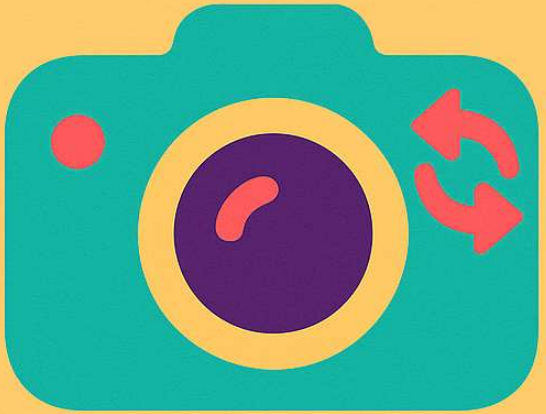


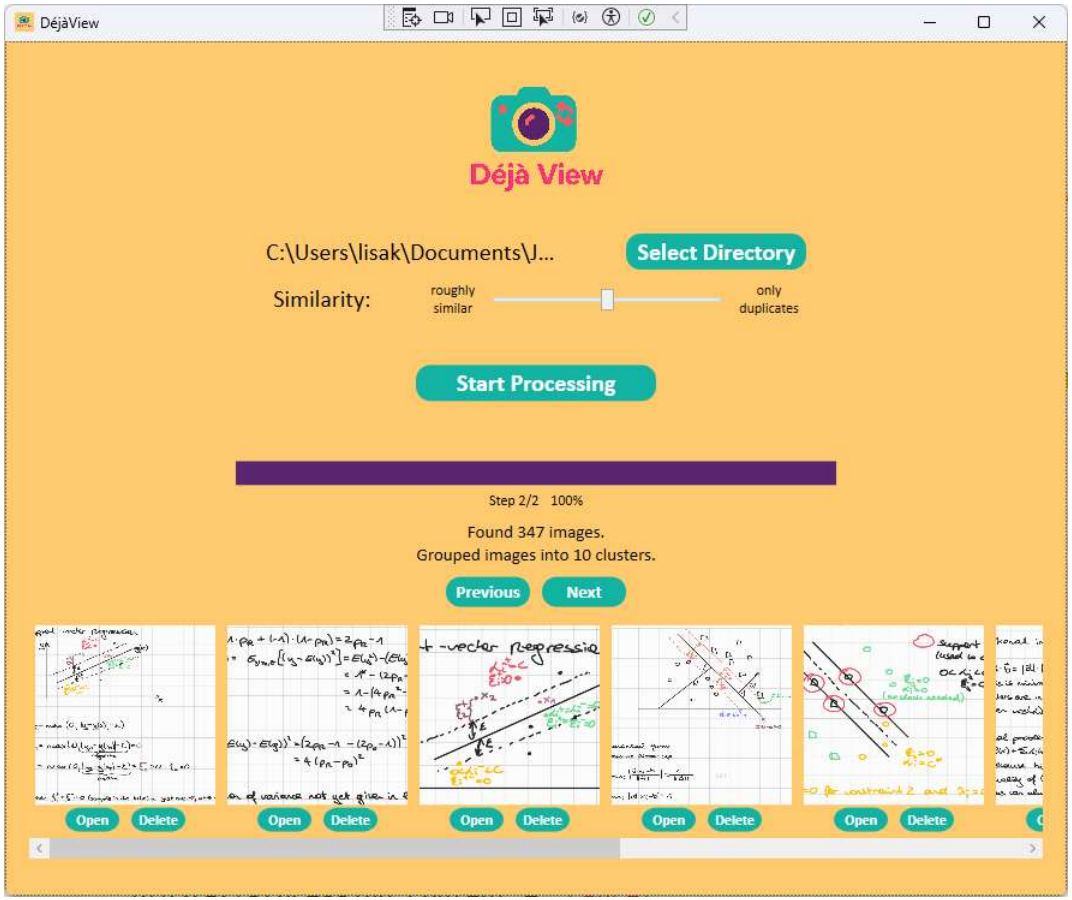
Déjà View

Martin Dallinger, Lisa Krimbacher

JOHANNES KEPLER UNIVERSITY LINZ

Find and Remove Similar Images

- WPF application
- Usage of TAP/TPL
- Define degree of similarity, group images into clusters
- Images are loaded asynchronously and embedded in parallel
- Open images in default application
- Delete images



Find and Remove Similar Images

Retrieve all image paths from directory and subdirectories (I/O operation: async)

```
// ConfigureAwait(false): don't necessarily come back to UI Thread
IEnumerable<string> directories = await Task.Run(() =>
            Directory.EnumerateDirectories(rootDirectory, "*", SearchOption.AllDirectories)
                                                .Prepend(rootDirectory), cancellationToken)
                                                .ConfigureAwait(false);

// Process each directory sequentially in an async loop.
foreach (string dir in directories)
{
    cancellationToken.ThrowIfCancellationRequested();
    try
    {
        string[] files = await Task.Run(() => Directory.GetFiles(dir), cancellationToken)
            .ConfigureAwait(false);

        foreach (string file in files)
        {
            cancellationToken.ThrowIfCancellationRequested();

            if (imageExtensions.Contains(Path.GetExtension(file)))
                result.Add(file);
        }
    }
    catch (UnauthorizedAccessException)
    {
        // Count directories we were not permitted to access
        nSkippedDirectories++;
    }
    catch (IOException)
    {
        // Count directories that encountered an I/O exception
        nIOExceptions++;
    }
}
```

Find and Remove Similar Images

Retrieve all image paths from directory and subdirectories (I/O operation: async)

Load images (async) and create embeddings with AI model (long-running: parallel)

```
int maxDegreeOfParallelism = Math.Max(1, Environment.ProcessorCount / 2); // Reduce load
ConcurrentDictionary<string, float[]> results = new ConcurrentDictionary<string, float[]>();

await Parallel.ForEachAsync(
    filePaths,
    new ParallelOptions
    {
        MaxDegreeOfParallelism = maxDegreeOfParallelism,
        CancellationToken = cancellationToken
    },
    async (file, token) =>
    {
        try
        {
            byte[] content = await File.ReadAllBytesAsync(file, token);
            results[file] = await Task.Factory.StartNew(
                () => SharedProcessorMobileNet.RunInference(content),
                token,
                TaskCreationOptions.LongRunning, // Creates a new Thread as RunInference is CPU-heavy
                TaskScheduler.Default
            );
        }
        catch (Exception)
        {
            Interlocked.Increment(ref nSkippedFiles);
        }

        if (progress != null)
        {
            int newCount = Interlocked.Increment(ref processedCount);
            progress.Report((int)Math.Ceiling(((double)newCount) / nFilePaths * 100));
        }
    });
```

Find and Remove Similar Images

Retrieve all image paths from directory and subdirectories (I/O operation: async)

↓

Load images (async) and create embeddings with AI model (long-running: parallel)

↓

Cluster embeddings (parallel)

```
// Do not block the caller
return await Task.Run(() =>
{
    List<string> filePaths = imageVectors.Keys.ToList();
    int n = filePaths.Count;
    ConcurrentBag<(int, int)> similarPairs = new ConcurrentBag<(int, int)>();

    // Counter for completed outer iterations
    int nCompletedImages = 0;

    // No shared mutual state, usage of Parallel.For possible
    ParallelOptions options = new ParallelOptions { CancellationToken = cancellationToken };
    Parallel.For(0, n, options, i =>
    {
        for (int j = i + 1; j < n; j++)
        {
            float similarity = CosineSimilarity(imageVectors[filePaths[i]], imageVectors[filePaths[j]]);
            if (similarity >= similarityThreshold)
                similarPairs.Add((i, j));
        }

        // Update progress after processing each image
        if (progress != null)
        {
            int done = Interlocked.Increment(ref nCompletedImages);
            double percent = ((double)done) / n * 100;
            progress.Report((int)Math.Ceiling(percent));
        }
    });

    // Union similar pairs to clusters (synchronously) ...
```

Find and Remove Similar Images

Retrieve all image paths from directory and subdirectories (I/O operation: async)

↓

Load images (async) and create embeddings with AI model (long-running: parallel)

↓

Cluster embeddings (parallel)

↓

Add clustered images to the UI (sync in UI thread)

```
try
{
    foreach (string imagePath in clusters[currClusterId])
    {
        StartOverCancellationToken.Token.ThrowIfCancellationRequested();
        await Task.Delay(10); // Free up the UI thread periodically
        imageWrapPanel.Children.Add(CreateImageContainer(imagePath));
    }
}
catch (OperationCanceledException)
{
    ClearImages();
}
```

Benchmarking: Read + Embedding

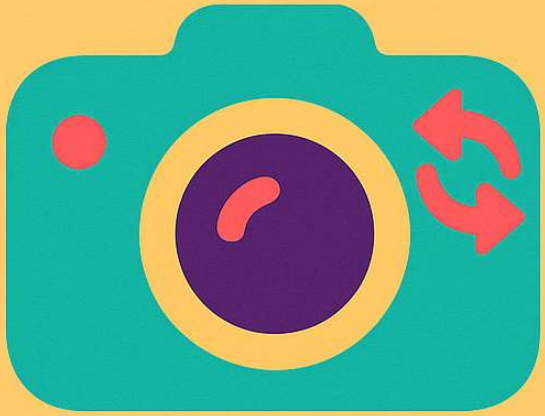
Executing with the library BenchmarkDotNet in a console process (390 images):

```
// * Summary *

BenchmarkDotNet v0.14.0, Windows 11 (10.0.26100.3775)
13th Gen Intel Core i7-13700H, 1 CPU, 20 logical and 14 physical cores
.NET SDK 9.0.201
  [Host]     : .NET 8.0.14 (8.0.1425.11118), X64 RyuJIT AVX2 [AttachedDebugger]
  Job-ZOKWSC : .NET 8.0.14 (8.0.1425.11118), X64 RyuJIT AVX2

InvocationCount=1  IterationCount=10  UnrollFactor=1
WarmupCount=1

| Method                                    | Mean     | Error    | StdDev   |
|------------------------------------------ |--------:|--------:|--------:|
| ProcessAllFilesLongRunningForEachAsync    | 13.34 s | 1.636 s | 0.855 s |
| ProcessAllFilesNoLongRunningForEachAsync  | 17.85 s | 5.113 s | 3.382 s |
| ProcessAllFilesLongRunningForEach         | 20.79 s | 5.685 s | 3.383 s |
| ProcessAllFilesNoLongRunningForEach       | 20.22 s | 1.281 s | 0.762 s |
| ProcessAllFilesNoParallelization          | 23.94 s | 2.123 s | 1.263 s |
```



Déjà View

Summary

- Async and parallel patterns
  → Find semantically similar images

- 1000 floats per image (embedding) are retained
  - Memory-efficient: Worker-thread discards the Bitmap information
  - Stable for bigger datasets

- Questions from your side?

GitHub