

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ»

В. И. Костылев,
Ю. С. Левицкая

**ОБРАБОТКА И АНАЛИЗ ИЗОБРАЖЕНИЙ
С ПОМОЩЬЮ ОБУЧЕНИЯ
НЕЙРОННЫХ СЕТЕЙ**

Учебное пособие

Воронеж
Издательский дом ВГУ
2019

Утверждено научно-методическим советом физического факультета ВГУ
24 апреля 2019 г., протокол № 4

Рецензент – доктор технических наук, профессор кафедры информационных технологий управления, доцент В.А. Дурденко

Подготовлено на кафедре электроники физического факультета Воронежского государственного университета.

Рекомендовано студентам магистратуры физического факультета Воронежского государственного университета.

Для направления 03.04.03 – Радиофизика

Содержание

Введение	5
1. Одиночное изображение с высоким разрешением с использованием глубокого обучения нейронной сети VDSR.....	8
1.1 Сеть VDSR	9
1.2 Загрузка учебных и тестовых данных	11
1.3 Определение мини-пакетного хранилища данных для обучения.....	12
1.4 Настройка слоев VDSR	13
1.5 Параметры обучения	15
1.6 Обучение сети	16
1.7 Выполнение одноразового супер разрешения с использованием сети VDSR.....	17
1.8 Создание образца изображения с низким разрешением.....	17
1.9 Улучшение разрешения изображения с использованием бикубической интерполяции	20
1.10 Улучшение разрешения изображения с помощью Pretrained VDSR NetWork	21
1.11 Визуальное и количественное сравнение.....	24
1.12 Резюме.....	26
2. Деблокирование изображения JPEG с использованием Deep Learnin, обучение шумоизолирующей сверхточной сети (DnCNN)	28
2.1 Сеть DnCNN	29
2.2 Загрузка данных обучения	30
2.3 Подготовка данных обучения	31
2.4 Определение мини-пакетного хранилища данных для обучения.....	32
2.5 Настройка слоев DnCNN.....	33
2.6 Варианты обучения.....	36
2.7 Обучение сети	37

2.8 Выполнение деблокирования JPEG с использованием сети DnCNN	38
2.9 Создание образцовых изображений с блокирующими артефактами	38
2.10 Сжатые изображения предпроцесса	41
2.11 Применение сети DnCNN	42
2.12 Количественное сравнение	44
2.13 Резюме	46
3. Сегментация изображений	48
3.1 Обнаружение ячейки с помощью сегмента изображений	48
3.2 Сегментация с контролируемым маркером	53
3.3 Семантическая сегментация мультиспектральных изображений с использованием глубокого обучения нейронной сети U-Net.....	65
3.3.1 Загрузка данных	66
3.3.2 Проверка данных обучения	67
3.3.3 Определение мини-пакетного хранилища для обучения	73
3.3.4 Создание сетевых сетей U-Net	74
3.3.5 Варианты обучения	78
3.3.6 Обучение сети	79
3.3.7 Предсказание результатов тестирования данных	80
3.3.8 Определение точности сегментации.....	85
3.3.9 Рассчет объема растительного покрова.....	85
3.3.10 Резюме.....	86
Заключение	87
Литература	88

Введение

Одним из наиболее эффективных инструментов для распознавания образов являются системы, построенные на искусственных нейронных сетьях. Это обусловлено тем, что способ обработки информации искусственной нейросети приближен к работе мозга человека, который в корне отличается от методов, применяемых обычными цифровыми компьютерами. Отличия заключаются в том, что мозг представляет собой чрезвычайно сложный, нелинейный, параллельный компьютер (систему обработки информации). Он обладает способностью организовывать свои структурные компоненты, называемые нейронами, так, чтобы они могли выполнять конкретные задачи (такие как распознавание образов, обработку сигналов органов чувств, моторные функции) во много раз быстрее, чем могут позволить самые быстродействующие современные компьютеры. Примером такой задачи обработки информации может служить обычное зрение. В функции зрительной системы входит создание представления окружающего мира в таком виде, который обеспечивает возможность взаимодействия с этим миром. Более точно, мозг последовательно выполняет ряд задач распознавания (например, распознавание знакомого лица в незнакомом окружении). На это у него уходит около 100-200 миллисекунд, в то время как выполнение аналогичных задач даже меньшей сложности на компьютере может занять несколько дней.

В первой главе показано, как обучать нейронную сеть Very-Deep Super-Resolution (VDSR), а затем использовать сеть VDSR для оценки изображения с высоким разрешением с одного изображения с низким разрешением. Также показано, как обучать сеть VDSR, а также предоставляет предварительно подготовленную сеть VDSR. Если вы решили обучить сеть VDSR, настоятельно рекомендуется использовать графический процессор

NVIDIA™ с поддержкой CUDA с вычислительной способностью 3.0 или выше. Для использования графического процессора требуется Parallel Computing Toolbox™.

Если вы не хотите загружать набор учебных данных или обучать сеть, вы можете загрузить предварительно подготовленную сеть VDSR, введя `load ('trainedVDSR-Epoch-100-ScaleFactors-234.mat')`; в командной строке. Затем перейдите непосредственно к разделу «Выполнение одиночного изображения с высоким разрешением с использованием раздела VDSR» в этом примере.

Во второй части методического пособия показан пример, как обучать шумоизолирующую сверхточную нейронную сеть (DnCNN), а затем использовать сеть для уменьшения артефактов сжатия JPEG в изображении.

В этом примере показано, как обучить сеть DnCNN, а также предоставить предварительно определенную сеть DnCNN. Если вы решите обучить сеть DnCNN, рекомендуется использовать графический процессор NVIDIA™ с поддержкой CUDA с вычислительной способностью 3.0 или выше (требуется Parallel Computing Toolbox™).

Если вы не хотите загружать набор учебных данных или обучать сеть, вы можете загрузить предварительно назначенную сеть DnCNN, введя `load('pretrainedJPEGDnCNN.mat')` в командной строке. Затем в этом примере перейдите непосредственно к разделу «Выполнение деблокирования JPEG с использованием сети DnCNN» .

Третья часть методического пособия посвящена сегментации изображений, где в подпункте 3.1 приведен пример как обнаружить ячейку с помощью обнаружения края и базовой морфологии. Объект может быть легко обнаружен в изображении, если он достаточно контрастен по сравнению с фоном. В этом примере, как ячейки, используются раковые клетки простаты

В разделе 3.2 рассмотрен пример, как использовать сегментацию водораздела для разделения касательных объектов на изображении. Преобразование водосбора обнаруживает «водосборные бассейны» и «линии водораздельных гребней» в изображении, рассматривая его как поверхность, где светлые пиксели высокие, а темные - низкие.

В разделе 3.3 приведен пример, где показано, как организовать сверточную нейронную сеть U-Net для выполнения семантической сегментации мультиспектрального изображения с семью каналами: трех цветовых каналов, трех ближне-инфракрасных каналов и маски.

В этом примере показано, как обучать сеть U-Net, а также предоставляет предварительно подготовленную сеть U-Net. Если вы решите обучить сеть U-Net, настоятельно рекомендуется использовать графический процессор NVIDIA™ с поддержкой CUDA с вычислительной способностью 3.0 или выше (требуется Parallel Computing Toolbox™).

1. Одиночное изображение с высоким разрешением с использованием глубокого обучения нейронной сети VDSR

Супер разрешение - это процесс создания изображений с высоким разрешением изображений с низким разрешением. В этом примере рассматривается одноразрядное супер разрешение (SISR), целью которого является восстановление одного изображения с высоким разрешением по одному изображению с низким разрешением. SISR является сложной задачей, поскольку содержимое высокочастотного изображения обычно не может быть восстановлено из изображения с низким разрешением. Без высокочастотной информации качество изображения с высоким разрешением ограничено. Кроме того, SISR является некорректной задачей, поскольку изображение с низким разрешением может давать несколько возможных изображений с высоким разрешением.

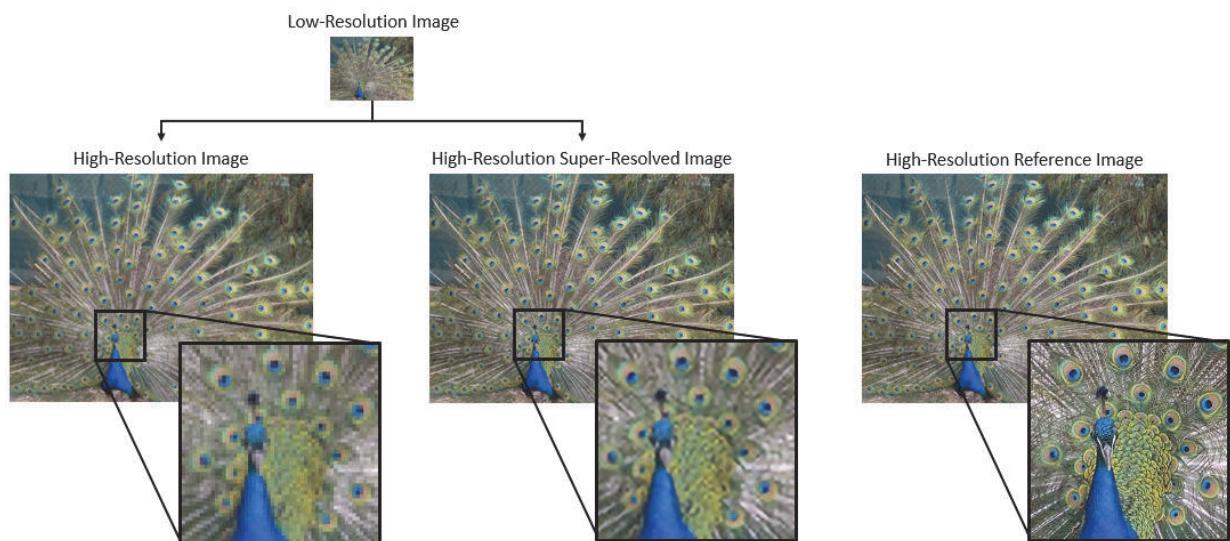


Рис. 1.1. Восстановление одного изображения с высоким разрешением по одному изображению с низким разрешением

Для выполнения SISR было предложено несколько методов, включая алгоритмы глубокого обучения. В этом примере рассматривается один глубокий алгоритм обучения для SISR, называемый очень глубоким супер разрешением (VDSR).

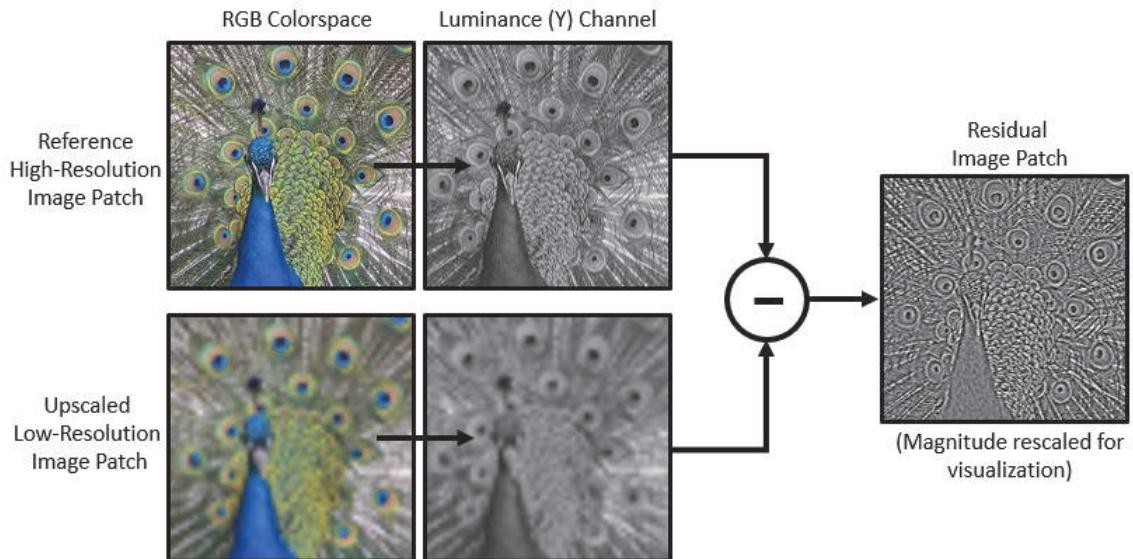
1.1 Сеть VDSR

VDSR - это сверточная нейронная сетевая архитектура, предназначенная для выполнения одного разрешения с высоким разрешением. Сеть VDSR изучает сопоставление изображений с низким и высоким разрешением. Это сопоставление возможно потому, что изображения с низким разрешением и высоким разрешением имеют одинаковое изображение и отличаются в основном высокочастотными деталями.

VDSR использует остаточную стратегию обучения, а это означает, что сеть учится оценивать остаточное изображение. В контексте сверхразрешения остаточное изображение представляет собой разницу между опорным изображением высокого разрешения и изображением с низким разрешением, которое было масштабировано с использованием бикубической интерполяции, чтобы соответствовать размеру эталонного изображения. Остаточное изображение содержит информацию о высокочастотных деталях изображения.

Сеть VDSR обнаруживает остаточное изображение из яркости цветного изображения. Канал яркости изображения Y представляет собой яркость каждого пикселя через линейную комбинацию значений красного, зеленого и синего пикселей. Напротив, два канала цветности изображения, C_b и C_r, представляют собой различные линейные комбинации красных, зеленых и синих значений пикселей, которые представляют информацию о различности цветов. VDSR обучается с использованием только канала яркости,

потому что восприятие человека более чувствительно к изменениям яркости, чем к изменениям цвета.



Rис. 1.2. Патч остаточного изображения

Если Y_{highres} есть яркость изображения с высоким разрешением, а Y_{lowres} яркость - изображение с низким разрешением, которое было масштабировано с использованием бикубической интерполяции, то вход в сеть VDSR является Y_{lowres} и сеть учится прогнозировать $Y_{\text{residual}} = Y_{\text{highres}} - Y_{\text{lowres}}$ данные обучения.

После того, как сеть VDSR научится оценивать остаточное изображение, вы можете восстановить изображения с высоким разрешением, добавив оценочное остаточное изображение в изображение с высоким разрешением с высоким разрешением, а затем преобразуя изображение обратно в цветовое пространство RGB.

Масштабный коэффициент связывает размер эталонного изображения с размером изображения с низким разрешением. По мере увеличения масштабного коэффициента SISR становится более беспорядочным, потому что

изображение с низким разрешением теряет больше информации о высокочастотном изображении. VDSR решает эту проблему, используя большое восприимчивое поле. В этом примере создается сеть VDSR с несколькими масштабными коэффициентами с использованием увеличения масштаба. Увеличение шкалы улучшает результаты при более масштабных коэффициентах, потому что сеть может использовать контекст изображения из-за меньших масштабных коэффициентов. Кроме того, сеть VDSR может обобщать, чтобы принимать изображения с межзвездными факторами.

1.2 Загрузка учебных и тестовых данных

Загрузите тестовый тест IAPR TC-12, который состоит из 20 000 неподвижных изображений. Набор данных включает фотографии людей, животных, городов и многое другое. Вы можете использовать вспомогательную функцию, downloadIAPRTC12Data чтобы загрузить данные. Размер файла данных составляет ~ 1,8 ГБ.

```
imagesDir = tempdir;
url = 'http://www-i6.informatik.rwth-aachen.de/imageclef/resources/iaprtc12.tgz';
downloadIAPRTC12Data(url,imagesDir);
```

В этом примере будет проведена тренировка сети с небольшим подмножеством данных Benchmark IAPR TC-12. Загрузите данные обучения imageCLEF. Все изображения - 32-битные цветные изображения JPEG.

```
trainImagesDir = fullfile(imagesDir,'iaprtc12','images','02');
exts = {'.jpg','.bmp','.png'};;
trainImages = imageDatastore(trainImagesDir,'FileExtensions',exts);
```

Перечислите количество обучающих изображений:

```
numel (trainImages.Files)
ans = 616.
```

1.3 Определение мини-пакетного хранилища данных для обучения

Мини-пакетное хранилище данных используется для передачи данных обучения в сеть. В этом примере определяется пользовательская реализация мини-пакетного хранилища данных, называемая vdsrImagePatchDatastore как удобный способ создания дополненных патчей изображений для обучения сети VDSR.

Эти vdsrImagePatchDatastore экстракты заплаты из входных изображений с низким разрешением и масштабирует заплаты на различных масштабных коэффициентов. Далее, хранилище данных дополняет патчи с использованием поворотов и отражений вокруг x-направления. Усовершенствование данных полезно для увеличения объема данных обучения. Наконец, datastpre создает остаточные патчи изображения, соответствующие каждому дополненному патчу изображения. Патчи с изображениями низкого разрешения действуют как сетевой вход. Остальные исправления являются желаемым выходом сети.

Каждая мини-партия содержит 64 патча размером 41 на 41 пиксель (выбор размера патча позже объясняется при настройке уровней VDSR). Только одна мини-партия будет извлекаться из каждого изображения во время обучения, и все патчи будут извлечены из случайных позиций на изображениях. Установите 'BatchesPerImage' количество случайных патчей, которые нужно извлечь на изображение за мини-пакет. Чтобы обучить сеть с множеством различных коэффициентов, установите 'ScaleFactor' на [2 3 4].

```
miniBatchSize = 64;
scaleFactors = [2 3 4];
source = vdsrImagePatchDatastore(trainImages, ...
    'MiniBatchSize', miniBatchSize, ...
    'PatchSize', 41, ...
    'BatchesPerImage', 1, ...
    'ScaleFactor', scaleFactors);
```

VdsrImagePatchDatastore обеспечивает мини-порций данных в сети на каждой итерации эпохи. Выполните операцию чтения в хранилище данных для изучения данных.

```
inputBatch = read(source);
summary(inputBatch)
```

Переменные:

lowResolutionPatches: ячейка 64×1

остатки: ячейка 64×1

1.4 Настройка слоев VDSR

В этом примере сеть VDSR использует 41 отдельный слой из Neural Network Toolbox™, включая:

- imageInputLayer - уровень ввода изображения;
- convolution2dLayer - 2-мерный слой свертки для сверточных нейронных сетей;
- reluLayer - слой Rectified linear unit (ReLU);
- regressionLayer - регрессионный выходной слой для нейронной сети.

Первый слой, imageInputLayer работает с изображениями. Размер патча основан на сетевом восприимчивом поле, которое представляет собой область пространственного изображения, которая влияет на отклик самого верхнего уровня в сети. В идеальном случае сетевое восприимчивое поле совпадает с размером изображения, чтобы оно могло видеть все высоковысокие функции изображения. В этом случае для сети глубины D восприимчивое поле равно $(2D + 1) \text{-by- } (2D + 1)$. Поскольку VDSR представляет собой 20-слойную сеть, допустимое поле и размер патча изображения составляют 41 на 41. Уровень ввода изображения принимает изображения с одним каналом, потому что VDSR обучается с использованием только канала яркости.

```

networkDepth = 20;
firstLayer =
er = imageInputLayer([41 41 1], 'Name', 'InputLayer', 'Normalization', 'none';

```

За слоем ввода изображения следует двумерный сверточный слой, который содержит 64 фильтра размером 3 на 3. Размер мини-партии определяет количество фильтров. Zero-pad вводит входы для каждого сверточного слоя, так что карты функций остаются теми же размерами, что и вход после каждой свертки. Он инициализирует весовые коэффициенты случайными значениями, так что существует асимметрия в обучении нейронов. За каждым сверточным слоем следует слой ReLU, который вводит нелинейность в сети.

```

convolutionLayer = convolution2dLayer(3,64,'Padding',1, ...
    'Name','Conv1');
convolutionLayer.Weights = sqrt(2/(9*64))*randn(3,3,1,64);
convolutionLayer.Bias = zeros(1,1,64);

```

Укажите слой ReLU.

```
reluLayer = reluLayer('Name','ReLU1');
```

Средние слои содержат 18 чередующихся сверточных и выпрямленных линейных единичных слоев. Каждый сверточный слой содержит 64 фильтра размером 3 на 3 на 64, где фильтр работает в пространственной области 3 на 3 по 64 каналам. Он инициализирует весовые коэффициенты случайными значениями. Как и прежде, слой ReLU следует за каждым сверточным слоем.

```

middleLayers = [convolutionLayer reluLayer];
for layerNumber = 2:networkDepth-1
    conv2dLayer = convolution2dLayer(3,64, ...
        'Padding',[1 1],...
        'Name',[ 'Conv' num2str(layerNumber)]);
    % He initialization
    conv2dLayer.Weights = sqrt(2/(9*64))*randn(3,3,64,64);
    conv2dLayer.Bias = zeros(1,1,64);

    reluLayer = reluLayer('Name',[ 'ReLU' num2str(layerNumber)]);

```

```
middleLayers = [middleLayers conv2dLayer relLayer];
end
```

Предпоследний слой представляет собой сверточный слой с единственным фильтром размером 3x3x64, который восстанавливает изображение.

```
conv2dLayer = convolution2dLayer(3,1, ...
    'NumChannels',64, ...
    'Padding',[1 1], ...
    'Name',[ 'Conv' num2str(networkDepth)]);
conv2dLayer.Weights = sqrt(2/(9*64))*randn(3,3,64,1);
conv2dLayer.Bias = zeros(1,1,1);
```

Последний слой является регрессионным слоем, а не уровнем ReLU. Уровень регрессии вычисляет среднеквадратичную ошибку между остаточным изображением и прогнозом сети.

```
finalLayers = [conv2dLayer
regressionLayer('Name','FinalRegressionLayer')];
Объединение всех слоев для создания сети VDSR.
layers = [firstLayer middleLayers finalLayers];
Кроме того, вы можете использовать эту вспомогательную функцию для
создания слоев VDSR.
layers = vdsrLayers();
```

1.5 Параметры обучения

Изучение сети с использованием стохастического градиентного спуска с оптимизацией импульса (SGDM). Задайте параметры гиперпараметра для SDGM с помощью `trainingOptions` функции.

Обучение глубокой сети занимает много времени. Ускорьте обучение, указав высокую скорость обучения. Однако это может привести к тому, что градиенты сети будут взрываться или расти неуправляемо, что препятствует успешному обучению сети. Чтобы сохранить градиенты в значимом диапазоне, включите градиентную обрезку, установив '`GradientThreshold`' значение 0,01 и укажите '`GradientThresholdMethod`' чтобы использовать L2-норму градиентов.

```
maxEpochs = 100;
epochIntervals = 1;
```

```

initLearningRate = 0.1;
learningRateFactor = 0.1;
l2reg = 0.0001;
options = trainingOptions('sgdm',...
    'Momentum',0.9,...
    'InitialLearnRate',initLearningRate,...
    'LearnRateSchedule','piecewise',...
    'LearnRateDropPeriod',10,...
    'LearnRateDropFactor',learningRateFactor,...
    'L2Regularization',l2reg,...
    'MaxEpochs',maxEpochs,...
    'MiniBatchSize',miniBatchSize,...
    'GradientThresholdMethod','l2norm',...
    'GradientThreshold',0.01);

```

1.6 Обучение сети

После настройки параметров обучения и мини-пакетного хранилища данных с помощью этой `trainNetwork` функции обучите сеть VDSR. Для обучения сети задайте `doTraining` параметр в следующем коде `true`. Для обучения рекомендуется использовать графический процессор NVIDIA™ с поддержкой CUDA с вычислительной способностью 3,0 или выше.

Если вы сохраните `doTraining` параметр в следующем коде как `false`, тогда пример возвращает предварительно подготовленную сеть VDSR, которая была обучена для супер-разрешения изображений для коэффициентов масштабирования 2, 3 и 4.

```

doTraining = false;
if doTraining
    modelDateTime = datestr(now,'dd-mmm-yyyy-HH-MM-SS');
    net = trainNetwork(source,layers,options);
    save(['trainedVDSR-' modelDateTime '-Epoch-' num2str(maxEpochs*
        *epochIntervals) 'ScaleFactors-
    ' num2str(234) '.mat'],'net','options');
else
    load('trainedVDSR-Epoch-100-ScaleFactors-234.mat');
end

```

1.7 Выполнение одноразового супер разрешения с использованием сети VDSR

Чтобы выполнить одноразовое супер разрешение (SISR) с использованием сети VDSR, выполните остальные шаги этого примера. В оставшейся части примера показано, как:

Создание с низким разрешением образца изображения из высокого разрешения опорного изображения.

Выполните SISR на изображении с низким разрешением, используя бикубическую интерполяцию, традиционное решение для обработки изображений, которое не полагается на глубокое обучение.

Выполните SISR на изображении с низким разрешением, используя нейронную сеть VDSR.

Визуально сравнить восстановленные изображения с высоким разрешением с использованием бикубической интерполяции и VDSR

Оценивается качество супер-разрешенных изображений путем количественной схожесть изображений эталонного изображения с высоким разрешением.

1.8 Создание образца изображения с низким разрешением

Создайте изображение с низким разрешением, которое будет использоваться для сравнения результатов суперразрешения с использованием глубокого обучения для результата с использованием традиционных методов обработки изображений, таких как бикубическая интерполяция. Набор тестовых данных testImages содержит 21 неискаженное изображение, загруженное в Image Processing Toolbox™. Загрузите изображения в imageDatastore.

```
exts = {'.jpg','.png'};  
fileNames = {'sherlock.jpg','car2.jpg','fabric.png','greens.jpg','hands1.jpg','kobi.png',...}
```

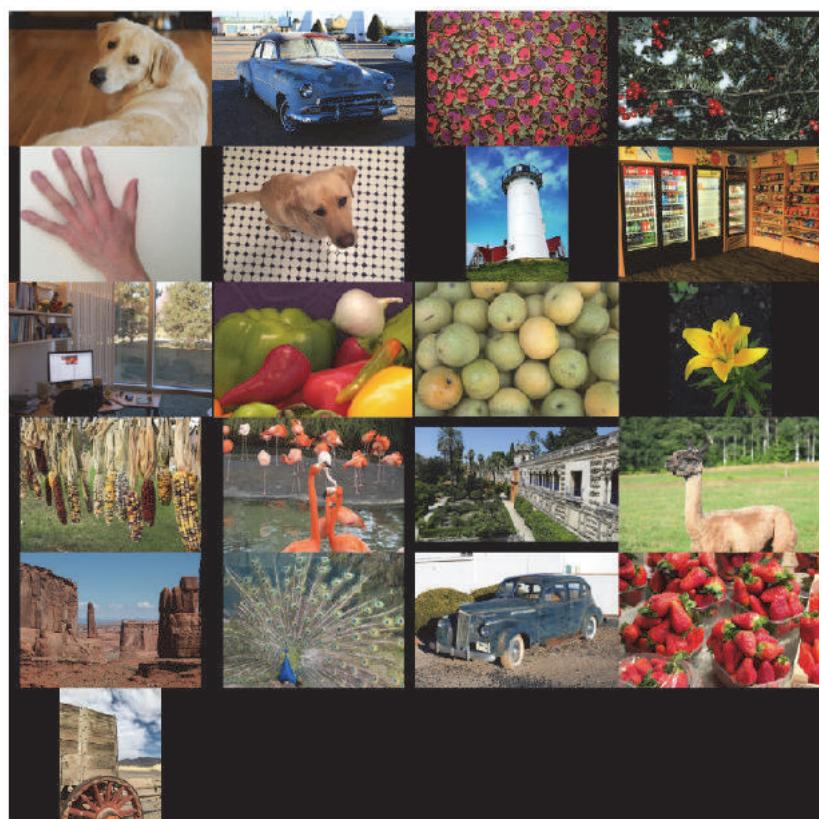
```

'lighthouse.png','micromarket.jpg','office_4.jpg','onion.png','pears.
png','yellowlily.jpg',...
'indiancorn.jpg','flamingos.jpg','sevilla.jpg','llama.jpg','parkavenu
e.jpg',...
    'peacock.jpg','car1.jpg','strawberries.jpg','wagon.jpg'}];
filePath  = [fullfile(matlabroot,'toolbox','images','imdata') file-
sep];
filePathNames = strcat(filePath,fileNames);
testImages = imageDatastore(filePathNames,'FileExtensions',exts);

```

Отображение тестовых изображений в качестве монтажа.

```
montage(testImages)
```



Rис. 1.3. Изображения, загруженные в imageDatastore

Выберите одно из изображений, которое будет использоваться в качестве эталонного изображения для супер разрешения. Вы можете по желанию использовать свое собственное изображение с высоким разрешением в качестве эталонного изображения.

```
indx = 1; % Index of image to read from the test image datastore  
Ireference = readimage(testImages,indx);  
Ireference = im2double(Ireference);  
imshow(Ireference)  
title('High-Resolution Reference Image')
```

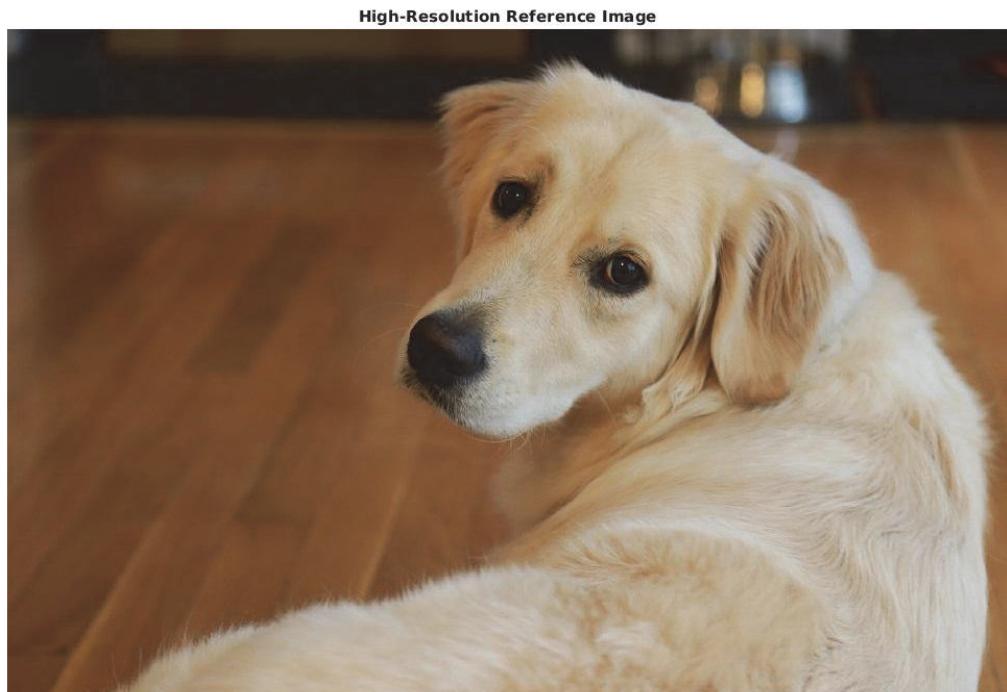


Рис. 1.4. Выбранное изображение для супер разрешения

Создать версию с низким разрешением опорного изображения высокого разрешения с использованием `imreslice` с коэффициентом масштабирования 0,25. Высокочастотные компоненты изображения теряются во время уменьшения масштаба.

```
scaleFactor = 0.25;  
Ilowres = imresize(Ireference,scaleFactor,'bicubic');
```

```
imshow(Ilowres)
title('Low-Resolution Image')
```



Рис. 1.5. Использование imresizec с коэффициентом масштабирования 0,25

1.9 Улучшение разрешения изображения с использованием бикубической интерполяции

Стандартный способ увеличения разрешения изображения без глубокого обучения - использовать бикубическую интерполяцию. Высококлассное низкое разрешение изображения с использованием бикубических интерполяций, так что в результате изображение с высоким разрешением имеет тот же размер в качестве опорного изображения.

```
[nrows,ncols,np] = size(Ireference);
Ibicubic = imresize(Ilowres,[nrows ncols],'bicubic');
imshow(Ibicubic)
title('High-Resolution Image Obtained Using Bicubic Interpolation')
```

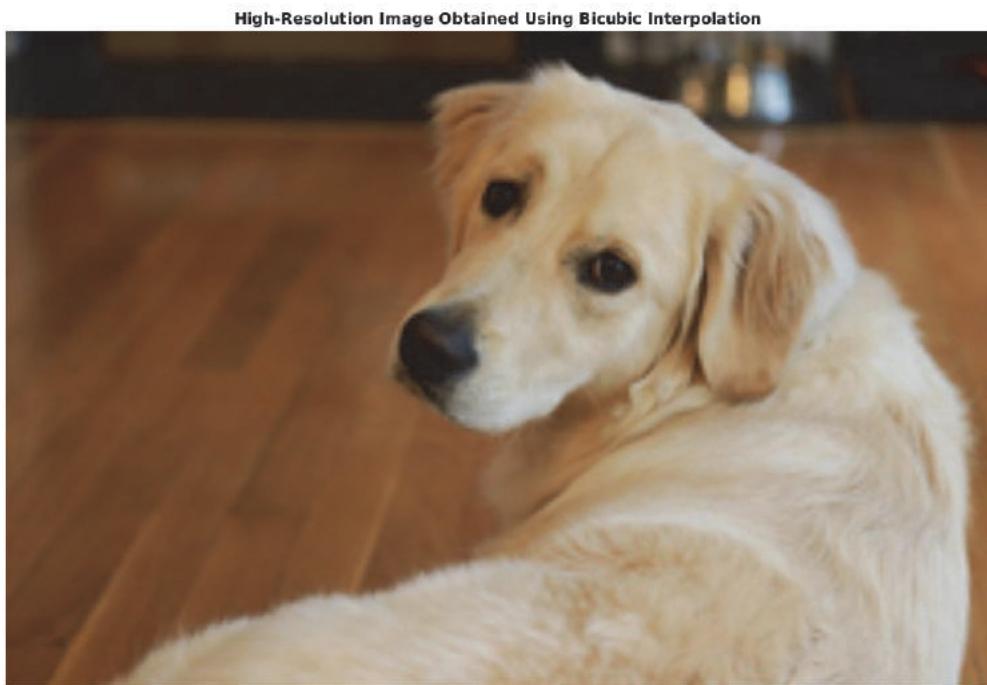


Рис. 1.6. Изображение с увеличенным разрешением, используя метод бикубической интерполяции

1.10 Улучшение разрешения изображения с помощью Pretrained VDSR Network

Напомним, что VDSR обучается с использованием только канала яркости изображения, потому что восприятие человека более чувствительно к изменениям яркости, чем к изменениям цвета.

Преобразуйте изображение с низким разрешением из цветового пространства RGB в каналы яркости (Iy) и цветности (Icb и Icr) с помощью rgb2ycbcr функции.

```
Iycbcr = rgb2ycbcr(Ilowres);  
Iy = Iycbcr(:,:,1);  
Icb = Iycbcr(:,:,2);  
Icr = Iycbcr(:,:,3);
```

Повысить яркость и два канала цветности с использованием бикубической интерполяции. Каналы цветности с улучшенными выборками Icb_bicubic и Icr_bicubic не требуют дальнейшей обработки.

```
Iy_bicubic = imresize(Iy,[nrows ncols],'bicubic');  
Icb_bicubic = imresize(Icb,[nrows ncols],'bicubic');  
Icr_bicubic = imresize(Icr,[nrows ncols],'bicubic');
```

Передайте компонент расширенной яркости Iy_bicubic через обучаемую сеть VDSR. Наблюдайте за активациями с конечного слоя (слоя регрессии). Выходной сигнал сети является желаемым остаточным изображением.

```
Iresidual = activations(net,Iy_bicubic,41);  
Iresidual = double(Iresidual);  
imshow(Iresidual,[])  
title('Residual Image from VDSR')
```



Рис. 1.7. Желаемое остаточное изображение, полученное на выходе обучаемой сети VDSR

Добавьте остаточное изображение в компонент с увеличенной яркостью, чтобы получить компонент яркости VDSR высокого разрешения.

```
Isr = Iy_bicubic + Iresidual;
```

Сконцентрируйте компонент яркости VDSR высокого разрешения с помощью масштабированных цветовых компонентов. Преобразуйте изображение в цветовое пространство RGB с помощью ycbcr2rgb функции. Результатом является окончательное цветное изображение с высоким разрешением с использованием VDSR.

```
Ivdsr = ycbcr2rgb(cat(3,Isr,Icb_bicubic,Icr_bicubic));
imshow(Ivdsr)
title('High-Resolution Image Obtained Using VDSR')
```



Рис. 1.8. Изображение с улучшенным разрешением, полученное с помощью Pretrained VDSR Network

1.11 Визуальное и количественное сравнение

Чтобы получить лучшее визуальное представление о изображениях с высоким разрешением, просмотрите небольшую область внутри каждого изображения. Указать область интереса (ROI) с использованием вектора гоів формате [x у ширина высота]. Элементы определяют координату x и у верхнего левого угла, а также ширину и высоту ROI.

```
roi = [320 30 480 400];
```

Обрезайте изображения с высоким разрешением для этой ROI и покажите результат как монтаж.

```
montage({imcrop(Ibicubic,roi),imcrop(Ivdsr,roi)})  
title('High-Resolution Results Using Bicubic Interpolation (Left) vs.  
VDSR (Right)');
```



Рис. 1.9. Сравнение изображений с улучшенным разрешением

Изображение VDSR имеет более четкие детали и более резкие края.

Используйте показатели качества изображения, чтобы количественно сравнить изображение с высоким разрешением с использованием бикубической интерполяции на изображение VDSR. Образное изображение является исходным изображением высокого разрешения Ireference, перед тем, как подготовить образец изображения с низким разрешением.

Измерьте пиковое отношение сигнал / шум (PSNR) каждого изображения к эталонному изображению. Большие значения PSNR обычно указывают на лучшее качество изображения. См. psnr Дополнительную информацию об этом показателе.

```
bicubicPSNR = psnr(Ibicubic,Ireference)  
bicubicPSNR = 38,4747  
vdsrPSNR = psnr(Ivdsr,Ireference)  
vdsrPSNR = 39,4473
```

Измерьте индекс структурного подобия (SSIM) каждого изображения. ДНЮС оценивает визуальное воздействие трех характеристик изображения: яркости, контраста и структуры, против опорного изображения. Чем ближе значение SSIM к 1, тем лучше тестовое изображение согласуется с эталонным изображением. Дополнительную информацию об этом показателе.

```
bicubicSSIM = ssim(Ibicubic,Ireference)  
bicubicSSIM = 0,9861  
vdsrSSIM = ssim(Ivdsr,Ireference)  
vdsrSSIM = 0,9878
```

Измерьте качество перцепционного изображения с помощью оценщика качества изображения естественности (NIQE). Меньшие оценки NIQE показывают лучшее качество восприятия. Дополнительную информацию об этом показателе.

```
bicubicNIQE = niqe (Ibicubic)  
bicubicNIQE = 5.1719  
vdsrNIQE = niqe (Ivdsr)  
vdsrNIQE = 4.7463
```

Вычислите средний PSNR и SSIM всего набора тестовых изображений для масштабных коэффициентов 2, 3 и 4. Это те же масштабные коэффициенты, которые использовались для определения vdsrImagePatchDatastore. Для простоты вы можете использовать вспомогательную функцию superResolutionMetrics, чтобы вычислить средние показатели.

```
superResolutionMetrics(net,testImages,scaleFactors);
```

Результаты для Масштабного коэффициента 2:

- средний PSNR для Bicubic = 31.809683;
- средний PSNR для VDSR = 32.9158534;
- средний SSIM для Bicubic = 0.938194;
- средний SSIM для VDSR = 0.953473.

Результаты для Масштабного коэффициента 3:

- средний PSNR для Bicubic = 28.170441;
- средний PSNR для VDSR = 28.802722;
- средний SSIM для бикубического = 0.884381;
- средний SSIM для VDSR = 0,898248.

Результаты для шкалы 4:

- средний PSNR для бикубического = 27.010839;
- средний PSNR для VDSR = 28.087250;
- средний SSIM для бикубического = 0.861604;
- средний SSIM для VDSR = 0.882349.

Для каждого масштабного коэффициента, по сравнению с бикубической интерполяцией, VDSR имеет лучшие показатели метрики.

1.12 Резюме

В этом примере показано, как создать и обучить сеть VDSR для нескольких факторов масштабирования, а затем использовать сеть для улучшения разрешения изображения с помощью супер разрешения. Это шаги по обучению сети:

Загрузите данные обучения.

Определите настраиваемый мини-пакетный хранилище данных, называемый a VDSRImagePatchDatastore, для извлечения патчей с изображений с низким разрешением и вычисления остатков цели из соответствующих

патчей в эталонных изображениях с высоким разрешением. Этот хранилище данных используется для передачи данных обучения в сеть.

Определите слои сети VDSR.

Укажите варианты обучения.

Постройте сеть, используя `trainNetwork` функцию.

После обучения сети VDSR или загрузки предварительно определенной сети VDSR, пример выполняет супер разрешение на изображении с низким разрешением. В примере сравнивается результат VDSR с супер разрешением с использованием бикубической интерполяции, которая не использует глубокое обучение. VDSR превосходит бикубическую интерполяцию как по качеству восприятия изображения, так и по количественным измерениям качества.

2. Деблокирование изображения JPEG с использованием Deep Learnin, обучение шумоизолирующей сверхточной сети (DnCNN)

Сжатие изображения используется для уменьшения объема памяти изображения. Один популярный и мощный метод сжатия используется в формате изображения JPEG, который использует коэффициент качества для определения объема сжатия. Уменьшение значения качества приводит к более высокому сжатию и уменьшению объема памяти за счет визуального качества изображения.

JPEG-сжатие является сжатием с «потерями», что означает, что процесс сжатия приводит к потере изображения. Для изображений JPEG эта потеря информации отображается как блокирующая артефакты изображения. Как показано на рисунке, большее сжатие приводит к большему количеству потерь информации и более сильным артефактам. Текстурированные области с высокочастотным контентом, такие как трава и облака, выглядят размытыми. Резкие края, такие как крыша дома и ограждения на маяке, показывают звон.



Рис. 2.1. Изображение с различным коэффициентом сжатия

JPEG-деблокирование – это процесс уменьшения эффектов артефактов сжатия в изображениях JPEG. Существует несколько методов деблокирования JPEG, включая более эффективные методы, которые используют глубокое обучение. В этом примере реализован один такой метод глубокого обучения, который пытается минимизировать влияние артефактов сжатия JPEG.

2.1 Сеть DnCNN

В этом примере используется встроенная глубоководная сверточная нейронная сеть, называемая DnCNN. Сеть была в первую очередь предназначена для удаления шума из изображений. Тем не менее, архитектура DnCNN также может быть обучена удалению артефактов сжатия JPEG или увеличению разрешения изображения.

Сеть DnCNN учится оценивать остаточное изображение. Остаточное изображение – это разница между нетронутым изображением иискаженной копией изображения. Остаточное изображение содержит информацию об искажении изображения. В этом примере искажение отображается как артефакты, блокирующие JPEG.

Сеть DnCNN обучается обнаружению остаточного изображения из яркости цветного изображения. Канал яркости изображения Y представляет яркость каждого пикселя через линейную комбинацию значений красного, зеленого и синего пикселей. В контрастности два канала цветности изображения Cb и Cr являются различными линейными комбинациями значений красного, зеленого и синего пикселей, которые представляют информацию о разности цветов. DnCNN обучается с использованием только канала яркости, потому что восприятие человека более чувствительно к изменениям яркости, чем изменения цвета.

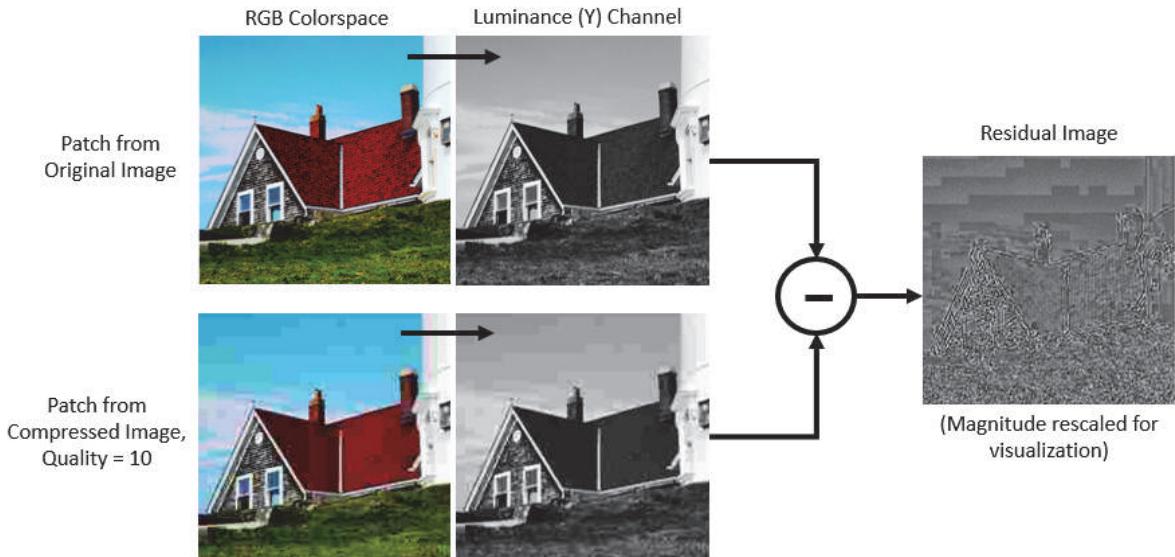


Рис. 2.2. Схема, реализующая остаточное изображение

Если Y_{Original} есть яркость нетронутого изображения и $Y_{\text{Compressed}}$ является яркостью изображения, содержащего артефакты сжатия JPEG, тогда вход в сеть DnCNN $Y_{\text{Compressed}}$ и сеть узнает, чтобы предсказать $Y_{\text{Residual}} = Y_{\text{Compressed}} - Y_{\text{Original}}$ из данных обучения.

Как только сеть DnCNN узнает, как оценивать остаточное изображение, она может восстановить неискаженную версию сжатого JPEG-изображения, добавив остаточное изображение в сжатый канал яркости, а затем преобразуя изображение обратно в цветовое пространство RGB.

2.2 Загрузка данных обучения

Загрузите тестовый тест IAPR TC-12, который состоит из 20 000 неподвижных изображений. Набор данных включает фотографии людей, животных, городов и многое другое. Вы можете использовать вспомогательную функцию, `downloadIAPRTC12Data`, чтобы загрузить данные. Размер файла данных составляет $\sim 1,8$ ГБ.

```
imagesDir = tempdir;
url = "http://www-i6.informatik.rwth-
aachen.de/imageclef/resources/iaprtc12.tgz" ;
```

```
downloadIAPRTC12Data (URL, imagesDir);
```

В этом примере будет проведена тренировка сети с небольшим подмножеством данных Benchmark IAPR TC-12. Загрузите данные обучения imageCLEF. Все изображения - 32-битные цветные изображения JPEG.

```
trainImagesDir = fullfile (imagesDir, 'iaprtc12' , 'images' , '00' );
exts = { '.jpg' , '.bmp' , '.png' };
trainImages = imageDatastore (trainImagesDir, 'FileExtensions' ,
exts);
```

Перечислите количество обучающих изображений.

```
numel (trainImages.Files)
```

```
ans = 251
```

2.3 Подготовка данных обучения

Чтобы создать набор данных обучения, прочитайте в нетронутых изображениях и выпишите изображения в формате JPEG с различными уровнями сжатия.

Создайте структуру папок, чтобы правильно организовать данные обучения.

```
originalFileLocation = fullfile(imagesDir,'iaprtc12','images','00');

% Make a folder structure for the training data
if
~exist(fullfile(imagesDir,'iaprtc12','JPEGDeblockingData','Original',
'dir')

mkdir(fullfile(imagesDir,'iaprtc12','JPEGDeblockingData','Original'));
end
if
~exist(fullfile(imagesDir,'iaprtc12','JPEGDeblockingData','Compressed
'),'dir')

mkdir(fullfile(imagesDir,'iaprtc12','JPEGDeblockingData','Compressed
'));
```

end

```

uncompressedFileLocation = full-
file(imagesDir,'iaprtc12','JPEGDeblockingData','Original');
compressedFileLocation = full-
file(imagesDir,'iaprtc12','JPEGDeblockingData','Compressed');

```

Укажите значения качества изображения JPEG, используемые для рендеринга артефактов сжатия изображения. Значения качества должны находиться в диапазоне [0, 100]. Малые значения качества приводят к большему сжатию и более сильным артефактам сжатия. Используйте более плотную выборку небольших значений качества, поэтому данные обучения имеют широкий диапазон артефактов сжатия.

```
JPEGQuality = [5: 5: 40 50 60 70 80];
```

Создайте оригинальные и сжатые изображения обучения из исходных данных.

```

Files = dir([originalFileLocation filesep '*.jpg']);
imNumber = 1;
for fileIndex = 1:size(files,1)
    fname = [originalFileLocation filesep files(fileIndex).name];
    im = imread(fname);
    if size(im,3) == 3
        im = rgb2gray(im);
    end
    for index = 1:length(JPEGQuality)
        imwrite(im,[uncompressedFileLocation filesep
num2str(imNumber) '.jpg'],'JPEG','Quality',100)
        imwrite(im,[compressedFileLocation filesep num2str(imNumber)
'.jpg'],'JPEG','Quality',JPEGQuality(index))
        imNumber = imNumber + 1;
    end
end

```

2.4. Определение мини-пакетного хранилища данных для обучения

Мини-пакетное хранилище данных используется для передачи данных обучения в сеть. В этом примере определяется пользовательская реализация мини-пакетного хранилища данных, называемая

JPEGImagePatchDatastore как удобный способ создания дополненных патчей изображений для обучения сети деблокирования JPEG.

Исправленные патчи изображения действуют как сетевой вход. Остальные исправления являются желаемым выходом сети. Каждая мини-партия содержит 128 патчей размером 50 на 50 пикселей. Только одна мини-партия будет извлекаться из каждого изображения во время обучения, и все патчи будут извлечены из случайных позиций на изображениях.

```
batchSize = 128;
patchSize = 50;
batchesPerImage = 1;

exts = { '.jpg' };
imdsUncompressed = imageDatastore (uncompressedFileLocation,
'FileExtensions' , exts);
imdsCompressed = imageDatastore (compressFileLocation,
'FileExtensions' , exts);

ds = JPEGImagePatchDatastore (imdsUncompressed, imdsCompressed, ...
    «MiniBatchSize» , batchSize, ...
    «PatchSize» , patchSize, ...
    «BatchesPerImage» , batchesPerImage);
```

Выполните операцию чтения в мини-пакетном хранилище данных для изучения данных.

```
inputBatch = read(ds);
summary(inputBatch)
```

Переменные:

jpegPatches: 128×1 cell

residuals: 128×1 cell

2.5 Настройка слоев DnCNN

Создайте слои встроенной сети DnCNN, используя dnCNNLayers функцию. По умолчанию глубина сети (количество слоев свертки) равно 20.

```
layers = dnCNNLayers ()
```

```
layer =
```

1x59 Layer array со слоями:

1. 'InputLayer' Image Input 50x50x1 images.
2. 'Conv1' Convolution 64 3x3x1 свертки с шагом [1 1] и отступом [1 1 1].
3. 'ReLU1' ReLU ReLU.
4. 'Conv2' свертка 64, 3x3x64 свертки с шагом [1 1] и отступом [1 1 1].
5. 'BNorm2' Batch Normalization. Нормализация партии с 64 каналами.
6. 'ReLU2' ReLU ReLU.
7. 'Conv3' Свертка 64, 3x3x64 свертки с шагом [1 1] и отступы [1 1 1 1].
8. 'BNorm3'. Пакетная нормализация. Пакетная нормализация с 64 каналами.
9. 'ReLU3' ReLU ReLU.
10. 'Conv4' Convolution 64, 3x3x64 свертки с шагом [1 1] и отступом [1 1 1 1].
11. 'BNorm4'. Пакетная нормализация. Пакетная нормализация с 64 каналами.
12. 'ReLU4' ReLU ReLU.
13. 'Conv5' Convolution 64, 3x3x64 свертки с шагом [1 1] и отступом [1 1 1 1].
14. 'BNorm5'. Пакетная нормализация. Пакетная нормализация с 64 каналами.
15. 'ReLU5' ReLU ReLU.
16. 'Conv6' Convolution 64 свертки 3x3x64 с шагом [1 1] и отступом [1 1 1 1].
17. 'BNorm6'. Пакетная нормализация. Пакетная нормализация с 64 каналами.
18. 'ReLU6' ReLU ReLU.
19. 'Conv7' Свертка 64 3x3x64 свертки с шагом [1 1] и отступы [1 1 1 1].
20. 'BNorm7'. Пакетная нормализация. Пакетная нормализация с 64 каналами.
21. 'ReLU7' ReLU ReLU.
22. 'Conv8' Свертка 64 3x3x64 свертки с шагом [1 1] и отступом [1 1 1 1].
23. 'BNorm8'. Пакетная нормализация. Пакетная нормализация с 64 каналами.

24. 'ReLU8' ReLU ReLU.
25. 'Conv9' Convolution 64 3x3x64 свертки с шагом [1 1] и отступом [1 1 1 1].
26. 'BNorm9'. Пакетная нормализация. Пакетная нормализация с 64 каналами.
27. 'ReLU9' ReLU ReLU.
28. 'Conv10' Convolution 64 3x3x64 свертки с шагом [1 1] и отступом [1 1 1 1].
29. 'BNorm10'. Пакетный режим. Нормализация партии с использованием 64 каналов.
30. 'ReLU10' ReLU ReLU.
31. 'Conv11' Convolution 64 свертки 3x3x64 с шагом [1 1] и отступом [1 1 1 1].
32. 'BNorm11'. Пакетная нормализация. Пакетная нормализация с 64 каналами.
33. 'ReLU11' ReLU ReLU.
34. 'Conv12' Свертка 64 3x3x64 свертки с шагом [1 1] и отступы [1 1 1 1].
35. 'BNorm12'. Пакетная нормализация. Пакетная нормализация с 64 каналами.
36. 'ReLU12' ReLU ReLU.
37. 'Conv13' Свертка 64 3x3x64 свертки с шагом [1 1] и отступом [1 1 1 1].
38. 'BNorm13'. Пакетная нормализация. Пакетная нормализация с 64 каналами.
39. 'ReLU13' ReLU ReLU.
40. 'Conv14' Свертка 64 3x3x64 свертки с шагом [1 1] и отступом [1 1 1 1].
41. 'BNorm14'. Пакетная нормализация. Пакетная нормализация с 64 каналами.
42. 'ReLU14' ReLU ReLU.
43. 'Conv15' Convolution 64 3x3x64 свертки с шагом [1 1] и отступом [1 1 1 1].
44. 'BNorm15'. Пакетная нормализация. Пакетная нормализация с 64 каналами.

45. 'ReLU15' ReLU ReLU.
46. 'Conv16' Convolution 64 свертки 3x3x64 с шагом [1 1] и отступом [1 1 1].
47. 'BNorm16'. Пакетная нормализация. Пакетная нормализация с 64 каналами.
48. 'ReLU16' ReLU ReLU.
49. 'Conv17' Свертка 64 3x3x64 свертки с шагом [1 1] и отступы [1 1 1].
50. 'BNorm17'. Пакетная нормализация. Пакетная нормализация с 64 каналами.
51. 'ReLU17' ReLU ReLU.
52. 'Conv18'. Свертка 64 3x3x64 свертки с шагом [1 1] и отступом [1 1 1].
53. 'BNorm18'. Пакетная нормализация. Пакетная нормализация с 64 каналами.
54. 'ReLU18' ReLU ReLU.
55. 'Conv19' Convolution 64 3x3x64 свертки с шагом [1 1] и отступом [1 1 1].
56. 'BNorm19' Batch Normalization. Нормализация партии с 64 каналами.
57. 'ReLU19' ReLU ReLU.
58. 'Conv20' Convolution 1 свертки 3x3x64 с шагом [1 1] и отступом [1 1 1].
59. «Конечная регрессионная линия». Регрессионная выходная ошибка среднего квадрата.

2.6 Варианты обучения

Обучение сети с использованием стохастического градиентного спуска с оптимизацией импульса (SGDM). Задайте параметры гиперпараметра для SDGM с помощью `trainingOptions` функции.

Обучение глубокой сети занимает много времени. Ускорьте обучение, указав высокую скорость обучения. Однако это может привести к тому, что градиенты сети будут взрываться или расти неуправляемо, что препятствует успешному обучению сети. Чтобы сохранить градиенты в значимом диапа-

зоне, включите градиентную обрезку, установив 'GradientThreshold' значение 0,01 и указывая 'GradientThresholdMethod' на использование абсолютного значения градиентов.

```
maxEpochs = 30;
initLearningRate = 0.1;
l2reg = 0.0001;
batchSize = 128;

options = trainingOptions('sgdm',...
    'Momentum',0.9,...
    'InitialLearnRate',initLearningRate,...
    'LearnRateSchedule','piecewise',...
    'GradientThresholdMethod','absolute-value',...
    'GradientThreshold',0.005,...
    'L2Regularization',l2reg,...
    'MiniBatchSize',batchSize,...
    'MaxEpochs',maxEpochs,...
    'Plots','training-progress');
```

2.7 Обучение сети

После настройки параметров обучения и мини-пакетного хранилища данных с помощью этой trainNetwork функции обучите сеть DnCNN . Для обучения сети задайте doTraining параметр в следующем коде true. Для обучения рекомендуется использовать графический процессор NVIDIA™ с поддержкой CUDA с вычислительной способностью 3,0 или выше.

Если вы сохраните doTraining параметр в следующем коде как false, тогда пример возвращает предварительно определенную сеть DnCNN.

```
% Training runs when doTraining is true
doTraining = false;
if doTraining
    [net, info] = trainNetwork(ds,layers,options);
else
    load('pretrainedJPEGDnCNN.mat');
end
```

Теперь вы можете использовать сеть DnCNN для удаления артефактов сжатия JPEG из новых изображений.

2.8 Выполнение деблокирования JPEG с использованием сети DnCNN

Чтобы выполнить деблокирование JPEG с помощью DnCNN, выполните остальные шаги этого примера. В оставшейся части примера показано, как:

- создайте образцы тестовых изображений с артефактами сжатия JPEG на трех разных уровнях качества;
- удалите артефакты сжатия, используя сеть DnCNN;
- визуально сравнить изображения до и после деблокирования;
- оцените качество сжатых и деблокированных изображений путем количественного определения их сходства с неискаженным эталонным изображением.

2.9 Создание образцовых изображений с блокирующими артефактами

Создайте образцы изображений для оценки результата деблокирования изображения JPEG с использованием сети DnCNN. Набор тестовых данных testImages содержит 21 неискаженное изображение, загруженное в Image Processing Toolbox™. Загрузите изображения в imageDatastore.

```
exts = {'.jpg', '.png'};  
fileNames =  
{'sherlock.jpg', 'car2.jpg', 'fabric.png', 'greens.jpg', 'hands1.jpg', 'kobi.png', ...  
  
'lighthouse.png', 'micromarket.jpg', 'office_4.jpg', 'onion.png', 'pears.png', 'yellowlily.jpg', ...  
  
'indiancorn.jpg', 'flamingos.jpg', 'sevilla.jpg', 'llama.jpg', 'parkavenue.jpg', ...  
    'peacock.jpg', 'car1.jpg', 'strawberries.jpg', 'wagon.jpg'};  
filePath = [fullfile(matlabroot, 'toolbox', 'images', 'imdata') filesep];  
filePathNames = strcat(filePath, fileNames);  
testImages = imageDatastore(filePathNames, 'FileExtensions', exts);
```

Отображение тестовых изображений в качестве монтажа.

`montage(testImages)`

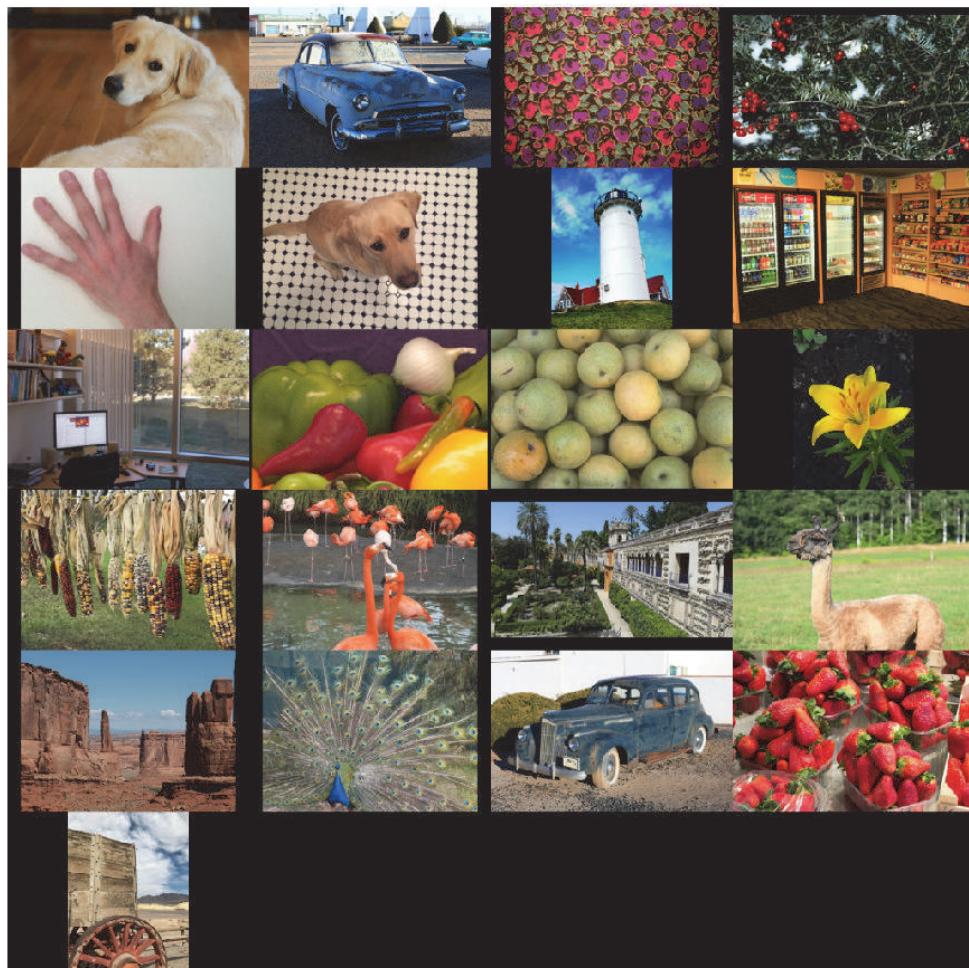


Рис. 2.3. Образцы изображений

Выберите одно из изображений, которое будет использоваться в качестве эталонного изображения для деблокирования JPEG. Вы можете по желанию использовать свое несжатое изображение в качестве эталонного изображения.

```
indx = 7; % Index of image to read from the test image datastore  
Ireference = readimage(testImages,indx);  
imshow(Ireference)  
title('Uncompressed Reference Image')
```

Uncompressed Reference Image



Рис. 2.4. Изображение, выбранное в качестве эталонного

Создайте три сжатых тестовых изображения с Quality значениями JPEG 10, 20 и 50.

```
Im-
write(Ireference,fullfile(tempdir,'testQuality10.jpg'),'Quality',10);
im-
write(Ireference,fullfile(tempdir,'testQuality20.jpg'),'Quality',20);
im-
write(Ireference,fullfile(tempdir,'testQuality50.jpg'),'Quality',50);
```

2.10 Сжатые изображения препроцесса

Прочтите сжатые версии изображения в рабочей области.

```
I10 = imread(fullfile(tempdir,'testQuality10.jpg'));
I20 = imread(fullfile(tempdir,'testQuality20.jpg'));
I50 = imread(fullfile(tempdir,'testQuality50.jpg'));
```

Отображение сжатых изображений в виде монтажа.

```
montage({I50,I20,I10}, 'Size',[1 3])
title('JPEG-Compressed Images with Quality Factor: 50, 20 and 10
(left to right)')
```



Рис. 2.5. Сжатые изображения в виде монтажа

Напомним, что DnCNN обучается с использованием только канала яркости изображения, потому что восприятие человека более чувствительно к изменениям яркости, чем изменения цвета. Преобразуйте JPEG-сжатые изображения из цветового пространства RGB в цветовое пространство YCbCr с помощью этой `rgb2ycbcr` функции.

```
I10ycbcr = rgb2ycbcr(I10);
I20ycbcr = rgb2ycbcr(I20);
I50ycbcr = rgb2ycbcr(I50);
```

2.11 Применение сети DnCNN

Для выполнения переадресации сети используйте эту denoiseImage функцию. Эта функция использует точно такие же процедуры обучения и тестирования для шумоподавления изображения. Вы можете представить артефакты сжатия JPEG как тип шума изображения.

```
I10y_predicted = denoiseImage(I10ycbcr(:,:,1),net);
I20y_predicted = denoiseImage(I20ycbcr(:,:,1),net);
I50y_predicted = denoiseImage(I50ycbcr(:,:,1),net);
```

Каналы цветности не нуждаются в обработке. Соедините выделенный канал яркости с исходными каналами цветности, чтобы получить деблокированное изображение в цветовом пространстве YCbCr.

```
I10ycbcr_predicted = cat(3,I10y_predicted,I10ycbcr(:,:,2:3));
I20ycbcr_predicted = cat(3,I20y_predicted,I20ycbcr(:,:,2:3));
I50ycbcr_predicted = cat(3,I50y_predicted,I50ycbcr(:,:,2:3));
```

Преобразуйте уменьшенное изображение YCbCr в цветовое пространство RGB с помощью ycbcr2rgb функции.

```
I10_predicted = ycbcr2rgb(I10ycbcr_predicted);
I20_predicted = ycbcr2rgb(I20ycbcr_predicted);
I50_predicted = ycbcr2rgb(I50ycbcr_predicted);
```

Отображение деблокированных изображений в качестве монтажа.

```
montage({I50_predicted,I20_predicted,I10_predicted}, 'Size',[1 3])
title('Deblocked Images with Quality Factor: 50, 20 and 10 (left to right)')
```



Рис. 2.6. Деблокированные изображения в качестве монтажа

Чтобы получить лучшее визуальное представление об улучшениях, изучите меньшую область внутри каждого изображения. Указать область интереса (ROI) с использованием вектора `roi` в формате [x у ширина высота]. Элементы определяют координату x и у верхнего левого угла, а также ширину и высоту ROI.

```
roi = [30 440 100 80];
```

Обрезайте сжатые изображения в этот ROI и покажите результат как монтаж.

```
i10 = imcrop(I10,roi);
i20 = imcrop(I20,roi);
i50 = imcrop(I50,roi);
montage({i50 i20 i10}, 'Size',[1 3])
title('Patches from JPEG-Compressed Images with Quality Factor: 50,
20 and 10 (left to right)')
```



Рис. 2.7. Обрезанные сжатые изображения в качестве монтажа

Обрежите деблокированные изображения с этим ROI и покажите результат как монтаж.

```
i10predicted = imcrop(I10_predicted,roi);
i20predicted = imcrop(I20_predicted,roi);
i50predicted = imcrop(I50_predicted,roi);
montage({i50predicted,i20predicted,i10predicted}, 'Size',[1 3])
title('Patches from Deblocked Images with Quality Factor: 50, 20 and 10 (left to right)')
```



Рис. 2.8. Обрезанные деблокированные изображения в качестве монтажа

2.12 Количествоное сравнение

Определите качество деблокированных изображений с помощью четырех показателей. Вы можете использовать displayJPEGResults вспомогательную функцию для вычисления этих показателей для сжатых и деблокированных изображений с коэффициентами качества 10, 20 и 50:

- индекс структурного сходства (SSIM). ДНЮС оценивает визуальное воздействие трех характеристик изображения: яркости, контраста и структуры, против опорного изображения. Чем ближе значение SSIM к 1, тем лучше тестовое изображение согласуется с эталонным изображением. Здесь эталонное изображение является неискаженным исходным изображением, Ireference перед сжатием JPEG. Дополнительную информацию об этом показателе;

- пиковое отношение сигнал / шум (PSNR). Чем больше значение PNSR, тем сильнее сигнал по сравнению с искажением. Дополнительную информацию об этом показателе;

- оценщик качества изображения естественности (NIQE). NIQE измеряет качество восприятия изображения с использованием модели, подготовленной из естественных сцен. Меньшие оценки NIQE показывают лучшее качество восприятия;

- оценщик пространственного качества слепых / нерегулярных изображений (BRISQUE). BRISQUE измеряет качество восприятия изображения с использованием модели, подготовленной из естественных сцен с искажением изображения. Меньшие оценки BRISQUE показывают лучшее качество восприятия.

```
displayJPEGResults (IReference, I10, I20, I50, I10_predicted,  
I20_predicted, I50_predicted)
```

Сравнение SSIM:

- I10: 0.90624 I10_predicted: 0.91286;
- I20: 0.94904 I20_predicted: 0.95444;
- I50: 0.97238 I50_predicted: 0.97482;

Сравнение ПСНР:

- I10: 26.6046 I10_predicted: 27.0793;
- I20: 28.8015 I20_predicted: 29.3378;

- I50: 31.4512 I50_predicted: 31.8584.

Сравнение NIQE:

- I10: 7,0989 I10_predicted: 3,9334;
- I20: 4,5065 I20_predicted: 3,0699;
- I50: 2,8866 I50_predicted: 2,4109.

ПРИМЕЧАНИЕ. Меньшая оценка NIQE означает лучшее качество восприятия.

Сравнение BRISQUE:

- I10: 52.2731 I10_predicted: 38.9688;
- I20: 45.5237 I20_predicted: 30.9583;
- I50: 27.7386 I50_predicted: 24.3889.

ПРИМЕЧАНИЕ. Меньший показатель BRISQUE означает лучшее качество восприятия

2.13 Резюме

В этом примере показано, как создать и обучить сеть DnCNN, а затем использовать сеть для уменьшения артефактов сжатия JPEG в изображениях. Это были шаги по обучению сети:

- загрузите данные обучения;
- создавайте учебные изображения, записывая оригинальные изображения в формате JPEG с различными уровнями сжатия;
- определите пользовательский мини-пакетный хранилище данных, называемый a JPEGImagePatchDatastore, для извлечения патчей из входного сжатого изображения и вычисления целевых остатков из соответствующих патчей в нетронутых изображениях. Это хранилище данных использовалось для передачи данных обучения в сеть;
- создайте слои сети DnCNN, используя dnCNNLayers функцию;
- укажите варианты обучения;

- постройте сеть, используя trainNetwork функцию.

После обучения сети DnCNN или загрузки предварительно назначенной сети DnCNN пример сжимает тестовое изображение с тремя значениями качества, а затем использует сеть для удаления артефактов сжатия.

3. Сегментация изображения

3.1. Обнаружение ячейки с помощью сегментации изображений

В этом примере показано, как обнаружить ячейку с помощью обнаружения края и базовой морфологии. Объект может быть легко обнаружен в изображении, если он достаточно контрастен по сравнению с фоном. В этом примере, как ячейки, используются раковые клетки простаты.

Шаг 1: чтение изображения.

Считываем файл `cell.tif`, который является изображением клетки рака простаты.

```
I = imread('cell.tif');
figure, imshow(I), title('original image');
text(size(I,2),size(I,1)+15, ...
    'Image courtesy of Alan Partin', ...
    'FontSize',7,'HorizontalAlignment','right');
text(size(I,2),size(I,1)+25, ...
    'Johns Hopkins University', ...
    'FontSize',7,'HorizontalAlignment','right');
```



Рис. 3.1. Изначальное изображение

Шаг 2: определение всей ячейки.

В этом изображении присутствуют две клетки, но в полном объеме можно увидеть только одну. Обнаружим её. По-другому обнаружение объектов называется сегментацией. Сегментированный объект значительно отличается от фонового изображения. Изменения в контрасте могут быть об-

наружены операторами, которые вычисляют градиент изображения. Градиент изображения и пороговое значение могут быть рассчитаны и применены для создания двоичной маски, содержащей сегментированную ячейку. Сначала мы используем **edge** оператор Sobel для вычисления порогового значения. Затем мы настраиваем пороговое значение **edge** и снова используем для получения бинарной маски, содержащей сегментированную ячейку.

```
[~, threshold] = edge(I, 'sobel');
fudgeFactor = .5;
BWs = edge(I, 'sobel', threshold * fudgeFactor);
figure, imshow(BWs), title('binary gradient mask');
```

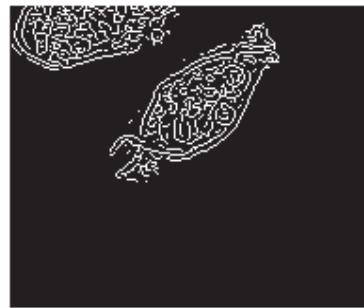


Рис. 3.2. Бинарная маска, содержащая сегментированную ячейку

Шаг 3: расширение изображения.

Маска двоичного градиента показывает линии с высокой контрастностью изображения. Эти линии не совсем определяют контур интересующего объекта. По сравнению с исходным изображением вы можете видеть разрывы в линиях, окружающих объект в маске градиента. Эти разрывы исчезнут, если изображение Собела будет расширено с помощью линейных структурных элементов, которые создаются с помощью функции **strel**.

```
se90 = strel ('line', 3, 90);
se0 = strel ('line', 3, 0);
```

Бинарная градиентная маска расширяется с использованием вертикального элемента структурирования, за которым следует горизонтальный структурирующий элемент. Функция `imdilate` расширяет изображение.

```
BWsdl = imdilate(BWs, [se90 se0]);  
figure, imshow(BWsdl), title('dilated gradient mask');
```



Ruc. 3.3. Название рисунка

Шаг 4: заполнение внутренних промежутков.

Маска расширенного градиента показывает контур клетки довольно хорошо, но внутри клетки все еще есть есть дыры. Чтобы заполнить эти отверстия, используем функцию `imfill`.

```
BWdfill = imfill(BWsdl, 'holes');  
figure, imshow(BWdfill);  
title('binary image with filled holes');
```



Ruc. 3.4. Использование функции imfill

Шаг 5: удаление соединенных областей на границе.

Ячейка, представляющая интерес, успешно сегментирована, но это не единственный объект, который был найден. Любые объекты, которые связаны с границей изображения, могут быть удалены с помощью функции `imclearborder`. Связность в функции `imclearborder` устанавливается равной 4, для удаления диагональных соединений.

```
BWnobord = imclearborder(BWdfill, 4);
figure, imshow(BWnobord), title('cleared border image');
```



Ruc. 3.5. Название рисунка

Шаг 6: сглаживание объекта.

Наконец, чтобы сделать сегментированный объект естественным, мы сглаживаем объект, дважды разрушая изображение с помощью элемента структурирования алмаза. Этот элемент создается с помощью функции `strel`.

```
seD = strel('diamond',1);
BWfinal = imerode(BWnobord,seD);
BWfinal = imerode(BWfinal,seD);
figure, imshow(BWfinal), title('segmented image');
```



Рис. 3.6. Сглаживаемый объект с помощью функции `strel`

Альтернативный метод для отображения сегментированного объекта - размещение контура вокруг сегментированной ячейки. Контур создается функцией `bwperim`.

```
BWoutline = bwperim(BWfinal);
Segout = I;
Segout(BWoutline) = 255;
figure, imshow(Segout), title('outlined original image');
```

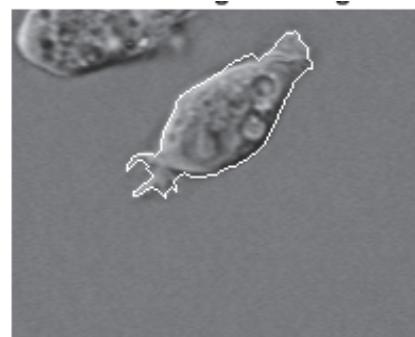


Рис. 3.7. Сегментированный объект, созданный функцией `bwperim`

3.2. Сегментация с контролируемым маркером

Сегментация с использованием преобразования водораздела работает лучше, если вы можете идентифицировать или отмечать объекты переднего и заднего плана. Сегментация водоразделов, контролируемых маркерами, выполняется по следующим основным пунктам:

1. Вычислить функцию сегментации. Это изображение, темные области которого являются объектами, которые вы пытаетесь сегментировать.
2. Вычислить маркеры переднего плана. Они связаны с блоками пикселей внутри каждого из объектов.
3. Вычислите фоновые метки. Это пиксели, которые не являются частью какого-либо объекта.
4. Измените функцию сегментации так, чтобы она имела минимальные значения на передних и задних маркерах.
5. Вычислить преобразование водораздела модифицированной функции сегментации.

В этом примере показаны различные функции Image Processing Toolbox™, включая `imgradient`, `watershed`, `label2rgb`, `labeloverlay`, `imopen`, `imclose`, `imreconstruct`, `imcomplement`, `imregionalmax`, `bwareaopen`, `graythresh`, и `imimposemin`.

Шаг 1: считайте цветное изображение и преобразуйте его в оттенки серого.

```
rgb = imread('pears.png');
I = rgb2gray(rgb);
imshow(I)

text(732,501,'Image courtesy of Corel(R)',...
    'FontSize',7,'HorizontalAlignment','right')
```



Image courtesy of Corel(R)

Рис. 3.8. Изображение в оттенках серого

Шаг 2: используйте масштаб градиента в качестве функции сегментации.

Вычислите величину градиента. Градиент высок на границах объектов и низкий (в основном) внутри объектов.

```
gmag = imggradient (I);
imshow (gmag, [])
title ('Gradient Magnitude')
```

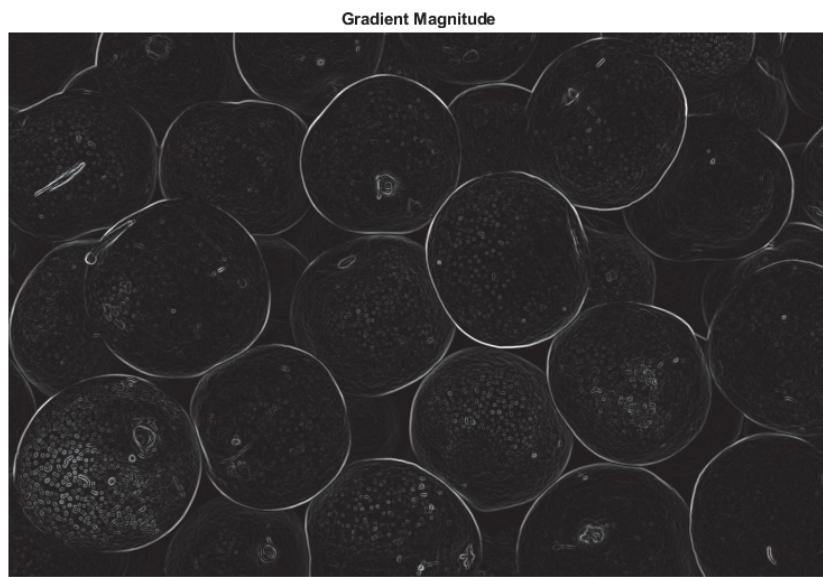


Рис. 3.9. Градиента в качестве функции сегментации

Можете ли вы сегментировать изображение, используя преобразование водораздела непосредственно на величину градиента?

```
L = водораздел (gmag);
Lrgb = label2rgb (L);
imshow (Lrgb)
title ('Watershed Transform of Gradient Magnitude')
```

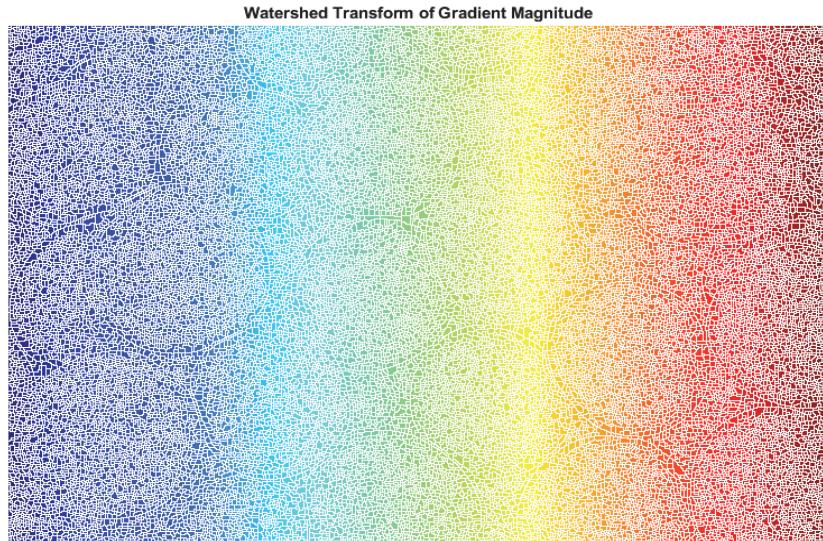


Рис. 3.10. Сегментирование изображения водораздельным преобразованием градиентной величины

Нет. Без дополнительной предварительной обработки, такой как вычисление маркеров, использование преобразования водораздела часто приводит к «перерегулированию».

Шаг 3: отметьте объекты переднего плана.

Здесь можно использовать множество процедур, для нахождения маркеров переднего плана, которые должны быть связаны с каплями пикселей внутри каждого из объектов. В этом примере используются морфологические методы, называемые «открытие по реконструкции» и «закрытие по реконструкции», чтобы «очистить» изображение. Эти операции будут создавать плоские максимумы внутри каждого объекта, их расположение определяется функцией `imregionalmax`.

Открытие - это эрозия с последующей дилатацией, а открытие по реконструкции - эрозия с последующей морфологической реконструкцией. Давайте сравним их. Сначала вычислим открытие, используя `imopen`.

```
se = strel('disk',20);
Io = imopen (I, se);
imshow (Io)
title('Opening')
```

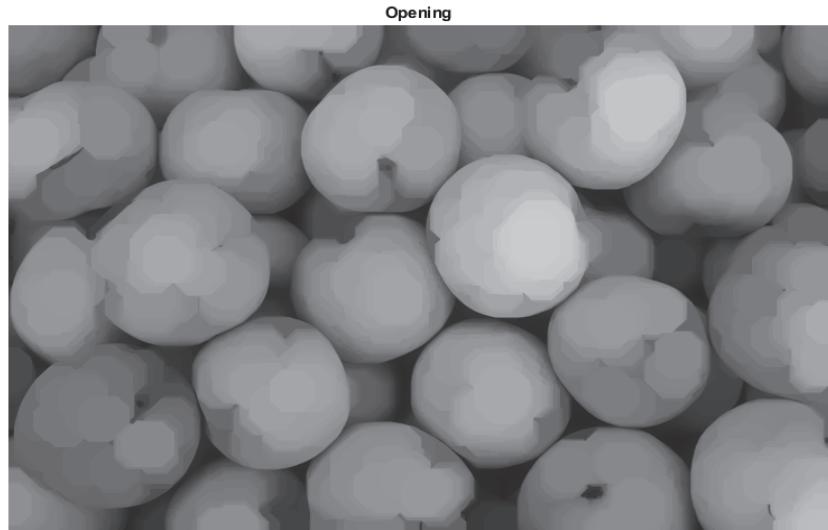


Рис. 3.11. Открытие рисунка с помощью `imopen`

Затем вычислим открытие по реконструкции с помощью `imerode` и `imreconstruct`.

```
Ie = imerode (I, se);
Iobr = imreconstruct (Ie, I);
imshow (Iobr)
title('Opening-by-Reconstruction')
```



Рис. 3.12. Открытие рисунка с помощью `imerode` и `imreconstruct`

После открытия с закрытием можно удалить темные пятна и следы стебля. Сравните регулярное морфологическое закрытие с закрытием по реконструкции. Сначала попробуйте `imclose`:

```
Ioc = imclose (Io, se);  
imshow (Ioc)  
title('Opening-Closing')
```



Рис. 3.13. Морфологическое закрытие

Теперь используем `imdilate`, за которым следует `imreconstruct`.

Обратите внимание, что вы должны дополнять ввод и вывод изображений `imreconstruct`.

```
Iobrd = imdilate(Iobr,se);
Iobrcbr = imreconstruct(imcomplement(Iobrd),imcomplement(Iobr));
Iobrcbr = imcomplement(Iobrcbr);
imshow(Iobrcbr)
title('Opening-Closing by Reconstruction')
```

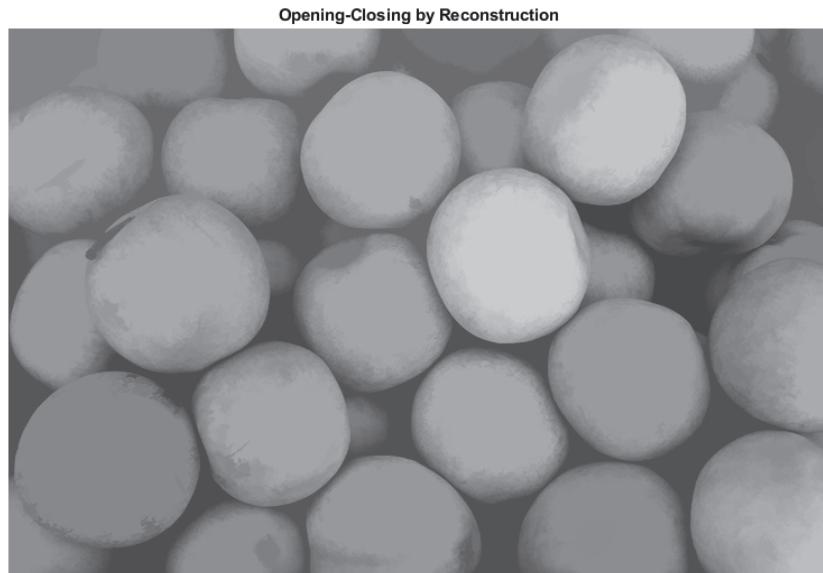


Рис. 3.14. Закрытием по реконструкции

Если сравнить `Iobrcbr` и `Ioc`, становится очевидно, что реконструкции на основе открытия и закрытия являются более эффективными, чем стандартные открытия и закрытия при удалении небольших пятен, не затрагиваемых общими формами объектов. Вычислим локальные максимумы `Iobrcbr` для получения хороших маркеров переднего плана.

```
fgm = imregionalmax(Iobrcbr);
imshow(fgm)
title('Regional Maxima of Opening-Closing by Reconstruction')
```

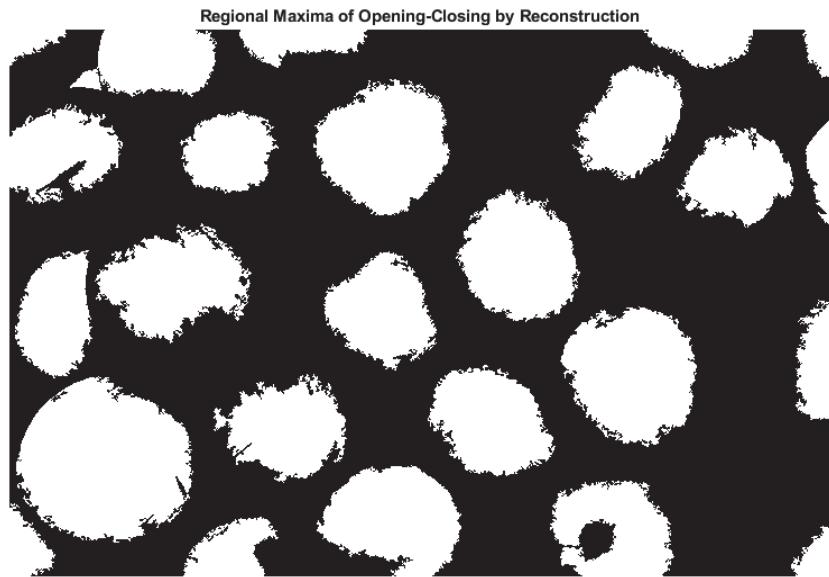


Рис. 3.15. Локальные максимумы изображения

Для простоты интерпритации результата, нарисуйте на исходном изображении изображение маркеры переднего плана.

```
I2 = labeloverlay(I,fgm);  
imshow(I2)  
title('Regional Maxima Superimposed on Original Image')
```

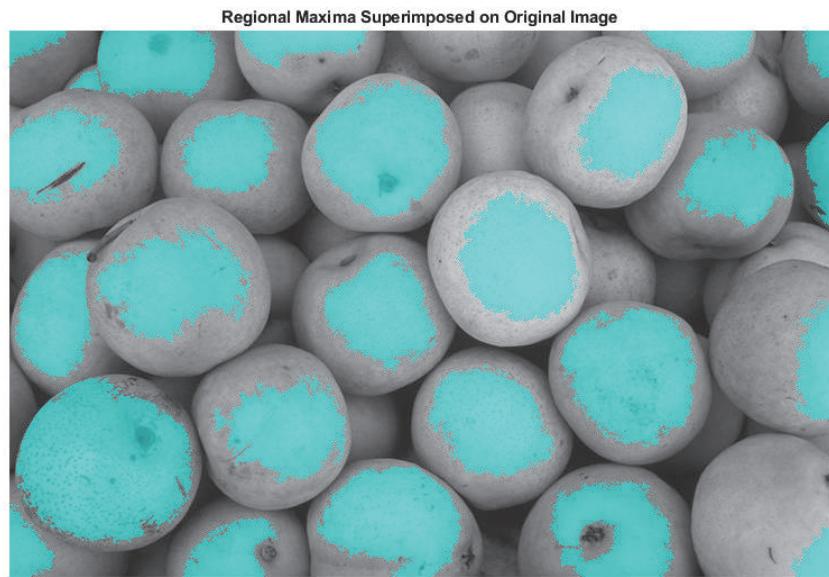


Рис. 3.16. Выделенные локальные максимумы на изображении

Следует обратить внимание на то, что некоторые из объектов с преимущественно закрытыми и затененными областями не отмечены, а это значит, что эти объекты не будут правильно сегментированы в конечном результате. Кроме того, маркеры переднего плана в некоторых объектах идут прямо к краю объектов. Это означает, что вы должны очистить края маркеров, а затем немного уменьшить их. Вы можете сделать это путем закрытия, за которым следует эрозия.

```
se2 = strel(ones(5,5));
fgm2 = imclose(fgm,se2);
fgm3 = imerode(fgm2,se2);
```

Эта процедура имеет тенденцию оставлять некоторые отклоненные изолированные пиксели, которые необходимо удалить. Вы можете сделать это, используя `bwareaopen`, который удаляет все капли, имеющие меньше определенного количества пикселей.

```
fgm4 = bwareaopen(fgm3,20);
I3 = labeloverlay(I,fgm4);
imshow(I3)
title('Modified Regional Maxima Superimposed on Original Image')
```

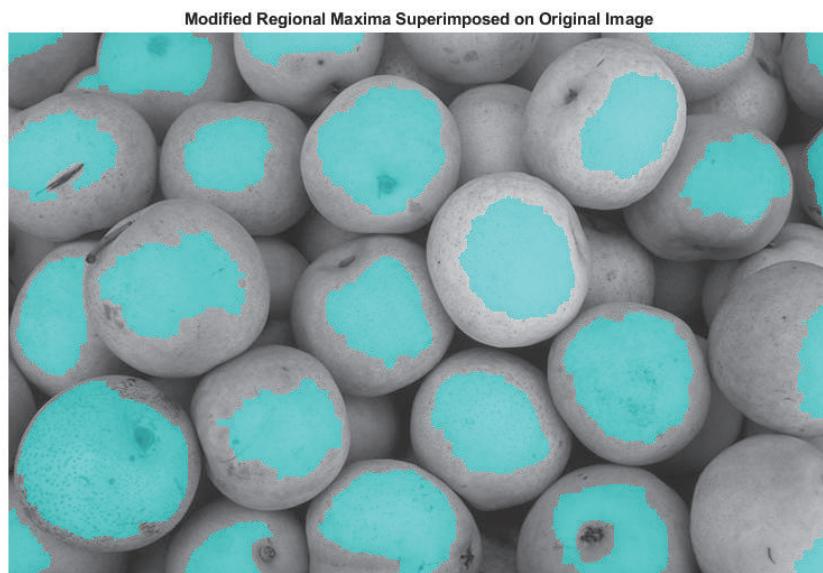


Рис. 3.17. Изображение с использованием `bwareaopen`

Шаг 4: вычисление фоновых маркеров.

Теперь вам нужно отметить фон. На очищенном изображении `Iobrcbr` темные пиксели относятся к фону, поэтому вы можете начать с операции определения порогового значения.

```
bw = imbinarize(Iobrcbr);
imshow(bw)
title('Thresholded Opening-Closing by Reconstruction')
```

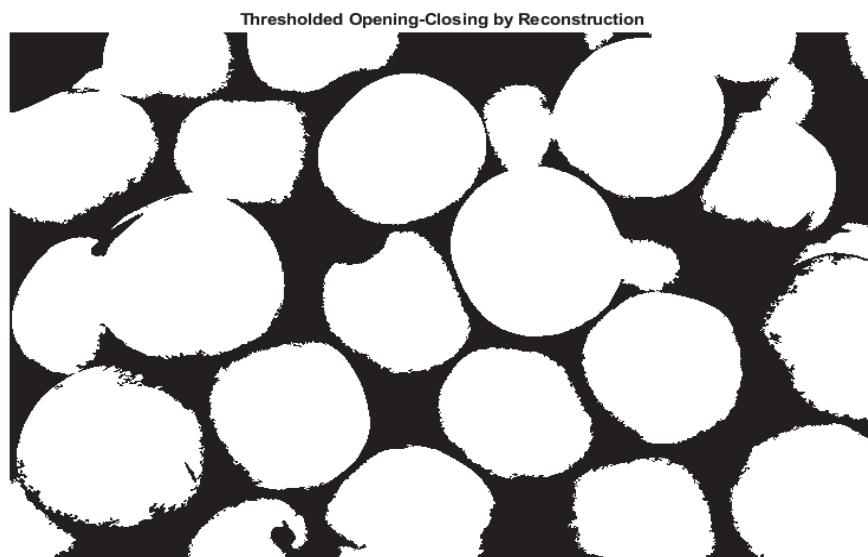


Рис. 3.18. Выставление порогового значения для выравнивания фона изображения

Фоновые пиксели черного цвета, но в идеале мы не хотим, чтобы фоновые метки были слишком близки к краям объектов, которые мы пытаемся сегментировать. «Истощаем» фон, вычисляя «скелет по зонам влияния» или `SKIZ`, на переднем плане `bw`. Это можно сделать, вычислив преобразование водораздела для преобразования расстояния `bw`, а затем найти линии хребтов водораздела (`DL == 0`).

```
D = bwdist(bw);
DL = watershed(D);
bgm = DL == 0;
imshow(bgm)
title('Watershed Ridge Lines')'
```

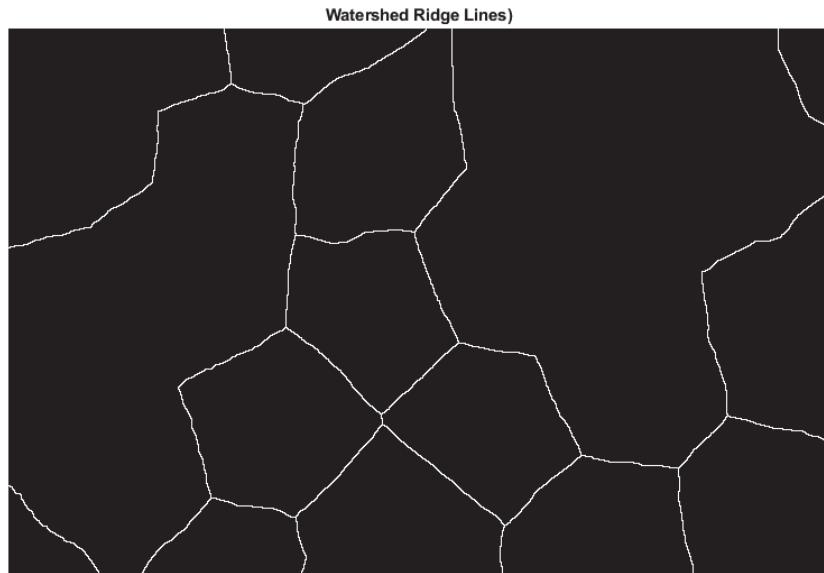


Рис. 3.19. Линии водораздела

Шаг 5: вычислите преобразование водораздела функции сегментации.

Функция `imimposemin` может использоваться для модификации изображения, т.к. она имеет локальные минимумы только в определенных местах. Здесь можно использовать `imimposemin` для изменения изображения градиентной величины, чтобы его единственные локальные минимумы возникали на пикселях переднего плана и фона.

```
gmag2 = imimposemin(gmag, bgm | fgm4);
```

Наконец, мы готовы вычислить сегментирование на основе водоразделов.

```
L = watershed(gmag2);
```

Шаг 6: визуализируйте результат.

Один из методов визуализации - наложение маркеров переднего плана, фоновых маркеров и сегментированных границ объекта на исходное изображение. По мере необходимости можно использовать дилатацию, чтобы сделать некоторые аспекты, такие как границы объектов, более заметными. Границы объектов расположены в области `L == 0`. Бинарные марке-

ры переднего и заднего фона масштабируются до разных значений целых чисел, так что им назначаются разные метки.

```
labels = imdilate(L==0,ones(3,3)) + 2*bgm + 3*fgm4;  
I4 = labeloverlay(I,labels);  
imshow(I4)  
title('Markers and Object Boundaries Superimposed on Original Image')
```

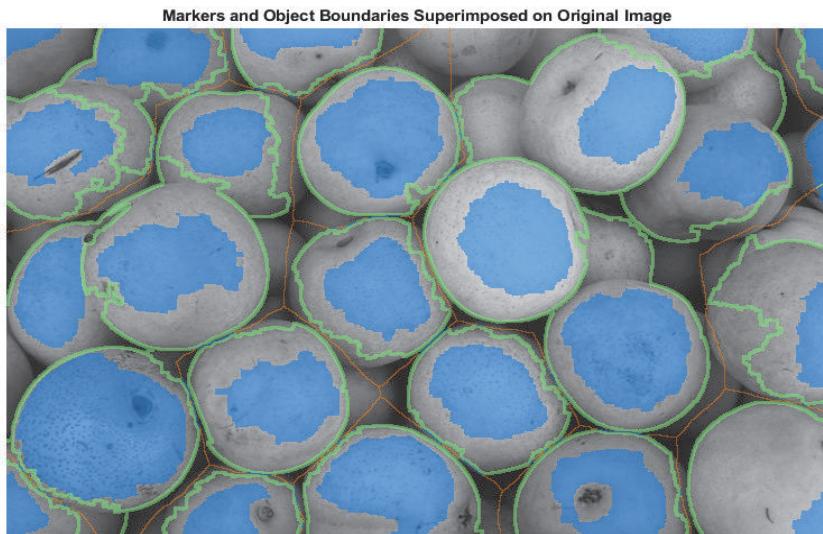


Рис. 3.20. Маркеры и границы объектов, наложенные на изображение

Эта визуализация иллюстрирует, как расположение маркеров переднего плана и фона влияет на результат. В нескольких местах частично закрытые темные объекты были объединены с их более яркими соседними объектами, потому что у перекрытых объектов не было маркеров переднего плана.

Другим полезным методом визуализации является отображение матрицы меток в виде цветного изображения. Матрицы меток, например, созданные `watershed` и `bwlabel`, могут быть преобразованы в исходные изображения для визуализации с использованием `label2rgb`.

```
Lrgb = label2rgb(L,'jet','w','shuffle');  
imshow(Lrgb)  
title('Colored Watershed Label Matrix')
```

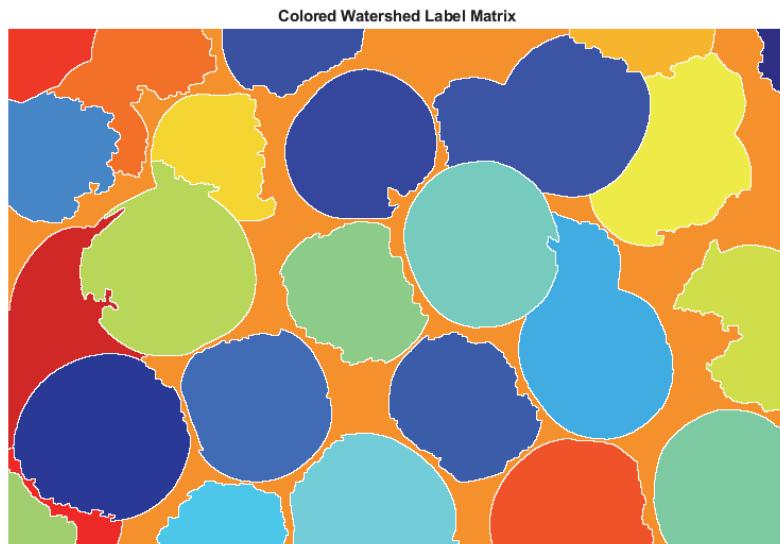


Рис. 3.21. Матрицы меток в виде цветного изображения

Можно использовать прозрачность, чтобы наложить матрицу псевдоцветных меток поверх исходного изображения интенсивности.

```
figure  
imshow(I)  
hold on  
himage = imshow(Lrgb);  
himage.AlphaData = 0.3;  
title('Colored Labels Superimposed Transparently on Original Image')
```

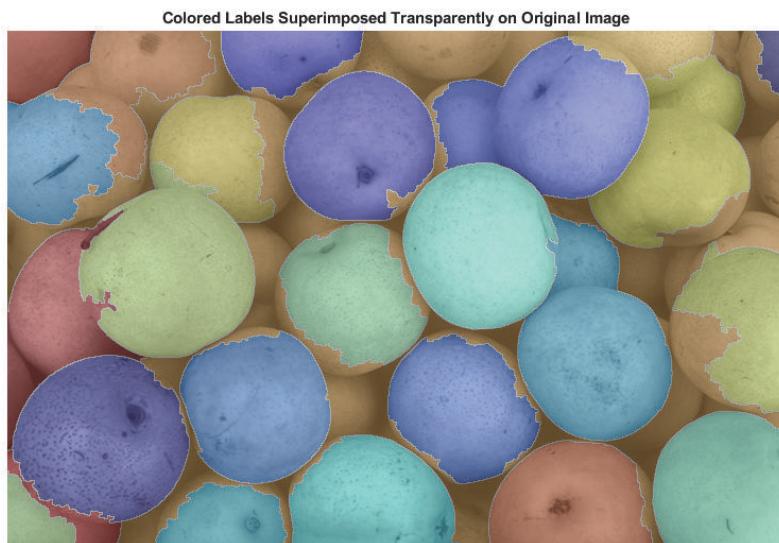


Рис. 3.22. Цветные метки прозрачно накладываются на исходное изображение

3.3. Семантическая сегментация мультиспектральных изображений с использованием глубокого обучения

Семантическая сегментация включает в себя маркировку каждого пикселя в изображении классом. Одним из применений семантической сегментации является отслеживание обезлесения, что связано с изменением лесного покрова с течением времени. Экологические агентства отслеживают обезлесение для оценки и количественной оценки экологического и экологического здоровья региона.

Семантическая сегментация на основе глубокого обучения может дать точное измерение растительного покрова с аэрофотоснимков высокого разрешения. Одной из задач является дифференциация классов со схожими визуальными характеристиками, например, попытка классифицировать зеленый пиксель как траву, кустарник или дерево. Чтобы повысить точность классификации, некоторые наборы данных содержат многоспектральные изображения, которые предоставляют дополнительную информацию о каждом пикселе. Например, набор данных Государственного парка Hamlin Beach дополняет цветные изображения с помощью инфракрасных каналов, которые обеспечивают более четкое разделение классов.

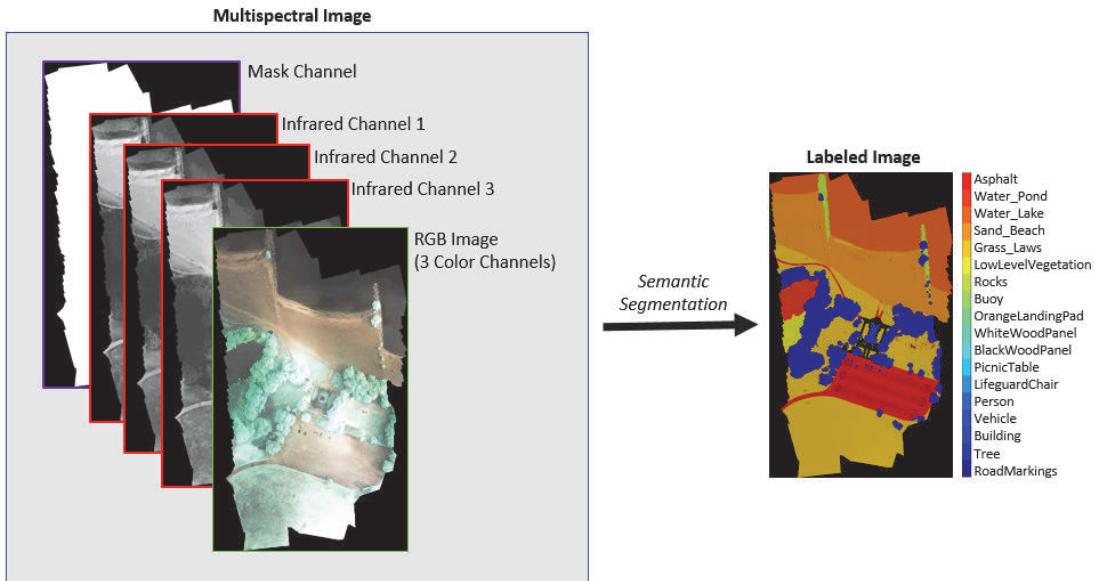


Рис. 3.23. Мультиспектральное изображение

В этом примере показано, как использовать методы семантической сегментации на основе глубокого обучения для расчета процентного покрытия растительности в регионе из набора мультиспектральных изображений.

3.3.1 Загрузка данных

В этом примере используется набор мультиспектральных данных высокого разрешения для обучения сети. Набор изображений был захвачен с помощью беспилотного летательного аппарата над государственным парком Хамлин-Бич, штат Нью-Йорк. Данные содержат маркированные классы обучения, проверки и тестирования с 18 ярлыками класса объектов.

Загрузите версию набора данных МАТ-файла. Вы можете использовать вспомогательную функцию `downloadHamlinBeachMSIData` чтобы загрузить данные. Размер файла данных составляет ~ 3,0 ГБ.

```
imageDir = tempdir;
url = 'http://www.cis.rit.edu/~rmk6217/rit18_data.mat';
```

```
downloadHamlinBeachMSIData(url,imageDir);
```

Кроме того, загрузите предварительно настроенную версию U-Net для этого набора данных. Предварительно подготовленная модель позволяет вам запускать весь пример, не дожидаясь завершения обучения.

```
trainedUnet_url =  
'https://www.mathworks.com/supportfiles/vision/data/multispectralUnet.mat';  
downloadTrainedUnet(trainedUnet_url,imageDir);
```

3.3.2 Проверка данных обучения

Загрузите набор данных в рабочее пространство MATLAB®.

```
load(fullfile(imageDir,'rit18_data','rit18_data.mat'));
```

Изучите структуру данных.

```
whos train_data val_data test_data
```

Name	Size	Bytes	Class
Attributes			
test_data	7x12446x7654	1333663576	uint16
train_data	7x9393x5642	741934284	uint16
val_data	7x8833x6918	855493716	uint16

Данные мультиспектрального изображения расположены в виде массивов numChannels - by -w idth -by- height . Тем не менее, в MATLAB многоканальные изображения располагаются как массивы массивов w -th- by- height- by- numChannels . Чтобы изменить форму данных, чтобы каналы находились в третьем измерении, используйте вспомогательную функцию switchChannelsToThirdPlane.

```
train_data = switchChannelsToThirdPlane(train_data);  
val_data = switchChannelsToThirdPlane(val_data);  
test_data = switchChannelsToThirdPlane(test_data);
```

Убедитесь, что данные имеют правильную структуру.

```
whos train_data val_data test_data
```

Name	Size	Bytes	Class
Attributes			
test_data	12446x7654x7	1333663576	uint16
train_data	9393x5642x7	741934284	uint16
val_data	8833x6918x7	855493716	uint16

Цветные каналы RGB представляют собой 4-й, 5-й и 6-й каналы изображения. Отобразите цветовой компонент обучения, проверки и тестовых изображений в качестве монтажа. Чтобы изображения выглядели ярче на экране, сравните их гистограммы с помощью `histeq` функции.

```
figure
montage(...  
    {histeq(train_data(:,:,4:6)), ...  
     histeq(val_data(:,:,4:6)), ...  
     histeq(test_data(:,:,4:6))}, ...  
    'BorderSize',10,'BackgroundColor','white')  
title('RGB Component of Training Image (Left), Validation Image (Center), and Test Image (Right)')
```

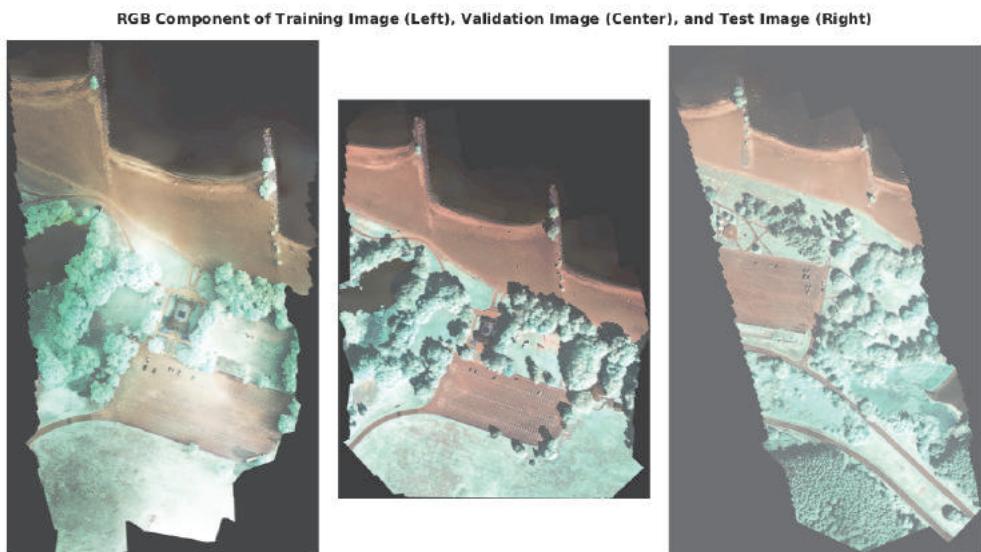


Рис. 3.24. RGB компоненты обучения изображения(слева), проверочное изображение(по центру) и тестовое изображение(справа)

Отображение первых трех сравниваемых гистограмм каналов данных обучения в качестве монтажа. Эти каналы соответствуют диапазонам ближнего инфракрасного диапазона и выделяют различные компоненты изображения на основе их сигнатур тепла. Например, деревья вблизи центра изо-

брожения 2-го канала показывают больше деталей, чем деревья в двух других каналах.

```
figure
montage(...
    {histeq(train_data(:,:,:1)), ...
    histeq(train_data(:,:,:2)), ...
    histeq(train_data(:,:,:3))}, ...
    'BorderSize',10,'BackgroundColor','white')
title('IR Channels 1 (Left), 2, (Center), and 3 (Right) of Training
Image')
```

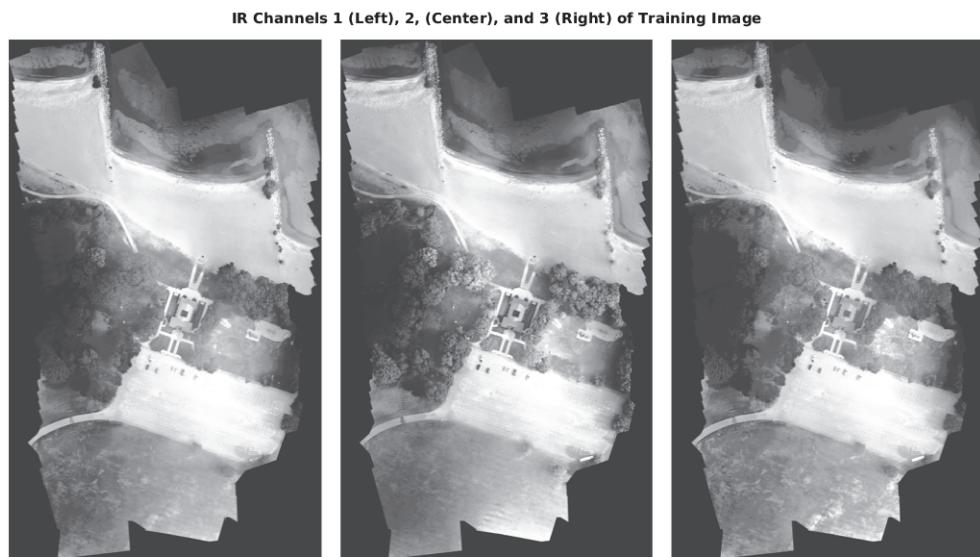


Рис. 3.25. Канал 1(слева), канал 2(по центру) и канал 3(слева) изображения

Канал 7 представляет собой маску, которая указывает действительную область сегментации. Отобразите маску для обучения, проверки и тестовых изображений.

```
figure
montage(...
    {train_data(:,:,:7), ...
    val_data(:,:,:7), ...
    test_data(:,:,:7)}, ...
    'BorderSize',10,'BackgroundColor','white')
```

```
title('Mask of Training Image (Left), Validation Image (Center), and  
Test Image (Right)')
```

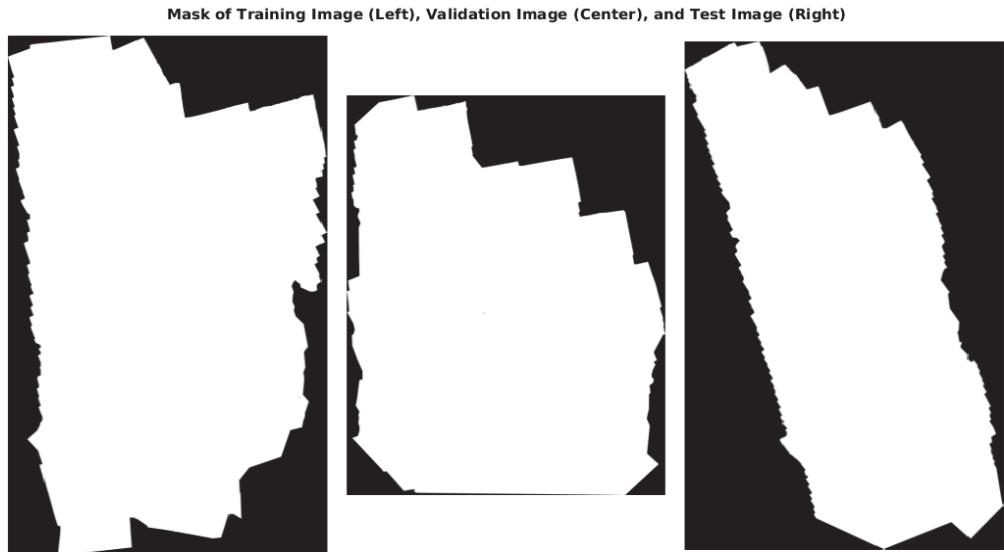


Рис. 3.26. Мaska обучения изображения(слева), проверочное изображение (по центру) и тестовое изображение (справа)

Маркированные изображения содержат данные наземной истины для сегментации, причем каждый пиксель присваивается одному из 18 классов. Получите список классов с соответствующими идентификаторами.

DISP (классы):

0. Другой класс / граница изображения.
1. Дорожная разметка.
2. Дерево.
3. Строительство.
4. Автомобиль (автомобиль, грузовик или автобус).
5. Лицо.
6. Председатель спасателя.
7. Стол для пикника.
8. Черная деревянная панель.

9. Белая деревянная панель.
10. Оранжевая посадочная площадка.
11. Водный буй.
12. Скалы.
13. Другая растительность.
14. Трава.
15. Песок.
16. Вода (озеро).
17. Вода (пруд).
18. Асфальт (автостоянка / дорожка).

Создайте вектор имен классов:

```
classNames = [ "RoadMarkings", "Tree", "Building", "Vehicle", "Person",
...
    "LifeguardChair", "PicnicTable", "BlackWoodPanel", ...
    "WhiteWoodPanel", "OrangeLandingPad", "Buoy", "Rocks", ...
    "LowLevelVegetation", "Grass_Lawn", "Sand_Beach", ...
    "Water_Lake", "Water_Pond", "Asphalt"];
```

Наложите надписи на изображение с изображениями RGB с выравниванием по гистограмме. Добавьте к изображению цветную панель.

```
cmap = jet(numel(classNames));
B = labelover-
lay(histeq(train_data(:,:,:,4:6)),train_labels,'Transparency',0.8,'Colo
rmap',cmap);

figure
title('Training Labels')
imshow(B)
```

Предупреждение: изображение слишком велико для установки на экран;
отображение на 8%

```
N = numel(classNames);
ticks = 1/(N*2):1/N:1;
color-
bar('TickLabels',cellstr(classNames),'Ticks',ticks,'TickLength',0,'Ti
ckLabelInterpreter','none');
colormap(cmap)
```

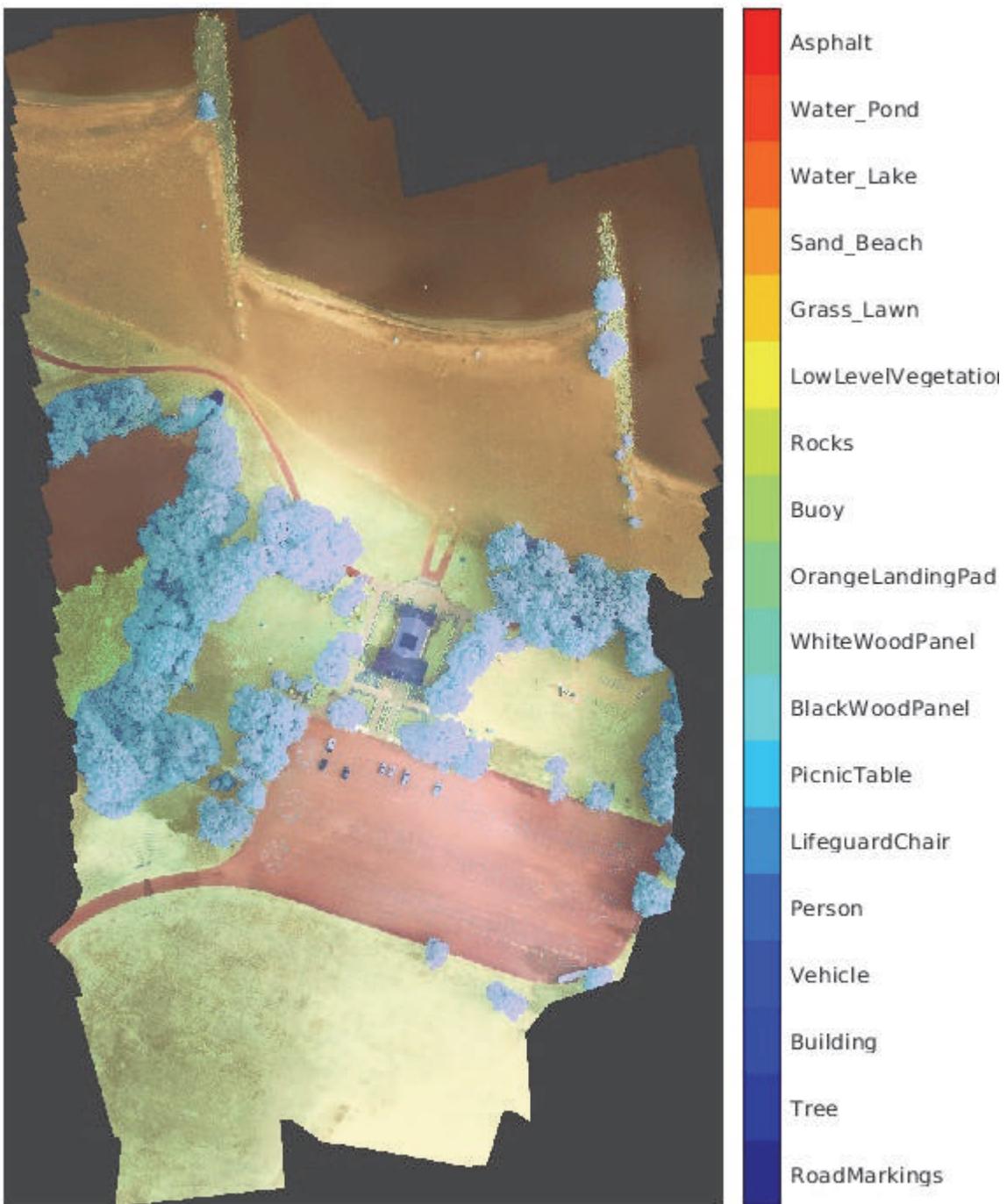


Рис. 3.27. Мультиспектральное изображение

Сохраните данные обучения как .МАТ-файл и учебные ярлыки в виде файла .PNG.

```
save('train_data.mat','train_data');  
imwrite(train_labels,'train_labels.png');
```

3.3.3 Определение мини-пакетного хранилища данных для обучения нейронной сети U-Net

Мини-пакетное хранилище данных используется для передачи данных обучения в сеть. В этом примере определяется пользовательская реализация хранилища данных mini-bach, называемая PixelLabelImagePatchDatastore как удобный способ создания дополненных изображений и патчей для обучения семантической сети сегментации.

Начните с хранения учебных изображений с 'train_data.mat' в imageDatastore. Поскольку формат файла МАТ является нестандартным форматом изображения, вы должны использовать считыватель файлов МАТ для чтения данных изображения. Вы можете использовать вспомогательный считыватель файлов МАТ, matReader который извлекает первые шесть каналов из учебных данных и пропускает последний канал, содержащий маску.

```
imds =  
imageDatastore('train_data.mat','FileExtensions','.mat','ReadFcn',@matReader);
```

Создайте a, pixelLabelDatastore чтобы сохранить метки, содержащие 18 помеченных областей.

```
pixelLabelIds = 1:18;  
pxds = pixelLabelDatastore('train_labels.png',classNames,pixelLabelIds);
```

Создайте PixelLabelImagePatchDatastore из хранилища данных изображений и хранилища данных ярлыков пикселей. Чтобы имитировать данные ландшафта, собранные беспилотным летательным аппаратом,

`PixelLabelImagePatchDatastore` увеличиваются пары патчей путем добавления случайного вращения, отражения и масштабирования.

Каждая мини-партия содержит 16 патчей размером от 256 до 256 пикселей. На каждой итерации эпохи добывается тысяча мини-партий. Все патчи извлекаются из случайных позиций на изображении.

```
ds = PixelLabelImagePatchDatastore(imds, pxd, 'PatchSize', 256, ...
                                    'MiniBatchSize', 16, ...
                                    'BatchesPerImage', 1000)
```

`ds` =
`PixelLabelImagePatchDatastore` со свойствами:

```
    MiniBatchSize: 16
    NumObservations: 16000
```

Чтобы понять, что проходит в сети во время обучения, прочитайте первую мини-партию из `PixelLabelImagePatchDatastore`.

```
sampleBatch = read(ds)
sampleBatch = таблица 16 × 2
    imPatches labelPatches
[256 × 256 × 6 uint16] [256 × 256 категориальный]
```

...

```
[256 × 256 × 6 uint16] [256 × 256 категориальный]
```

Сбросьте хранилище данных до его исходного состояния после прочтения партии образцов.

```
reset(ds)
```

3.3.4 Создание сетевых сетей U-Net

В этом примере используется сеть U-Net. В U-Net начальная серия сверточных слоев чередуется с максимальными уровнями объединения, последовательно уменьшая разрешение входного изображения. За этими слоями следует ряд сверточных слоев, чередующихся с операторами `upsampling`, последовательно увеличивающих разрешение входного изо-

бражения. Название U-Net исходит из того факта, что сеть может быть нарисована с симметричной формой, такой как буква U.

Этот пример изменяет U-Net на использование нулевого заполнения в свертках, так что вход и вывод в свертки имеют одинаковый размер. Используйте вспомогательную функцию,, `createUnet` чтобы создать U-Net с несколькими предварительно выбранными гиперпараметрами.

```
inputTileSize = [256,256,6];
lgraph = createUnet(inputTileSize);
disp(lgraph.Layers)
```

58x1 Layer array со слоями:

1. 'ImageInputLayer' Image Input 256x256x6 изображений с нормализацией нумерации.
2. 'Encoder-Section-1-Conv-1' Свертка 64 3x3x6 свертки с шагом [1 1], коэффициент расширения [1 1] и отступы [1 1 1 1].
3. 'Encoder-Section-1-ReLU-1' ReLU ReLU.
4. 'Encoder-Section-1-Conv-2' Свертка 64 3x3x64 свертки с шагом [1 1], коэффициент расширения [1 1] и отступы [1 1 1 1].
5. 'Код-Раздел-1-ReLU-2' ReLU ReLU.
6. 'Encoder-Section-1-MaxPool'. Максимальное объединение 2x2 максимального объединения с шагом [2 2] и отступом [0 0 0 0].
7. 'Encoder-Section-2-Conv-1' Свертка 128 3x3x64 свертки с шагом [1 1], коэффициент расширения [1 1] и отступы [1 1 1 1].
8. 'Код-Раздел-2-ReLU-1' ReLU ReLU.
9. 'Encoder-Section-2-Conv-2' Свертка 128 3x3x128 свертки с шагом [1 1], коэффициент расширения [1 1] и отступы [1 1 1 1].
10. 'Код-Раздел-2-ReLU-2' ReLU ReLU.
11. 'Encoder-Section-2-MaxPool' Максимальное объединение 2x2 максимального объединения с шагом [2 2] и отступом [0 0 0 0].

12. 'Encoder-Section-3-Conv-1' Свертка 256 3x3x128 свертки с шагом [1 1], коэффициент расширения [1 1] и отступы [1 1 1 1].
13. 'Encoder-Section-3-ReLU-1' ReLU ReLU.
14. 'Encoder-Section-3-Conv-2' Свертка 256 3x3x256 свертки с шагом [1 1], коэффициент расширения [1 1] и отступы [1 1 1 1].
15. 'Encoder-Section-3-ReLU-2' ReLU ReLU.
16. 'Encoder-Section-3-MaxPool' Максимальное объединение 2x2 максимальное объединение с шагом [2 2] и отступ [0 0 0 0].
17. 'Encoder-Section-4-Conv-1' Свертка 512 3x3x256 свертки с шагом [1 1], коэффициент расширения [1 1] и отступы [1 1 1 1].
18. 'Encoder-Section-4-ReLU-1' ReLU ReLU.
19. 'Encoder-Section-4-Conv-2' Convolution 512 3x3x512 свертки с шагом [1 1], коэффициент расширения [1 1] и отступы [1 1 1 1].
20. 'Encoder-Section-4-ReLU-2' ReLU ReLU.
21. «Отсекатель энкодера-раздела-4-DropOut» Отключение 50%.
22. 'Encoder-Section-4-MaxPool' Максимальное объединение 2x2 максимальное объединение с шагом [2 2] и отступ [0 0 0 0].
23. 'Конверсия среднего конв-1' 1024 3x3x512 свертки с шагом [1 1], коэффициент расширения [1 1] и отступы [1 1 1 1].
24. 'ReLU ReLU для средних релей-1'.
25. 'Революция Конд-2' 1024 3x3x1024 свертки с шагом [1 1], коэффициент расширения [1 1] и отступы [1 1 1 1].
26. 'ReLU ReLU для среднего уровня ReLU-2'.
27. «Выпадение середины-DropOut» Отключение 50%.
28. 'Конвертор-Секция-1-UpConv' Транспонированная свертка 512 2x2x1024 транспонированные свертки с шагом [2 2] и выходное обрезка [0 0].
29. 'Реле-Раздел-1-UpReLU' ReLU ReLU.

30. 'Decoder-Section-1-DepthConcatenation' Конкатенация глубины Конкатенация глубины из 2 входов.
31. 'Конвертор-Секция-1-Конв-1' Свертка 512 3x3x1024 свертки с шагом [1 1], коэффициент расширения [1 1] и отступы [1 1 1].
32. 'Decoder-Section-1-ReLU-1' ReLU ReLU.
33. 'Конвертор-Секция-1-Конв-2' Свертка 512 3x3x512 свертки с шагом [1 1], коэффициент расширения [1 1] и отступы [1 1 1].
34. 'Декодер-Раздел-1-ReLU-2' ReLU ReLU.
35. 'Конвертор-Раздел-2-UpConv' Транспонированная свертка 256 2x2x512 транспонированные свертки с шагом [2 2] и выходное обрезка [0 0].
36. 'Реле-Раздел-2-UpReLU' ReLU ReLU.
37. 'Декодер-Раздел-2-Глубина. Конкатенация'. Конкатенация глубины. Конкатенация глубины 2 входов.
38. 'Декодер-Раздел-2-Конв-1' Свертка 256 3x3x512 свертки с шагом [1 1], коэффициент расширения [1 1] и отступы [1 1 1].
39. 'Декодер-Раздел-2-ReLU-1' ReLU ReLU.
40. 'Декодер-Раздел-2-Конв-2' Свертка 256 3x3x256 свертки с шагом [1 1], коэффициент расширения [1 1] и отступы [1 1 1].
41. 'Декодер-Раздел-2-ReLU-2' ReLU ReLU.
42. 'Конвертор-Секция-3-UpConv' Транспонированная свертка 128 2x2x256 транспонированные свертки с шагом [2 2] и выходное обрезка [0 0].
43. 'Реле-Раздел-3-UpReLU' ReLU ReLU.
44. 'Декодер-Раздел-3-DepthConcatenation'. Конкатенация глубины. Конкатенация глубины 2 входов.
45. 'Декодер-Секция-3-Конв-1' Свертка 128 3x3x256 свертки с шагом [1 1], коэффициент расширения [1 1] и отступы [1 1 1].
46. 'Декодер-Раздел-3-ReLU-1' ReLU ReLU.

47. 'Декодер-Секция-3-Конв-2' Свертка 128 3x3x128 свертки с шагом [1 1], коэффициент расширения [1 1] и отступы [1 1 1].
48. 'Декодер-Раздел-3-ReLU-2' ReLU ReLU.
49. 'Конвертор-Секция-4-UpConv' Транспонированная свертка 64 2x2x128 транспонированные свертки с шагом [2 2] и выходное обрезка [0 0].
50. 'Decoder-Section-4-UpReLU' ReLU ReLU.
51. 'Декодер-Раздел-4-DepthConcatenation'. Конкатенация глубины. Конкатенация глубины 2 входов.
52. 'Decoder-Section-4-Conv-1' Свертка 64 3x3x128 свертки с шагом [1 1], коэффициент расширения [1 1] и отступы [1 1 1].
53. 'Декодер-Раздел-4-ReLU-1' ReLU ReLU.
54. 'Декодер-Секция-4-Конв-2' Свертка 64 3x3x64 свертки с шагом [1 1], коэффициент расширения [1 1] и отступы [1 1 1].
55. 'Декодер-Раздел-4-ReLU-2' ReLU ReLU.
56. 'Свертка Final-ConvolutionLayer' 18 свертки 1x1x64 с шагом [1 1], коэффициент расширения [1 1] и отступы [0 0 0 0].
57. «Softmax-Layer» Softmax softmax.
58. «Сегментационный слой». Класс классификации пикселов. Поглощение кросс-энтропии.

3.3.5 Варианты обучения

Обучение сети с использованием стохастического градиентного спуска с оптимизацией импульса (SGDM). Задайте параметры гиперпараметра для SDGM с помощью trainingOptions функции.

Обучение глубокой сети занимает много времени. Ускорьте обучение, указав высокую скорость обучения. Однако это может привести к тому, что градиенты сети будут взрываться или расти неуправляемо, что препятствует успешному обучению сети. Чтобы сохранить градиенты в значимом диапа-

зоне, включите градиентную обрезку, установив 'GradientThreshold' значение 0,05 и укажите, 'GradientThresholdMethod' чтобы использовать L2-норму градиентов.

```
initialLearningRate = 0.05;
maxEpochs = 150;
minibatchSize = 16;
l2reg = 0.0001;

options = trainingOptions('sgdm',...
    'InitialLearnRate', initialLearningRate, ...
    'Momentum', 0.9, ...
    'L2Regularization', l2reg, ...
    'MaxEpochs', maxEpochs, ...
    'MiniBatchSize', minibatchSize, ...
    'VerboseFrequency', 20, ...
    'LearnRateSchedule', 'piecewise', ...
    'Shuffle', 'every-epoch', ...
    'Plots', 'training-progress', ...
    'GradientThresholdMethod', 'l2norm', ...
    'GradientThreshold', 0.05);
```

3.3.6 Обучение сети

После настройки параметров обучения и мини-пакетного хранилища данных, обучите сеть U-Net с помощью этой trainNetwork функции. Для обучения сети задайте doTraining параметр в следующем коде true. Для обучения рекомендуется использовать графический процессор NVIDIA™ с поддержкой CUDA с вычислительной способностью 3,0 или выше.

Если вы сохраните doTraining параметр в следующем коде как false, то пример возвращает предварительно подготовленную сеть U-Net.

```
doTraining = false;
if doTraining
    [net,info] = trainNetwork(ds,lgraph,options);
else
    load(fullfile(imageDir,'trainedUnet','multispectralUnet.mat'));
end
```

Теперь вы можете использовать U-Net для семантического сегмента мультиспектрального изображения.

3.3.7 Предсказание результатов тестирования данных

Для выполнения переадресации в обученной сети используйте вспомогательную функцию `segmentImage`, с набором данных проверки. `segmentImage` выполняет сегментацию на патчах изображения с помощью `semanticseg` функции.

```
predictPatchSize = [1024 1024];
segmentedImage = segmentImage(val_data,net,predictPatchSize);
```

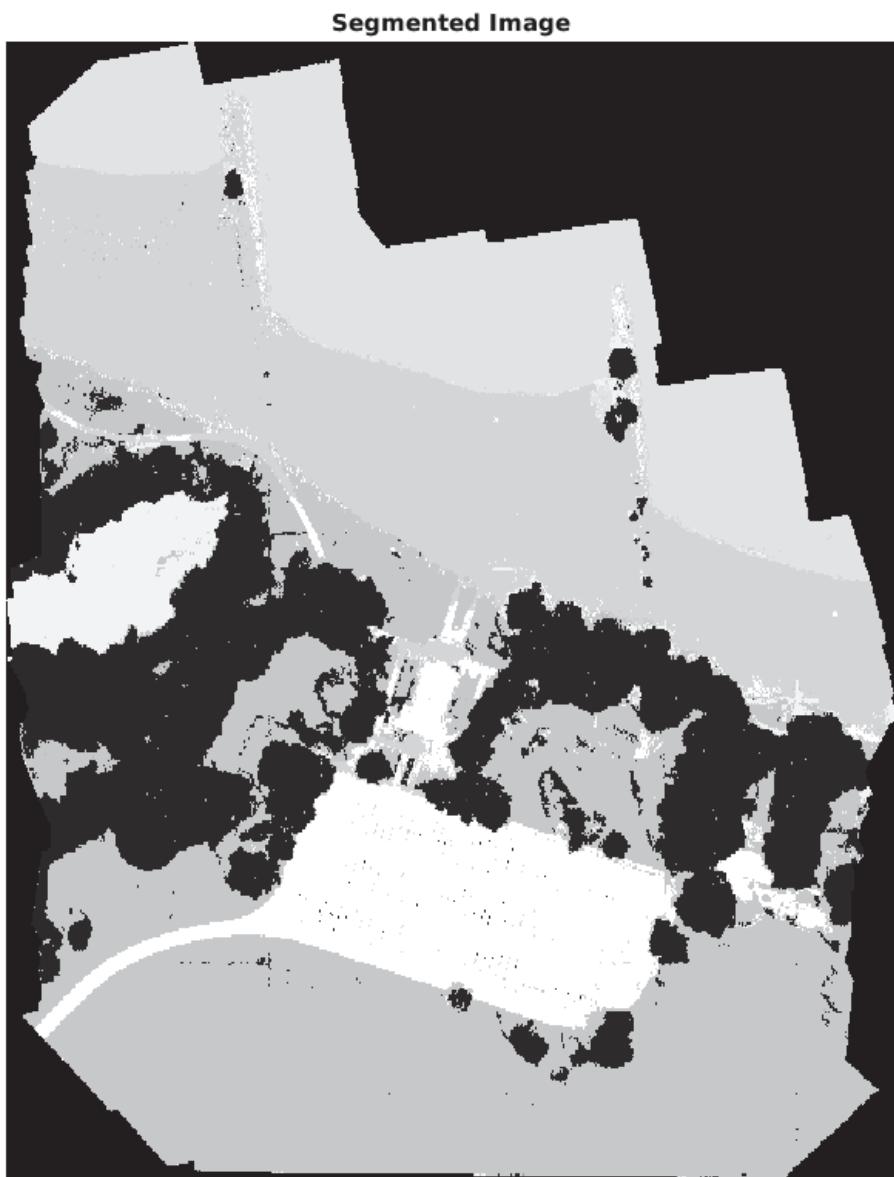
Чтобы извлечь только действительную часть сегментации, умножьте сегментированное изображение на канал маски данных валидации.

```
segmentedImage = uint8(val_data(:,:,7)~=0) .* segmentedImage;

figure
imshow(segmentedImage,[])
```

Предупреждение: изображение слишком велико для установки на экран; отображение на 8%

```
title('Segmented Image')
```



*Рис. 3.28. Сегментация на патчах изображения с помощью
segmentImage функции*

Вывод семантической сегментации является шумным. Выполните обработку изображений после удаления изображений и удаленных пикселей. Визуализируйте сегментированное изображение с удаленным шумом.

```
segmentedImage = medfilt2(segmentedImage,[7,7]);  
imshow(segmentedImage,[]);
```

Предупреждение: изображение слишком велико для установки на экран; отображение на 8%

```
title('Segmented Image with Noise Removed')
```

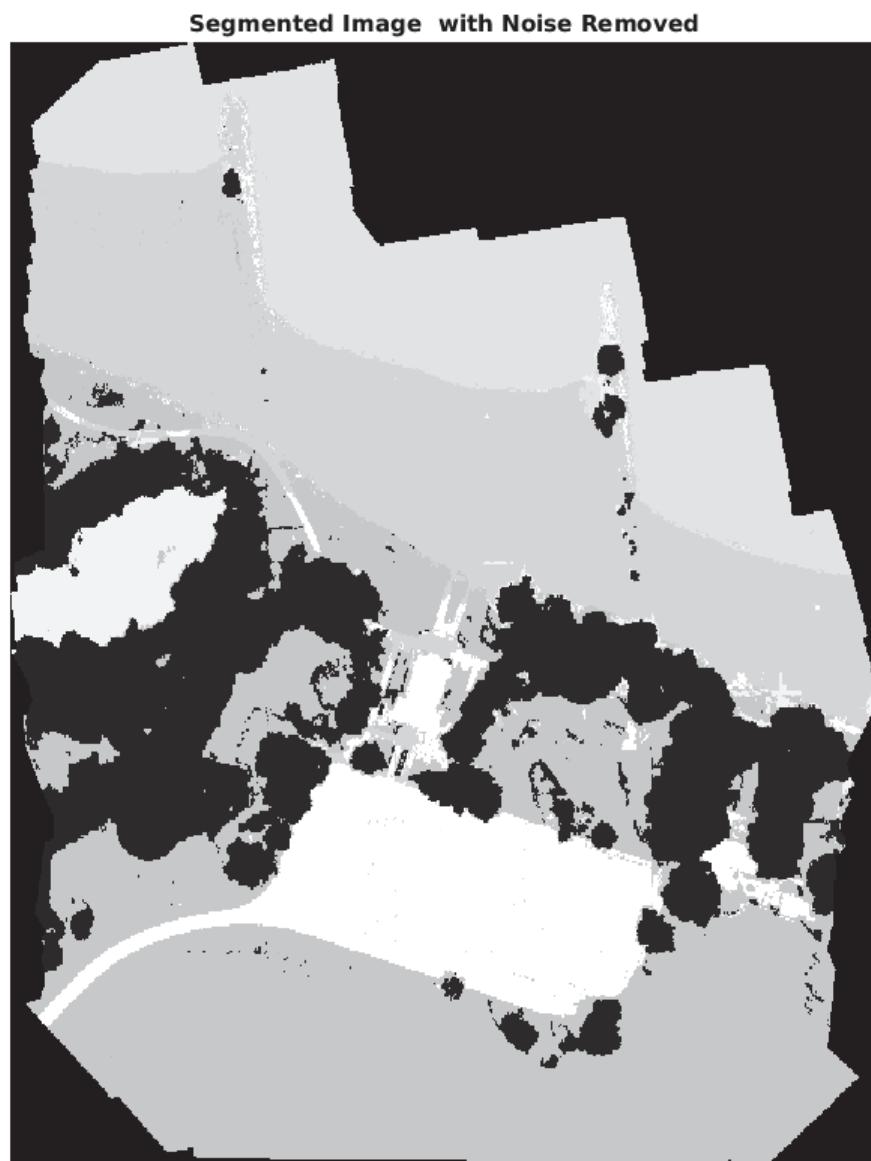


Рис. 3.29. Сегментированное изображение с удаленным шумом

Накладывайте сегментированное изображение на изображение с подтверждённой гистограммой RGB.

```
B = labelover-
lay(histeq(val_data(:,:,4:6)),segmentedImage,'Transparency',0.8,'Colo
rmap',cmap);

figure
imshow(B)
```

Предупреждение: изображение слишком велико для установки на экран; отображение на 8%

```
title('Labeled Validation Image')
color-
bar('TickLabels',cellstr(classNames),'Ticks',ticks,'TickLength',0,'Ti
ckLabelInterpreter','none');
colormap(cmap)
```

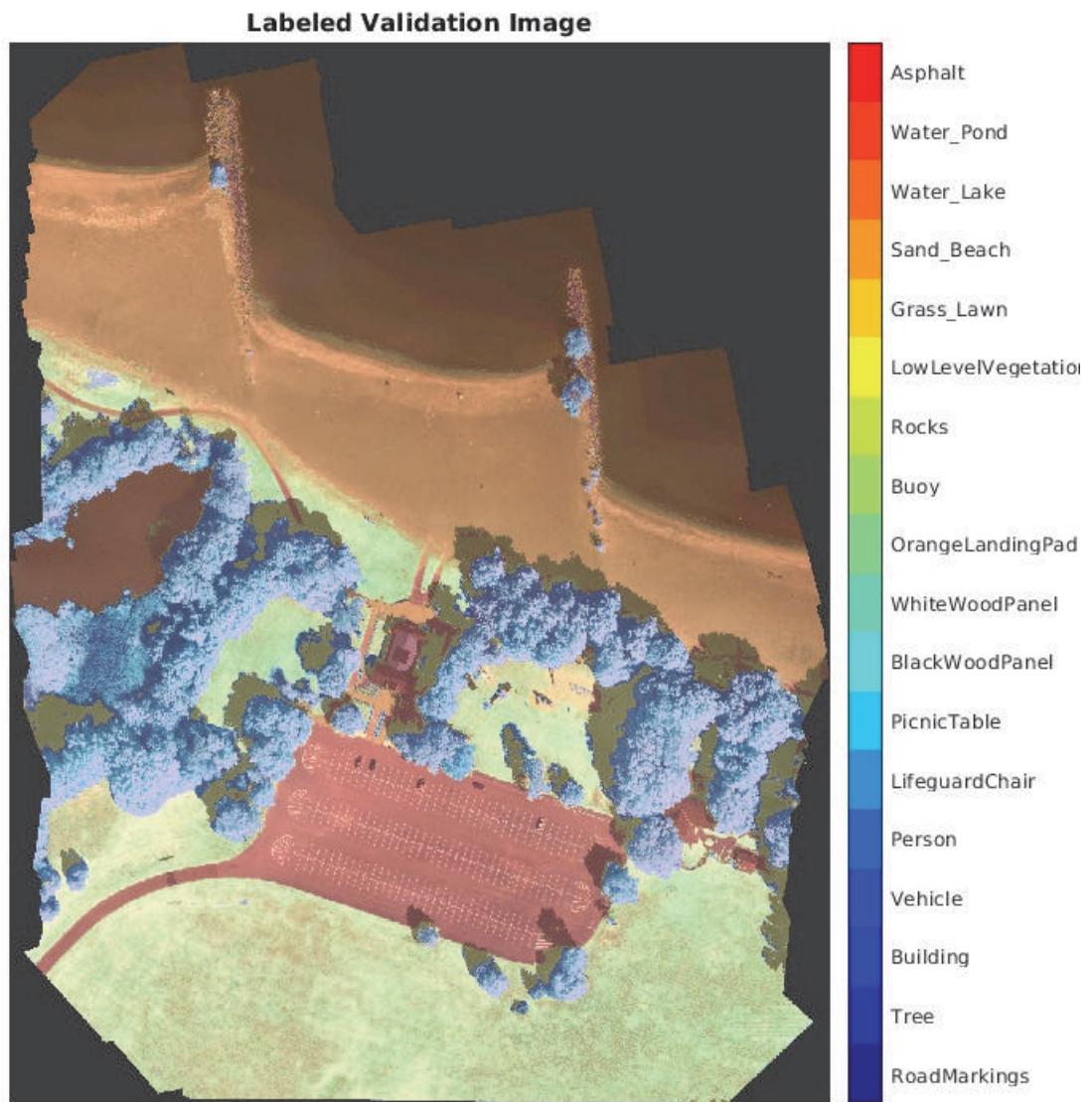


Рис. 3.30. Сегментированное изображение, наложенное на изображение с подтвержденной гистограммой RGB

Сохраните метки сегментированного изображения и метки истины земли как .PNG-файлы. Они будут использоваться для вычисления показателей точности.

```
imwrite(segmentedImage, 'results.png');
imwrite(val_labels, 'gtruth.png');
```

3.3.8 Определение точности сегментации

Создайте pixelLabelDatastore для результатов сегментации и метки истины земли.

```
pxdsResults = pixelLabelDatastore('results.png',classNames,pixelLabelIds);  
pxdsTruth = pixelLabelDatastore('gtruth.png',classNames,pixelLabelIds);
```

Измерьте глобальную точность семантической сегментации с помощью evaluateSemanticSegmentation функции.

```
Ssm = evaluateSemanticSegmentation(pxdsResults,pxdsTruth,'Metrics','global-accuracy');
```

Оценка результатов семантической сегментации:

- выбранные показатели: глобальная точность;
- обработка 1 изображения ... 100%.

Истекшее время: 00:00:25.

Предполагаемое время: 00:00:00.

- завершение Готово;
- метрики набора данных: GlobalAccuracy ... 0,90698.

Глобальная оценка точности показывает, что более 90% пикселей классифицируются правильно.

3.3.9 Рассчитать объем растительного покрова

Конечная цель этого примера - рассчитать степень покрытия растительности в мультиспектральном изображении.

Найдите количество пикселей, обозначенных растительностью. Идентификаторы ярлыков 2 («Деревья»), 13 («LowLevelVegetation») и 14 («Grass_Lawn») - это классы растительности. Также найдите общее количество действительных пикселей, суммируя пиксели в ROI изображения маски.

```
vegetationClassIds = uint8([2,13,14]);  
vegetationPixels = ismember(segmentedImage(:),vegetationClassIds);  
validPixels = (segmentedImage~=0);
```

```
numVegetationPixels = sum(vegetationPixels(:));  
numValidPixels = sum(validPixels(:));
```

Рассчитайте процент растительного покрова, разделив количество пикселов растительности на количество допустимых пикселей.

```
percentVegetationCover = (numVegetationPixels/numValidPixels)*100;  
fprintf('The percentage of vegetation cover is  
%3.2f%%.',percentVegetationCover);
```

Процент растительного покрова составляет 51,72%.

3.3.10 Резюме

В этом примере показано, как создать и обучить сеть U-Net для семантической сегментации семиканального мультиспектрального изображения. Это были шаги по обучению сети:

- загрузите и измените данные обучения;
- определите пользовательский мини-пакетный хранилище данных, называемое а PixelLabelImagePatchDatastoreиз хранилища данных изображения и хранилища данных ярлыков пикселей. Это хранилище данных использовалось для передачи данных обучения в сеть;
- определите слои сети U-Net;
- укажите варианты обучения;
- постройте сеть, используя trainNetwork функцию.

После обучения сети U-Net или загрузки предварительно установленной сети U-Net пример выполняет семантическую сегментацию данных валидации и измеряет точность сегментации.

Заключение

Тема цифровой обработки изображений, основанной на системах с использованием нейронных сетей, на сегодняшний день является весьма актуальной [3,6]. В данном методическом пособии приведен ряд примеров решения прикладных задач посредством обработки изображений разных объектов, показано, как подготовливать к обучению нейронную сеть и каким образом происходит обучение. Также уделено внимание сегментации изображений, как организовать сверточную нейронную сеть для выполнения семантической сегментации мультиспектрального изображения.

Литература

1. Гонсалес Р. Цифровая обработка изображений в среде MATLAB / Р. Гонсалес, Р. Вудс, С. Эддинс, пер. с англ. В.В. Чепыжова. М.: Техносфера, 2006. – 615с.
2. Новейшие методы обработки изображений / под ред. А.А. Потапова. М.: ФИЗМАТЛИТ, 2008. - 496с.
3. Старовойтов В.В. Цифровые изображения: от получения до обработки / В.В. Старовойтов, Ю.И. Голуб. Минск: ОИПИ НАН Беларусии, 2014. – 202с.
4. Яне Б. Цифровая обработка изображений. Москва: Техносфера, 2007. - 584с.
5. Хайкин С. Нейронные сети: полный курс, 2-е издание / пер. с англ. под ред. Н.Н.Куссуль, М.: Издательский дом «Вильямс», 2006. – 1104с.
6. Галушкин А.И. Нейронные сети: основы теории. М.: Горячая линия – Телеком, 2012. - 496с.

Учебное издание

**Костылев Владимир Иванович,
Левицкая Юлия Сергеевна**

**ОБРАБОТКА И АНАЛИЗ ИЗОБРАЖЕНИЙ
С ПОМОЩЬЮ ОБУЧЕНИЯ
НЕЙРОННЫХ СЕТЕЙ**

*Издано в авторской редакции
Компьютерная верстка Н. А. Сегида*

Подписано в печать 14.06.2019. Формат 60×84/16.
Уч.-изд. л. 5,0. Усл. печ. л. 5,2. Тираж 30 экз. Заказ 342

Издательский дом ВГУ
394018 Воронеж, пл. Ленина, 10
Отпечатано в типографии Издательского дома ВГУ
394018 Воронеж, ул. Пушкинская, 3