Image Compression Using SVD

We consider an grayscale image $A \in \mathbb{R}^{n \times n}$ with entries $A_{ij} \in [0,1]$ representing the gray intensity.

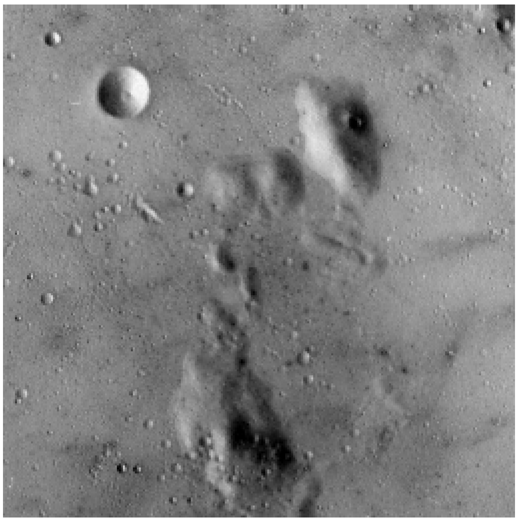
Singular Value Decomposition

A square matrix $A \in \mathbb{R}^{n \times n}$ can be written as SVD, defined as:

$$A = U\Sigma V^T = \sum_{i=1}^{n} u_i \sigma_i v_i^T = u_1 \sigma_1 v_1^T + \cdots + u_r \sigma_r v_r^T$$

```
using Images, TestImages, LinearAlgebra, PlutoUI, Plots
```

Load test image from the `TestImages` package.



```
begin
    # img = float.(testimage("lena_gray_512"))
    img = float.(testimage("moonsurface"))
end
```

Compressed Image

We construct the compressed image $\tilde{A} \in \mathbb{R}^{n \times n}$ as rank $r = 13$ approimxation, defined as:

$$\tilde{A} = \sum_{i=1}^{r} u_i \sigma_i v_i^T = u_1 \sigma_1 v_1^T + \cdots + u_r \sigma_r v_r^T$$

with rank r.

Storage Requirement of Compressed Matrix

Instead of storing n^2 matrix entires, we could now only store the r-times the summation tuple $\{u_i, \sigma_i, v_i^T\}$ which leads to a size

$$\text{size}(\tilde{A}) = r(n+1+n) = r(2n+1) \ll n^2 = \text{size}(A) \text{ for } r \ll n$$

```
compressed (generic function with 1 method)
```

```
function compressed(img, rank)
    U, Σ, Vt = svd(img);
    return Gray.(sum([U[:,i] * Σ[i] * Vt[:,i]' for i=1:rank])) # Gray
end
```

13

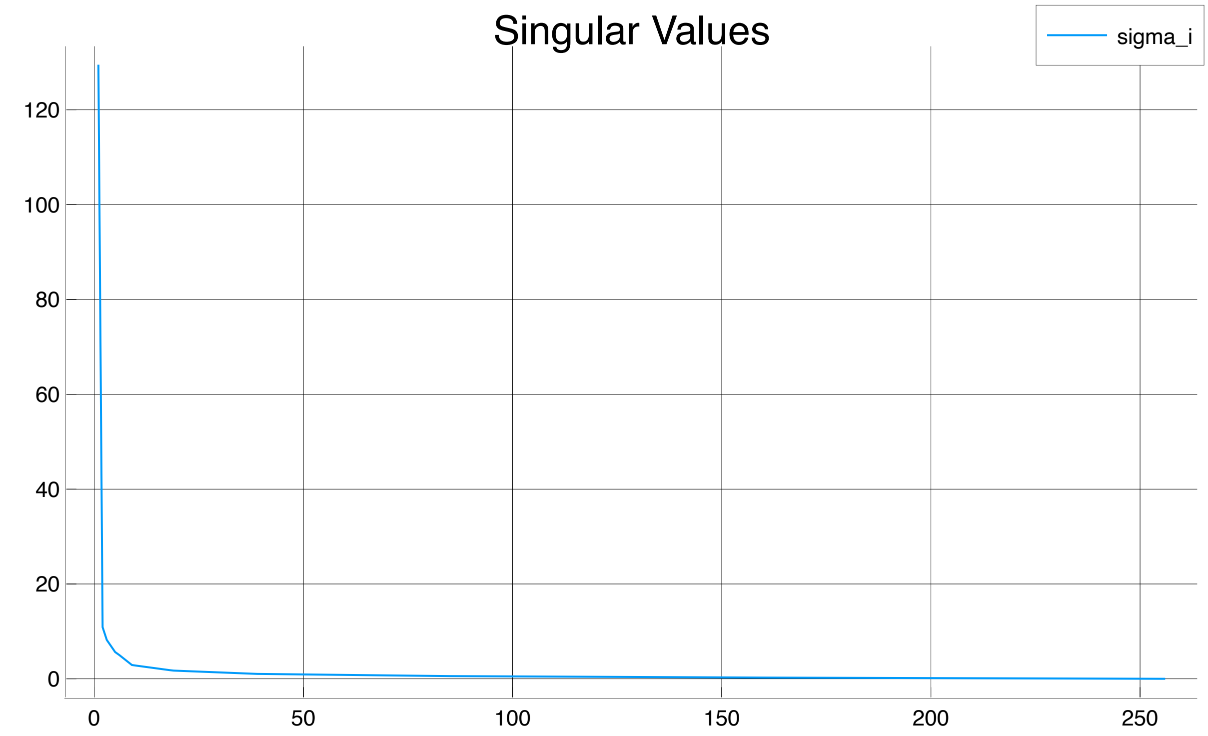
```
compressed(img, rank)
```

Check the Singular Values

Rule of thumb: *If the decrease of SVs is strong, we have a low rank stucture and can compress.*

```
Plots.PlotlyBackend()
```

```
plotly()
```




```
plot(svd(img).S, title="Singular Values", label="sigma_i")
```