числа слов в словаре, параметров алгоритма обучения с учителем «случайный лес»: числа деревьев в ансамбле, максимальной глубины деревьев, входящих в ансамбль).
- Реализуйте построение словаря на основе Gaussian Mixture Model и сравните полученные результаты с ранее реализованным подходом.

# Авторский коллектив


- Половинкин Алексей Николаевич,
  м.н.с. НИЛ кафедры
  Математического обеспечения ЭВМ факультета ВМК ННГУ

  alexey.polovinkin@gmail.com