



UNIVERSITÀ DEGLI STUDI DI PARMA



# Using small checkerboards as size reference: A model-based approach

Hamid Hassannejad, Guido Matrella, Monica Mordonini, and Stefano Cagnoni

1<sup>st</sup> International Workshop on Multimedia Assisted Dietary Management  
September 2015

# Introduction

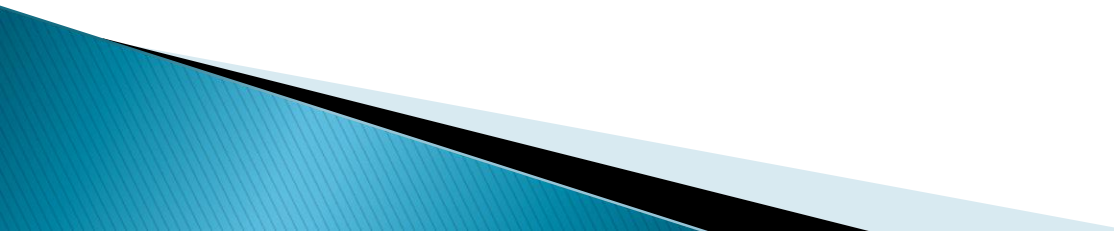


# Introduction

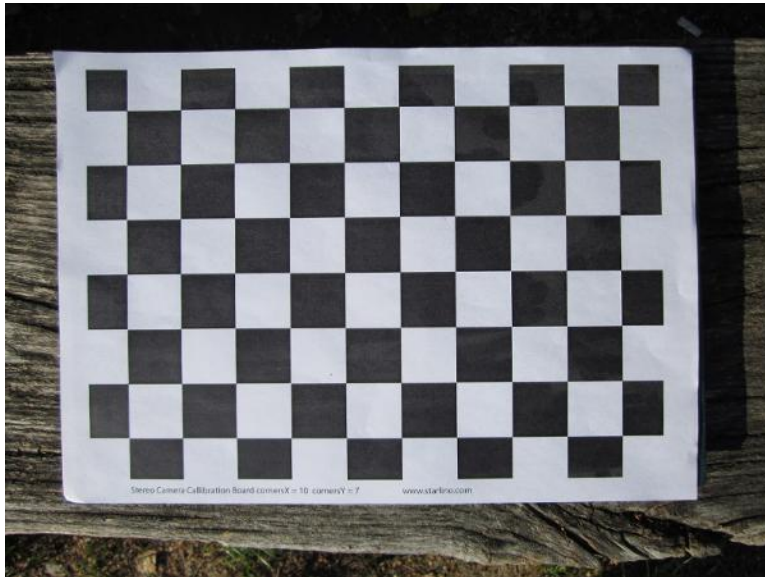


- ▶ In automatic diet monitoring, food amount estimation is a main objective.
- ▶ Food volume estimation is the most direct approach to automate the computation of calories or nutrients of food intake.
- ▶ Volume estimation from images can be obtained through different procedures, but up to a scale factor which must be determined to compute the exact volume.

# Size Reference vs. Calibration

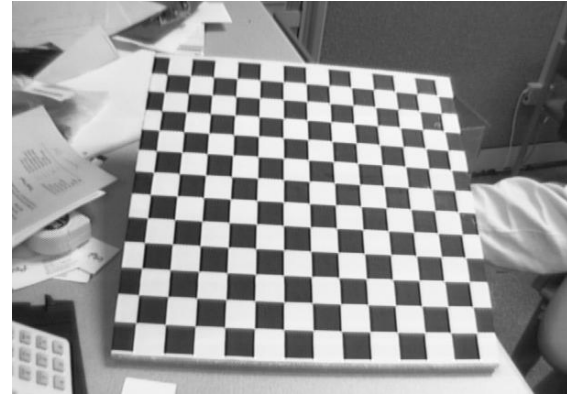
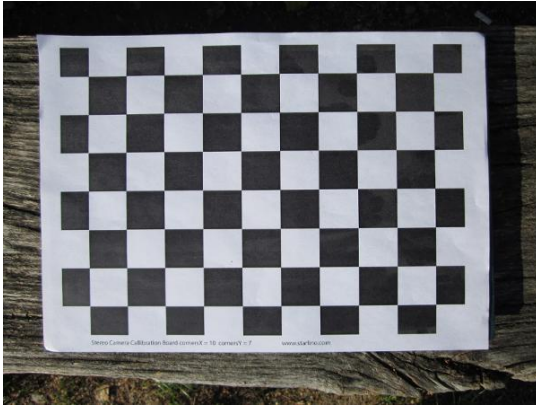
- ▶ Simplicity of the pattern and availability of effective detection algorithms, makes a checkerboard a proper candidate as size reference.
  - ▶ However, off-the-shelf checkerboard detection algorithms are usually designed to be means for camera calibration or pose-detection processes, which require that the checkerboards occupy most of the image.
- 

# Size Reference vs. Calibration

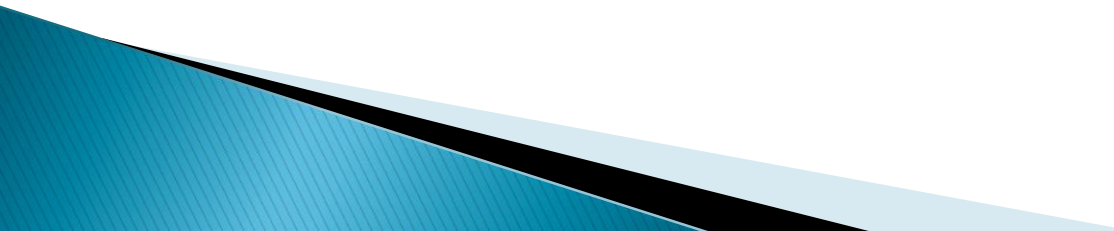




# Size Reference vs. Calibration

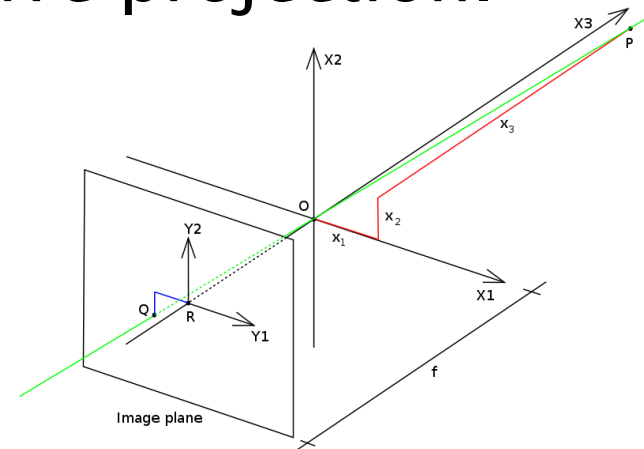
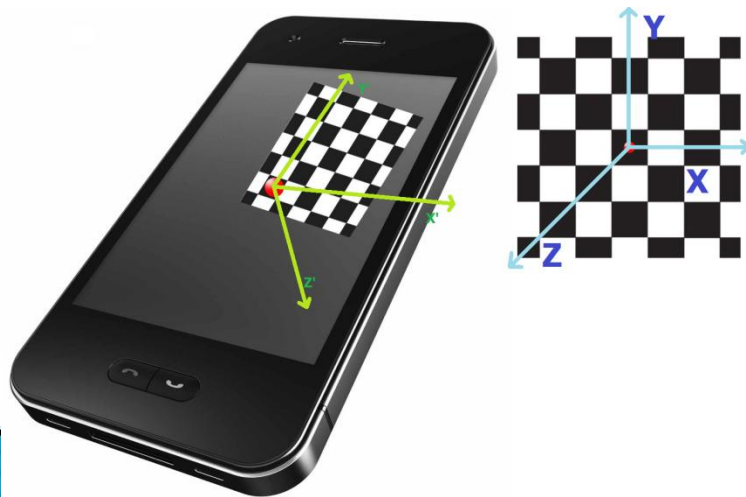


# Method

- ▶ Phase 1: Detect approximate location of the checkerboard.
  - ▶ Phase 2: Detect the exact position of the corners using a corner-detection algorithm applied only to the region where the pattern was detected.
- 

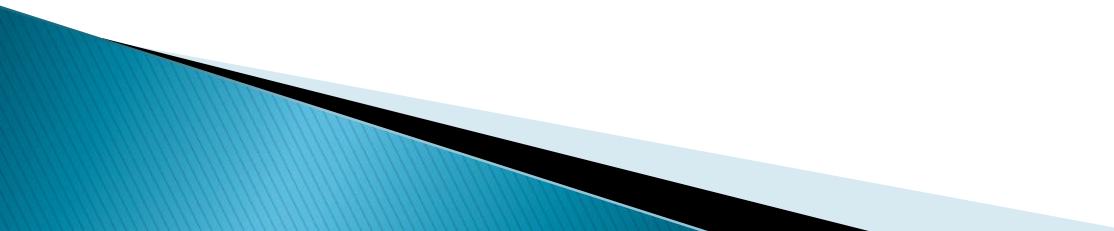
# Locating the Checkerboard

- ▶ In this work, a stochastic approach is used to find the object pattern in the image.
- ▶ To find the pattern, if the relative position of the camera and the checkerboard was known, we could determine the corresponding point on the image by perspective projection.



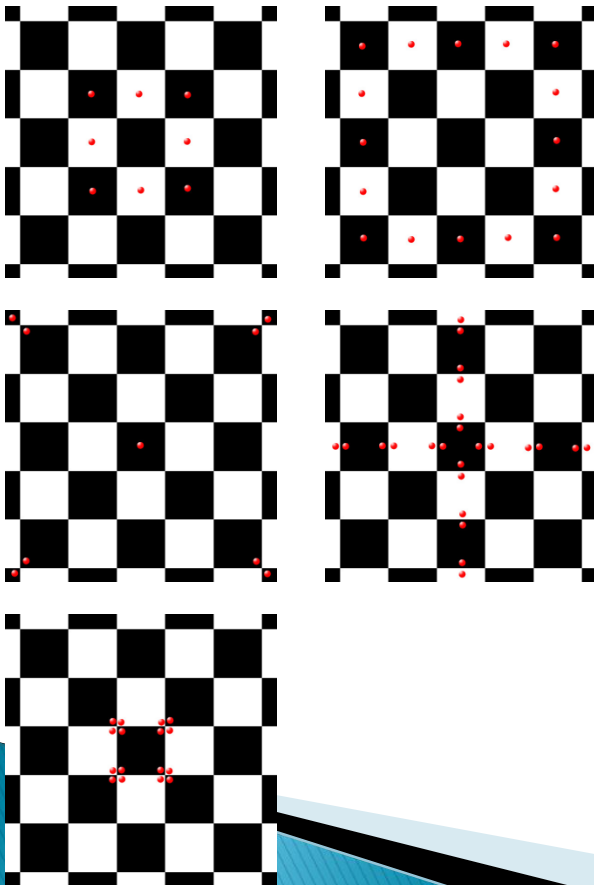


# Locating the Checkerboard

- ▶ Differential Evolution (DE) is employed to locate the checkerboard in the image.
  - ▶ Each member of the DE population corresponds to an estimation of the pose of a checkerboard model.
  - ▶ Along the generations better and better estimates are generated based on a similarity function (fitness function) between the re-projected model and the actual image.
- 

# Locating the Checkerboard

## Checkerboard model



---

### Algorithm 1 Fitness Function

---

**function** FITNESSFUNCTION(PoseVector)

    Calc Rotation & Translation matrices from PoseVector

$Score \leftarrow Score + FirstLevelCenterCheck()$

**if** Score > 6 **then**

$Score \leftarrow Score + SecondLevelCenterCheck()$

**end if**

**if** Score > 20 & the center is black **then**

$Score \leftarrow Score + PoseCheck()$

**end if**

**if** Score > 23 **then**

$Score \leftarrow Score + EdgesCheck()$

$Score \leftarrow Score + verticesCheck()$

**end if**

**return** Score

**end function**

---

# Locating the Checkerboard



# Corner Detection

- ▶ The image region where the checkerboard was detected in the first phase can be cropped.
- ▶ A customized algorithm was designed to detect the checkerboard corners on the cropped image and refine the checkerboard position estimation.

# Corner Detection

---

**function** FINDCHECKERBOARDCORNERS

*Image*  $\leftarrow$  *GetCroppedRGBImage*()

*Image*  $\leftarrow$  *RGB2SinglePrecision*(*Image*)

▷ Calculate second derivatives at zero and 45 degrees

*D0*  $\leftarrow$  *CalcSecondDerivative*(*Image*, 0)

*D45*  $\leftarrow$  *CalcSecondDerivative*(*Image*, 45)

▷ Find the pixels with higher values

*Corners0*  $\leftarrow$  *FindCorners*(*D0*)

*Corners45*  $\leftarrow$  *FindCorners*(*D45*)

▷ Find 3 closest corners to the image center

*CentralCorners0*  $\leftarrow$  *Find3CentralCorners*(*Corners0*)

*CentralCorners45*  $\leftarrow$  *Find3CentralCorners*(*Corners45*)

▷ Expand the candidate checkerboards based on the central corners

*Candidates0*  $\leftarrow$  *ExpandCentralCorners*(*CentralCorners0*)

*Candidates45*  $\leftarrow$  *ExpandCentralCorners*(*CentralCorners45*)

▷ Choose the best candidate

*BestCandidate*  $\leftarrow$  *ScoreCandidates*(*Candidates0*, *Candidates45*)

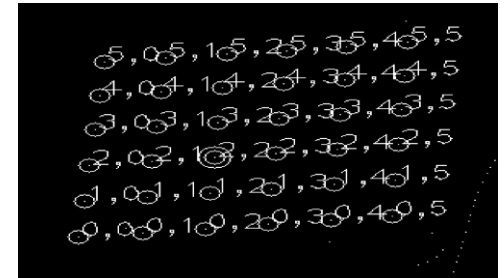
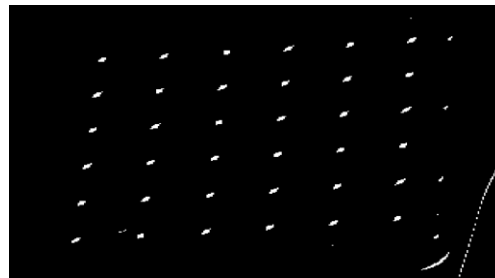
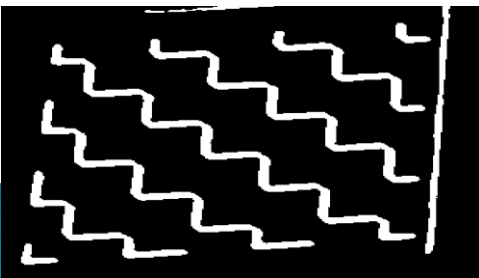
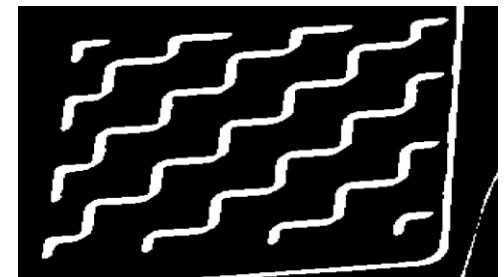
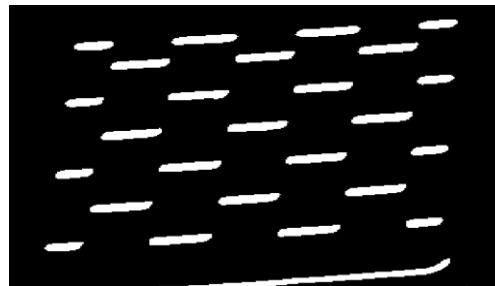
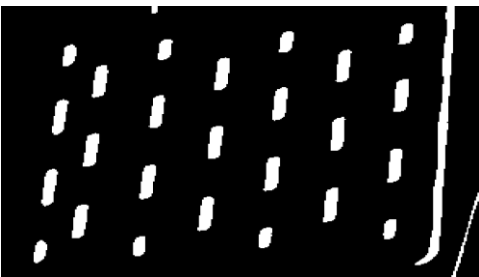
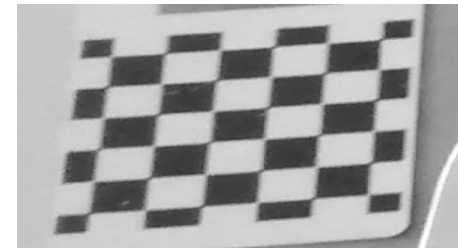
**return** *BestCandidate*

**end function**

---

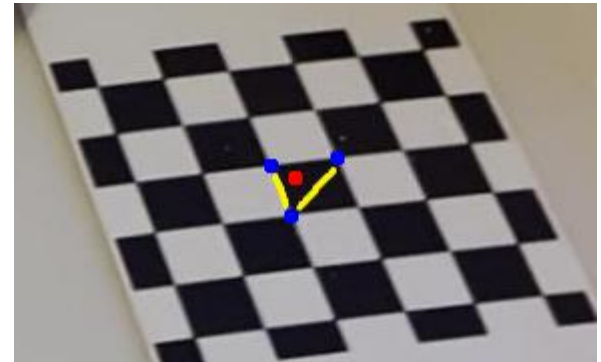
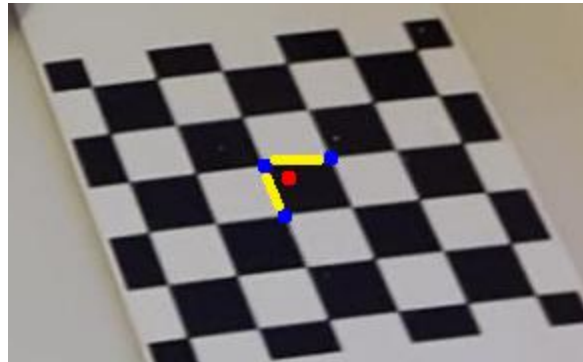
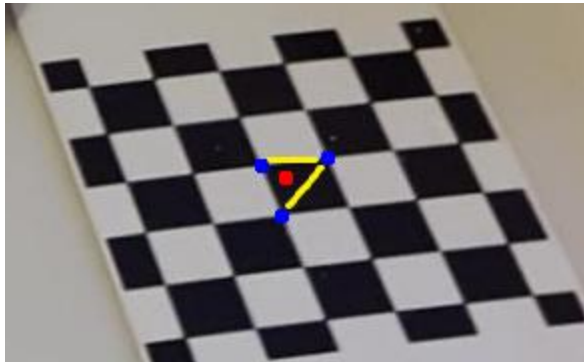


# Corner Detection

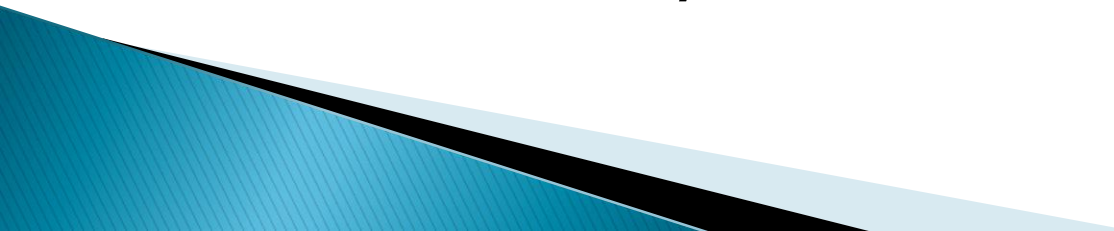


# Corner Detection

- ▶ After detection of the corners, their order should be exposed.
- ▶ First, we find the central square, and then we expand the checkerboard pattern.



# Results

- ▶ The algorithm was tested on four image sets, including 458 food images in total.
  - ▶ DE was iterated up to 1000 times for every image. Also, DE was allowed to run up to four times for each image if a satisfactory match had not been found.
  - ▶ After locating the checkerboard, corners were located by two basic algorithms (OpenCV and Matlab) and by our customized algorithm.
- 

# Results

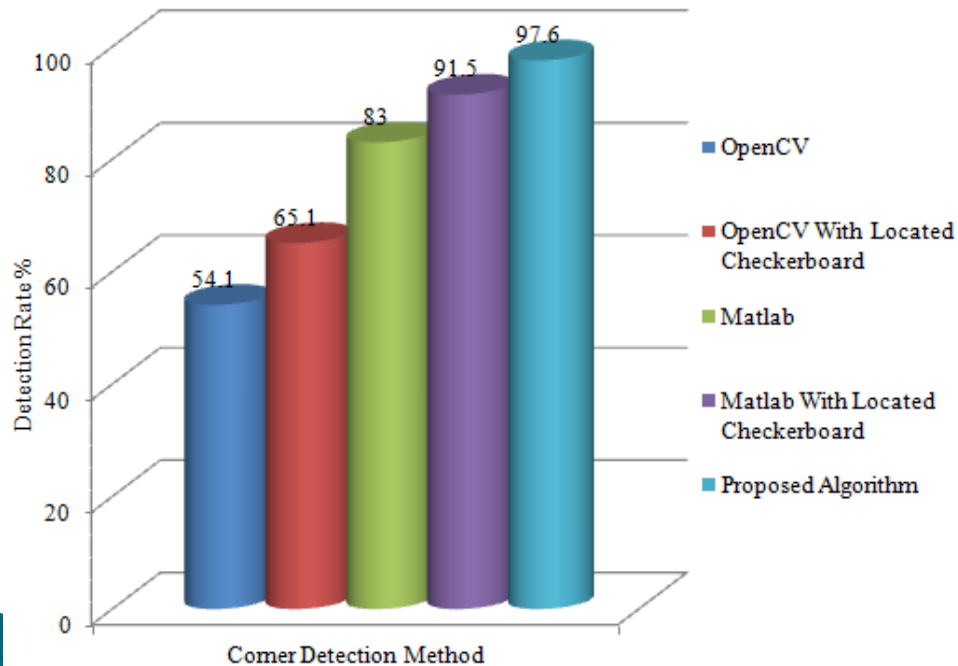
Results of the DE-based checkerboard locating algorithm.

	Images No.	Success	First try Success	DE tries
Motorola MotoG	179	176	143	1.34
Samsung Galaxy Note 1	19	19	11	1.3
Samsung Galaxy S3	130	129	123	1.2
Samsung Galaxy S3 scaled (0.5)	130	127	125	1.13
Total	458	451	402	1.24

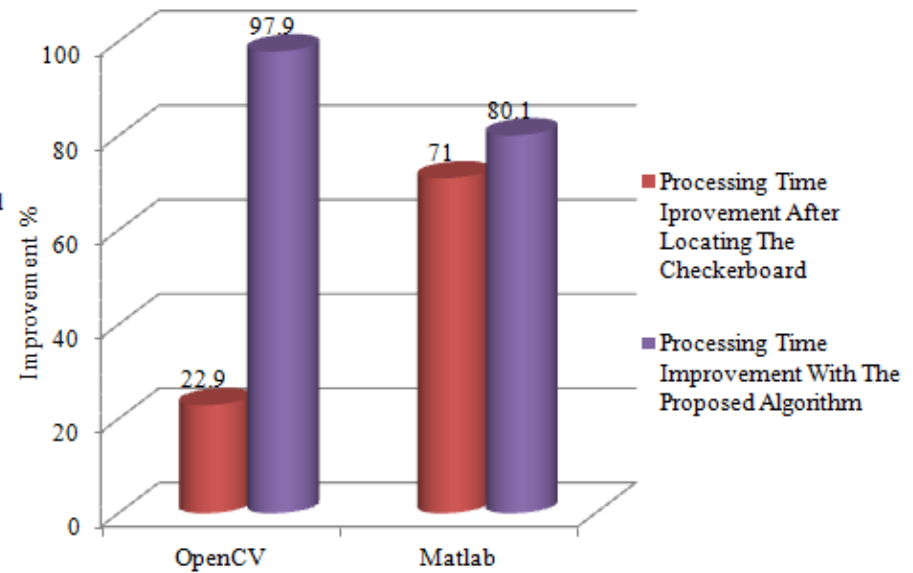
- ▶ In 98% of the cases the checkerboard was correctly located.

# Results

## Detection rate

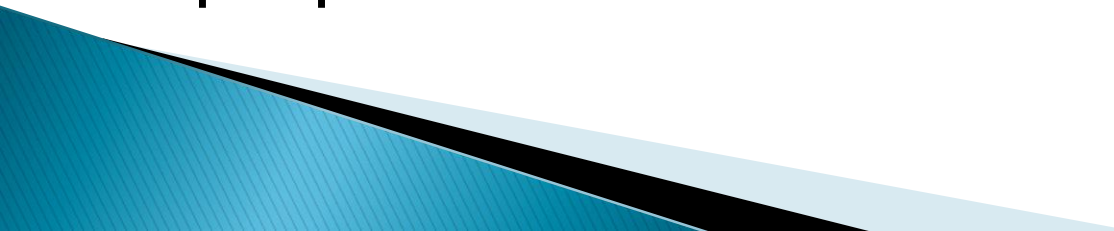


## Processing time





# Summary

- ▶ The pre-processing phase based on DE allows one to focus on the image region where the pattern is located.
  - ▶ This improves the performance of corner detection algorithms and, at the same time,
  - ▶ Reduces the execution time of such algorithms whose speed is usually inversely proportional to the difficulty of the task.
- 

# Future work

- ▶ Using asymmetric checkerboards: symmetric checkerboard as the one used in this work may cause inconsistent or ambiguous pose detections.
- ▶ Taking into consideration the intrinsic parallel nature of DE, parallelizing the algorithm on GPU using platforms like CUDA or OpenCL.

# Thank you

## Questions