Laboratory: Processing Thermographic Test Results in MATLAB

Course Objective

The aim of this lab is to become familiar with processing thermographic test data using a simulation of heating a sample with the 'AGH' inscription. Students will learn to:
- Load and visualize thermographic data,
- Perform temporal analysis,
- Apply PCA and PPT analysis,
- Binarize images and estimate 'defect' area,
- Conduct Fourier analysis of images.

Data Description and Temperature Profile

The file `termogram.mat` contains a variable `data` with dimensions [height, width, time].

The temperature profile is divided into three phases:
T1 – Initial temperature (artificially added): 0–0.1 s
T2 – Heating phase: 0.5 s
T3 – Cooling phase: 5.5 s

The initial temperature has been subtracted from the data to analyze only the temperature changes.

Students should:
- Determine the end of heating (T2) based on the temperature plot at point (100,100),
- Trim the data to the T3 phase (cooling),
- Perform PCA and PPT analysis only on this cooling data.

Key Methods

Principal Component Analysis (PCA)

PCA is a statistical method that extracts key patterns of change in data. In thermography, PCA reduces many time frames to a single image representing dominant temperature changes. This makes it easier to visualize subtle defects (like the 'AGH' text) while filtering out noise.

Pulse Phase Thermography (PPT)

PPT is based on Fourier analysis of time-domain signals. Instead of analyzing each frame separately, it transforms the signal into the frequency domain and generates images showing amplitude and phase of thermal response. Phase images are especially useful for defect detection as they are insensitive to heating non-uniformity and surface conditions.

General Guidelines

- You may (and are encouraged to) use ChatGPT, MATLAB documentation, and other online resources. Any code generated with ChatGPT must be fully understood by the user. If any part of the code is unclear, use ChatGPT to explain the functions and how they work.

- The report should include screenshots of results at each step, as well as screenshots of MATLAB code with comments explaining its function and showing user understanding.

- Teamwork is allowed (2 people), but each student must submit an individual report.

Your final report should include:

- A short summary (5–10 sentences),
- A temperature plot for pixel (100,100) with the end of heating marked,
- Screenshots of:
  - A selected frame,
  - PCA image and the resulting binary mask,
  - Estimated "defect" area from PCA,
  - PPT images (Amplitude and Phase),
  - Binary mask from the PPT phase image,
  - Estimated "defect" area from PPT,
- **Screenshots of MATLAB code** with **comments explaining its function** — comments must demonstrate the author's understanding of the code,
- A description of the functions used and the results obtained,
- Final conclusions

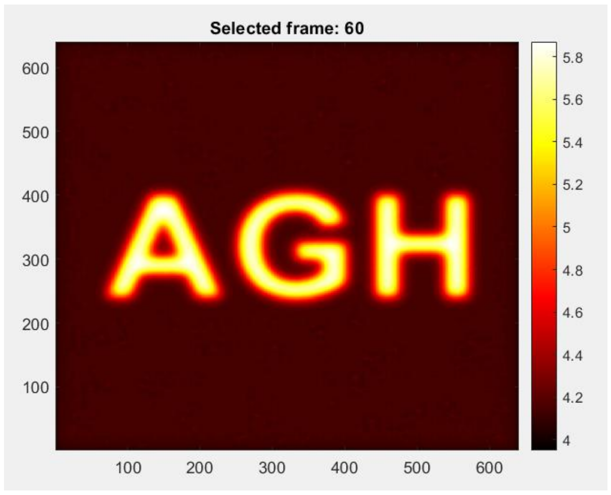
Tasks

1. Loading data and checking dimensions

Functions: load, size

2. Display a selected frame

Functions: imagesc, colormap, colorbar, title, set(gca, 'YDir', 'normal');

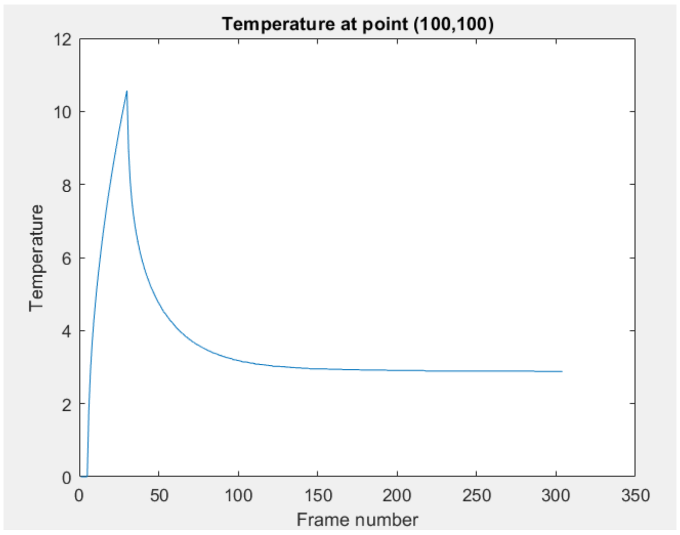


3. Temperature animation

Functions: for, pause, drawnow

4. Temperature plot at pixel (100,100)

Functions: squeeze, plot, xlabel, ylabel



5. Find end of heating (manually or automatically)

Functions: find, max

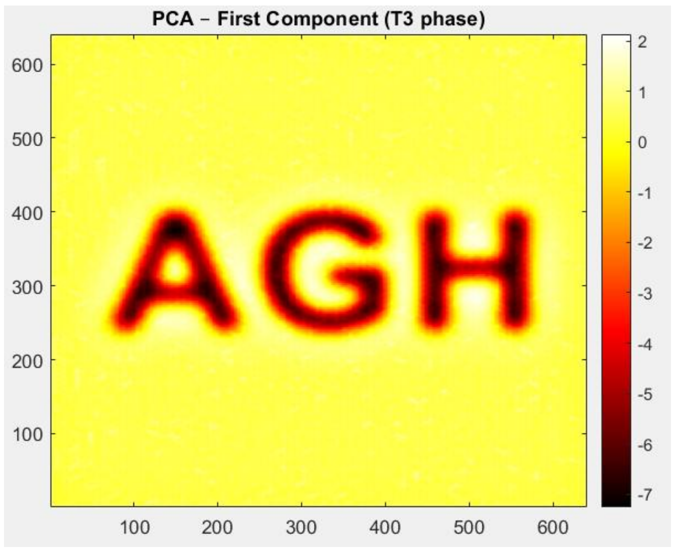
6. Trim data to cooling phase (T3)

Variable: data(:,:,frame_cutoff:end) → save to cooling_data
Dimensions cooling data save as [rows_cool, cols_cool, frames_cool]

7. PCA Analysis – T3 Phase Only

Use the code below:

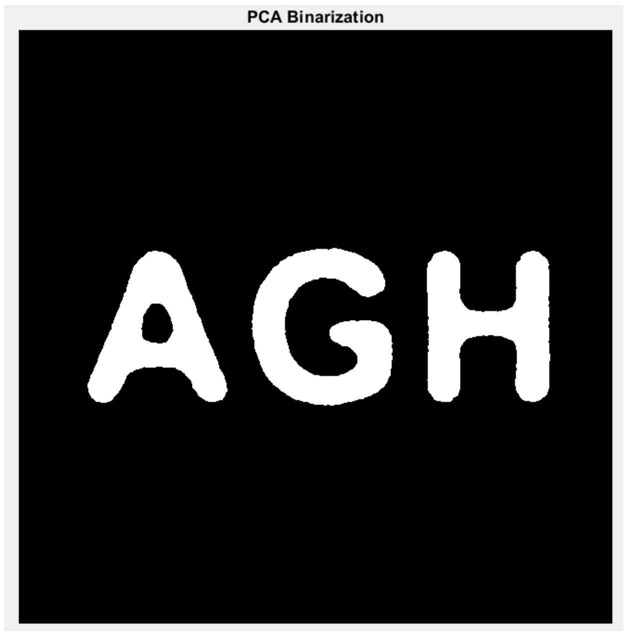
```
% 7. PCA on cooling phase (T3)
reshaped = reshape(cooling_data, rows_cool * cols_cool, frames_cool);

% Centering data
mean_signal = mean(reshaped, 2);  % mean for each pixel
centered_data = reshaped - mean_signal;

% PCA: columns = frames (variables), rows = pixels (observations)
[coeff, score, ~] = pca(centered_data);

% First PCA component
pca_image = reshape(score(:,1), rows_cool, cols_cool);

%%
% Displaying PCA
figure;
imagesc(pca_image);
colormap hot; colorbar;
set(gca, 'YDir', 'normal');
title('PCA - First Component (T3 phase)');
```



8. Binarization of PCA image, display of binarized image

Functions: graythresh, imbinarize, imshow

pca_norm = mat2gray(pca_image);   % normalization (binarization requires a range [0,1])
pca_inverted = 1 - pca_norm;   % value inversion (if the AGH string had lower values than the background)



9. Defect area calculation

100 mm = 640 Pixels ⇒ 1 pixel = 0.15625 mm
Functions: sum, *, ^2

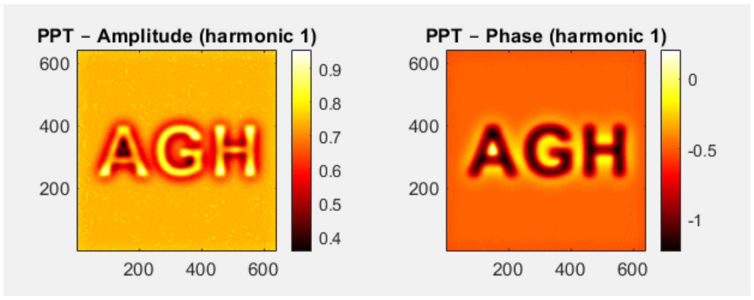
```
Defect area: 945.21 mm^2
```

10. PPT (Pulse Phase Thermography) – T3 phase only

Use the code below:

```
% 10. PPT (Pulse Phase Thermography)

% Define basic variables
L = size(cooling_data,3);  % Length of the signal

% FFT
Y = fft(cooling_data,[],3);

% Single-sided spectrum
P2 = abs(Y/L);
P1 = P2(:,:,1:floor(L/2)+1);
P1(2:end-1) = 2*P1(2:end-1);
```

```
% Assign amplitude and phase
Amplitude = P1;
Phase = angle(Y(:,:,1:floor(L/2)+1));

% Choose harmonic for analysis (e.g., 2 = 1st harmonic)
harmonic_idx = 2;
Amplitude_img = Amplitude(:,:,harmonic_idx);
Phase_img = Phase(:,:,harmonic_idx);

% Visualization
figure;
subplot(1,2,1);
imagesc(Amplitude_img);
set(gca, 'YDir', 'normal');
colormap hot; colorbar;
axis image;
title(['PPT - Amplitude (harmonic ', num2str(harmonic_idx-1), ')']);

subplot(1,2,2);
imagesc(Phase_img);
set(gca, 'YDir', 'normal');
colormap hot; colorbar;
axis image;
title(['PPT - Phase (harmonic ', num2str(harmonic_idx-1), ')']);
```



11. Binarization PPT

```
phase_norm = (Phase_img + pi) / (2 * pi);  % shift and scale
phase_inverted = 1 - phase_norm;     % invert if AGH area had lower values than backgorund
```

Functions: graythresh, imbinarize, imshow

12. Defect Area Calculation

100 mm = 640 Pixels ⇒ 1 pixel = 0.15625 mm

Functions: sum, *, ^2

```
Defect area: 904.52 mm^2
```