OCR Solutions for C# and Microsoft.Net Developers

Aquaforest OCR SDK Cookbook

Aquaforest

Version 2.30
April 2019

Contents

1 Aquaforest OCR Engine

The table below shows the prerequisites needed for building applications using the Aquaforest OCR engine.

Application Platform	VC++ Redistributable	Minimum .NET Framework Version	Minimum Visual Studio Version
x86	VC ++ 2017 x86	.NET Framework 4.5.2	Visual Studio 2012
x64	VC ++ 2017 x86 VC ++ 2017 x64		
Any CPU	VC ++ 2017 x86 VC ++ 2017 x64		

Note: If you are using Visual Studio 2012 or 2013 to build your application(s), you need to ensure that you also have the .NET Framework 4.5.2 Multi-targeting pack installed so as to be able to build applications that target .NET Framework 4.5.2.

1.1 Convert TIFF to searchable PDF

1.1.1 Requirement

Convert a multi-page TIFF file to text-searchable PDF document.

1.1.2 Solution

```
using System;
using System.IO;
using Aquaforest.OCR.Api;

namespace ConvertTIFFToSearchablePDF
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                using (Ocr ocr = new Ocr())
                {
                    string resourceFolder = @"C:\Aquaforest\OCRSDK\bin";

                    string currentEnvironmentVariables = Environment.GetEnvironmentVariable("PATH");
                    if (!currentEnvironmentVariables.Contains(resourceFolder))
                    {
                        Environment.SetEnvironmentVariable(
                            "PATH", currentEnvironmentVariables + ";" + resourceFolder);
                    }

                    // Set OCR options
                    ocr.ResourceFolder = resourceFolder;
                    ocr.Language = SupportedLanguages.English;
                    ocr.EnableConsoleOutput = true;
                    ocr.EnablePdfOutput = true;

                    // Set PreProcessor options
                    PreProcessor preProcessor = new PreProcessor();
                    preProcessor.Deskew = true;
                    preProcessor.Autorotate = false;

                    // Read source TIFF file
                    ocr.ReadTIFFSource(
                        Path.GetFullPath(@"..\..\..\..\documents\source\sample.tif"));

                    // Perform OCR recognition
                    if (ocr.Recognize(preProcessor))
                    {
                        // Save output as searchable PDF
                        ocr.SavePDFOutput(
                            Path.GetFullPath(@"..\..\..\..\documents\output\sample.pdf"), true);
                    }

                    ocr.DeleteTemporaryFiles();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
        }
    }
}
```

1.1.3 Comments

Note that the "`Autorotate`" option ensures pages will be rotated to a "top-to-bottom, left-to-right" orientation if required.

1.2 Convert BMP, JPG, PNG to searchable PDF

1.2.1 Requirement

Generate a fully searchable PDF from a BMP, PNG or JPG source file.

1.2.2 Solution

```
using System;
using System.IO;
using Aquaforest.OCR.Api;

namespace ConvertBMP_JPEG_PNGToSearchablePDF
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                using (Ocr ocr = new Ocr())
                {
                    string resourceFolder = @"C:\Aquaforest\OCRSDK\bin";
                    string currentEnvironmentVariables = Environment.GetEnvironmentVariable("PATH");
                    if (!currentEnvironmentVariables.Contains(resourceFolder))
                    {
                        Environment.SetEnvironmentVariable(
                            "PATH", currentEnvironmentVariables + ";" + resourceFolder);
                    }

                    // Set OCR options
                    ocr.ResourceFolder = resourceFolder;
                    ocr.Language = SupportedLanguages.English;
                    ocr.EnablePdfOutput = true;
                    ocr.EnableConsoleOutput = true;

                    // Set PreProcessor options
                    PreProcessor preProcessor = new PreProcessor();
                    preProcessor.Deskew = true;
                    preProcessor.Autorotate = false;

                    // Read source BMP file
                    ocr.ReadBMPSource(@"..\..\..\..\documents\source\BMPSample.bmp");

                    // Read source PNG file
                    //ocr.ReadBMPSource(@"..\..\..\..\documents\source\PNGSample.png");

                    // Read source JPEG file
                    //ocr.ReadBMPSource(@"..\..\..\..\documents\source\JPEGSample.jpg");

                    // OCR the image
                    if (ocr.Recognize(preProcessor))
                    {
                        // Save output as searchable PDF
                        ocr.SavePDFOutput(@"..\..\..\..\documents\output\output.pdf", true);
                    }

                    ocr.DeleteTemporaryFiles();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
        }
    }
}
```

1.3 OCR an image-only PDF

1.3.1 Requirement

Generate a fully searchable PDF from an existing Image PDF file.

1.3.2 Solution

```
using System;
using System.IO;
using Aquaforest.OCR.Api;

namespace OCRImagePDF
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                using (Ocr ocr = new Ocr())
                {
                    string resourceFolder = @"C:\Aquaforest\OCRSDK\bin";

                    string currentEnvironmentVariables = Environment.GetEnvironmentVariable("PATH");
                    if (!currentEnvironmentVariables.Contains(resourceFolder))
                    {
                        Environment.SetEnvironmentVariable(
                            "PATH", currentEnvironmentVariables + ";" + resourceFolder);
                    }

                    // Set OCR options
                    ocr.ResourceFolder = resourceFolder;
                    ocr.EnableConsoleOutput = true;
                    ocr.Language = SupportedLanguages.English;
                    ocr.EnablePdfOutput = true;

                    // Set PreProcessor options
                    PreProcessor preProcessor = new PreProcessor();
                    preProcessor.Deskew = true;
                    preProcessor.Autorotate = false;

                    // Read source PDF file
                    ocr.ReadPDFSource(
                        Path.GetFullPath(@"..\..\..\..\documents\source\image_pdf.pdf"));

                    // Perform OCR recognition
                    if (ocr.Recognize(preProcessor))
                    {
                        // Save output as searchable PDF
                        ocr.SavePDFOutput(
                            Path.GetFullPath(@"..\..\..\..\documents\output\searchable.pdf"), true);
                    }

                    ocr.DeleteTemporaryFiles();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
        }
    }
}
```

1.4 OCR an image-only PDF stream

1.4.1 Requirement

Generate a fully searchable PDF from an existing Image PDF stream.

1.4.2 Solution

```
using System;
using System.IO;
using Aquaforest.OCR.Api;

namespace OCRImagePDFFromStream
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                using (Ocr ocr = new Ocr())
                {
                    string resourceFolder = @"C:\Aquaforest\OCRSDK\bin";

                    string currentEnvironmentVariables = Environment.GetEnvironmentVariable("PATH");
                    if (!currentEnvironmentVariables.Contains(resourceFolder))
                    {
                        Environment.SetEnvironmentVariable(
                            "PATH", currentEnvironmentVariables + ";" + resourceFolder);
                    }

                    // Set OCR options
                    ocr.ResourceFolder = resourceFolder;
                    ocr.EnableConsoleOutput = true;
                    ocr.Language = SupportedLanguages.English;
                    ocr.EnablePdfOutput = true;

                    // Set PreProcessor options
                    PreProcessor preProcessor = new PreProcessor();
                    preProcessor.Deskew = true;
                    preProcessor.Autorotate = false;

                    using (Stream sourceStream = File.Open(
                              Path.GetFullPath(@"..\..\..\..\documents\source\image_pdf.pdf"),
                              FileMode.Open, FileAccess.Read, FileShare.Read))

                    {
                        // Read source PDF stream
                        ocr.ReadPDFSource(sourceStream);

                        // Perform OCR recognition
                        if (ocr.Recognize(preProcessor))
                        {
                            // Save output as searchable PDF stream
                            Stream outputStream;
                            ocr.SavePDFOutput(out outputStream);

                            // OR
                            // Save output as searchable PDF file
                            // ocr.SavePDFOutput(
                               Path.GetFullPath(@"..\..\..\..\documents\output\searchable.pdf"),
true);
                         }

                        ocr.DeleteTemporaryFiles();
                    }
                }
            }
            catch (Exception e)
            {
```

```
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
        }
    }
}
```

1.5 Get text from TIFFs or image-only PDFs

1.5.1 Requirement

OCR and get text from each page of a TIFF or image-only PDF file.

1.5.2 Solution

```
using System;
using System.IO;
using Aquaforest.OCR.Api;
using Aquaforest.OCR.Definitions;

namespace GetTextFromPage
{
    class Program
    {
        static Ocr ocr;

        static void Main(string[] args)
        {
            try
            {
                string resourceFolder = @"C:\Aquaforest\OCRSDK\bin";

                string currentEnvironmentVariables = Environment.GetEnvironmentVariable("PATH");
                if (!currentEnvironmentVariables.Contains(resourceFolder))
                {
                    Environment.SetEnvironmentVariable(
                        "PATH", currentEnvironmentVariables + ";" + resourceFolder);
                }

                // Set OCR options
                ocr = new Ocr();
                ocr.ResourceFolder = resourceFolder;
                ocr.EnableConsoleOutput = true;
                ocr.EnableTextOutput = true;
                ocr.Language = SupportedLanguages.English;
                ocr.StatusUpdate += new Ocr.StatusUpdateEventHandler(OcrStatusUpdate);

                // Set PreProcessor options
                PreProcessor preProcessor = new PreProcessor();
                preProcessor.Deskew = true;
                preProcessor.Autorotate = false;

                // Read source TIFF file
                ocr.ReadTIFFSource(
                    Path.GetFullPath(@"..\..\..\..\documents\source\sample.tif"));

                // Read source PDF file
                // ocr.ReadPDFSource(
                //    Path.GetFullPath(@"..\..\..\..\documents\source\image_pdf.pdf"));

                // Perform OCR recognition
                ocr.Recognize(preProcessor);

                ocr.DeleteTemporaryFiles();
            }
            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
            finally
            {
                if (ocr != null) ocr.Dispose();
            }
        }
```

```
        static void OcrStatusUpdate(object sender, StatusUpdateEventArgs pageCompletedEventArgs)
        {
            Console.WriteLine("Page {0}:", pageCompletedEventArgs.PageNumber);

            if (pageCompletedEventArgs.TextAvailable)
            {
                Console.WriteLine(ocr.ReadPageString(pageCompletedEventArgs.PageNumber));
            }
            else
            {
                Console.WriteLine("No text found");
            }
        }
    }
}
```

1.5.3 Comments

The `ReadPageStringMethod` takes the page number as an argument and returns all the text extracted from the page as a string.

Note: This example will only get text from image-only pages. If a page already had text in it prior to the OCR, that text won't be retrieved. The following example shows how to extract text from image-only, partially searchable and fully searchable documents.

1.6 Extract text from Image-only, Partially Searchable and Fully Searchable PDFs

1.6.1 Requirement

Get text from each page of a partially searchable or fully searchable PDF document. If a page is image-only, it should be OCRed first.

1.6.2 Solution

```
using Aquaforest.OCR.Api;
using Aquaforest.OCR.Definitions;
using Aquaforest.PDF;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;

namespace ExtractText
{
    class Program
    {
        static Ocr ocr;

        static void Main(string[] args)
        {
            DirectoryInfo directory =
                new DirectoryInfo(@"C:\Aquaforest\OCRSDK\samples\documents\source");

            string outputDirectory = @"C:\Aquaforest\OCRSDK\samples\documents\output";
            if (!Directory.Exists(outputDirectory)) Directory.CreateDirectory(outputDirectory);

            foreach (var file in directory.GetFiles("*.pdf", SearchOption.AllDirectories))
            {
                Console.WriteLine("Processing PDF file: {0}", file.FullName);

                var documentPagesText = new Dictionary<int, string>();
                bool hasImageOnlyPage = CheckDocumentUsingToolkit(file.FullName, documentPagesText);

                if (hasImageOnlyPage)
                {
                    OcrPdf(file.FullName, documentPagesText);
                }

                if (documentPagesText.Count > 0)
                {
                    string output =
                        Path.Combine(outputDirectory, Path.ChangeExtension(file.Name, "txt"));
                    WriteTextToFile(output, documentPagesText);
                }
                else
                {
                    Console.WriteLine("No text found.");
                }

                Console.WriteLine("");
            }
        }

        /// <summary>
        /// Check each page of source PDF document to see if it has any text. If there is text,
        /// store it in a dictionary or else set the hasImageOnlyPage flag to true.
        /// </summary>
        static bool CheckDocumentUsingToolkit(string source,
            Dictionary<int, string> documentPagesText)
        {
```

```
        PDFDocument doc = null;
        bool hasImageOnlyPage = false;

        try
        {
            doc = new PDFDocument(source);
            for (int i = 1; i <= doc.NumberOfPages; i++)
            {
                string pageText = doc.GetText(i);
                documentPagesText[i] = pageText;

                // If the document has at least 1 image-only page, it will be OCRed
                if (!hasImageOnlyPage)
                {
                    // Check if this page is image-only.
                    // ASSUMPTION: if the page doesn't have any text, it is considered as
                    // image-only, even if it is a blank page with no images
                    hasImageOnlyPage = string.IsNullOrEmpty(
                        pageText.Replace("\r\n", "").Replace("\n", "").Replace("\r", ""));
                }
            }
        }
        catch (Exception e)
        {
            Console.WriteLine("Error reading PDF file: {0}", e.Message);
        }
        finally
        {
            if (doc != null) doc.Close();
        }

        return hasImageOnlyPage;
    }


    static bool OcrPdf(string source,
        Dictionary<int, string> documentPagesText)
    {
        try
        {
            Console.WriteLine("OCRing...");

            using (ocr = new Ocr())
            {
                string resourceFolder = @"C:\Aquaforest\OCRSDK\bin";

                string currentEnvironmentVariables =
                Environment.GetEnvironmentVariable("PATH");
                if (!currentEnvironmentVariables.Contains(resourceFolder))
                {
                    Environment.SetEnvironmentVariable("PATH",
                        currentEnvironmentVariables + ";" + resourceFolder);
                }

                ocr.ResourceFolder = resourceFolder;
                ocr.EnableConsoleOutput = true;
                ocr.Language = SupportedLanguages.English;
                ocr.EnablePdfOutput = true;
                ocr.StatusUpdate += (sender, pageCompleteEventArgs) =>
                    OcrStatusUpdate(sender, pageCompleteEventArgs, documentPagesText);

                PreProcessor preProcessor = new PreProcessor();
                preProcessor.Deskew = true;
                preProcessor.Autorotate = false;

                ocr.ReadPDFSource(source);

                ocr.Recognize(preProcessor);
```

```
                ocr.DeleteTemporaryFiles();

                return true;
            }
        }
        catch (Exception e)
        {
            Console.WriteLine("Error in OCR Processing : {0}", e.Message);
            return false;
        }
    }

    static void OcrStatusUpdate(object sender,
        StatusUpdateEventArgs pageCompleteEventArgs,
        Dictionary<int, string> documentPagesText)
    {
        if (pageCompleteEventArgs.TextAvailable)
        {
            int pageNumber = pageCompleteEventArgs.PageNumber;
            string pageString = ocr.ReadPageString(pageNumber);

            // Optional
            string existingText;
            if (documentPagesText.TryGetValue(pageNumber, out existingText))
            {
                if (!string.IsNullOrEmpty(existingText))
                {
                    pageString += Environment.NewLine + existingText;
                }
            }

            documentPagesText[pageNumber] = pageString;
        }
    }

    static void WriteTextToFile(string output,
        Dictionary<int, string> documentPagesText)
    {
        Console.WriteLine("Writing text to: {0}", output);

        for (int i = 1; i <= documentPagesText.Keys.Max(); i++)
        {
            string pageText;
            if (documentPagesText.TryGetValue(i, out pageText))
            {
                File.AppendAllText(output, pageText + Environment.NewLine);
            }
        }
    }
  }
}
```

1.6.3 Comments

This is a very basic example and has room for improvements/optimizations.

Note: This example requires the Aquaforest PDF Toolkit as well as the OCR SDK.

1.7 Convert folder of files

1.7.1 Requirement

Convert multiple TIFF or PDF files located in a folder.

1.7.2 Solution

```
using System;
using System.IO;
using Aquaforest.OCR.Api;

namespace ConvertFolderOfFiles
{
    class Program
    {
        static void Main(string[] args)
        {
            string resourceFolder = @"C:\Aquaforest\OCRSDK\bin";

            string currentEnvironmentVariables = Environment.GetEnvironmentVariable("PATH");
            if (!currentEnvironmentVariables.Contains(resourceFolder))
            {
                Environment.SetEnvironmentVariable(
                    "PATH", currentEnvironmentVariables + ";" + resourceFolder);
            }

            // Set PreProcessor options
            PreProcessor preProcessor = new PreProcessor();
            preProcessor.Deskew = true;
            preProcessor.RemoveLines = true;

            string sourcePath = @"..\..\..\..\documents\source";
            string outputPath = @"..\..\..\..\documents\output";

            if (!Directory.Exists(outputPath)) Directory.CreateDirectory(outputPath);

            // folder containing your tiff files
            DirectoryInfo dInfo = new DirectoryInfo(sourcePath);
            foreach (FileInfo f in dInfo.GetFiles("*.tif"))
            {
                Console.WriteLine("Processing file: {0}", f.FullName);

                using (Ocr ocr = new Ocr())
                {
                    // Set OCR options
                    ocr.ResourceFolder = resourceFolder;
                    ocr.Language = SupportedLanguages.English;
                    ocr.EnableConsoleOutput = true;
                    ocr.EnablePdfOutput = true;

                    // Read source TIFF file
                    ocr.ReadTIFFSource(f.FullName);

                    if (ocr.Recognize(preProcessor)) // Perform OCR recognition
                    {
                        // Save output as searchable PDF
                        ocr.SavePDFOutput(
                            Path.Combine(outputPath, Path.ChangeExtension(f.Name, ".pdf")), true);
                    }

                    ocr.DeleteTemporaryFiles();
                }
            }
        }
    }
}
```

1.7.3 Comments

To process PDF files, change the following:

- "`foreach (FileInfo f in dInfo.GetFiles("*.tif"))`" to "`foreach (FileInfo f in dInfo.GetFiles("*.pdf"))`"
- `ocr.ReadTIFFSource(f.FullName)` to `ocr.ReadPDFSource(f.FullName)`

1.8 Zonal OCR

1.8.1 Requirement

Extract text from a specific area of an image

1.8.2 Solution

```
using System;
using System.Drawing;
using Aquaforest.OCR.Api;
using System.IO;

namespace ZonalOCR
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                using (Ocr ocr = new Ocr())
                {
                    string resourceFolder = @"C:\Aquaforest\OCRSDK\bin";

                    string currentEnvironmentVariables = Environment.GetEnvironmentVariable("PATH");
                    if (!currentEnvironmentVariables.Contains(resourceFolder))
                    {
                        Environment.SetEnvironmentVariable(
                            "PATH", currentEnvironmentVariables + ";" + resourceFolder);
                    }

                    ocr.ResourceFolder = resourceFolder;
                    ocr.Language = SupportedLanguages.English;
                    ocr.EnablePdfOutput = true;
                    ocr.EnableConsoleOutput = true;
                    ocr.License = "";

                    PreProcessor preProcessor = new PreProcessor();
                    preProcessor.Deskew = true;
                    preProcessor.Autorotate = false;

                    // Variables for the coordinates
                    int x, y, height, width;

                    // Assign values for the coordinates of the area you wish to convert
                    x = 320; y = 210; height = 210; width = 900;
                    //Create a new rectangle object thta represents the zone you are interested in
                    Rectangle region = new Rectangle(x, y, width, height);

                    // Read the image for processing
                    ocr.ReadTIFFSource(
                         Path.GetFullPath(@"..\..\..\..\..\..\docs\tiffs\sample.tif"));

                    // OCR the image
                    if (ocr.Recognize(preProcessor))
                    {
                        Console.WriteLine("Recognized : ");
                        // Use the page number and the region to get the text in that specific zone.
                        Console.WriteLine(ocr.ReadPageString(1, region));
                    }

                    ocr.DeleteTemporaryFiles();
                }
            }
            catch (Exception e)
            {
```

```
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
        }
    }
}
```

1.8.3 Comments

There are two methods that can be used to achieve zonal OCR – either using the built-in methods to identify text within particular co-ordinates or by explicitly cropping the image to the area of interest.

The above code shows how to get text from a particular area of a file using the built in method.
Firstly, a rectangle representing the zone that needs to be extracted should be created; this rectangle along with the page number should be passed to the `ocr.ReadPageString` method. This method can be used for both PDF and image files.

The next method is shown in the next section.

1.9 Processing zones with different orientations

1.9.1 Requirement

OCR a document that contains pages that have text in multiple orientations.

1.9.2 Solution

```
static void Main(string[] args)
{
    try
    {
        using (var img =

Image.FromFile(@"..\..\..\..\documents\source\different_text_orientations.tif", true))
        using (var bmp = new Bitmap(img))
        {
            // Variables for the coordinates
            int x, y, height, width;

            // First example - United State Patent block.
            // Assign values for the coordinates of the area you wish to convert
            x = 320; y = 110; height = 200; width = 690;

            // Crop the image
            using (Bitmap croppedBmp =
                        bmp.Clone(new Rectangle(x, y, width, height), img.PixelFormat))
            {
                croppedBmp.SetResolution(img.HorizontalResolution, img.VerticalResolution);

                string zoneText = GetZoneText(croppedBmp, false);
                Console.WriteLine("Zone 1: " + Environment.NewLine + zoneText);

            }

            // Second example - Inventor Details block.
            // Assign values for the coordinates of the area you wish to convert
            x = 65; y = 465; height = 650; width = 160;

            //Crop the image
            using (var croppedBmp =
                            bmp.Clone(new Rectangle(x, y, width, height), bmp.PixelFormat))
            {
                croppedBmp.SetResolution(img.HorizontalResolution, img.VerticalResolution);

                // Set autorotate = true so that the text will be read in the correct orientation.
                string zoneText = GetZoneText(croppedBmp, true);
                Console.WriteLine("Zone 2: " + Environment.NewLine + zoneText);
            }
        }
    }
    catch (Exception e)
    {
        Console.WriteLine("Error in OCR Processing :" + e.Message);
    }
}


private static string GetZoneText(Bitmap bmp, bool autorotate)
{
    using (Ocr ocr = new Ocr())
    {
        string ocredText = "";

        try
        {
            string resourceFolder = Path.GetFullPath(@"..\..\..\..\..\bin\");
```

```
            string currentEnvironmentVariables = Environment.GetEnvironmentVariable("PATH");
            if (!currentEnvironmentVariables.Contains(resourceFolder))
            {
                Environment.SetEnvironmentVariable("PATH",
                     currentEnvironmentVariables + ";" + resourceFolder);
            }

            // Set OCR options
            ocr.ResourceFolder = resourceFolder;
            ocr.Language = SupportedLanguages.English;
            ocr.EnableConsoleOutput = true;
            ocr.EnableTextOutput = true;

            // Set PreProcessor options
            PreProcessor preProcessor = new PreProcessor();
            preProcessor.Deskew = true;
            preProcessor.Autorotate = autorotate;

            ocr.ReadImageSource(bmp);

            if (ocr.Recognize(preProcessor))
            {
                ocredText = ocr.ReadPageString(1);
            }
        }
        catch (Exception e)
        {
            Console.WriteLine("Error in OCR Processing :" + e.Message);
        }
        finally
        {
            ocr.DeleteTemporaryFiles();
        }

        return ocredText;
    }
}
```

1.9.3 Comments

This example application demonstrates how the SDK may be used to process specific image zones. The examples use the two zones, one of which is at a 90 degree orientation to the other.

The process required to deal with these zones are as follows:

1. Identify the zone by finding the top left X and Y coordinates of the zone together with the width and height of the zone. This should be any rotation needed to fit with the sample code.
2. Create code based around the model shown below to process each zone by creating a cropped copy of the original image, rotating where necessary and then performing recognition.

1.10 Multithreading

1.10.1 Requirement

Use multiple threads to OCR documents

1.10.2 Solution

```
using Aquaforest.OCR.Api;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MultiThread
{
    class Program
    {
        private static string outputFolder = Path.GetFullPath(@"..\..\..\..\documents\output");
        private static string inputFolder = Path.GetFullPath(@"..\..\..\..\documents\source");
        private static int maxDegreeOfParallelism = 2;
        private static string resourceFolder;
        private static string tempPath;

        static void Main(string[] args)
        {
            tempPath = Path.Combine(Path.GetTempPath(), "MultiThread");
            resourceFolder = Path.GetFullPath(@"..\..\..\..\..\bin");

            string currentEnvironmentVariables = Environment.GetEnvironmentVariable("PATH");
            if (!currentEnvironmentVariables.Contains(resourceFolder))
            {
                Environment.SetEnvironmentVariable("PATH",
                    currentEnvironmentVariables + ";" + resourceFolder);
            }

            Parallel.ForEach(Directory.EnumerateFiles(inputFolder, "*.tif"),
                new ParallelOptions { MaxDegreeOfParallelism = maxDegreeOfParallelism }, f =>
            {
                var outputFilePath = Path.Combine(outputFolder,
                    Path.GetFileNameWithoutExtension(f) + ".pdf");
                if (File.Exists(outputFilePath)) return;

                try
                {
                    ProcessDocument(f, outputFilePath);
                }
                catch (Exception e)
                {
                    Console.WriteLine("Error: " + e.Message);
                }
            });
        }

        static void ProcessDocument(string source, string output)
        {
            string tempFolder = Path.Combine(tempPath, Guid.NewGuid().ToString());

            try
            {
                using (Ocr ocr = new Ocr())
                {
                    ocr.ResourceFolder = resourceFolder;

                    // Set OCR options
```

```
                    ocr.TempFolder = tempFolder;
                    ocr.EnablePdfOutput = true;
                    ocr.Language = SupportedLanguages.English;

                    // Set PreProcessor options
                    PreProcessor preProcessor = new PreProcessor();
                    preProcessor.Deskew = true;

                    // Read source TIFF file
                    ocr.ReadTIFFSource(source);

                    // Perform OCR recognition
                    if (ocr.Recognize(preProcessor))
                    {
                        // Save output as searchable PDF
                        ocr.SavePDFOutput(output, true);

                        Console.WriteLine(source + " successfully processed.");
                    }
                    else
                    {
                        Console.WriteLine(source + " Failed");
                    }
                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
            finally
            {
                if (Directory.Exists(tempFolder)) Directory.Delete(tempFolder, true);
            }
        }
    }
}
```

1.10.3 Comments

The number of threads (`MaxDegreeOfParallelism`) allowed depends on your license.

There is also a VB.NET example which can be found in the samples folder: [SDK installation path]\samples\vb.net\VB.NET Multi-thread.

1.11 Determine Word coordinates

1.11.1 Requirement

Get the location of each word recognized by the SDK.

1.11.2 Solution

```
using System;
using System.IO;
using Aquaforest.OCR.Api;
using Aquaforest.OCR.Definitions;

namespace DetermineWordCoordinates
{
    class Program
    {
        static Ocr ocr;

        static void Main(string[] args)
        {
            try
            {
                string resourceFolder = @"C:\Aquaforest\OCRSDK\bin";

                string currentEnvironmentVariables = Environment.GetEnvironmentVariable("PATH");
                if (!currentEnvironmentVariables.Contains(resourceFolder))
                {
                    Environment.SetEnvironmentVariable(
                        "PATH", currentEnvironmentVariables + ";" + resourceFolder);
                }

                // Set OCR options
                ocr = new Ocr();
                ocr.ResourceFolder = resourceFolder;
                ocr.EnableConsoleOutput = true;
                ocr.EnableTextOutput = true;
                ocr.Language = SupportedLanguages.English;
                ocr.StatusUpdate += new Ocr.StatusUpdateEventHandler(OcrStatusUpdate);

                // Set PreProcessor options
                PreProcessor preProcessor = new PreProcessor();
                preProcessor.Deskew = true;
                preProcessor.Autorotate = false;

                // Read source TIFF file
                ocr.ReadTIFFSource(Path.GetFullPath(@"..\..\..\..\documents\source\sample.tif"));

                // Perform OCR recognition
                ocr.Recognize(preProcessor);

                ocr.DeleteTemporaryFiles();
            }
            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
            finally
            {
                if (ocr != null) ocr.Dispose();
            }
        }

        static void OcrStatusUpdate(object sender, StatusUpdateEventArgs pageCompletedEventArgs)
        {
            Console.WriteLine("Page {0}:", pageCompletedEventArgs.PageNumber);
```

```
            Words words = ocr.ReadPageWords(pageCompletedEventArgs.PageNumber);

            foreach (WordData w in words)
            {
                Console.WriteLine("{0} [{1}, {2}]", w.Word, w.Left, w.Bottom);
            }
        }
    }
}
```

1.11.3 Comments

The above code uses the `StatusUpdate` event of the OCR SDK to find out if a page has been processed. After each page is processed, the `ocrStatusUpdate` event handler is invoked. This method then gets all the words in the page as objects of `WordData`. The `WordData` object holds the coordinates information of the words.

1.12 Read characters after a given string

1.12.1 Requirement

Ability to read characters after a given string – for example reading a number after "Invoice:"

1.12.2 Solution

```
using System;
using System.IO;
using Aquaforest.OCR.Api;

namespace WordReader
{
    class ReadCharactersAfterAGivenString
    {
        static void Main(string[] args)
        {
            string result = "output";

            //Ocr the document and retrive the text as a string object
            string pageString = GetPageString(1, @"..\..\..\..\documents\source\invoice.tif");

            if (pageString != "failed")
            {
                // Get the Invoice Number
                if (pageString.Contains("Invoice Number: "))
                {
                    // Extract the next word(s) after "Invoice Number: "
                    // until the next new line character.
                    result = GetValueOfStringByKeyword(pageString, "Invoice Number: ",
                        Environment.NewLine);
                    Console.WriteLine("Invoice Number : " + result);
                }

                // Get the Address
                if (pageString.Contains("Address:"))
                {
                    // Extract the next word(s) after "Address:"
                    // until the next new line character
                    var result1 = GetValueOfStringByKeyword(pageString, "Address:",
                        Environment.NewLine);
                    Console.WriteLine("Address: " + result1);
                }
            }
        }

        /// <param name="wordlist">The text of a the page</param>
        /// <param name="startString">Keyword, e.g. Name, Invoice No.</param>
        /// <param name="stopString">The string to stop at.</param>
        /// <returns>The value of the keyword (the text between the startString
        /// and the stopString)</returns>
        public static string GetValueOfStringByKeyword(string wordlist, string startString,
            string stopString)
        {
            var subset = wordlist.Substring(wordlist.IndexOf(startString));
            subset = ReplaceFirst(subset, startString, " ");
            var finalResult = subset.TrimStart();
            finalResult = finalResult.Substring(0, finalResult.IndexOf(stopString));

            if (finalResult == string.Empty)
            {
                return "Could not find the word: " + startString;
```

```
        }
        return finalResult;
    }

    public static string ReplaceFirst(string text, string search, string replace)
    {
        int pos = text.IndexOf(search);
        if (pos < 0)
        {
            return text;
        }
        return text.Substring(0, pos) + replace + text.Substring(pos + search.Length);
    }

    public static string GetPageString(int pageNumber, string inputFile)
    {
        string pageString = "";
        try
        {
            using (Ocr ocr = new Ocr())
            {
                string resourceFolder = Path.GetFullPath(@"..\..\..\..\bin\");

                string currentEnvironmentVariables =
                    Environment.GetEnvironmentVariable("PATH");
                if (!currentEnvironmentVariables.Contains(resourceFolder))
                {
                    Environment.SetEnvironmentVariable("PATH",
                        currentEnvironmentVariables + ";" + resourceFolder);
                }

                // Set OCR options
                ocr.ResourceFolder = resourceFolder;
                ocr.License = "";
                ocr.EnableConsoleOutput = true;
                ocr.EnableTextOutput = true;

                // Set PreProcessor options
                PreProcessor preProcessor = new PreProcessor();
                preProcessor.Deskew = true;
                preProcessor.Autorotate = false;
                preProcessor.Binarize = -1;
                preProcessor.Morph = "c2.2";

                ocr.Language = SupportedLanguages.English;

                if (Path.GetExtension(inputFile).ToLower() == ".tif")
                {
                    // Read source TIFF file
                    ocr.ReadTIFFSource(Path.GetFullPath(inputFile));
                }
                else if (Path.GetExtension(inputFile).ToLower() == ".pdf")
                {
                    // Read source PDF file
                    ocr.ReadPDFSource(Path.GetFullPath(inputFile));
                }
                else
                {
                    return "failed";
                }

                // Perform OCR recognition
                if (ocr.Recognize(preProcessor))
                {
```

```
                        // Save output as searchable PDF
                        pageString = ocr.ReadPageString(pageNumber);
                    }
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
                pageString = "failed";
            }

            return pageString;
        }
    }
}
```

1.12.3 Comments

The above code shows the user how to read string from a page using the `ocr.ReadPageString` method. It analyzes the string and returns some information such as address and invoice numbers. The `GetValueOfStringByKeyword` method takes some agruements and performs these analysis to extract some special information.

1.13 Write recognition results into a CSV file

1.13.1 Requirement

Extract words after a given string/field and write them to a CSV file.

1.13.2 Solution

```
using System;
using System.IO;
using Aquaforest.OCR.Api;
using Aquaforest.OCR.Definitions;

namespace RecognitionToExcel
{
    class Program
    {
        static string csvOutputFilePath = @"..\..\..\..\documents\output\invoices.csv";

        static void Main(string[] args)
        {
            string resourceFolder = @"C:\Aquaforest\OCRSDK\bin";

            string currentEnvironmentVariables = Environment.GetEnvironmentVariable("PATH");
            if (!currentEnvironmentVariables.Contains(resourceFolder))
            {
                Environment.SetEnvironmentVariable(
                    "PATH", currentEnvironmentVariables + ";" + resourceFolder);
            }

            // Set PreProcessor options
            PreProcessor preProcessor = new PreProcessor();
            preProcessor.Deskew = true;
            preProcessor.RemoveLines = true;

            InitialiseExcel();

            DirectoryInfo dInfo = new DirectoryInfo(@"..\..\..\..\documents\source\invoices");

            foreach (FileInfo f in dInfo.GetFiles("*.tiff"))
            {
                using (Ocr ocr = new Ocr())
                {
                    // Set OCR options
                    ocr.ResourceFolder = resourceFolder;
                    ocr.Language = SupportedLanguages.English;
                    ocr.EnableConsoleOutput = true;
                    ocr.EnableTextOutput = true;

                    Console.WriteLine("Processing file: {0}", f.FullName);
                    ocr.ReadTIFFSource(f.FullName); // Read source TIFF file

                    // Perform OCR recognition
                    if (!ocr.Recognize(preProcessor)) continue;

                    bool[] skip = new bool[3];
                    ExcelRow row = new ExcelRow();

                    // Below is one method of getting text after a string or set of strings.
                    // Alternatively, you can use the method used in example 8 of the
                    // Aquaforest OCR SDK Cookbook - "Read Characters after a given string".
                    // The full Visual Studio project for that example can be found at
                    // '[SDK installation path]\samples\csharp\ReadCharactersAfterAGivenString'
                    Words words = ocr.ReadPageWords(1);

                    for (int i = 0; i < words.Count; i++)
```

```
                {
                    if (!skip[0] && (words[i].Word == "Invoice" && (i + 1 < words.Count)
                        && words[i + 1].Word == "Number:"))
                    {
                        if (i + 2 < words.Count)
                        {
                            row.InvoiceNumber = words[i + 2].Word;
                            i = i + 2;
                            skip[0] = true;
                            continue;
                        }
                    }

                    if (!skip[1] && (words[i].Word == "Invoice" && (i + 1 < words.Count)
                        && words[i + 1].Word == "Date:"))
                    {
                        if (i + 2 < words.Count)
                        {
                            row.InvoiceDate = words[i + 2].Word;
                            i = i + 2;
                            skip[1] = true;
                            continue;
                        }
                    }

                    if (!skip[2] && (words[i].Word == "Customer" && (i + 1 < words.Count)
                        && words[i + 1].Word == "Information:"))
                    {
                        int j = 2;

                        while (i + j < words.Count)
                        {
                            if (words[i + j].Word == "Billing")
                            {
                                i = i + j;
                                skip[2] = true;
                                break;
                            }
                            else
                            {
                                row.CustomerName += words[i + j].Word + " ";
                            }

                            j++;
                        }
                    }

                    if (skip[0] && skip[1] && skip[2])
                    {
                        break;
                    }
                }

                WriteRow(row);
            }
        }
    }

    static void InitialiseExcel()
    {
        File.WriteAllText(csvOutputFilePath,
            "Invoice Number,Invoice Date,Customer Name" + Environment.NewLine);
    }

    static void WriteRow(ExcelRow row)
    {
        string contents = string.Format("{0},{1},{2}{3}",
            row.InvoiceNumber, row.InvoiceDate, row.CustomerName, Environment.NewLine);
```

```
            Console.WriteLine(contents);
            File.AppendAllText(csvOutputFilePath, contents);
        }
    }

    class ExcelRow
    {
        public string InvoiceNumber { get; set; }
        public string InvoiceDate { get; set; }
        public string CustomerName { get; set; }
    }
}
```

1.13.3 Comments

This example will produce accurate results only for the files given in the folder "[Installation path]\samples\documents\source\invoices". You will need to make some amendments to the code for documents with different formatting or template.

1.14 Process color images

1.14.1 Requirement

Optimally process color images for recognition and resultant PDF size by using binarization and MRC compression.

1.14.2 Solution

```
using System;
using System.IO;
using Aquaforest.OCR.Api;

namespace ColorImages
{
    class ColorImages
    {
        static void Main(string[] args)
        {
            try
            {
                using (Ocr ocr = new Ocr())
                {
                    string resourceFolder = @"C:\Aquaforest\OCRSDK\bin";

                    string currentEnvironmentVariables = Environment.GetEnvironmentVariable("PATH");
                    if (!currentEnvironmentVariables.Contains(resourceFolder))
                    {
                        Environment.SetEnvironmentVariable(
                            "PATH", currentEnvironmentVariables + ";" + resourceFolder);
                    }

                    // Set OCR options
                    ocr.ResourceFolder = resourceFolder;
                    ocr.EnableConsoleOutput = true;
                    ocr.Language = SupportedLanguages.English;
                    ocr.EnablePdfOutput = true;

                    // Set PreProcessor options
                    PreProcessor preProcessor = new PreProcessor();
                    preProcessor.Deskew = true;
                    preProcessor.Autorotate = false;
                    preProcessor.MRC = true; // Set compression to true

                    // Separate the text from any background images and colors.
                    preProcessor.Binarize = -1;

                    // Read source TIFF file
                    ocr.ReadTIFFSource(
                        Path.GetFullPath(@"..\..\..\..\documents\source\color.tif"));

                    // Perform OCR recognition
                    if (ocr.Recognize(preProcessor))
                    {
                        // Save output as searchable PDF
                        ocr.SavePDFOutput(
                            Path.GetFullPath(@"..\..\..\..\documents\output\color.pdf"), true);
                    }

                    // Clean up temporary files
                    ocr.DeleteTemporaryFiles();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
```

```
            }
        }
    }
}
```

1.14.3 Comments

The above code shows how you can pre-process a color image file to obtain the best OCR results and also to reduce the size of the output PDF.

First setting the `preProcessor.Binarize` to `-1` instructs the engine to separate the text from background images and color. Then setting the `preProcessor.MRC = true` will cause a dramatic reduction in the size of the output PDF.

1.15 Apply compression to output PDFs

1.15.1 Requirement

Optimally process images for resultant PDF size by using MRC compression.

1.15.2 Solution

```
using System;
using System.IO;
using Aquaforest.OCR.Api;

namespace PDFCompression
{
    class PDFCompression
    {
        static void Main(string[] args)
        {
            try
            {
                using (Ocr ocr = new Ocr())
                {
                    string resourceFolder = @"C:\Aquaforest\OCRSDK\bin";

                    string currentEnvironmentVariables = Environment.GetEnvironmentVariable("PATH");
                    if (!currentEnvironmentVariables.Contains(resourceFolder))
                    {
                        Environment.SetEnvironmentVariable(
                            "PATH", currentEnvironmentVariables + ";" + resourceFolder);
                    }

                    // Set OCR options
                    ocr.ResourceFolder = resourceFolder;
                    ocr.EnableConsoleOutput = true;
                    ocr.Language = SupportedLanguages.English;
                    ocr.EnablePdfOutput = true;

                    // Set PreProcessor options
                    PreProcessor preProcessor = new PreProcessor();
                    preProcessor.Deskew = true;
                    preProcessor.Autorotate = false;
                    preProcessor.MRC = true; //Set compression to true

                    // Read source TIFF file
                    ocr.ReadTIFFSource(Path.GetFullPath(@"..\..\..\..\documents\source\sample.tif"));

                    // Perform OCR recognition
                    if (ocr.Recognize(preProcessor))
                    {
                        // Save output as searchable PDF
                        ocr.SavePDFOutput(Path.GetFullPath(
                            @"..\..\..\..\documents\output\samplecompressed.pdf"), true);
                    }

                    ocr.DeleteTemporaryFiles();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
        }
    }
}
```

1.15.3 Comments

Setting `preProcessor.MRC = true` will cause a dramatic reduction in the size of the output PDF.

1.16 Merging PDFs

1.16.1 Requirement

Assemble a PDF from multiple individual PDFs – usually as a result of OCRing a set of single page PDFs or TIFFs.

1.16.2 Solution

```
using System;
using System.IO;
using Aquaforest.OCR.Api;

namespace MergingPDFs
{
    class Program
    {
        static void Main(string[] args)
        {
            string resourceFolder = @"C:\Aquaforest\OCRSDK\bin";

            string currentEnvironmentVariables = Environment.GetEnvironmentVariable("PATH");
            if (!currentEnvironmentVariables.Contains(resourceFolder))
            {
                Environment.SetEnvironmentVariable(
                    "PATH", currentEnvironmentVariables + ";" + resourceFolder);
            }

            // Set PreProcessor options
            PreProcessor preProcessor = new PreProcessor();
            preProcessor.Deskew = true;
            preProcessor.RemoveLines = true;

            string sourcePath = @"..\..\..\..\documents\source";
            string outputPath = @"..\..\..\..\documents\output\pdfs_to_merge";

            if (!Directory.Exists(outputPath)) Directory.CreateDirectory(outputPath);

            // folder containing your tiff files
            DirectoryInfo dInfo = new DirectoryInfo(sourcePath);
            foreach (FileInfo f in dInfo.GetFiles("*.tif"))
            {
                Console.WriteLine("Processing file: {0}", f.FullName);

                using (Ocr ocr = new Ocr())
                {
                    // Set OCR options
                    ocr.ResourceFolder = resourceFolder;
                    ocr.Language = SupportedLanguages.English;
                    ocr.EnableConsoleOutput = true;
                    ocr.EnablePdfOutput = true;

                    // Read source TIFF file
                    ocr.ReadTIFFSource(f.FullName);

                    // Perform OCR recognition
                    if (ocr.Recognize(preProcessor))
                    {
                        // Save output as searchable PDF
                        ocr.SavePDFOutput(
                            Path.Combine(outputPath, Path.ChangeExtension(f.Name, ".pdf")), true);
                    }

                    ocr.DeleteTemporaryFiles();
                }
            }
```

```
            Console.WriteLine("Merging...");
            dInfo = new DirectoryInfo(outputPath);
            PdfMerger merger = new PdfMerger(@"..\..\..\..\documents\output\merge.pdf");

            foreach (FileInfo f in dInfo.GetFiles("*.pdf"))
            {
                merger.Append(f.FullName);
            }

            merger.Close();
            dInfo.Delete(true);
        }
    }
}
```

1.17 OCR an in-memory image

1.17.1 Requirement

Perform OCR recognition on an image that is held in memory rather than in a file.

1.17.2 Solution

```
using System;
using System.Drawing;
using System.IO;
using Aquaforest.OCR.Api;

namespace OCRImagesInMemory
{
    class OCRImagesInMemory
    {
        static void Main(string[] args)
        {
            try
            {
                using (Ocr ocr = new Ocr())
                {
                    string resourceFolder = @"C:\Aquaforest\OCRSDK\bin";

                    string currentEnvironmentVariables = Environment.GetEnvironmentVariable("PATH");
                    if (!currentEnvironmentVariables.Contains(resourceFolder))
                    {
                        Environment.SetEnvironmentVariable(
                            "PATH", currentEnvironmentVariables + ";" + resourceFolder);
                    }

                    // Set OCR options
                    ocr.ResourceFolder = resourceFolder;
                    ocr.Language = SupportedLanguages.English;
                    ocr.EnablePdfOutput = true;
                    ocr.EnableConsoleOutput = true;

                    // Set PreProcessor options
                    PreProcessor preProcessor = new PreProcessor();
                    preProcessor.Deskew = true;
                    preProcessor.Autorotate = false;

                    Image image = Image.FromFile(@"..\..\..\..\documents\source\sample.tif");

                    ocr.ReadImageSource(image);

                    // Perform OCR recognition
                    if (ocr.Recognize(preProcessor))
                    {
                        // Save output as searchable PDF
                        ocr.SavePDFOutput(@"..\..\..\..\documents\output\inMemorySample.pdf", true);
                    }

                    ocr.DeleteTemporaryFiles();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
        }
    }
}
```

1.17.3 Comments

The above sample shows how you can load an image file into a C# image object in memory before you OCR it. You can also perform operations like cropping and rotating the image before you OCR it.

1.18 OCR an image stream

1.18.1 Requirement

Perform OCR recognition on an image that is held in a stream rather than in a file.

1.18.2 Solution

```
using System;
using System.IO;
using Aquaforest.OCR.Api;

namespace OCRImageFromStream
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                using (Ocr ocr = new Ocr())
                {
                    string resourceFolder = @"C:\Aquaforest\OCRSDK\bin";
                    string currentEnvironmentVariables = Environment.GetEnvironmentVariable("PATH");
                    if (!currentEnvironmentVariables.Contains(resourceFolder))
                    {
                        Environment.SetEnvironmentVariable(
                            "PATH", currentEnvironmentVariables + ";" + resourceFolder);
                    }

                    // Set OCR options
                    ocr.ResourceFolder = resourceFolder;
                    ocr.Language = SupportedLanguages.English;
                    ocr.EnablePdfOutput = true;
                    ocr.EnableConsoleOutput = true;

                    // Set PreProcessor options
                    PreProcessor preProcessor = new PreProcessor();
                    preProcessor.Deskew = true;
                    preProcessor.Autorotate = false;

                    using (Stream sourceStream = File.Open(
                             Path.GetFullPath(@"..\..\..\..\documents\source\PNGSample.png"),
                             FileMode.Open, FileAccess.Read, FileShare.Read))
                    {
                        // Read source image stream
                        ocr.ReadBMPSource(sourceStream);

                        // OCR the image
                        if (ocr.Recognize(preProcessor))
                        {
                            // Save output as searchable PDF stream
                            Stream outputStream;
                            ocr.SavePDFOutput(out outputStream);

                            // OR
                            // Save output as searchable PDF
                            // ocr.SavePDFOutput(@"..\..\..\..\documents\output\output.pdf", true);
                        }
                    }
                    ocr.DeleteTemporaryFiles();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
        }
    }
}
```

1.19 Generate PDF/A compliant files

1.19.1 Requirement

Convert a TIFF file to a searchable PDF that is compliant with PDF/A-1b.

1.19.2 Solution

```
using System;
using System.IO;
using Aquaforest.OCR.Api;

namespace ocr
{
    class PDFACompliance
    {
        static void Main(string[] args)
        {
            try
            {
                using (Ocr ocr = new Ocr())
                {
                    string resourceFolder = @"C:\Aquaforest\OCRSDK\bin";

                    string currentEnvironmentVariables = Environment.GetEnvironmentVariable("PATH");
                    if (!currentEnvironmentVariables.Contains(resourceFolder))
                    {
                        Environment.SetEnvironmentVariable(
                            "PATH", currentEnvironmentVariables + ";" + resourceFolder);
                    }

                    // Set OCR options
                    ocr.ResourceFolder = resourceFolder;
                    ocr.EnableConsoleOutput = true;
                    ocr.Language = SupportedLanguages.English;
                    ocr.EnablePDFAOutput = true;

                    // Set PreProcessor options
                    PreProcessor preProcessor = new PreProcessor();
                    preProcessor.Deskew = true;
                    preProcessor.Autorotate = false;

                    // Read source TIFF file
                    ocr.ReadTIFFSource(Path.GetFullPath(@"..\..\..\..\documents\source\sample.tif"));

                    // Perform OCR recognition
                    if (ocr.Recognize(preProcessor))
                    {
                        // Save output as searchable PDF
                        ocr.SavePDFAOutput(Path.GetFullPath(
                            @"..\..\..\..\documents\output\samplePDFA.pdf"), true,
                            PDFAVersion.PDF_A2b);
                    }

                    ocr.DeleteTemporaryFiles();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
        }
    }
}
```

1.19.3 Comments

You can only convert image documents to PDF/A compliant files because the OCR SDK processes PDF files natively and therefore does not generate a new file.

1.20 ASP.NET conversion

1.20.1 Requirement

Perform OCR from a file posted in an ASP.NET application.

1.20.2 Solution

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
    Inherits="Web_Application._Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
        <head runat="server">
                <title>Untitled Page</title>
        </head>
        <body>
                <form id="form1" runat="server">
                        <div>

        <asp:Label ID="Label1" runat="server" Text="BMP or TIFF : "></asp:Label>
                                <asp:FileUpload ID="FileUploadSource" runat="server" />

        <asp:Label ID="Label2" runat="server" Text="Convert To : "></asp:Label>

        <asp:DropDownList ID="DropDownListDestinationType" runat="server" Width="224px"
                                Height="26px">

        <asp:ListItem Selected="True" Value="0">Portable Document Format (.pdf)</asp:ListItem>
                                        <asp:ListItem Value="1">Text (.txt)</asp:ListItem>
                                        <asp:ListItem Value="2">Rich Text (.rtf)</asp:ListItem>
                                </asp:DropDownList>
                        </div>
                        <div>

        <asp:Table ID="Table1" runat="server" Height="150px" Width="178px">
                                        <asp:TableRow>
                                                <asp:TableCell>
                                                        <Label>Despeckle</Label>
                                                </asp:TableCell>
                                                <asp:TableCell>

        <asp:DropDownList ID="DropDownListDespeckle" runat="server">

        <asp:ListItem Selected="True" Value="0">0</asp:ListItem>

        <asp:ListItem Value="1">1</asp:ListItem>

        <asp:ListItem Value="2">2</asp:ListItem>

        <asp:ListItem Value="3">3</asp:ListItem>

        <asp:ListItem Value="4">4</asp:ListItem>

        <asp:ListItem Value="5">5</asp:ListItem>

        <asp:ListItem Value="6">6</asp:ListItem>

        <asp:ListItem Value="7">7</asp:ListItem>

        <asp:ListItem Value="8">8</asp:ListItem>

        <asp:ListItem Value="9">9</asp:ListItem>
                                                                </asp:DropDownList>
                                                </asp:TableCell>
```

```
                                        </asp:TableRow>
                                        <asp:TableRow>
                                                <asp:TableCell>
                                                        <Label>Autorotate</Label>
                                                </asp:TableCell>
                                                <asp:TableCell>

        <asp:CheckBox ID="CheckBoxAutorotate" runat="server">
                                                        </asp:CheckBox>
                                                </asp:TableCell>
                                        </asp:TableRow>
                                        <asp:TableRow>
                                                <asp:TableCell>
                                                        <Label>Pictures</Label>
                                                </asp:TableCell>
                                                <asp:TableCell>

        <asp:CheckBox ID="CheckBoxPictures" runat="server">
                                                        </asp:CheckBox>
                                                </asp:TableCell>
                                        </asp:TableRow>
                                        <asp:TableRow>
                                                <asp:TableCell>
                                                        <Label>Tables</Label>
                                                </asp:TableCell>
                                                <asp:TableCell>

        <asp:CheckBox ID="CheckBoxTables" runat="server">
                                                        </asp:CheckBox>
                                                </asp:TableCell>
                                        </asp:TableRow>
                                        <asp:TableRow>
                                                <asp:TableCell>
                                                        <Label>Deskew</Label>
                                                </asp:TableCell>
                                                <asp:TableCell>

        <asp:CheckBox ID="CheckBoxDeskew" runat="server">
                                                        </asp:CheckBox>
                                                </asp:TableCell>
                                        </asp:TableRow>
                                        <asp:TableRow>
                                                <asp:TableCell>
                                                        <Label>Box size</Label>
                                                </asp:TableCell>
                                                <asp:TableCell>

        <asp:TextBox ID="TextBoxBoxSize" runat="server"></asp:TextBox>
                                                </asp:TableCell>
                                        </asp:TableRow>
                                        <asp:TableRow>
                                                <asp:TableCell>
                                                        <Label>Binarization threshold</Label>
                                                </asp:TableCell>
                                                <asp:TableCell>

        <asp:TextBox ID="TextBoxBinarize" runat="server"></asp:TextBox>
                                                </asp:TableCell>
                                        </asp:TableRow>
                                        <asp:TableRow>
                                                <asp:TableCell>
                                                        <Label>Language</Label>
                                                </asp:TableCell>
                                                <asp:TableCell>

        <asp:DropDownList ID="DropDownListLanguage" runat="server">
                                                        </asp:DropDownList>
                                                </asp:TableCell>
```

```
                                        </asp:TableRow>
                                    </asp:Table>
                                </div>
                                <br />
                                <asp:Button ID="ButtonConvert" runat="server" Text="Convert"
                                onclick="ButtonConvert_Click" />
                                <br />
                                <br />
                        </form>
                        <div>
                        </div>
                </body>
</html>



using System;
using System.IO;
using Aquaforest.OCR.Api;

namespace Web_Application
{
    public partial class _Default : System.Web.UI.Page
    {
        private Ocr ocr;
        private PreProcessor preProcessor;

        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                foreach (string language in Enum.GetNames(typeof(SupportedLanguages)))
                {
                    DropDownListLanguage.Items.Add(language);
                }

                int index = 0;
                foreach (int value in Enum.GetValues(typeof(SupportedLanguages)))
                {
                    DropDownListLanguage.Items[index].Value = value.ToString();
                    index++;
                }
            }
        }

        protected void ButtonConvert_Click(object sender, EventArgs e)
        {
            string tempLogFile = "";
            StreamWriter sw = null;

            try
            {
                tempLogFile = Path.GetTempFileName();
                sw = new StreamWriter(tempLogFile);
                sw.AutoFlush = true;
                Console.SetOut(sw);

                if (FileUploadSource.HasFile)
                {
                    string fileName = Path.GetFileName(FileUploadSource.FileName);
                    string extension = Path.GetExtension(FileUploadSource.FileName);
                    string sourceFile = Path.Combine(Path.GetTempPath(), fileName);
                    FileUploadSource.SaveAs(sourceFile);

                    SetupOcr();

                    switch (extension.ToLower())
                    {
```

```
                    case ".jpg":
                    case ".jpeg":
                    case ".png":
                    case ".bmp":
                        ocr.ReadBMPSource(sourceFile);
                        break;

                    case ".tif":
                    case ".tiff":
                        ocr.ReadTIFFSource(sourceFile);
                        break;

                    case ".pdf":
                        ocr.ReadPDFSource(sourceFile);
                        break;
                }

                ocr.EnablePdfOutput = true;
                ocr.EnableRtfOutput = true;
                ocr.EnableTextOutput = true;

                if (ocr.Recognize(preProcessor))
                {
                    SaveOcrOutput();
                }
                else
                {
                    Response.Write("ERROR<br />");
                    using (StreamReader sr = new StreamReader(tempLogFile))
                    {
                        Response.Write(sr.ReadToEnd().Replace(Environment.NewLine, "<br />"));
                    }
                    Response.End();
                }
            }
        }
        catch (Exception exception)
        {
            Response.Write(exception.Message.Replace(Environment.NewLine, "<br />"));
            Response.End();
        }
        finally
        {
            if (ocr != null) ocr.Dispose();
            if (sw != null) sw.Dispose();
            if (File.Exists(tempLogFile)) File.Delete(tempLogFile);
        }

        Response.End();
    }

    private void SaveOcrOutput()
    {
        string destinationFile = Path.GetTempFileName() + "output.";
        int outputType = Convert.ToInt32(DropDownListDestinationType.SelectedItem.Value);
        switch (outputType)
        {
            case 0:
                Response.Write("Saving PDF<BR>");
                destinationFile += "pdf";
                ocr.SavePDFOutput(destinationFile, true);
                break;
            case 1:
                Response.Write("Saving TXT<BR>");
                destinationFile += "txt";
                ocr.SaveTextOutput(destinationFile, true);
                break;
            case 2:
```

```
                Response.Write("Saving RTF<BR>");
                destinationFile += "rtf";
                ocr.SaveRTFOutput(destinationFile, true);
                break;
            //case 3:
            //    break;
        }
        ocr.DeleteTemporaryFiles();
        DownloadFile(destinationFile, true);
    }

    private void SetupOcr()
    {
        ocr = new Ocr();

        //IMPORTANT: You need a license to run this ASP.NET sample.
        ocr.License = "";

        ocr.EnableDebugOutput = 16383;
        ocr.Language = (SupportedLanguages)Convert.ToInt32(
                       DropDownListLanguage.SelectedItem.Value);

        // Enter full path of the resource folder because web applications
        // run from the context of the web server
        string resourceFolder = @"C:\Aquaforest\OCRSDK\bin";

        string currentEnvironmentVariables = Environment.GetEnvironmentVariable("PATH");
        if (!currentEnvironmentVariables.Contains(resourceFolder))
        {
            Environment.SetEnvironmentVariable(
                "PATH", currentEnvironmentVariables + ";" + resourceFolder);
        }
        ocr.ResourceFolder = resourceFolder;

        preProcessor = new PreProcessor();
        preProcessor.Autorotate = CheckBoxAutorotate.Checked;
        if (!string.IsNullOrEmpty(TextBoxBinarize.Text))
        {
            preProcessor.Binarize = Convert.ToInt32(TextBoxBinarize.Text);
        }

        if (!string.IsNullOrEmpty(TextBoxBoxSize.Text))
        {
            preProcessor.BoxSize = Convert.ToInt32(TextBoxBoxSize.Text);
        }

        preProcessor.Deskew = CheckBoxDeskew.Checked;
        preProcessor.Despeckle = DropDownListDespeckle.SelectedIndex;
        preProcessor.NoPictures = !CheckBoxPictures.Checked;
        preProcessor.Tables = CheckBoxTables.Checked;
    }

    private void DownloadFile(string fileName, bool forceDownload)
    {
        string path = fileName;
        string name = Path.GetFileName(path);
        string ext = Path.GetExtension(path);
        string type = "";

        if (ext != null)
        {
            switch (ext.ToLower())
            {
                case ".htm":
                case ".html":
                    type = "text/HTML";
                    break;
```

```
                    case ".txt":
                        type = "text/plain";
                        break;

                    case ".doc":
                    case ".rtf":
                        type = "Application/msword";
                        break;

                    case ".pdf":
                        type = "Application/pdf";
                        break;
                }
            }

            if (forceDownload)
            {
                Response.Clear();
                Response.AppendHeader("content-disposition", "attachment; filename=" + name);
            }

            if (type != "")
            {
                Response.ContentType = type;
            }

            Response.WriteFile(path);

        }
    }
}
```

1.20.3 Comments

This sample is a simple ASP.NET form that submits an image file for conversion and if the conversion succeeds, the OCRed file is downloaded.

1.21 Handling password-protected PDFs

1.21.1 Requirement

Password-protected PDFs need special treatment to be successfully processed.

1.21.2 Solution

```
using System;
using System.IO;
using Aquaforest.OCR.Api;

namespace PasswordProtectedPDF
{
    class PasswordProtectedPDF
    {
        static void Main(string[] args)
        {
            try
            {
                using (Ocr ocr = new Ocr())
                {
                    string resourceFolder = @"C:\Aquaforest\OCRSDK\bin";

                    string currentEnvironmentVariables = Environment.GetEnvironmentVariable("PATH");
                    if (!currentEnvironmentVariables.Contains(resourceFolder))
                    {
                        Environment.SetEnvironmentVariable(
                            "PATH", currentEnvironmentVariables + ";" + resourceFolder);
                    }

                    // Set OCR options
                    ocr.ResourceFolder = resourceFolder;
                    ocr.EnableConsoleOutput = true;
                    ocr.Language = SupportedLanguages.English;
                    ocr.EnablePdfOutput = true;

                    // Set PreProcessor options
                    PreProcessor preProcessor = new PreProcessor();
                    preProcessor.Deskew = true;
                    preProcessor.Autorotate = false;

                    string source = @"..\..\..\..\documents\source\secured.pdf";

                    // Read source PDF file (password = ddd)
                    ocr.ReadPDFSource(Path.GetFullPath(source), "ddd");

                    // Perform OCR recognition
                    if (ocr.Recognize(preProcessor))
                    {
                        // Save output as searchable PDF
                        ocr.SavePDFOutput(Path.GetFullPath(
                            @"..\..\..\..\documents\output\secured_searchable.pdf"), true);
                    }

                    ocr.DeleteTemporaryFiles();
                }
            }

            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
        }
    }
}
```

1.22 Error handling and debugging

1.22.1 Requirement

To ensure the creation of robust applications, appropriate error handling should be in place. In the event of any issues, the debugging facilities built within the Aquaforest OCR can be very useful.

1.22.2 Solution

There are two options regarding error handling using the API:

1. Using the default settings various exceptions can be thrown by the `Ocr` object so these should be trapped within the calling code. Below are the different `DebugLevels` that can be set to `ocr.EnableDebugOutput`:

Member name	Value	Description
NONE	0	
AUTOROTATE_INFO	1	Autorotate information
AUTOROTATE_WORDS	2	Words used in autorotate decision
DEAD_PROCESS	4	Notification of Dead Child Process
ERRORS	8	Handled errors
EXCEPTIONS	16	Stack Trace and exception messages
IMAGE_PREPROCESSING	32	Image pre-processing
LEAVE_TEMP_FILES_IN_PLACE	64	Do not delete temporary files
MESSAGES	128	Interprocess messages
PDF_IMAGE_EXTRACTION	256	PDF image extraction
PIPE_EVENTS	512	Pipe connections/disconnections events
PROGRESS_UPDATES	1024	General progress messages
TEMP_FILE_OPERATION	2048	Temp File Operations
TIMEOUTS	4096	Timeout messages
LOG_TO_FILE	8192	Log output to file
STOP_ON_ERROR	16384	Exit application on certain errors

2. Alternatively `HandleExceptionsInternally` can be set to true with the result that method calls will return false on error but throw no exceptions. The calling code can obtain the last exception from the `LastException` property if details of the failure are required.

1.23 Use pre-processing settings

1.23.1 Requirement

Certain image type can benefit from pre-processing options such as using `Deskew`, `Despeckle`, `Morph` and `Binarize` options within the SDK. When enabled these settings are applied to the image before it is passed to the OCR engine.

1.23.2 Solution

```
_preProcessor.Binarize = -1;
```

We recommended the above to be used with color image documents. When enabled, color images will be converted to bitonal for the OCR process.

```
_preProcessor.Morph = "d.2.2";
```

This can be used on faint images, when enabled dilation will be applied to all black pixels.

```
_preProcessor.Morph = "e.2.2";
```

This can be useful on heavy prints, when enabled erosion will be applied to all black pixel areas.

```
_preProcessor.Morph = "c.2.2";
```

If enabled the process will perform a 2×2 dilation followed by a 2×2 erosion with the result that holes and gaps in the characters are filled.

```
_preProcessor.Deskew = true;
```

This option can improve OCR results by straightening crooked pages.

```
_preProcessor.Despeckle = 2;
```

This pre-processing option removes isolated “dots” within the image which can cause recognition problems, and makes the result image “cleaner”.

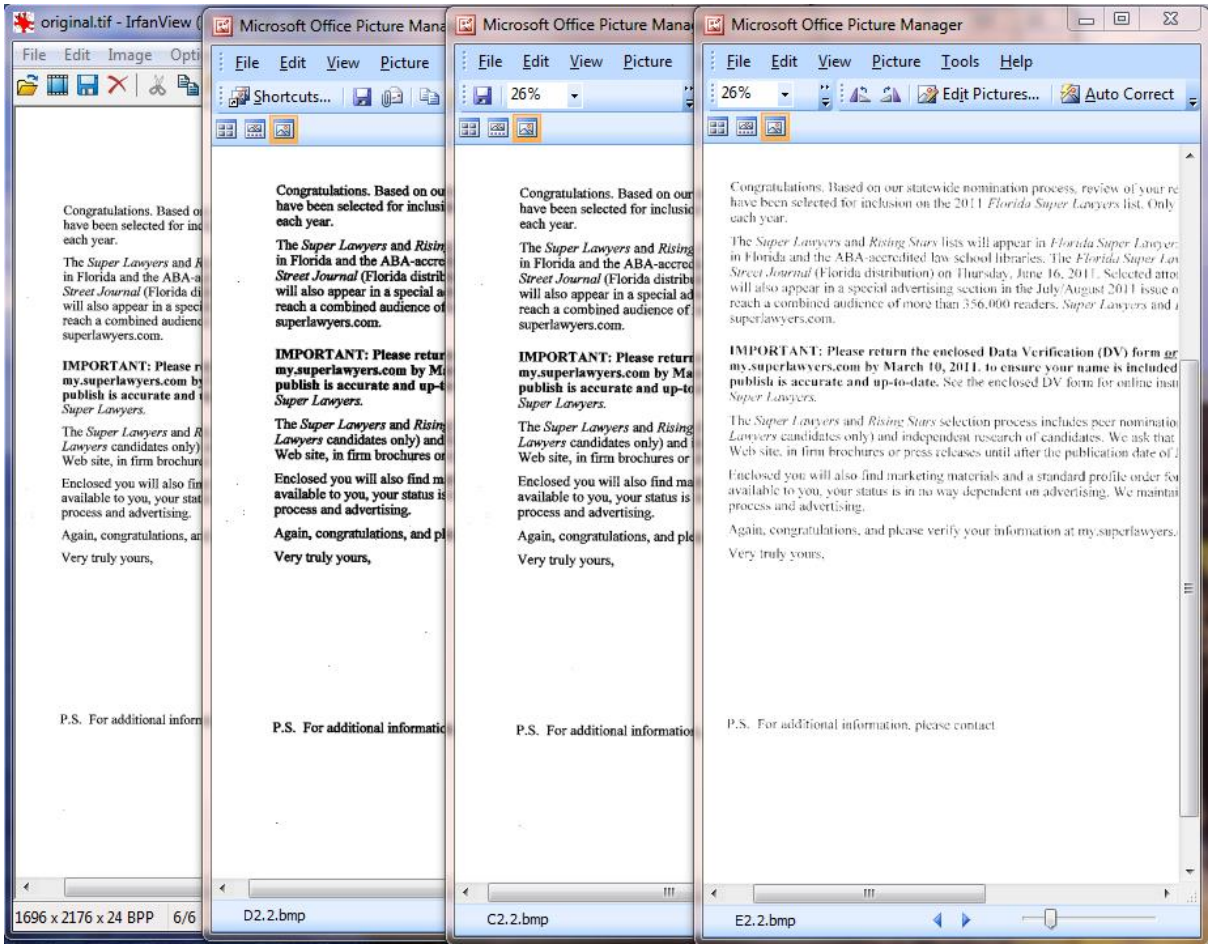


Image which shows original document followed by the intermediary bmp files showing the effects of dilation, closing and erosion.

1.24 Using advanced pre-processing (Optimized OCR)

1.24.1 Requirement

Use multiple pre-processing sets against a single page, allowing the system to choose the best OCR results.

1.24.2 Solution

Enable the following property on the OCR object:

```
_ocr.AdvancedPreProcessing = true;
```

1.24.3 Comments

When the `AdvancePreProcessing` property on the OCR object is set to false the OCR and image processing engines will use the settings in the `ImagePreProcessingDefaults` section of the file Properties.xml modified by any properties set on the OCR and `PreProcessing` objects.

Setting the `AdvancePreProcessing` property to true will enable the use of these default settings first (without modification by the properties set on the OCR and `PreProcessing` objects) followed by the same defaults modified by the values in the `ImagePreProcessing` sections from ID="1" to ID="n" where n is the last consecutive set defined in Properties.xml.

Using heuristics and dictionary lookup the quality of the OCR output is then compared in order to determine the optimum set to output. In this way it is possible to define different sets of OCR and pre-processing conditions that are suited to different types of source documents. This approach can also improve the handling of documents that contain different types of pages, e.g. scanned at different qualities, containing different languages, containing standard and dot matrix prints, etc.

2 Extended OCR Engine

The table below shows the prerequisites needed for building applications using the Extended OCR engine:

Application Platform	VC++ Redistributable	Minimum .NET Framework Version	Minimum Visual Studio Version
x86	VC ++ 2017 x86	.NET Framework 4.5.2	Visual Studio 2012
x64	VC ++ 2017 x64		
Any CPU	VC ++ 2017 x86 VC ++ 2017 x64		

Note: If you are using Visual Studio 2012 or 2013 to build your application(s), you need to ensure that you also have the .NET Framework 4.5.2 Multi-targeting pack installed so as to be able to build applications that target the .NET Framework 4.5.2.

2.1 Convert TIFF to searchable PDF

2.1.1 Requirement

Convert a multi-page TIFF file to searchable PDF document.

2.1.2 Solution

```
using Aquaforest.ExtendedOCR.Api;
using System;
using System.IO;

namespace ConvertTIFFToSearchablePDF
{
    internal class Program
    {
        private static void Main(string[] args)
        {
            try
            {
                string resourceFolder = @"C:\Aquaforest\OCRSDK\xbin\resources";

                using (Ocr ocr = new Ocr(resourceFolder))
                {
                    // Set OCR options
                    ocr.EnableConsoleOutput = true;
                    ocr.EnableDebugOutput = false;
                    ocr.Language = SupportedLanguages.English;
                    ocr.EnablePdfOutput = true;

                    // Set PreProcessor options
                    PreProcessor preProcessor = new PreProcessor();
                    preProcessor.Autorotate = true;
                    preProcessor.Deskew = true;
                    preProcessor.KeepOriginalImage = true;

                    // Read source TIFF file
                    ocr.ReadTIFFSource(
                        Path.GetFullPath(@"..\..\..\..\documents\source\sample.tif"));

                    // Perform OCR recognition
                    if (ocr.Recognize(preProcessor))
                    {
                        // Save output as searchable PDF
                        ocr.SavePDFOutput(
                            Path.GetFullPath(@"..\..\..\..\documents\output\sample.pdf"), true);
                    }

                    ocr.DeleteTemporaryFiles();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
        }
    }
}
```

2.2 Convert BMP, JPG, PNG to Searchable PDF

2.2.1 Requirement

Generate a fully searchable PDF from a BMP, PNG or JPG source file.

2.2.2 Solution

```
using Aquaforest.ExtendedOCR.Api;
using System;
using System.IO;

namespace ConvertBMP_JPEG_PNGToSearchablePDF
{
    internal class Program
    {
        private static void Main(string[] args)
        {
            try
            {
                string resourceFolder = @"C:\Aquaforest\OCRSDK\xbin\resources";

                using (Ocr ocr = new Ocr(resourceFolder))
                {
                    // Set OCR options
                    ocr.Language = SupportedLanguages.English;
                    ocr.EnablePdfOutput = true;
                    ocr.EnableConsoleOutput = true;

                    // Set PreProcessor options
                    PreProcessor preProcessor = new PreProcessor();
                    preProcessor.Deskew = true;
                    preProcessor.Autorotate = false;
                    preProcessor.KeepOriginalImage = true;

                    // Read source BMP file
                    ocr.ReadBMPSource(@"..\..\..\..\documents\source\BMPSample.bmp");

                    // Read source PNG file
                    //ocr.ReadBMPSource(@"..\..\..\..\documents\source\PNGSample.png");

                    // Read source JPEG file
                    //ocr.ReadJPEGSource(@"..\..\..\..\documents\source\JPEGSample.jpg");

                    //OCR the image
                    if (ocr.Recognize(preProcessor))
                    {
                        // Save output as searchable PDF
                        ocr.SavePDFOutput(@"..\..\..\..\documents\output\output.pdf", true);
                    }

                    ocr.DeleteTemporaryFiles();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
        }
    }
}
```

2.3 OCR an image-only PDF

2.3.1 Requirement

Generate a fully searchable PDF from an image-PDF file.

2.3.2 Solution

```
using Aquaforest.ExtendedOCR.Api;
using System;

namespace OCRImagePDF
{
    internal class Program
    {
        private static void Main(string[] args)
        {
            try
            {
                using (Ocr ocr = new Ocr(@"C:\Aquaforest\OCRSDK\xbin\resources"))
                {
                    // Set OCR options
                    ocr.EnableConsoleOutput = true;
                    ocr.EnableDebugOutput = true;
                    ocr.Language = SupportedLanguages.English;
                    ocr.EnablePdfOutput = true;

                    // Set PreProcessor options
                    PreProcessor preProcessor = new PreProcessor();
                    preProcessor.Deskew = true;

                    // Read source PDF file
                    ocr.ReadPDFSource(@"..\..\..\..\documents\source\image_pdf.pdf");

                    // Perform OCR recognition
                    if (ocr.Recognize(preProcessor))
                    {
                        // Save output as searchable PDF
                        ocr.SavePDFOutput(@"..\..\..\..\documents\output\image_pdf.pdf", true);
                    }

                    ocr.DeleteTemporaryFiles();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
        }
    }
}
```

2.4 OCR an image-only PDF stream

2.4.1 Requirement

Perform OCR recognition on an image-only PDF that is held in a stream rather than in a file.

2.4.2 Solution

```
using Aquaforest.ExtendedOCR.Api;
using System.IO;
using System;

namespace OCRImagePDFFromStream
{
    internal class Program
    {
        private static void Main(string[] args)
        {
            try
            {
                using (Ocr ocr = new Ocr(@"C:\Aquaforest\OCRSDK\xbin\resources"))
                {
                    // Set OCR options
                    ocr.EnableConsoleOutput = true;
                    ocr.EnableDebugOutput = true;
                    ocr.Language = SupportedLanguages.English;
                    ocr.EnablePdfOutput = true;

                    // Set PreProcessor options
                    PreProcessor preProcessor = new PreProcessor();
                    preProcessor.Deskew = true;

                    using (Stream sourceStream = File.Open(
                              Path.GetFullPath(@"..\..\..\..\documents\source\image_pdf.pdf"),
                              FileMode.Open, FileAccess.Read, FileShare.Read))
                    {
                        // Read source PDF stream
                        ocr.ReadPDFSource(sourceStream);

                        // Perform OCR recognition
                        if (ocr.Recognize(preProcessor))
                        {
                            // Save output as searchable PDF stream
                            Stream outputStream;
                            ocr.SavePDFOutput(out outputStream);

                            // OR
                            // Save output as searchable PDF
                            // ocr.SavePDFOutput(@"..\..\..\..\documents\output\image_pdf.pdf", true);
                        }
                    }

                    ocr.DeleteTemporaryFiles();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
        }
    }
}
```

2.5 OCR an "in-memory" Image

2.5.1 Requirement

Perform OCR recognition on an image that is held in memory rather than in a file.

2.5.2 Solution

```
using Aquaforest.ExtendedOCR.Api;
using System;
using System.Drawing;

namespace OCRImagesInMemory
{
    internal class OCRImagesInMemory
    {
        private static void Main(string[] args)
        {
            try
            {
                using (Ocr ocr = new Ocr(@"C:\Aquaforest\OCRSDK\xbin\resources"))
                {
                    // Set OCR options
                    ocr.Language = SupportedLanguages.English;
                    ocr.EnablePdfOutput = true;
                    ocr.EnableConsoleOutput = true;

                    // Set PreProcessor options
                    PreProcessor preProcessor = new PreProcessor();
                    preProcessor.Deskew = true;
                    preProcessor.Autorotate = false;

                    Image image = Image.FromFile(@"..\..\..\..\documents\source\sample.tif");

                    ocr.ReadImageSource(image);

                    // Perform OCR recognition
                    if (ocr.Recognize(preProcessor))
                    {
                        // Save output as searchable PDF
                        ocr.SavePDFOutput(@"..\..\..\..\documents\output\inMemorySample.pdf", true);
                    }

                    ocr.DeleteTemporaryFiles();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
        }
    }
}
```

2.6 OCR an image stream

2.6.1 Requirement

Perform OCR recognition on an image that is held in a stream rather than in a file.

2.6.2 Solution

```
using Aquaforest.ExtendedOCR.Api;
using System.IO;
using System;

namespace OCRImageFromStream
{
    internal class Program
    {
        private static void Main(string[] args)
        {
            try
            {
                using (Ocr ocr = new Ocr(@"C:\Aquaforest\OCRSDK\xbin\resources"))
                {
                    // Set OCR options
                    ocr.Language = SupportedLanguages.English;
                    ocr.EnablePdfOutput = true;
                    ocr.EnableConsoleOutput = true;

                    // Set PreProcessor options
                    PreProcessor preProcessor = new PreProcessor();
                    preProcessor.Deskew = true;
                    preProcessor.Autorotate = false;

                    using (Stream sourceStream = File.Open(
                              Path.GetFullPath(@"..\..\..\..\documents\source\PNGSample.png"),
                              FileMode.Open, FileAccess.Read, FileShare.Read))
                    {
                        // Read source image stream
                        ocr.ReadBMPSource(sourceStream);

                        // Perform OCR recognition
                        if (ocr.Recognize(preProcessor))
                        {
                            // Save output as searchable PDF stream
                            Stream outputStream;
                            ocr.SavePDFOutput(out outputStream);

                            // OR
                            // Save output as searchable PDF
                            // ocr.SavePDFOutput(@"..\..\..\..\documents\output\image_pdf.pdf", true);
                        }
                    }

                    ocr.DeleteTemporaryFiles();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
        }
    }
}
```

2.7 Get text from TIFFs or image PDFs

2.7.1 Requirement

OCR and get text from each page of a TIFF or image-only PDF file.

2.7.2 Solution

```
using Aquaforest.ExtendedOCR.Api;
using Aquaforest.ExtendedOCR.Shared;
using System;
using System.IO;

namespace GetTextFromPage
{
    internal class Program
    {
        private static Ocr ocr;

        private static void Main(string[] args)
        {
            try
            {
                // Set OCR options
                ocr = new Ocr(@"C:\Aquaforest\OCRSDK\xbin\resources");
                ocr.EnableConsoleOutput = true;
                ocr.Language = SupportedLanguages.English;
                ocr.StatusUpdate += OcrStatusUpdate;

                // Set PreProcessor options
                PreProcessor preProcessor = new PreProcessor();
                preProcessor.Deskew = true;
                preProcessor.Autorotate = false;

                // Read source TIFF file
                ocr.ReadTIFFSource(Path.GetFullPath(@"..\..\..\..\documents\source\sample.tif"));

                // Read source PDF file
                // ocr.ReadPDFSource(
                //     Path.GetFullPath(@"..\..\..\..\documents\source\image_pdf.pdf"));

                // Perform OCR recognition
                ocr.Recognize(preProcessor);

                ocr.DeleteTemporaryFiles();
            }
            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
            finally
            {
                if (ocr != null) ocr.Dispose();
            }
        }

        private static void OcrStatusUpdate(object sender,
            StatusUpdateEventArgs pageCompletedEventArgs)
        {
            Console.WriteLine("Page {0}:", pageCompletedEventArgs.PageNumber);

            if (pageCompletedEventArgs.TextAvailable)
            {
                Console.WriteLine(ocr.ReadPageString(pageCompletedEventArgs.PageNumber));
            }
            else
            {
```

```
                Console.WriteLine("No text found");
            }
        }
    }
}
```

2.7.3 Comments

The above code uses the `StatusUpdate` event of the OCR SDK to find out if a page has been processed. In order to use the StatusUpdate event, a reference to `Aquaforest.ExtendedOCR.Shared` needs to be added.

The `ReadPageString` method is used to return all the OCRed text from the specified page number as `String`.

2.8 Zonal OCR

2.8.1 Requirement

Extract text from a specific area of an image.

2.8.2 Solution

```
using Aquaforest.ExtendedOCR.Api;
using System;
using System.Drawing;
using System.IO;

namespace ZonalSimple
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                using (Ocr ocr = new Ocr(@"C:\Aquaforest\OCRSDK\xbin\resources"))
                {
                    ocr.Language = SupportedLanguages.English;
                    ocr.EnablePdfOutput = true;
                    ocr.EnableConsoleOutput = true;
                    ocr.License = "";

                    PreProcessor preProcessor = new PreProcessor();
                    preProcessor.Deskew = true;
                    preProcessor.Autorotate = false;

                    //Variables for the coordinates
                    int x, y, height, width;

                    //United State Patent block.
                    //Assign values for the coordinates of the area you wish to convert
                    x = 320; y = 210; height = 210; width = 900;

                    //Create a new rectangle object thta represents the zone you are interested in
                    Rectangle region = new Rectangle(x, y, width, height);

                    //Read the image for processing
                    ocr.ReadTIFFSource(
                        Path.GetFullPath(@"..\..\..\..\documents\source\sample.tif"));

                    if (ocr.Recognize(preProcessor))
                    {
                        Console.WriteLine("Recognized : ");

                        //Use the page number and the region to get the text in that specific zone.
                        Console.WriteLine(ocr.ReadPageString(1, region));
                    }

                    ocr.DeleteTemporaryFiles();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }

        }
    }
}
```

2.8.3 Comments

There are two methods that can be used to achieve zonal OCR – either using the built-in methods to identify text within particular coordinates or by explicitly cropping the image to the area of interest.

The above code shows how to get text from a particular area of a file using the built in method. Firstly, a rectangle representing the zone that needs to be extracted is created. This rectangle along with the page number is passed to the `ocr.ReadPageString` method.

2.9 Multithreading

2.9.1 Requirement

Process multiple documents in parallel

2.9.2 Solution

```
using Aquaforest.ExtendedOCR.Api;
using System;
using System.IO;
using System.Threading.Tasks;

namespace MultiThreading
{
    class Program
    {
        private static string outputFolder = @"..\..\..\..\documents\output";
        private static string inputFolder = @"..\..\..\..\documents\source";
        private static int maxDegreeOfParallelism = 4;
        private static string resourceFolder;
        private static string tempPath;

        static void Main(string[] args)
        {
            tempPath = Path.Combine(Path.GetTempPath(), "MultiThread");
            resourceFolder = @"..\..\..\..\..\xbin\resources";

            Parallel.ForEach(Directory.EnumerateFiles(Path.GetFullPath(inputFolder), "*.tif"),
                 new ParallelOptions { MaxDegreeOfParallelism = maxDegreeOfParallelism }, f =>
            {
                var outputFilePath = Path.Combine(outputFolder,
                     Path.GetFileNameWithoutExtension(f) + ".pdf");

                if (File.Exists(outputFilePath)) return;

                try
                {
                    ProcessDocument(f, outputFilePath);
                }
                catch (Exception e)
                {
                    Console.WriteLine("ProcessDocument: " + e.Message);
                }
            });
        }

        static void ProcessDocument(string source, string output)
        {
            string tempFolder = Path.Combine(tempPath, Guid.NewGuid().ToString());

            try
            {
                using (Ocr ocr = new Ocr(resourceFolder))
                {
                    // Set OCR options
                    ocr.TempFolder = tempFolder;
                    ocr.EnablePdfOutput = true;
                    ocr.Languages = new SupportedLanguages[]
                    {
                        SupportedLanguages.English,
                        SupportedLanguages.French,
                        SupportedLanguages.Spanish,
                        SupportedLanguages.German,
                    };

                    // Set PreProcessor options
```

```
                    PreProcessor preProcessor = new PreProcessor();
                    preProcessor.Deskew = true;

                    // Read source PDF file
                    ocr.ReadTIFFSource(source);

                    // Perform OCR recognition
                    if (ocr.Recognize(preProcessor))
                    {
                        // Save output as searchable PDF
                        ocr.SavePDFOutput(output, true);
                    }
                    else
                    {
                        Console.WriteLine(source + " Failed");
                    }
                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
            finally
            {
                if (Directory.Exists(tempFolder)) Directory.Delete(tempFolder, true);
            }
        }
    }
}
```

2.9.3 Comments

The number of threads (`MaxDegreeOfParallelism`) allowed depends on your license.

2.10 Intelligent High Quality Compression

2.10.1 Requirement

Compress output PDF files using Intelligent High Quality Compression (IHQC).

2.10.2 Solution

```
using Aquaforest.ExtendedOCR.Api;
using System;

namespace IHQC_Compression
{
    class Program
    {
        private static void Main(string[] args)
        {
            try
            {
                using (Ocr ocr = new Ocr(@"..\..\..\..\..\xbin\resources"))
                {
                    // Set OCR options
                    ocr.EnableConsoleOutput = true;
                    ocr.EnableDebugOutput = true;
                    ocr.Language = SupportedLanguages.English;
                    ocr.EnablePdfOutput = true;

                    // Set Intelligent High Quality Compression
                    ocr.IHQCCompression = new IHQCCompression
                    {
                        EnableIHQCCompression = true,
                        IHQCCompressionLevel = IHQCCompressionLevel.Level1,
                        IHQCQualityFactor = IHQCQualityFactor.Medium
                    };

                    // ExtractImageMethod must be set to ConvertToTiff for IHQC compression to work
                    // (only if the document being processed is a PDF file.
                    // This setting is not necessary if the source is an image file)
                    ocr.ExtractImageMethod = ExtractImageMethod.ConvertToTiff;

                    // Set PreProcessor options
                    PreProcessor preProcessor = new PreProcessor();
                    preProcessor.Deskew = true;

                    // Read source PDF file
                    ocr.ReadPDFSource(@"..\..\..\..\documents\source\compression_test.pdf");

                    // Perform OCR recognition
                    if (ocr.Recognize(preProcessor))
                    {
                        // Save output as compressed searchable PDF
                        ocr.SavePDFOutput(@"..\..\..\..\documents\output\compression_test.pdf", true);
                    }

                    ocr.DeleteTemporaryFiles();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
        }
    }
}
```

2.10.3 Comments

Note: You need to have the IHQC license to use compression.

2.11 Determine word coordinates

2.11.1 Requirement

Get the coordinates of each word recognized by the SDK.

2.11.2 Solution

```
using Aquaforest.ExtendedOCR.Api;
using Aquaforest.ExtendedOCR.Shared;
using System;
using System.IO;

namespace DetermineWordCoordinates
{
    internal class Program
    {
        private static Ocr ocr;

        private static void Main(string[] args)
        {
            try
            {
                // Set OCR options
                ocr = new Ocr(@"C:\Aquaforest\OCRSDK\xbin\resources");
                ocr.EnableConsoleOutput = true;
                ocr.EnableDebugOutput = true;
                ocr.Language = SupportedLanguages.English;
                ocr.StatusUpdate += OcrStatusUpdate;

                // Set PreProcessor options
                PreProcessor preProcessor = new PreProcessor();
                preProcessor.Deskew = true;
                preProcessor.Autorotate = false;

                // Read source TIFF file
                ocr.ReadTIFFSource(Path.GetFullPath(@"..\..\..\..\documents\source\sample.tif"));

                // Perform OCR recognition
                ocr.Recognize(preProcessor);

                ocr.DeleteTemporaryFiles();
            }
            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
            finally
            {
                if (ocr != null) ocr.Dispose();
            }
        }

        private static void OcrStatusUpdate(object sender,
            StatusUpdateEventArgs pageCompletedEventArgs)
        {
            Console.WriteLine("Page {0}:", pageCompletedEventArgs.PageNumber);

            Words words = ocr.ReadPageWords(pageCompletedEventArgs.PageNumber);

            foreach (WordData w in words)
            {
                Console.WriteLine("{0} [{1}, {2}]", w.Word, w.Left, w.Bottom);
            }
        }
    }
}
```

2.11.3 Comments

The above code uses the `StatusUpdate` event of the OCR SDK to find out if a page has been processed. After each page is processed, the `OcrStatusUpdate` event handler is invoked. This method then gets all the words in the page as objects of `WordData`. The `WordData` object holds the coordinates information of the words.

In order to use the `StatusUpdate` event, a reference to `Aquaforest.ExtendedOCR.Shared` needs to be added.

2.12 Confidence score

2.12.1 Requirement

Get the OCR confidence score at the page, word and character level.

2.12.2 Solution

```
using Aquaforest.ExtendedOCR.Api;
using Aquaforest.ExtendedOCR.Shared;
using System;
using System.Collections.Generic;
using System.IO;
using System.Text;

namespace ConfidenceScore
{
    class Program
    {
        private static void Main(string[] args)
        {
            try
            {
                string resourceFolder = Path.GetFullPath(@"..\..\..\..\..\xbin\resources");

                using (Ocr ocr = new Ocr(resourceFolder))
                {
                    ocr.License = Helper.LICENSE_KEY;

                    // Set OCR options
                    ocr.EnableConsoleOutput = true;
                    ocr.Language = SupportedLanguages.English;

                    // IMPORTANT: Set 'GetAdvancedOCRData' to 'true' to get confidence scores
                    ocr.GetAdvancedOCRData = true;

                    // OPTION 1
                    ocr.StatusUpdate += OcrPageCompleted;

                    // Set PreProcessor options
                    PreProcessor preProcessor = new PreProcessor
                    {
                        Deskew = true,
                    };

                    // Read source TIFF file
                    ocr.ReadTIFFSource(Path.GetFullPath(@"..\..\..\..\documents\source\sample.tif"));

                    // Perform OCR recognition
                    if (ocr.Recognize(preProcessor))
                    {
                        // OPTION 2
                        // DocumentCompleted(ocr);
                    }

                    ocr.DeleteTemporaryFiles();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
        }

        // OPTION 1:
        // Use this to get the confidence score while the document is still processing
        // (after each page has processed)
        private static void OcrPageCompleted(object sender, StatusUpdateEventArgs pageInfo)
        {
            Console.WriteLine("Page {0} processed.", pageInfo.PageNumber);
```

```
            if (pageInfo.TextAvailable)
            {
                // retrieve page confidence score
                Console.WriteLine("Page {0} confidence score: {1}",
                    pageInfo.PageNumber, pageInfo.ConfidenceScore);

                // get confidence score for each word and blob (character) in the page
                GetWordAndBlobConfidenceScore(pageInfo.PageNumber, pageInfo.PageLines);
            }
        }

        // OPTION 2:
        // Use this to get the confidence score after the whole document has processed.
        private static void DocumentCompleted(Ocr ocr)
        {
            for (int pageNumber = 1; pageNumber <= ocr.NumberPages; pageNumber++)
            {
                // retrieve page confidence score
                var pageConfidenceScore = ocr.GetPageConfidence(pageNumber);
                Console.WriteLine("Page {0} confidence score: {1}", pageNumber, pageConfidenceScore);

                var pageLines = ocr.ReadPageLines(pageNumber);

                // get confidence score for each word and blob (character) in the page
                GetWordAndBlobConfidenceScore(pageNumber, pageLines);
            }
        }

        private static void GetWordAndBlobConfidenceScore(int pageNumber, List<LineData> pageLines)
        {
            if (pageLines == null || pageLines.Count == 0) return;

            var sb = new StringBuilder();

            foreach (var line in pageLines)
            {
                foreach (var word in line.Words)
                {
                    // retrieve confidence score of each word in the page
                    sb.AppendLine(word.Word + "\t\t" + word.ConfidenceScore);

                    foreach (var blob in word.CharacterList)
                    {
                        // retrieve confidence score for each blob (usually character) in the word
                        sb.AppendLine(blob.Character + "\t\t"
                            + blob.AdvancedCharacterData.ConfidenceScore);
                    }

                    sb.AppendLine();
                }
            }

            if (sb.Length > 0)
            {
                string filePath = Path.GetFullPath(@"..\..\..\..\documents\output\"
                                    + pageNumber + ".txt");
                Console.WriteLine("Words and blobs confidence score for page {0}:", pageNumber);
                Console.WriteLine(filePath);
                Console.WriteLine();

                File.WriteAllText(filePath, sb.ToString());
            }
        }
    }
}
```

2.12.3 Comments

The above code shows 2 ways of retrieving the confidence score.

The first option (OPTION 1 in the code) is to use the `StatusUpdate` event to get the confidence score of a page immediately after it has processed. In order to use the `StatusUpdate` event, a reference to `Aquaforest.ExtendedOCR.Shared` needs to be added.

The second option (OPTION 2 in the code) is to get the confidence score after the whole document has processed by looping through each page and getting the page information.

2.13 Multiple Languages

2.13.1 Requirement

OCR a document that has text in different languages per page.

2.13.2 Solution

```
using Aquaforest.ExtendedOCR.Api;
using System;
using System.IO;

namespace MultipleLanguages
{
    internal class Program
    {
        private static void Main(string[] args)
        {
            try
            {
                string resourceFolder = @"C:\Aquaforest\OCRSDK\xbin\resources";

                using (Ocr ocr = new Ocr(resourceFolder))
                {
                    // Set OCR options
                    ocr.EnableConsoleOutput = true;
                    ocr.EnablePdfOutput = true;
                    ocr.Languages = new SupportedLanguages[]
                    {
                        SupportedLanguages.English,
                        SupportedLanguages.French,
                        SupportedLanguages.Spanish,
                        SupportedLanguages.German,
                        SupportedLanguages.Italian
                    };

                    // Set PreProcessor options
                    PreProcessor preProcessor = new PreProcessor();
                    preProcessor.Deskew = true;
                    preProcessor.KeepOriginalImage = true;

                    // Read source TIFF file
                    ocr.ReadTIFFSource(@"..\..\..\..\documents\source\languages.tif");

                    // Perform OCR recognition
                    if (ocr.Recognize(preProcessor))
                    {
                        // Save output as searchable PDF
                        ocr.SavePDFOutput(@"..\..\..\..\documents\output\languages.pdf", true);
                    }

                    ocr.DeleteTemporaryFiles();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
        }
    }
}
```

2.13.3 Comments

Extended OCR can only process a maximum of 8 languages at a time.

2.14 Asian Languages

2.14.1 Requirement

OCR a document that has text in one of the supported Asian languages (Japanese, Korean, Simple Chinese, Traditional Chinese).

2.14.2 Solution

```
using Aquaforest.ExtendedOCR.Api;
using System;
using System.IO;

namespace AsianLanguages
{
    class Program
    {
        private static void Main(string[] args)
        {
            try
            {
                string resourceFolder = Path.GetFullPath(@"..\..\..\..\..\xbin\resources");

                using (Ocr ocr = new Ocr(resourceFolder))
                {
                    // Set OCR options
                    ocr.EnableConsoleOutput = true;
                    ocr.EnablePdfOutput = true;

                    // Set language to the desired Asian language (Japanese, Korean, Tchinese, Schinese)
                    // NOTE: Extended OCR cannot load more than one Asian language at a time.
                    // An Asian language can only be used in conjunction with 'English' and nothing else.
                    ocr.Languages = new SupportedLanguages[]
                    {
                        SupportedLanguages.Korean,
                        SupportedLanguages.English
                    };

                    // Optional: Set the engine to use for Asian languages.
                    // This can be helpful if one engine does not give good OCR results.
                    ocr.AsianOCREngine = AsianOCREngine.OcrAsian1;

                    // Set PreProcessor options
                    PreProcessor preProcessor = new PreProcessor();
                    preProcessor.Deskew = true;
                    preProcessor.KeepOriginalImage = true;

                    // Read source TIFF file
                    ocr.ReadTIFFSource(@"..\..\..\..\documents\source\korean.tif");

                    // Perform OCR recognition
                    if (ocr.Recognize(preProcessor))
                    {
                        // Save output as searchable PDF
                        ocr.SavePDFOutput(@"..\..\..\..\documents\output\korean.pdf", true);
                    }

                    ocr.DeleteTemporaryFiles();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
        }
    }
}
```

2.14.3 Comments

Extended OCR cannot load more than one Asian language at a time. An Asian language can only be used in conjunction with 'English' language and nothing else.

Note: You need to have the Asian language license to process documents with text in Japanese, Korean, Simple Chinese or Traditional Chinese.

2.15 Arabic Language

2.15.1 Requirement

Process documents with Arabic text.

2.15.2 Solution

```
using Aquaforest.ExtendedOCR.Api;
using System;
using System.IO;

namespace ArabicLanguage
{
    class Program
    {
        private static void Main(string[] args)
        {
            try
            {
                string resourceFolder = Path.GetFullPath(@"..\..\..\..\..\xbin\resources");

                using (Ocr ocr = new Ocr(resourceFolder))
                {
                    // Set OCR options
                    ocr.EnableConsoleOutput = true;
                    ocr.EnablePdfOutput = true;

                    // Set language to Arabic
                    ocr.Language = SupportedLanguages.Arabic;

                    // Set PreProcessor options
                    PreProcessor preProcessor = new PreProcessor();
                    preProcessor.Deskew = true;
                    preProcessor.KeepOriginalImage = true;

                    // Read source TIFF file
                    ocr.ReadTIFFSource(@"..\..\..\..\documents\source\arabic.tif");

                    // Perform OCR recognition
                    if (ocr.Recognize(preProcessor))
                    {
                        // Save output as searchable PDF
                        ocr.SavePDFOutput(@"..\..\..\..\documents\output\arabic.pdf", true);
                    }

                    ocr.DeleteTemporaryFiles();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
        }
    }
}
```

2.15.3 Comments

Note: You need to have the Arabic language license to process documents with Arabic text.

2.16 Hebrew Language

2.16.1 Requirement

Process documents with Hebrew text.

2.16.2 Solution

```
using Aquaforest.ExtendedOCR.Api;
using System;
using System.IO;
using System.Text;

namespace HebrewLanguage
{
    class Program
    {
        private static void Main(string[] args)
        {
            try
            {
                // Set output encoding enable Hebrew characters to be written to the console
                Console.OutputEncoding = Encoding.GetEncoding("Windows-1255");

                string resourceFolder = Path.GetFullPath(@"..\..\..\..\..\xbin\resources");

                using (Ocr ocr = new Ocr(resourceFolder))
                {
                    // Set OCR options
                    ocr.EnableConsoleOutput = true;
                    ocr.EnablePdfOutput = true;

                    // Set language to Hebrew - you will need a license that supports Hebrew
                    ocr.Language = SupportedLanguages.Hebrew;

                    // Set PreProcessor options
                    PreProcessor preProcessor = new PreProcessor();
                    preProcessor.Deskew = true;
                    preProcessor.LineRemoval = new LineRemoval { RemoveLines = true };
                    preProcessor.KeepOriginalImage = true;

                    // Read source TIFF file
                    ocr.ReadTIFFSource(@"..\..\..\..\documents\source\hebrew.tif");

                    // Perform OCR recognition
                    if (ocr.Recognize(preProcessor))
                    {
                        // Save output as searchable PDF
                        ocr.SavePDFOutput(@"..\..\..\..\documents\output\hebrew.pdf", true);
                    }

                    ocr.DeleteTemporaryFiles();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
        }
    }
}
```

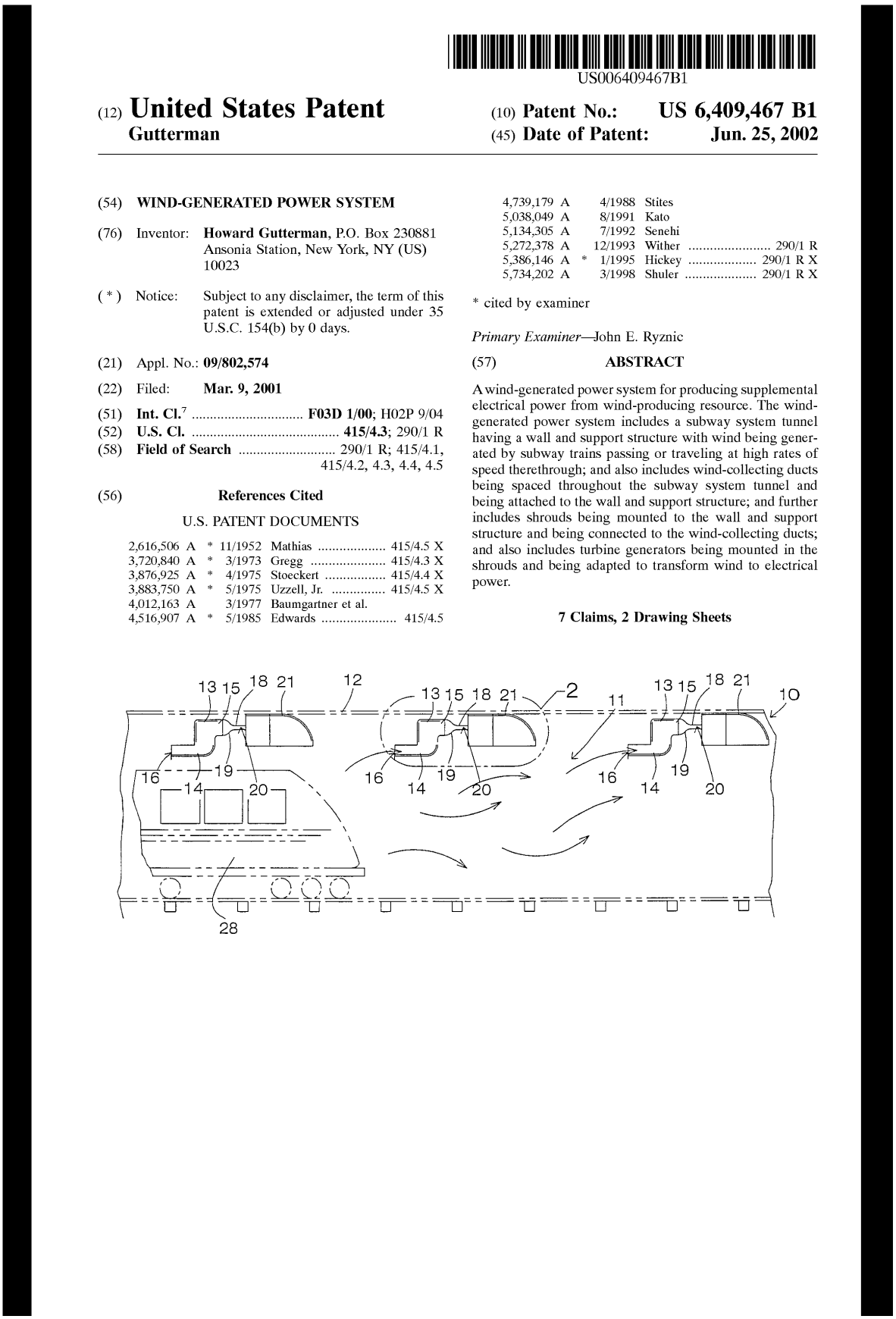
2.16.3 Comments

Note: You need to have the Hebrew language license to process documents with Hebrew text.

2.17 Dark Borders

2.17.1 Requirement

Remove dark borders surrounding a page. For instance:



US006409467B1

(12) **United States Patent**
Gutterman

(10) **Patent No.: US 6,409,467 B1**
(45) **Date of Patent: Jun. 25, 2002**

(54) **WIND-GENERATED POWER SYSTEM**

(76) Inventor: **Howard Gutterman**, P.O. Box 230881 Ansonia Station, New York, NY (US) 10023

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/802,574**

(22) Filed: **Mar. 9, 2001**

(51) **Int. Cl.**[7] **F03D 1/00**; H02P 9/04
(52) **U.S. Cl.** ..... **415/4.3**; 290/1 R
(58) **Field of Search** ..... 290/1 R; 415/4.1, 415/4.2, 4.3, 4.4, 4.5

(56) **References Cited**

U.S. PATENT DOCUMENTS

2,616,506	A	*	11/1952	Mathias	415/4.5 X
3,720,840	A	*	3/1973	Gregg	415/4.3 X
3,876,925	A	*	4/1975	Stoeckert	415/4.4 X
3,883,750	A	*	5/1975	Uzzell, Jr.	415/4.5 X
4,012,163	A		3/1977	Baumgartner et al.	
4,516,907	A	*	5/1985	Edwards	415/4.5
4,739,179	A		4/1988	Stites	
5,038,049	A		8/1991	Kato	
5,134,305	A		7/1992	Senehi	
5,272,378	A		12/1993	Wither	290/1 R
5,386,146	A	*	1/1995	Hickey	290/1 R X
5,734,202	A		3/1998	Shuler	290/1 R X

* cited by examiner

Primary Examiner—John E. Ryznic

(57) **ABSTRACT**

A wind-generated power system for producing supplemental electrical power from wind-producing resource. The wind-generated power system includes a subway system tunnel having a wall and support structure with wind being generated by subway trains passing or traveling at high rates of speed therethrough; and also includes wind-collecting ducts being spaced throughout the subway system tunnel and being attached to the wall and support structure; and further includes shrouds being mounted to the wall and support structure and being connected to the wind-collecting ducts; and also includes turbine generators being mounted in the shrouds and being adapted to transform wind to electrical power.

7 Claims, 2 Drawing Sheets

2.17.2 Solution

```
using Aquaforest.ExtendedOCR.Api;
using System;
using System.IO;

namespace RemoveDarkBorders
{
    internal class Program
    {
        private static void Main(string[] args)
        {
            try
            {
                using (Ocr ocr = new Ocr(@"C:\Aquaforest\OCRSDK\xbin\resources"))
                {
                    // Set OCR options
                    ocr.Language = SupportedLanguages.English;
                    ocr.EnablePdfOutput = true;
                    ocr.EnableConsoleOutput = true;

                    // Set PreProcessor options
                    PreProcessor preProcessor = new PreProcessor();
                    preProcessor.Deskew = true;
                    preProcessor.Autorotate = false;
                    preProcessor.RemoveDarkBorders = true;
                    preProcessor.KeepOriginalImage = false;

                    // Read source TIFF file
                    ocr.ReadTIFFSource(@"..\..\..\..\documents\source\dark_borders.tif");

                    // Perform OCR recognition
                    if (ocr.Recognize(preProcessor))
                    {
                        // Save output as searchable PDF
                        ocr.SavePDFOutput(@"..\..\..\..\documents\output\output.pdf", true);
                    }

                    ocr.DeleteTemporaryFiles();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
        }
    }
}
```

2.17.3 Comments

Note: The borders should be touching the edges of the page for this to work.

2.18 Generate PDF/A document

2.18.1 Requirement

OCR a TIFF document and generate a PDF/A compliant PDF file.

2.18.2 Solution

```
using Aquaforest.ExtendedOCR.Api;
using System;
using System.IO;

namespace ocr
{
    internal class PDFACompliance
    {
        private static void Main(string[] args)
        {
            try
            {
                using (Ocr ocr = new Ocr(@"C:\Aquaforest\OCRSDK\xbin\resources"))
                {
                    // Set OCR options
                    ocr.EnableConsoleOutput = true;
                    ocr.Language = SupportedLanguages.English;
                    ocr.EnablePdfOutput = true;
                    ocr.PDFVersion = PDFVersion.PDF_A2b;

                    // Set PreProcessor options
                    PreProcessor preProcessor = new PreProcessor();
                    preProcessor.Autorotate = false;
                    preProcessor.Deskew = true;

                    // Read source TIFF file
                    ocr.ReadTIFFSource(
                        Path.GetFullPath(@"..\..\..\..\documents\source\sample.tif"));

                    // Perform OCR recognition
                    if (ocr.Recognize(preProcessor))
                    {
                        // Save output as searchable PDF
                        ocr.SavePDFOutput(
                           Path.GetFullPath(@"..\..\..\..\documents\output\samplePDFA.pdf"), true);
                    }

                    ocr.DeleteTemporaryFiles();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Error in OCR Processing :" + e.Message);
            }
        }
    }
}
```

2.18.3 Comments

The `PDFVersion` enumerator has more PDF/A versions such as PDF 1.4 A-1a, PDF 1.7 A-2b and PDF 1.7 A-2a.

Note: PDF/A-1a and PDF/A-2a are not supported in Trial mode and when using "Native" `ExtractImageMethod`.

2.19 ASP.NET conversion

2.19.1 Requirement

Perform OCR from a file posted in an ASP.NET application.

2.19.2 Solution

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="Web_Application
._Default" %>

<!DOCTYPE html PUBLIC "-
//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
        <head runat="server">
                <title>Untitled Page</title>
        </head>
        <body>
                <form id="form1" runat="server">
                        <div>

        <asp:Label ID="Label1" runat="server" Text="BMP or TIFF : "></asp:Label>
                                <asp:FileUpload ID="FileUploadSource" runat="server" />

        <asp:Label ID="Label2" runat="server" Text="Convert To : "></asp:Label>

        <asp:DropDownList ID="DropDownListDestinationType" runat="server" Width="224px" Height="26p
x">

        <asp:ListItem Selected="True" Value="0">Portable Document Format (.pdf)</asp:ListItem>
                                        <asp:ListItem Value="1">Text (.txt)</asp:ListItem>
                                        <asp:ListItem Value="2">Rich Text (.rtf)</asp:ListItem>
                                        <asp:ListItem Value="3">Words (.docx)</asp:ListItem>
                                        <asp:ListItem Value="4">Excel (.xslx)</asp:ListItem>
                                </asp:DropDownList>
                        </div>
                        <div>

        <asp:Table ID="Table1" runat="server" Height="150px" Width="178px">
                                        <asp:TableRow>
                                                <asp:TableCell>
                                                        <Label>Despeckle</Label>
                                                </asp:TableCell>
                                                <asp:TableCell>

        <asp:DropDownList ID="DropDownListDespeckle" runat="server">

        <asp:ListItem Selected="True" Value="0">0</asp:ListItem>

        <asp:ListItem Value="1">1</asp:ListItem>

        <asp:ListItem Value="2">2</asp:ListItem>

        <asp:ListItem Value="3">3</asp:ListItem>

        <asp:ListItem Value="4">4</asp:ListItem>

        <asp:ListItem Value="5">5</asp:ListItem>

        <asp:ListItem Value="6">6</asp:ListItem>

        <asp:ListItem Value="7">7</asp:ListItem>

        <asp:ListItem Value="8">8</asp:ListItem>

        <asp:ListItem Value="9">9</asp:ListItem>
```

```
                                                    </asp:DropDownList>
                                                </asp:TableCell>
                                            </asp:TableRow>
                                            <asp:TableRow>
                                                <asp:TableCell>
                                                    <Label>Autorotate</Label>
                                                </asp:TableCell>
                                                <asp:TableCell>

        <asp:CheckBox ID="CheckBoxAutorotate" runat="server"></asp:CheckBox>
                                                </asp:TableCell>
                                            </asp:TableRow>
                                            <asp:TableRow>
                                                <asp:TableCell>
                                                    <Label>Deskew</Label>
                                                </asp:TableCell>
                                                <asp:TableCell>

        <asp:CheckBox ID="CheckBoxDeskew" runat="server"></asp:CheckBox>
                                                </asp:TableCell>
                                            </asp:TableRow>
                                            <asp:TableRow>
                                                <asp:TableCell>
                                                    <Label>Binarization</Label>
                                                </asp:TableCell>
                                                <asp:TableCell>

        <asp:CheckBox ID="CheckBoxBinarization" runat="server"></asp:CheckBox>
                                                </asp:TableCell>
                                            </asp:TableRow>
                                            <asp:TableRow>
                                                <asp:TableCell>
                                                    <Label>Language</Label>
                                                </asp:TableCell>
                                                <asp:TableCell>

        <asp:DropDownList ID="DropDownListLanguage" runat="server">
                                                    </asp:DropDownList>
                                                </asp:TableCell>
                                            </asp:TableRow>
                                        </asp:Table>
                                    </div>
                                    <br />
                                    <asp:Button ID="ButtonConvert" runat="server" Text="Convert"
           OnClick="ButtonConvert_Click" />
                                    <br />
                                    <br />
                                </form>
                                <div>
                                </div>
        </body>
</html>
```

```
using Aquaforest.ExtendedOCR.Api;
using System;
using System.IO;

namespace Web_Application
{
    public partial class _Default : System.Web.UI.Page
    {
        private Ocr ocr;
        private PreProcessor preProcessor;

        protected void Page_Load(object sender, EventArgs e)
        {
            if (!IsPostBack)
            {
                foreach (string language in Enum.GetNames(typeof(SupportedLanguages)))
                {
                    DropDownListLanguage.Items.Add(language);
                }

                int index = 0;
                foreach (int value in Enum.GetValues(typeof(SupportedLanguages)))
                {
                    DropDownListLanguage.Items[index].Value = value.ToString();
                    index++;
                }
            }
        }

        protected void ButtonConvert_Click(object sender, EventArgs e)
        {
            string tempLogFile = "";
            StreamWriter sw = null;

            try
            {
                tempLogFile = Path.GetTempFileName();
                sw = new StreamWriter(tempLogFile);
                sw.AutoFlush = true;
                Console.SetOut(sw);

                if (FileUploadSource.HasFile)
                {
                    string fileName = Path.GetFileName(FileUploadSource.FileName);
                    string extension = Path.GetExtension(FileUploadSource.FileName);

                    string sourceFile = Path.Combine(Path.GetTempPath(), fileName);
                    FileUploadSource.SaveAs(sourceFile);

                    SetupOcr();

                    switch (extension.ToLower())
                    {
                        case ".jpg":
                        case ".jpeg":
                        case ".png":
                        case ".bmp":
                            ocr.ReadBMPSource(sourceFile);
                            break;

                        case ".tif":
                        case ".tiff":
                            ocr.ReadTIFFSource(sourceFile);
                            break;

                        case ".pdf":
                            ocr.ReadPDFSource(sourceFile);
                            break;
```

```
                }

                SetOutputType();

                if (ocr.Recognize(preProcessor))
                {
                    SaveOcrOutput();
                }
                else
                {
                    Response.Write("ERROR<br />");
                    using (StreamReader sr = new StreamReader(tempLogFile))
                    {
                        Response.Write(sr.ReadToEnd().Replace(Environment.NewLine, "<br />"));
                    }
                    Response.End();
                }
            }
        }
        catch (Exception ex)
        {
            Response.Write(ex.Message.Replace(Environment.NewLine, "<br />") + "<br />");
            Response.End();
        }
        finally
        {
            if (ocr != null) ocr.Dispose();
            if (sw != null) sw.Dispose();
            if (File.Exists(tempLogFile)) File.Delete(tempLogFile);
        }

        Response.End();
    }

    private void SetOutputType()
    {
        int outputType = Convert.ToInt32(DropDownListDestinationType.SelectedItem.Value);
        switch (outputType)
        {
            case 1:
                ocr.EnableTextOutput = true;
                break;

            case 2:
                ocr.EnableRtfOutput = true;
                break;

            case 3:
                ocr.EnableDocxOutput = true;
                break;

            case 4:
                ocr.EnableXlsxOutput = true;
                break;

            case 0:
            default:
                ocr.EnablePdfOutput = true;
                break;
        }
    }

    private void SaveOcrOutput()
    {
        string destinationFile = Path.GetTempFileName() + "output.";
        int outputType = Convert.ToInt32(DropDownListDestinationType.SelectedItem.Value);
        switch (outputType)
        {
```

```
            case 1:
                Response.Write("Saving TXT<BR>");
                destinationFile += "txt";
                ocr.SaveTextOutput(destinationFile, true);
                break;

            case 2:
                Response.Write("Saving RTF<BR>");
                destinationFile += "rtf";
                ocr.SaveRTFOutput(destinationFile, true);
                break;

            case 3:
                Response.Write("Saving DOCX<BR>");
                destinationFile += "docx";
                ocr.SaveDOCXOutput(destinationFile, true);
                break;

            case 4:
                Response.Write("Saving XLSX<BR>");
                destinationFile += "xlsx";
                ocr.SaveXLSXOutput(destinationFile, true);
                break;

            case 0:
            default:
                Response.Write("Saving PDF<BR>");
                destinationFile += "pdf";
                ocr.SavePDFOutput(destinationFile, true);
                break;
        }

        ocr.DeleteTemporaryFiles();
        DownloadFile(destinationFile, true);
    }

    private void SetupOcr()
    {
        if (ocr != null) ocr.Dispose();

        string resourcefolder = @"C:\Aquaforest\OCRSDK\xbin\resources";

        SetSearchPath(resourcefolder);

        ocr = new Ocr(resourcefolder);
        ocr.License = "";

        ocr.Language = (SupportedLanguages)Convert.ToInt32(
                            DropDownListLanguage.SelectedItem.Value);

        preProcessor = new PreProcessor();
        preProcessor.Autorotate = CheckBoxAutorotate.Checked;
        preProcessor.Deskew = CheckBoxDeskew.Checked;
        preProcessor.Despeckle = DropDownListDespeckle.SelectedIndex;
        preProcessor.Binarization = new Binarization()
        {
            Binarize = CheckBoxBinarization.Checked
        };
    }

    private void SetSearchPath(string resourcefolder)
    {
        string searchPath = Directory.GetParent(resourcefolder).FullName;

        if (IntPtr.Size == 4)
        {
            searchPath = Path.Combine(searchPath, "x86");
        }
```

```
            else if (IntPtr.Size == 8)
            {
                searchPath = Path.Combine(searchPath, "x64");
            }

            string currentEnvironmentVariables = Environment.GetEnvironmentVariable("PATH");
            if (!currentEnvironmentVariables.Contains(searchPath))
            {
                Environment.SetEnvironmentVariable(
                    "PATH", currentEnvironmentVariables + ";" + searchPath);
            }
        }

        private void DownloadFile(string fileName, bool forceDownload)
        {
            string path = fileName;// MapPath(fileName);
            string name = Path.GetFileName(path);
            string ext = Path.GetExtension(path);
            string type = "";

            if (ext != null)
            {
                switch (ext.ToLower())
                {
                    case ".htm":
                    case ".html":
                        type = "text/HTML";
                        break;

                    case ".txt":
                        type = "text/plain";
                        break;

                    case ".doc":
                    case ".rtf":
                        type = "Application/msword";
                        break;

                    case ".docx":
                        type = "application/vnd.openxmlformats-
officedocument.wordprocessingml.document";
                        break;

                    case ".xlsx":
                        type = "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet";
                        break;

                    case ".pdf":
                        type = "Application/pdf";
                        break;
                }
            }

            if (forceDownload)
            {
                Response.Clear();
                Response.AppendHeader("content-disposition", "attachment; filename=" + name);
            }

            if (type != "")
            {
                Response.ContentType = type;
            }

            Response.WriteFile(path);
        }
    }
}
```

2.19.3 Comments

Make sure to set the correct path of the resource folder in the `Ocr` object `ocr = new Ocr(resourcefolder)]` and add the following in the `web.config` file:

```
<runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
        <probing privatePath="bin\resources"/>
        <probing privatePath="bin\x86"/>
        <probing privatePath="bin\x64"/>
    </assemblyBinding>
</runtime>
```

2.20 Write recognition results to CSV file

2.20.1 Requirement

Extract name-value pairs from an image and write them to a CSV file.

2.20.2 Solution

```
using Aquaforest.ExtendedOCR.Api;
using Aquaforest.ExtendedOCR.Shared;
using System;
using System.IO;

namespace RecognitionToExcel
{
    internal class Program
    {
        private static string csvOutputFilePath = @"..\..\..\..\documents\output\invoices.csv";

        private static void Main(string[] args)
        {
            // Set PreProcessor options
            PreProcessor preProcessor = new PreProcessor();
            preProcessor.Deskew = true;

            InitialiseExcel();

            DirectoryInfo dInfo = new DirectoryInfo(@"..\..\..\..\documents\source\invoices");

            foreach (FileInfo f in dInfo.GetFiles("*.tiff"))
            {
                using (Ocr ocr = new Ocr(@"C:\Aquaforest\OCRSDK\xbin\resources"))
                {
                    // Set OCR options
                    ocr.Language = SupportedLanguages.English;
                    ocr.EnableConsoleOutput = true;

                    Console.WriteLine("Processing file: {0}", f.FullName);
                    ocr.ReadTIFFSource(f.FullName); // Read source TIFF file

                    if (!ocr.Recognize(preProcessor)) continue;

                    bool[] skip = new bool[2];
                    ExcelRow row = new ExcelRow();

                    Words words = ocr.ReadPageWords(1);

                    for (int i = 0; i < words.Count; i++)
                    {
                        if (!skip[0] && (words[i].Word == "Invoice" && (i + 1 < words.Count)
                            && words[i + 1].Word == "Number:"))
                        {
                            if (i + 2 < words.Count)
                            {
                                row.InvoiceNumber = words[i + 2].Word;
                                i = i + 2;
                                skip[0] = true;
                                continue;
                            }
                        }

                        if (!skip[1] && (words[i].Word == "Invoice" && (i + 1 < words.Count)
                            && words[i + 1].Word == "Date:"))
                        {
                            if (i + 2 < words.Count)
                            {
                                row.InvoiceDate = words[i + 2].Word;
```

```
                                i = i + 2;
                                skip[1] = true;
                                continue;
                            }
                        }

                        if (skip[0] && skip[1])
                        {
                            break;
                        }
                    }

                    WriteRow(row);
                }
            }
        }

        private static void InitialiseExcel()
        {
            File.WriteAllText(csvOutputFilePath,
                "Invoice Number,Invoice Date" + Environment.NewLine);
        }

        private static void WriteRow(ExcelRow row)
        {
            string contents = string.Format("{0},{1}{2}", row.InvoiceNumber, row.InvoiceDate,
                Environment.NewLine);

            Console.WriteLine(contents);
            File.AppendAllText(csvOutputFilePath, contents);
        }
    }

    internal class ExcelRow
    {
        public string InvoiceNumber { get; set; }
        public string InvoiceDate { get; set; }
    }
}
```

2.20.3 Comments

This example will produce accurate results only for the files given in the folder "[Installation path]\xsamples\documents\source\invoices". You will need to make some amendments to the code for documents with different formatting or template.

3 Barcode

3.1 Get a single barcode from page

3.1.1 Requirement

Scans pages for barcodes and try to identify one barcode per page.

3.1.2 Solution

```
using System;
using System.IO;
using System.Linq;
using Aquaforest.BarcodeReader.Api;
using Aquaforest.BarcodeReader.Common;

namespace GetOneBarcodePerPage
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                BarcodeReaderSettings settings = new BarcodeReaderSettings();

                //Barcode formats to look for
                settings.BarcodeFormats[BarcodeFormat.All_1D] = true;
                settings.BarcodeFormats[BarcodeFormat.AZTEC] = true;
                settings.BarcodeFormats[BarcodeFormat.DATA_MATRIX] = true;
                settings.BarcodeFormats[BarcodeFormat.MAXICODE] = true;
                settings.BarcodeFormats[BarcodeFormat.MSI] = true;
                settings.BarcodeFormats[BarcodeFormat.PDF_417] = true;
                settings.BarcodeFormats[BarcodeFormat.PLESSEY] = true;
                settings.BarcodeFormats[BarcodeFormat.QR_CODE] = true;
                settings.BarcodeFormats[BarcodeFormat.UPC_EAN_EXTENSION] = true;

                BarcodeReader reader = new BarcodeReader(settings);
                string resourceFolder = Path.GetFullPath(@"C:\Aquaforest\OCRSDK\bin");
                string currentEnvironmentVariables = Environment.GetEnvironmentVariable("PATH");
                if (!currentEnvironmentVariables.Contains(resourceFolder))
                {
                     Environment.SetEnvironmentVariable(
                        "PATH", currentEnvironmentVariables + ";" + resourceFolder);
                }
                reader.ResourceFolder = resourceFolder;

                reader.DecodeImage(
                    Path.GetFullPath(@"..\..\..\..\documents\source\sample-barcodes.tif"));

                Console.WriteLine();
                Console.WriteLine("Decode Results");
                Console.WriteLine(new string('-', 30));
                foreach (var result in reader.DecodeResults.Where(r => r.Value.Success))
                {
                    BarcodeResult r = result.Value.BarcodeResult;
                    Console.WriteLine("Found barcode on page: {0}", result.Key);
                    Console.WriteLine("Barcode Format: {0}", r.BarcodeFormat.ToString());
                    Console.WriteLine("Text: {0}", r.Text);
                    Console.WriteLine("Type: {0}", r.Type.ToString());
                    Console.WriteLine();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
        }
    }
}
```

3.2 Get a multiple barcodes from page

3.2.1 Requirement

Scans pages for barcodes and try to identify all barcodes available in each page.

3.2.2 Solution

```
using System;
using System.IO;
using System.Linq;
using Aquaforest.BarcodeReader.Api;
using Aquaforest.BarcodeReader.Common;

namespace GetMultipleBarcodesPerPage
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                BarcodeReaderSettings settings = new BarcodeReaderSettings();
                settings.ReadMultipleBarcodes = true; //read multiple barcodes per page

                //Barcode formats to look for
                settings.BarcodeFormats[BarcodeFormat.CODE_39] = true;
                settings.BarcodeFormats[BarcodeFormat.CODE_128] = true;
                settings.BarcodeFormats[BarcodeFormat.CODE_93] = true;
                settings.BarcodeFormats[BarcodeFormat.ITF] = true;
                settings.BarcodeFormats[BarcodeFormat.QR_CODE] = true;

                BarcodeReader reader = new BarcodeReader(settings);
                string resourceFolder = Path.GetFullPath(@"C:\Aquaforest\OCRSDK\bin");
                string currentEnvironmentVariables = Environment.GetEnvironmentVariable("PATH");
                if (!currentEnvironmentVariables.Contains(resourceFolder))
                {
                    Environment.SetEnvironmentVariable(
                        "PATH", currentEnvironmentVariables + ";" + resourceFolder);
                }
                reader.ResourceFolder = resourceFolder;

                reader.DecodeImage(
                    Path.GetFullPath(@"..\..\..\..\documents\source\sample-barcodes.tif"));

                Console.WriteLine();
                Console.WriteLine("Decode Results");
                Console.WriteLine(new string('-', 30));

                foreach (var result in reader.DecodeResults.Where(r => r.Value.Success))
                {
                    var pageResults = result.Value.BarcodeResults;
                    Console.WriteLine("Found {0} barcode{1} on page {2}"+Environment.NewLine,
                    pageResults.Count, pageResults.Count > 1 ? "s":"", result.Key);

                    foreach (var r in pageResults)
                    {
                        Console.WriteLine("Barcode Format: {0}", r.BarcodeFormat.ToString());
                        Console.WriteLine("Text: {0}", r.Text);
                        Console.WriteLine("Type: {0}", r.Type.ToString());
                        Console.WriteLine();
                    }
                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
        }
    }
}
```

3.3 Split multi-page document by barcode

3.3.1 Requirement

Split a multi-page document by barcodes identified.

3.3.2 Solution

```
using System;
using System.IO;
using System.Linq;
using Aquaforest.BarcodeReader.Api;
using Aquaforest.BarcodeReader.Common;

namespace SplitDocumentByBarcode
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                BarcodeReaderSettings settings = new BarcodeReaderSettings();
                settings.Morph = "e2.2";

                BarcodeReader reader = new BarcodeReader(settings);

                string resourceFolder = Path.GetFullPath(@"C:\Aquaforest\OCRSDK\bin");

                string currentEnvironmentVariables =
                    Environment.GetEnvironmentVariable("PATH");
                if (!currentEnvironmentVariables.Contains(resourceFolder))
                {
                    Environment.SetEnvironmentVariable(
                        "PATH", currentEnvironmentVariables + ";" + resourceFolder);
                }
                reader.ResourceFolder = resourceFolder;

                reader.PerformPreprocessing = true;
                reader.EnableConsoleOutput = true;
                reader.SplitByBarcode = true;
                reader.SplitMode = SplitMode.BARCODE_FIRST_PAGE;

                reader.SplitOutputPath =
                    Path.GetFullPath(@"..\..\..\..\documents\output\%VALUE%_%INDEX%.pdf");

                reader.DecodePdf(
                    Path.GetFullPath(@"..\..\..\..\documents\source\multipage.pdf"));
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
        }
    }
}
```

3.3.3 Comments

This example splits the document each time a barcode is recognized, making the barcode page the first page of each split document. This is achieved by using `SplitMode.BARCODE_FIRST_PAGE`. There are two other split modes available:

- `SplitMode.BARCODE_LAST_PAGE`
  The identified barcodes will be in the last page of each split document.

- `SplitMode.REMOVE_BARCODE_PAGE`
  The barcodes will be removed from each split document.

The `%VALUE%` parameter in the `SplitOutputPath` is replaced by the decoded barcode value whereas the `%INDEX%` parameter is replaced by the current document split number.