Lab 2
Energy efficient image processing

Objective and organization

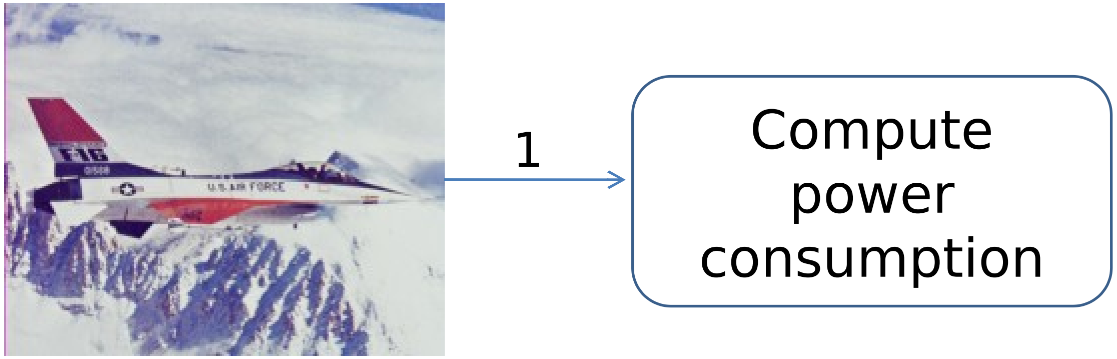
- Demonstrates how manipulation of an image can be used to tradeoff image quality to save power in emissive displays
  - 1 report – 2 days
  - Matlab + C (Arduino)

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

Assignment 2

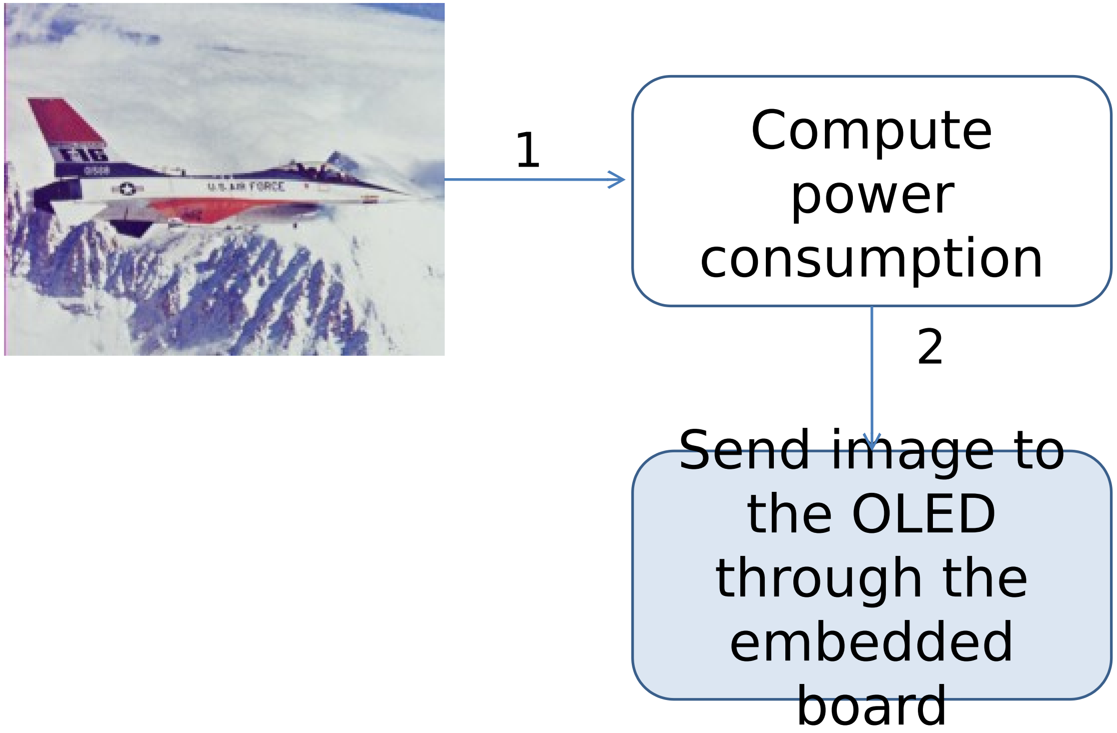
Objective

- Test the pixel transformations studied in the first session on a real compact OLED display.
  - Check the *real* visual quality of the output on the screen.



1

Compute power consumption

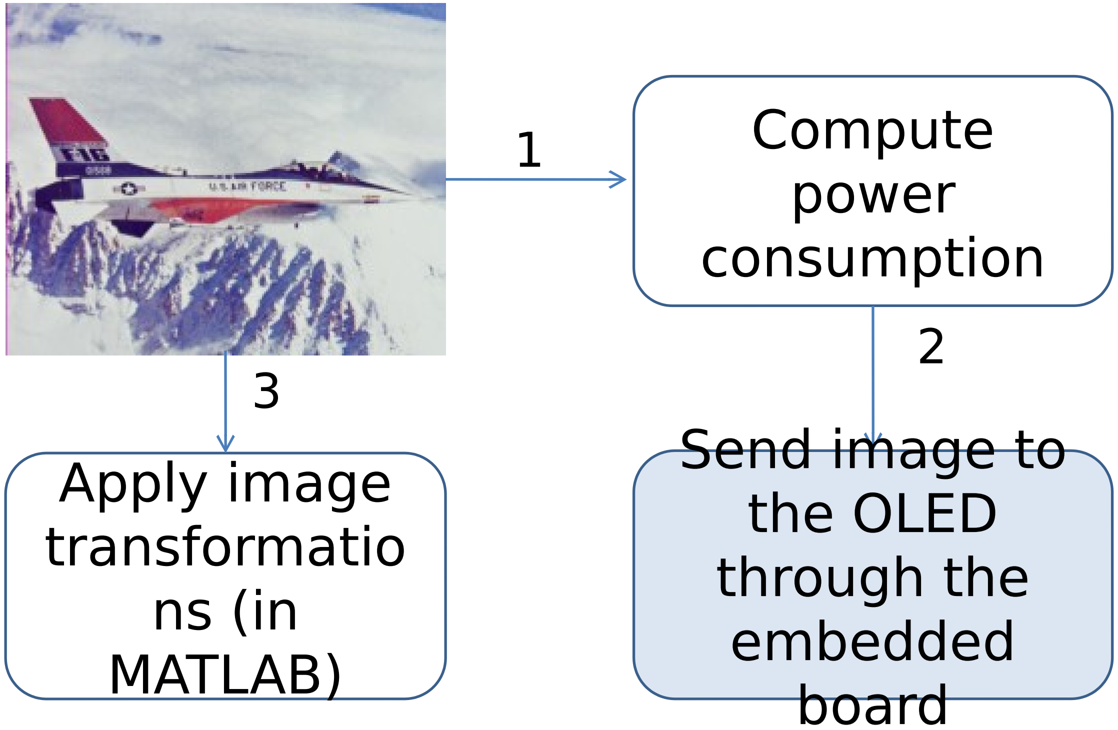


1

Compute power consumption

2

Send image to the OLED through the embedded board



1

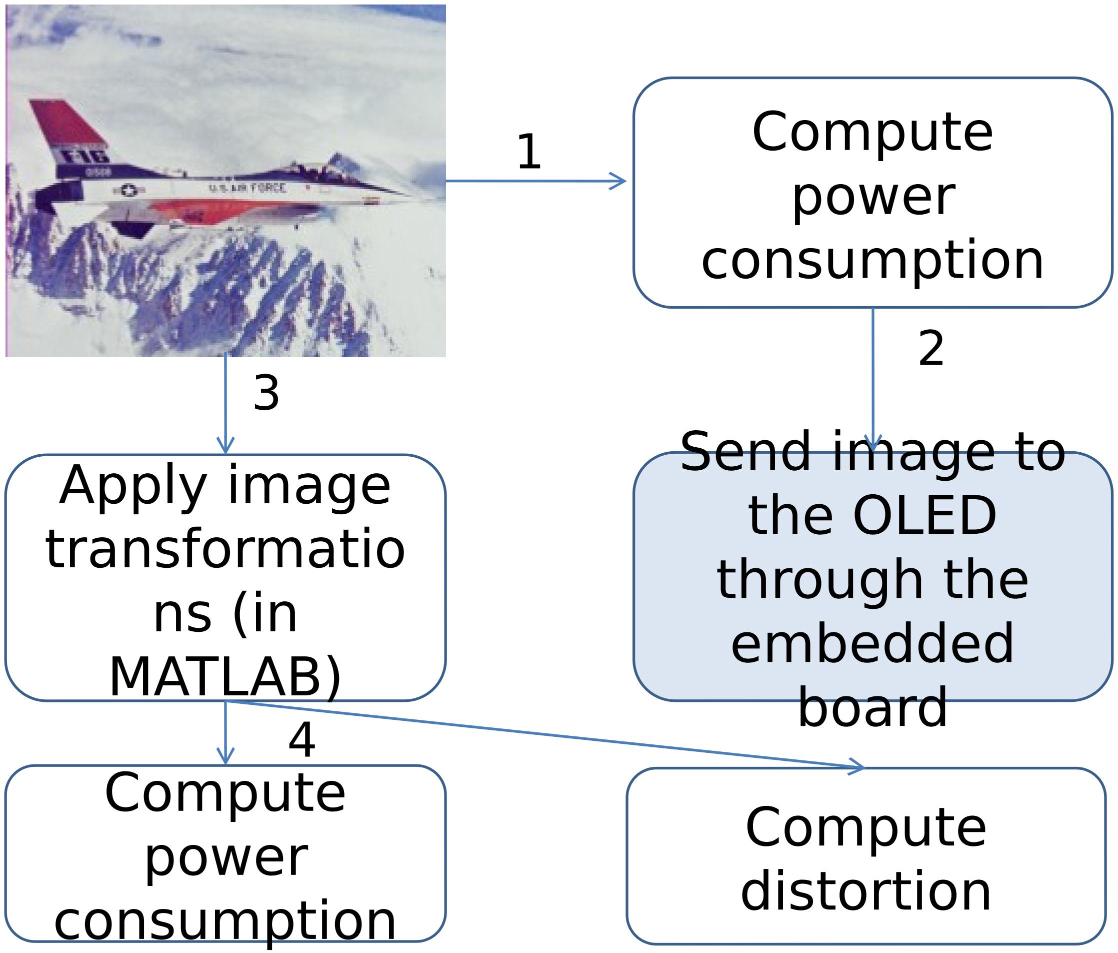
Compute power consumption

2

Send image to the OLED through the embedded board

3

Apply image transformations (in MATLAB)



1

Compute power consumption

2

Send image to the OLED through the embedded board

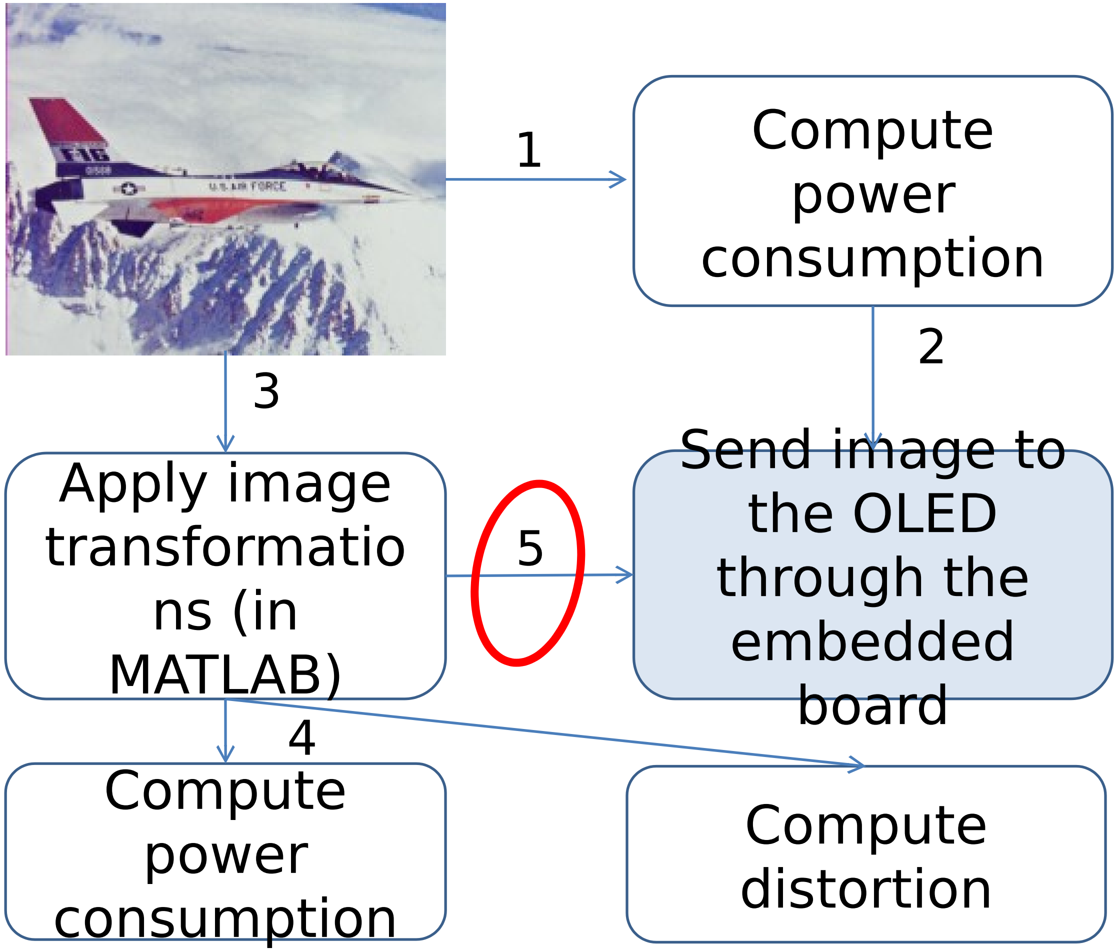
3

Apply image transformations (in MATLAB)

4

Compute power consumption

Compute distortion



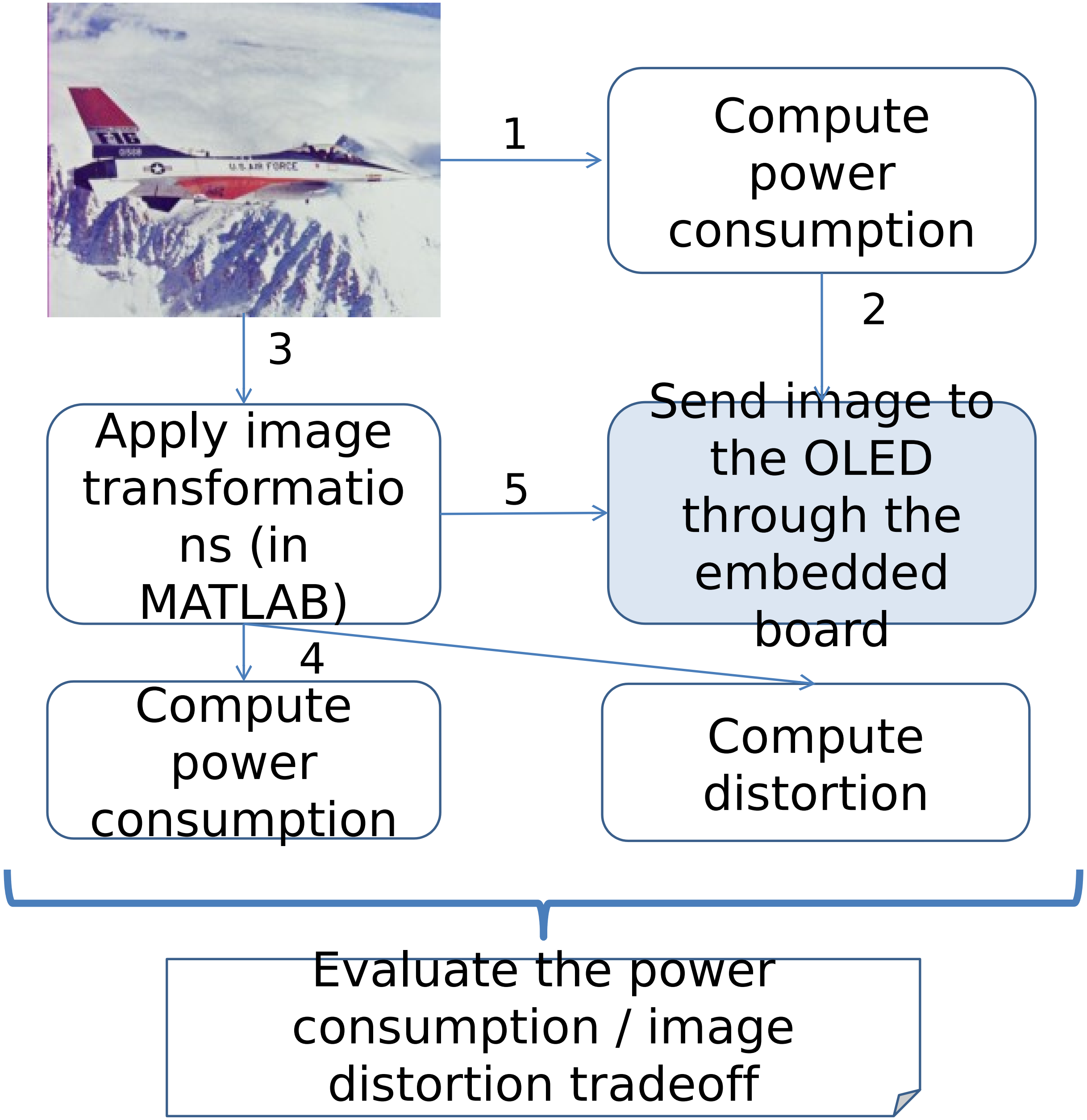
Compute power consumption

Send image to the OLED through the embedded board

Apply image transformations (in MATLAB)

Compute power consumption

Compute distortion



Compute power consumption

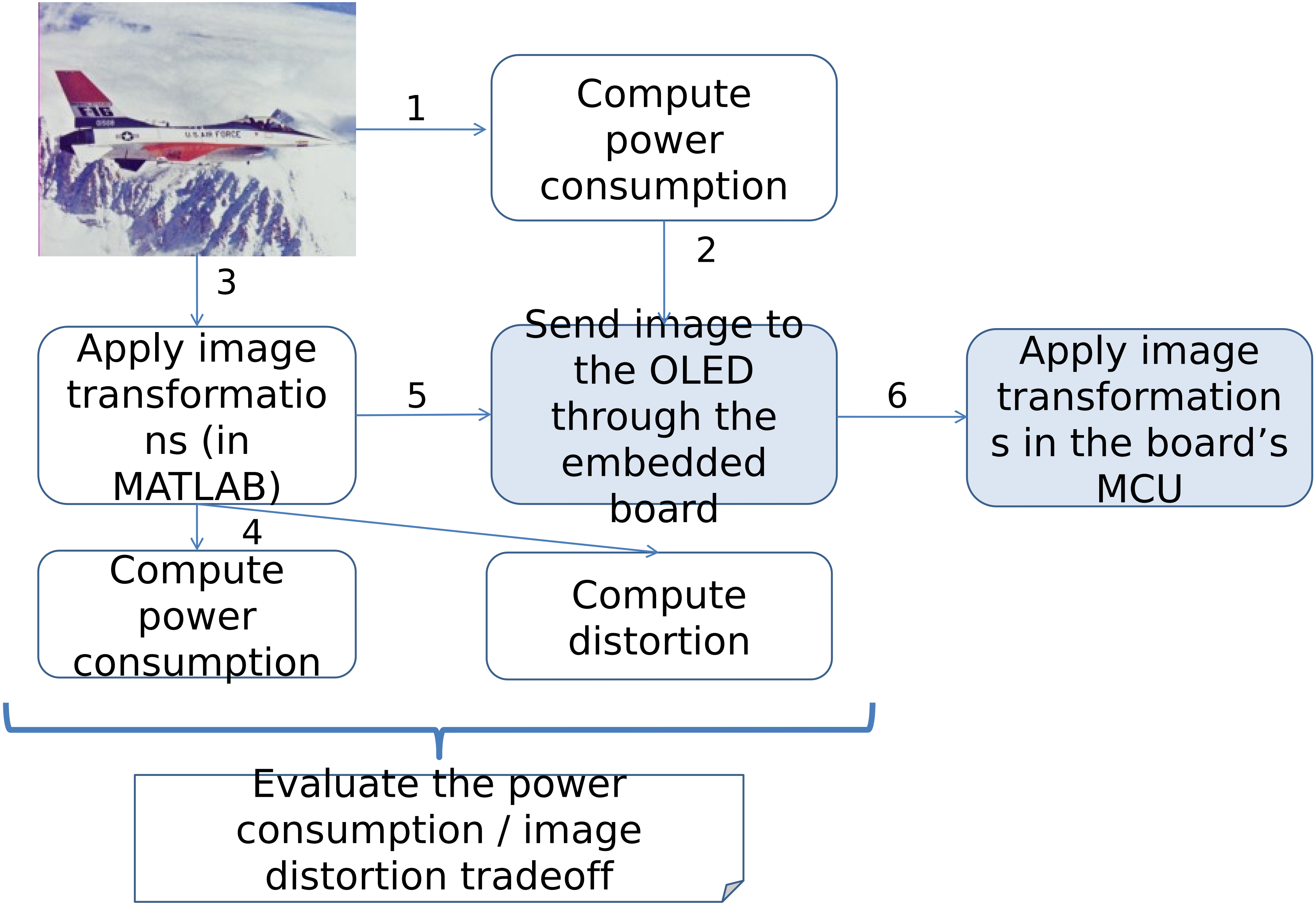
Apply image transformations (in MATLAB)

Send image to the OLED through the embedded board

Compute power consumption

Compute distortion

Evaluate the power consumption / image distortion tradeoff



Compute power consumption

Apply image transformations (in MATLAB)

Send image to the OLED through the embedded board

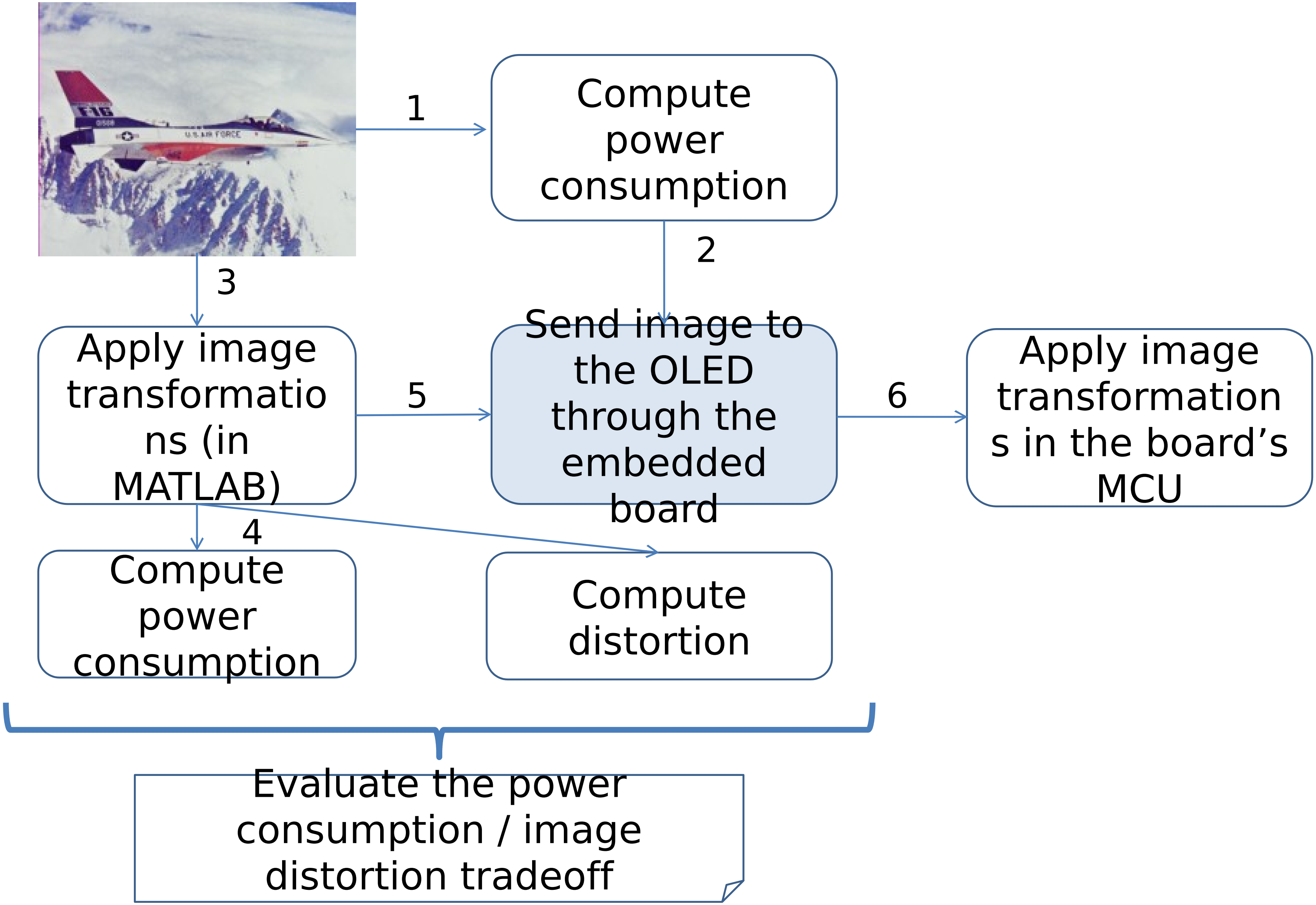
Apply image transformations in the board’s MCU

Compute power consumption

Compute distortion

Evaluate the power consumption / image distortion tradeoff

Assignment 2: how to



1

Compute power consumption

2

3

Apply image transformations (in MATLAB)

5

Send image to the OLED through the embedded board

6

Apply image transformations in the board’s MCU

4

Compute power consumption

Compute distortion

Evaluate the power consumption / image distortion tradeoff

2. Send image to the OLED

- Provided setup:
  - **Arduino Uno** board (https://store.arduino.cc/arduino-uno-rev3)
  - **128x128 New Haven Display NHD-1.5-128128ASC3** (https://eu.mouser.com/datasheet/2/291/NHD-1.5-128128ASC3-784468.pdf)
  - **USB cable (??)**
  - **GPIO connection cables**

2. Send image to the OLED

- Communication mechanisms:
  - The OLED receives configurations and pixel data through an SPI interface (implemented through the Arduino GPIOs)
  - For sending images from the PC to the Arduino, we use RS-232 over USB
  - **IMPORTANT:** The Arduino Uno’s MCU does not have enough memory to store the entire 128x128 image (128x128x3 bytes required). So, pixels must be immediately forwarded from the RS-232 bus to the SPI bus.

2. Send image to the OLED

- Arduino/OLED connection:

PIN Function	**SCK**	**MOSI**	**NRST**	**Chip Select**	**Data/ Control**	**VDD**	**GND**
Arduino Name	**D13**	**D11**	**D6**	**D5**	**D4**	**3.3V**	**GND**
OLED Name	**SCK**	**MOSI**	**/RES**	**OLEDCS**	**D/C**	**VDD**	**GND**

- Writing firmware for Arduino Uno:
  - Done using the Arduino IDE (https://www.arduino.cc/en/main/software)
  - Simplified C/C++ dialect

2. Send image to the OLED

- MATLAB code to implement:
  1. **Read** an image (imread)
  2. **Resize it** to fit the display (imresizse)
  3. **Convert it** to the appropriate color representation
     - The provided OLED uses **18-bit RGB** (6-bit per color channel)
     - It also expects data in **B-G-R order** (vs. R-G-B)
  4. **Send pixels** to the Arduino through RS-232
     - Using MATLAB’s serial I/O functionalities
     - serial(), fopen(), frwrite(), flushoutput(), fclose()

2. Send image to the OLED

- MATLAB code to implement (cont’d):
  - To make the communication more reliable, add a mechanism to signal the beginning of a new image to the Arduino
    - Useful in case of interrupted communication
    - Exploit the 6-bit per channel color representation (the 2 MSBs of each byte sent on the RS-232 are unused, and typically will be 0s).
    - Send a special byte with those 2 bits at 1 at the beginning to signal the start of a new image.

2. Send image to the OLED

- MATLAB code to implement (cont’d):
  - Set the RS-232 Baud Rate in MATLAB (and Arduino) to a value <= 115200 to avoid data losses.
  - Opening the RS-232 file on the PC resets the Arduino.
    - Add a «wait» of few seconds before starting to send data
    - Also wait some time between the first «start» byte and the rest of the data, to allow the OLED to reconfigure itself.

2. Send image to the OLED

- Arduino code provided:
  - Basic OLED functions (initialize, setup, send pixel, etc.)
- Arduino code to implement:
  - **Initialize OLED and communication interfaces** at boot time
  - **Forward pixel values** received on the RS-232 to the OLED
    - Reset OLED address when a «start» byte is received

2. Send image to the OLED

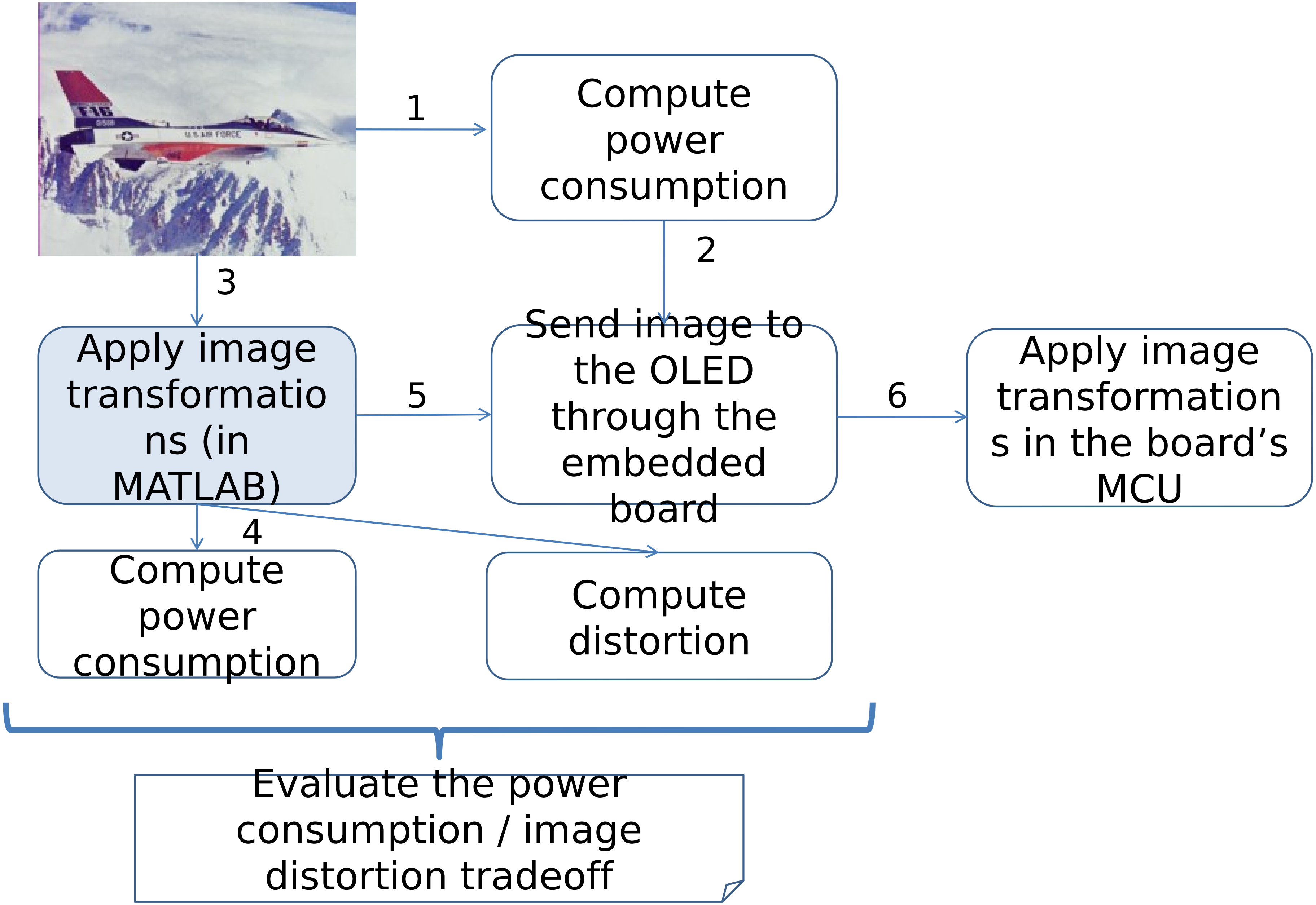
- Arduino code to implement (cont’d):
  - **Initialize OLED and communication interfaces** at boot time
    - **OLED_setup():** setup pins used by the Arduino/OLED interface, configure Serial (RS232) and SPI interfaces
    - **OLED_init():** initialize OLED display with default settings

2. Send image to the OLED

- Arduino code to implement (cont'd):
  - **Forward pixel values** received on the RS-232 to the OLED
    - **void serialEvent():** default ISR to handle RS232 events (received bytes)
    - **Serial.read():** read one char from the RS232 channel (cast output to uint8_t).
    - **Serial.available():** check if there are bytes available for reading in the RS232 channel

2. Send image to the OLED

- Arduino code to implement (cont’d):
  - **Forward pixel values** received on the RS-232 to the OLED
    - **OLED_Data_128128RGB():** write the next color component to the screen
      - Pixel addresses are updated automatically.
      - Remember the BGR color order
    - **OLED_SetRowAddress_128128RGB()** and **OLED_SetColumnAddress_128128RGB():** set the starting address for a new image
    - **OLED_WriteMemoryStart_128128RGB():** enable writing to the OLED’s internal memory

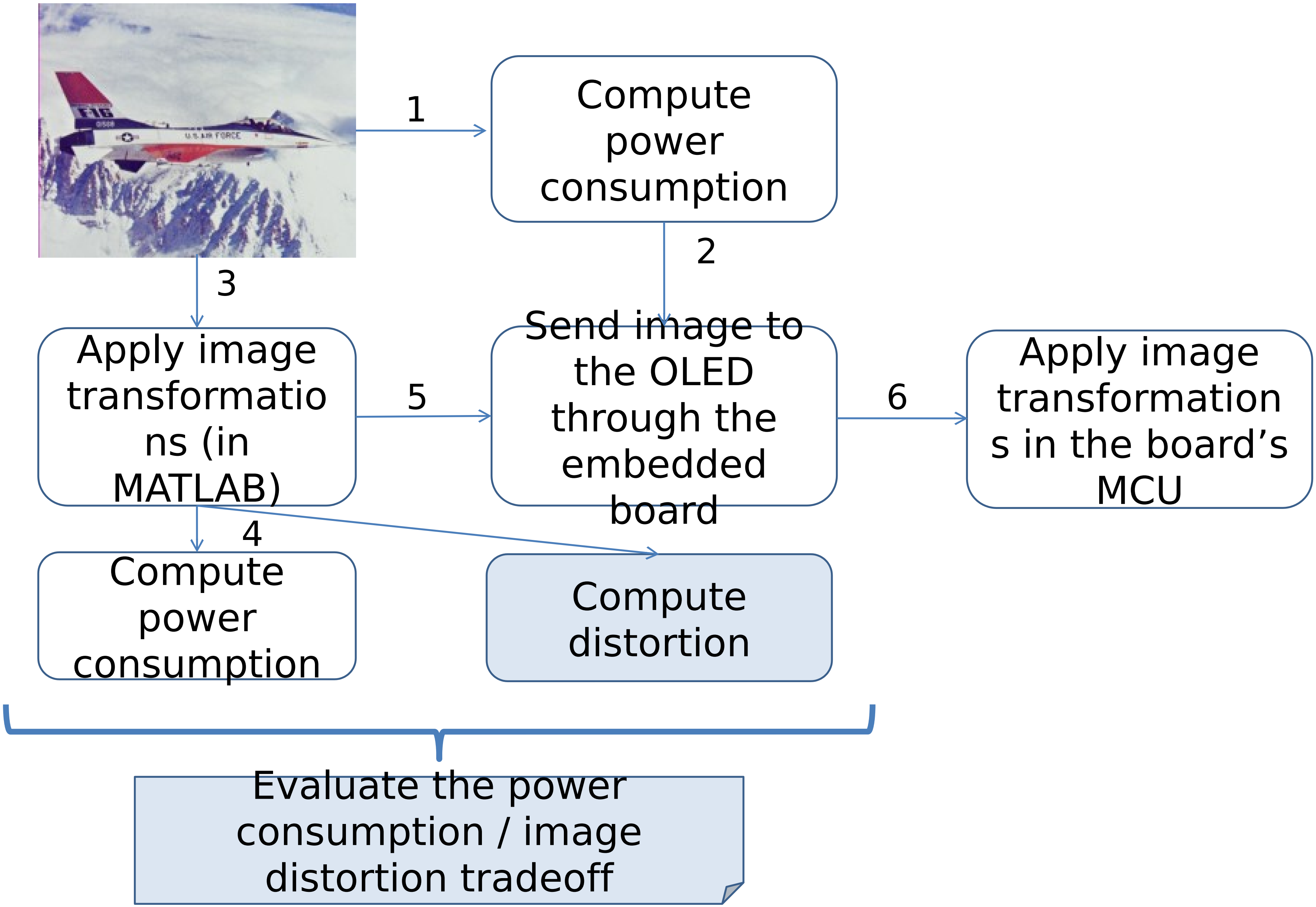


1
Compute power consumption
2
3
Apply image transformations (in MATLAB)
5
Send image to the OLED through the embedded board
6
Apply image transformations in the board’s MCU
4
Compute power consumption
Compute distortion

Evaluate the power consumption / image distortion tradeoff

3. Apply image transformations

- Same as Session 1:
  - Apply pixel-level transformations (blue reduction, pixel value scaling, cubic transformation, etc.)
  - Apply these transformations **before** processing the image for transmission



Compute power consumption

Send image to the OLED through the embedded board

Apply image transformations (in MATLAB)

Apply image transformations in the board’s MCU

Compute power consumption

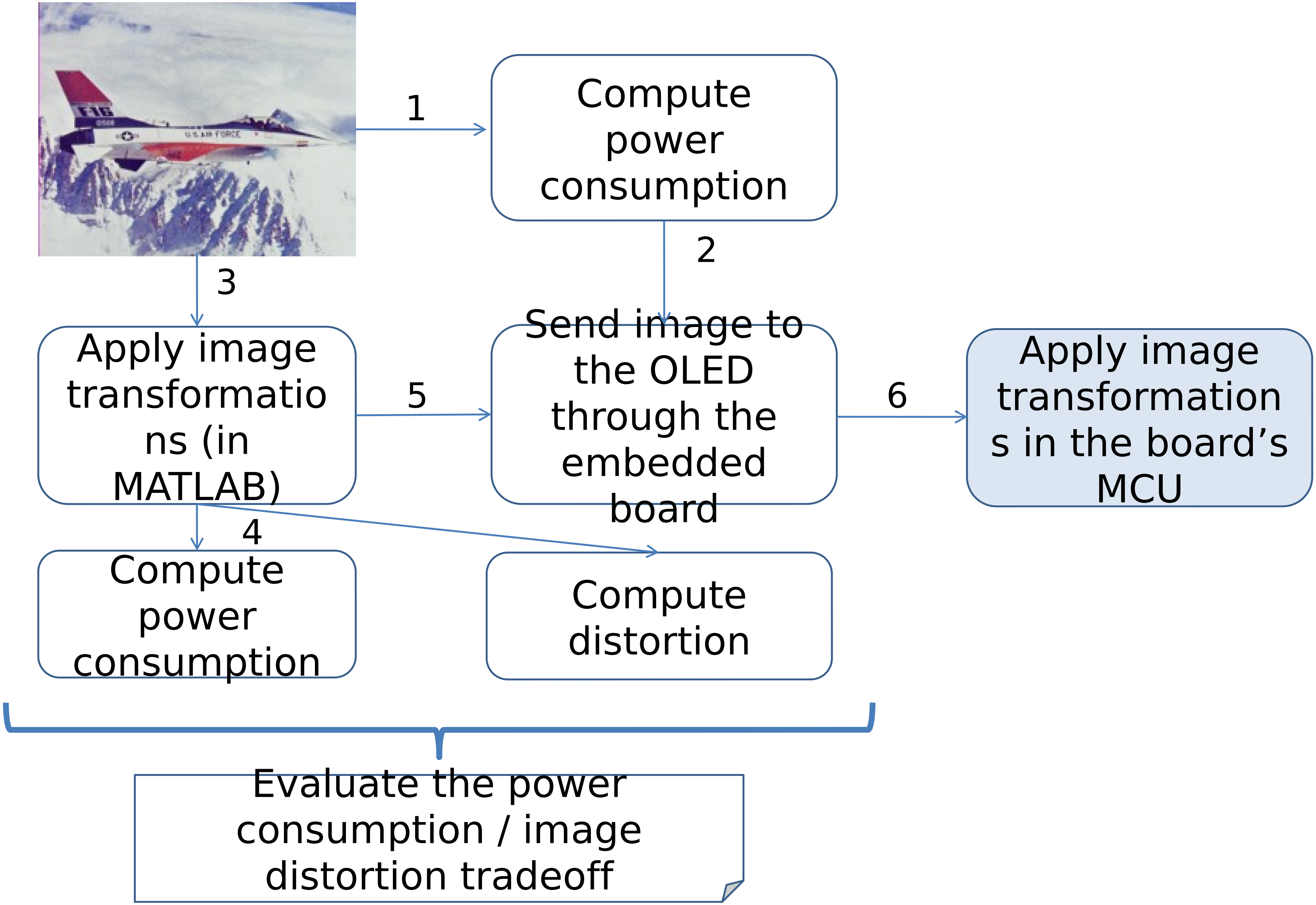
Compute distortion

Evaluate the power consumption / image distortion tradeoff

4. Compute distortion and evaluate trade-off

- Almost the same as Session 1

- Small difference:
  - Some «distortions» are eliminated when converting to 6-bit RGB and resizing the image



Compute power consumption

Apply image transformations (in MATLAB)

Send image to the OLED through the embedded board

Apply image transformations in the board's MCU

Compute power consumption

Compute distortion

Evaluate the power consumption / image distortion tradeoff

6. Apply image transformations in the board

- What kind of transformations can be applied?

- What are the differences compared to MATLAB?