AJEA – Actas de Jornadas y Eventos Académicos de UTN
Universidad Tecnológica Nacional
30 y 31 de octubre de 2024
Libro de Actas de Resúmenes DOI: https://doi.org/10.33414/ajea.1846.2025
Enlace: https://rtyc.utn.edu.ar/index.php/ajea/article/view/1837

Aplicación de Análisis de Componentes Principales para la Detección de Fisuras en Álabes de Turbina

Application of Principal Component Analysis for Crack Detection in Turbine Blades

Presentación: 30/10/2024

Augusto Riedinger
Universidad Tecnológica Nacional, Facultad Regional Bahía Blanca, Argentina.
riedinger.augusto@gmail.com

Héctor R. Bambill
Universidad Tecnológica Nacional, Facultad Regional Bahía Blanca, Argentina.
heribam@utn.frbb.ar

Patricia Baldini
Universidad Tecnológica Nacional, Facultad Regional Bahía Blanca, Argentina.
pnbaldi@ utn.frbb.ar

Resumen

En este trabajo se implementa el Análisis de Componentes Principales (PCA) para la detección de fisuras en álabes de turbina de gas. El problema simulado consiste en una fábrica que produce álabes en serie, donde algunas piezas presentan fisuras como resultado del proceso de fabricación. A partir de imágenes en blanco y negro de los álabes, que se consideran matrices de píxeles, se busca identificar las piezas defectuosas. El objetivo del trabajo es desarrollar un sistema automatizado capaz de detectar estas anomalías mediante PCA.

La metodología aplicada incluye la simulación gráfica de N imágenes de álabes, la conversión de cada imagen en un vector columna, y la conformación de una matriz de datos de M x N. A continuación, se calculan los autovectores y autovalores de la matriz de covarianza, seleccionando los componentes principales más significativos. Finalmente, se proyectan los datos en un nuevo espacio de menor dimensión y se establecen límites de decisión para identificar los álabes defectuosos.

El método implementado demostró ser efectivo para detectar álabes con fisuras, permitiendo su separación para reparación o descarte.

Palabras clave: PCA, detección de fisuras, álabes de turbinas.

Abstract

In this work, Principal Component Analysis (PCA) is implemented for detecting cracks in gas turbine blades. The simulated problem involves a factory that produces blades in series, where some pieces exhibit cracks as a result of the manufacturing process. Based on black-and-white images of the blades, considered as pixel matrices, the goal is to identify defective pieces. The objective of the work is to develop an automated system capable of detecting these anomalies

Libro de Actas de Resúmenes DOI: https://doi.org/10.33414/ajea.1846.2025
Enlace: https://rtyc.utn.edu.ar/index.php/ajea/article/view/1837

using PCA.

The methodology applied includes the graphical simulation of N blade images, converting each image into a column vector, and forming a data matrix of M x N. Then, the eigenvectors and eigenvalues of the covariance matrix are calculated, selecting the most significant principal components. Finally, the data is projected into a new, lower-dimensional space, and decision boundaries are established to identify defective blades.

The implemented method proved effective in detecting blades with cracks, allowing their separation for repair or disposal.

Keywords: Instructions: PCA, crack detection, turbine blades.

Introducción

El Análisis de Componentes Principales (PCA, por sus siglas en inglés) es una técnica de reducción de dimensionalidad ampliamente utilizada en el campo del aprendizaje automático y la estadística (Jolliffe, 2016). PCA transforma un conjunto de datos de alta dimensionalidad en un espacio de menor dimensión, preservando la mayor parte de la variabilidad presente en los datos originales. Esto se logra identificando las direcciones (componentes principales) en las que los datos varían más y proyectando los datos a lo largo de estas direcciones (Bishop, 2006). En esencia, PCA busca encontrar una representación más simple de los datos que retenga la mayor cantidad de información posible. Al hacerlo, no solo facilita la visualización y el procesamiento de los datos, sino que también ayuda a identificar patrones ocultos, tendencias y posibles anomalías (Yang, 2004). Esto es particularmente útil cuando se trabaja con datos complejos y de alta dimensionalidad, como imágenes, donde cada píxel puede ser considerado una dimensión diferente. El concepto de PCA fue introducido por el matemático Karl Pearson en 1901. Pearson, conocido por sus contribuciones a la estadística, desarrolló PCA como una técnica para identificar patrones en los datos y reducir la dimensionalidad de grandes conjuntos de variables. A lo largo de las décadas siguientes, PCA fue refinada y ampliada por varios investigadores (Shlens, 20014). En la década de 1930, Harold Hotelling, un estadístico estadounidense, contribuyó significativamente a la formalización de PCA, extendiéndola a lo que se conoce como "análisis de factores" en psicometría (Hotelling, 1933). Durante la segunda mitad del siglo XX, PCA comenzó a ser ampliamente utilizado en una variedad de disciplinas, incluyendo biología, economía y climatología, para la simplificación de modelos y la identificación de patrones subyacentes en datos complejos (Pearson, 1901). Con el advenimiento de la computación moderna y la explosión de grandes volúmenes de datos, PCA se ha convertido en una técnica fundamental en el aprendizaje automático y la inteligencia artificial. En la era del Big Data, donde los conjuntos de datos pueden tener miles o incluso millones de variables, la capacidad de PCA para reducir la dimensionalidad sin perder información esencial es invaluable. En la actualidad, PCA es ampliamente utilizado en áreas como Visión por Computadora, Bioinformática, Finanzas, Procesamiento de Señales, y Ciencia de Materiales. El avance en algoritmos como el PCA Kernelizado y la integración con técnicas de Deep Learning han ampliado aún más su aplicabilidad, manteniéndose como una herramienta clave para científicos y analistas de datos en la actualidad .

Detección de Fisuras en Álabes de Turbina usando PCA

Una fisura en un álabe de turbina, si no es detectada y tratada a tiempo, puede progresar y llevar a una falla catastrófica de la turbina. Por ello es fundamental implementar un monitoreo continuo del estado de los álabes que son los componentes críticos dado que operan bajo condiciones extremas de temperatura y estrés mecánico. Métodos no destructivos como la termografía, ultrasonidos, y análisis de vibraciones, junto con técnicas como la que en este trabajo presentamos, Análisis de Componentes Principales (PCA) aplicado a imágenes de álabes, pueden detectar anomalías que indiquen la presencia de fisuras en etapas tempranas permitiendo la reparación oportuna o su reemplazo. La implementación del método requiere:

1. Preparación y Preprocesamiento de las Imágenes: Se recopilan imágenes de álabes

Libro de Actas de Resúmenes DOI: https://doi.org/10.33414/ajea.1846.2025
Enlace: https://rtyc.utn.edu.ar/index.php/ajea/article/view/1837

defectuosos como sanos que conforman una base de datos conocidos. Las imágenes son normalizadas en términos de tamaño, escala de grises, y nivel de iluminación para garantizar la consistencia, que de manera práctica son fotografías de los álabes tomadas en condiciones similares. Cada imagen, que es una matriz 2D de píxeles, se convierte en un vector columna 1D apilando sus columnas. Así se generará una matriz de datos de tamaño M x N, donde M representa el número de píxeles en cada imagen y N es el número total de imágenes de la base de datos, permitiendo que cada imagen se represente como un punto en un espacio de alta dimensión.

2. Construcción de la Matriz de Datos y Aplicación de PCA: La matriz de datos se construye utilizando estos vectores, donde cada columna corresponde a una imagen de álabe. PCA se aplica para encontrar las principales direcciones de variabilidad en los datos, reduciendo la dimensionalidad al espacio definido por los primeros componentes principales.
3. Detección de Fisuras mediante Análisis de Anomalías: Se proyectan los datos de las imágenes de álabes sanos en el espacio reducido, creando un modelo de lo que constituye un álabe "normal". Imágenes de álabes nuevos se proyectan en este mismo espacio. Si un álabe proyectado se desvía significativamente del patrón de los álabes sanos, puede ser identificado como anómalo, indicando la posible presencia de una fisura.
4. Validación y Evaluación: Se validan los resultados utilizando conjuntos de datos de prueba que contienen tanto álabes sanos como con fisuras. Se evalúa la capacidad del modelo para identificar correctamente las fisuras mediante métricas como la sensibilidad, especificidad y precisión.

Con este enfoque no solo automatiza y mejora la precisión en la detección de fisuras, sino que también reduce el tiempo y los costos asociados con la inspección manual, garantizando la seguridad y la eficiencia en la operación de turbinas.

Algoritmo de resolución

El código presentado implementa un algoritmo para la detección de diferencias significativas entre imágenes mediante un enfoque que combina procesamiento de imágenes y análisis estadístico avanzado.

El algoritmo de resolución está basado en el Algoritmo 1.

Algoritmo 1	Detección de fisuras
Inicio	Para cada imagen en una lista:
Paso 1	Intentar abrir la imagen
Paso 2	Convertir la imagen a formato OpenCV (BGR)
Paso 3	Definir un rango de colores para detectar "blades" en la imagen
Paso4	Crear una máscara binaria que solo conserve los colores dentro del rango definido
Paso 5	Aplicar la máscara binaria a la imagen original y devolver la imagen resultante
Paso 6	Convertir la imagen a escala de grises
Paso 7	Aplicar un umbral adaptativo (thresholding) para binarizar la imagen y devolver la imagen resultante
Paso 8	Convertir la imagen a un arreglo y aplanarlo (flatten)
Paso 9	Combinar las imágenes aplanadas en una única matriz
Paso 10	Aplicar PCA a la matriz combinada
Paso 11	Evaluar la varianza de los componentes principales para determinar si hay una diferencia significativa entre las dos imágenes
Fin	Si la variación es significativa, devolver True; en caso contrario, devolver False.

AJEA – Actas de Jornadas y Eventos Académicos de UTN
Universidad Tecnológica Nacional
30 y 31 de octubre de 2024
Libro de Actas de Resúmenes DOI: https://doi.org/10.33414/ajea.1846.2025
Enlace: https://rtyc.utn.edu.ar/index.php/ajea/article/view/1837

A continuación se describen las partes integrales del algoritmo.

Bibliotecas utilizadas

Se utilizaron las siguientes bibliotecas:

```
from PIL import Image as img
import numpy as np
from sklearn.decomposition import PCA
import os
import sys
import cv2
import warnings
```

Donde:

- PIL: Utilizada para la manipulación y apertura de imágenes.
- NumPy: Proporciona soporte para arrays multidimensionales y funciones matemáticas.
- Scikit-learn (PCA): Para la reducción de dimensionalidad utilizando Análisis de Componentes Principales (PCA).
- OpenCV: Utilizada para el procesamiento de imágenes, en particular para la manipulación y análisis de imágenes en formato BGR.
- os y sys: Manejo del sistema de archivos y argumentos de línea de comandos.
- warnings: Controla la emisión de advertencias durante la ejecución.

Lectura de imagenes

```
def readImage(filePath):
      try:
      image = img.open(filePath)
      return image
      except IOError:
      print(f"Unable to open image file: {filePath}")
      return None
```

Esta función intenta cargar una imagen desde una ruta especificada utilizando PIL. Si la imagen se carga correctamente, se devuelve el objeto imagen. En caso de un fallo (por ejemplo, si el archivo no existe o es ilegible), se imprime un mensaje de error y se devuelve None.

Segmentación de imagenes

```
def segmentBlades(image):
      opencvImage = cv2.cvtColor(np.array(image), cv2.COLOR_RGBA2BGR)

      lower_bound = np.array([0, 0, 0], dtype=np.uint8)
      upper_bound = np.array([100, 100, 100], dtype=np.uint8)

      mask = cv2.inRange(opencvImage, lower_bound, upper_bound)

      resultImage    =    img.fromarray(cv2.bitwise_and(opencvImage,    opencvImage,
mask=mask))

      return resultImage
```

El propósito de esta función es segmentar una parte específica de la imagen, definida por un rango de colores. El procedimiento es el siguiente:

AJEA – Actas de Jornadas y Eventos Académicos de UTN
Universidad Tecnológica Nacional
30 y 31 de octubre de 2024
Libro de Actas de Resúmenes DOI: https://doi.org/10.33414/ajea.1846.2025
Enlace: https://rtyc.utn.edu.ar/index.php/ajea/article/view/1837

- Conversión de formato: La imagen se convierte del formato RGBA al formato BGR, que es el estándar utilizado por OpenCV.
- Definición del rango de colores: Se especifica un rango de colores en BGR para identificar las regiones de interés ("blades") en la imagen. Estos valores de límite son ajustables.
- Creación de la máscara binaria: Se genera una máscara binaria donde los píxeles que caen dentro del rango de colores definido se marcan como 1 (blanco), y los demás como 0 (negro).
- Aplicación de la máscara: La máscara se aplica a la imagen original, conservando solo las áreas de interés.

Umbral adaptativo

```
def dynamicThresholding(image):
      grayscaleImage = cv2.cvtColor(np.array(image), cv2.COLOR_RGBA2GRAY)

      _, thresholdedImage = cv2.threshold(grayscaleImage, 0, 255, cv2.THRESH_BINARY
+ cv2.THRESH_OTSU)

      return img.fromarray(thresholdedImage)
```

Esta función aplica un umbral adaptativo para binarizar la imagen, lo que facilita la detección de bordes y formas relevantes. El proceso es:

- Conversión a escala de grises: La imagen se convierte a un formato en escala de grises, reduciendo la complejidad y enfocando el análisis en la intensidad de los píxeles.
- Umbralización: Se aplica la técnica de Otsu para determinar un umbral óptimo que binariza la imagen, separando los píxeles en dos grupos: fondo y objeto.

Aplanamiento de imagenes

```
def flattenImage(image):
      return np.array(image).flatten()
```

La función convierte la imagen en un array unidimensional. Esta representación aplanada es esencial para el análisis posterior, ya que facilita la comparación y manipulación matemática de las imágenes.

Detección de diferencias entre imagenes

```
def detectImageDifference(image1, image2):
      thresholdedImage1 = dynamicThresholding(segmentBlades(image1))
      thresholdedImage2 = dynamicThresholding(segmentBlades(image2))

      flattenedImage1 = flattenImage(thresholdedImage1)
      flattenedImage2 = flattenImage(thresholdedImage2)

      combinedMatrix = np.vstack((flattenedImage1, flattenedImage2))

      with warnings.catch_warnings():
      warnings.simplefilter("ignore")  # Suppress the warnings
      pca = PCA()
      principalComponents = pca.fit_transform(combinedMatrix)

      if np.var(principalComponents, axis=0)[0] > 0.01:
      return True
      else:
```

AJEA – Actas de Jornadas y Eventos Académicos de UTN
Universidad Tecnológica Nacional
30 y 31 de octubre de 2024
Libro de Actas de Resúmenes DOI: https://doi.org/10.33414/ajea.1846.2025
Enlace: https://rtyc.utn.edu.ar/index.php/ajea/article/view/1837

```
    return False
```

Esta función es el núcleo del algoritmo y se encarga de detectar diferencias significativas entre dos imágenes:

- Segmentación y umbralización: Primero, las dos imágenes se segmentan utilizando la función segmentBlades y luego se binarizan con dynamicThresholding.
- Aplanamiento de imágenes: Ambas imágenes binarizadas se convierten en arrays unidimensionales.
- Combinación de datos: Los arrays resultantes se apilan para formar una matriz que será utilizada en el análisis de componentes principales.
- Aplicación de PCA: Se aplica PCA para reducir la dimensionalidad de los datos y extraer las características principales que explican la variabilidad entre las imágenes.
- Evaluación de la variabilidad: La varianza de los componentes principales se evalúa para determinar si hay una diferencia significativa. Si la varianza del primer componente principal excede un umbral (0.01 en este caso), se concluye que hay una diferencia significativa entre las imágenes.

Bloque principal

El bloque principal del programa se encarga de la gestión de los archivos y la comparación de imágenes:

```
if __name__ == "__main__":
    if len(sys.argv) < 3:
    print("Please provide a folder path and a base image filename as arguments.")
    sys.exit(1)

    folderPath = sys.argv[1]
    baseImagePath = sys.argv[2]

    baseImage = readImage(baseImagePath)

    if baseImage is None:
    print("Unable to read the base image.")
    sys.exit(1)

    imageFiles = sorted([file for file in os.listdir(folderPath) if
file.endswith(".png") or file.endswith(".jpg")])
    print("Reading images list:")
    print(imageFiles)

    if len(imageFiles) < 2:
    print("Please provide at least two image files in the folder.")
    sys.exit(1)

    for i in range(0, len(imageFiles)):  # Start from the second image
    imagePath = os.path.join(folderPath, imageFiles[i])
    currentImage = readImage(imagePath)

    if currentImage is not None:
        with warnings.catch_warnings():
            warnings.simplefilter("ignore")  # Suppress the warnings
            result = detectImageDifference(baseImage, currentImage)
        print(f"File {imageFiles[i]}: {'failure' if result else 'pass'}.")
```

AJEA – Actas de Jornadas y Eventos Académicos de UTN
Universidad Tecnológica Nacional
30 y 31 de octubre de 2024
Libro de Actas de Resúmenes DOI: https://doi.org/10.33414/ajea.1846.2025
Enlace: https://rtyc.utn.edu.ar/index.php/ajea/article/view/1837

Resultados

Para validar el rendimiento del algoritmo, se utilizó un conjunto de 25 imágenes de álabes, como se muestra en la Figura 1, donde algunas presentaban fracturas de diversa magnitud y distribución, mientras que otras no contenían defectos. Los resultados fueron consistentes con las expectativas.
El conjunto de fallas incluía:

- Imágenes con fracturas grandes y medianas que fueron correctamente identificadas como defectuosas.
- Imágenes con fracturas menores que también fueron detectadas de manera precisa.
- Incluso cambios en el fondo de la imagen no afectaron la capacidad del algoritmo para discernir la integridad del objeto de interés.

Figura 1: Álabe utilizado como entrada del Algoritmo.

Conclusiones

En el presente trabajo, se desarrolló y validó un algoritmo basado en técnicas de procesamiento de imágenes y análisis estadístico para la detección automática de fracturas en álabes. El enfoque se centra en la segmentación de la imagen y la aplicación de umbralización dinámica, seguida de un análisis de componentes principales (PCA) para evaluar la variabilidad significativa entre imágenes comparadas.

El algoritmo demostró una capacidad robusta para distinguir imágenes defectuosas de aquellas sin defectos, logrando una precisión del 100% en la detección de las seis imágenes con fallas dentro del conjunto de prueba de 25 imagenes totales. Estos resultados subrayan la eficacia del enfoque propuesto para aplicaciones industriales y de control de calidad, donde la detección automática de defectos es crucial para garantizar la integridad del producto.

Referencias

1. Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
2. Jolliffe, I. T., & Cadima, J. (2016). *Principal Component Analysis: A Review and Recent Developments*. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374(2065), 20150202.
3. Hotelling, H. (1933). *Analysis of a Complex of Statistical Variables into Principal Components*. Journal of Educational Psychology, 24(6), 417-441.
4. Pearson, K. (1901). *On Lines and Planes of Closest Fit to Systems of Points in Space*. Philosophical Magazine, 2(11), 559-572.
5. Shlens, J. (2014). *A Tutorial on Principal Component Analysis*. arXiv preprint arXiv:1404.1100.
6. Yang, J., Zhang, D., Frangi, A. F., & Yang, J. Y. (2004). *Two-Dimensional PCA: A New Approach to Appearance-Based Face Representation and Recognition*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(1), 131-137.