Comparison of Denoising Filters

5 April 2003

More about last MSU method: http://compression.ru/video/denoising/

Table of contents

Introduction

В ходе работы были рассмотрены некоторые из наиболее распространенных фильтров для шумоподавления, действующие как в пространственной (spatial), так и во временной (temporal) областях (все фильтры разработаны для программы VirtualDub). Были исследованы алгоритмы их работы и проведено их сравнение по качеству работы (для этого применялась программа LUVMetric).

Итак, вначале будут рассмотрены следующие фильтры для VirtualDub (Результаты показаны на видеоролике susi.avi, 194 кадр; все фильтры проверялись с настройками по умолчанию):

Пространственные (spatial)

- 2d cleaner filter

- Spatial Smoother filter

Временные (temporal)
- Chroma Noise Reduction filter
- Dynamic Noise Reduction filter

И другие (кратко):
- VHS filter
- KNRC filter
- Smart Smoother 1.1 filter
- Smart Smoother 2.11 filter
- Static Noise Reduction filter
- Area Smoother filter
- Temporal Cleaner filter
- Video DeNoise filter

Затем будут рассмотрены лучшие, по нашему мнению, фильтры при различных исходных параметрах и для каждого фильтра будут подобраны лучшие универсальные параметры.

И, наконец, будут показаны результаты обработки этими фильтрами при найденных параметрах ряда фильмов.

Spatial Filters

Основные параметры для пространственных фильтров – это порог (threshold) и радиус (radius).

Порог задает максимальную разницу между исходным пикселом и соседним с ним, при которой значение соседнего пиксела будет оказывать влияние на значение исходного. Чем больше порог, тем сильнее происходит видимое размытие изображения.

Радиус определяет область вокруг исходного пиксела, точки которой будут рассматриваться при определении нового значения исходного пиксела. Чем больше радиус, тем лучше удаляется шум, но вместе с тем значительно замедляется скорость работы фильтра.

Рассмотрим поподробнее два пространственных фильтра.

2D Cleaner Filter **by Jim Casaburi**

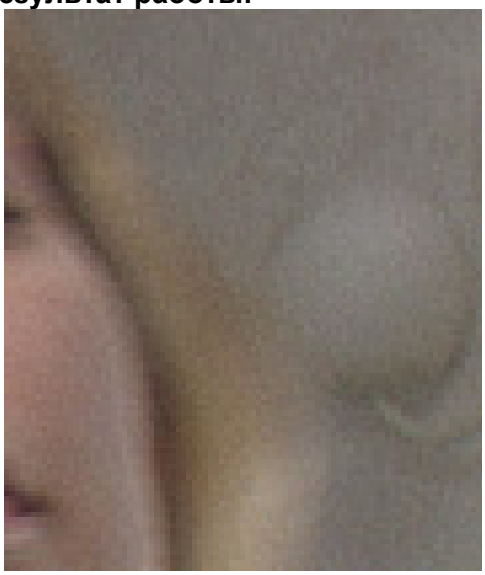
Принцип работы этого фильтра основан на следующем: он заменяет значение каждого пиксела на среднее значение тех его соседей, значение которых отличается от данного пиксела не более чем на заданную величину (порог). При этом соседи рассматриваются в области, определенной радиусом.

Благодаря этому низкоуровневый шум размывается, а резкие детали остаются нетронутыми.

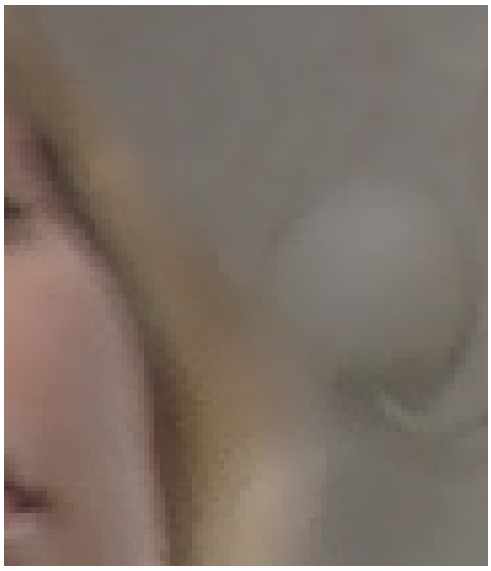
Алгоритм:

```
for (each pixel of the current video frame)
{
    GetRGB (source_pixel, r, g, b);
    tot_red = tot_green = tot_blue = 0;
    count_red = count_green = count_blue = 0;
    for (each pixel in the specified radius)
    {
        GetRGB (neighbour_pixel, r1, g1, b1);
        if  (abs(r1-r) < Threshold)
            tot_red += r1;  count_red ++;
        if  (abs(g1-g) < Threshold)
            tot_green += g1;  count_green ++;
        if  (abs(b1-b) < Threshold)
            tot_blue += b1;  count_blue ++;
    }
    destination_pixel = RGB (tot_red / count_red,
                             tot_green / count_green ,
                             tot_blue / count_blue );
}
```

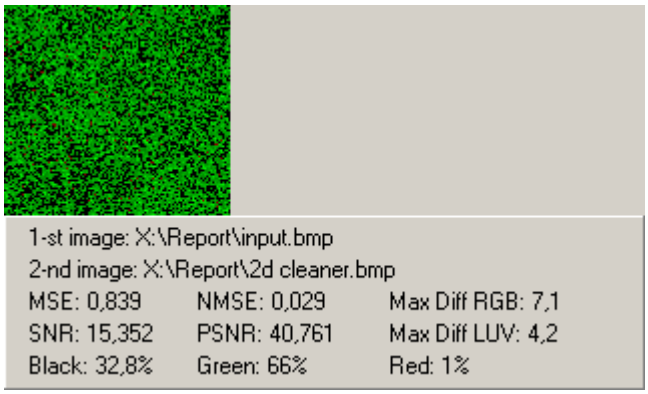
Результат работы:



Picture 1. Source image (susi.avi; 194 frame)



Picture 2. Processed image (default parameters)



Picture 3. Comparison of the source and processed images using LUV Metric

Spatial Smoother **by Ioura Batugowski**

Общий принцип его работы такой же, как и у фильтра 2d cleaner за некоторыми отличиями: если в 2d cleaner окружающие пикселы влияют на центральный одинаково – независимо от того, на каком расстоянии они от него находятся, то в Spatial Smoother чем дальше пиксел от центрального, тем меньше он оказывает на него влияние. Это достигается благодаря использованию специальной таблицы, которая задается следующим образом:

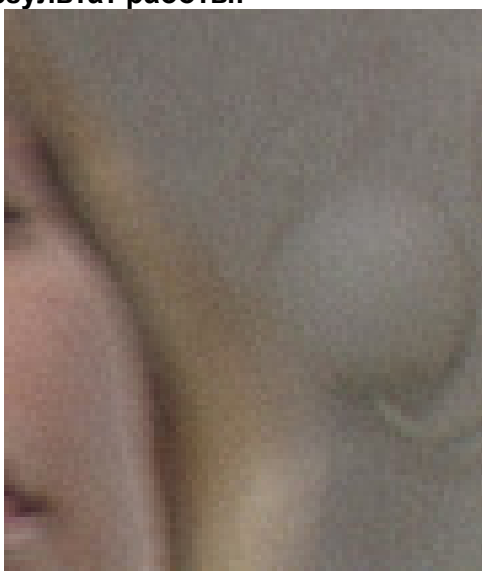
```
for  (i=0; i<=511; ++i)  square_tab[i] = (i-255) ^ 2;
```

Так же, если в 2d cleaner учитывается близость значений окружающих пикселей к центральному по каждой цветовой компоненте отдельно, то в Spatial Smoother пикселы сравниваются по всем компонентам сразу.

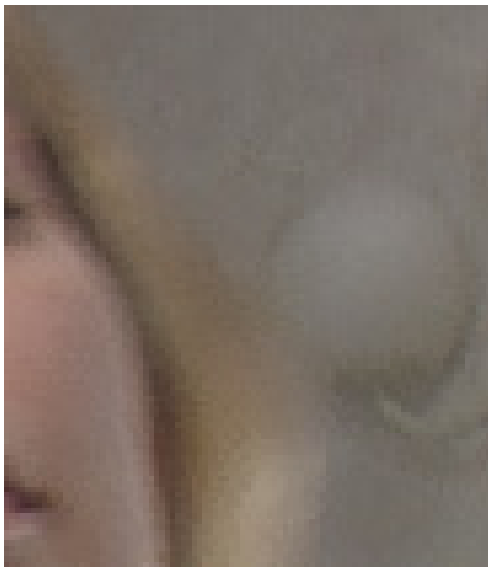
Алгоритм:

```
for (each pixel of the current video frame)
{
    GetRGB (source_pixel, r, g, b);
    tot_red = tot_green = tot_blue = 0;
    count = 0;
    r_tab = square_tab[255 - r];
    g_tab = square_tab[255 - g];
    b_tab = square_tab[255 - b];
    for (each pixel in the specified radius)
    {
        GetRGB (neighbour_pixel, r1, g1, b1);
        square_error = (r_tab[r1] + g_tab[g1] + b_tab[b1]) >>
                        Strength;
        if (square_error > 16) square_error =16;
        square_error = 16 - square_error ;
        tot_red += r1* square_error;
        tot_green += g1* square_error;
        tot_blue += b1* square_error;
        count += square_error;
    }
    destination_pixel = RGB (tot_red/count, tot_green / count ,
                              tot_blue / count);
}
```

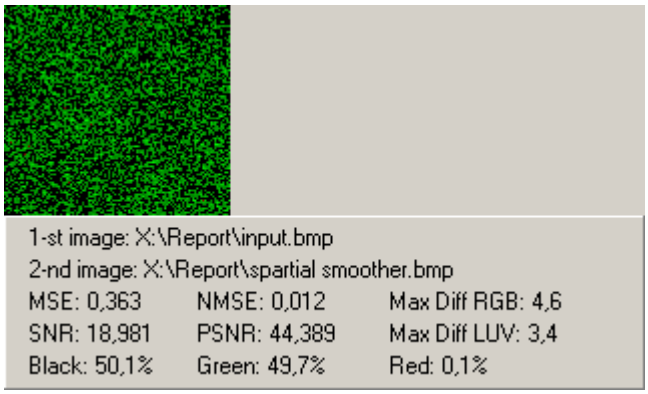
Результат работы:



Picture 4. Source image (susi.avi; 194 frame)



Picture 5. Processed image (default parameters)



Picture 6. Comparison of the source and processed images using LUV Metric

Temporal Filters

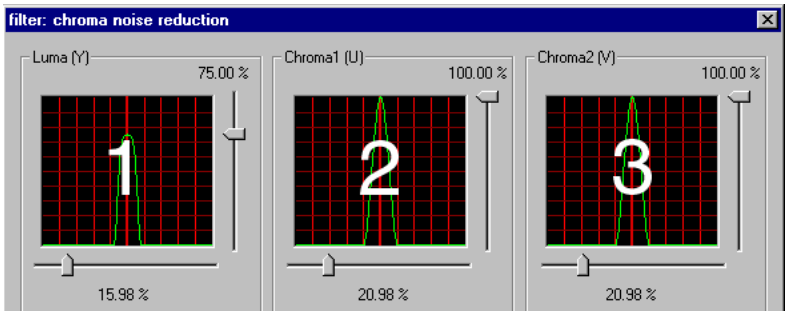
Если пространственные фильтры обрабатывают каждый кадр независимо от остальных, то временные обрабатывают каждый кадр на основе одного или нескольких предыдущих кадров.

Chroma Noise Reduction **by Gilles Mouchard**

Этот фильтр работает в цветовом пространстве YUV и изменяет только цветностные компоненты пикселей (U и V), не трогая яркостную (Y). Дело в том, что при передаче аналогового видеосигнала диапазон частот, используемый для передачи цветности, меньше, чем используемый для передачи яркости, поскольку человеческий глаз менее восприимчив к цветности, чем к яркости. Как следствие, цветность более восприимчива к шуму. Таким образом, этот фильтр

уменьшает уровень шума на цветностной компоненте, предполагая, что яркостная компонента заведомо имеет хорошее качество.

Выбор параметров работы фильтра:



Каждая кривая показывает, как комбинировать предыдущий кадр с текущим в зависимости от различий в яркостной и цветностных компонентах.

На оси X указывается максимальная разница (отдельно для каждой из Y,U,V компонент) между пикселем текущего кадра и соответствующим пикселем предыдущего кадра, при которой значение пикселя из предыдущего кадра оказывает влияние на значение пиксела из текущего (d()). По оси Y указывается db процентах степень этого влияния, в зависимости от того, как близки значения этих пикселов (q()).

Новые значения U и V компонент пиксела вычисляются по следушим формулам:

*U(new frame) = q(Y) * q(U) * U(previous frame) + (1 - q(Y)) * (1 - q(U)) * U(current frame)*

*V(new frame) = q(Y) * q(V) * V(previous frame) + (1 - q(Y)) * (1 - q(V)) * V(current frame)*

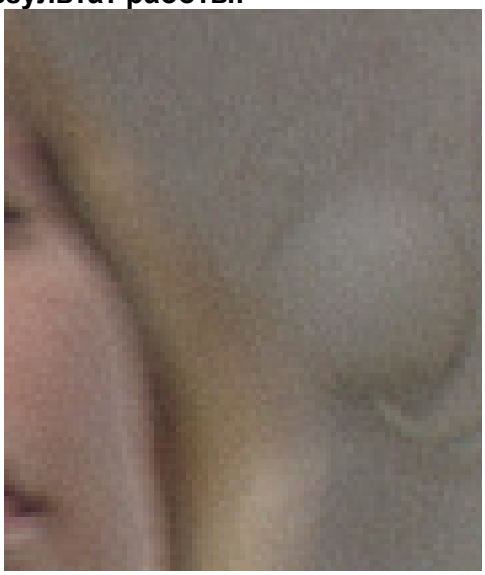
Алгоритм:

Вначале по заданным выше кривым формируются соответствующие таблицы для каждой из компонент (*U_tab, V_tab* are defined in a similar way)

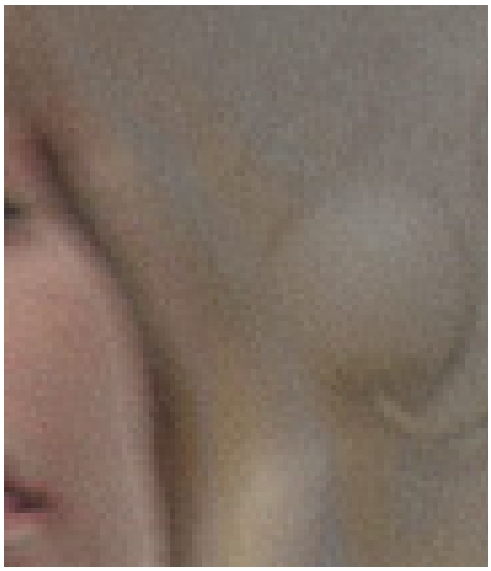
$$Y\_tab[i] = \begin{cases} 0, & i < -Y\_ymax,\ i > Y\_ymax \\ Y\_xmax/2*(1+\cos(pi*i/Y\_ymax)), & -Y\_ymax < i < -Y\_ymax \end{cases}$$

```
for (each pixel of the current video frame)
{
    (RGB) -> (YUV)
    Y_diff = Y - Y_prev;
    U_diff = U - U_prev;
    V_diff = V - V_prev;
    U_new = (Y_tab[Y_diff] * U_tab[U_diff] * U_prev + (1 -
                Y_tab[Y_diff]) * (1 - U_tab[U_diff]) * U);
    V_new = (Y_tab[Y_diff] * V_tab[V_diff] * V_prev + (1 -
                Y_tab[Y_diff]) * (1 - V_tab[V_diff]) * V);
    (YUV) -> (RGB)
}
```

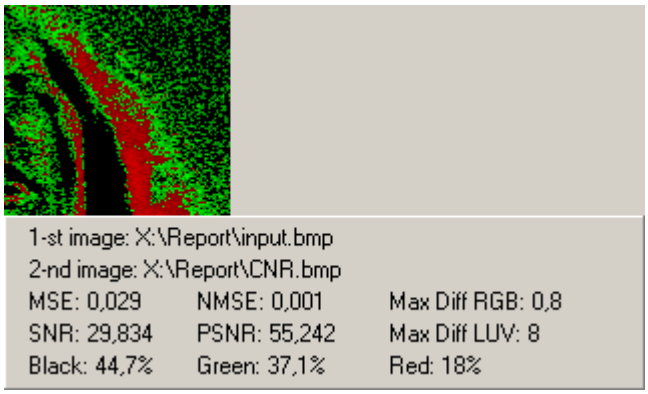
Результат работы:



Picture 7. Source image (susi.avi; 194 frame)



Picture 8. Processed image (default parameters)



Picture 9. Comparison of the source and processed images using LUV Metric

Dynamic Noise Reduction

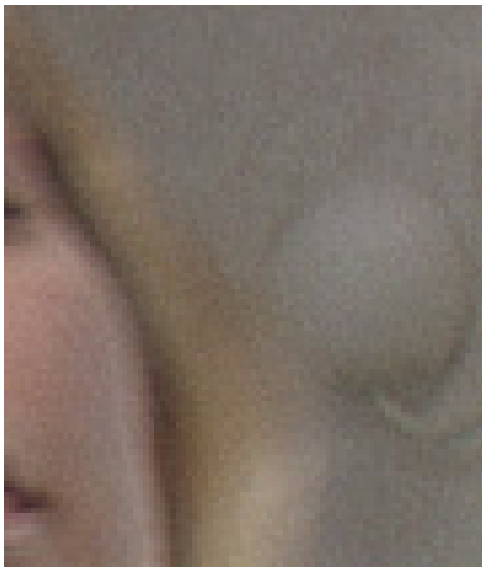
Принцип работы тот же, что и у предыдущего алгоритма – усреднение текущего кадра по предыдущему, только таблица формируется по-другому и обработка ведется уже не в YUV-пространстве, а в RGB. Единственным параметром этого фильтра является максимальная разница между соответствующими пикселями двух соседних кадров (Level).

Алгоритм:

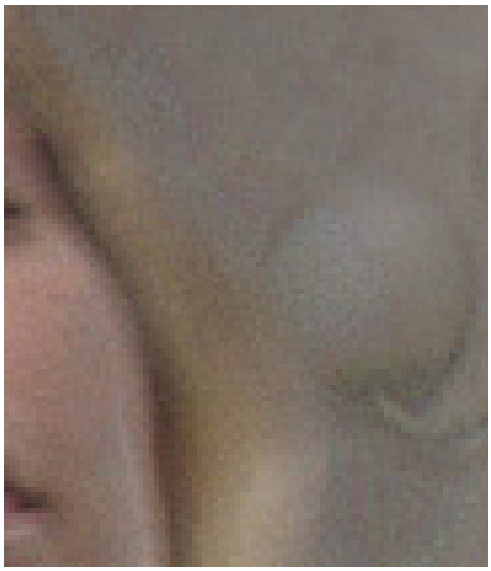
```
for (int index = 0; index < 766; index++)
    divTable[index] = index/3;

for (each pixel of the current video frame)
{
    GetRGB(source_pixel, r, g, b);
    r_diff = abs(r - r_prev);
    g_diff = abs(g - g_prev);
    b_diff = abs(b - b_prev);
    if (r_diff < Level)
    if  (r_diff > (Level >> 1))  r = divTable [2*r + r_prev];
        else r = r_prev;
    … (the same for each color component)
    destination_pixel = RGB (r, g, b);
}
```

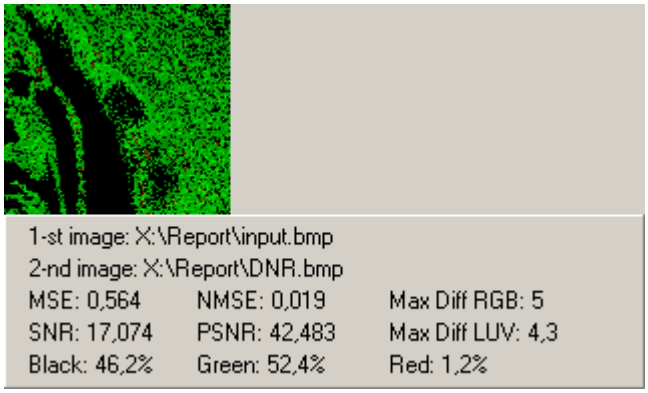
Результат работы:



Picture 10. Source image (susi.avi; 194 frame)



Picture 11. Processed image (default parameters)



Picture 12. Comparison of the source and processed images using LUV Metric

Действие фильтра при различных параметрах:

Picture 13. Processed image (susi.avi; 194 frame)

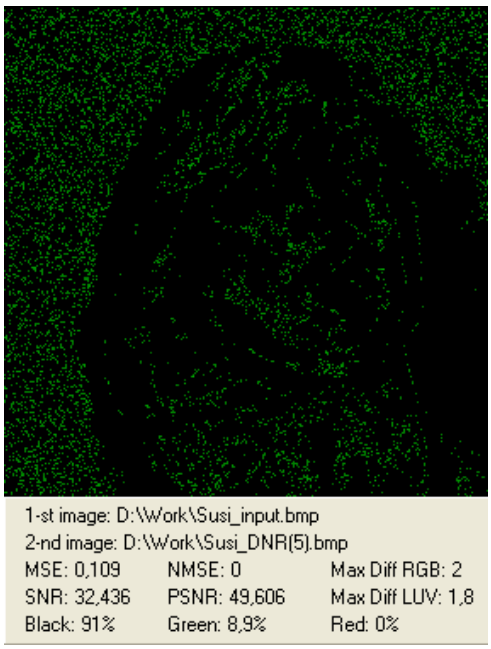
Picture 14. *threshold=5*

Слишком слабо убирает шум.

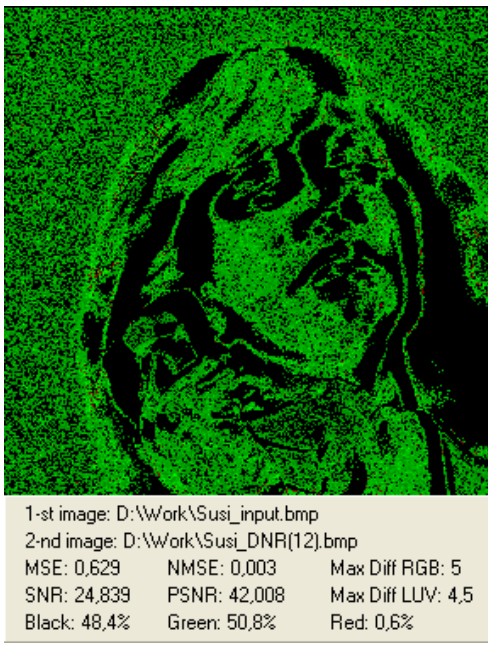
Picture 16. Processed image (susi.avi; 194 frame)

Picture 17. *threshold=12*

Наилучший параметр, хорошо убирает шум, не трогает границы.



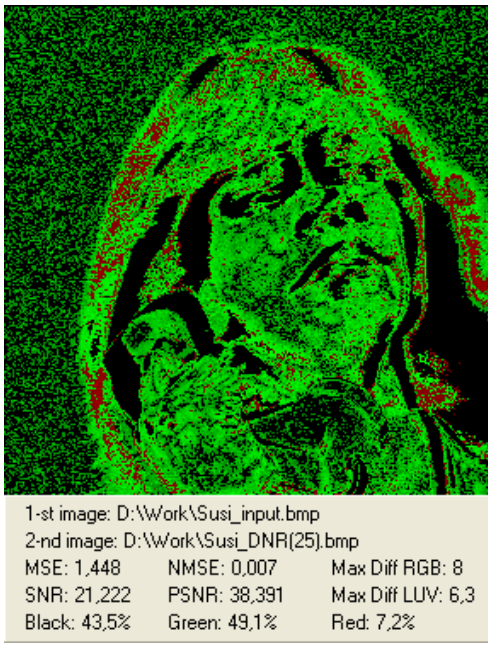
Picture 15. Comparison of the source and processed images using LUV Metric



Picture 18. Comparison of the source and processed images using LUV Metric

Picture 19. Processed image (susi.avi; 194 frame)

Picture 20. *threshold=25*



Picture 21. Comparison of the source and processed images using LUV Metric

Чересчур сильно изменяет картинку – изменения неприемлемы.

Other filters

KNRC

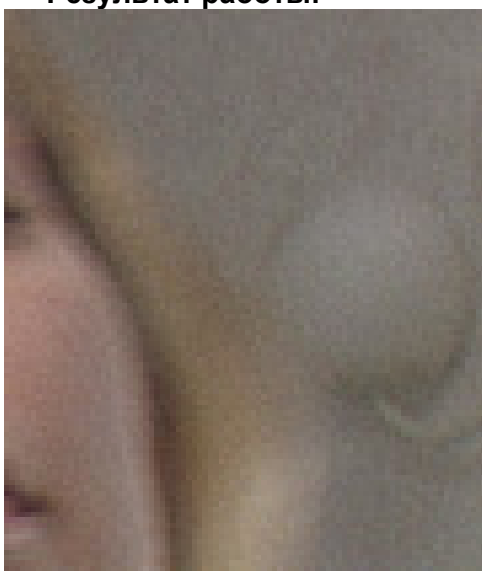
Фильтр kNRC разработан, чтобы уменьшать разницу между кадрами (для лучшей компрессии) и убирать шум рядом с краями.

Принцип его работы следующий:

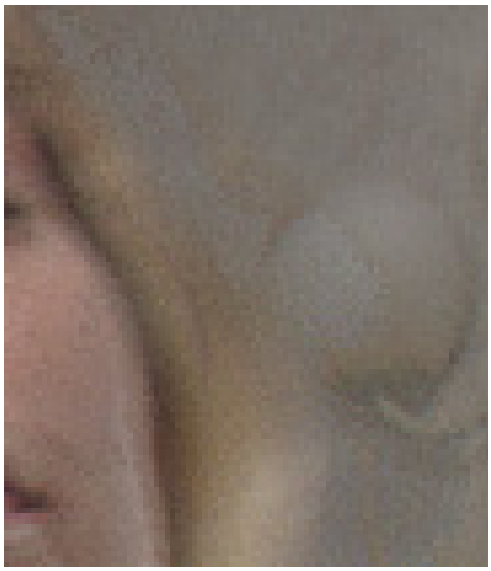
1. Старый и новый пиксели смешиваются.
2. Вычисляется яркость нового пикселя и определяется как порог.
3. Если модуль разности между смешанным пикселем и старым пикселем превышает порог, то используется неизмененный пиксель, иначе используется смешанный пиксель.

Важно то, как определяется порог из яркости нового пикселя. Человеческий глаз наиболее чувствителен в темных областях, поэтому автор использует различные пороги в соответствии с яркостью пикселей. Также так как человеческий глаз чувствителен к изменениям яркости логарифмически, автор разбивает ряд яркостей на 8 частей и распределяет порог между этими частями.

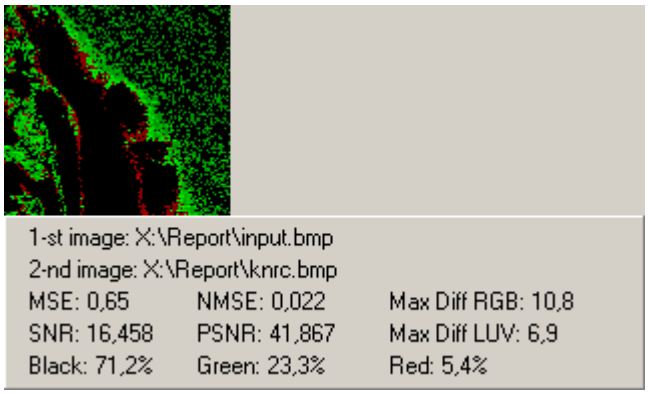
Результат работы:



Picture 22. Source image (susi.avi; 194 frame)



Picture 23. Processed image (default parameters)

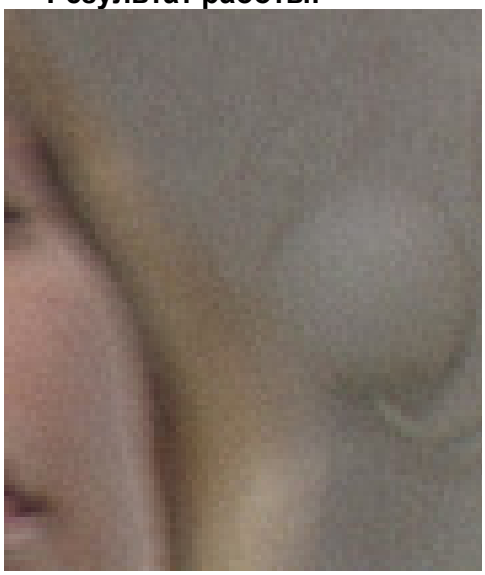


Picture 24. Comparison of the source and processed images using LUV Metric

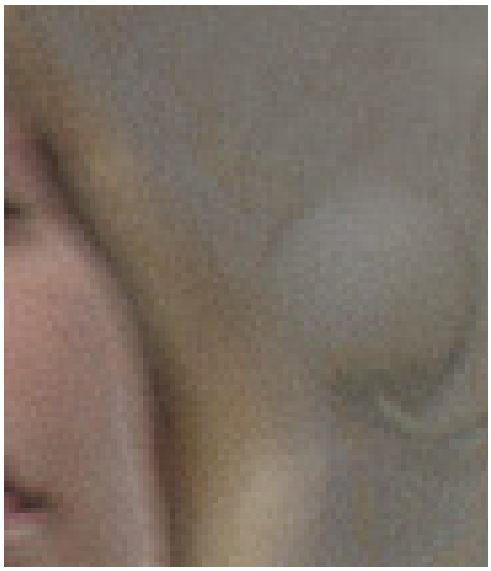
VHS

Убирает шумы примерно так же, как и фильтр kNRC. Принцип работы неизвестен, так как к данному фильтру не было найдено исходников.

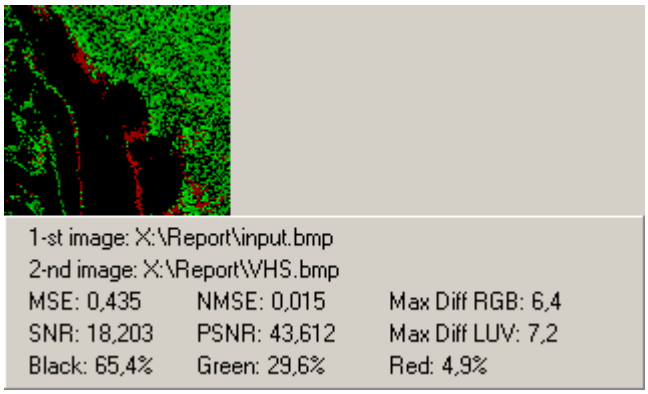
Результат работы:



Picture 25. Source image (susi.avi; 194 frame)



Picture 26. Processed image (default parameters)

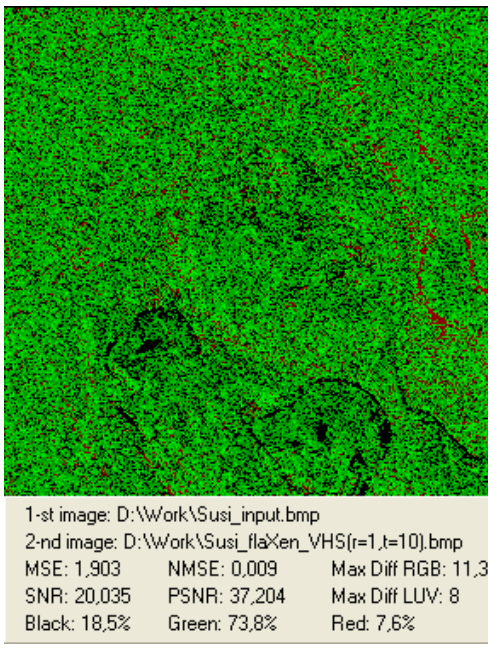


Picture 27. Comparison of the source and processed images using LUV Metric

Действие фильтра при различных параметрах:

Picture 28. Processed image (susi.avi; 194 frame)

Picture 29. *radius=1, threshold=10*

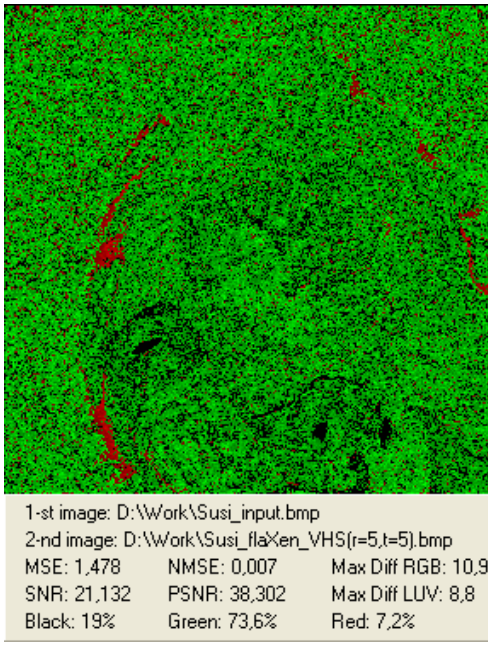


Picture 30. Comparison of the source and processed images using LUV Metric

Наилучший параметр – хорошо убирает шум, минимальное количество неприемлемых изменений.

Picture 31. Processed image (susi.avi; 194 frame)

Picture 32. *radius=5, threshold=5*

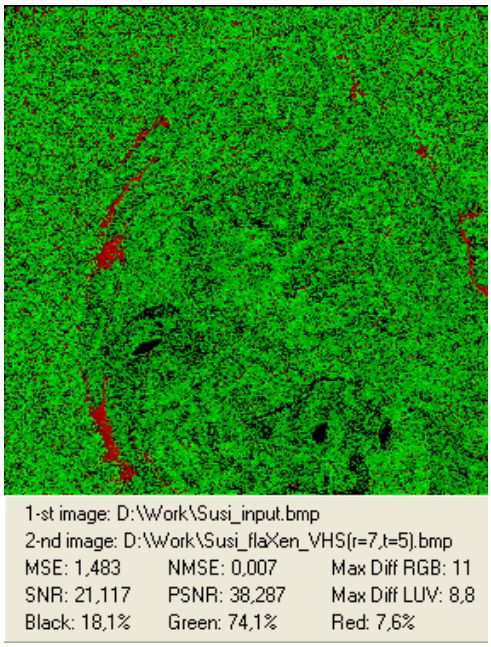


Picture 33. Comparison of the source and processed images using LUV Metric

Много сильных изменений, которые ухудшают изображение.

Picture 34. Processed image (susi.avi; 194 frame)

Picture 35. *radius=7, threshold=5*



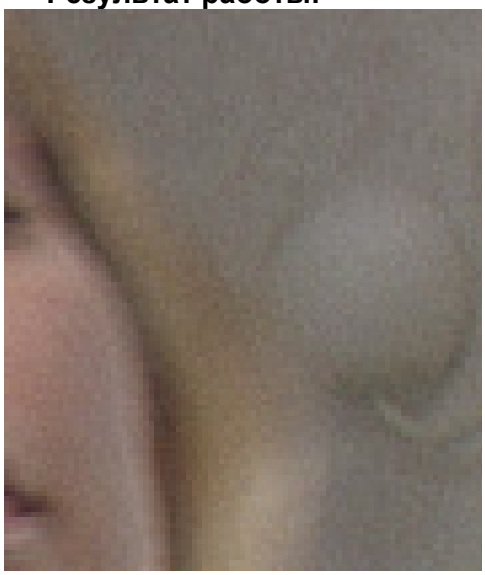
Picture 36. Comparison of the source and processed images using LUV Metric

Много неприемлемых изменений.

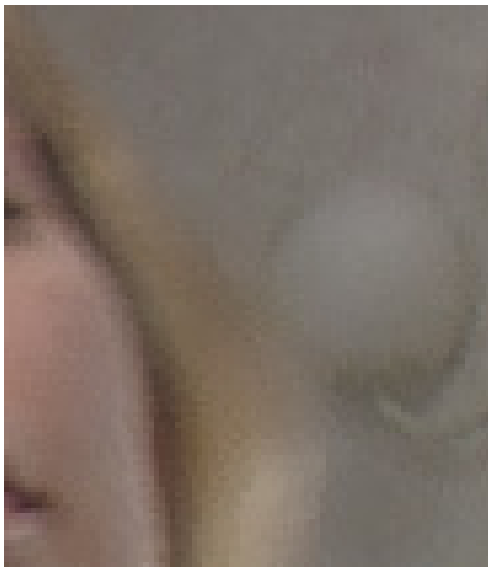
Smart Smoother Filter **(Version 1.1)**

Этот фильтр производит сглаживание, которое очень удачно устраняет видео шум и вместе с тем блочные артефакты, которые возникают при MPG/JPG сжатии. Отличительной особенностью этого фильтра является то, что он практически не размывает структуру картинки (края, углы и т.д.), он до некоторой степени их даже делает резче.

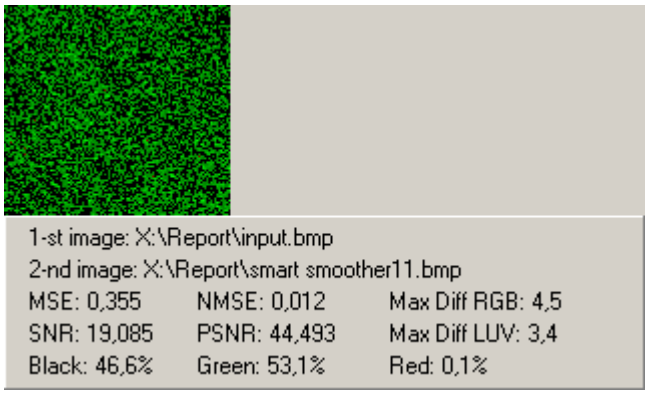
Результат работы:



Picture 37. Source image (susi.avi; 194 frame)



Picture 38. Processed image (default parameters)



Picture 39. Comparison of the source and processed images using LUV Metric

Smart Smoother Filter (Version 2.11)

Эта реализация была сделана как добавление к оригинальному фильтру Smart Smoother для выполнения лучших результатов на обычном видео.

Принцип работы фильтра:

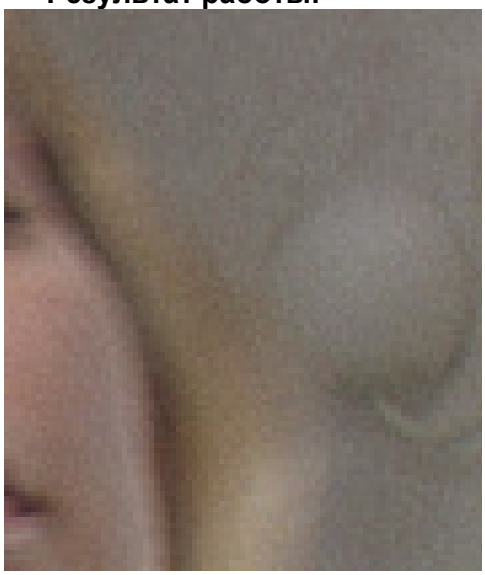
Фильтр работает в двух режимах, которые выполняют одну и ту же задачу. Для начала необходимо задать диаметр, на котором будет работать фильтр. Более большой диаметр - более большая площадь может быть размыта. Обычно чем больше диаметр - тем лучше качество, но меньше скорость. Наиболее употребимый 5-7.

Также необходимо задать порог. Порог показывает на сколько должны быть близки цвета пикселей, чтобы их смешать. Если разница в цвете пикселей будет больше порога, то они не будут влиять друг на друга.

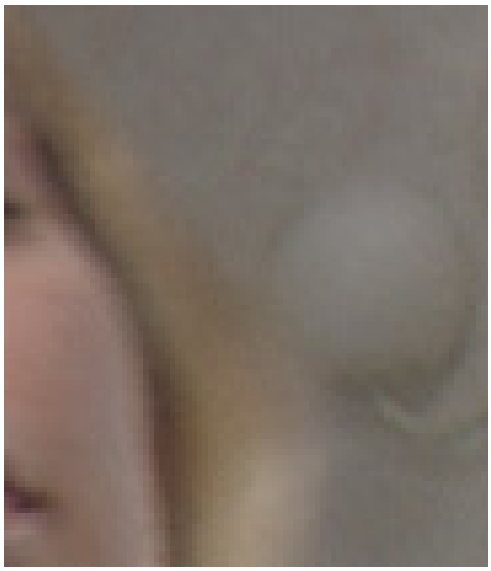
Фильтр обрабатывает пиксели в ромбовидной форме, чтобы избежать размытия областей, которые не соединены.

К тому же пиксели оцениваются по расстоянию до текущего пикселя (в Weighed average mode). Чем дальше пиксель от текущего пикселя, тем меньше влияние он на него оказывает.

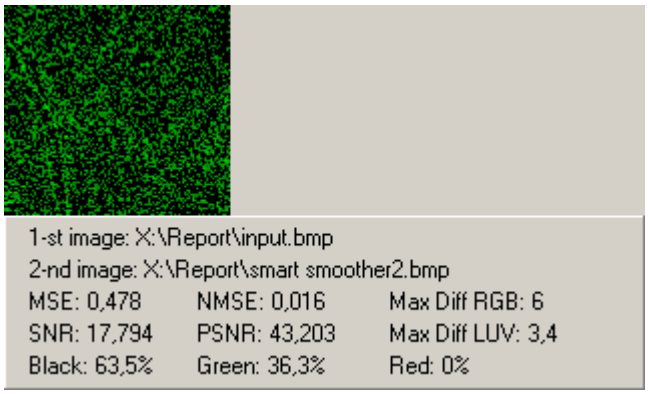
Результат работы:



Picture 40. Source image (susi.avi; 194 frame)



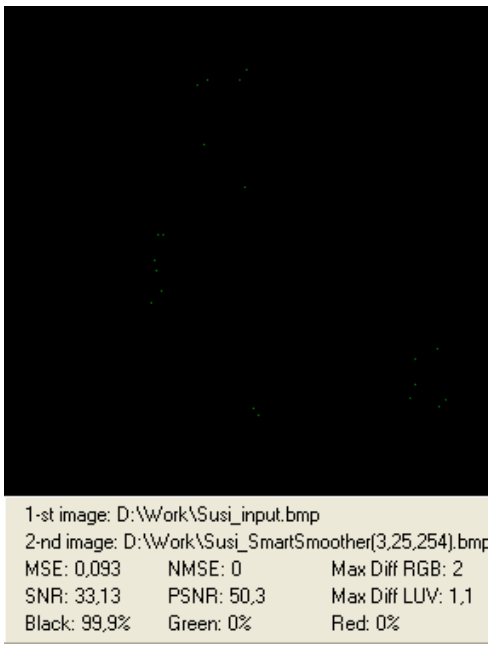
Picture 41. Processed image (default parameters)



Picture 42. Comparison of the source and processed images using LUV Metric

Действие фильтра при различных параметрах:

Picture 43. Processed image (susi.avi; 194 frame)

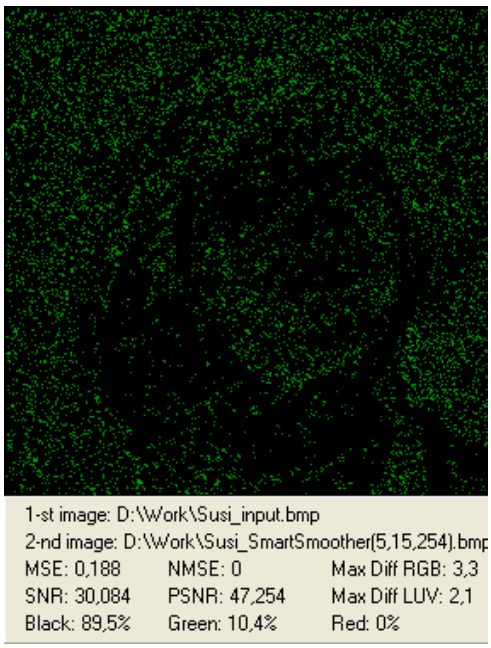


Picture 45. Comparison of the source and processed images using LUV Metric

Picture 44. *diameter=3, threshold=25*

Практически не убирает шум.

Picture 46. Processed image (susi.avi; 194 frame)

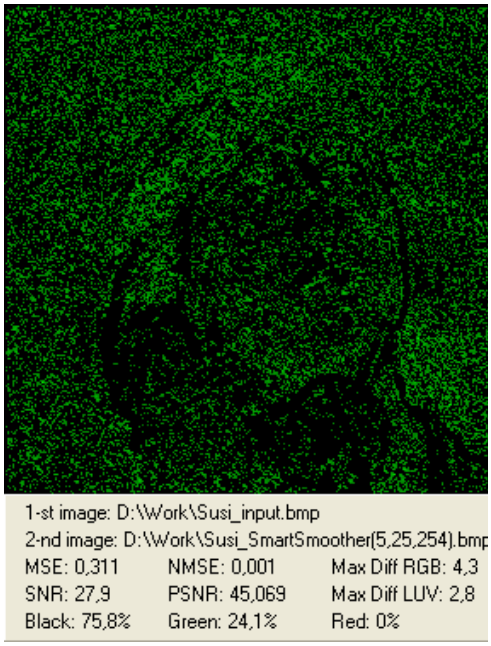


Picture 48. Comparison of the source and processed images using LUV Metric

Picture 47. *diameter=5, threshold=15*

Слабо убирает шум.

Picture 49. Processed image (susi.avi; 194 frame)

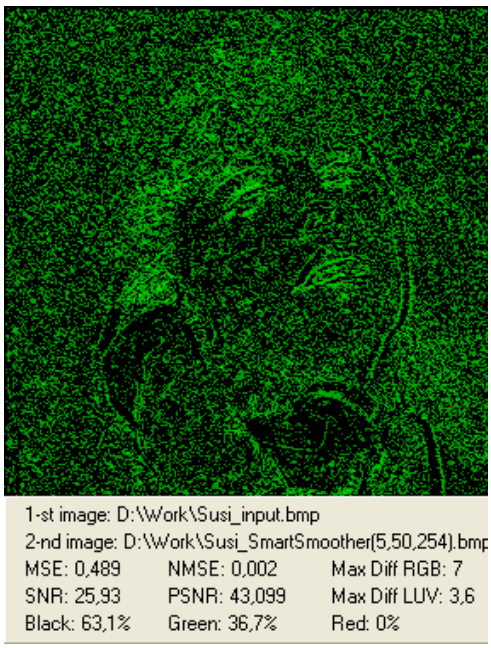


Picture 51. Comparison of the source and processed images using LUV Metric

Picture 50. *diameter=5, threshold=25*

Хорошо, но тем не менее слабо убирает шум.

Picture 52. Processed image (susi.avi; 194 frame)



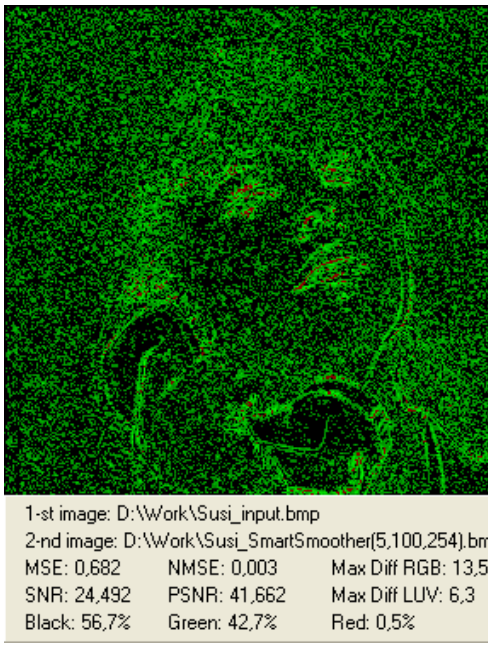
Picture 54. Comparison of the source and processed images using LUV Metric

Picture 53. *diameter=5, threshold=50*

Наилучший параметр – сильно убирает шум, нет неприемлемых изменений.

Picture 55. Processed image (susi.avi; 194 frame)

Picture 56. *diameter=5, threshold=100*

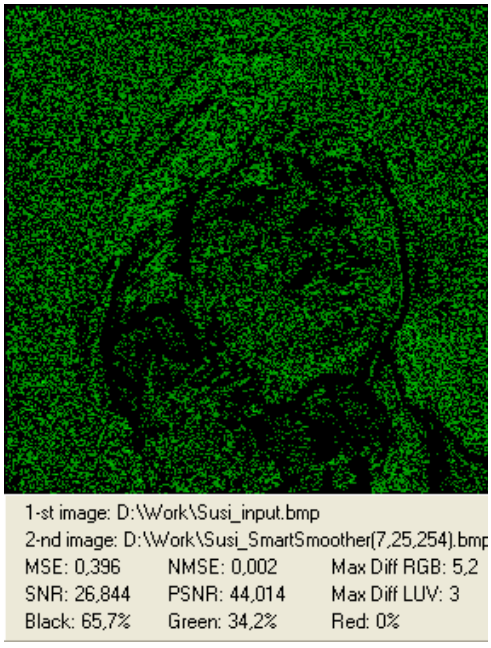


Picture 57. Comparison of the source and processed images using LUV Metric

Появляются неприемлемые изменения.

Picture 58. Processed image (susi.avi; 194 frame)

Picture 59. *diameter=7, threshold=25*



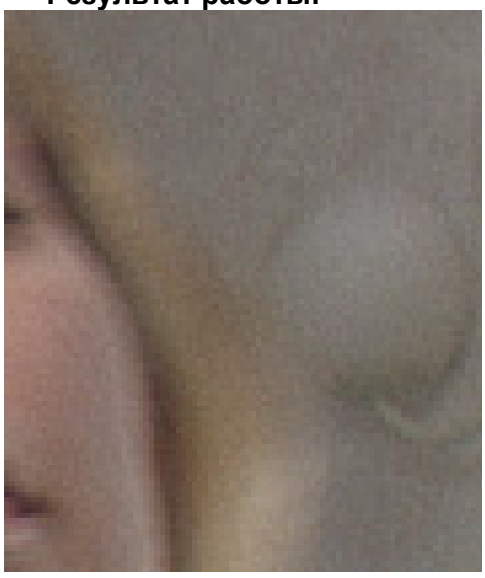
Picture 60. Comparison of the source and processed images using LUV Metric

Слабо убирает шум.

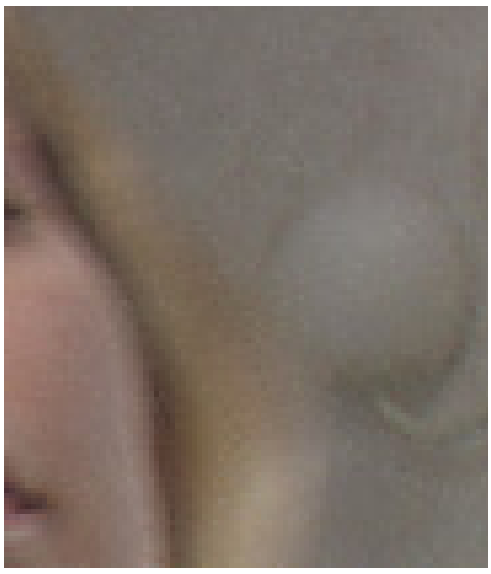
Static Noise Reduction Filter

Уменьшает случайный шум, сохраняя при этом контрастность, что помогает MPEG-сжатию и слегка улучшает качество изображения. (Без исходников)

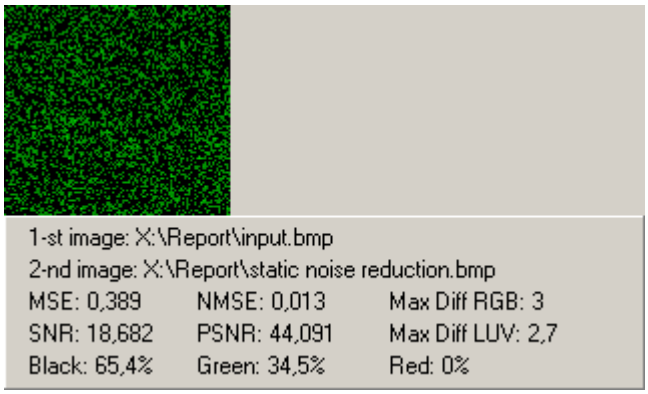
Результат работы:



Picture 61. Source image (susi.avi; 194 frame)



Picture 62. Processed image (default parameters)



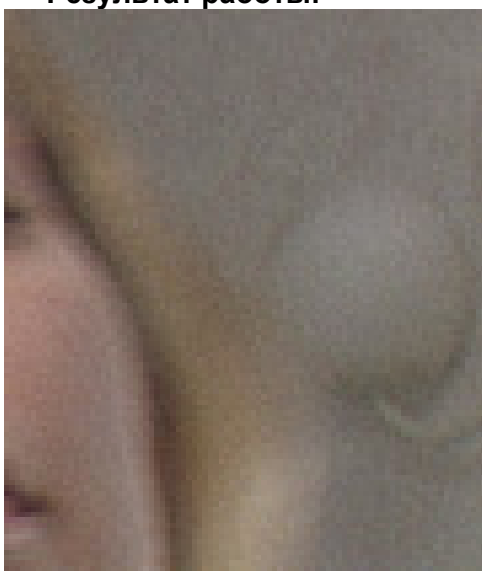
Picture 63. Comparison of the source and processed images using LUV Metric

Video DeNoise Filter

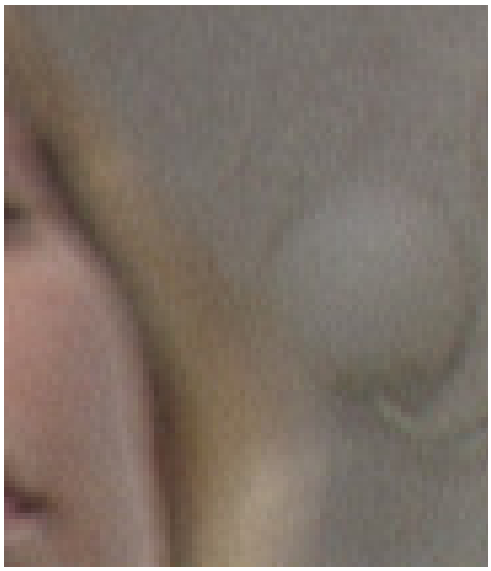
Основная задача этого фильтра – убрать шум, представленный маленькими полосками красного или синего, реже зеленого, цвета, который всегда присутствует на изображении с видеоленты.

Фильтр анализирует каждый кадр независимо от остальных и, как утверждает автор этого фильтра, определяет шум с высокой точностью. Уровень удаления шума регулируется для каждой цветовой компоненты отдельно. (Без исходников)

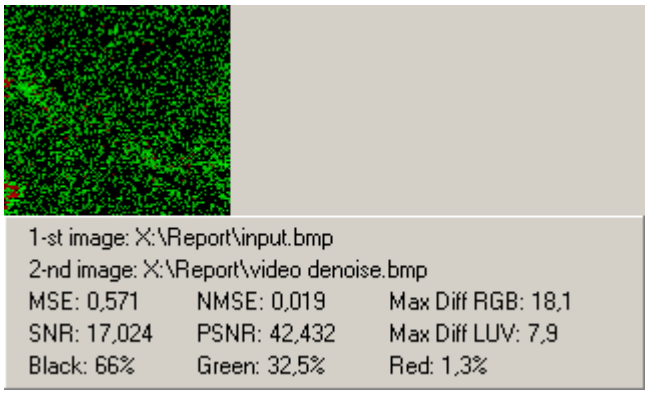
Результат работы:



Picture 64. Source image (susi.avi; 194 frame)



Picture 65. Processed image (default parameters)

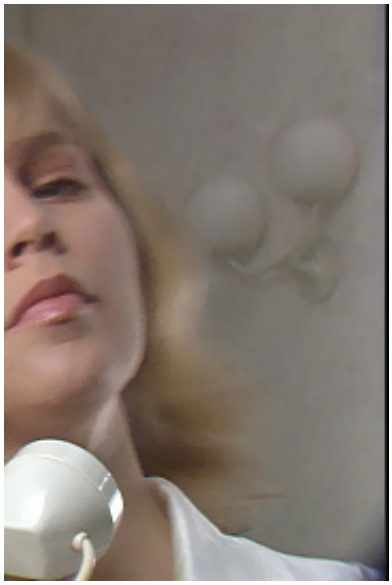


Picture 66. Comparison of the source and processed images using LUV Metric

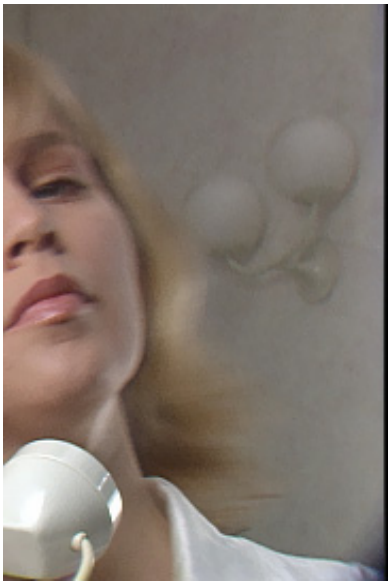
Сравнение существующих фильтров для шумоподавления на различных тестовых последовательностях.

Susi_resize.avi frame number 194

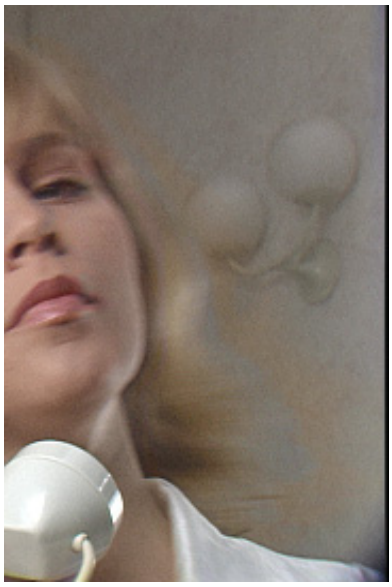
Picture 67. Input Image



Picture 68. 2d Cleaner
Threshold = 10, Area = 5x5

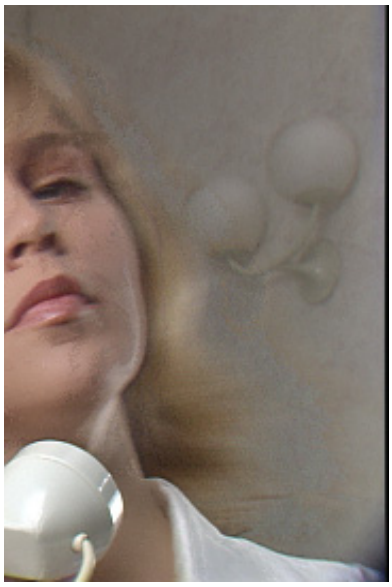


Picture 69. Spatial Smoother
Diameter = 5, Strenght = 3



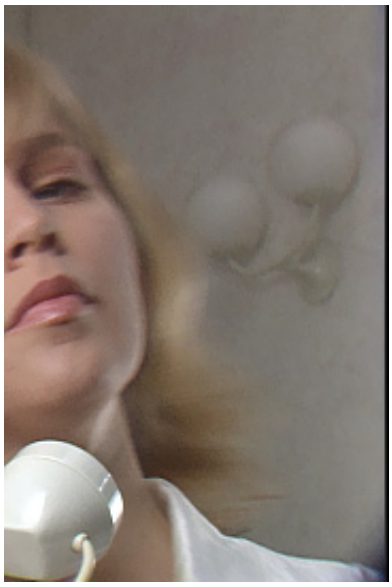
Picture 70. Chroma Noise Reduction

**Picture 71. Dynamic Noise Reduction
Threshold = 12**

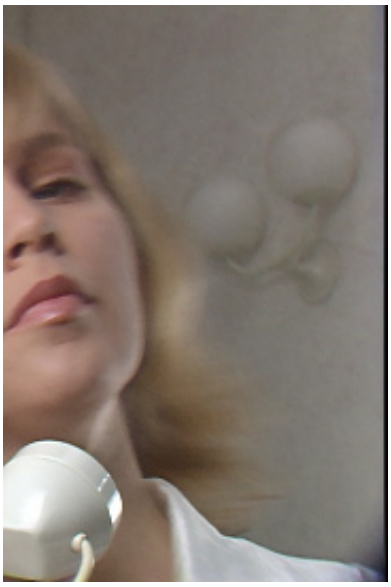


**Picture 72. kNRC
Threshold: DarkPixels=4, Light Pixels=12**

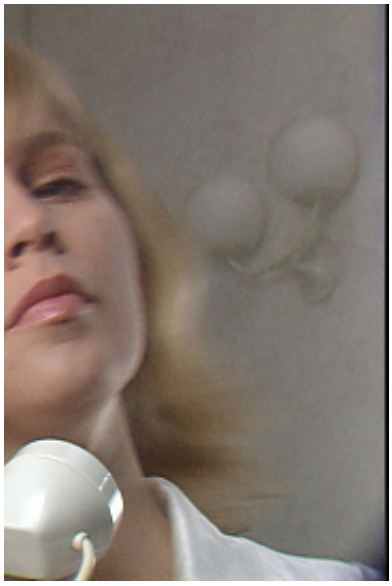
**Picture 73. VHS
Pre-filter, Post-filter, Radius=1,Threshold=10**



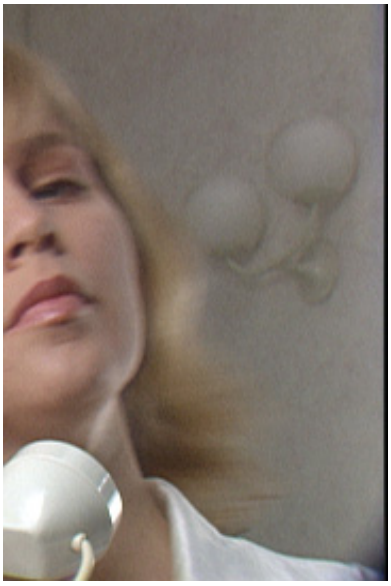
Picture 74. Smart Smoother 1.1
Diameter = 5, Strength =25



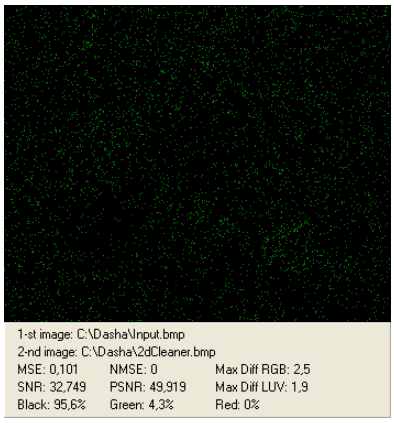
Picture 75. Smart Smoother 2.11
Diameter = 5, Threshold = 50



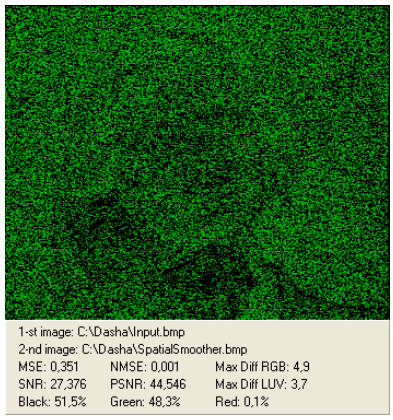
Picture 76. Static Noise Reduction 1.2
Threshold = 6



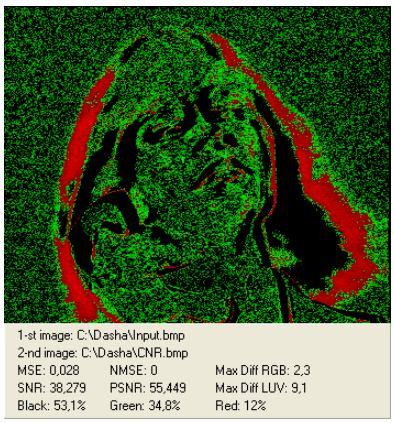
Picture 77. Video Denoise



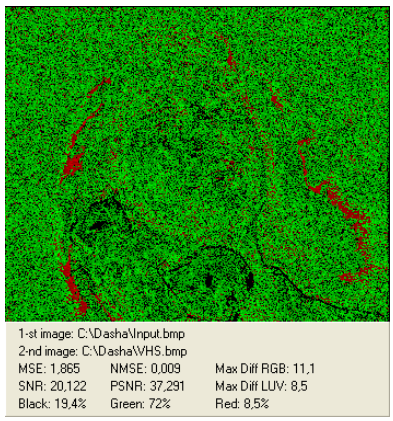
Picture 78. 2d Cleaner Threshold = 10, Area = 5x5



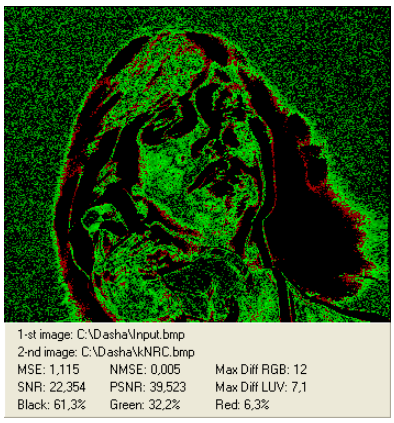
Picture 79. Spatial Smoother Diameter = 5, Strenght = 3



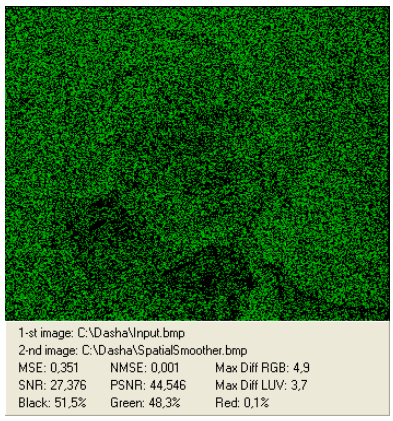
Picture 80. Chroma Noise Reduction



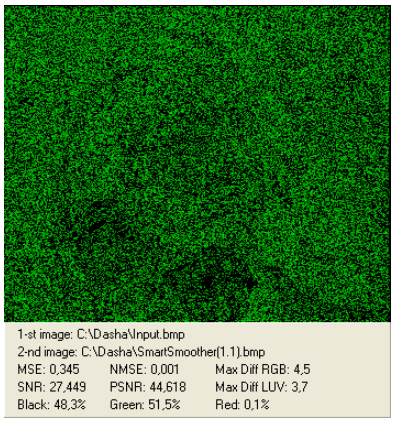
Picture 81. Dynamic Noise Reduction Threshold = 12



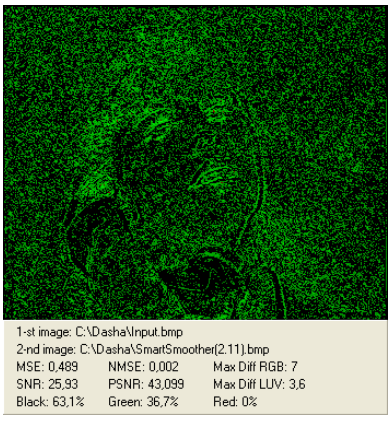
Picture 82. kNRC Threshold: DarkPixels=4, Light Pixels=12



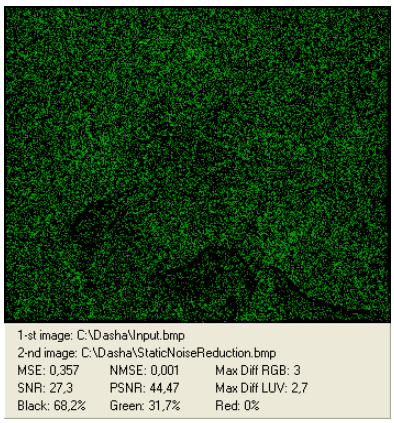
Picture 83. VHS Pre-filter, Post-filter, Radius = 1,Threshold=10



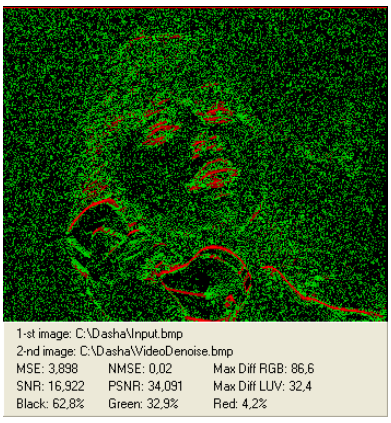
Picture 84. Smart Smoother 1.1 Diameter = 5, Strength =25



Picture 85. Smart Smoother 2.11 Diameter = 5, Threshold = 50



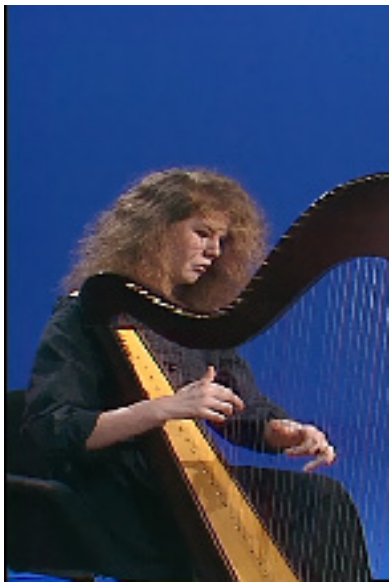
Picture 86. Static Noise Reduction 1.2 Threshold = 6



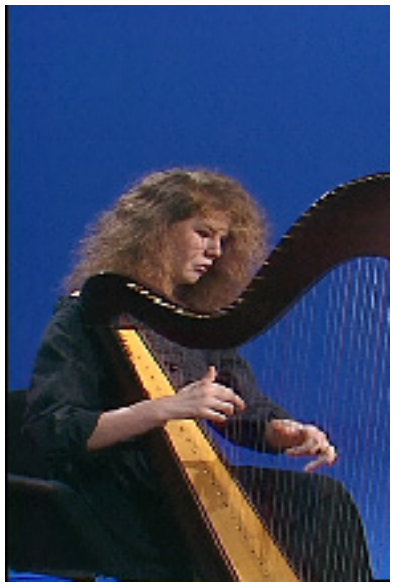
Picture 87. Video Denoise

src3_ref_625_resize.avi frame number 70

Picture 88. Input image



Picture 89. 2d Cleaner
Threshold = 10, Area = 5x5

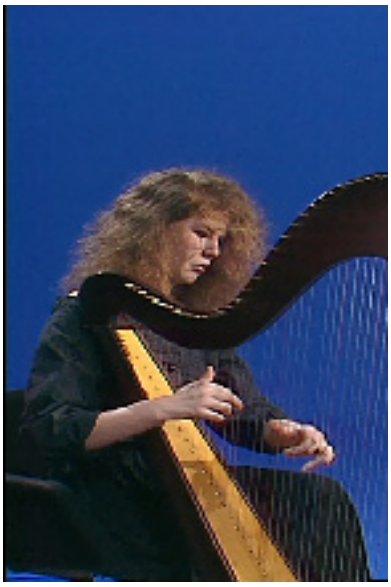


Picture 90. Spatial Smoother
Diameter = 5, Strenght = 3

Picture 91. Chroma Noise Reduction

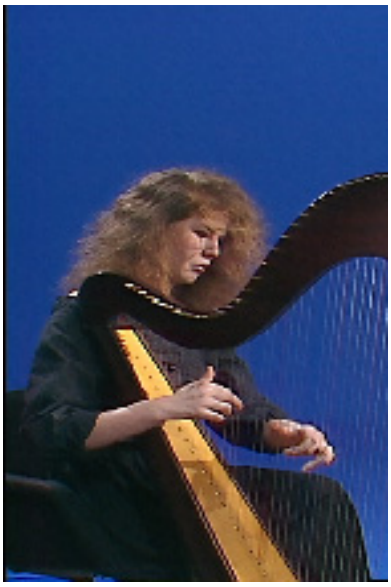
Picture 92. Dynamic Noise Reduction
Threshold = 12

Picture 93. kNRC
Threshold: DarkPixels=4, Light Pixels=12



Picture 94. VHS
Pre-filter, Post-filter, Radius=1,Threshold=10

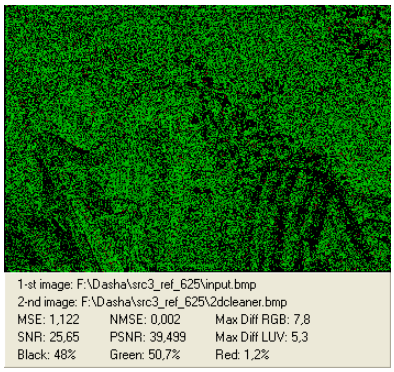
Picture 95. Smart Smoother 1.1
Diameter = 5, Strength =25



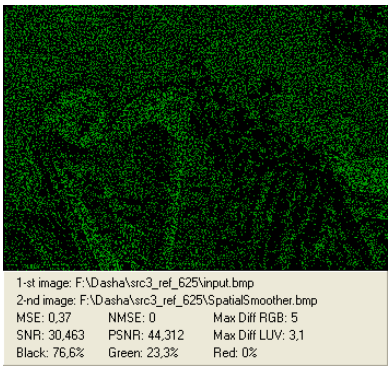
Picture 96. Smart Smoother 2.11
Diameter = 5, Threshold = 50

Picture 97. Static Noise Reduction 1.2
Threshold = 6

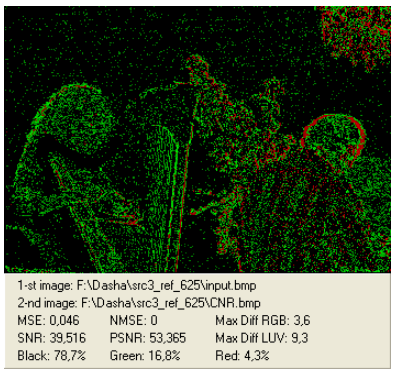
Picture 98. Video Denoise



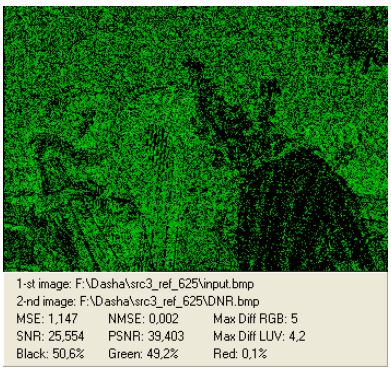
Picture 99. 2d Cleaner Threshold = 10, Area = 5x5



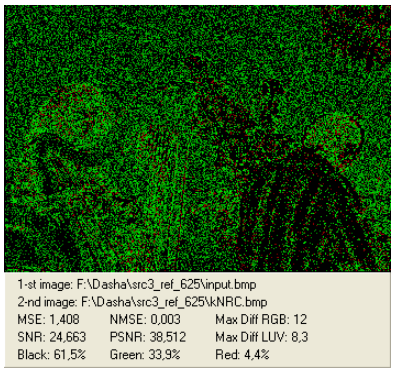
Picture 100. Spatial Smoother Diameter = 5, Strength = 3



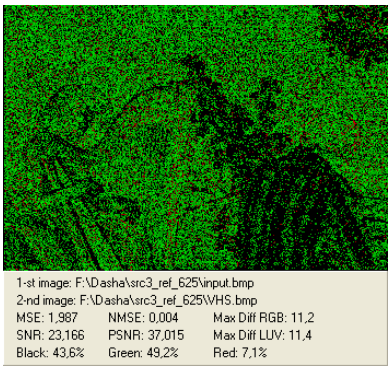
Picture 101. Chroma Noise Reduction



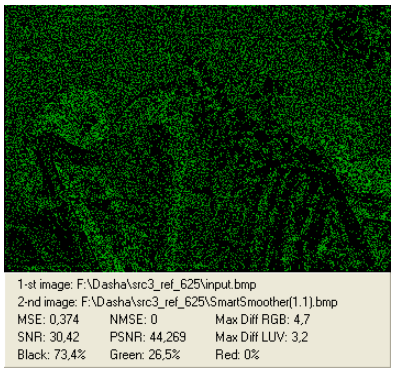
Picture 102. Dynamic Noise Reduction Threshold = 12



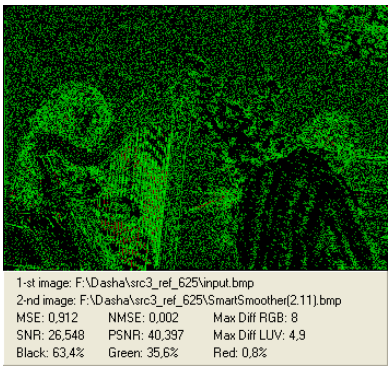
Picture 103. kNRC Threshold: Dark Pixels=4, Light Pixels=12



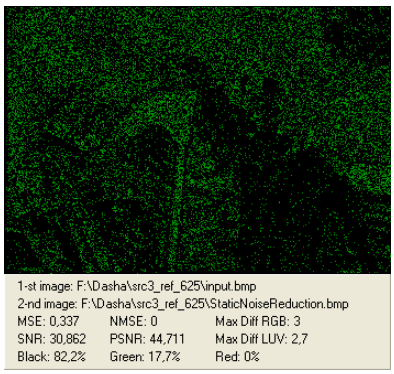
Picture 104. VHS Pre-filter, Post-filter, Radius=1,Threshold=10



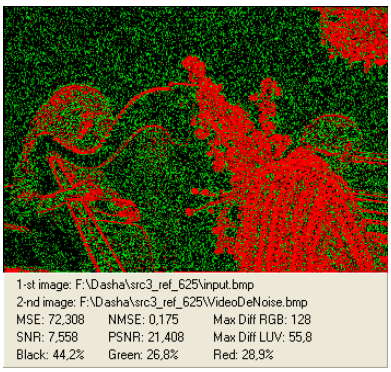
Picture 105. Smart Smoother 1.1
Diameter = 5, Strength =25



Picture 106. Smart Smoother 2.11
Diameter = 5, Threshold = 50

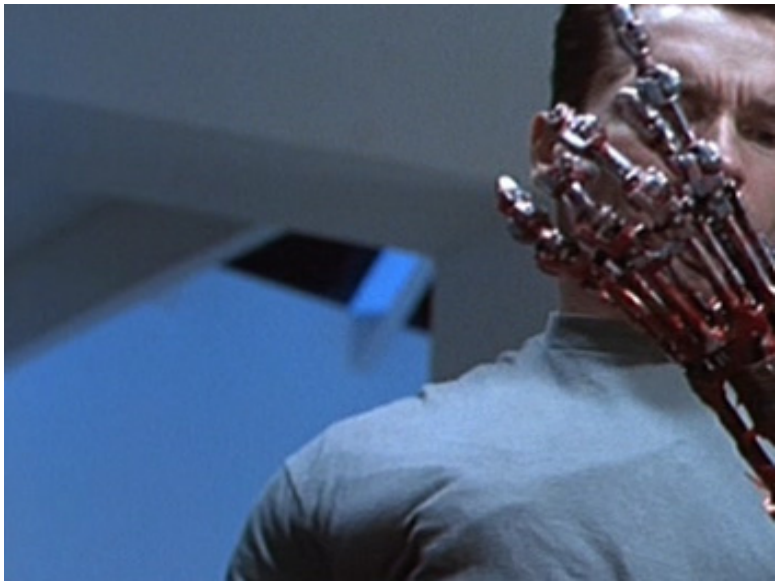


Picture 107. Static Noise Reduction 1.2
Threshold = 6



Picture 108. Video Denoise

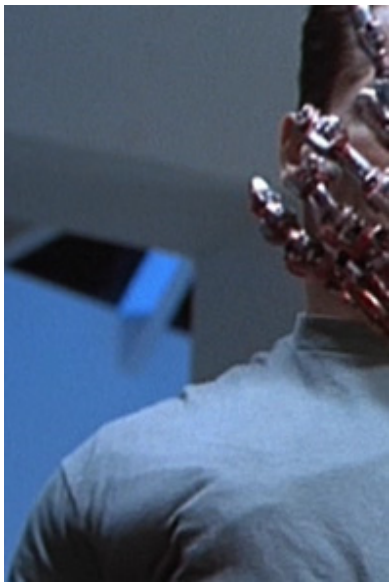
ruka.avi frame number 810



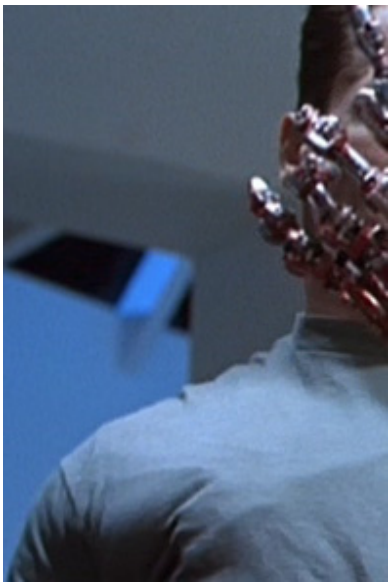
Picture 109. Input image

Picture 110. 2d Cleaner
Threshold = 10, Area = 5x5

Picture 111. Spatial Smoother
Diameter = 5, Strenght = 3



Picture 112. Chroma Noise Reduction



**Picture 113. Dynamic Noise Reduction
Threshold = 12**

**Picture 114. kNRC
Threshold: Dark Pixels=4, Light Pixels=12**

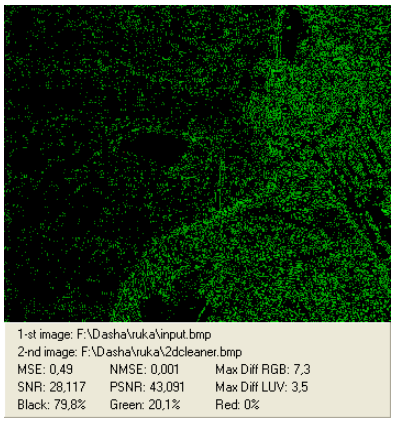
**Picture 115. VHS
Pre-filter, Post-filter, Radius=1,Threshold=10**

Picture 116. Smart Smoother 1.1
Diameter = 5, Strength =25

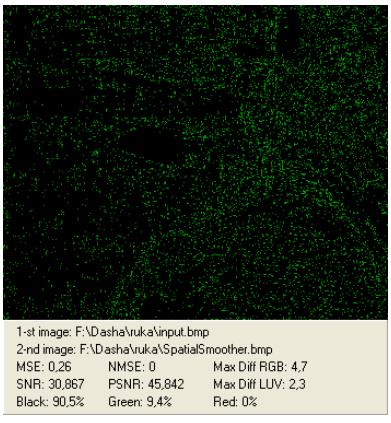
Picture 117. Smart Smoother 2.11
Diameter = 5, Threshold = 50

Picture 118. Static Noise Reduction 1.2
Threshold = 6

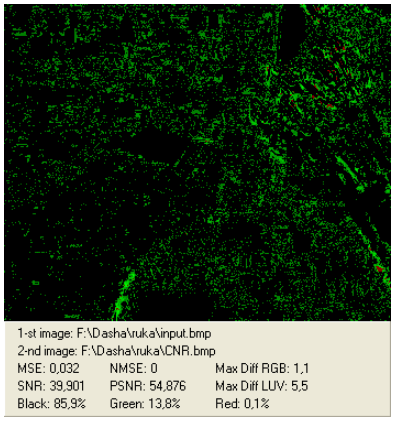
Picture 119. Video Denoise



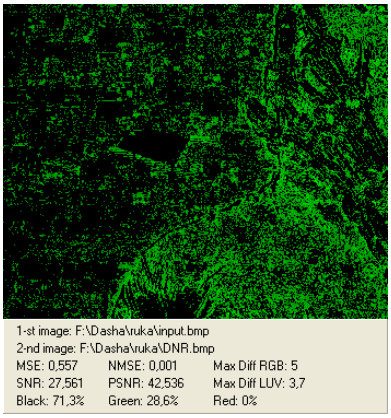
Picture 120. 2d Cleaner Threshold = 10, Area = 5x5



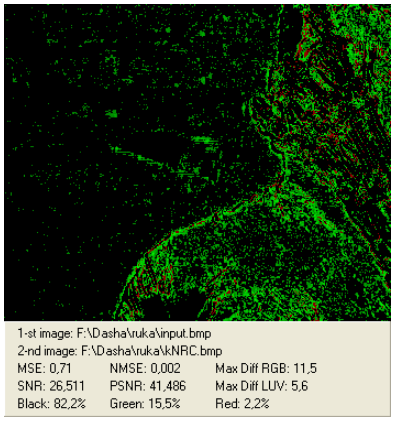
Picture 121. Spatial Smoother Diameter = 5, Strenght = 3



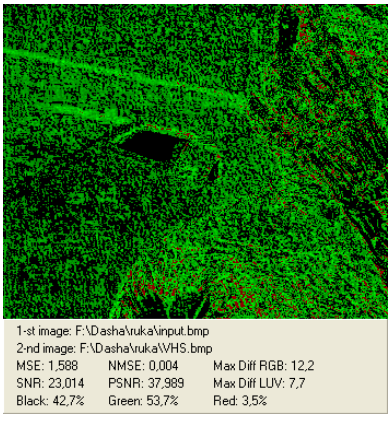
Picture 122. Chroma Noise Reduction



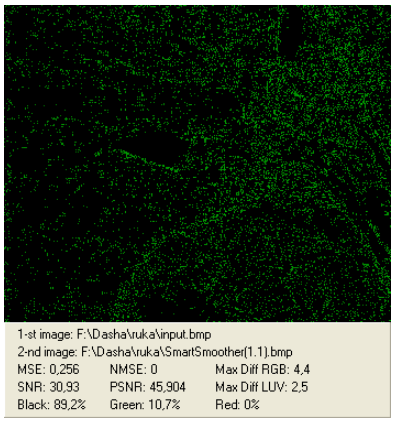
Picture 123. Dynamic Noise Reduction Threshold = 12



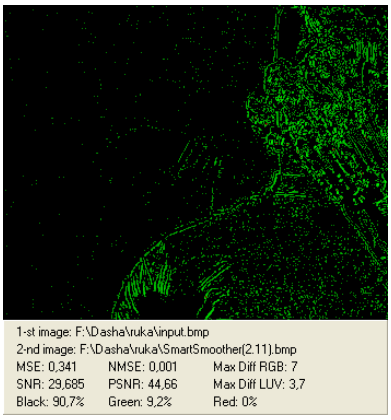
Picture 124. kNRC Threshold: Dark Pixels=4, Light Pixels=12



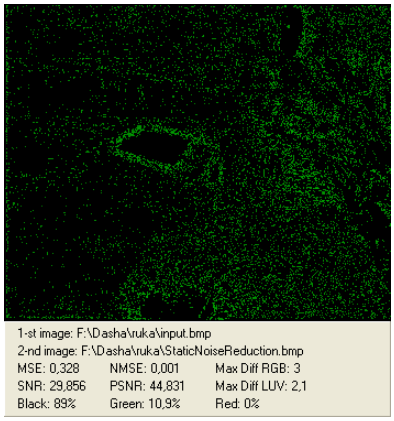
Picture 125. VHS Pre-filter, Post-filter, Radius=1,Threshold=10



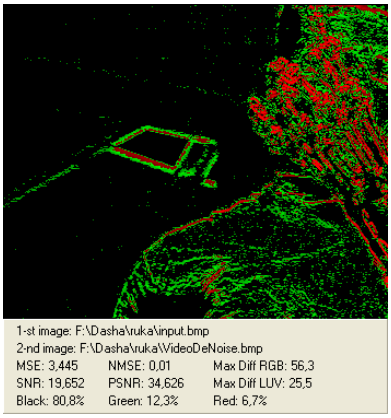
Picture 126. Smart Smoother 1.1
Diameter = 5, Strength =25



Picture 127. Smart Smoother 2.11
Diameter = 5, Threshold = 50



Picture 128. Static Noise Reduction 1.2
Threshold = 6



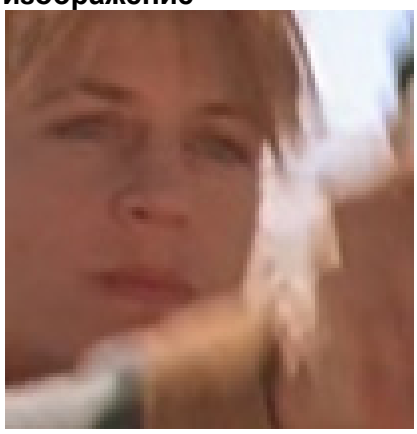
Picture 129. Video Denoise

Сравнение лучших фильтров при лучших параметрах

Отрывок из фильма Терминатор (rancho.avi) Кадр 236

Исходное изображение

Исходное изображение



Smart Smoother 2.11

Dynamic Noise Reduction

VHS

MSU denoising

More about last MSU method: http://compression.ru/video/denoising/