Lab 2
Energy Efficient Displays

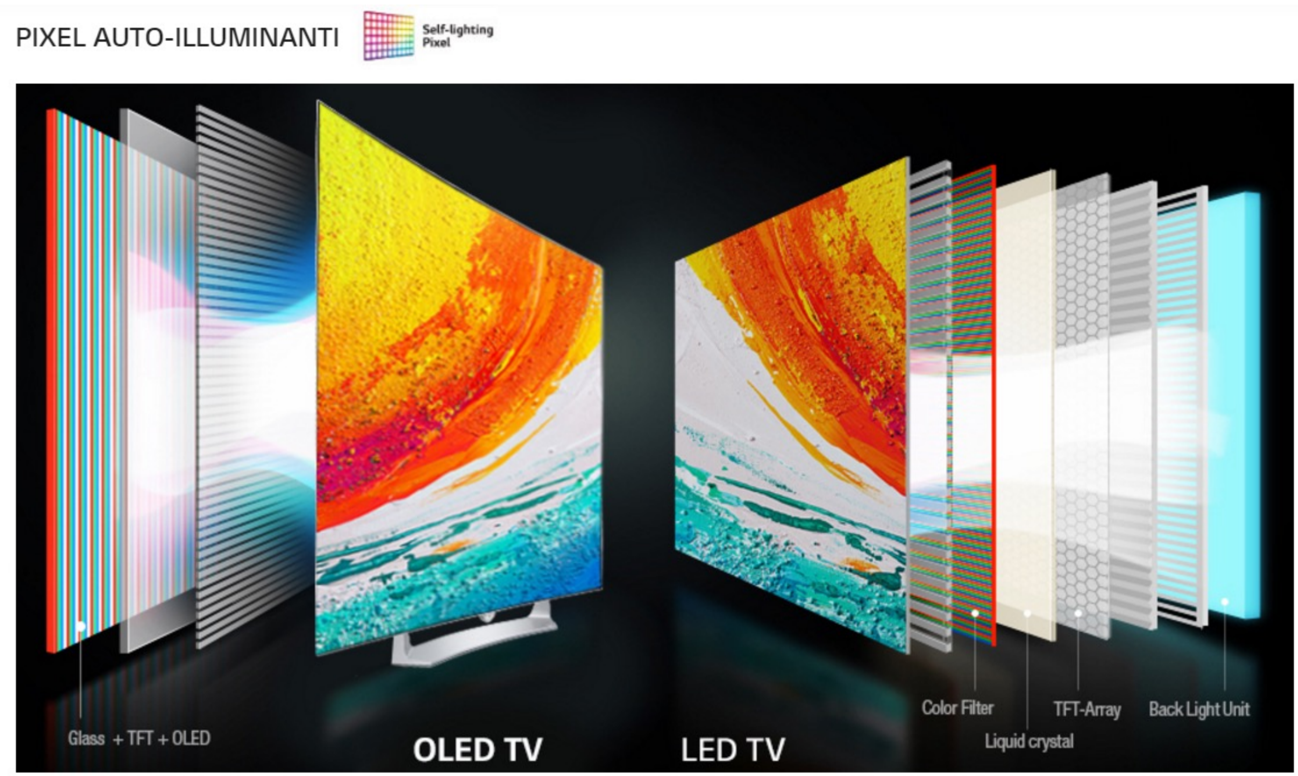
Objective and organization

- Demonstrates how manipulation of an image can be used to tradeoff image quality and power saving in emissive displays
  - 1 report – 2 days
  - Python

- Organize all implemented methods in functions and scripts to **automatically** test and evaluate all images and all techniques

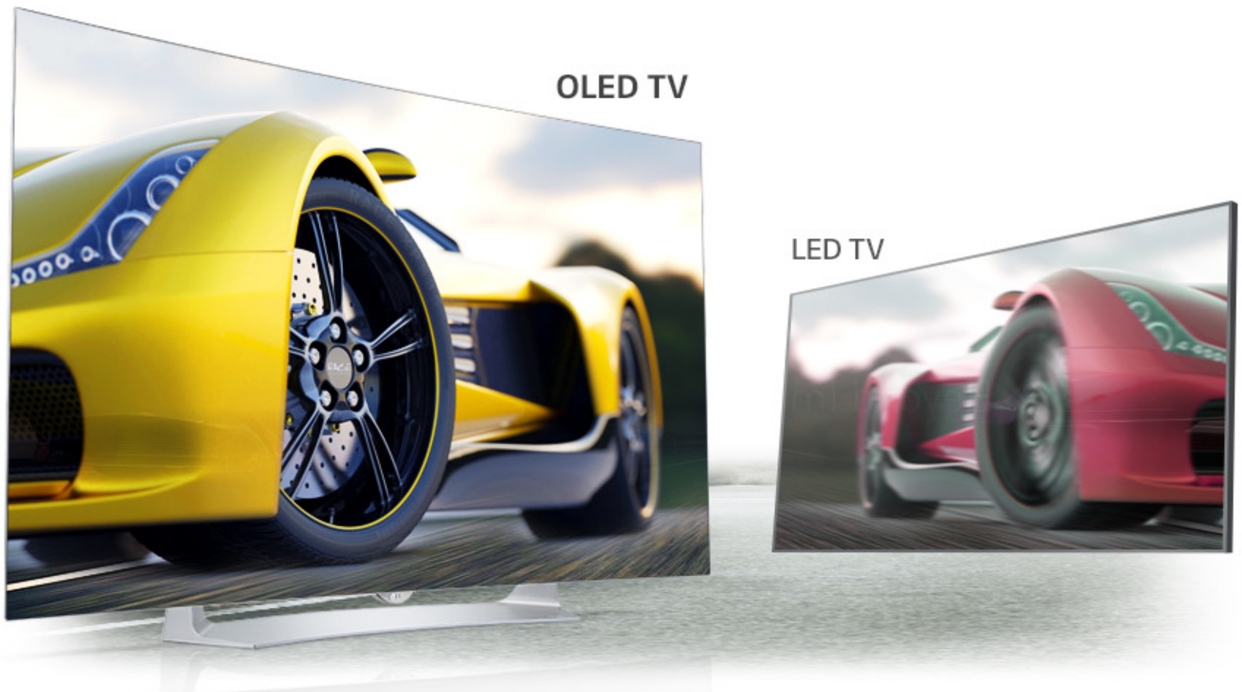
OLED vs LED

- OLED TVs
  - Do not require external lighting
    - Better black levels



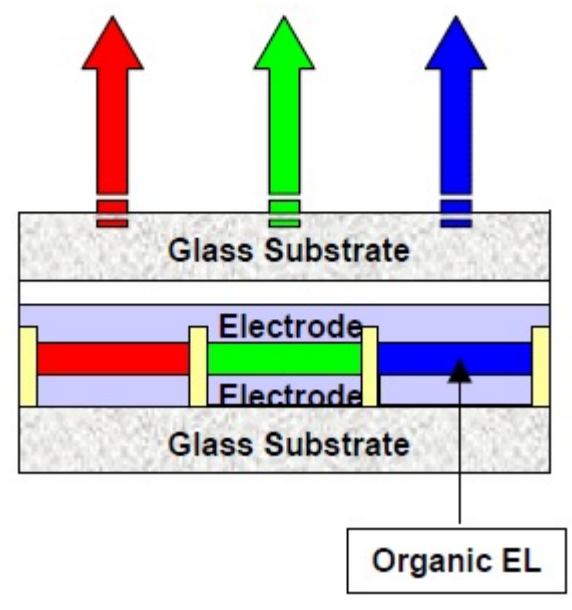
OLED vs LED

- OLED TVs
  - Pixels are independent from each other
    - More sense of depth
    - Higher contrast makes images more realistic



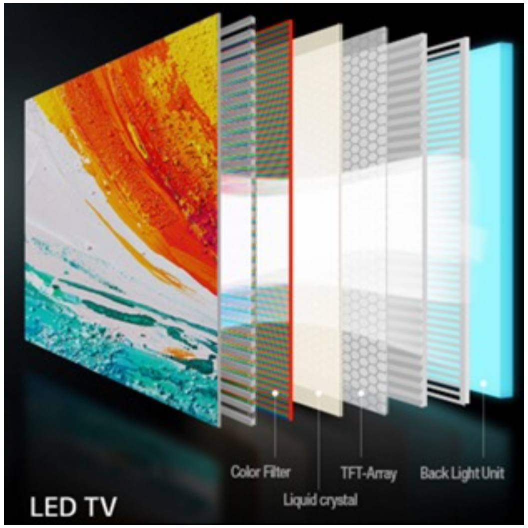
OLED

- Interesting case study from our perspective…
- Organic light-emitting diode (OLED)
  - Do not require external lighting
  - Pixels are emissive
    - Emissive layer is a film of organic compound which *emits light in response to an electric current*
  - Each pixel is made of three devices corresponding to red, green and blue components



OLED

- In LCDs, backlight dominates power consumption and color has only negligible power impact



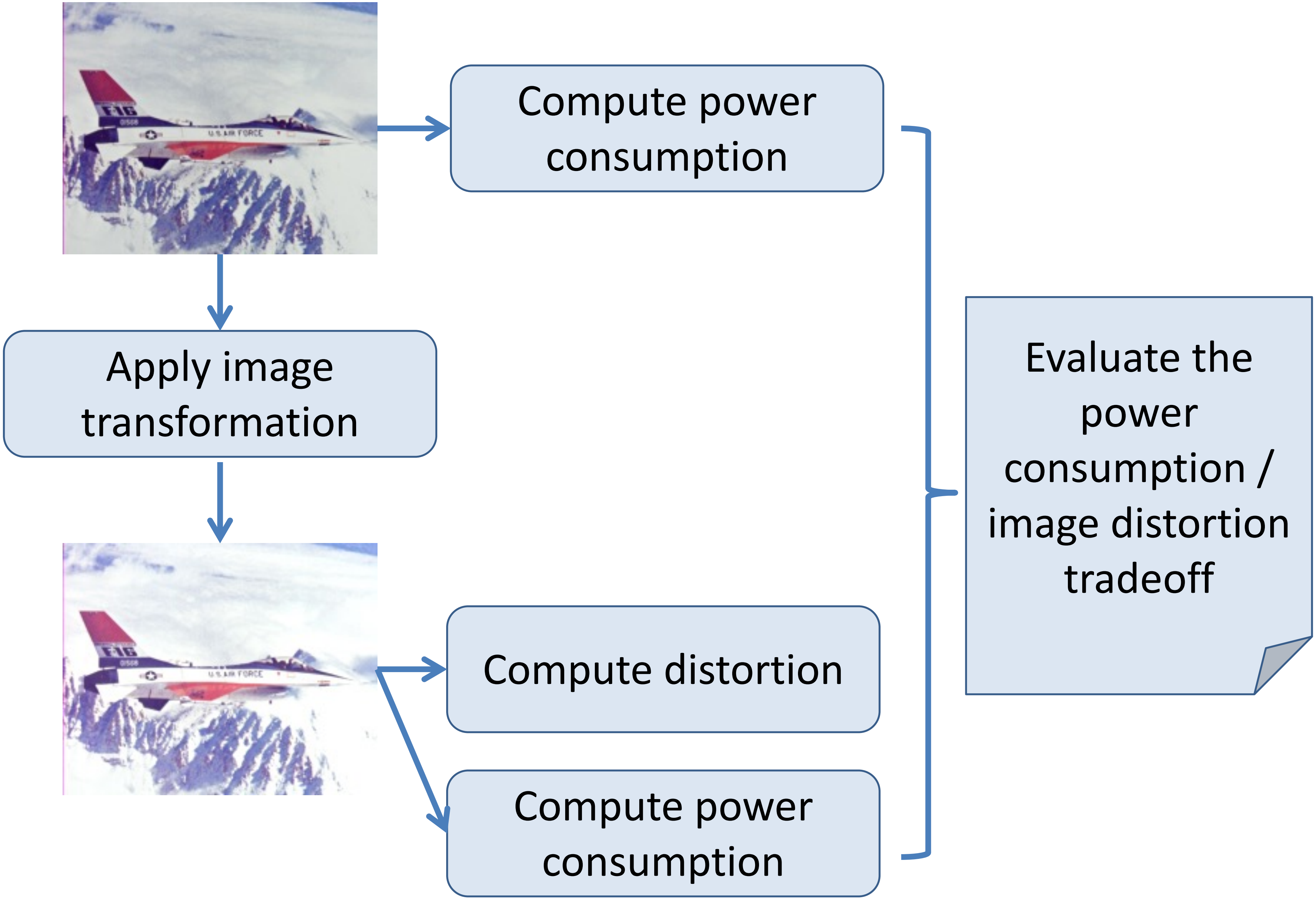
- With OLED displays, the color of a pixel impacts on power consumption
  - E.g., hungry blue
  - Different luminance efficacies
  - Different images imply different power consumption

Day 1
Energy efficient image processing

OLED

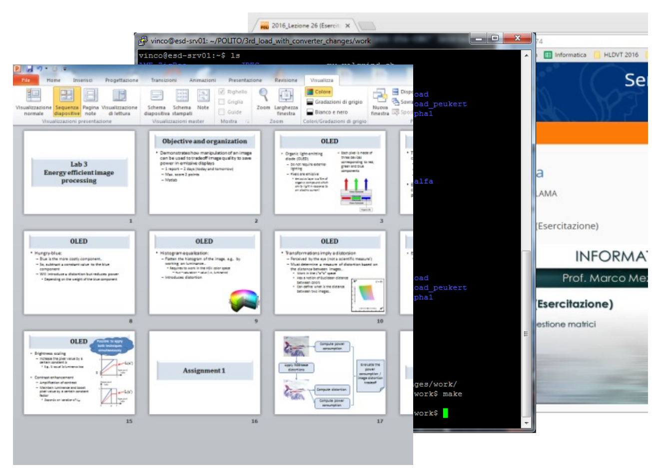
- Power consumption depends on color components of a pixel…
  - So we can save power by changing the spectrum of the image!
  - First class of power saving methods:
    - Change pixel color
    - Given a certain tolerance level on color distorsion

Assignment 2 – Part 1


Compute power consumption
Apply image transformation
Compute distortion
Compute power consumption
Evaluate the power consumption / image distortion tradeoff

1. Identification of images

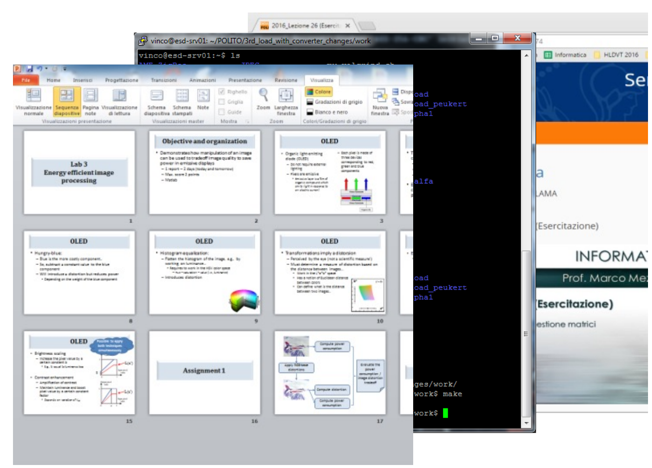
- Test images will be:
  - The images from the USC SIPI database
    - http://sipi.usc.edu/database/database.php?volume=misc
  - The images from the BSDS500 training set
    - https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/BSR/BSR_bsds500.tgz
  - 5 images representing screenshots of your computer
- Different colors and characteristics…



1. Identification of images

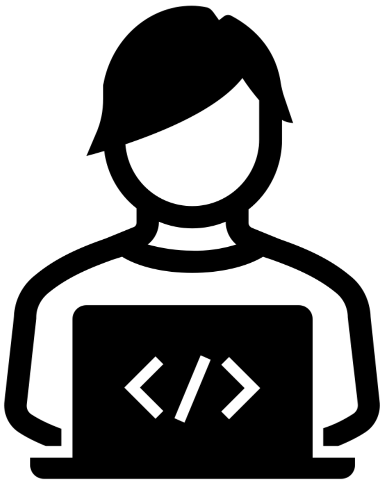
- Test images will be:
  - The images from the USC SIPI database
    - http://sipi.usc.edu/database/database.php?volume=misc
  - The images from the BSD 00 training set
    - n/grouping
  - 5 r
- Diff colors and characteristics…

Some grayscale images are present, you can skip them!!



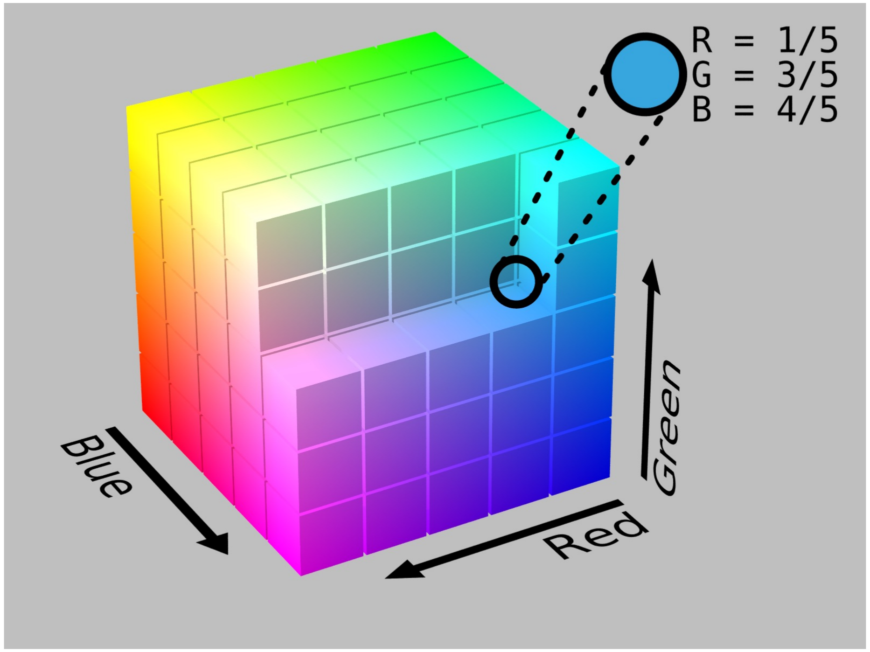
2. Manipulation of images

- Experiments require to adopt different color spaces…
- **TASK:** Learn how to:
  - Import the image
    - `Image.open()` function
  - Extract the R, G, B channels
  - Convert between different color spaces



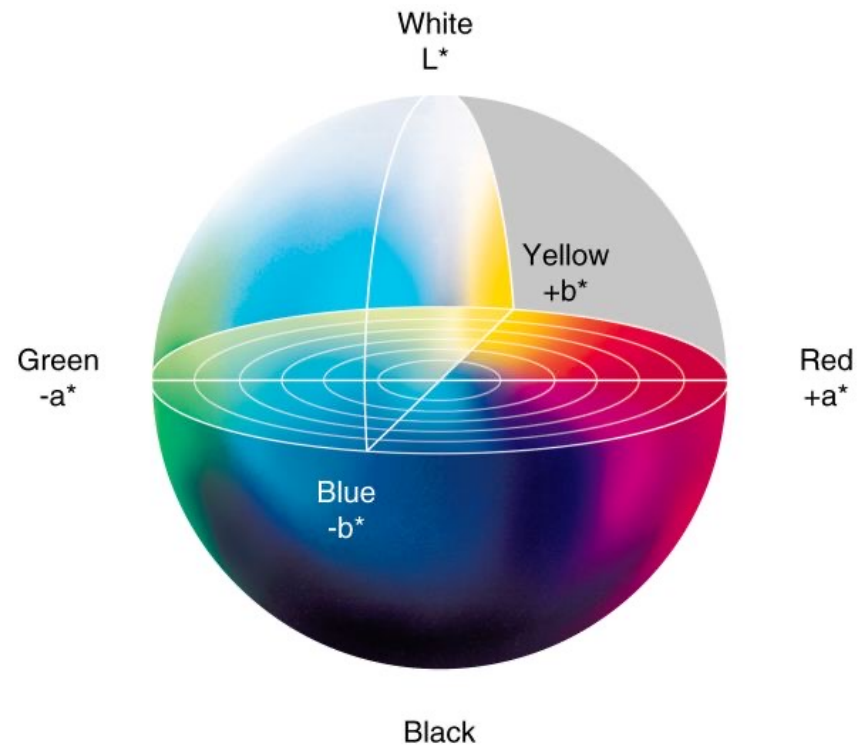
2. Manipulation of images

- RGB
  - Additive color space
    - All possible colors that can be made from three colorants for red, green and blue
  - Stores individual values for red, green and blue
  - Convenient color model for computer graphics as it is similar to the human visual system
    - Used in LCDs



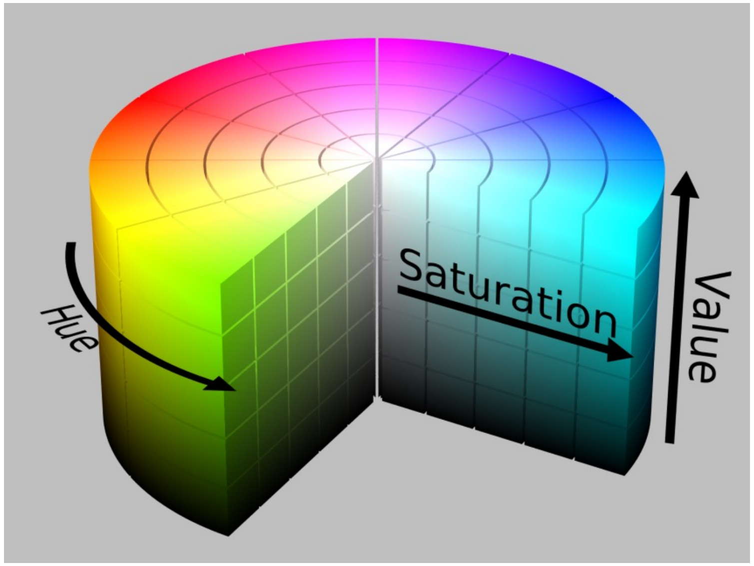
2. Manipulation of images

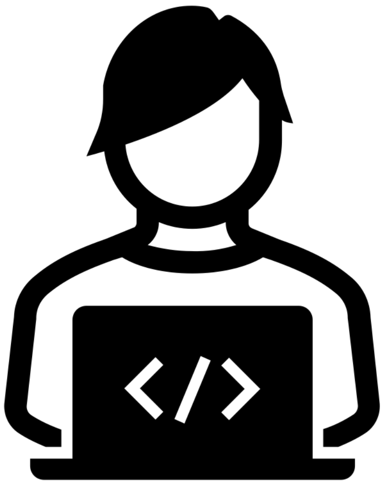
- Lab
  - One channel for luminance (L) and two color channels (a and b)
  - Includes all perceivable colors
    - Super-set of RGB
  - The space is a three-dimensional Real number space
    - Allows the definition of Euclidean distance

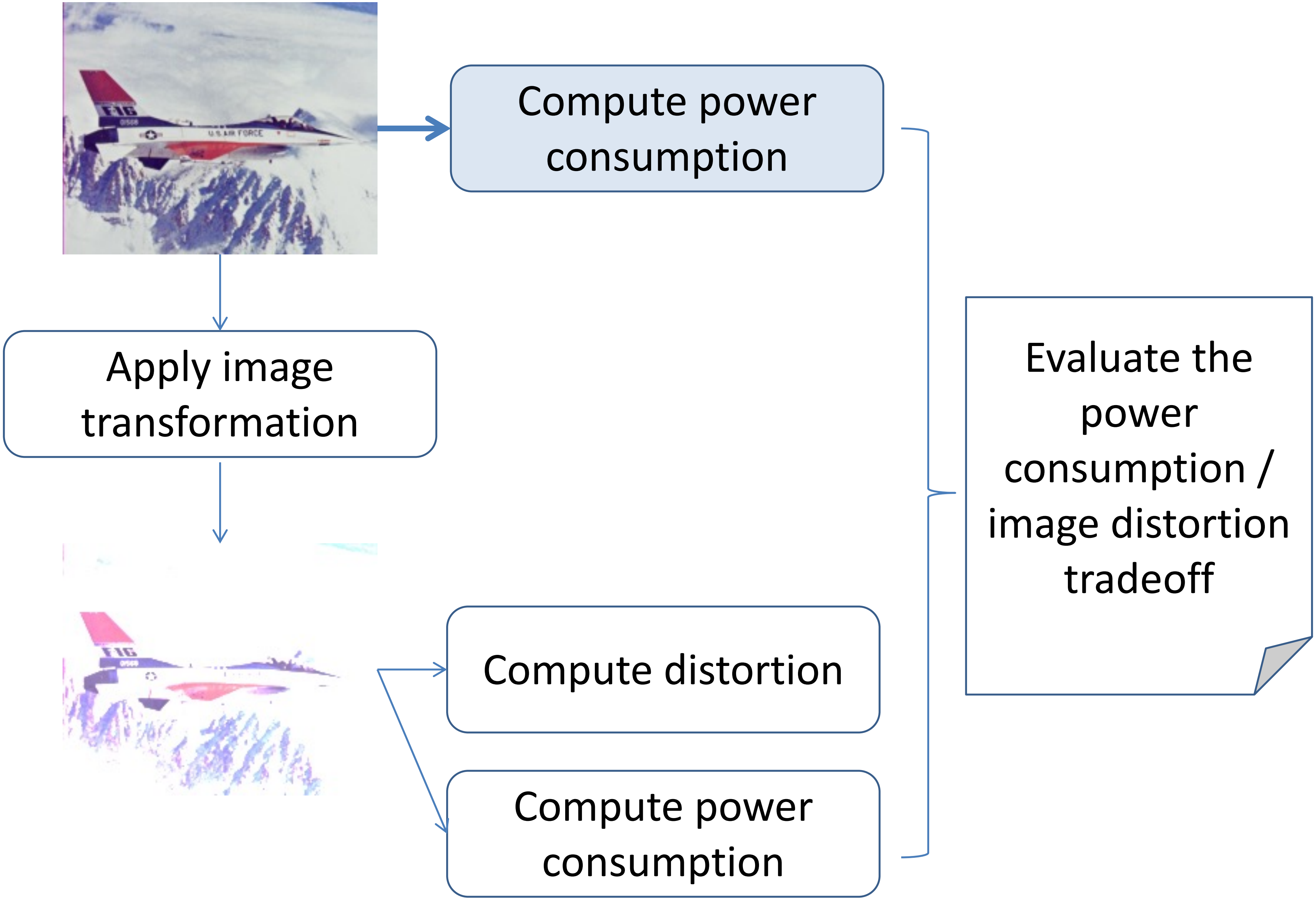


2. Manipulation of images

- HSV
  - Hue
    - Perceived color
  - Saturation
    - Colorfulness, amount of white component
  - Value
    - Brightness
  - Cylindrical-coordinate representations of points in an RGB color model
  - Widely used in computer graphics





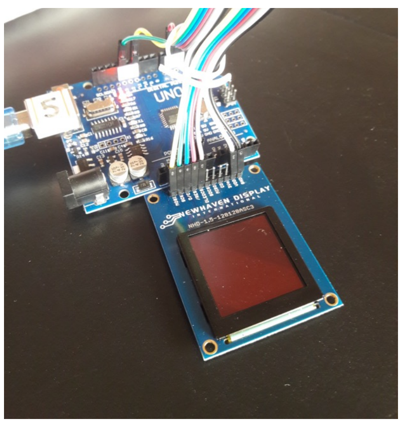
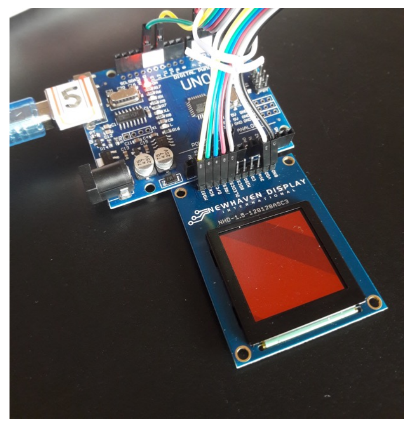
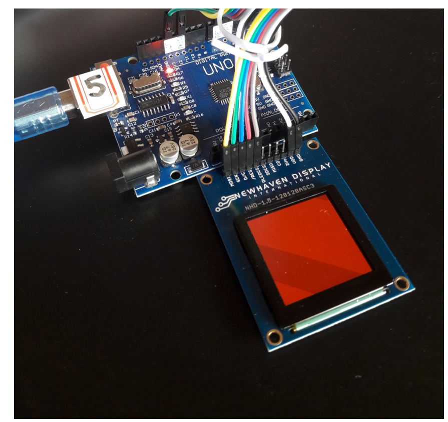

Compute power consumption
Apply image transformation
Compute distortion
Compute power consumption
Evaluate the power consumption / image distortion tradeoff

3. Evaluation of power consumption

- Power model
  - $P_{pixel} = f(R) + h(G) + k(B)$
    - Depends on pixel color in terms of RGB components
    - f, h and k determined experimentally by:
      - Setting black screen to estimate C
      - For f, set G and B components to 0 and vary R component
      - Similar for h and k
  - $P_{image} = C + \sum_{i=1}^{n}\{f(R_i) + h(G_i) + k(B_i)\}$
    - Sums up power contributions of single pixels
    - C static power independent of pixel values

1. Evaluation of power consumption

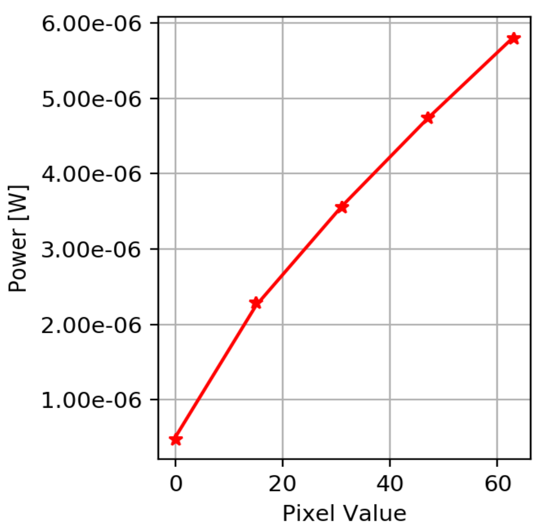
- Experimental Setup:
  - Show monochromatic images with different RGB values on the OLED, e.g.:

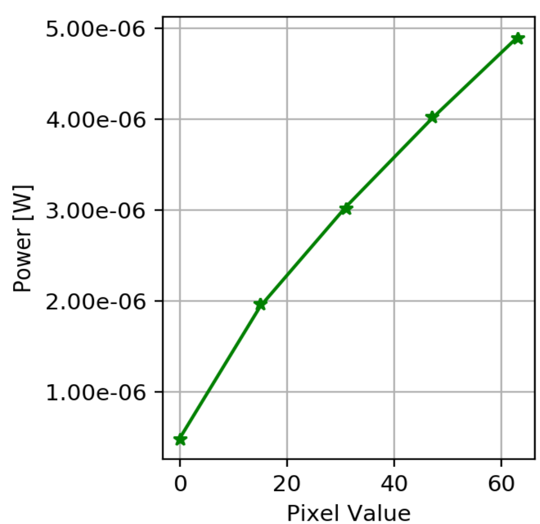




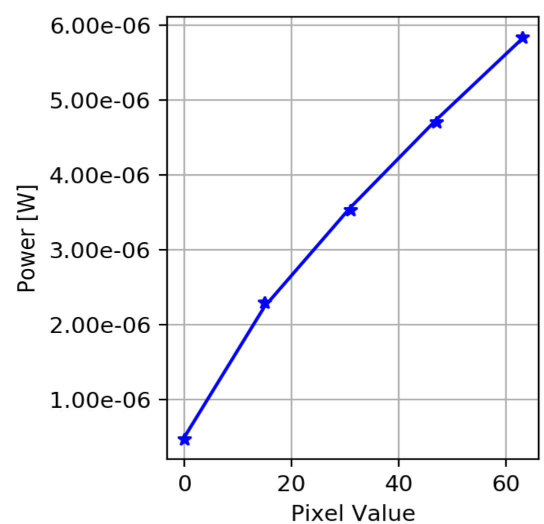
  - Measure power supply current (and convert to power)

1. Power model for the provided OLED (cont'd)

- Interpolation:
  - Find regression model type that fits best the data and determine the corresponding parameters



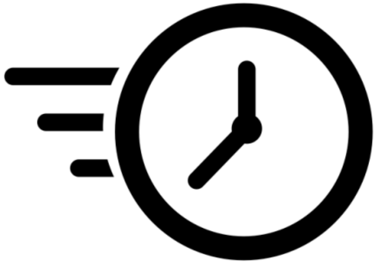




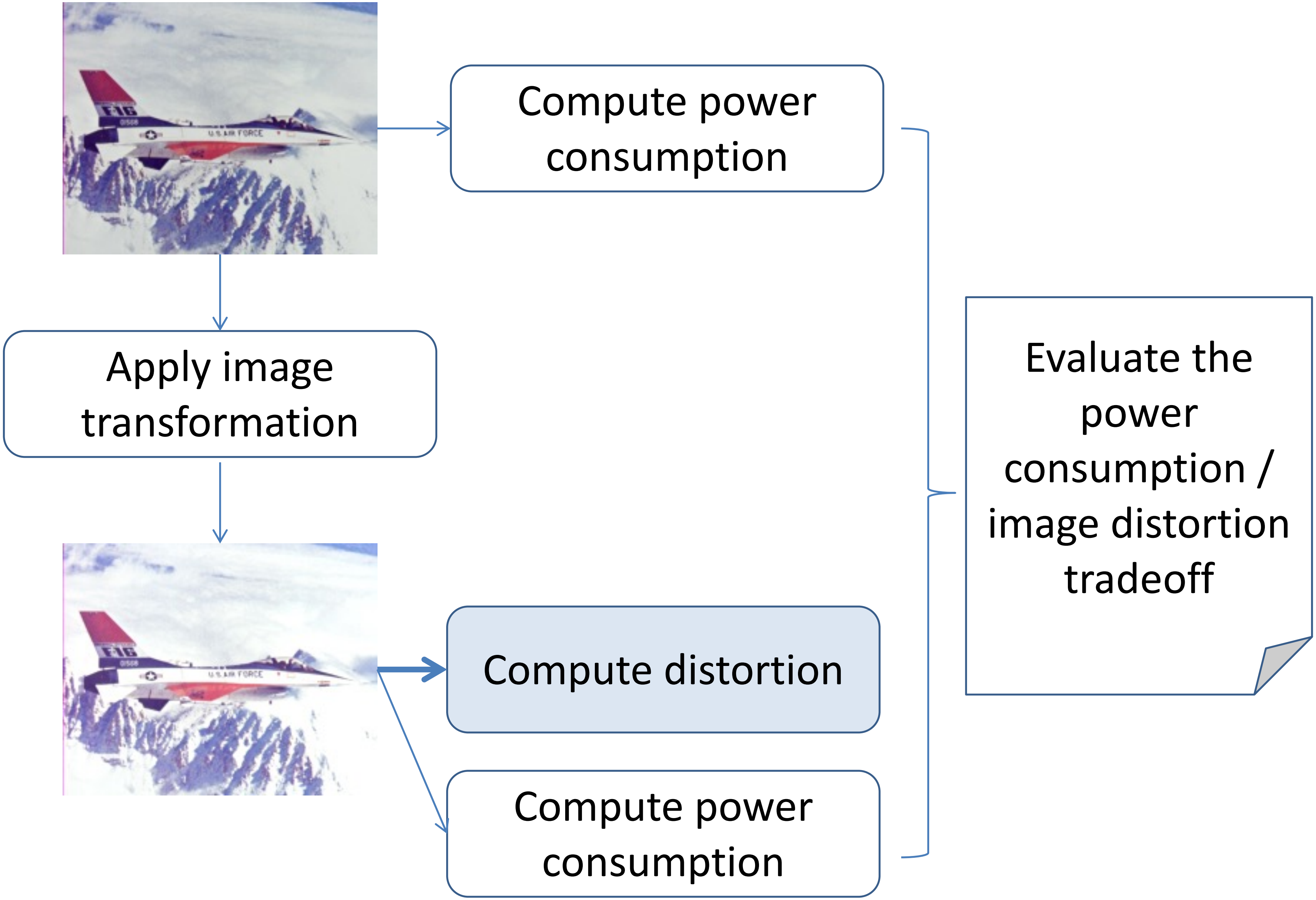
  - For this Lab, we used a model format from literature (see next slide)

Assignment 2 – Part 1

- **TASK:** Define a Python function that estimates power consumed to display an image
  - $P_{pixel} = w_R * R^{\gamma} + \ w_G * G^{\gamma} + w_B * \ B^{\gamma}$
  - $P_{image} = w_0 + \ \sum_{i=1}^{n}\{P_i(R,G,B)\}$
  - R, G, B are pixel values between 0 and 255

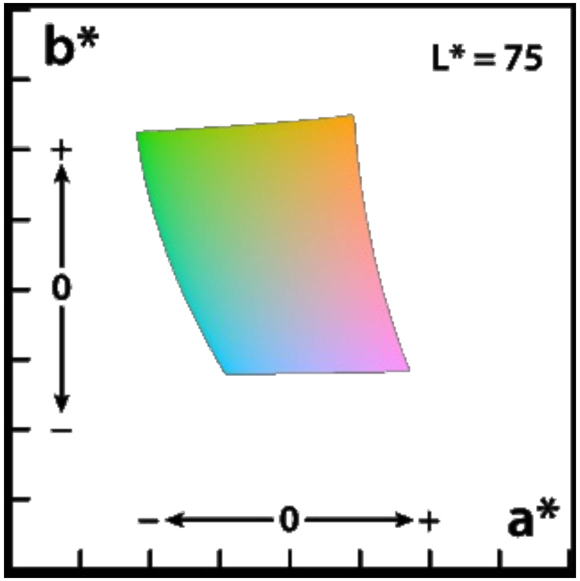


γ	W_0	W_R	W_G	W_B
0.7755	**$1.48169521*10^{-6}$**	**$2.13636845*10^{-7}$**	**$1.77746705*10^{-7}$**	**$2.14348309*10^{-7}$**


Compute power consumption
Apply image transformation
Compute distortion
Compute power consumption
Evaluate the power consumption / image distortion tradeoff

4. Evaluation of image distortion

- Transformations imply a distorsion
  - Must determine a measure of distortion based on the *distance* between images…
    - We will work in the L*a*b* space
    - Has a notion of Euclidean distance between colors that well matches the perceived distortion
    - Can define what is the distance between two images..
  - Importantly, distortion is different from perceived **visual quality**, which is subjective, not a scientific measure!



4. Evaluation of image distortion

- Evaluation of image distortion
  - Difference between two images
    - $\varepsilon\left(image_i, image_j\right) =$
      $\sum_{k=1}^{N}\left(\sqrt{(L_{i,k} - L_{j,k})^2 + (a_{i,k} - a_{j,k})^2 + (b_{i,k} - b_{j,k})^2}\right)$
    - N = number of pixels
    - k = k^{th} pixel
    - Pixel per pixel, compute the difference of L, a and b components between the two images

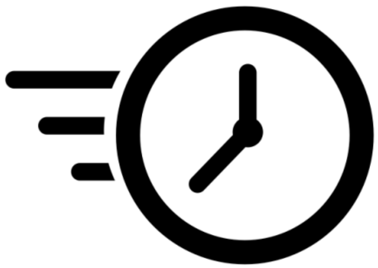
Assignment 2 – Part 1

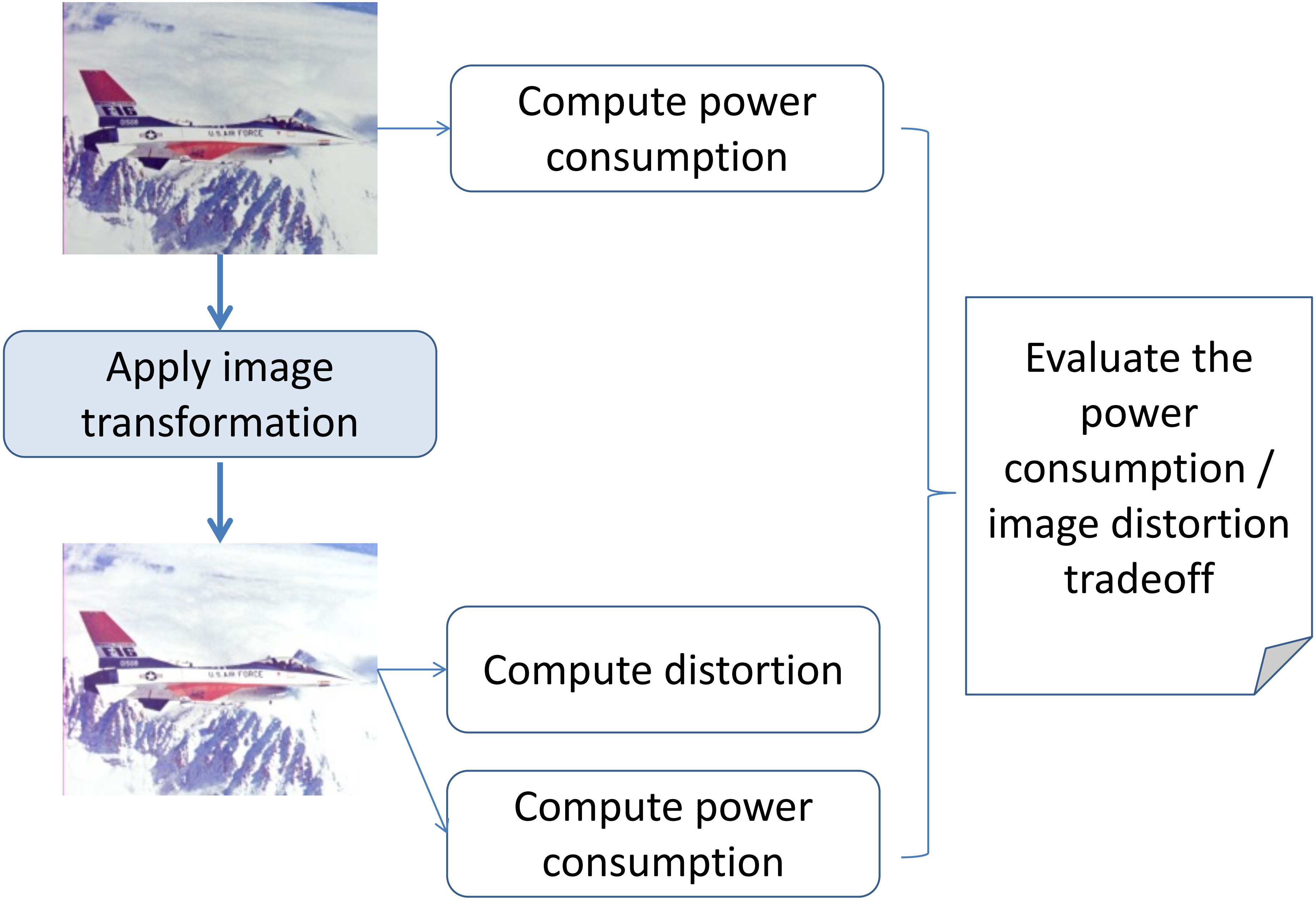
- **TASK:** Define a python function that estimated the distortion w.r.t. the original image
  - $\varepsilon\left(image_i, image_j\right) =$
  $$\sum_{k=1}^{N}\left(\sqrt{(L_{i,k}-L_{j,k})^2 + (a_{i,k} - a_{j,k})^2 + (b_{i,k} - b_{j,k})^2}\right)$$
  - Work in the L*a*b* space and compute the Euclidian distance pixel per pixel
  - Convert by using scikit-image functions *`rgb2lab()`* and *`lab2rgb()`* functions

Assignment 2 – Part 1

- Easier to reason in terms of **percentage** distortion
  - E.g., distortion of new image w.r.t. maximum possible distance between 2 images in Lab space.
  - $dist = \frac{\varepsilon(image_{new}, image_{orig})}{W * H * \sqrt{(100^2 + 255^2 + 255^2)}} \cdot 100 \qquad (\%)$

NOTE: This will be quite small for most transformations!
So, use a small constraint (1%, 2%, 3%)




Compute power consumption
Apply image transformation
Compute distortion
Compute power consumption
Evaluate the power consumption / image distortion tradeoff

Assignment 2 – Part 1

- **TASK:** Experiment with various **image manipulation strategies** to reduce power consumption:
  - Pixel-wise transformations
    - Work on colors
  - Histogram equalization
    - Work on luminance (requires HVS color space)
  - Other types of brightness/contrast modifications

Assignment 2 – Part 1

- Apply each transformation **to all images!**
  - In your report, show (and comment) summary tables. For example:
    - Average, min, max power saving
    - Average, min, max distortion
  - Moreover, show (and comment) **some representative examples** of transformations outputs
    - E.g., the images for which you get most/least saving/distortion.
    - Do **not** include 50 pictures for each transformation in the report!!

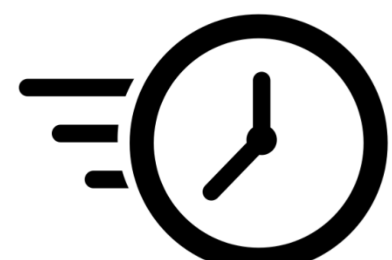
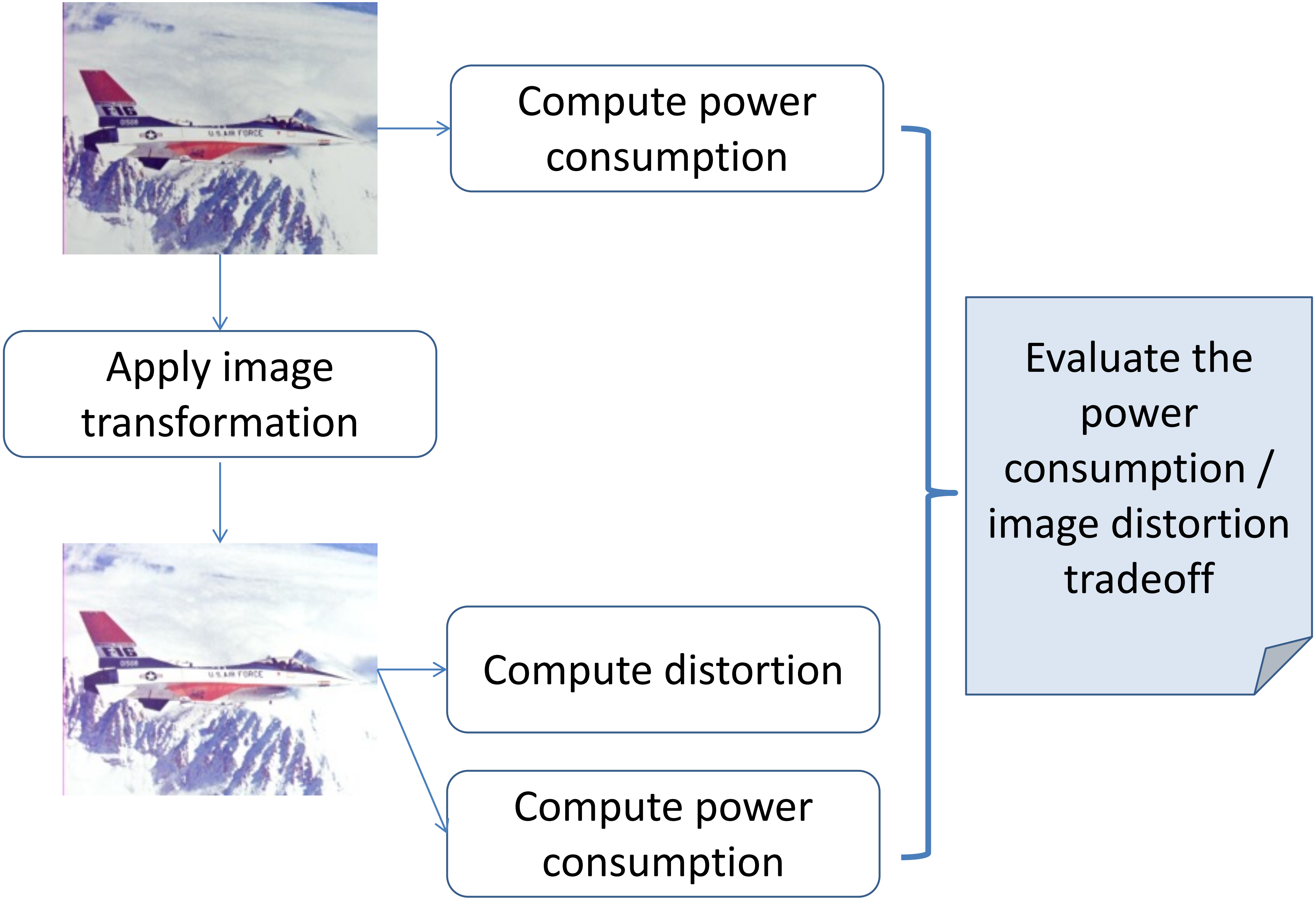


Image Transformations

- **Hungry-blue:**
  - Blue is the more costly component..
  - So, subtract a constant value to the blue component
  - Will introduce a distortion but reduces power
    - Depending on the weight of the blue component

- **Histogram equalization:**
  - Flatten the histogram of the image, e.g., by working on luminance…
  - Requires to work in the HSV color space
    - Hue – saturation – value (i.e., luminance)
  - Introduces distortion. What about power?

- **Other types of brightness/contrast transformations:**
  - E.g. Convert to HSV and scale the value component (V → k*V with k < 1) or do some more complex transformation
  - **Use your creativity!!!**

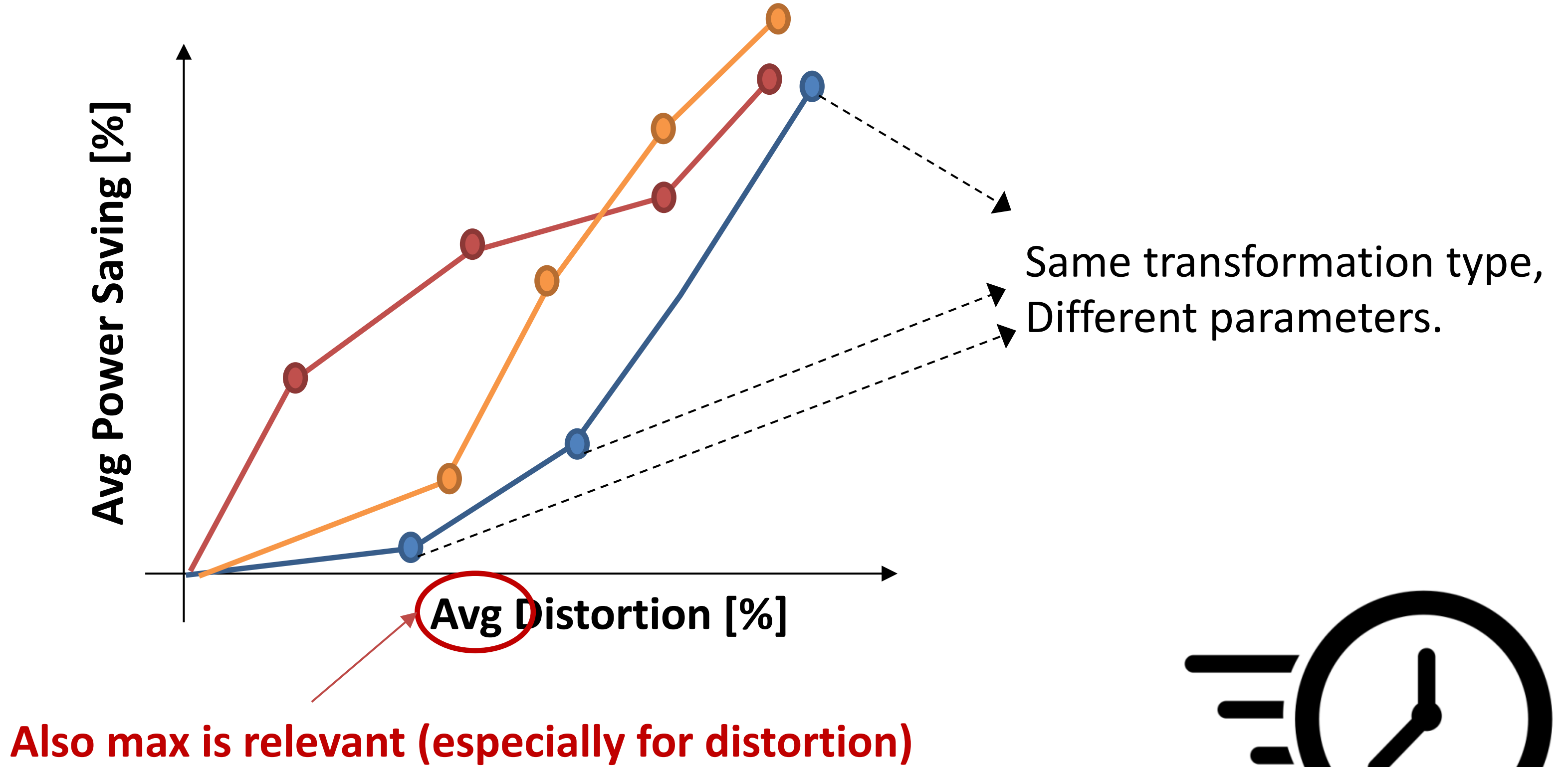

Compute power consumption
Apply image transformation
Compute distortion
Compute power consumption
Evaluate the power consumption / image distortion tradeoff

Assignment 2 – Part 1

- Analyse power/distortion tradeoff
  - Do different images behave differently?
  - What changes in terms of power consumption with different manipulation strategies?
  - How can I save more power with lower distorsions?
  - Etc.

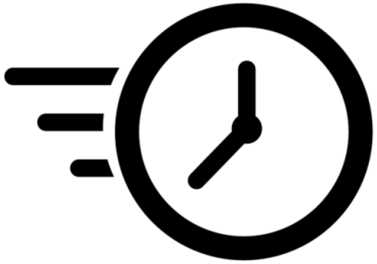
Assignment 2 – Part 1

- Example: Pareto curve



Assignment 2 – Part 1

- Compare the transformations you applied and find the solution that:
  - **Minimizes the average power consumption** (i.e., maximizes avg power saving)
  - Under an **average distortion** constraint (e.g., avg distortion smaller than 0.5%, 1%, 2%, 3%)



Assignment 2 – Part 1

- Example:
  - Blue reduction
    - Power saving 29.11%
    - Distortion 3.99%
  - Histogram equalization
    - Power saving 11.99%
    - Distortion 2.46%

Original image

After blue reduction

After histogram equalization

Day 2
Dynamic Voltage Scaling

Dynamic Voltage Scaling of OLEDs

- Power consumption of OLEDs depends only on pixels…
  - No back light
  - Pixels are emissive, i.e., emits light in response to an electric current

- … and pixels power consumption depends on:
  - Displayed colors
    - Hungry blue / low power green
  - **Input current**

DVS for OLEDs

- Supply voltage is set to maximum to support full luminance of pixel
  - But maximum luminance may not be necessary

- Dynamic Voltage Scaling
  - Scale the supply voltage
    - Reduces maximum current that can flow
    - Saves power
  - Note that reducing current implies changing the RGB color of some pixels!
    - Sacrifice image quality for power saving

DVS for OLEDs

- Effects of DVS
  - Reducing current implies changing the RGB color of pixels!
    - Emitted color strictly depends on input current
    - Reduced voltage → reduced current through some pixels

ORIGINAL IMAGE

SIMULATED VOLTAGE SCALING

DVS for OLEDs

- Effects of DVS
  - Sacrifice image quality for power saving
    - Reduced color luminance
    - Color distortion in displayed images
  - Saved power

ORIGINAL IMAGE

SIMULATED VOLTAGE SCALING APPROX. 20% POWER SAVING

DVS for OLEDs

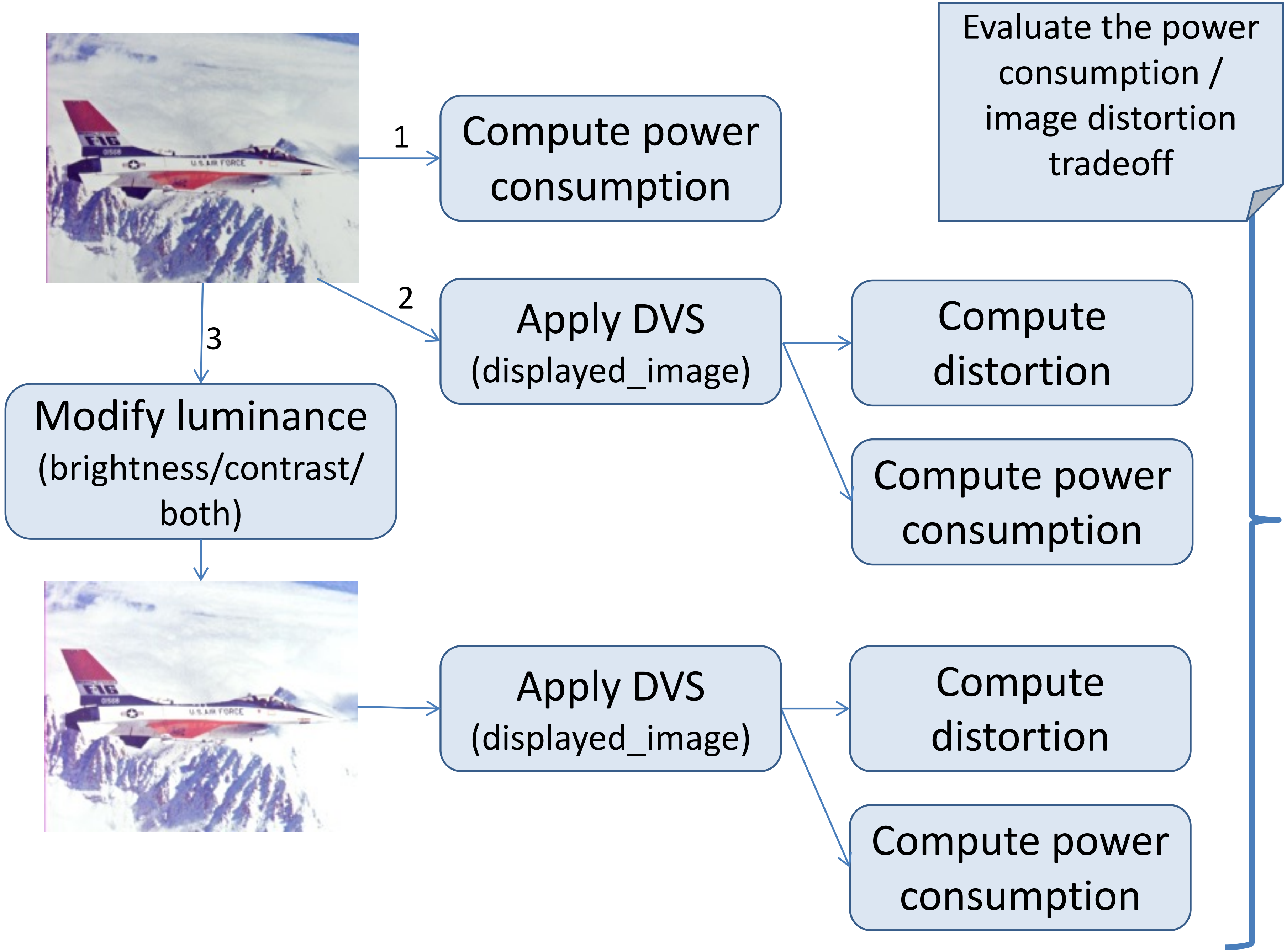
- Can compensate the image distortion by applying an image compensation
  - E.g., working on image luminance

Original image

Effect of voltage scaling

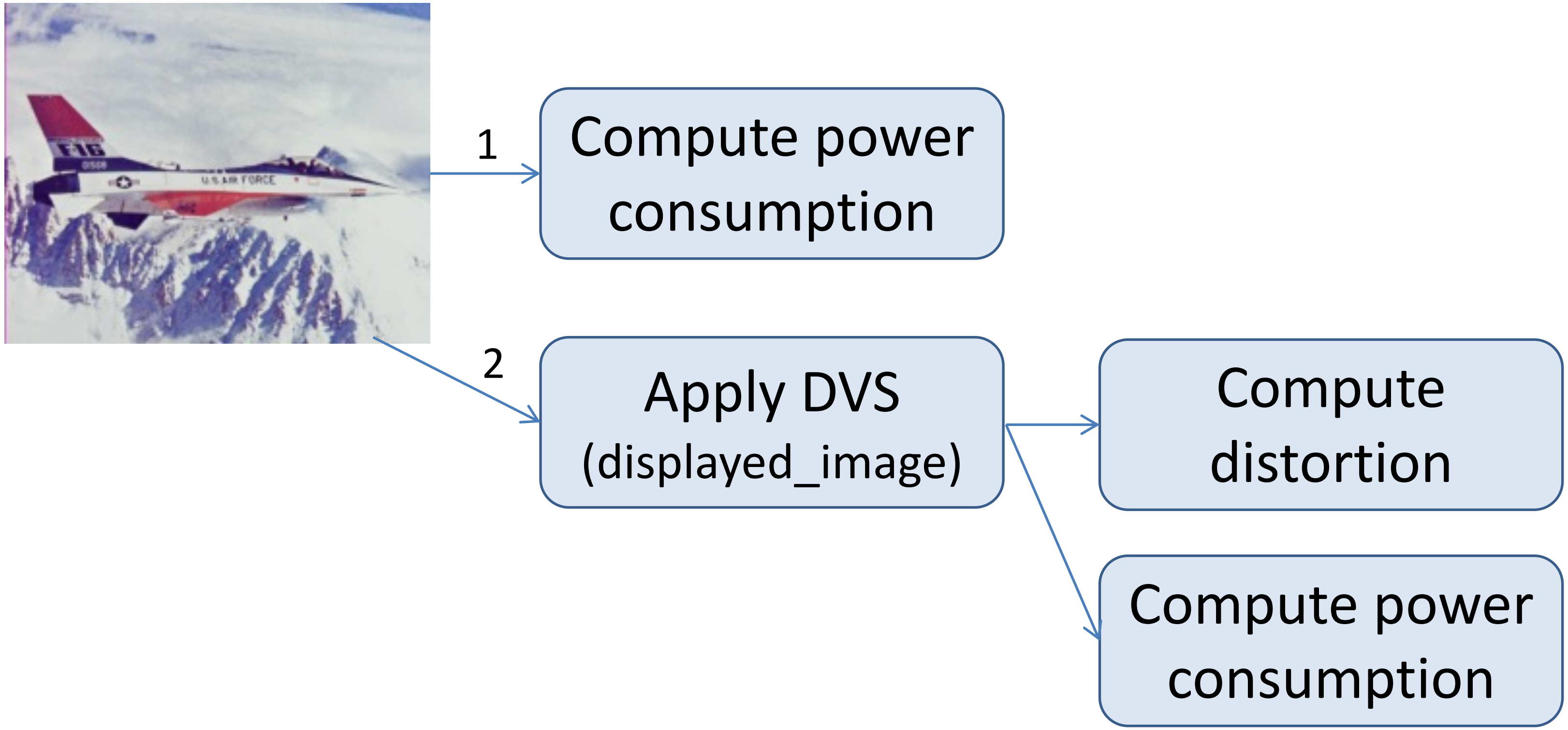
Effect of image compensation + voltage scaling

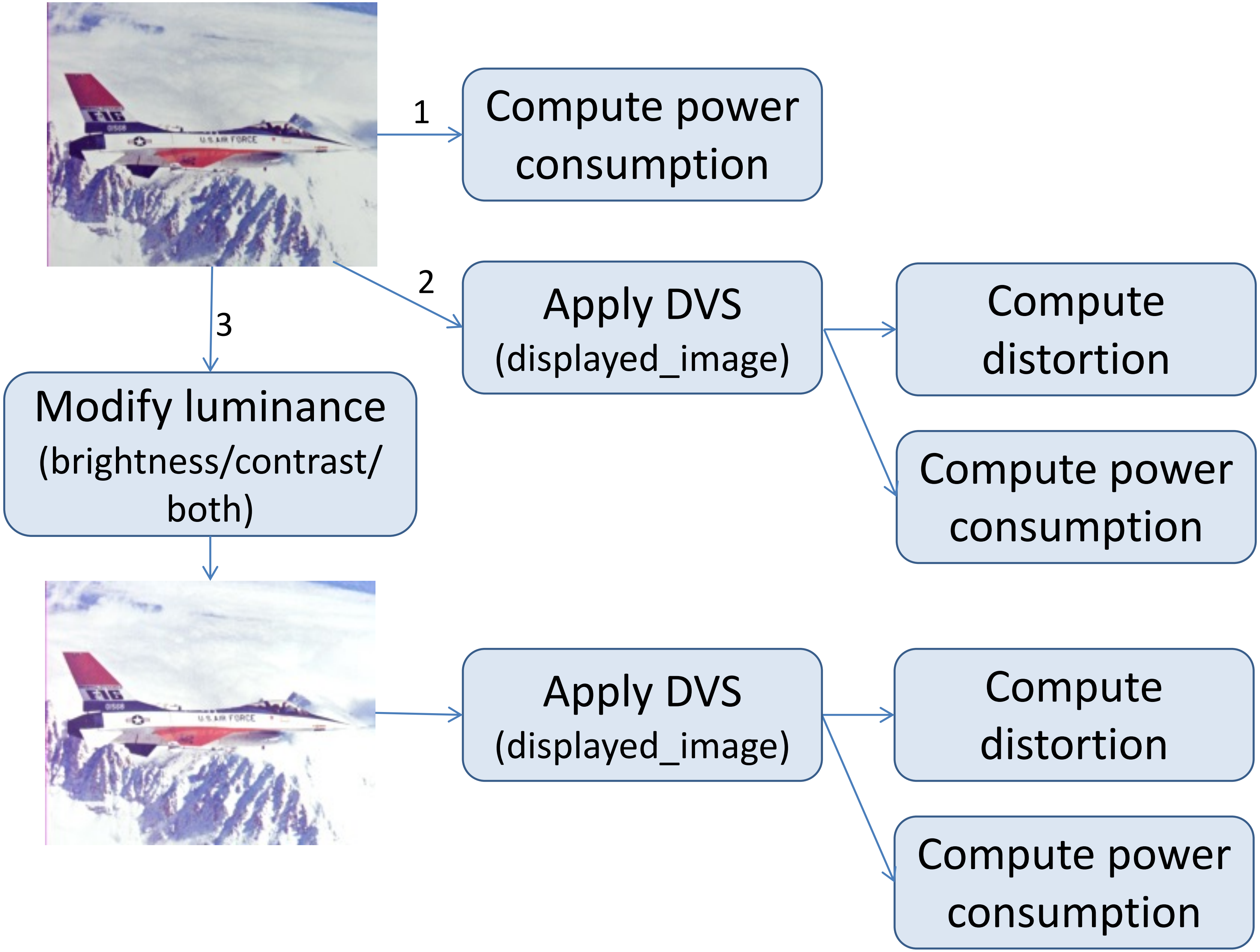
Assignment 2 – Part 2

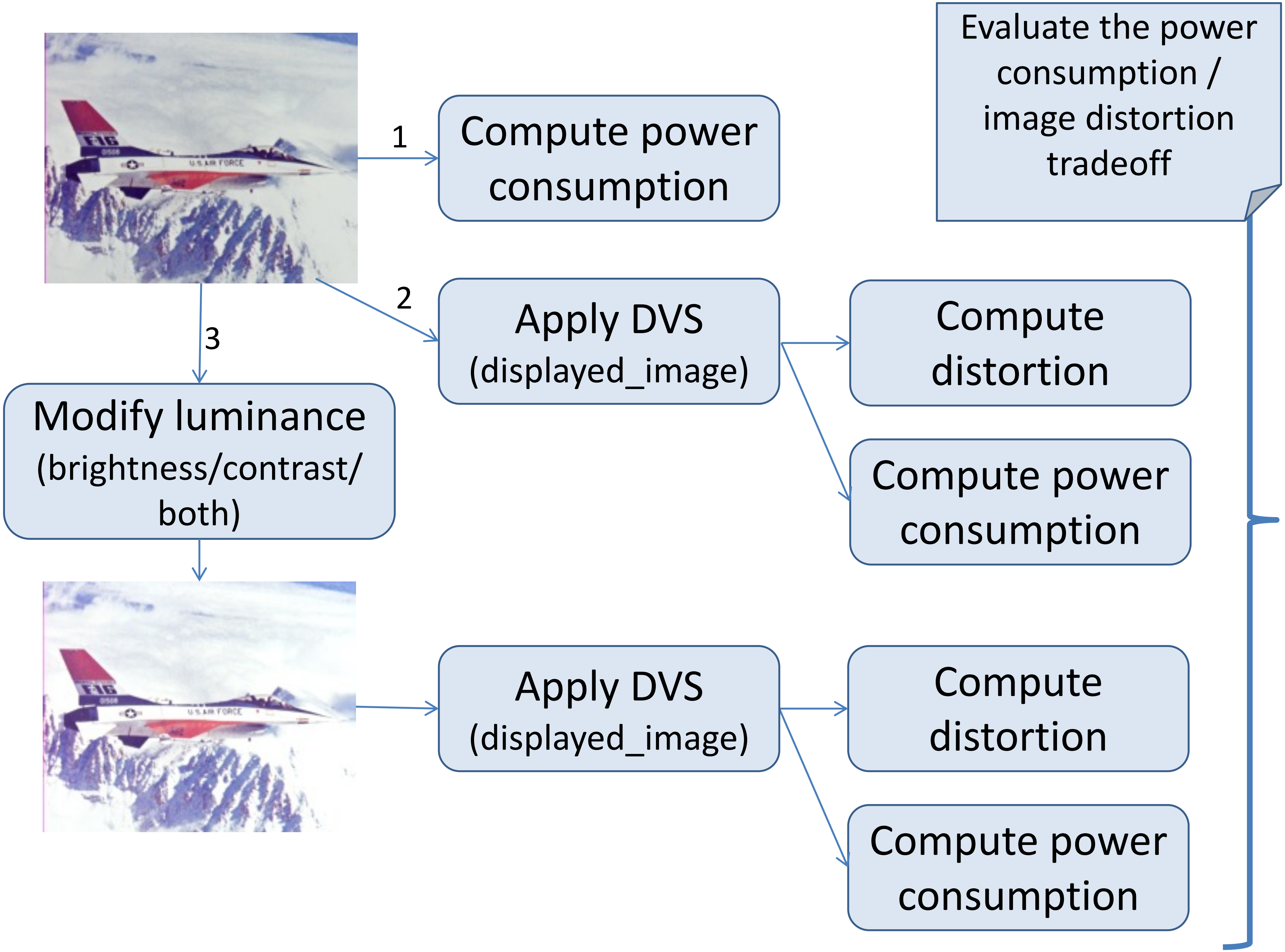

Evaluate the power consumption / image distortion tradeoff
1
Compute power consumption
2
Apply DVS (displayed_image)
Compute distortion
Compute power consumption
3
Modify luminance (brightness/contrast/ both)
Apply DVS (displayed_image)
Compute distortion
Compute power consumption

1

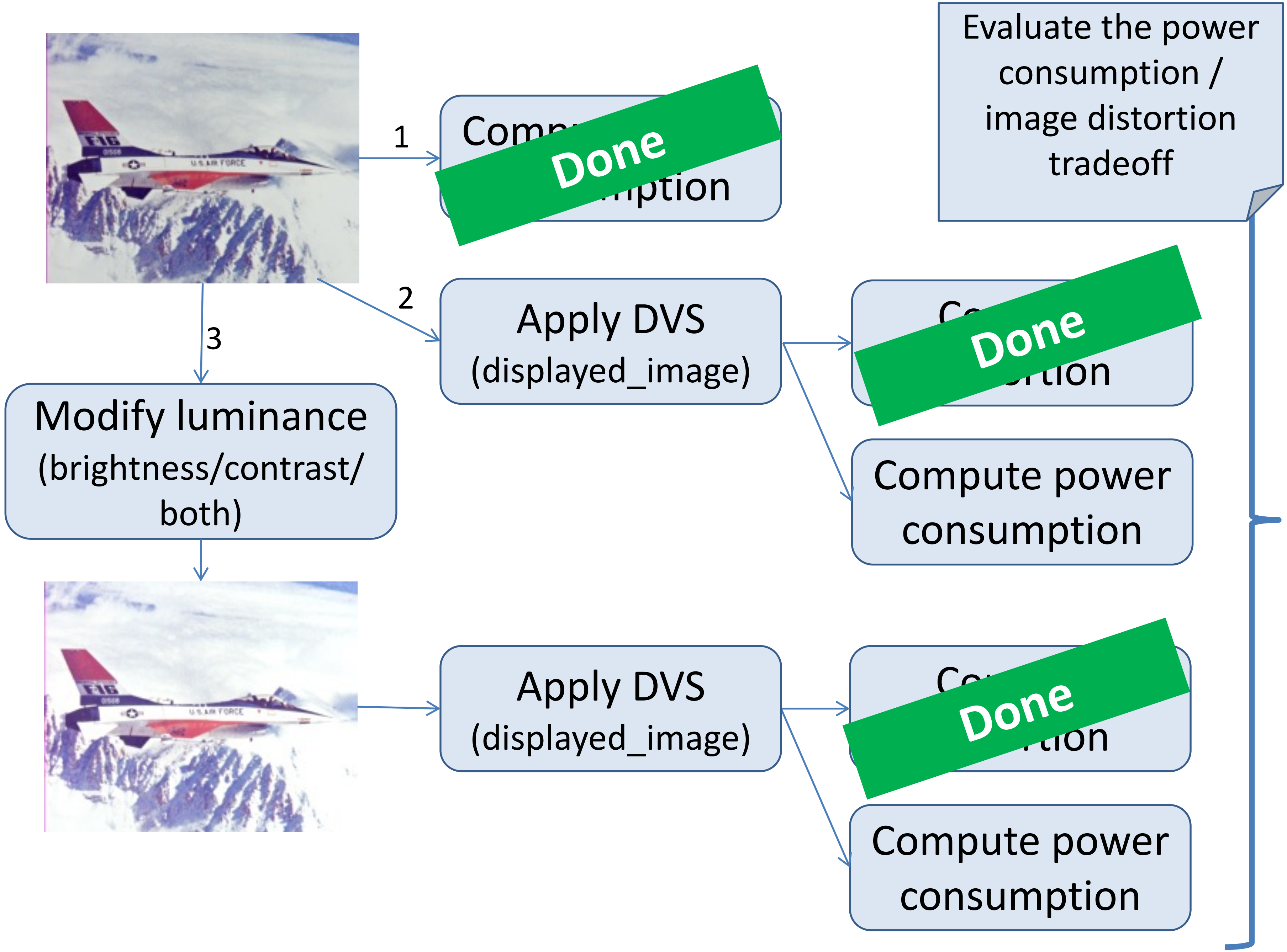
Compute power consumption


1
Compute power consumption
2
Apply DVS
(displayed_image)
Compute distortion
Compute power consumption


1
Compute power consumption
2
Apply DVS (displayed_image)
Compute distortion
Compute power consumption
3
Modify luminance (brightness/contrast/ both)
Apply DVS (displayed_image)
Compute distortion
Compute power consumption

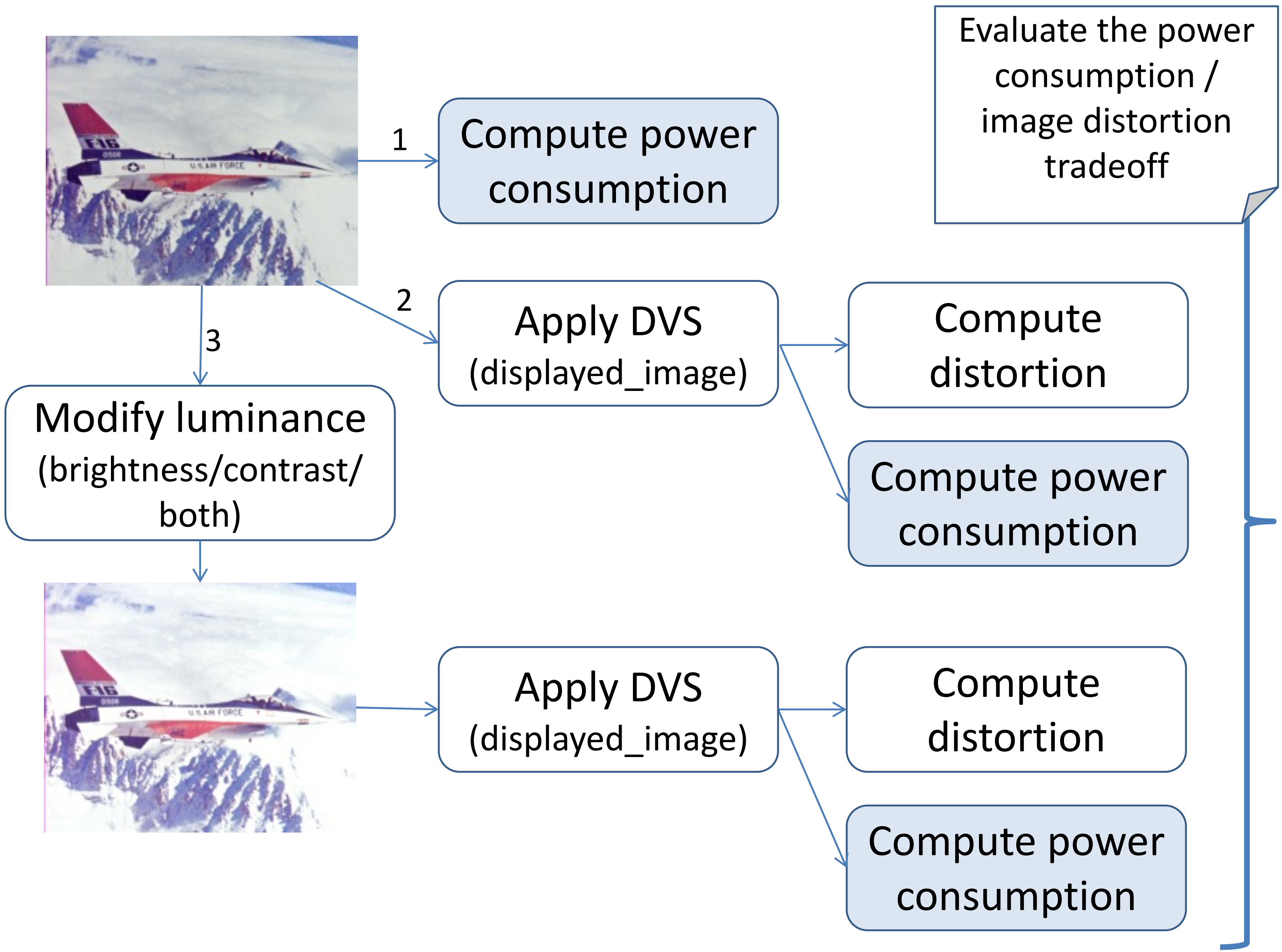


Evaluate the power consumption / image distortion tradeoff
1
Compute power consumption
2
Apply DVS (displayed_image)
Compute distortion
Compute power consumption
3
Modify luminance (brightness/contrast/ both)
Apply DVS (displayed_image)
Compute distortion
Compute power consumption



Evaluate the power consumption / image distortion tradeoff
1
Done
2
Apply DVS
(displayed_image)
Done
Compute power consumption
3
Modify luminance
(brightness/contrast/ both)
Apply DVS
(displayed_image)
Done
Compute power consumption

Assignment 2 - Part 2: How To



Evaluate the power consumption / image distortion tradeoff
1
Compute power consumption
2
Apply DVS (displayed_image)
Compute distortion
Compute power consumption
3
Modify luminance (brightness/contrast/ both)
Apply DVS (displayed_image)
Compute distortion
Compute power consumption

1. Cell current calculation and evaluation of power consumption

- Given the RGB color of each pixel, determine current flowing through the cell

$$I_{cell} = \frac{p_1 \, V_{dd} \, D_{RGB}}{255} + \frac{p_2 \, D_{RGB}}{255} + p_3 \qquad [mA]$$

D_{RGB} is the RGB color value of current pixel

- Determine power consumption
  - **Different (less accurate) model w.r.t. the one used in Part 1**, but expressing dependency from DVS
  - $P_{panel} = V_{dd} \sum_{i=1}^{W} \sum_{j=1}^{H} \sum_{\{R,G,B\}} I_{cell(i,j)} \qquad [mW]$

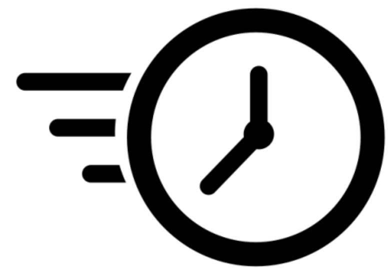
Assignment 2 – Part 2

- **TASK:** Implement the new current and power models

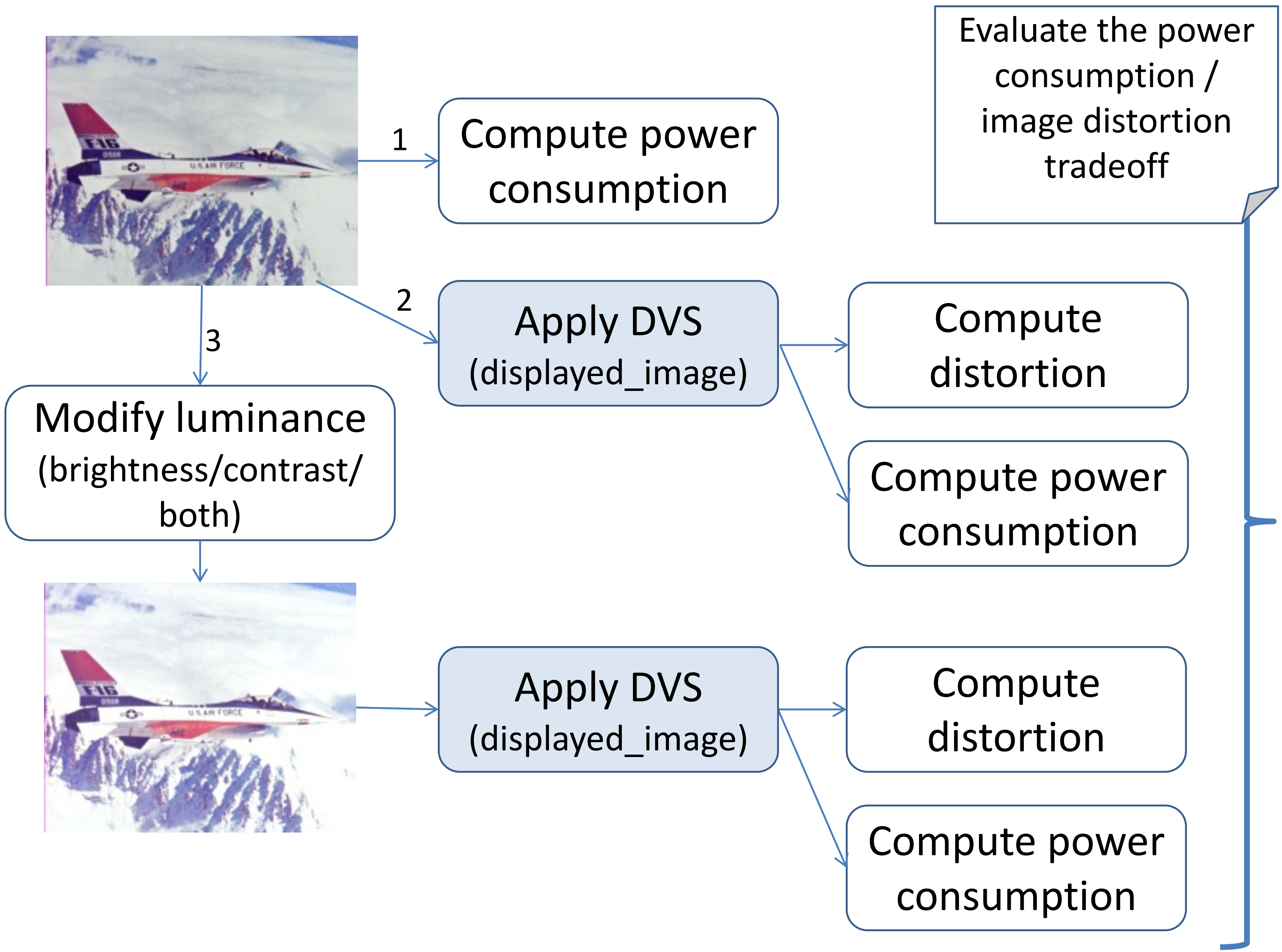
$$- I_{cell} = \left(\frac{p_1 V_{dd} D_{RGB}}{255}\right) + \left(\frac{p_2 D_{RGB}}{255}\right) + p_3 \qquad [mA]$$

  - $p_1 = +4.251e\text{-}05$
  - $p_2 = -3.029e\text{-}4$
  - $p_3 = +3.024e\text{-}5$
  - Default $V_{dd} = 15V$

Better to have two separate functions (see later)



$$- P_{panel} = V_{dd} \sum_{i=1}^{W} \sum_{j=1}^{H} \sum_{\{R,G,B\}} I_{cell(i,j)} \qquad [mW]$$



Evaluate the power consumption / image distortion tradeoff
1
Compute power consumption
2
Apply DVS (displayed_image)
Compute distortion
Compute power consumption
3
Modify luminance (brightness/contrast/ both)
Apply DVS (displayed_image)
Compute distortion
Compute power consumption

2. Application of voltage scaling

- Voltage supply determines the maximum current that can flow in the OLED
  - Current value → pixel color

- Effect simulated by the function

$$\mathtt{displayed\_image}(I_{cell},\ V_{dd})$$

  - Given an image as the matrix of currents corresponding to pixels
  - Applies voltage scaling with the specified V_{dd}
  - **This function is provided. You don't have to implement it.**

2. Application of voltage scaling

- Effect simulated by the `displayed_image()` function
  - Computes the maximum current that can flow with the new V_{dd}
  - Determines the corresponding maximum RGB value RGB_{max}
  - Any RGB value higher than RGB_{max} is saturated to RGB_{max}

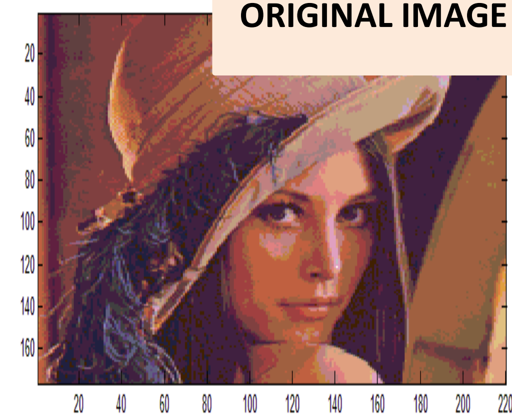
2. Application of voltage scaling

- Given
  - $I_{cell} = \left(\frac{p_1 \, V_{dd} \, D_{RGB}}{255}\right) + \left(\frac{p_2 \, D_{RGB}}{255}\right) + p_3$
- The maximum current given V_{dd} is:
  - $I_{max} = \left(\frac{p_1 \, V_{dd} \, [255 \; 255 \; 255]}{255}\right) + \left(\frac{p_2 [255 \; 255 \; 255]}{255}\right) + p_3$
- …and the maximum RGB that can be displayed without distortion is:
  - $RGB_{max} = \frac{(I_{max} \; - p_3) \, 255}{p_1 V_{dd} + p_2}$
    - Whenever I_{cell} > I_{max} the pixel is assigned RGB value RGB_{max}
    - Saturate to the maximum RGB value that can be generated given V_{dd}

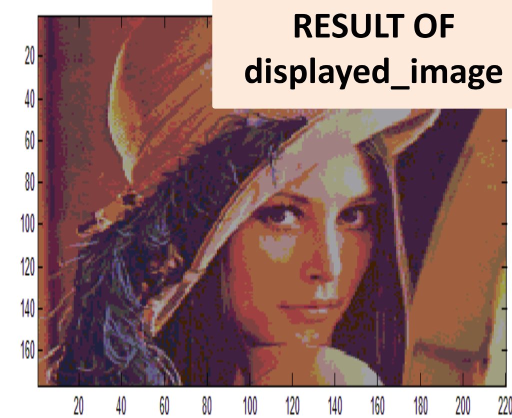
2. Application of voltage scaling

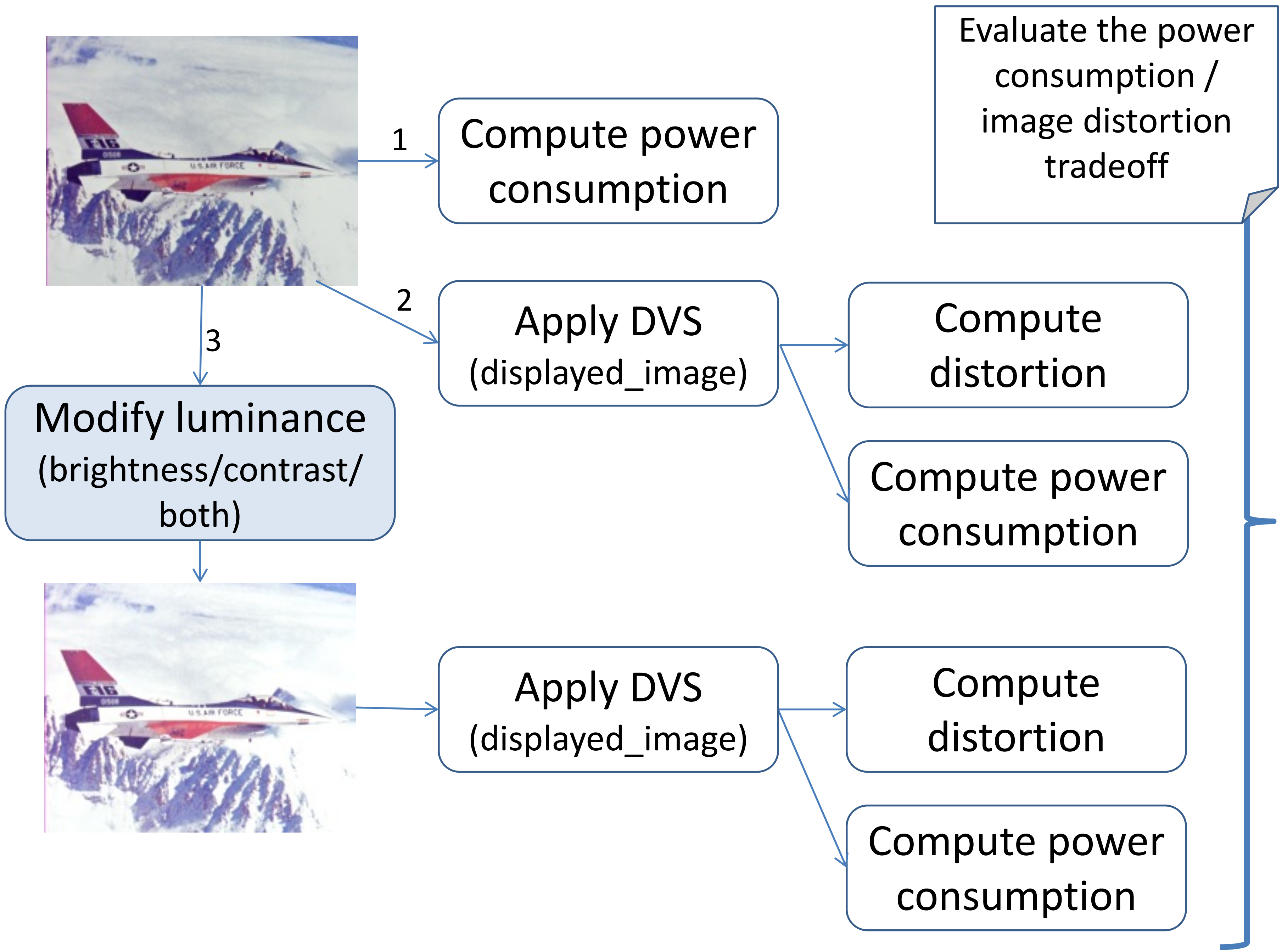
- Note: the image RGB values do not actually change!
  - What changes is the *effect* on the display
  - `displayed_image()` function simulates this effect

ORIGINAL IMAGE



RESULT OF displayed_image




Evaluate the power consumption / image distortion tradeoff
1
Compute power consumption
2
Apply DVS (displayed_image)
Compute distortion
Compute power consumption
3
Modify luminance (brightness/contrast/ both)
Apply DVS (displayed_image)
Compute distortion
Compute power consumption

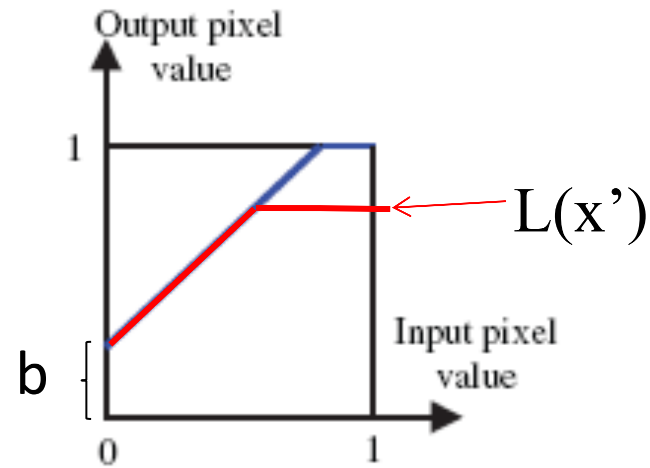
3. Apply image compensation

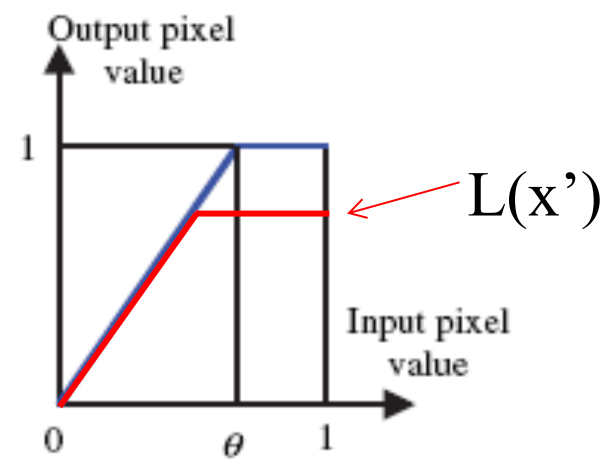
- Want to improve quality of resulting image
  - Apply some techniques **before** DVS!
  - Enhance brightness/contrast of image

- The goal is to increase the perceived image quality!

3. Apply image compensation

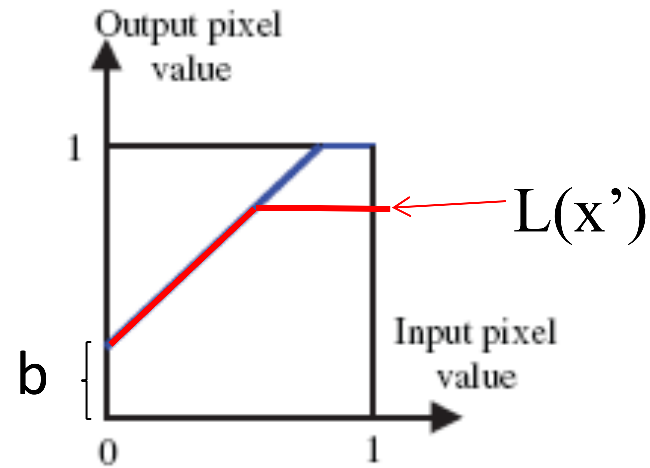
- Brightness scaling
  - Increase the pixel values by a certain constant b
    - E.g., b equal to luminance loss
    - V’ = v + b
- Contrast enhancement
  - Amplification of contrast
    - V’ = V * b
  - Multiply pixel values by a certain constant factor
    - Depends on variation of V_{dd}

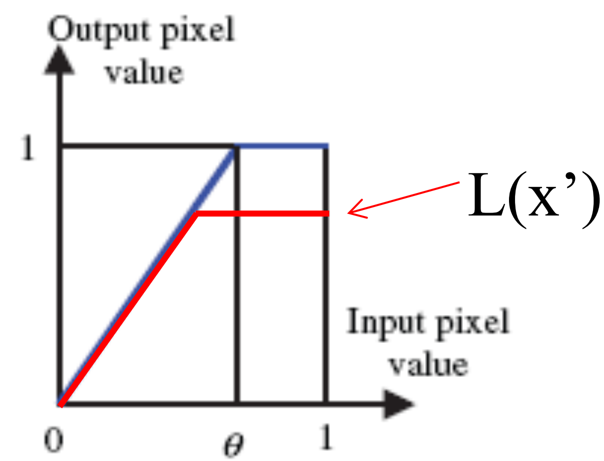




3. Apply image compensation

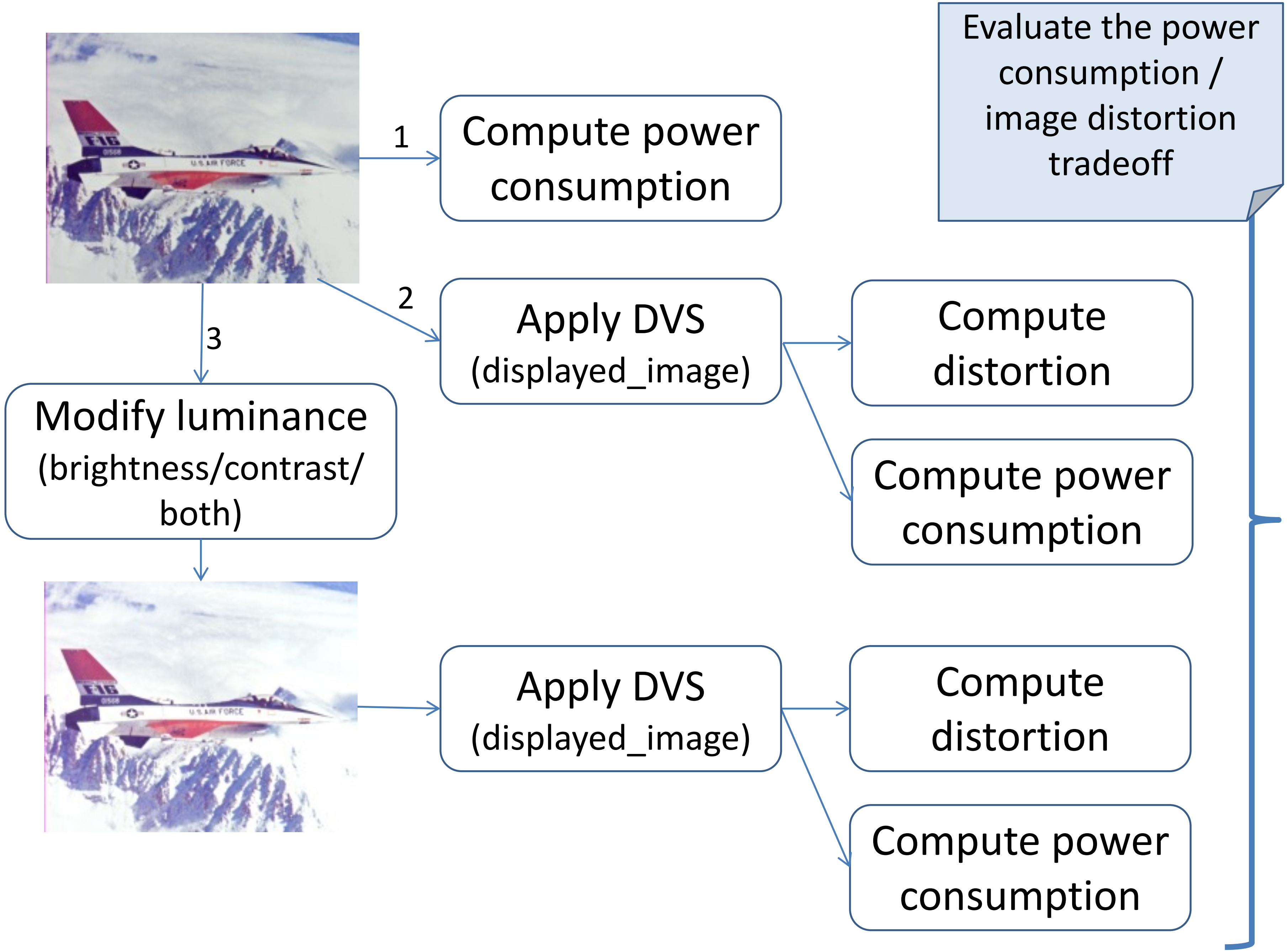
- **Implemented in the HSV space**
- You can determine the factor b as dependent from the original V_{dd} and new (scaled) V_{dd}
  - Brightness compensation
    - V' = V + b
    - b($V_{dd}\ original, V_{dd}\ new$)
  - Contrast enhancement
    - V' = V * b
    - b($V_{dd}\ original, V_{dd}\ new$)
  - **Application of both**





Assignment 2 – Part 2

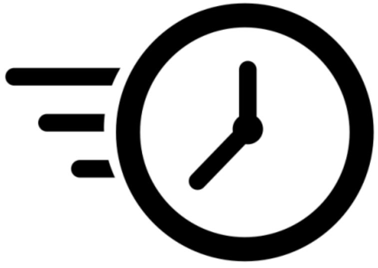
- **TASK:** Experiment with various **image compensation strategies:**
  - Brightness scaling
  - Contrast enhancement
  - Combined BS + CE
  - Others… (**again, use your creativity!**).



Evaluate the power consumption / image distortion tradeoff
1
Compute power consumption
2
Apply DVS (displayed_image)
Compute distortion
Compute power consumption
3
Modify luminance (brightness/contrast/ both)
Apply DVS (displayed_image)
Compute distortion
Compute power consumption

Assignment 2 – Part 2

- Compare the different DVS + image compensation strategies
  - With respect to part 1, you have a new «free variable» → **The DVS voltage**



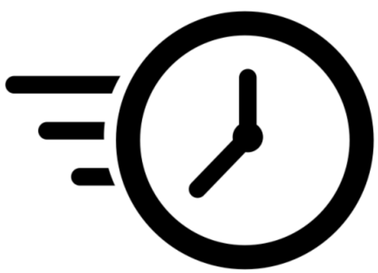
Assignment 2 – Part 2

- The goal becomes *«find optimal supply voltage and compensated image»* to:
  - Minimize power consumption
  - Maximize **perceived visual quality**

  - **Important:** image compensations will typically *increase* the distortion.
    - Remember: **visual quality** is different from **distortion**!
    - But the former is only qualitative…

Assignment 2 – Part 2

- So, what you can do, is:
  - Impose a maximum distortion constraint (e.g., 1%, 2%, 3%), as in Part 1
    - Use the LAB distance for distortion, as in Part 1
  - Among the (V_{DD}, compensation) pairs that meet this constraint, select the one that *in your opinion* yields the *best-quality* images
    - **Of course, this is subjective!!!**



Assignment 2 – Part 2

- Apply the overall flow to all images!
  - Automatically, with a script