Курсовая работа по теме: «Сокрытие ЦВЗ в стегоноконтейнер в формате BMP методом блочного скрытия»

Выполнил студент группы БПЗ1501
Иванов И.И.

Метод блочного скрытия

- Данный метод позволяет встраивать в изображение сообщение, путем разбивания контейнера на блоки произвольной конфигурации и встраивания в каждый блок одного бита сообщения.
- Встраивание происходит в несколько этапов:

1) Изображение-оригинал разбивается на непересекающиеся блоки произвольной конфигурации. Для каждого блока вычисляется бит четности – XOR всех младших битов;

2) В каждом блоке скрывается один бит сообщения. Если бит четности совпадает со скрываемым, то инвертируется один из НЗБ блока, чтобы они совпадали;

3) Выбор блока происходит произвольно, возможно, с использованием стегоключа.

Алгоритм встраивания изображения

Достоинства и недостатки

- Для метода блочного скрытия можно выделить следующие преимущества: существование возможности модифицировать значение такого пикселя в блоке, изменение которого приведет к минимальному изменению статистики контейнера; влияние последствий встраивания секретных данных в контейнер можно уменьшить за счет увеличения размера блока. Недостатком этого метода является низкая устойчивость к искажениям.

Задание:

1. Тип стенографического контейнера: видео,
2. Тип скрываемого файла: ЦВЗ (QR-код),
3. Метод стенографического скрытия: блочный метод,
4. Алгоритмы оценки качества восприятия стеганоконтейнера после скрытия информации:
    - 4.1 Среднеквадратическая ошибка (Mean Squade Error -MSE),
    - 4.2 Нормированная среднеквадратическая ошибка (Normalized Mean Square Error – NMSE),
    - 4.3 Отношение «сигнал/шум» (Signal to Noise Ratio - SNR),
    - 4.4 Максимальное отношение «сигнал/шум» (Peak Signal to Noise Ratio – PSNR).

Исходные контейнер и QR-код

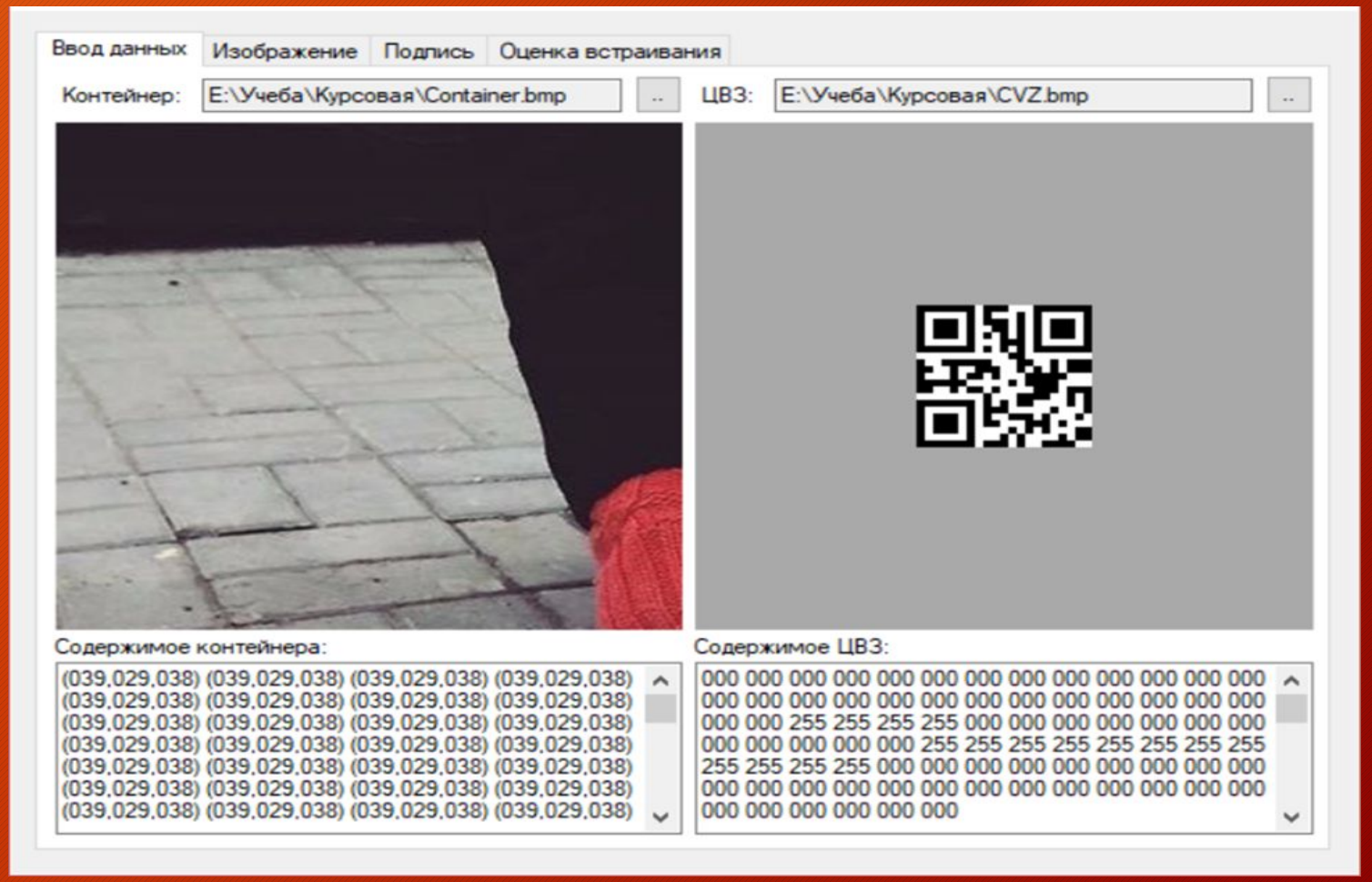
Контейнер-изображение формата bmp 750*744

ЦВЗ формата bmp 84*84

Пошаговая реализация метода

- Шаг №1. Загружаем исходные: контейнер-изображения и ЦВЗ.

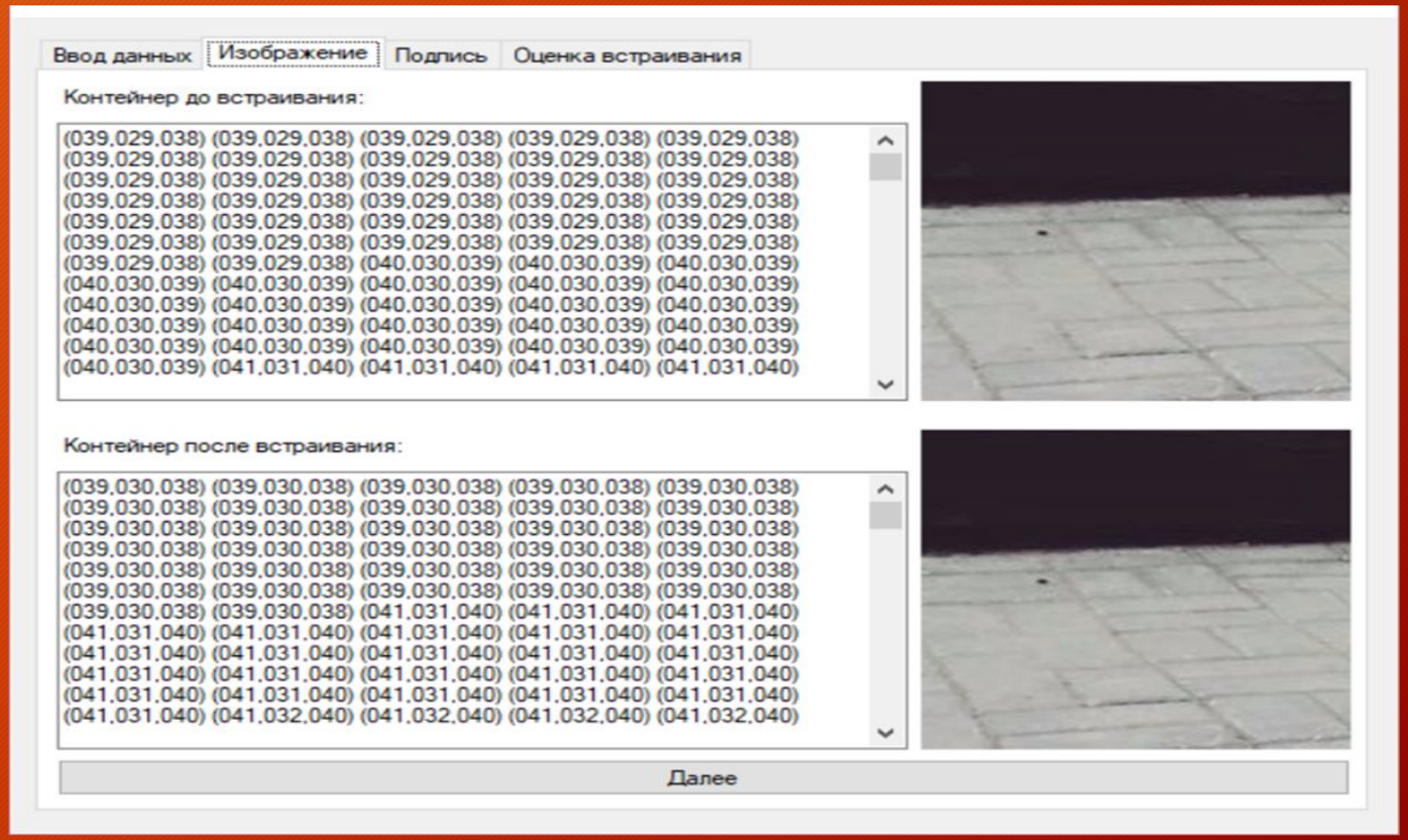
Считываем изображения, представляем их в цветовой модели RGB и получаем размер ЦВЗ.



```
using System.Drawing;
using System.IO;

namespace tsvsi_vlasov
{
   class SignatureProvider_Image : SignatureProviderBase, ISignatureProvider
   {
      public SignatureProvider_Image()
      {
         Data = null;
         DataLength = 0;
      }

      public bool SetSignature(string data)
      {
         if (data == null || !File.Exists(data)) return false;
         if (!Path.HasExtension(data) || !Path.GetExtension(data).Equals(".bmp"))
return false;

         var img = Image.FromFile(data) as Bitmap;
         if (img == null) return false;
         DataLength = img.Width;
         Data = new byte[img.Height][];
         for (var i = 0; i < img.Height; i++)
         {
            Data[i] = new byte[img.Width];
            for (var j = 0; j < img.Width; j++)
            {
               var value = (img.GetPixel(j, i).R + img.GetPixel(j, i).G +
img.GetPixel(j, i).B) / 3;
               if (value > 255) value = 255;
               if (value < 0) value = 0;
               Data[i][j] = (byte)value;
            }
         }
         img.Dispose();

         return true;
      }

      public void ExportToFile(string path)
      {
         if (File.Exists(path)) File.Delete(path);
         //var img = new Bitmap(Data[0].Length, Data.Length);
         var img = new Bitmap(Data[0].Length, Data[0].Length);
         for (var j = 0; j < Data[0].Length; j++)
            for (var i = 0; i < Data[j].Length; i++)
            {
               img.SetPixel(i, j, Color.FromArgb(255, Data[j][i], Data[j][i],
Data[j][i]));
            }
         img.Save(path);
         img.Dispose();
      }

      public override string ToString()
      {
         return "Image";
      }
   }
}
```

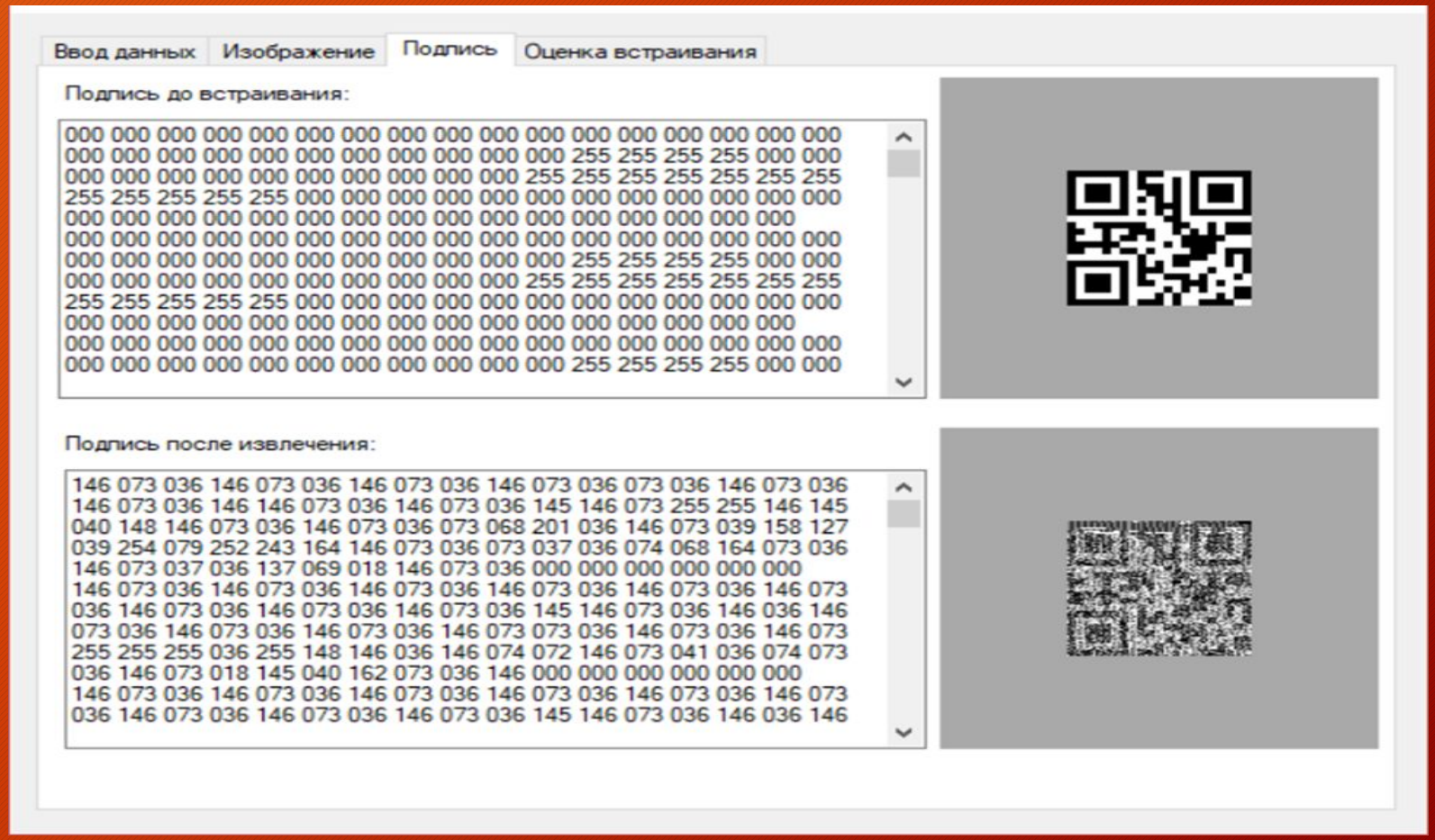
- Шаг №2. Создание изображения, содержащего ЦВЗ.

На этапе встраивания контейнер делим на блоки по количеству встраиваемых данных. В каждый блок контейнера встраиваем свой бит ЦВЗ.



```
using System;
using System.Drawing;
using System.Text;

namespace tsvsi_vlasov
{
  /// <summary>
  /// Алгоритм блочного скрытия
  /// </summary>
  class Algorithm_BlockHiding : AlgorithmBase, ISteganographyAlgorithm
  {
    public int BlockCount { get; set; }
    private int _blocksPerWidth, _blocksPerHeight;

    public Algorithm_BlockHiding()
    {
      BlockCount = 0;
    }

    public bool SignImage()
    {
      if (DataImage == null || Signature == null) return false;

      // Prepare signature
      var signatureBin = new StringBuilder();
      var signatureIndex = 0;
      var sign = Signature.GetSignature();
      _blocksPerWidth = 0;
      _blocksPerHeight = sign.Length;
      for (var i = 0; i < sign.Length; i++)
      {
        if (_blocksPerWidth < sign[i].Length * 8) _blocksPerWidth =
sign[i].Length * 8;
        for (var j = 0; j < sign[i].Length; j++)
          signatureBin.Append(Convert.ToString(sign[i][j], 2).PadLeft(8, '0'));

        //signatureBin.Append('|');
      }

      if (BlockCount > 0)
      {
        if (BlockCount < _blocksPerWidth * _blocksPerHeight * 8) return false;

        // Reinit maxWidth/maxHeight
        throw new ArgumentException("Не поддерживается");
      }
      else
      {
        BlockCount = _blocksPerWidth * _blocksPerHeight * 8;
      }

      var blockWidth = (int)Math.Floor(1.0 * DataImage.Width /
_blocksPerWidth);
      var blockHeight = (int)Math.Floor(1.0 * DataImage.Height /
_blocksPerHeight);

      for (var y = 0; y < _blocksPerHeight && signatureIndex <
signatureBin.Length; y++)
      {
        for (var x = 0; x < _blocksPerWidth && signatureIndex <
signatureBin.Length; x++)
        {
          int bit = (signatureBin[signatureIndex++] == '0' ? 0 : 1);
          if (SumsBlock(x * blockWidth, y * blockHeight, blockWidth,
blockHeight) != bit)
            InvertDataBlock(x * blockWidth, y * blockHeight, blockWidth,
blockHeight);
        }
      }

      return (signatureIndex >= signatureBin.Length);
    }

    private int SumsBlock(int x, int y, int width, int height)
    {
      var sum = 0;
      for (var yImg = y; yImg < y + height; yImg++)
      {
        for (var xImg = (int)Math.Floor(x / 3.0); width > 0; xImg++)
        {
          for (var cImg = x % 3; cImg < 3; cImg++)
          {
            var col = DataImage.GetPixel(xImg, yImg);

            switch (cImg)
            {
              case 0:
                sum += col.R % 2;
                break;
              case 1:
                sum += col.G % 2;
                break;
              case 2:
                sum += col.B % 2;
                break;
            }
            if (sum > 1) sum = 0;
            width--;
          }
        }
      }

      return sum;
    }

    private void InvertDataBlock(int x, int y, int width, int height)
    {
      for (var yImg = y; yImg < y + height; yImg++)
      {
        for (var xImg = (int)Math.Floor(x / 3.0); width > 0; xImg++)
        {
          for (var cImg = x % 3; cImg < 3; cImg++)
          {
            var col = DataImage.GetPixel(xImg, yImg);

            switch (cImg)
            {
              case 0:
                if (col.R > 0 && col.R < 255)
                {
                  DataImage.SetPixel(xImg, yImg, Color.FromArgb(255,
col.R + 1, col.G, col.B));
                  return;
                }
                break;
              case 1:
                if (col.G > 0 && col.G < 255)
                {
                  DataImage.SetPixel(xImg, yImg, Color.FromArgb(255,
col.R, col.G + 1, col.B));
                  return;
                }
                break;
              case 2:
                if (col.B > 0 && col.B < 255)
                {
                  DataImage.SetPixel(xImg, yImg, Color.FromArgb(255,
col.R, col.G, col.B + 1));
                  return;
                }
                break;
            }

            width--;
          }
        }
      }
    }
```

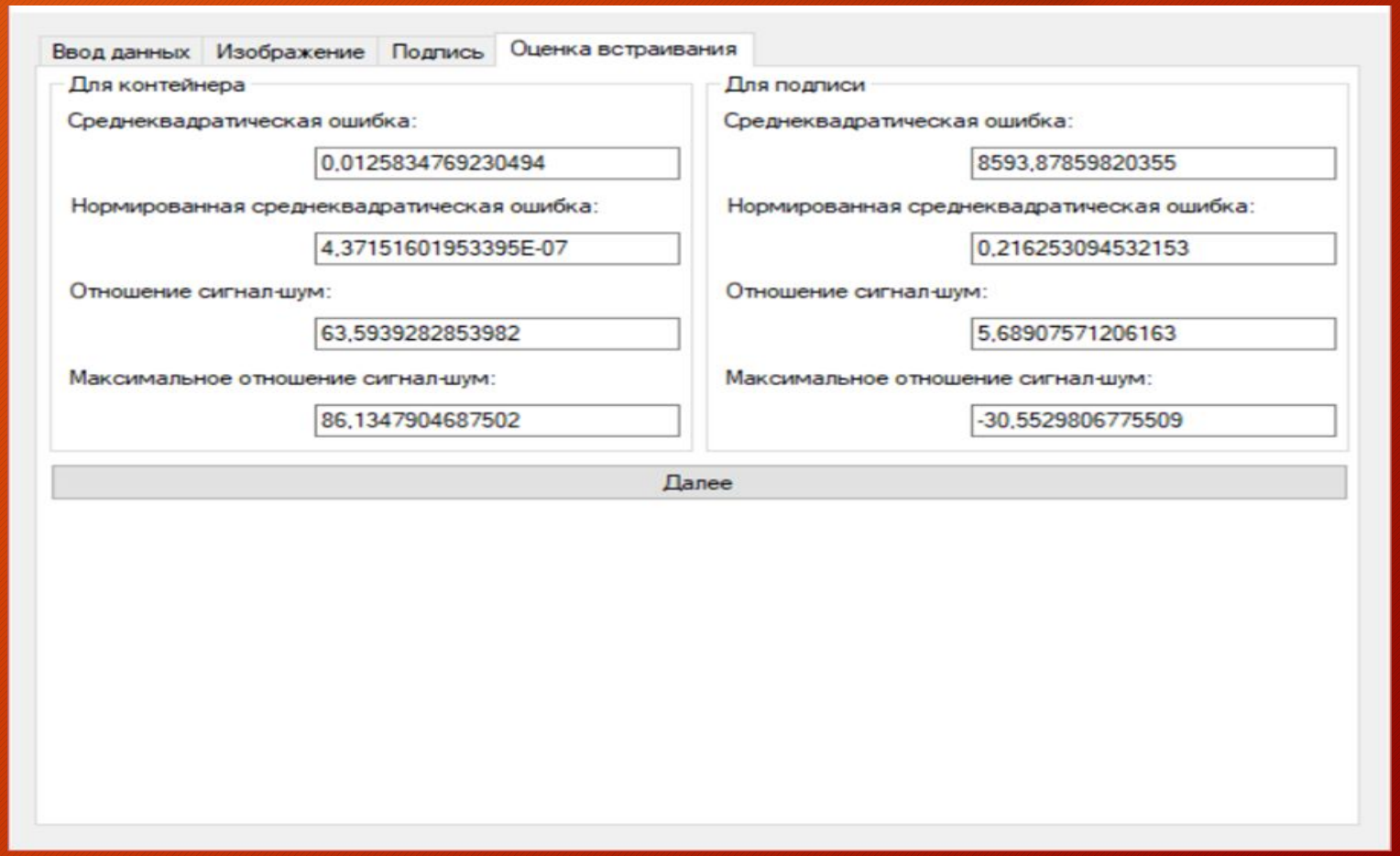
- Шаг №3. Извлечение скрытого сообщения из изображения.
- При извлечении нужно знать на сколько блоков делили контейнер(ключ для извлечения). Задаем эти числа (по ширине и по высоте). Считываем данные из блоков и сохраняем их в новую ЦВЗ.



```
public bool UnsignImage()
        {
            if (DataImage == null) return false;
            var signatureBin = new StringBuilder();

            var blockWidth = (int)Math.Floor(1.0 * DataImage.Width /
_blocksPerWidth);
            var blockHeight = (int)Math.Floor(1.0 * DataImage.Height /
_blocksPerHeight);

            for (var y = 0; y < _blocksPerHeight; y++)
            {
                for (var x = 0; x < _blocksPerWidth; x++)
                {
                    int tmp = SumsBlock(x * blockWidth, y * blockHeight, blockWidth,
blockHeight);
                    signatureBin.Append(tmp);
                }
                signatureBin.Append('\n');
            }

            var signatureStr = signatureBin.ToString().Split('\n');
            var sign = new byte[signatureStr.Length][];
            for (var i = 0; i < signatureStr.Length; i++)
            {
                sign[i] = new byte[(int)Math.Floor(signatureStr[i].Length / 8.0)];
                for (var j = 0; j < sign[i].Length; j++)
                    sign[i][j] = Convert.ToByte(signatureStr[i].Substring(j * 8, 8), 2);
            }
            Signature.UpdateSignature(sign);

            return true;
        }

        public override string ToString()
        {
            return "BlockHiding";
        }
    }
}
```

Оценка качества встраивания информации

- После встраивания информации необходимо произвести оценку изображения контейнера и ЦВЗ. Оценка производится в соответствии с заданием.



Спасибо за внимание