Università degli studi di Brescia
Dipartimento di ingegneria dell'informazione

Image processing and visualization course
Professors: Nicola Adami and Sergio Benini

Automatic blurred/non-blurred regions identification and image matting

Alessandro Gnutti
Andrea Sanzogni

Overview

- First part: automatic approach for blur/non-blur region identification
  - “Automatic blur region segmentation approach using image matting”. Jufeng Zhao, Huajun Feng, Zhihai Xu, Qi Li and Xiaoping Tao
- Second part: estimate of opacity value for each pixel in the image
  - “An iterative optimization approach for unified image segmentation and matting”. Jue Wang and Micheal F. Cohen

Blur region detection

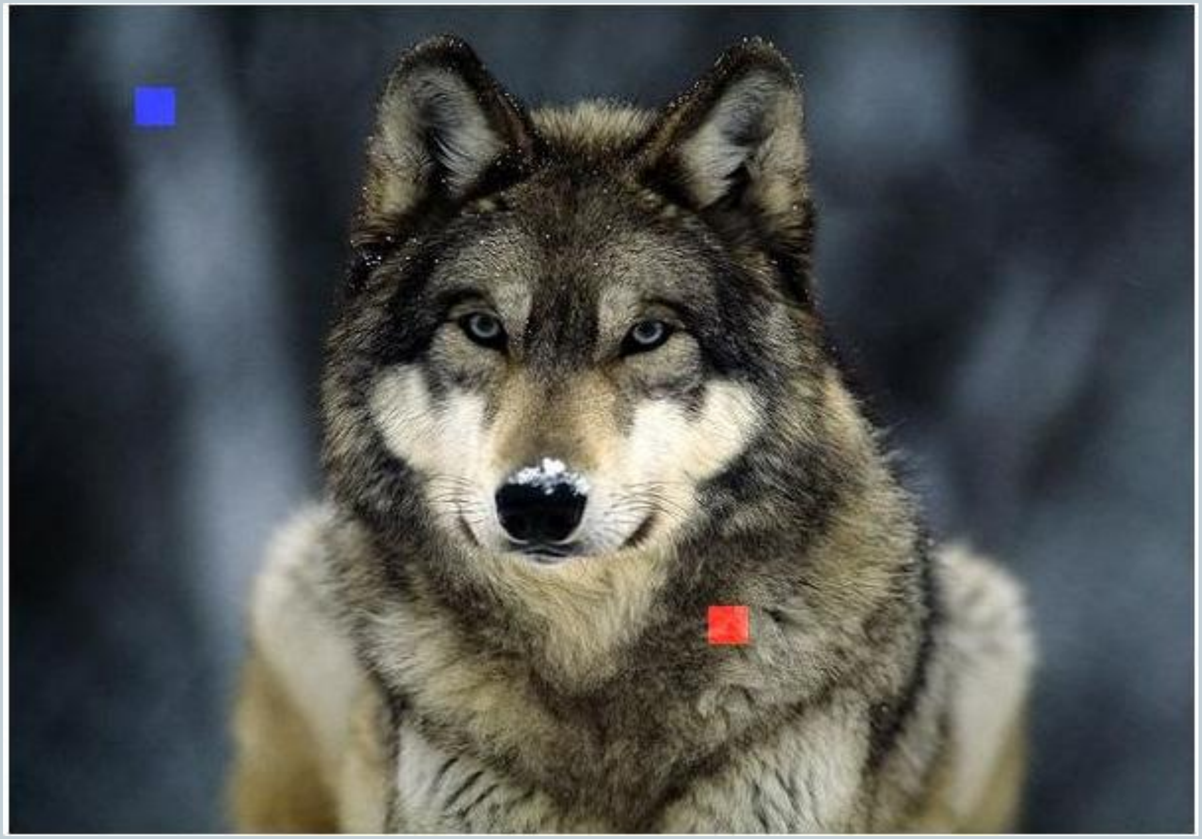
- We will quantify three blur features <u>for every patch</u> in the image:
  - Gradient histogram span
  - Local mean square error map
  - Maximum saturation

- At last we will combine the three features in order to create a more correct blur/non-blur mask

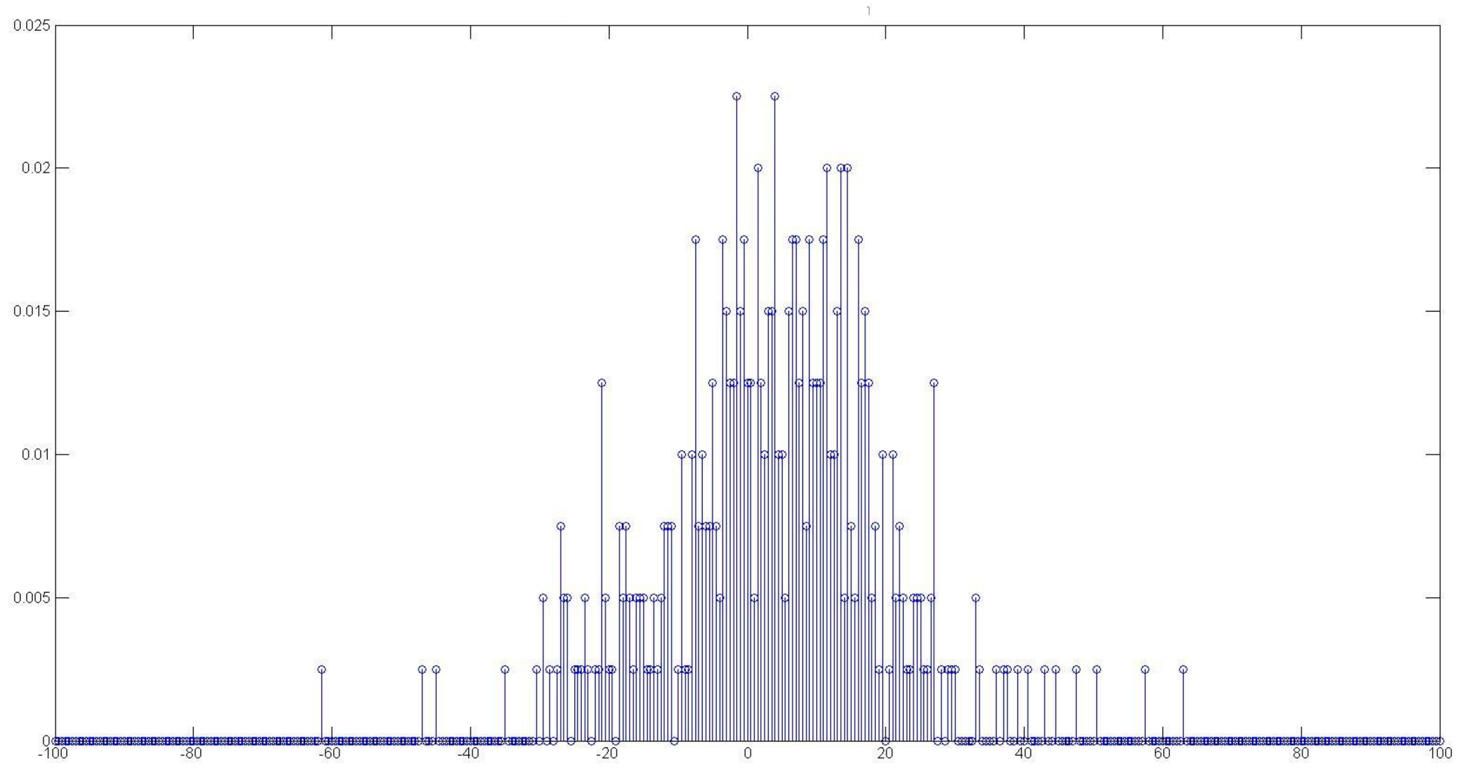
Gradient histogram span

- Note: the edge of blurred image is not as sharp as the edge in the non-blurred image
  - That means that the gradient of blurred image is lower
- Gradient distribution:
  - Blur image: small variance
  - Non-blur image: big variance and presence of heavy-tail compared with the former
- Moreover: the statistics of gradient response in natural images usually follow a mix-gaussian distribution

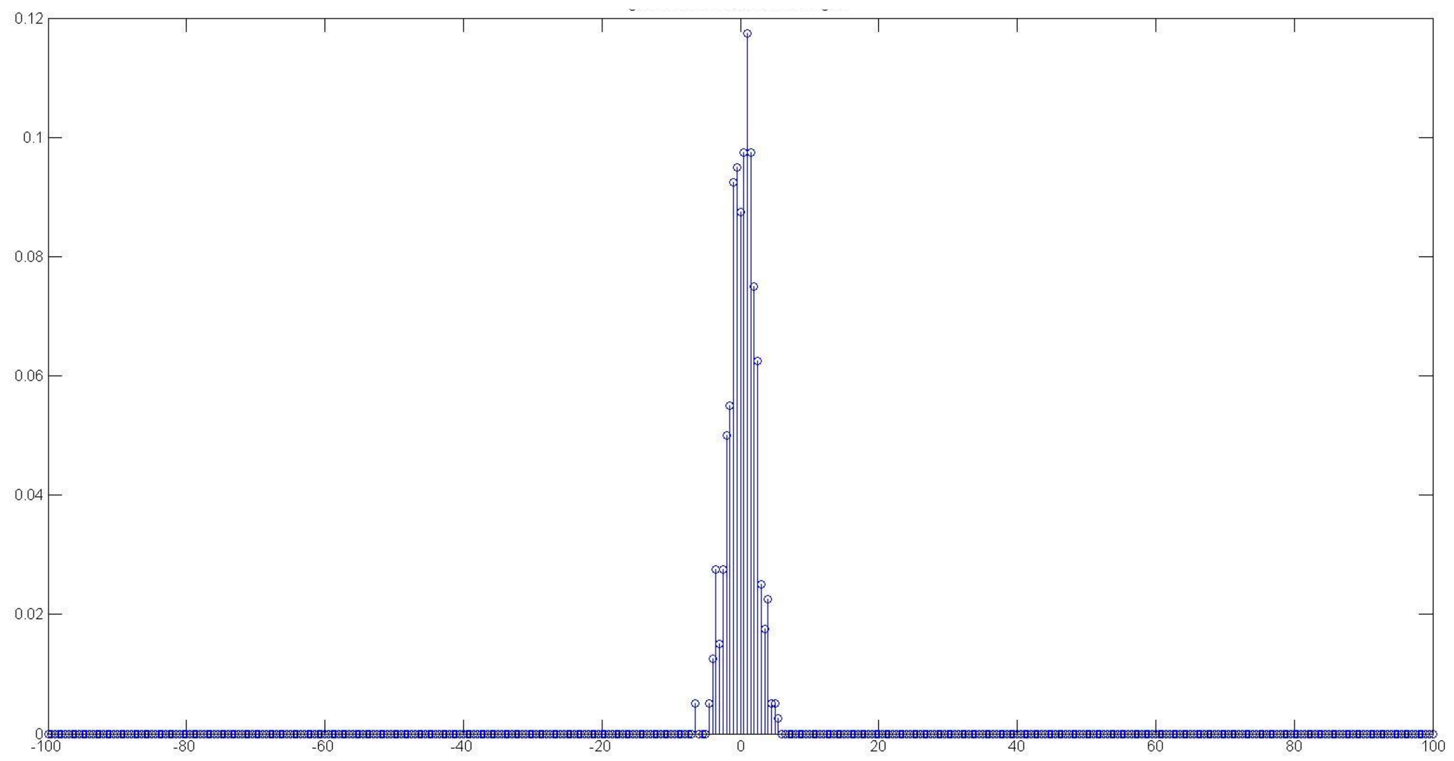
Gradient histogram span



Gradient distribution non-blur region



Gradient distribution blur region



Gradient histogram span

- A mixture of two-component Gaussian model is used to approximate the distribution:

$$G = a_1 \exp\left(\frac{-(x-\mu_1)^2}{2\sigma_1{}^2}\right) + a_2 \exp\left(\frac{-(x-\mu_2)^2}{2\sigma_2{}^2}\right)$$

- $\mu_1 = \mu_2 = 0$
- a_1 , a_2 are constants
- $\sigma_2 > \sigma_1$

Gradient histogram span

- The gaussian component with larger variance is mainly responsible for causing the heavy-tail in the original distribution of the gradient

- For this reason we set σ_2 as a blur factor

$$\boxed{q_1 = \sigma_2}$$

Expectation maximization

- “The Expectation Maximization algorithm estimates the parameters of the multivariate probability density function in the form of a Gaussian mixture distribution with a specified number of mixtures”
- We have exploited the openCV function *EM*
- Input: gradient of image, number of components of the gaussian model (=2)
- Output: list of parameters
  - Means
  - **Variances**
  - Weights

Local mean square error map

- We define LMSE as the sum of all pixel’s mean square in the patch, expressed by:

$$V_p = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - mean)^2}$$

- It’s a measure of the variance between the pixel and the mean value:
  - Large values near sharp edge
  - Small values in the blur regions

Local mean square error map

- We consider the relative local to global variance as our blur factor. Let be V_o the mean square error of the entire image, so:

$$\boxed{q_2 = \frac{V_p - V_0}{V_0}}$$

Maximum saturation

- It’s observed that blurred pixels tend to have less vivid colors than non-blurred pixels
- For this reason color information is also useful for blur detection
- We compute the saturation color for every pixel as:

$$S_p = 1 - \frac{3}{R+G+B} \min\{R, G, B\}$$

Maximum saturation

- Then we find the maximum value for every patch

- At last we compare it with the maximum saturation value of the whole image (S_o), in order to obtain the third blur factor:

$$\boxed{q_3 = \frac{\max\{S_p\} - \max\{S_0\}}{\max\{S_0\}}}$$

Blur/non-blur mask

- What do we do?
  - The image is partioned to patches size of 20x20
  - We set different thresholds T_b and T_d for each blur measure
  - If $q_i < T_b$ the patch is marked as blurred with white color
  - If $q_i > T_d$ the patch is marked as non-blurred region with black color
- Results:
  - There can be some detection errors
  - Not all blur regions can be picked up
- Reasons: for example color information between blur and non-blur regions is similar

Blur/non-blur mask

- Hence we combine the three features to improve accuracy

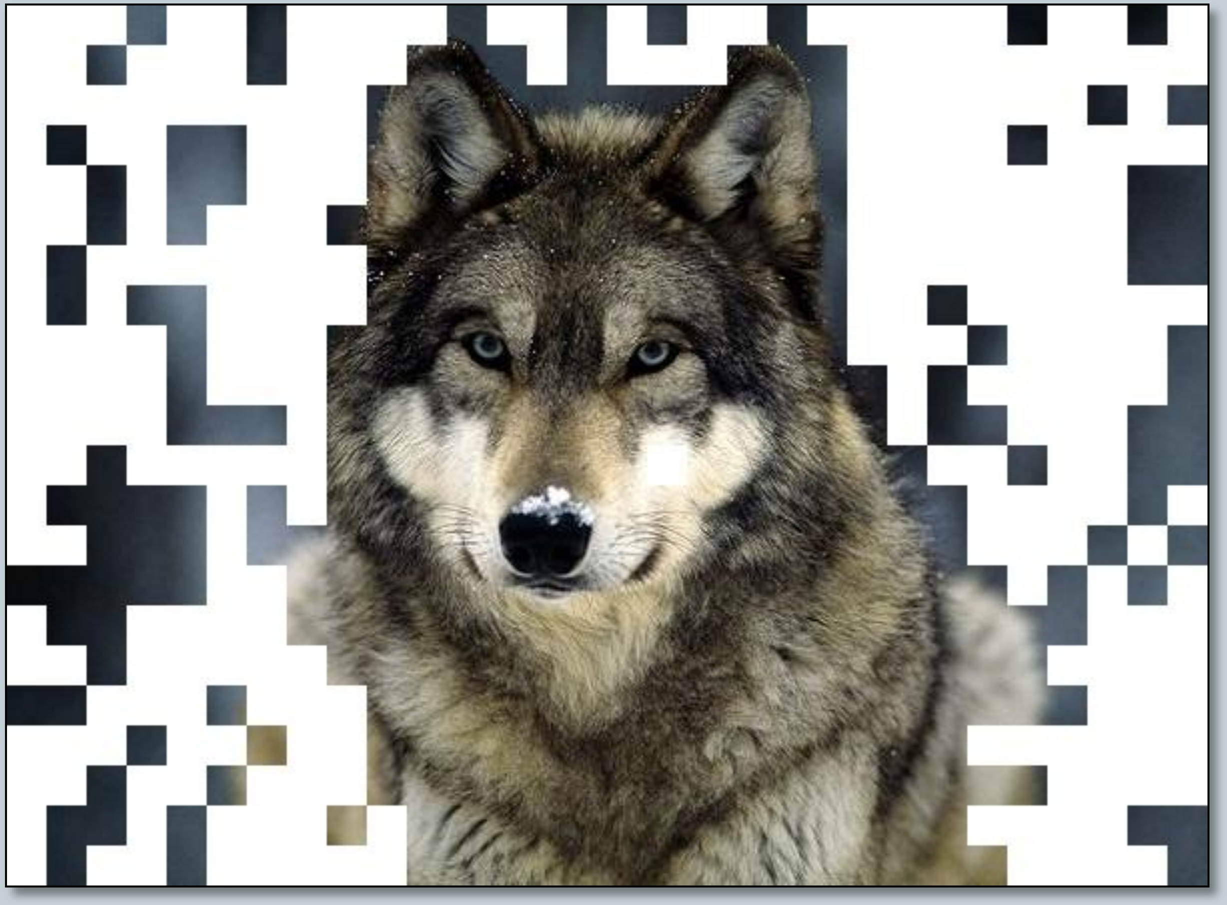
- That is:
  - If all of the three features regard a patch blurred, then this patch is marked blurred
  - If all of the three features regard a patch non-blurred, then this patch is marked non-blurred
  - At the contrary patch is not marked

Results

- We show obtained results for three images (blur regions and non-blur regions)
- For everyone we display also the global trimap
- It's important to highlight that we have to set different threshold T_b and T_d for every feature and for every image
- Let be *min_val(q_i)* the minimum value of all features q_i of the patches, and *max_val(q_i)* the maximum value of all features q_i of the patches:
  - $T_b = min\_val + x\ \%\ of\ (max\_val - min\_val)$
  - $T_d = max\_val - y\ \%\ of\ (max\_val - min\_val)$

Test 1

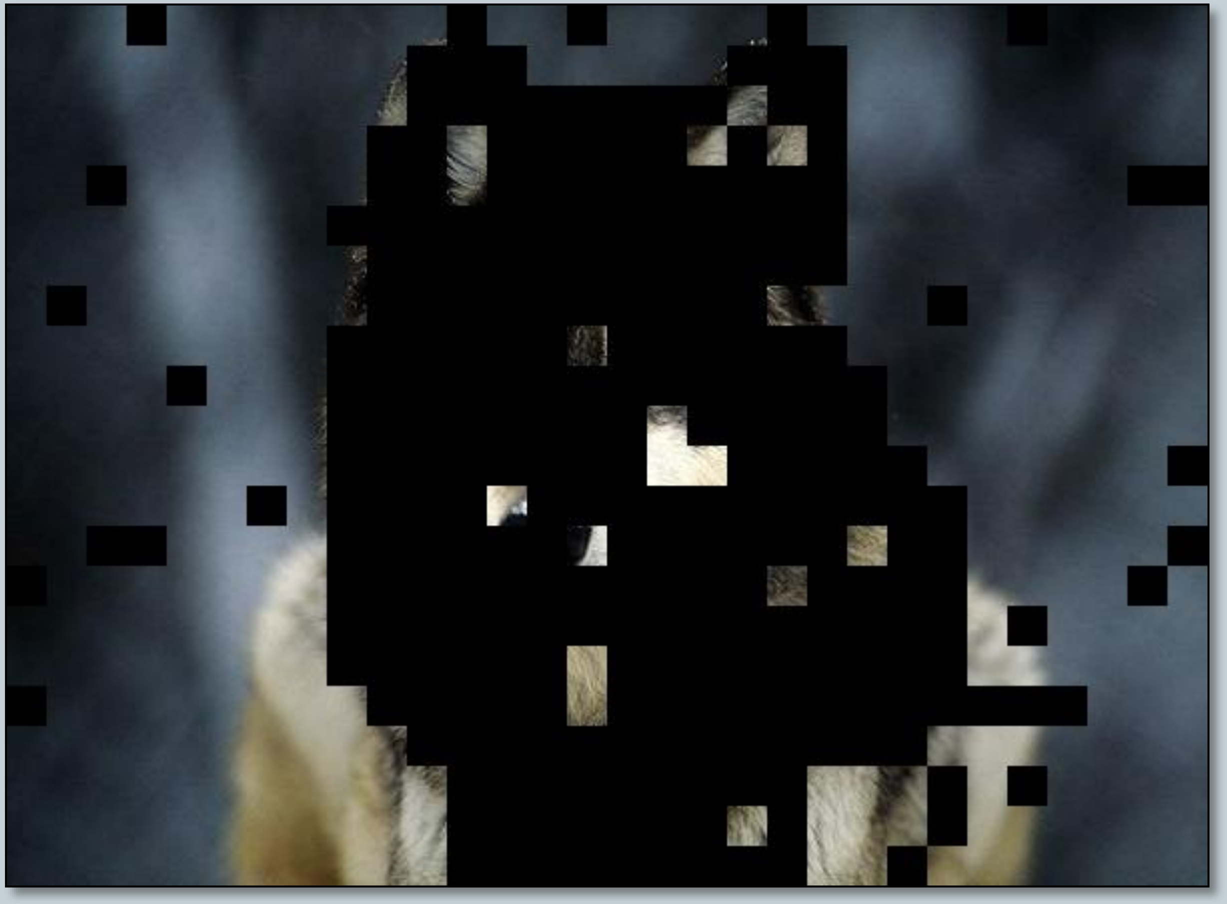
Test 1: Gradient histogram span



Blur regions:

T_b = min_val + 9% of range

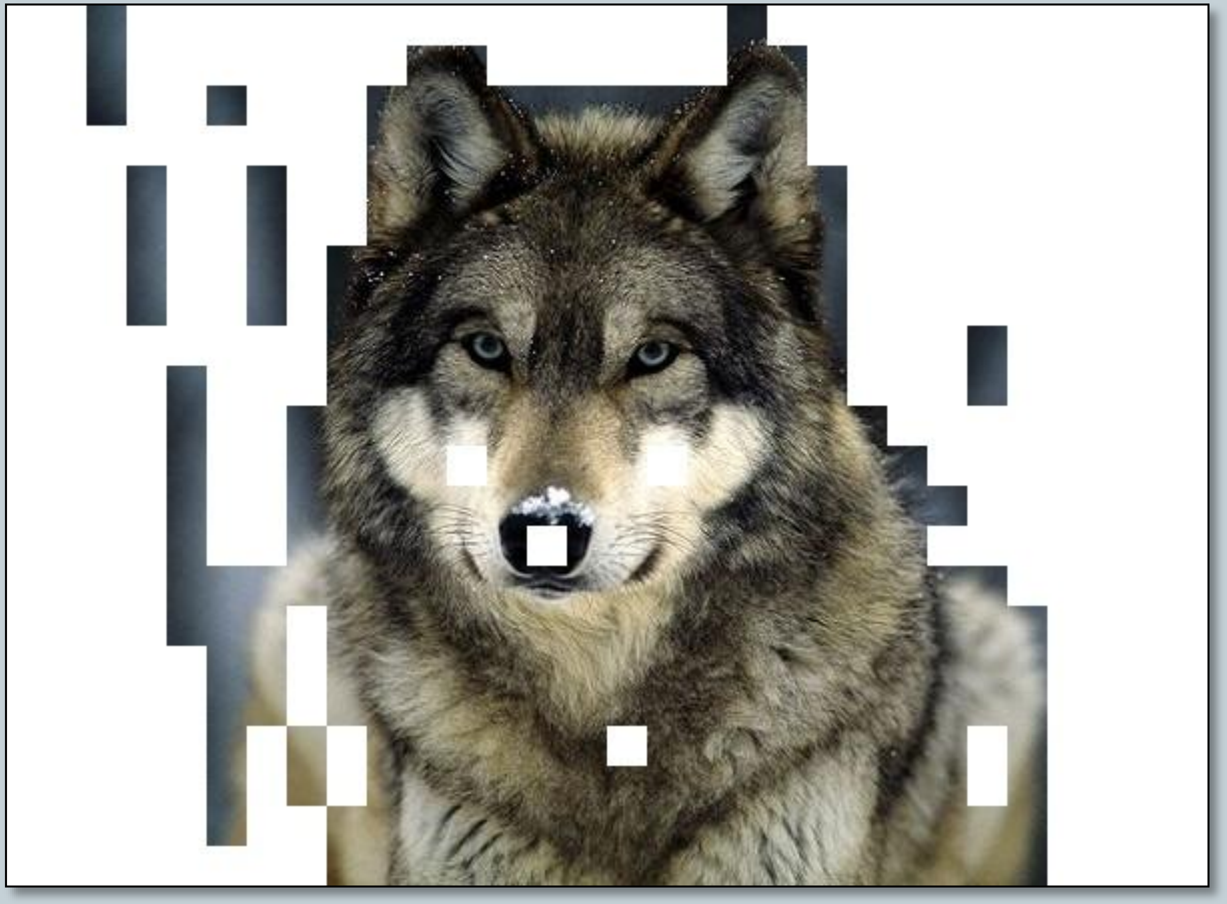
Test 1: Gradient histogram span



Non-blur regions:

T_d = max_val - 50% of range

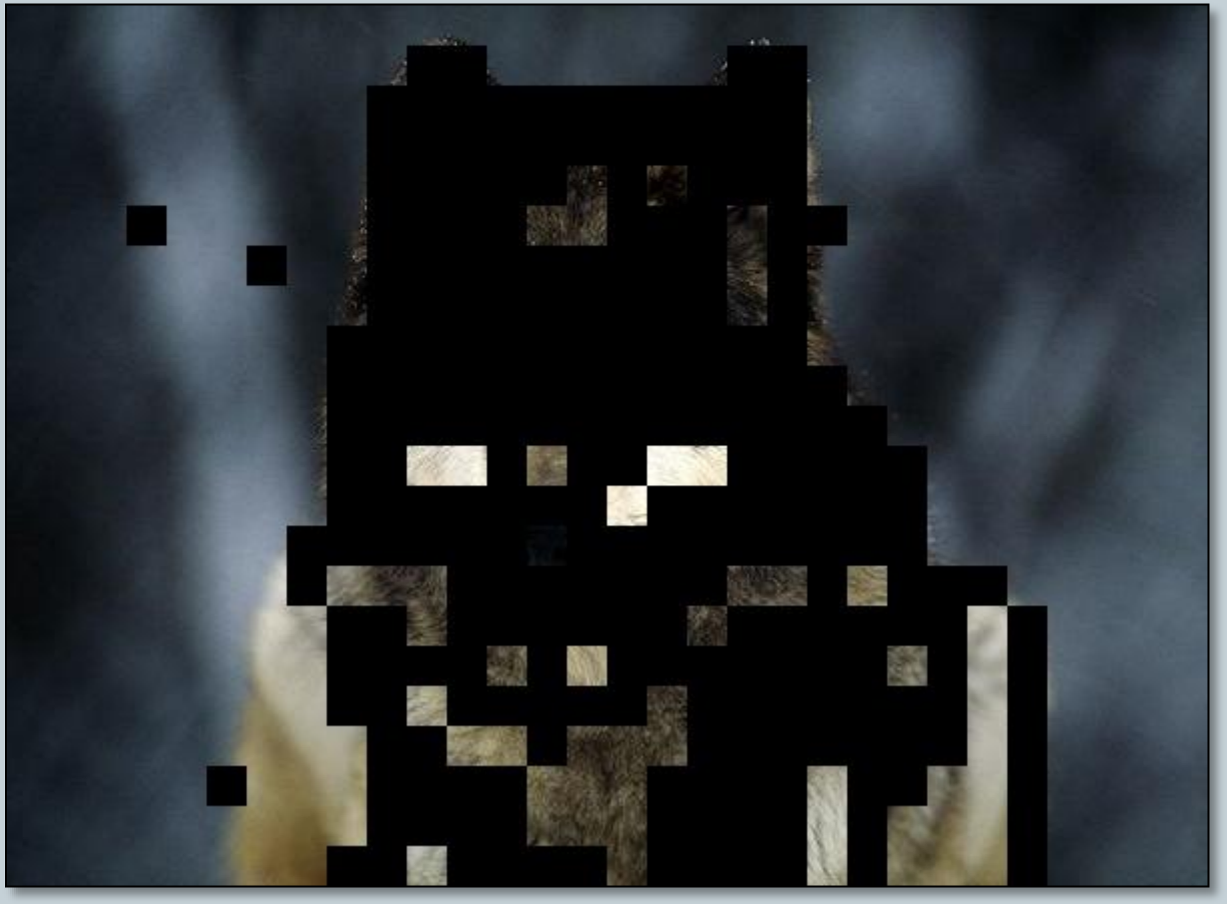
Test 1: Local mean square error map



Blur regions:

T_b = min_val + 12.5% of range

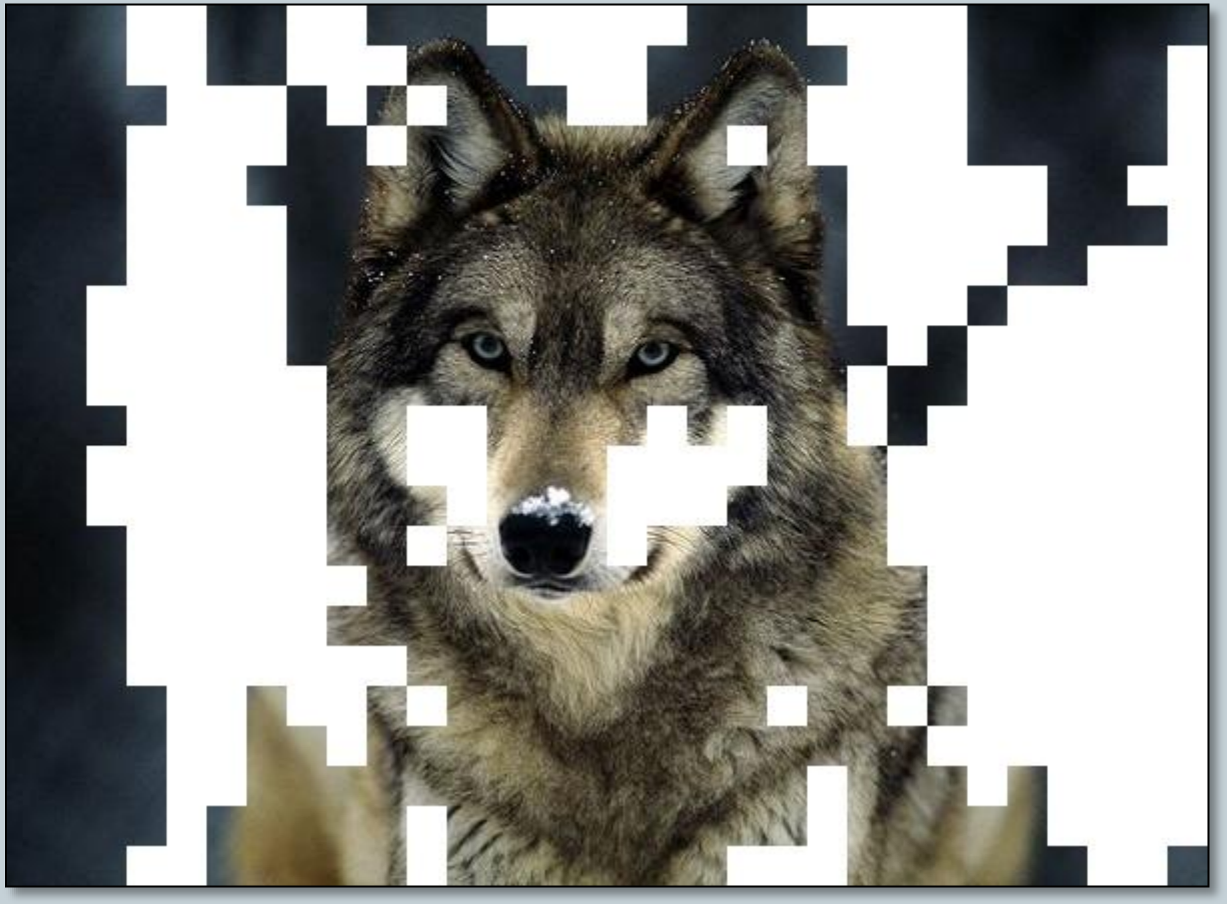
Test 1: Local mean square error map



Non-blur regions:

T_d = max_val - 80% of range

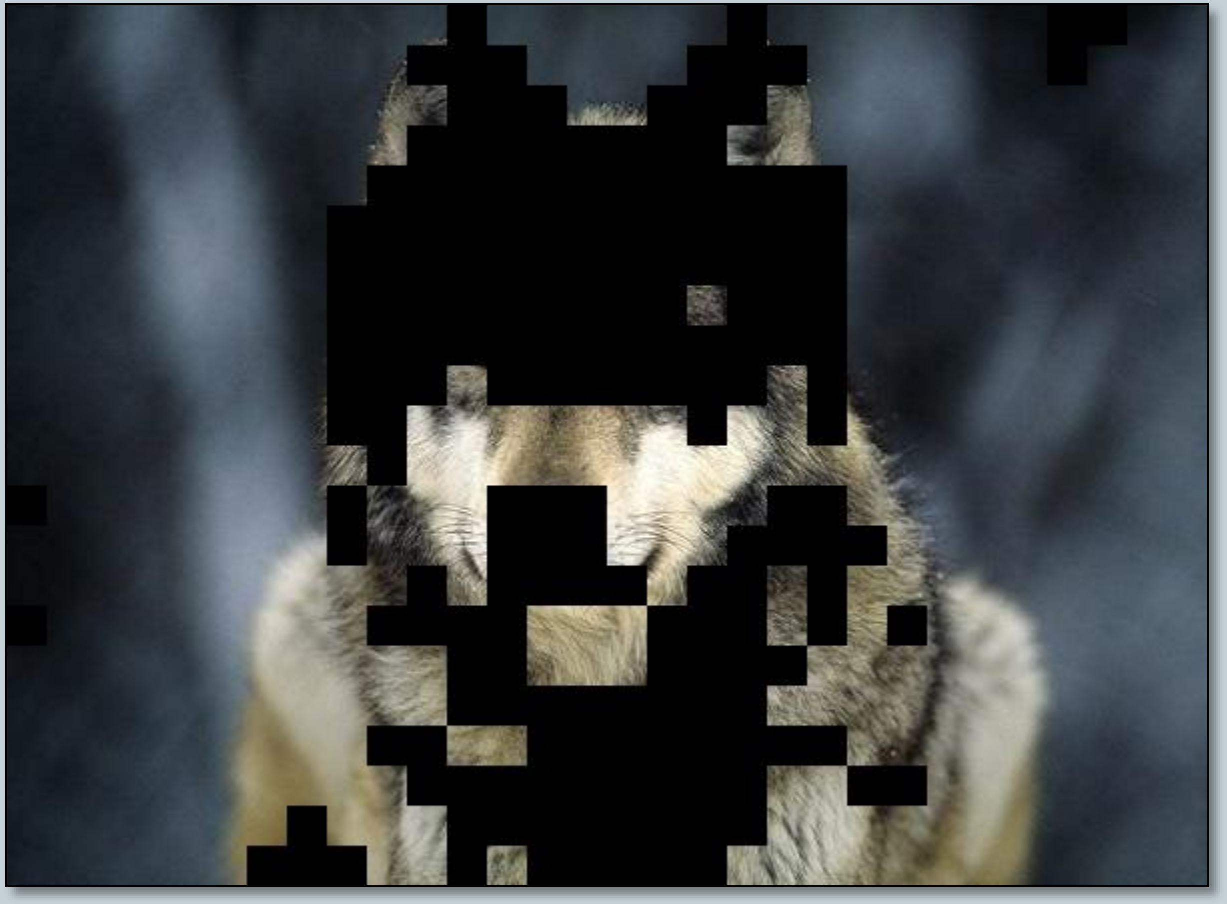
Test 1: Maximum saturation



Blur regions:

T_b = min_val + 17% of range

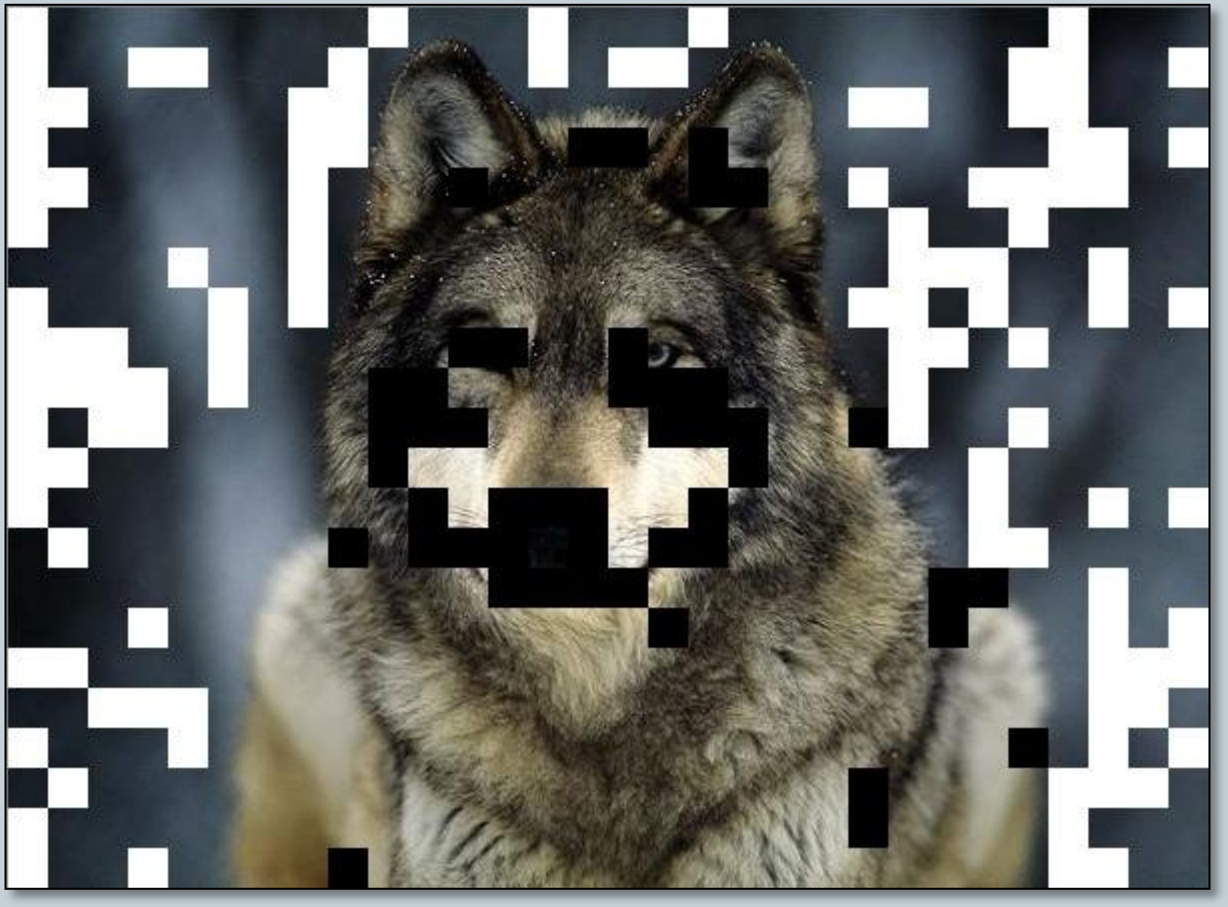
Test 1: Maximum saturation



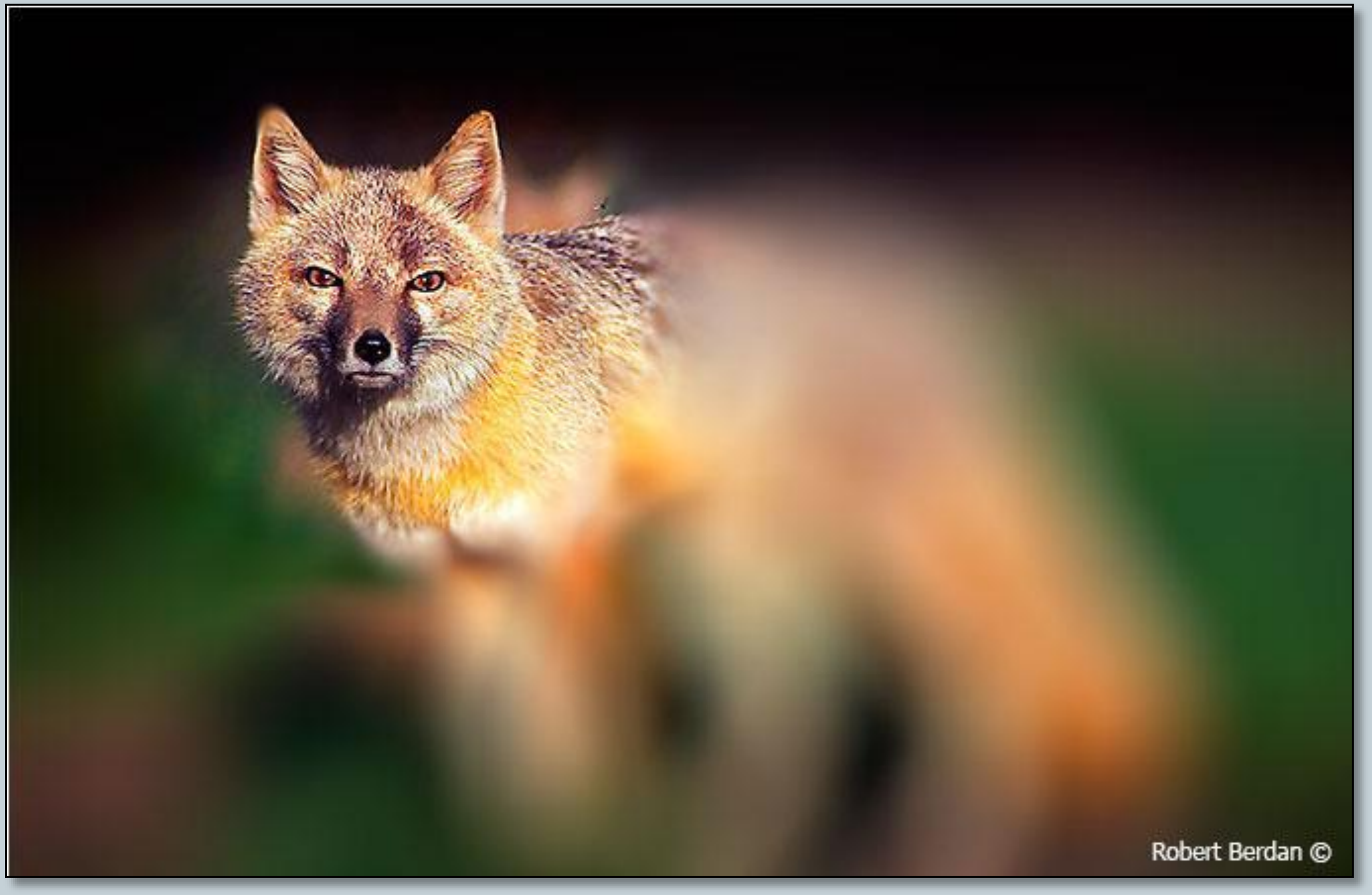
Non-blur regions:

T_d = max_val - 62.5% of range

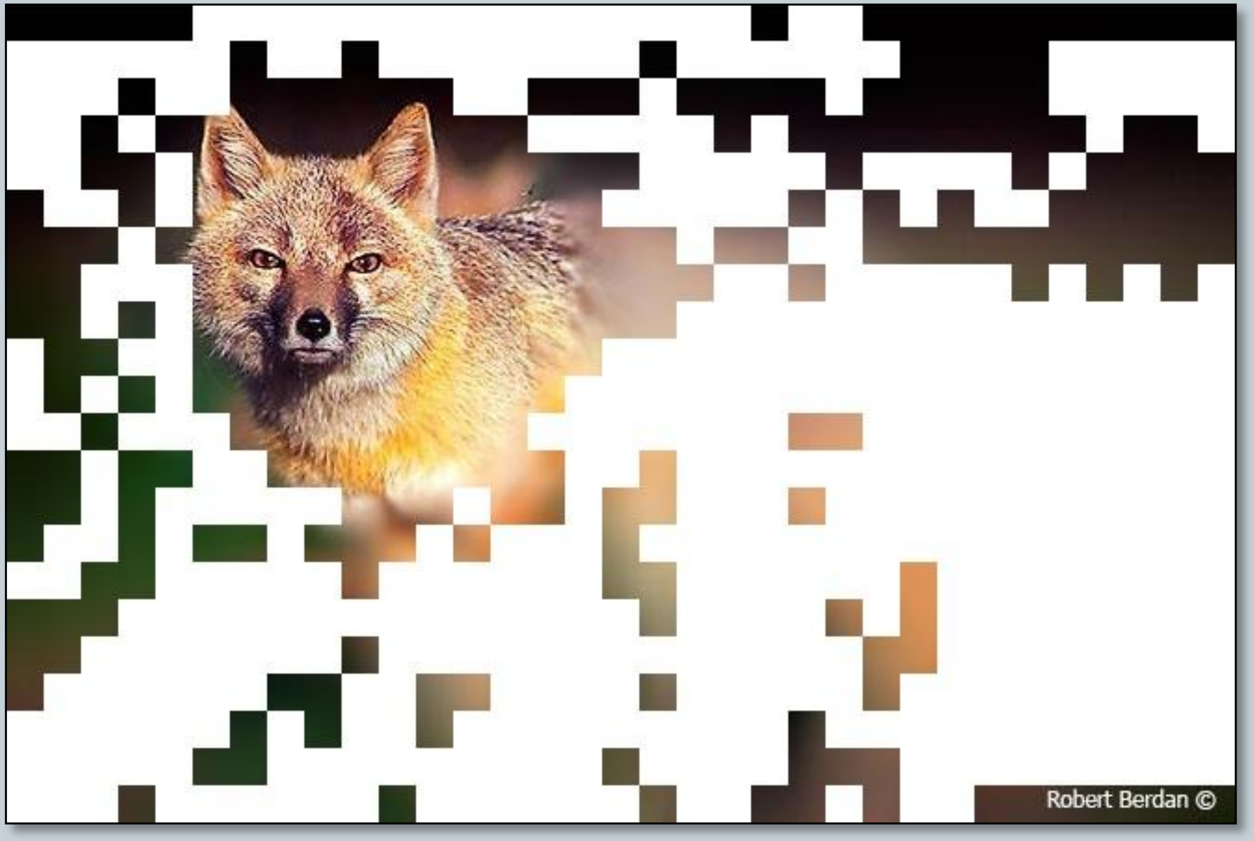
Test 1: trimap



Test 2



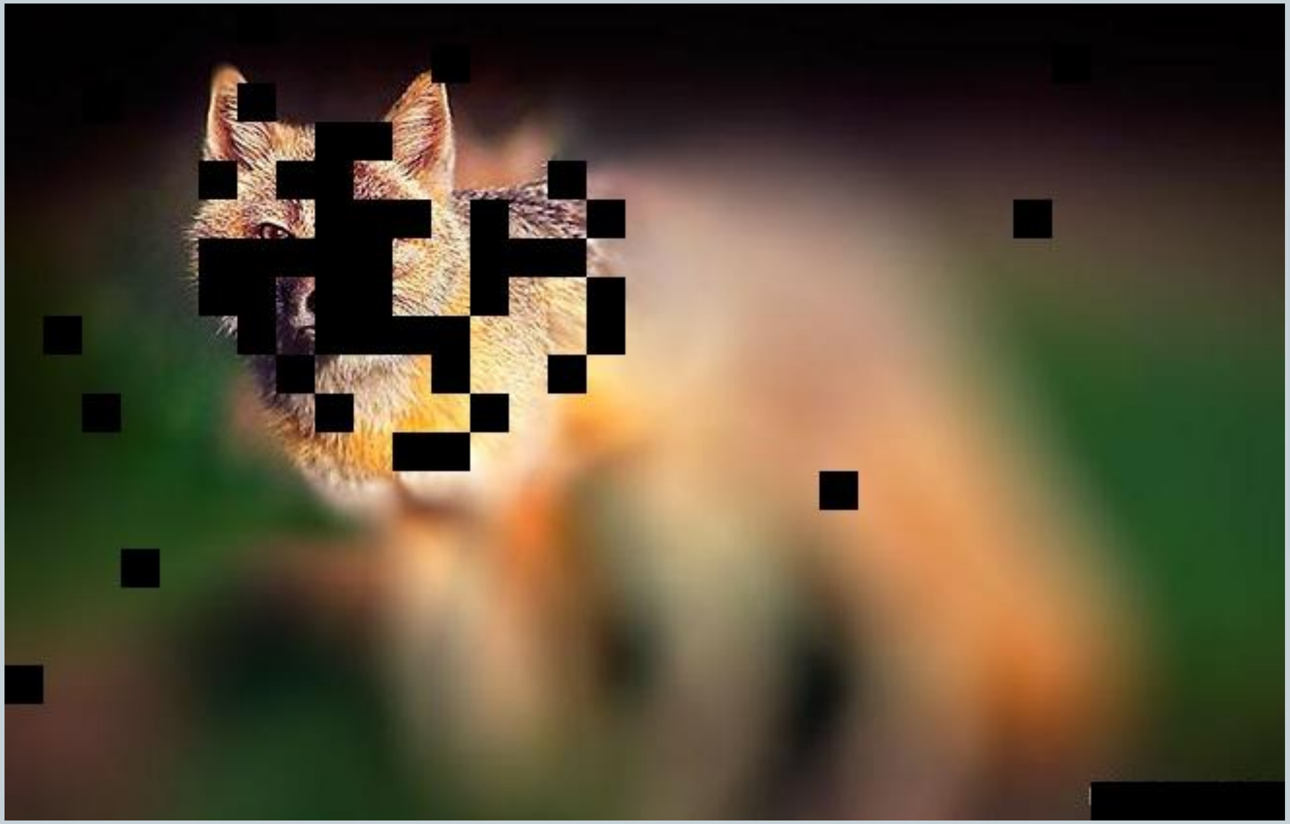
Test 2: Gradient histogram span



Blur regions:

T_b = min_val + 8.5% of range

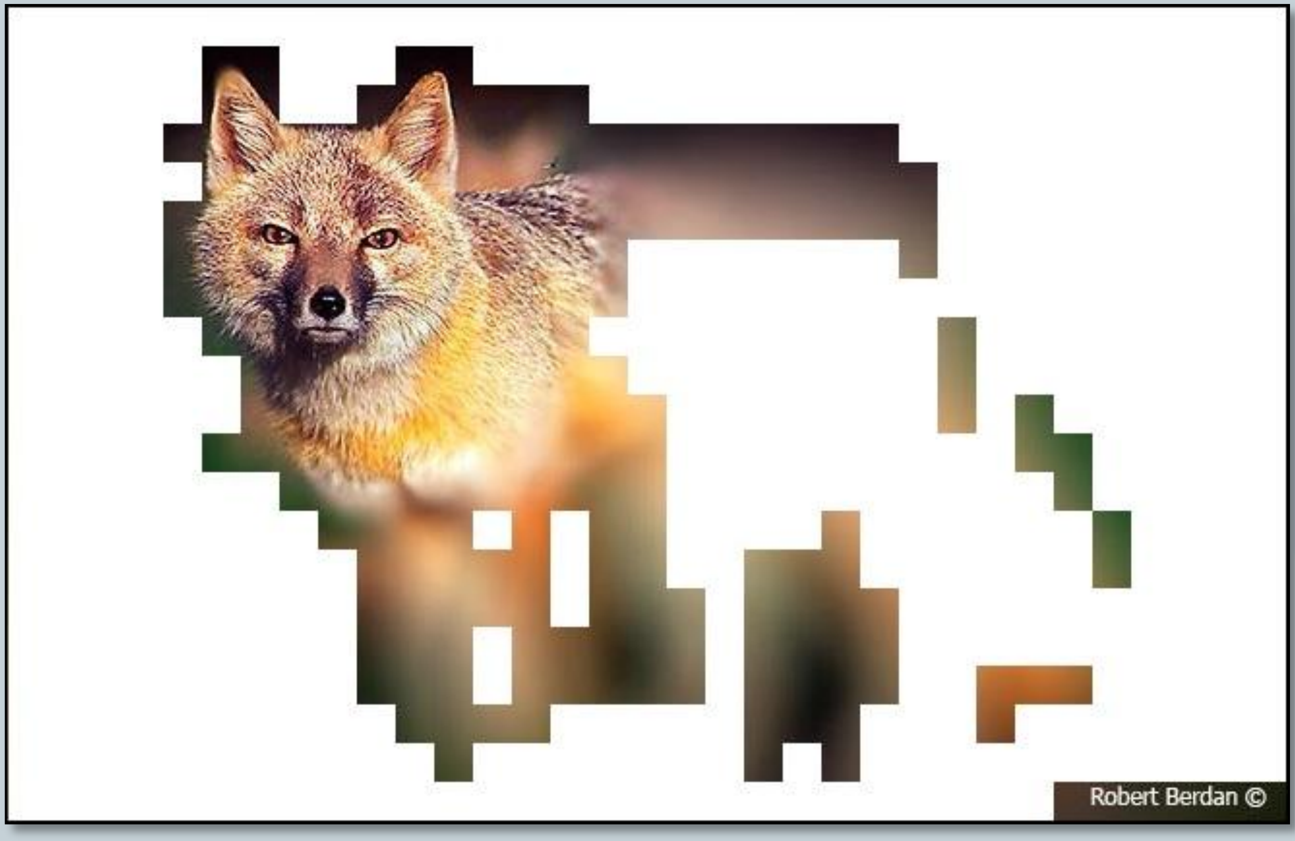
Test 2: Gradient histogram span



Non-blur regions:

T_d = max_val - 50% of range

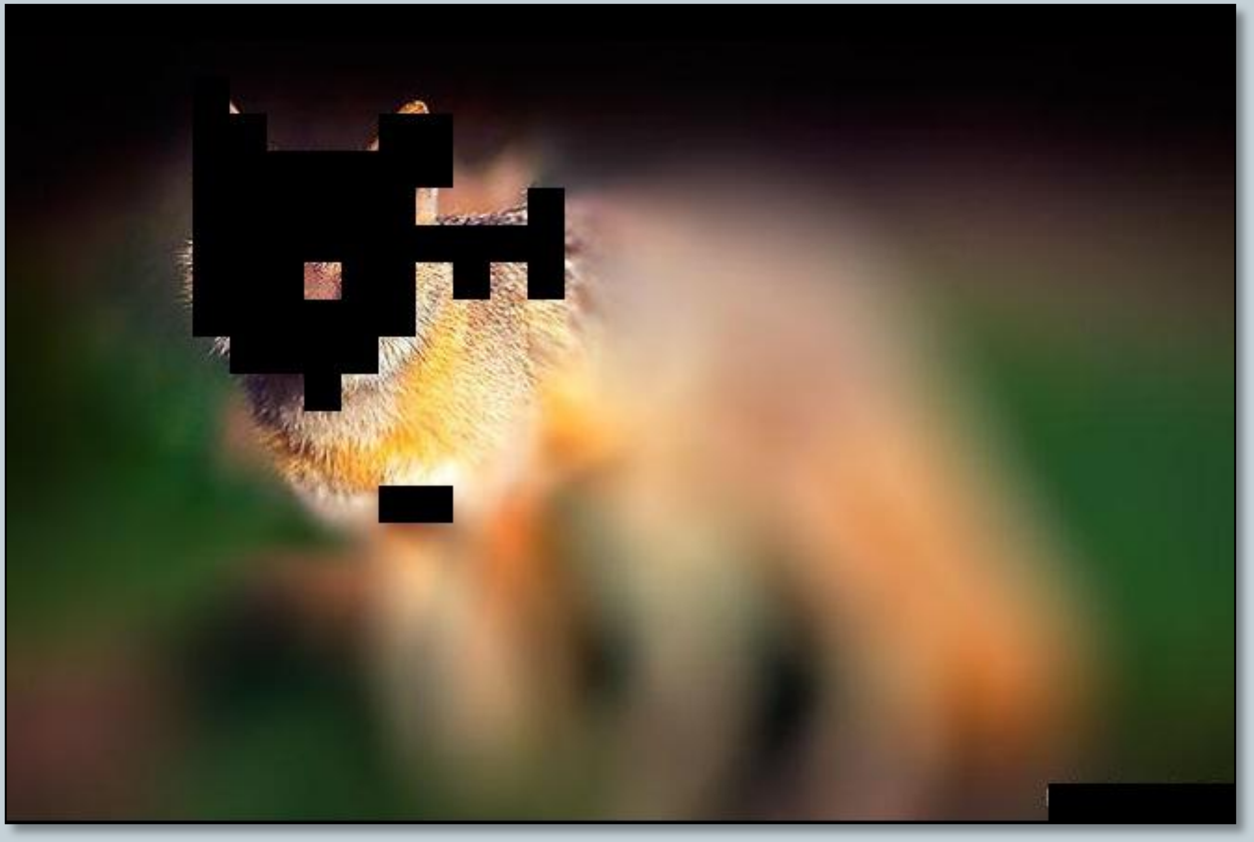
Test 2: Local mean square error map



Blur regions:

T_b = min_val + 12.5% of range

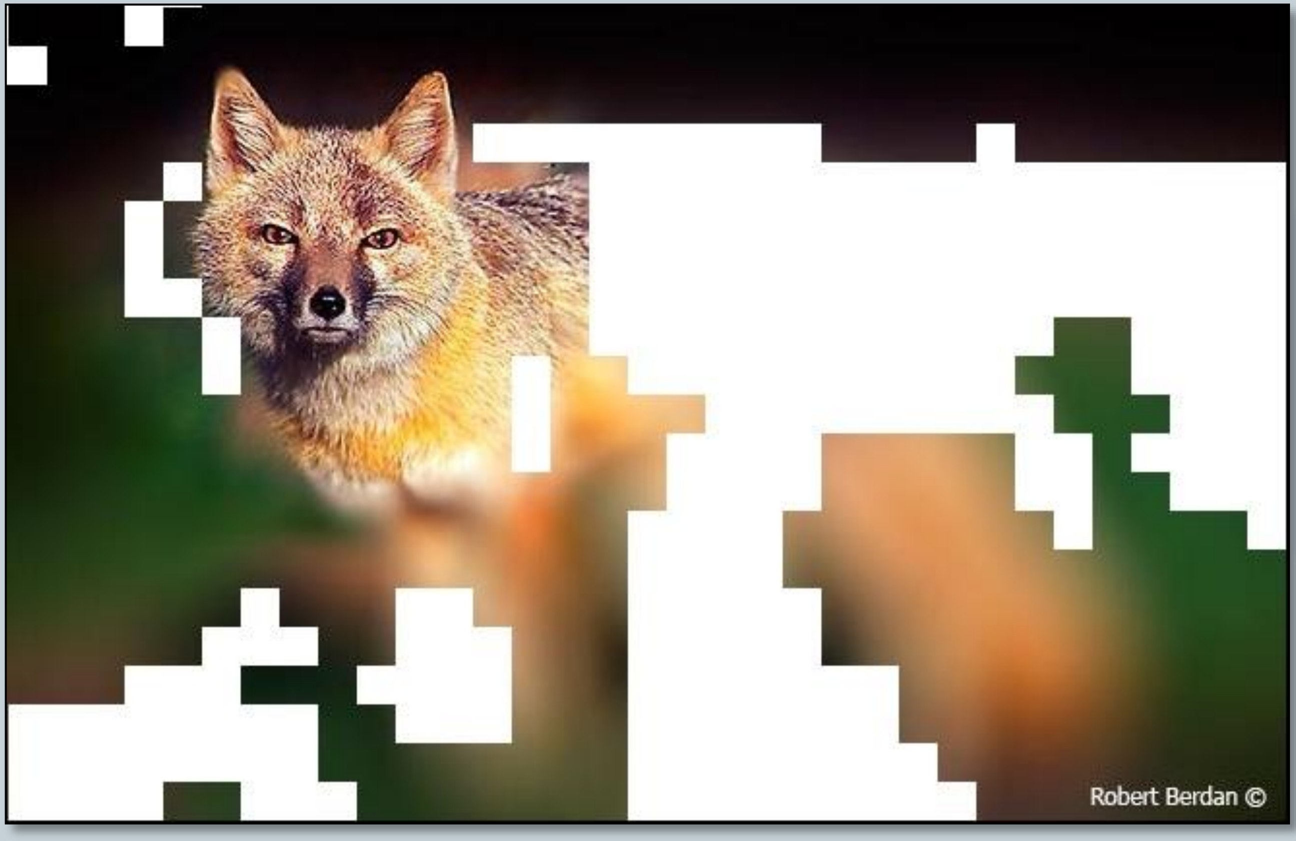
Test 2: Local mean square error map



Non-blur regions:

T_d = max_val - 76% of range

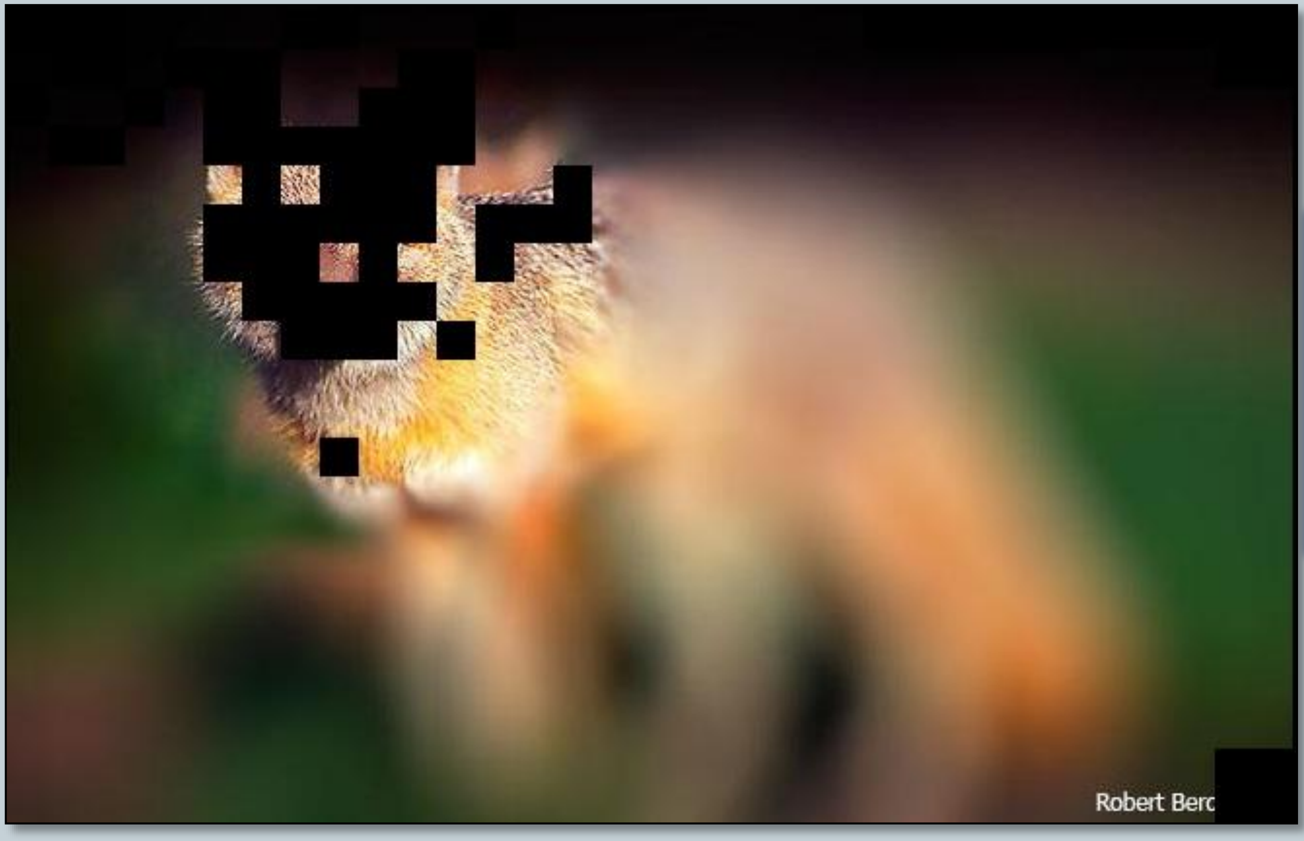
Test 2: Maximum saturation



Blur regions:

T_b = min_val + 1% of range

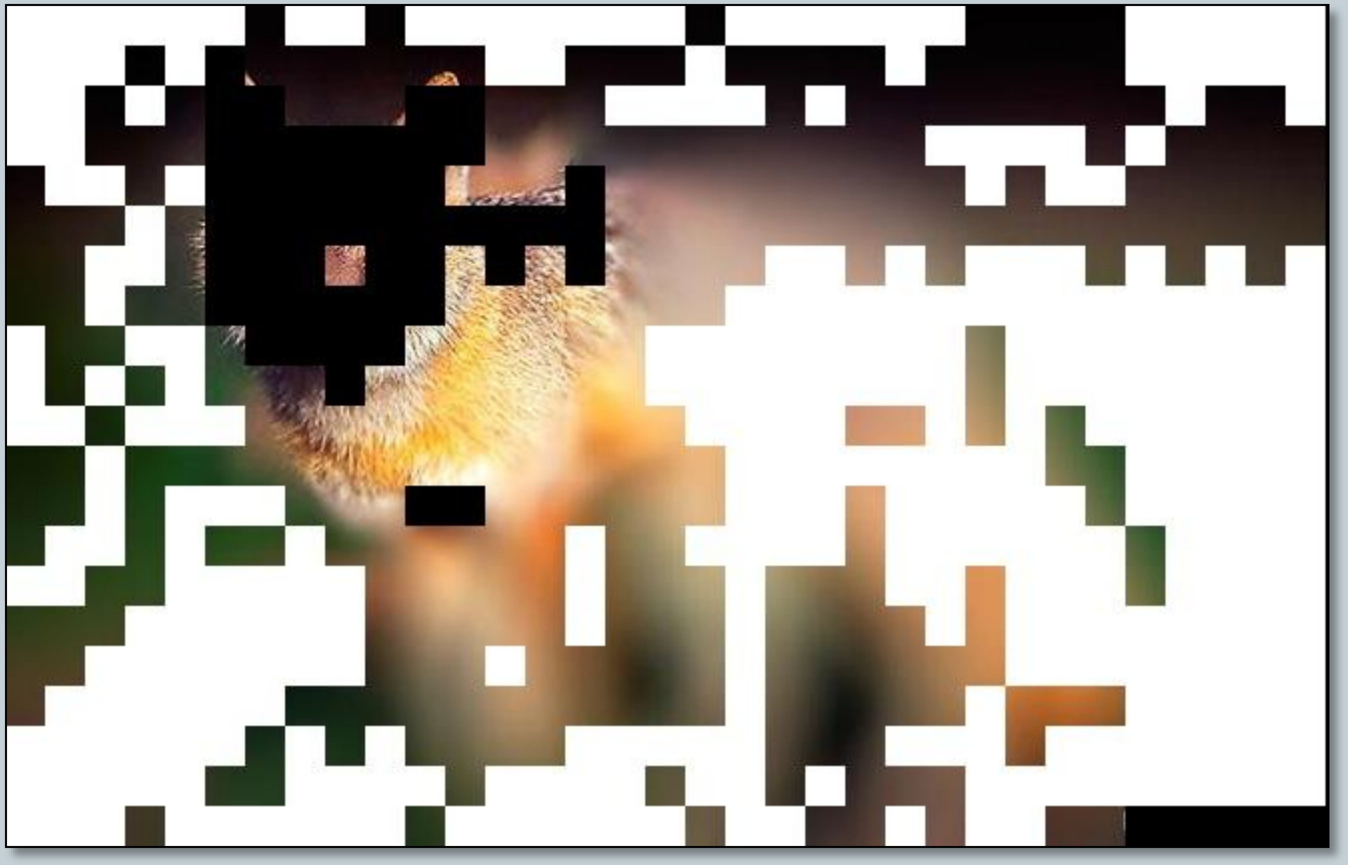
Test 2: Maximum saturation



Non-blur regions:

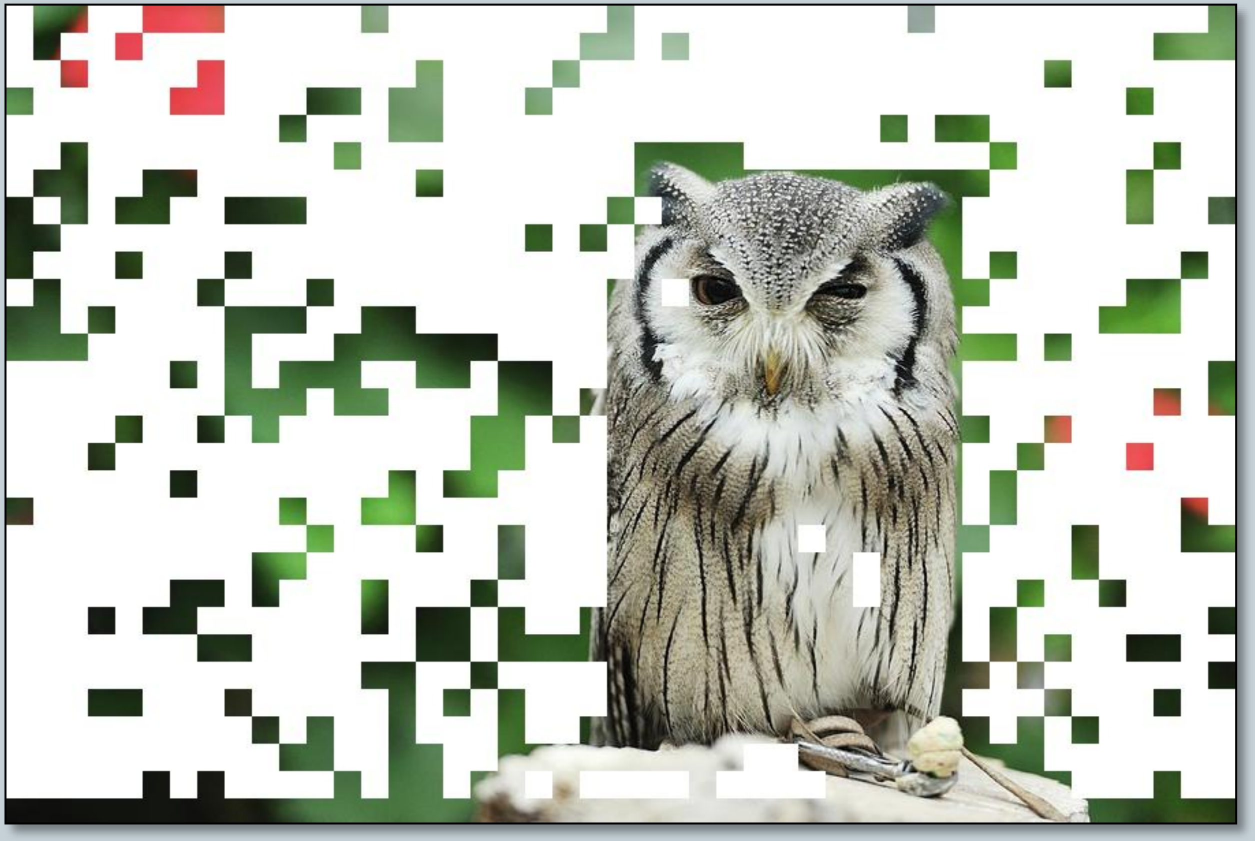
T_d = max_val - 66% of range

Test 2: trimap



Test 3

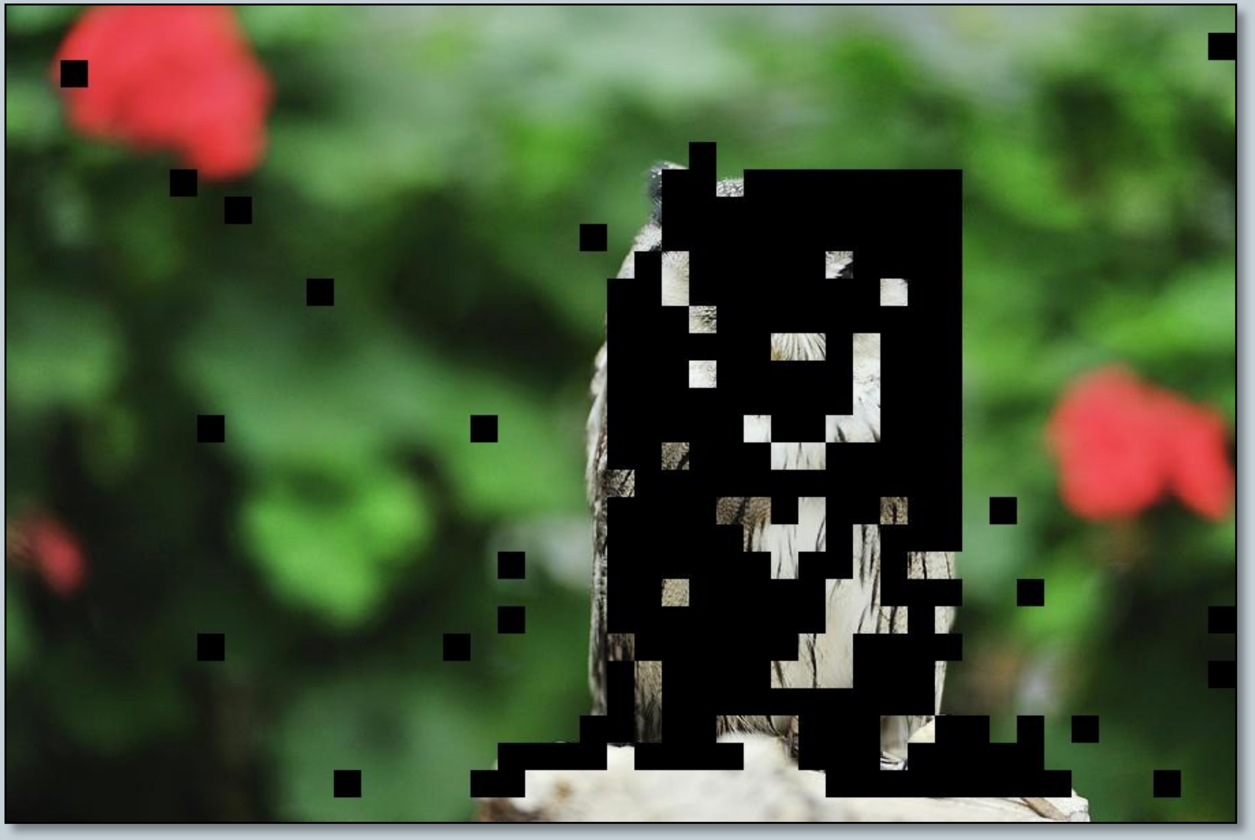
Test 3: Gradient histogram span



Blur regions:

T_b = min_val + 6% of range

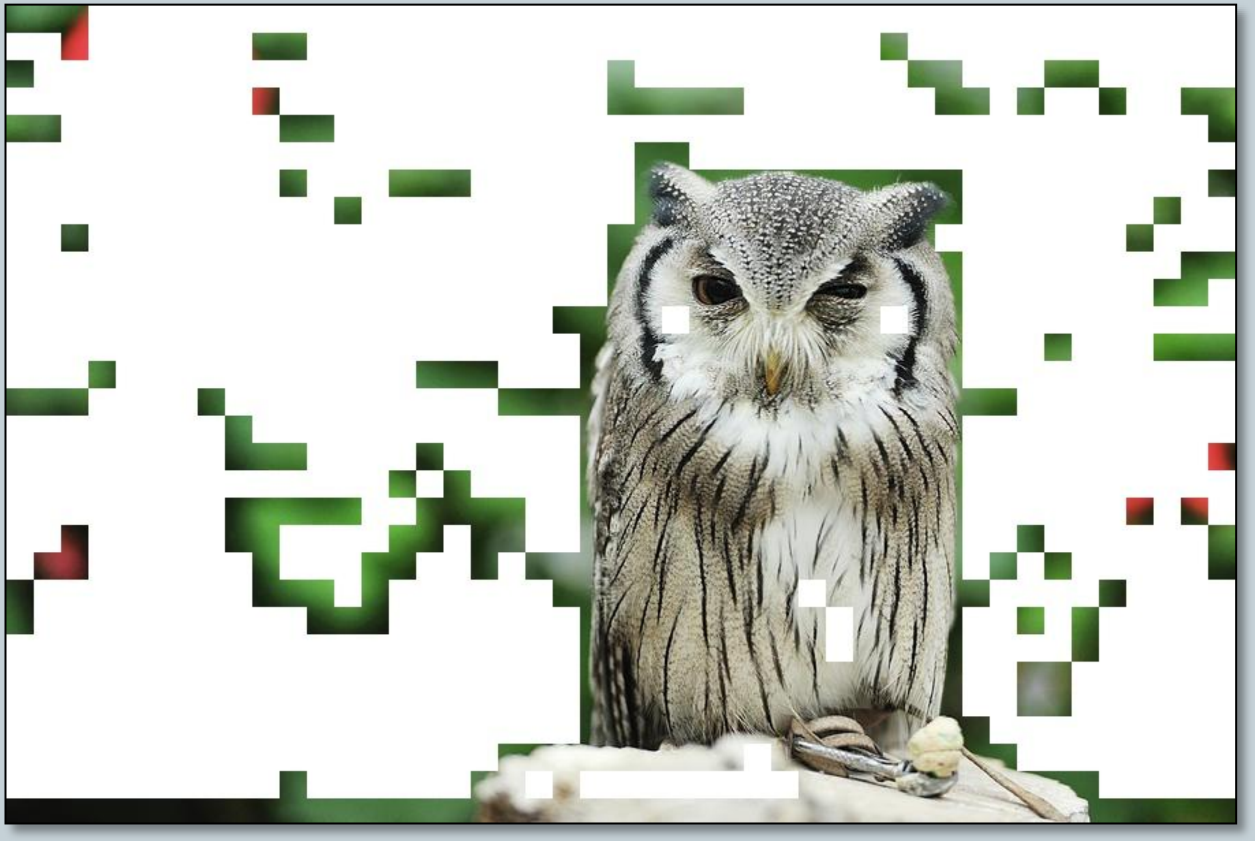
Test 3: Gradient histogram span



Non-blur regions:

T_d = max_val - 62.5% of range

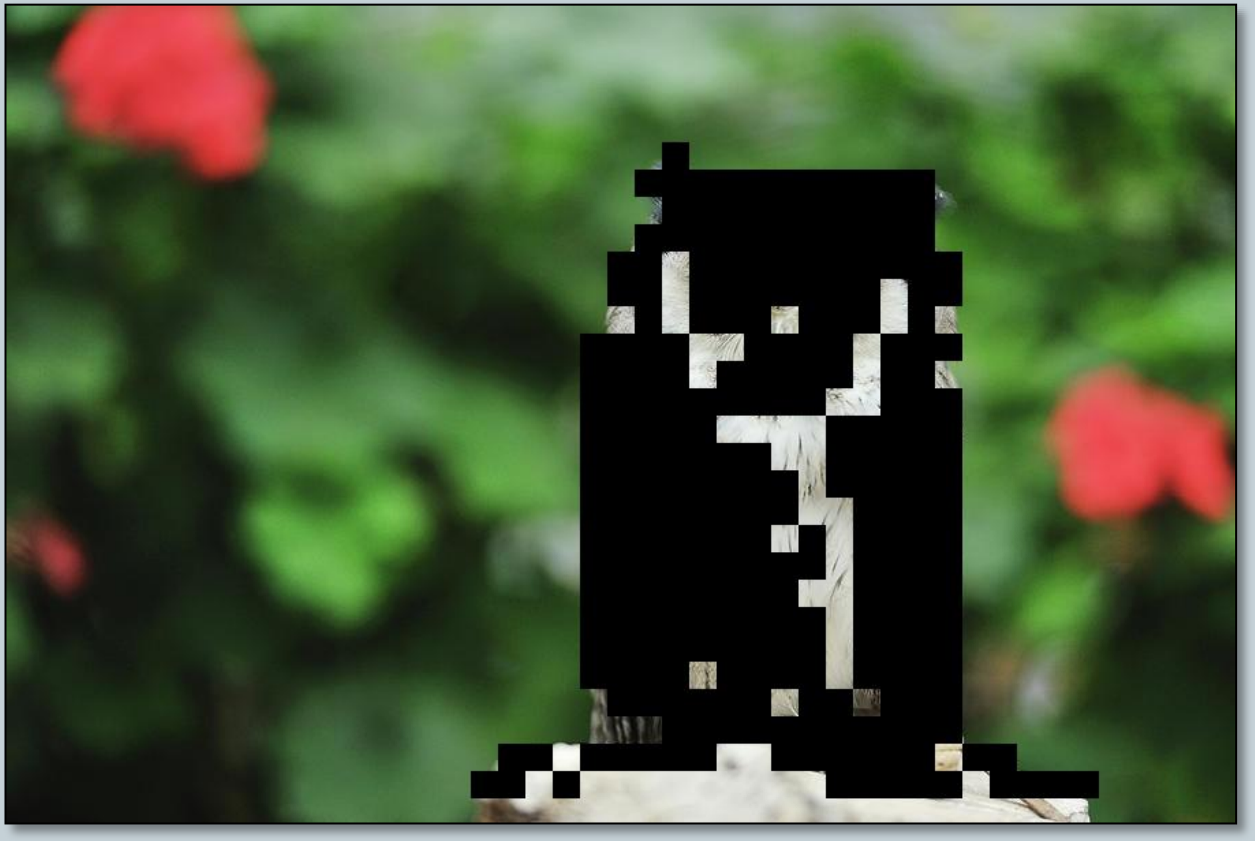
Test 3: Local mean square error map



Blur regions:

T_b = min_val + 14% of range

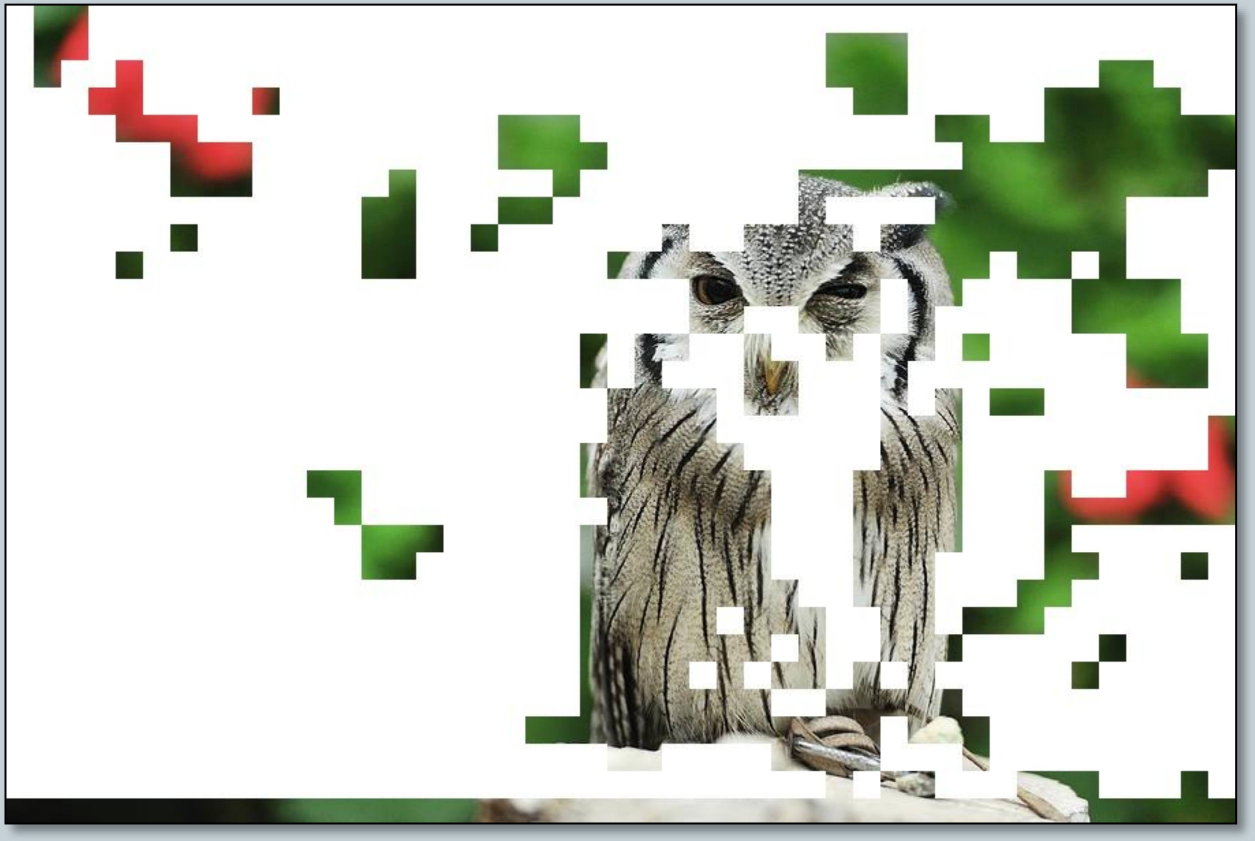
Test 3: Local mean square error map



Non-blur regions:

T_d = max_val - 66% of range

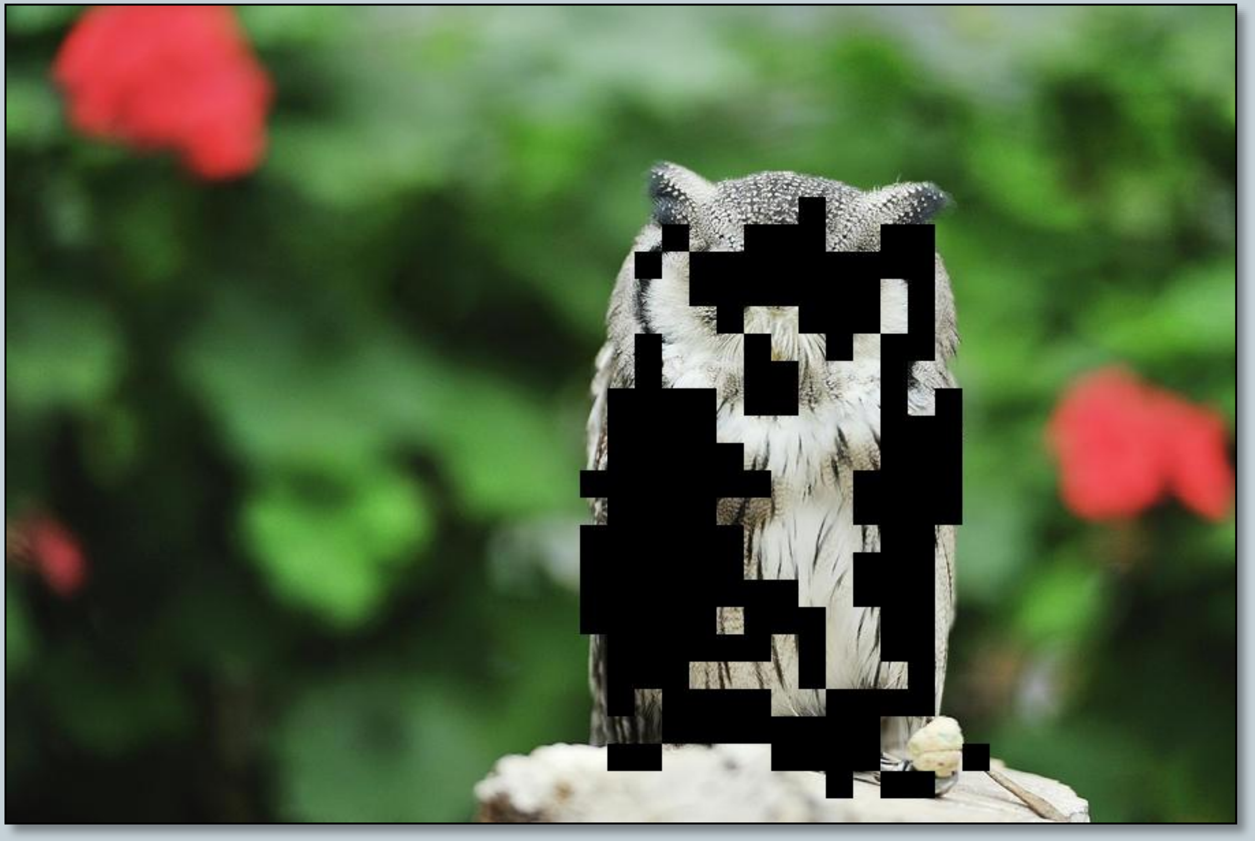
Test 3: Maximum saturation



Blur regions:

T_b = min_val + 50% of range

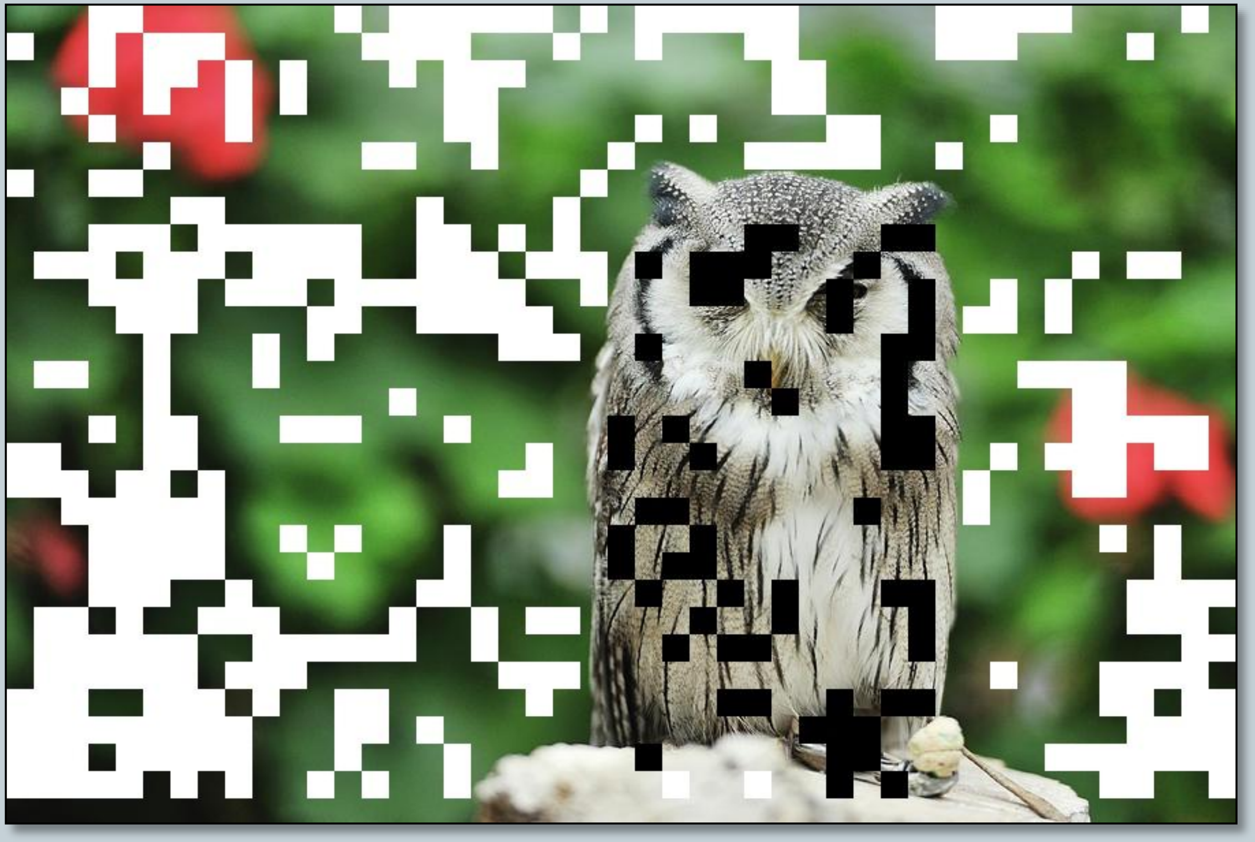
Test 3: Maximum saturation



Non-blur regions:

T_d = max_val - 1% of range

Test 3: trimap



Remarks

- Starting from results we can observe that:
  - Method 1 and method 2 work efficiently
  - Method 3 presents some imperfection more
  - Trimaps allow to remove all blur/non-blur regions identified wrongly
- Possible reasons:
  - Color information similar between blur and non-blur regions
  - Choice of thresholds
- All of these observations agree in principle with that one of the original paper

Image matting

- Separation of a foreground object from the background
- Idea: it's assumed that each pixel x = (i, j) in an image I (x) is a linear combination of a foreground color and a background color. In particular:

$$I(x) = \alpha_x F(x) + (1 - \alpha_x) B(x)$$

- α_x is the opacity value for each pixel, ranged from 0 to 1

Overview

- In the original paper, Wang proposed an iterative optimization algorithm to generate a matte, starting from a few user specified foreground and background pixels
- In our work, blur and non-blur regions substitute those marked by user
- The goal of this method is to determine, for each pixel *p*:
  - a foreground color, F
  - a background color, B
  - an alpha value between 0 to 1, α

  and to reduce the uncertainty *u* (also between 0 and 1) of these values

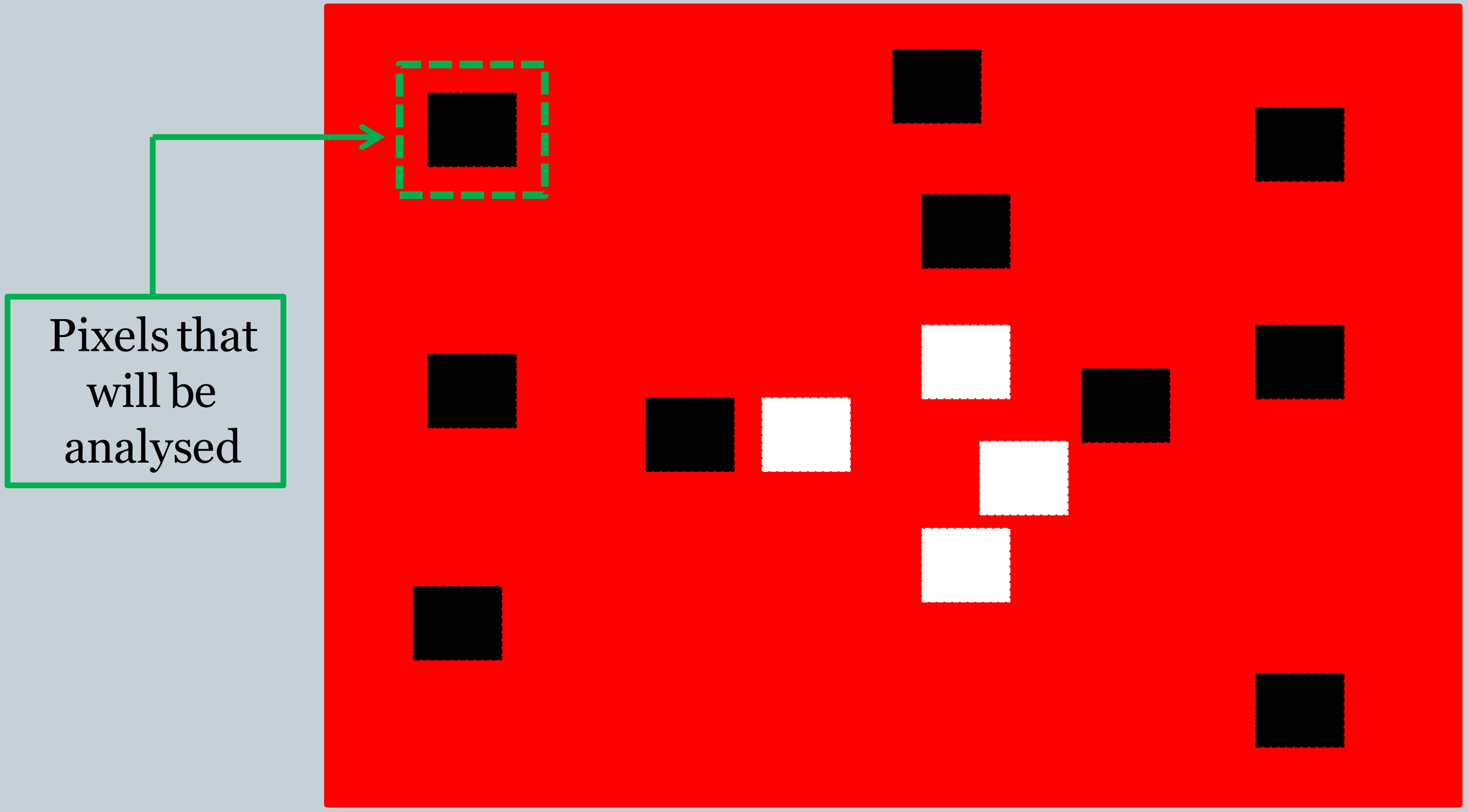
Overview

- Input:
  - Mixed color of each pixel, C
  - Few foreground and background pixels (obtained from first part)
- Marked pixels are characterized by:
  - Uncertainty of 0
  - Alpha value of 0 (background) or 1 (foreground)
  - Their foreground or background values (C)
- All other pixels are initialized to have $\alpha = 0.5$ and $u = 1$

Overview

- We divide pixels in three groups:
  - U_c: pixels whose alpha values have been estimated in previous iterations and uncertainty is equal to zero
  - U_e: pixels whose alpha values have been estimated in previous iterations and uncertainty is not equal to zero
  - U_n: pixels not yet considered
- The approach proceeds iteratively:
  - Scan each pixel of the image:
    - If p € (U_n or U_e) and it's nearby to ones in (U_c or U_e) (within 15 pixels) then it's added to U_e and (F, B, α and u) are estimated or re-estimated
  - The algorithm stops when U_n is null and the sum of all pixel uncertainties cannot be reduced anymore

Overview



Algorithm: local sampling area

- We discretize the possible alpha value to 25 levels between 0 and 1: α^k, k = 1, 2, …, 25
- In order to estimate α_p we sample a group of previously estimated foreground or background colors from the neighborhood of the node *p* (local neighborhood area with radius $r = 20$ around *p*)
- In order to be a vaild…
  - …foreground sample then $\alpha_s > \alpha_p$
  - …background sample then $\alpha_s < \alpha_p$

Algorithm: weights

- The set of valid samples, p_i, is then weighted in this way:

$$\omega_i^x = (1 - u(p_i))\exp\left(\frac{-s(p, p_i)^2}{\sigma_w^2}\right)$$

- Where:
  - $s(p, p_i)$ is the spatial distance between the two points
  - $\sigma_w = r/2$
  - x represents the foreground sample (x = F) or the background sample (x = B)
- We consider the N = 12 largest weights found in the local area (both for foreground and background pixels)

Algorithm: global sampling area

- Problem: it's possible that there aren't N valid foreground and background samples for a pixel
- Solution used in the paper:
  - GrubCut system proposed by Rother , Kolmogorov and Blake
  - Idea: train a Gaussian Mixture Model on the user specified foreground and background pixels and then to assign each marked pixel to a single Gaussian in the GMM
  - Random selection of pixel by each gaussian
- We have not implemented this method

Algorithm: global sampling area

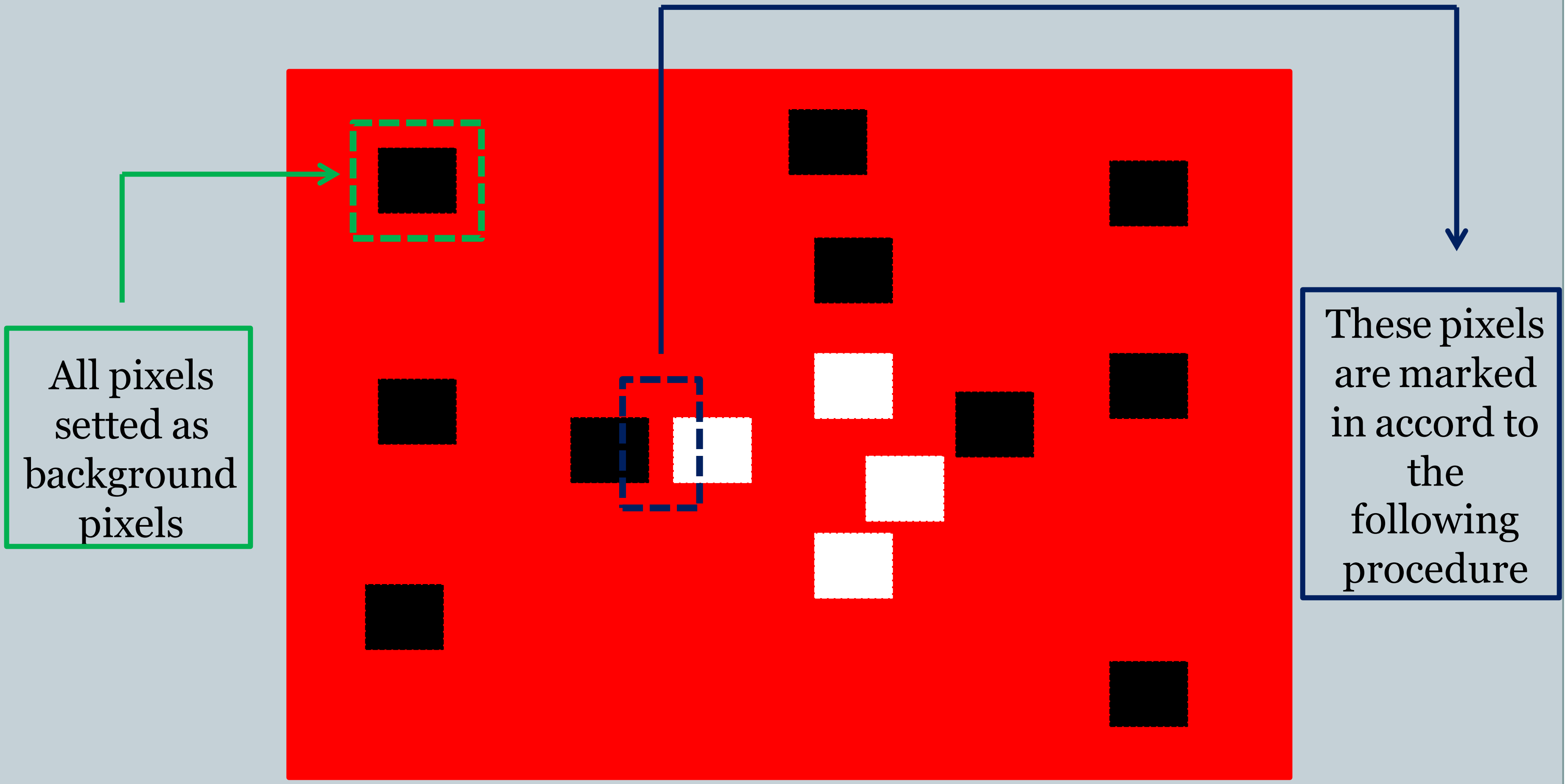
- Initial implemented solution:
  - We save all the positions of foreground pixels marked at the beginning of the process (blur/non-blur mask)
  - Let be:
    - X: number of foreground samples found in the local area
    - f_p: array containing the positions of foreground pixels marked at the beginning of the algorithm
    - L: length of f_p
  - So we extract N-X random numbers between 1 and L, indicating locations of foreground pixels to consider
  - Same method is applied to get background samples

Algorithm: global sampling area

- Problem: extracted samples are too distant from analysed pixel…
- …following operations don't work accurately

- Conclusive solution:
  - If a pixel is surrounded only by foreground samples, then that pixel is setted as foreground (following steps will be not considered until update operation)
  - Similar procedure for background samples

Overall view


All pixels setted as background pixels
These pixels are marked in accord to the following procedure

Algorithm: probability histogram of alpha levels

- So, given the foreground and background samples and corresponding weights, we compute the likelihood of each alpha level α^k:

$$L_k(p) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \omega_i^F \omega_j^B \exp\left[\frac{-d_c\left(C_p, \alpha^k F_i + \left(1-\alpha^k\right) B_j\right)^2}{2\sigma_d^{k\,2}} \right]$$

- Where:
  - F_i are foreground sample colors
  - B_i are background sample colors
  - C_i is the actual color of the examinated pixel *p*
  - d_c is the euclidean distance in RGB space (distance between colors)
  - σ_d^k is the covariance

Algorithm: covariance

- The covariance is computed as:

$$\sigma_d^k = \alpha^k \sigma_F + (1 - \alpha^k)\sigma_B$$

- Where:
  - σ_F is the distance covariance among foreground samples
  - σ_B is the distance covariance among background samples

Algorithm: distance covariance

- Let X_k with k = 1,..,N the values of x-ground sample
- First, compute elements:

$$a_{j,k} = \left| X_j - X_k \right| \qquad j,k = 1,2,\ldots,N$$

- And then the matrix:

$$\boxed{A_{j,k} = a_{j,k} - \bar{a}_{j.} - \bar{a}_{.k} + \bar{a}_{..}}$$

$\bar{a}_{j.}$ is the j-th row mean

$\bar{a}_{.k}$ is the k-th column mean

$\bar{a}_{..}$ is the grand mean of the distance matrix

Algorithm: distance covariance

- Finally, the distance covariance is computed by:

$$\boxed{\sigma_x = dCov_N(X, X) = \sqrt{\frac{1}{N^2}\sum_{j,k=1}^{N} A_{j,k}^2}}$$

Algorithm: update

- Alfa:

$$\alpha_p = \frac{k-1}{24}, k = \arg\left\{\max_k \left[L_k(p)\right]\right\}$$

- If $\alpha_p = 1$:
  - $F_p = C_p$, $B_p = 0$ and $u_p = 0$

- If $\alpha_p = 0$:
  - $B_p = C_p$, $F_p = 0$ and $u_p = 0$

Algorithm: update

- Else:

$$F_p, B_p = \arg\left\{ \min_{F_i, B_j} \left[C_p - \alpha_p F_i - (1-\alpha_p) B_j \right]^2 \right\}$$

- Finally:

$$u_p = 1 - \sqrt{\left(\omega^F \omega^B\right)}$$

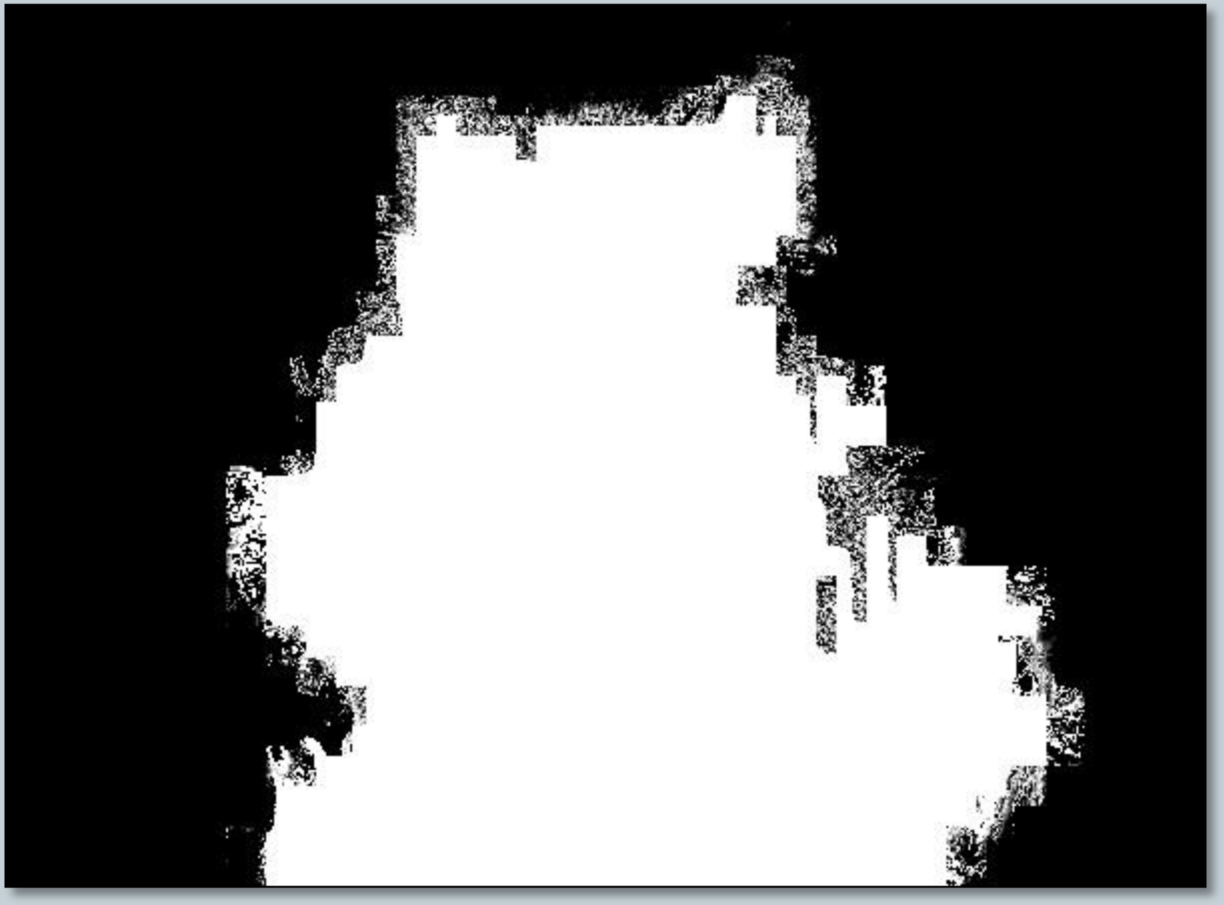
- Where:
  - ω^F is the weigth for the foreground sample
  - ω^B is the weigth for the background sample

Algorithm: end

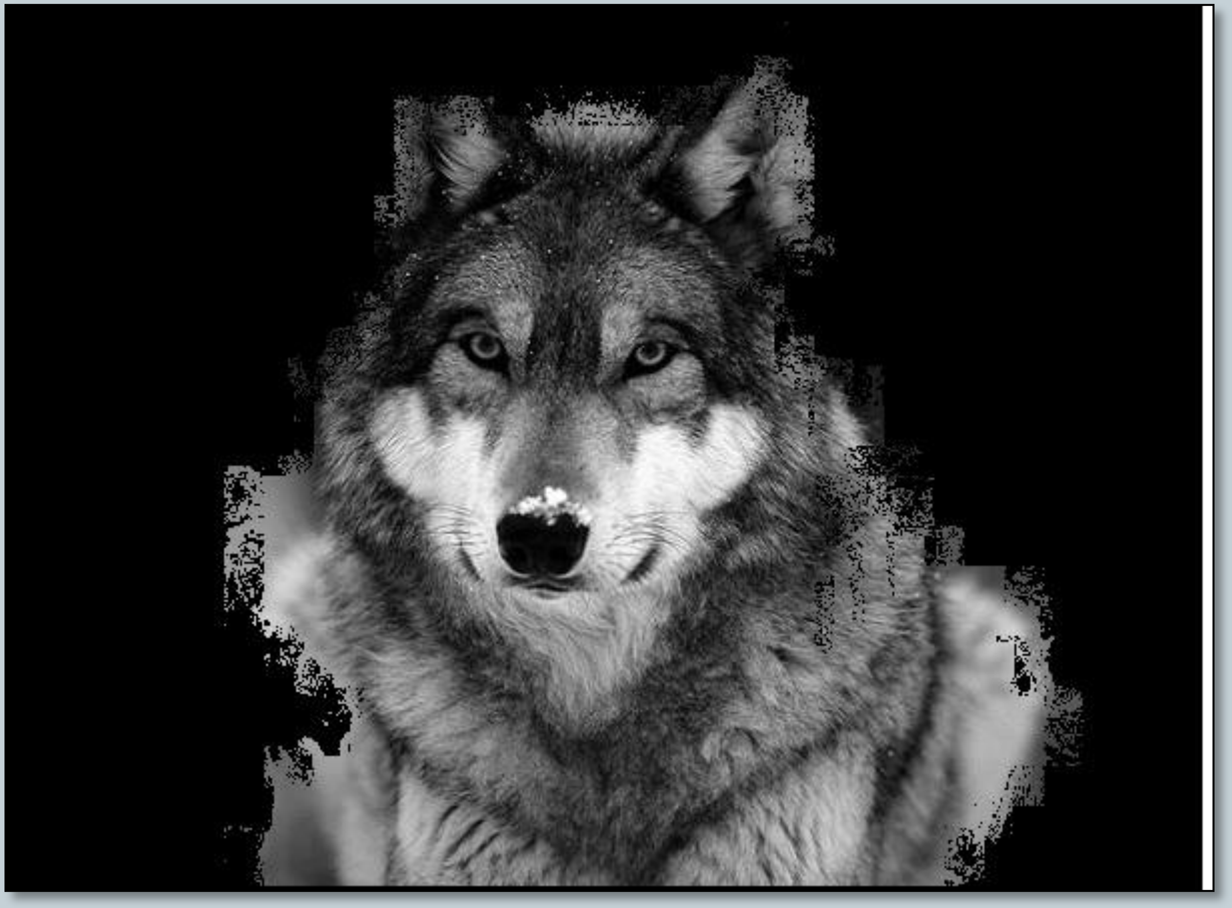
- The algorithm halts when U_n is null (each pixel has been analyzed) and the total uncertainty of the whole matte cannot be reduced any further

…Now, starting from trimap of the three analyzed images, let's show opacity (alpha) map for each one…

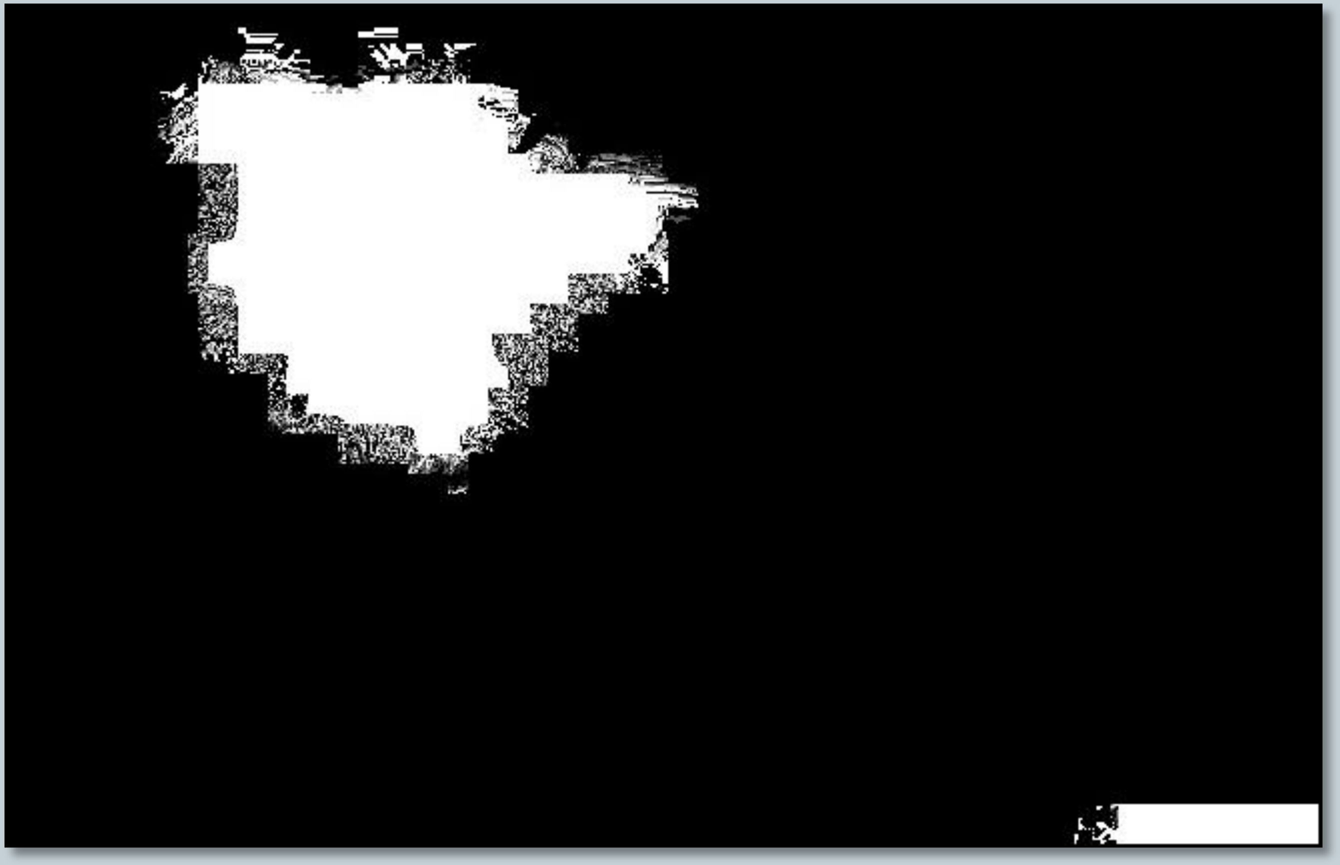
Test 1: extracted matte



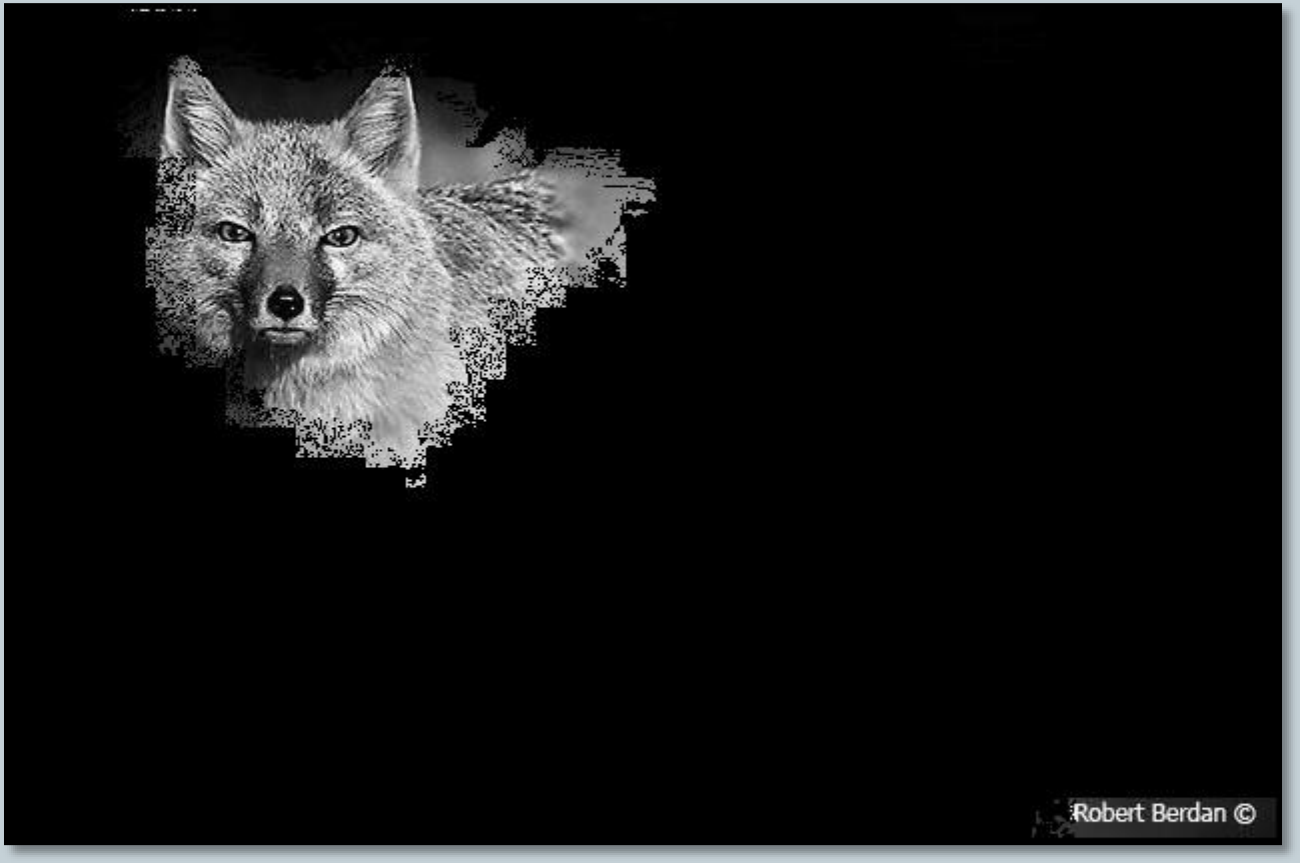
Test 1: foreground



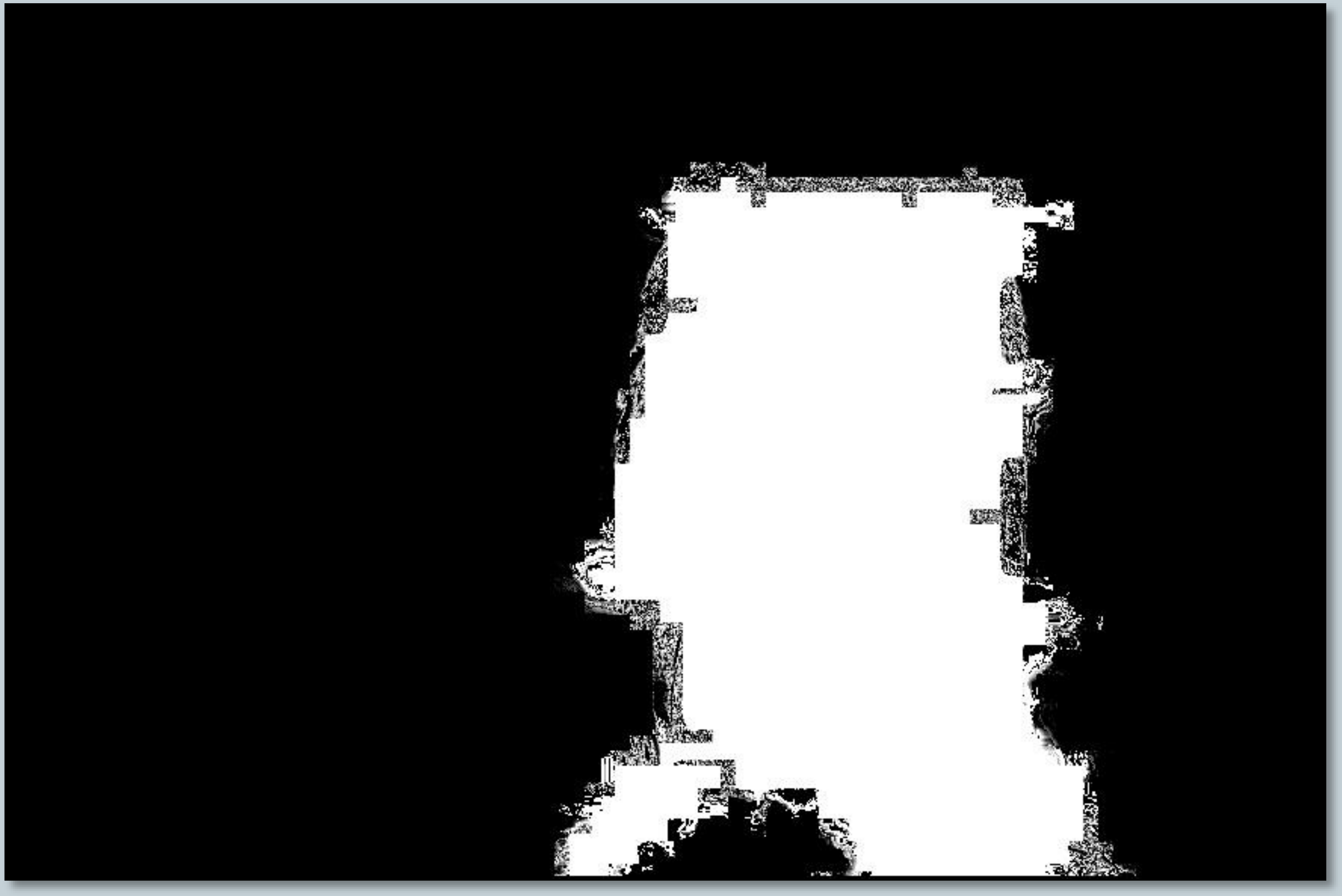
Test 2: extracted matte



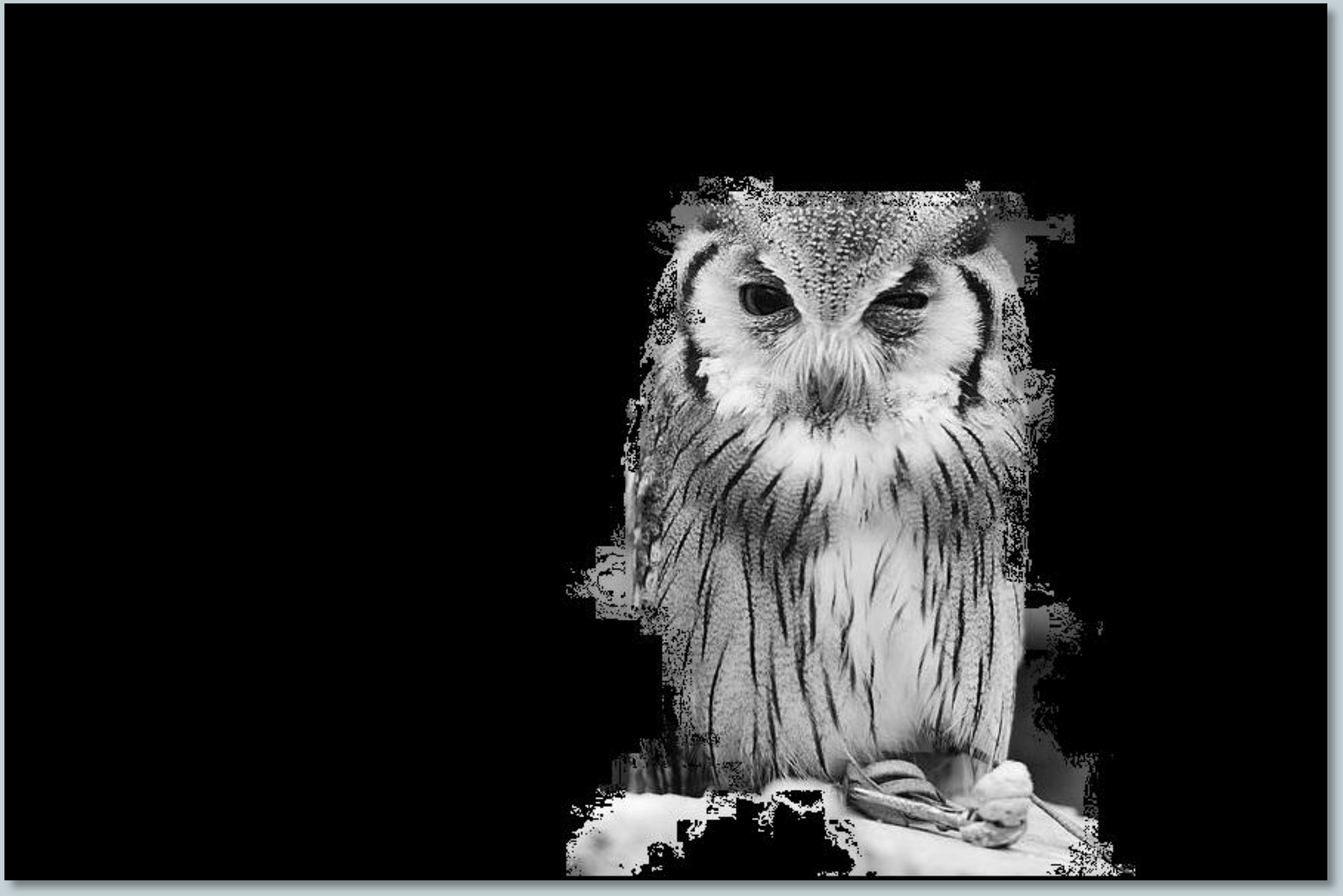
Test 2: foreground



Test 3: extracted matte



Test 3: foreground



Remarks

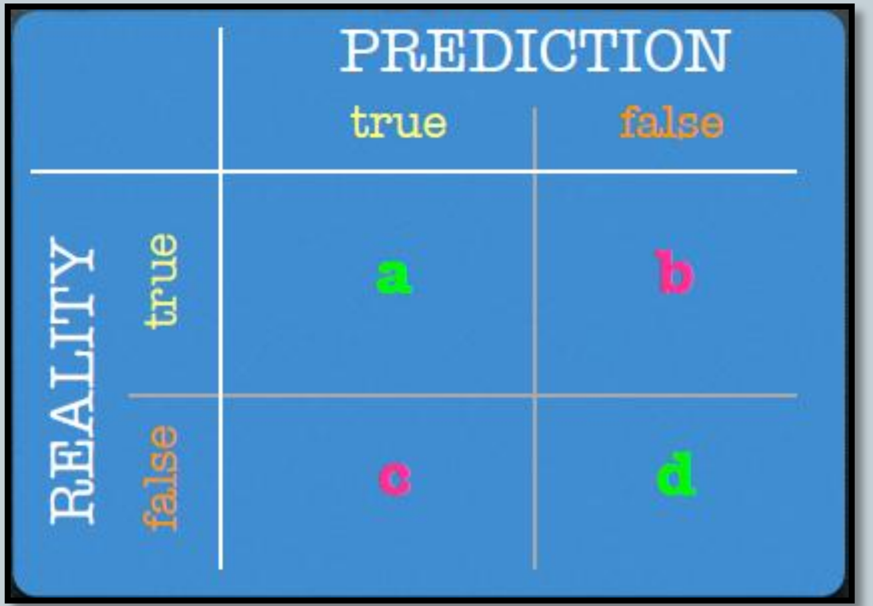
- Results are not fully correct
- However we expected this kind of results, indeed:
  - Original paper has implemented a **smooth operation** we have not considered
  - Slight errors in the trimap…

Some pixels are wrongly considered foreground samples

Effectiveness measures

- Now we want to show numerical results about implemented work
- For this reason we have:
  - searched single thresholds to apply to each image in order to obtain the best trimap
  - effected post-processing operations
  - computed algorithm to a set of 10 images
- Then we have evaluated the efficiency by means of three parameters:
  - accuracy
  - precision
  - recall

Calculating Accuracy, Precision and Recall



		PREDICTION	
		true	false
REALITY	true	a	b
	false	c	d

a = properly observed foreground
b = wrongly observed background
c = wrongly observed foreground
d = properly observed foreground

Calculating Accuracy, Precision and Recall

- Accuracy:

$$\boxed{\frac{a+d}{a+b+c+d}}$$

- Precision:

Foreground $\rightarrow$ $\boxed{\frac{a}{a+c}}$ Background $\rightarrow$ $\boxed{\frac{d}{b+d}}$

- Recall:

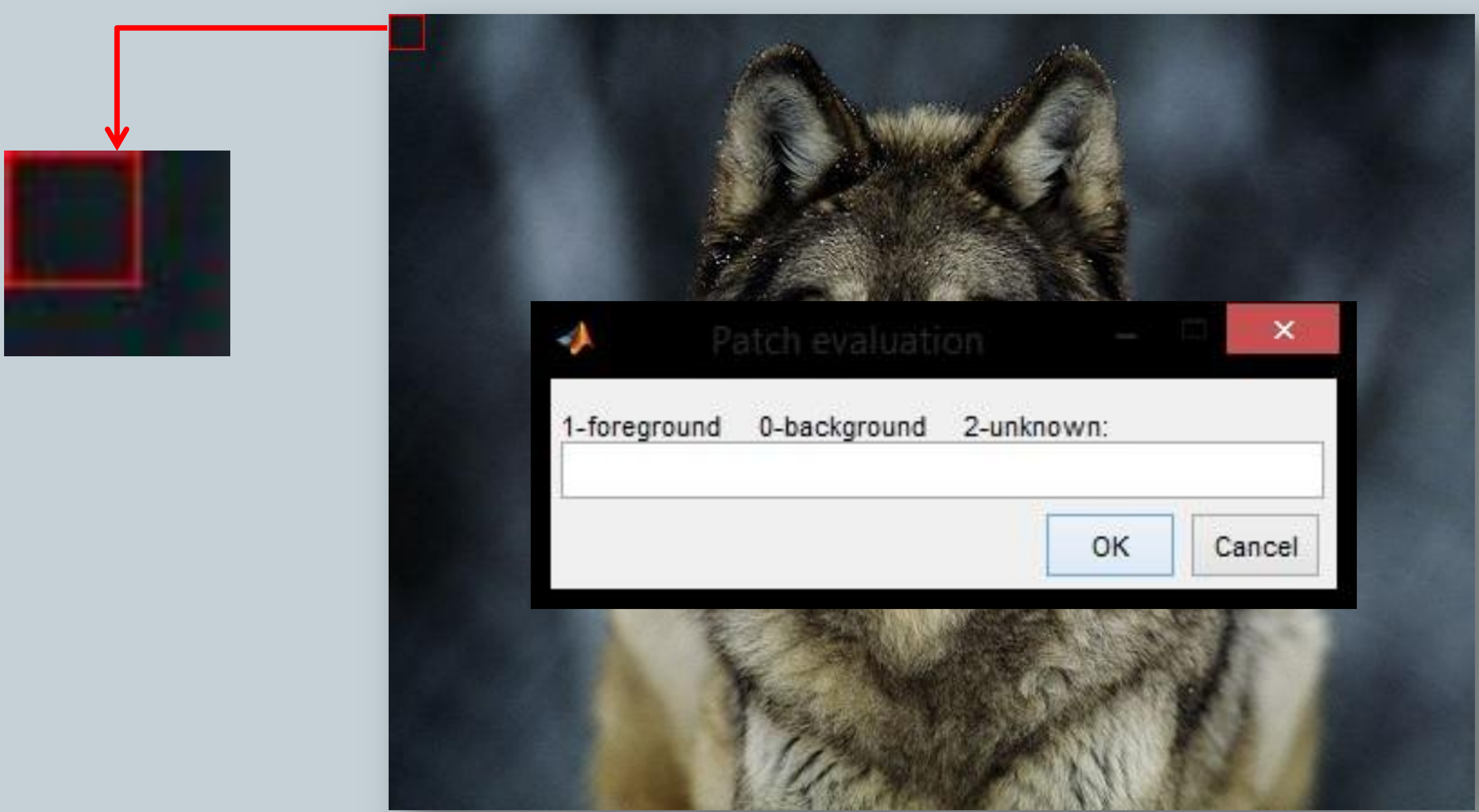
Foreground $\rightarrow$ $\boxed{\frac{a}{a+b}}$ Background $\rightarrow$ $\boxed{\frac{d}{c+d}}$

Ground truth

- In order to obtain the ground truth we have manually selected and segmentd each image into 20x20 patches and we have marked them as foreground or background

- We have created an easy script with Matlab in order to simplify this operation

Ground truth



Parameters

- Thresholds:
  - Gradient histogram span: $\boxed{\begin{aligned} Pb &= 5\% \pm u \\ Pd &= 60\% \pm u \end{aligned}}$
  - Local mean square error: $\boxed{\begin{aligned} Pb &= 10\% \pm u \\ Pd &= 60\% \pm u \end{aligned}}$
  - Maximum saturation: $\boxed{\begin{aligned} Pb &= 35\% \pm u \\ Pd &= 12\% \pm u \end{aligned}}$

$\boxed{u = 0.02}$

Parameters

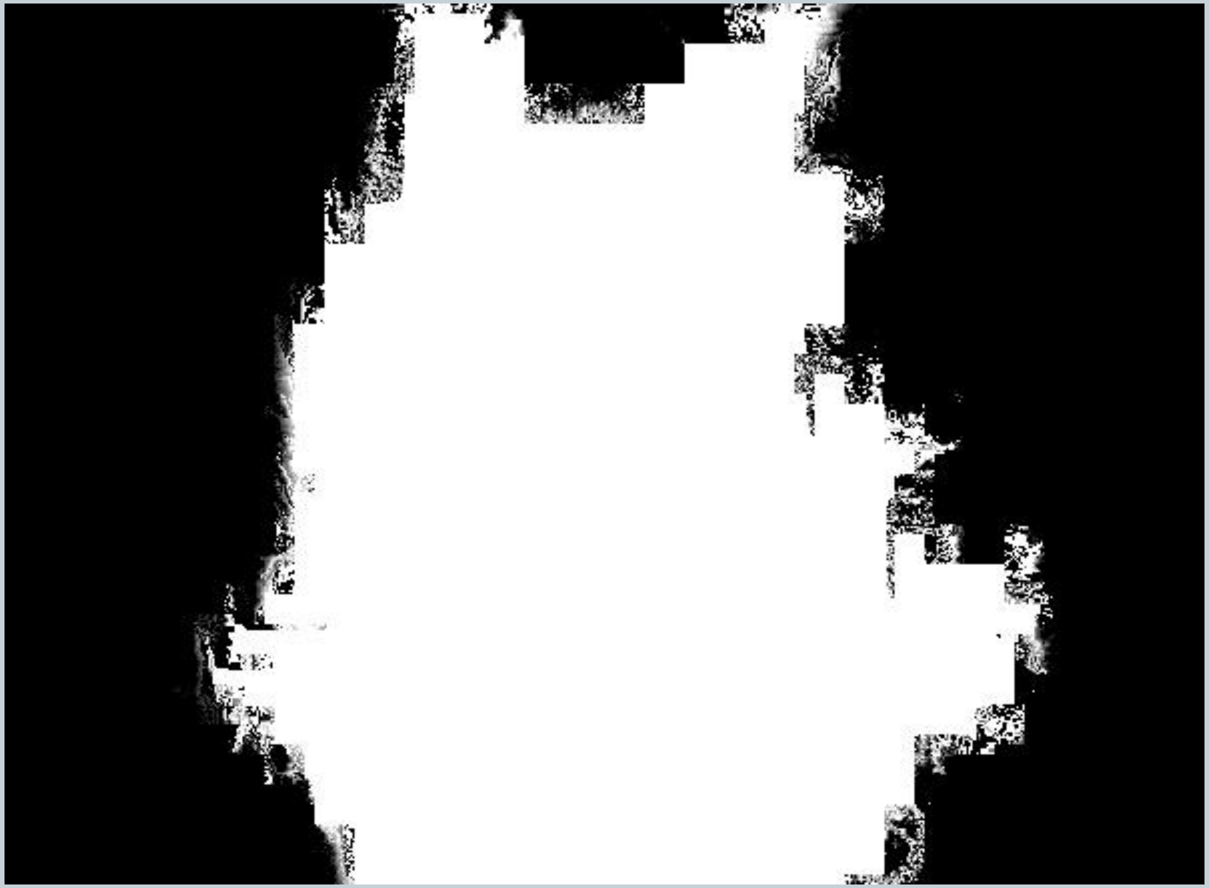
- In order to obtain a better trimap we've applied a change
- We have assigned to each patch a counter:
  - if a patch is marked as foreground by one of the three features then counter is increased by one
  - if a patch is marked as background by one of the three features then counter is rediced by one
- When a patch's been evaluated by all the three features:
  - if counter = 1 then $\alpha = 2/3$ and $u = 2/3$
  - if counter = 2 then $\alpha = 2/3$ and $u = 1/3$
  - if counter = 3 then $\alpha = 1$ and $u = 0$
  - if counter = -3 then $\alpha = 0$ and $u = 0$
  - else $\alpha = 1/2$ and $u = 1$

Post-processing

- Furthermore we have operated some enhancement on the extracted matte, in order to obtain a more truthful result
- In particular we have applied on the extracted matte:
  - A 3x3 mean filter
  - Segmentation
  - The morphological operation closing
  - The morphological operation opening

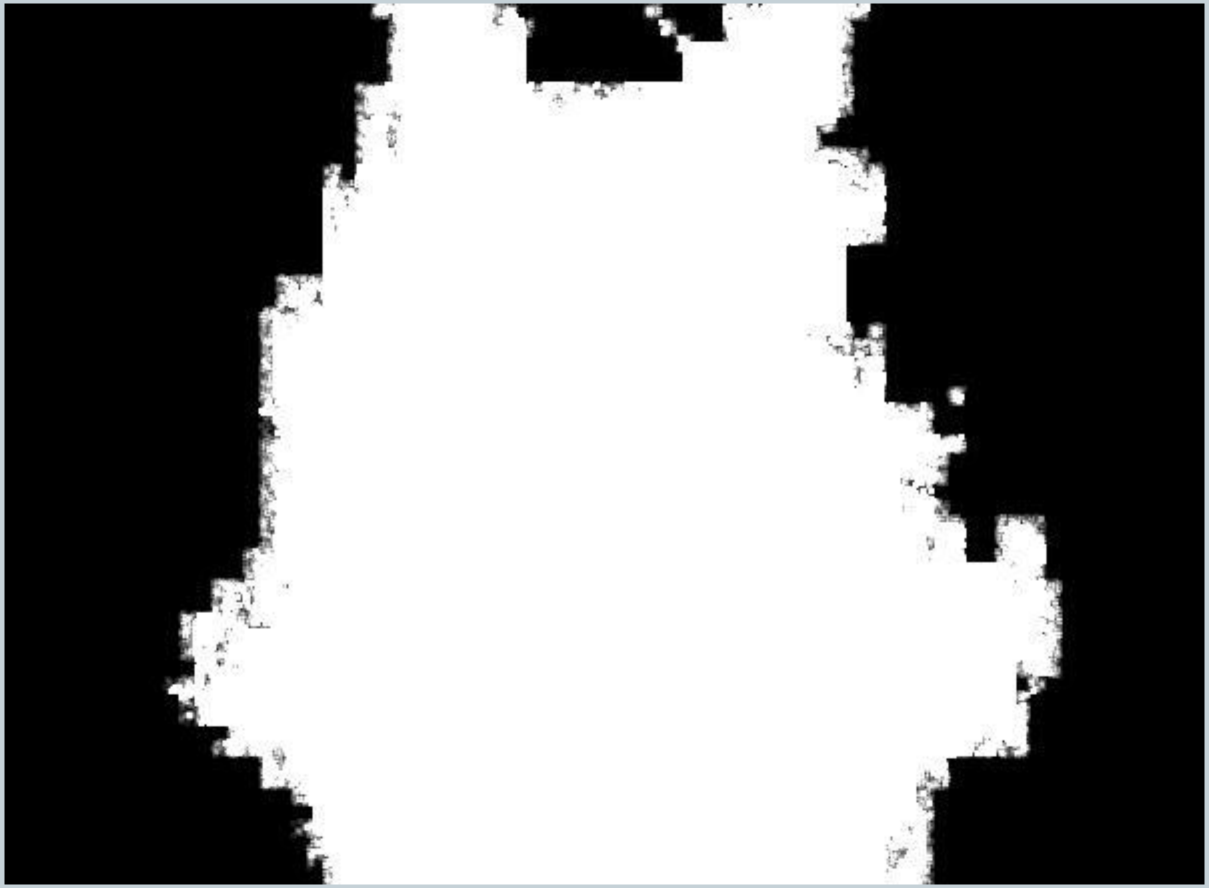
Post-processing

- Original extracted matte



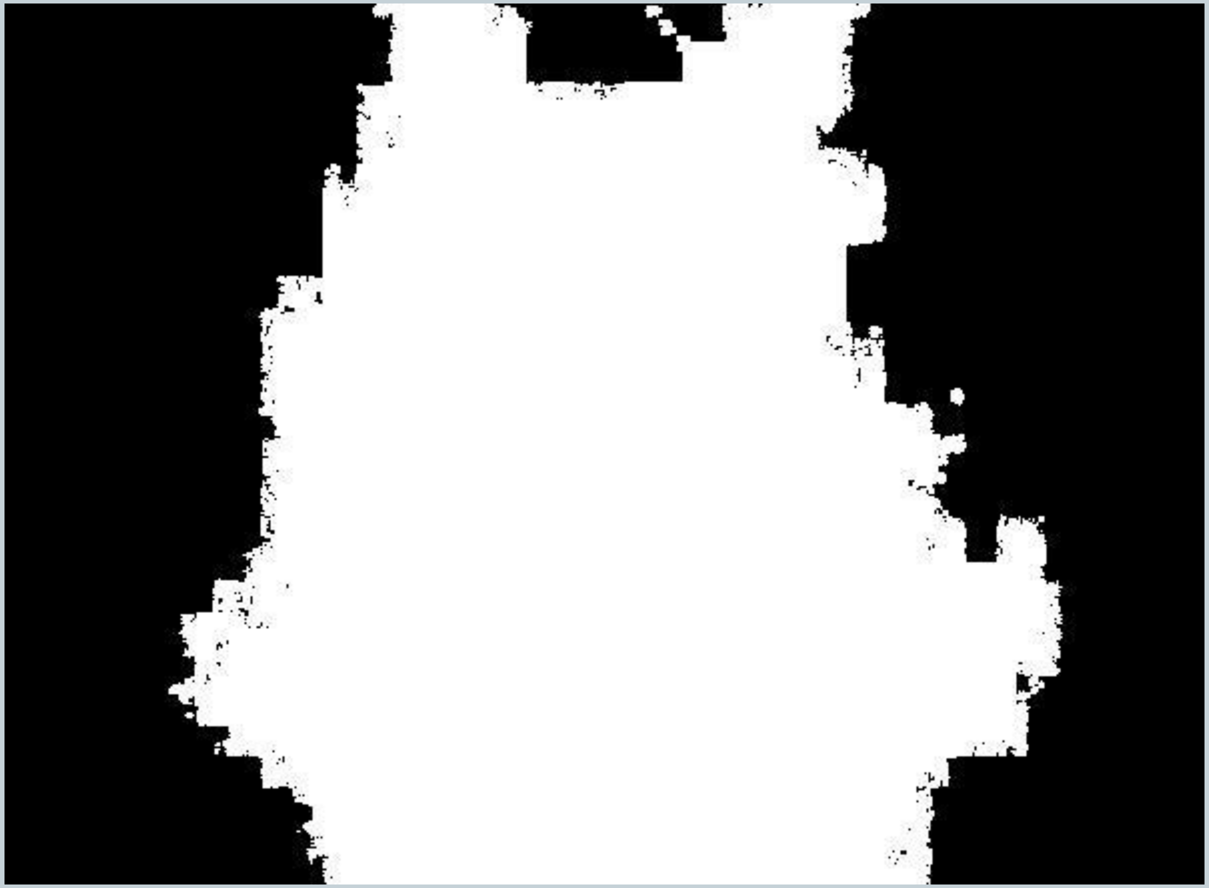
Mean filter

- Uniforming the opacity map



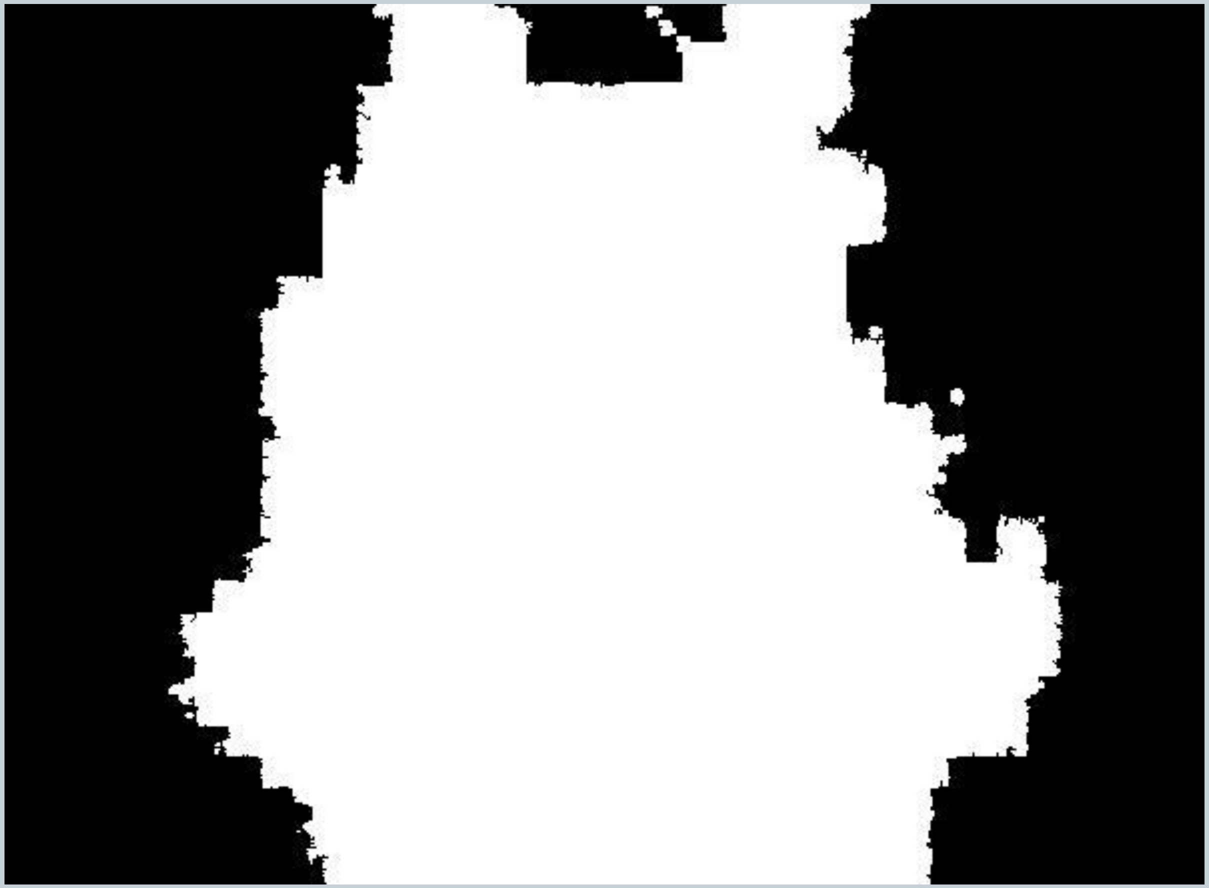
Segmentation

- Separating the foreground from the background



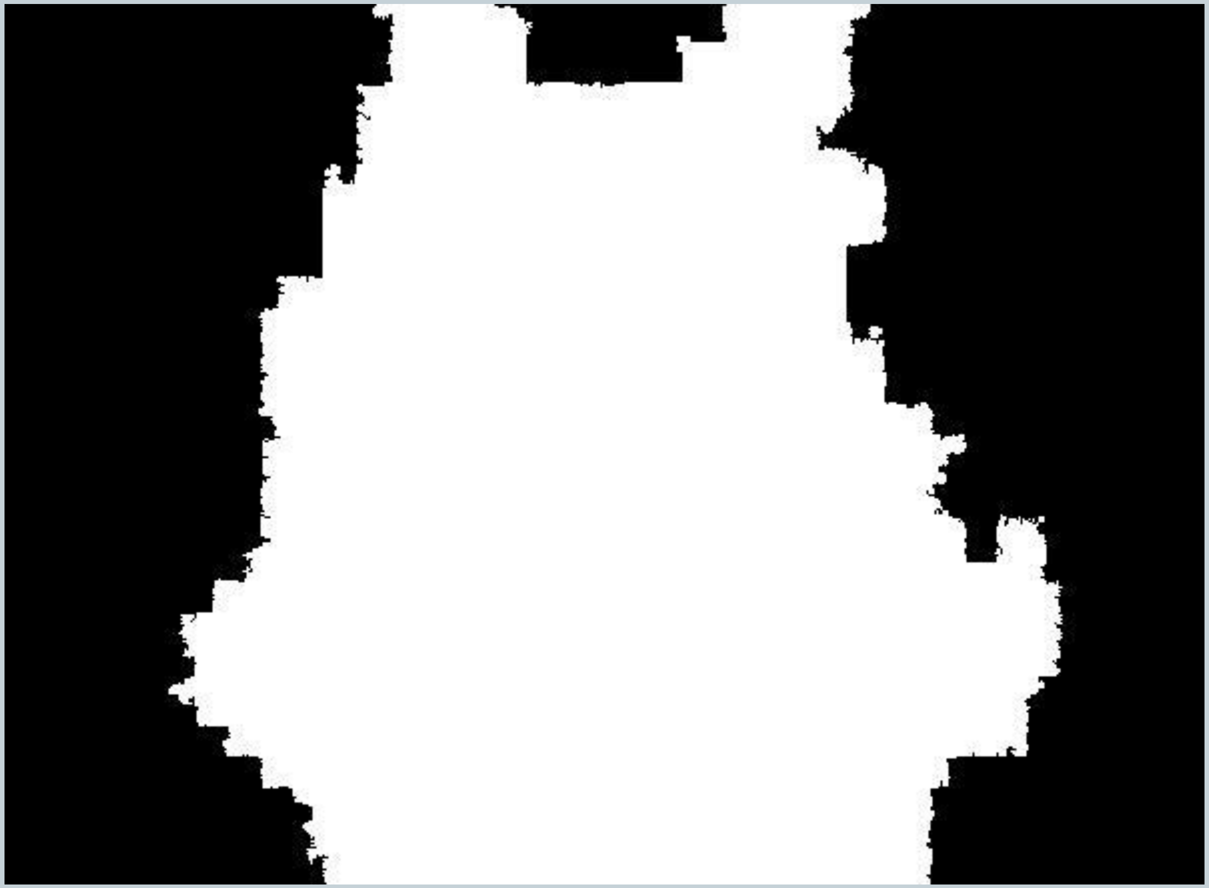
Closure

- Removing of "holes" from the foreground

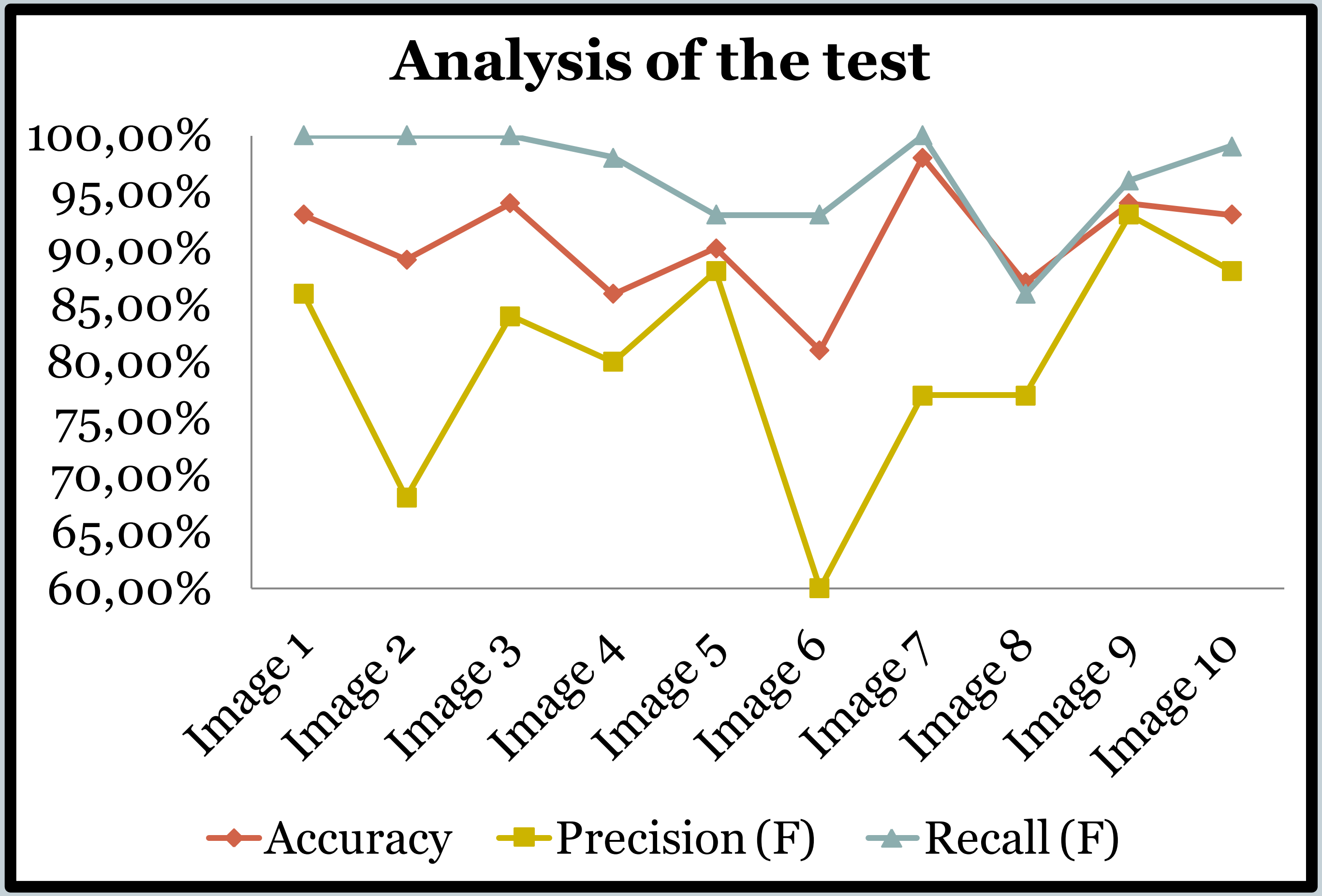


Opening

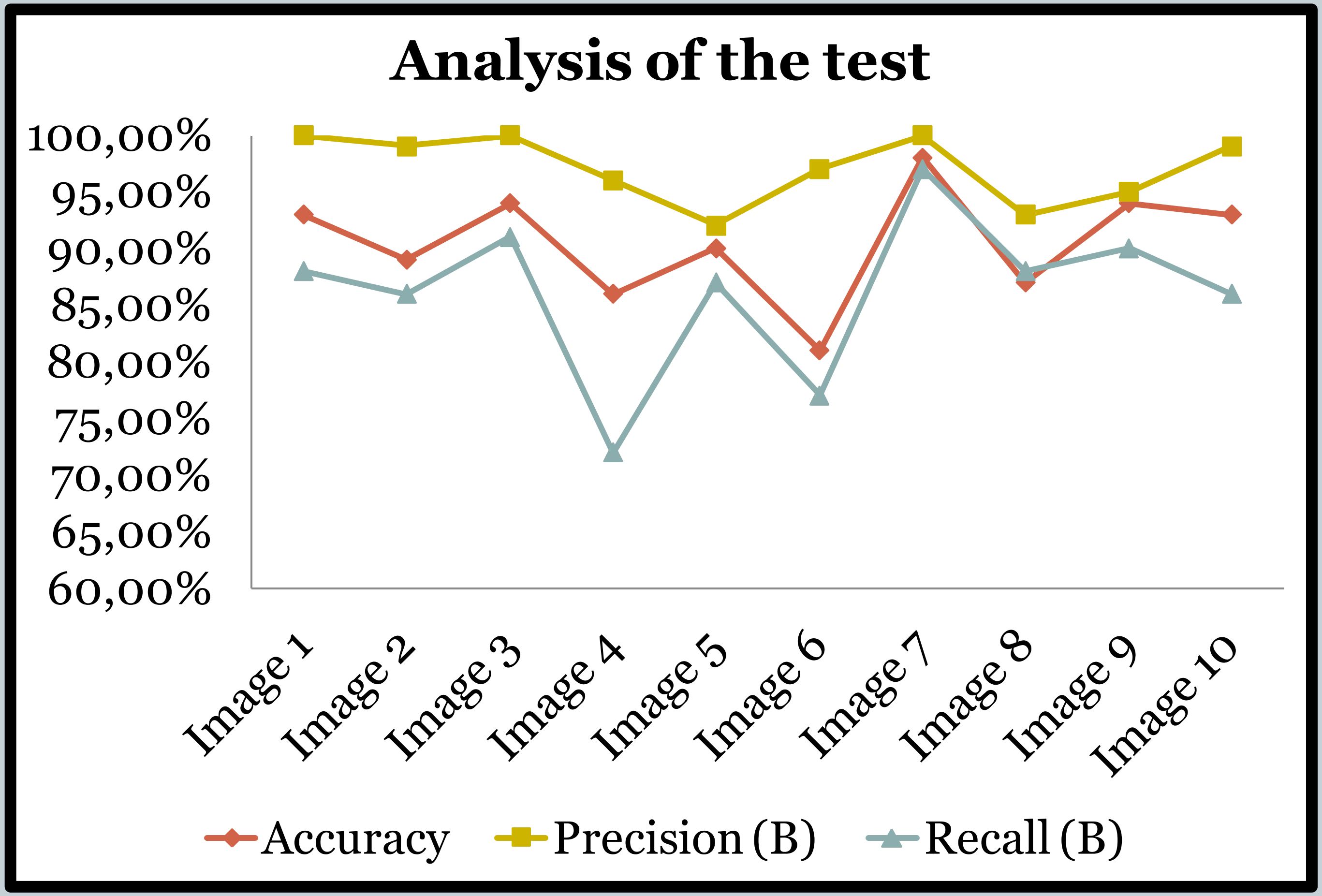
- Removing small objects from background



Final results



Final results



Final results

- Here we show the overall data of accuracy -precision - recall considering the combination of all the images:

Accuracy	Precision Of Foreground	Recall Of Foreground	Precision Of Background	Recall Of Background
90%	84,5%	87,5%	93%	91,5%