KAUNAS UNIVERSITY OF TECHNOLOGY

INFORMATICS FACULTY

SOFTWARE ENGINEERING DEPARTMENT

Rasa Zavistanavičiūtė

OBJECT DETECTION ALGORITHMS ANALYSIS AND IMPLEMENTATION FOR AUGMENTED REALITY SYSTEM

Advisor: Dr. Andrej Ušaniov

KAUNAS, 2012

KAUNAS UNIVERSITY OF TECHNOLOGY

INFORMATICS FACULTY

SOFTWARE ENGINEERING DEPARTMENT

Rasa Zavistanavičiūtė

OBJECT DETECTION ALGORITHMS ANALYSIS AND IMPLEMENTATION FOR AUGMENTED REALITY SYSTEM

Reviewer: Dr. Sigitas Drąsutis

2012-05-28

Advisor: Dr. Andrej Ušaniov

2012-05-28

Author: Rasa Zavistanavičiūtė

IFM-0/2

2012-05-28

KAUNAS, 2012

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

SUMMARY

Object detection is the initial step in any image analysis procedure and is essential for the performance of object recognition and augmented reality systems. Research concerning the detection of edges and blobs is particularly rich and many algorithms or methods have been proposed in the literature. This master's thesis presents 4 most common blob and edge detectors, proposes method for detected numbers separation and describes the experimental setup and results of object detection and detected numbers separation performance. Finally, we determine which detector demonstrates the best results for mobile augmented reality system.

SANTRAUKA

Objektų aptikimas yra pagrindinis žingsnis vaizdų analizės procese ir yra pagrindinis veiksnys apibrėžiantis našumą objektų atpažinimo ir papildytosios realybės sistemose. Literatūroje gausu metodų ir algoritmų aprašančių sričių ir ribų aptikimą. Šiame magistro laipsnio darbe aprašomi 4 dažniausiai naudojami sričių ir ribų aptikimo algoritmai, pasiūlomas metodas aptiktų skaičių atskyrimo problemai išspręsti. Pateikiami atliktų eksperimentų rezultatai, palyginmas šių algoritmų našumas. Galiausiai yra nustatoma, kuris iš jų yra geriausias.

1. INTRODUCTION

Image processing and augmented reality on mobile phones is a new and exciting field with many challenges due to limited hardware, connectivity and processing power. Phones with HD cameras, powerful CPUs and memory storage devices are becoming increasingly common.

Research concerning image processing and detection of edges and blobs, therefore real world two-dimensional objects, is particularly rich and many algorithms have been proposed in the literature.

In this thesis we analyze 4 most common blob and edge detectors and suggest a method for solving detected numbers separation problem.

1.1. ORGANIZATION OF THE THESIS

This thesis is organized as follows. Chapter 2 gives a description of the augmented reality, object separation and object detection algorithms implemented in Augmented Reality System "Nonogram Solver". Chapter 3 presents Augmented Reality System "Nonogram Solver". This system was implemented as part of master's thesis. Chapter 4 describes detailed problems and goals of this thesis and suggested method for detected numbers separation. Chapter 5 presents experimentation results. Finally, in chapter 6, conclusions are drawn from our experiment.

2. SUBJECT ANALYSIS

2.1. DEFINITIONS

Augmented reality (AR) is a live, direct or indirect, view of a physical, real-world environment whose elements are augmented by computer-generated sensory input such as sound, video, graphics or GPS data.

Human–computer Interaction (**HCI**) involves the study, planning, and design of the interaction between people (users) and computers.

Nonogram is picture logic puzzle in which cells in a grid have to be colored or left blank according to numbers given at the side of the grid to reveal a hidden picture.

Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos.

Gradient is a vector that represents rate of change of intensity with change in distance.

2.2. AUGMENTED REALITY

Augmented reality (AR) is an innovative use of computer graphics in combination with real world data to create a new kind of video image. AR seamlessly integrates technology with the real world, allowing for a naturally enhanced computing experience. (1)

Figure 1 shows an informal model of an Augmented Reality system that combines real world and virtual world objects (or data). Real world data are accepted over sensors and data of the virtual world are queried from an information system. Real and virtual data are combined in human-computer interaction (HCI) devices. (2)

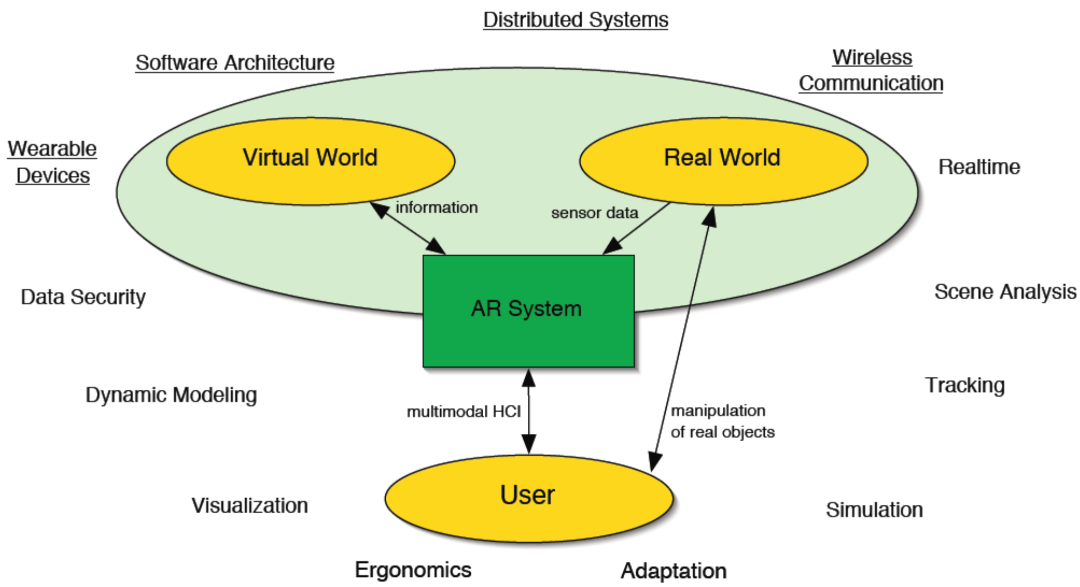


Figure 1 Informal model of Augmented Reality

The first steps towards AR were taken in the late 60's when Sutherland and his colleagues constructed the first see-through HMD (3) which mixed a view of the real world with computer generated images. Such displays were used during following decades in research on helmet-mounted displays in aircraft cockpits e.g. in the US Air Force's Super Cockpit program1 organized by Furness III, where fighter pilot's views were augmented.

2.3. AUGMENTED REALITY APPLICATIONS TYPES

Nowadays there are three major display techniques for Augmented Reality:

Head Mounted Displays (HMD) places images of both the physical world and registered virtual graphical objects over the user's view of the world. In Figure 2 HMD optical see-through glasses are displayed.

Figure 2 HMD optical see-through glasses

Spatial Displays. In *Spatially Augmented Reality (SAR),* the user's physical environment is augmented with images that are integrated directly into user's environment, not simply into their visual field. Object can be projected onto real objects using digital light projectors or embedded directly in the environment with flat panel displays. (4). In Figure 3 AR pool system is displayed. This system allows it's user to see ball's trajectories.

Figure 3 AR pool system by RCVLab at Queen's University

Handheld Displays. Handheld Augment Reality employs a small computing device with a display that fits in a user's hand. All handheld AR solutions to date have video see-through techniques to overlay the graphical information to the physical world.

Figure 4 Hand held device example

Head-mounted displays (HMD) ad Spatial Augmented Reality (SAR) systems are expensive, fragile and inconvenient to wear. That's why handheld display AR systems promises to be the first commercial success for AR technologies. (5)

2.4. WHAT IS A NONOGRAM?

Nonograms, also known as Paint by Numbers or Griddlers are logic puzzles in which cells in a grid have to be colored or left blank according to numbers given at the side of the grid to reveal a hidden picture. In this puzzle type, the numbers measure how many unbroken lines of filled-in squares there are in any given row or column (6).

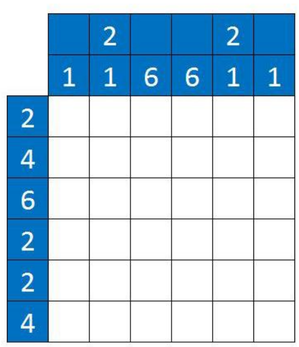


Figure 5 Simple nonogram

In Figure 5 a simple nonogram is given. Take for example the second column of nonogram puzzle in Figure 5. Here 2,1 means that in this column block of two cells and block of one cell needs to be painted with at least one white space between. Unfortunately there are six positions available; these permutations are also shown in Figure 6, it is not clear which cells needs to be painted. The third line has number six to it, so this line can be entirely painted.

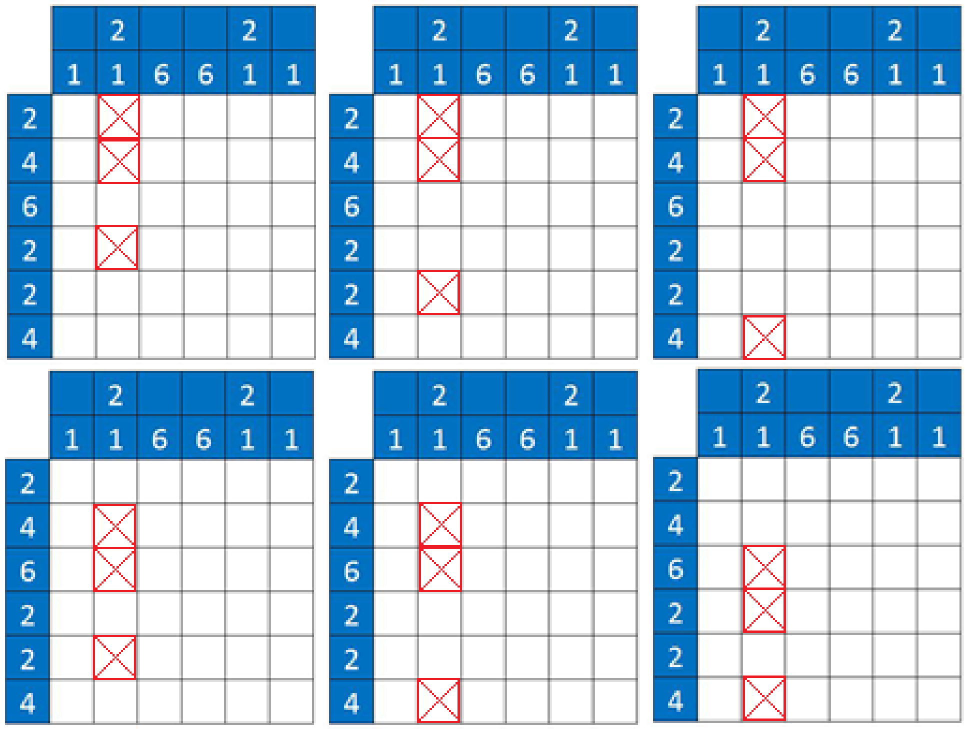


Figure 6 Possibilities to solve the second nonogram column

As gradually more cells are being filled more information becomes available which cells must be filled and which must be left empty. When all the rows and columns are painted according to their numbers the puzzle is solved (Figure 7).

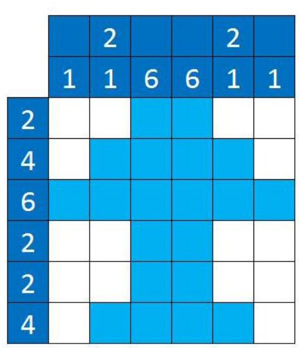


Figure 7 Solved nonogram

2.5. OBJECT DETECTION ALGORITHMS

2.5.1. BLOB DETECTION

Before going into detail on blob detection, first let describe what a blob is. Lindeberg (7) defines a blob as being region associated with at least one local extremum, either a maximum or a minimum for a bright or dark blob. Regarding the image intensity, the extent of a blob is limited by a saddle point where the intensity stops decreasing and starts increasing for bright blobs and vice versa for bright blobs.

A blob is represented as a pair consisting of one extremum point and one saddle point. Hinz (8) just describes blob as a rectangle with a homogenous area, i. e. a constant contrast, which becomes a local extremum under sufficient amount of scaling. Rosenfield (9) defines a blob as a crossing of lines perpendicular to edge tangent directions, surrounded by 6 or more directions like in a Figure 8.

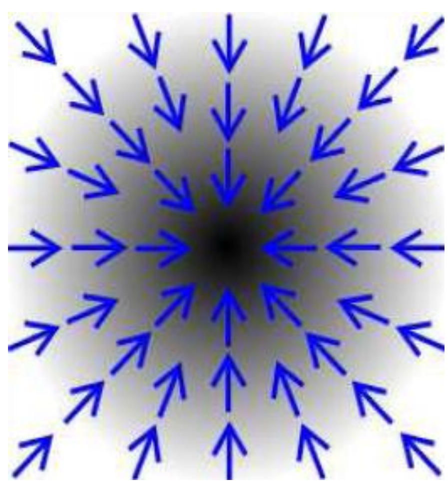
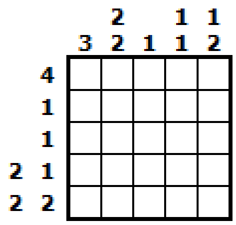


Figure 8 A blob surrounded by different directions

For this master's thesis 2 blob detection algorithms were selected: simple blob detection and blob detection using labeling. These algorithms are describes in the following 2 subsections.

2.5.2. SIMPLE BLOB DETECTION

The goal of this algorithm is to be able to identify blobs in black and white images, such as in Figure 9.



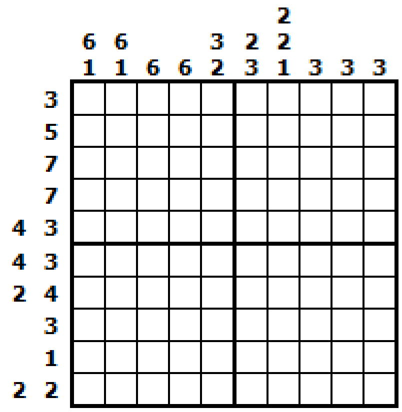


Figure 9 Black and white picture examples

In this algorithm blob is defines as a collection of pixels that share any of the 8 possible common borders. For example, take pixel A in Figure 10, out of 8 possible neighbor pixels it has 6 (marked as SP).

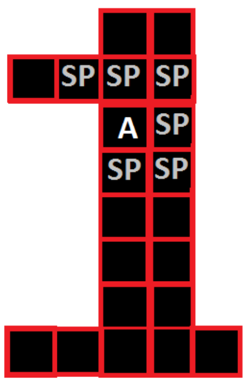


Figure 10 Pixelated image for blob detection

Using this main rule, algorithm counts the number of pixels in the blob. Simple blob detection example is shown in .

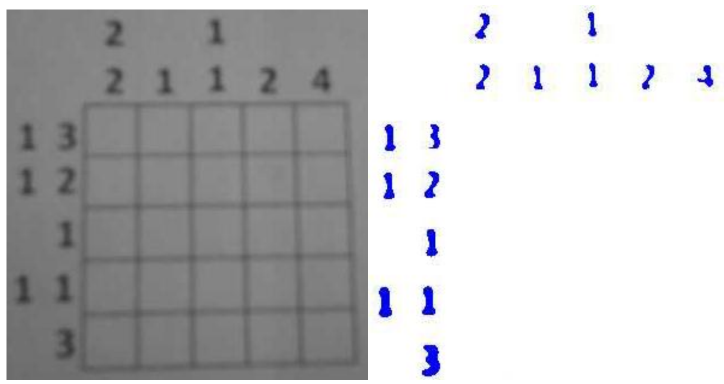


Figure 11 Simple blob detection example

2.5.3. BLOB DETECTION USING LABELING

Goal of this algorithm is to label each pixel within a blob with the same label number. The first stage in achieving this is to iterate through all pixels, checking the label number of neighboring pixels as shown in Figure 12.

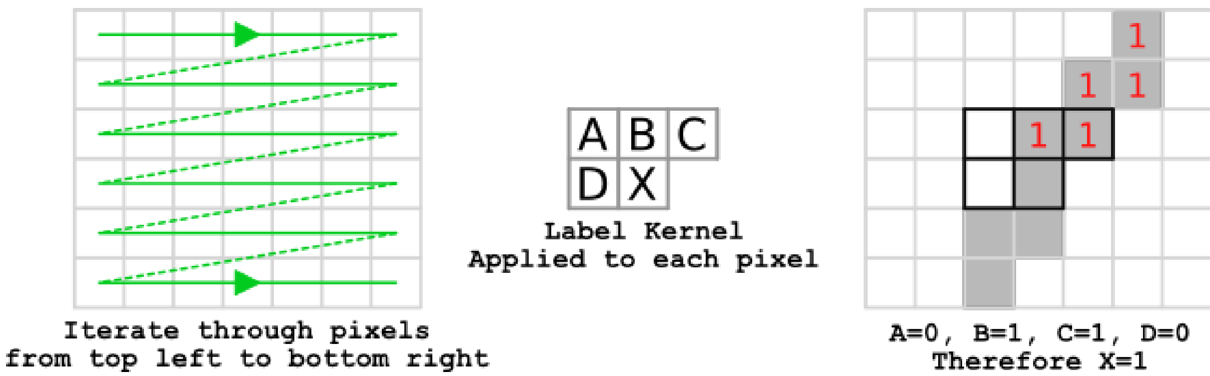


Figure 12 Pixel labeling process

Blob detection using labeling example is shown in Figure 13.

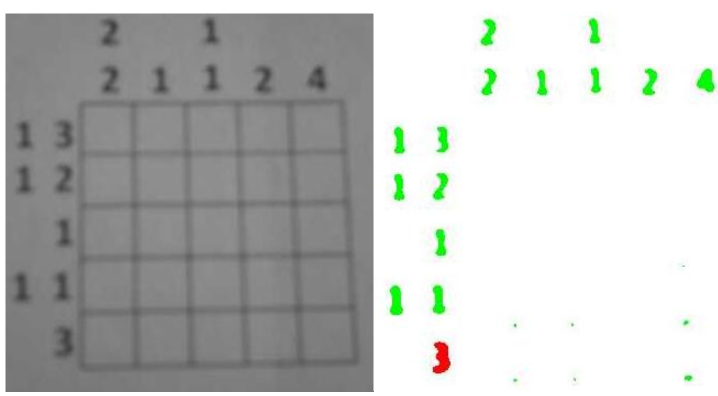


Figure 13 Blob detection using labeling example

2.5.4. EDGE DETECTION

Edges define the boundaries between regions in an image, which helps with segmentation and object recognition. They can show where shadows fall in an image or any other distinct change in the intensity of an image. Edge detection is a fundamental of low-level image processing and good edges are necessary for higher level processing. (10)

Edges in images are areas with strong intensity contrasts a jump in intensity from one pixel to the next. Edge detection of an image reduces the amount of data and discards useless information, while preserving the important structural properties in an image (like outline, shape). (11)

In the following chapters 2 common edge detector will be presented.

2.5.5. CANNY EDGE DETECTOR

Canny edge detection algorithm was developed by John F. Canny in 1986 (12). Even though it is quite old, it has become one of the standard edge detection methods and is still used in research (13), (14), (15), (16).

Canny (12) defines optimal edge finding as a set of criteria that maximize the probability of detecting true edges while minimizing the probability of false edges.

The algorithm runs in 5 separate steps:

1. Smoothing
2. Findings gradients
3. Non-maximum suppression
4. Double thresholding
5. Edge tracking

Each step is described in the flowing sections.

Smoothing

All images taken from camera contains some king of noise. To prevent that noise is mistaken for edges, noise must be reduced.

To do that image is smoothed by applying a Gaussian filter. The standard Gaussian filter is shown in Figure 14.

$$B = \frac{1}{159} \cdot \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

Figure 14 Gaussian filter for smoothing

Finding gradients

The Canny algorithm finds edges where gray scale intensity of the image changes the most. To find these areas, gradients of the images are determined. Gradients at each pixel are determined by applying Sobel operator.

Non-maximum suppression

This step converts the blurred edges in the images to sharp edges. Tod o that algorithm keeps only those pixels on an edge with highest gradient magnitude. The maximum magnitudes should occur right on the edge boundary.

Double thresholding

After step 3 the remaining edge pixels are marked with their strength pixel-by-pixel. Many of these will probably be true edges in the image, but there still remain pixels caused by noise or color variations.

The Canny edge detection algorithm uses double thresholding to remove “bad” edge pixels. Edge pixels stringer than the high value of threshold are marked as string; edge pixels weaker than low threshold are suppressed and pixels between two thresholds are marked as weak.

Edge tracking

Strong edges are interpreted as certain edges and can be used as certain object edges in following actions. Canny edge detection example is shown in Figure 15.

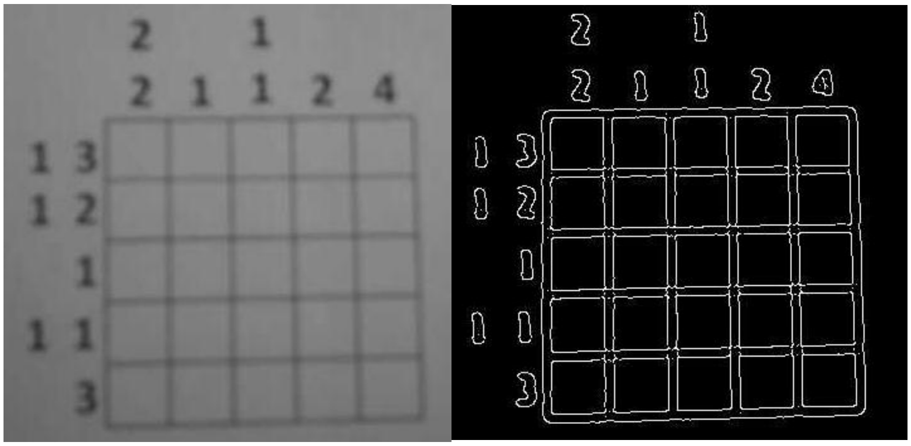


Figure 15 Canny algorithm example

2.5.6. SOBEL EDGE DETECTOR

The Sobel operator performs a two dimensional spatial gradient measurement on an image (11). It uses a pair of 3x3 matrices, one measures the gradient in the x-direction (columns) and the other measures the gradient in the y-direction (rows).

A convolution mask is usually much smaller than the actual image. As a result, the mask is slid over the image, manipulating a square of pixels at a time. The actual Sobel masks are shown in Figure 16.

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

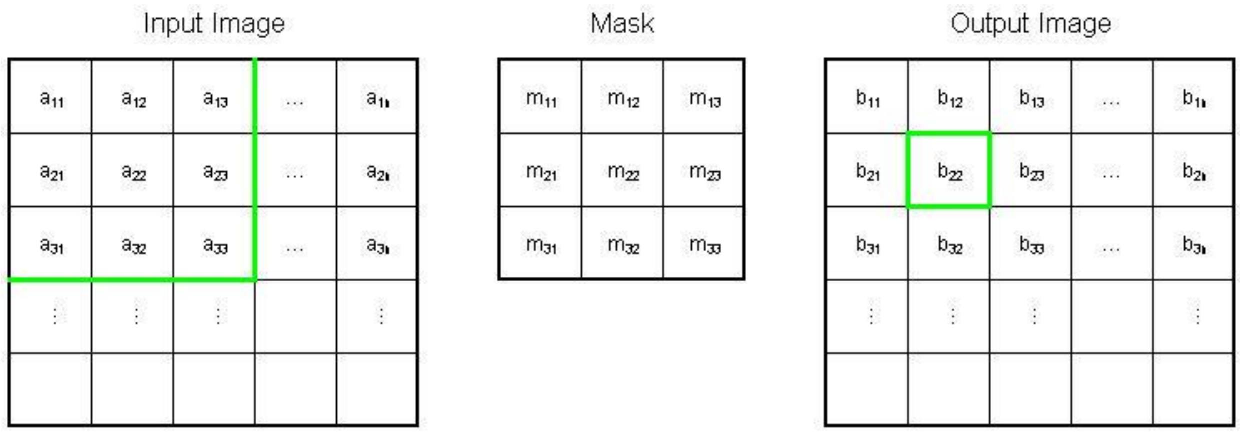
Gy

Figure 16 Sobel masks

The mask is slid over an area of the input image, changes that pixel's value ant then shifts one pixel to the right and continues to the right until it reaches the end row. It then starts at the beginning of the next row.

The example in Figure 17 shows the mask being slid over the top left portion of the input image represented by green outline. The formula shows how a particular pixel in the output image would be calculated.

The center of the mask is places over the pixel you are manipulating in the image.



$$b_{22} = (a_{11}*m_{11}) + (a_{12}*m_{12}) + (a_{13}*m_{13}) + (a_{21}*m_{21}) + (a_{21}*m_{21}) + (a22*m_{22}) + (a_{23}*m_{23}) + (a_{31}*m_{31}) + (a_{32}*m_{32}) + (a_{33}*m_{33})$$

Figure 17 Image manipulations with Sobel mask

The Gx mask highlights the edges in the horizontal direction while Gy mask highlights the edges in the vertical direction. After taking the magnitude of both, the resulting output detects edges in both directions.Sobel edge detection example is shown in Figure 18.

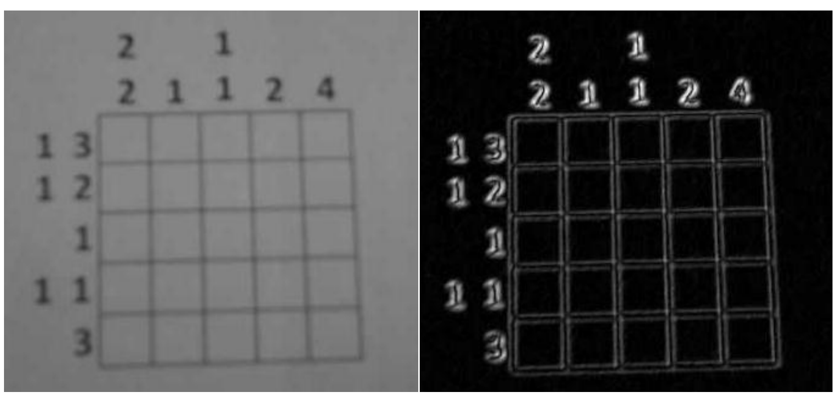


Figure 18 Sobel algorithm example

2.6. DETECTED OBJECTS SEPARATION AND SEGMENTATION

Many approaches to image segmentation and separation have been proposed over the years. (17) (18) (19) (20). Of these various methods, k-means clustering is one of the simplest, and has been widely used in segmentation of grey level images.

The k-means algorithm aims to minimize the sum of squared distances between all points and the cluster center. This procedure consists of the following steps, as described by Tou and Gonzalez (21).

1. Choose K initial cluster centers $z_1(1), z_2(1), \ldots, z_k(1)$.
2. At the k-th iterative step, distributed the samples {x} among the K clusters using the relation in Figure 19.

$$x \in C_j(k) \text{ if } \|x - z_j(k)\| < \|x - z_i(k)\|$$

for all $i = 1, 2, \ldots, K$; $i \neq j$; where $C_j(k)$ denotes the set of samples whose cluster centre is $z_j(k)$.

Figure 19 Formula for samples distribution among the clusters

3. Compute the new cluster centers $z_j(k+1), j= 1,2,\ldots,K$ such that the sum of the squared distances from all points in $C_j(k)$ to the new cluster center is minimized. The measure which minimizes this is simply the sample mean of $C_j(k)$. Therefore, the new cluster center is given by formula in Figure 20, where N_j is the number of samples in $C_j(k)$.

$$z_j(k+1) = \frac{1}{N_j} \sum_{x \in C_j(k)} x, \quad j = 1,2,\ldots,K$$

Figure 20 Formula for new cluster centers

4. If $z_j(k+1) = z_j(k)$ *for* $j = 1,2,\ldots,K$ then the algorithm converged and the procedure is terminated. Otherwise go to step 2.

It is obvious in this description that the final clustering will depend on the initial cluster centers chosen and on the value of K. the latter is of the most concern since this requires some prior knowledge of the number of cluster present in the data.

In we describe approach for calculating number of cluster present in the data of detected nonograms number objects. Then k-means algorithm is user for number separation.

3. PROJECT

In this chapter we will describe Augmented Reality System: Nonogram Solver. This system was implemented as part of master's thesis.

3.1. SYSTEM

3.1.1. SYSTEM DESCRIPTION

Nonogram Solver is an augmented reality system, which uses augmented reality to provide nonogram solution. All user has to do is point mobile device at nonogram, take picture and solution is displayed.

Nonogram solver can detect nonogram objects (numerals, grid), solve nonogram and show augmented solution for user (Figure 21).

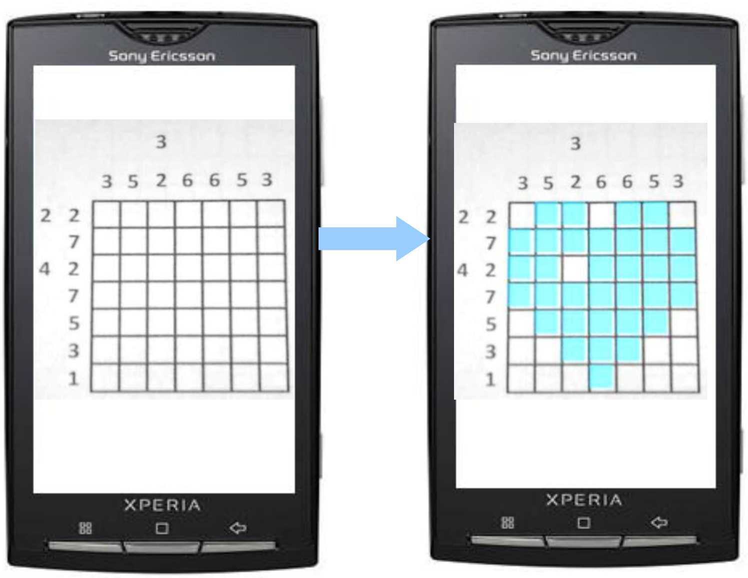


Figure 21 Principle of nonogram solver

3.1.2. THE PURPOSE OF THE SYSTEM

Mobile phones, which were not long ago "brick-like" devices limited to phone calls, have evolved into smart phones, with increased storage, communication and computational resources. After analyzing worldwide smart phones market, we concluded that smart phones with Android OS become more and more popular.

According to Gartner analysis (22) in Figure 1 Android OS made an impressive 1200% gain in devices shipped, and shot them all the way from 3.5% to 25.5% market share.

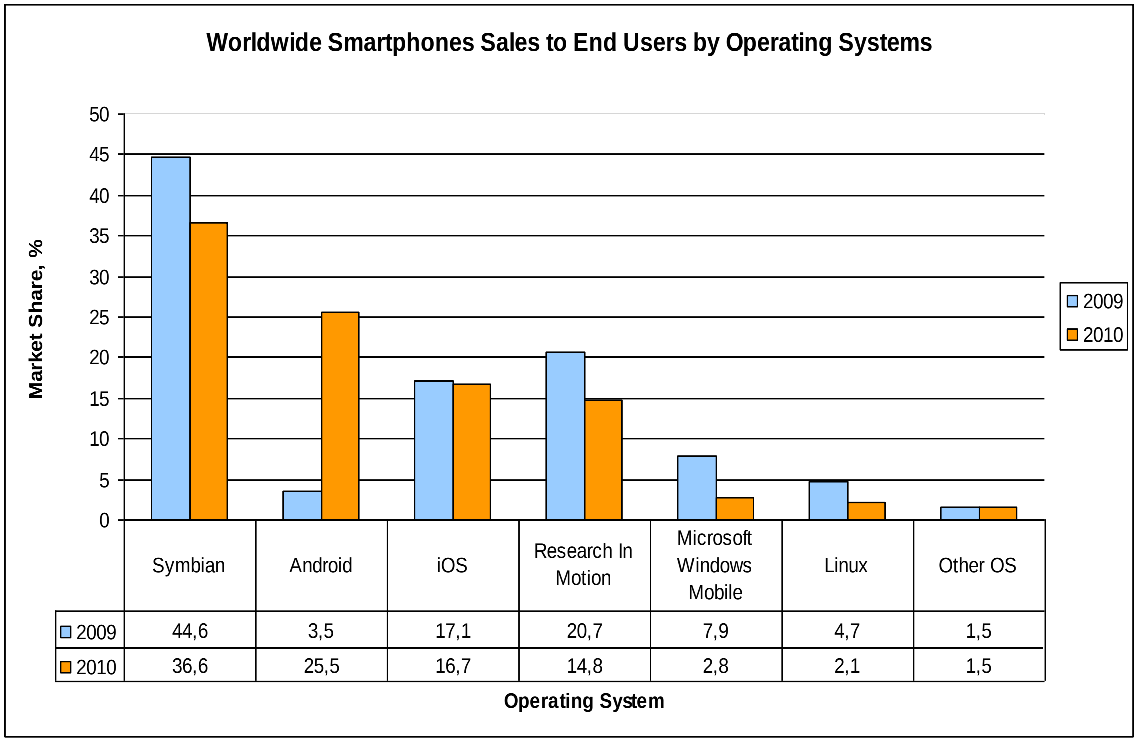


Figure 22 Worldwide smart phones sales in 2009-2010

In 2011 according to Millennial Android operating system reached 50 % of market share of smartphones operating systems (Figure 23).

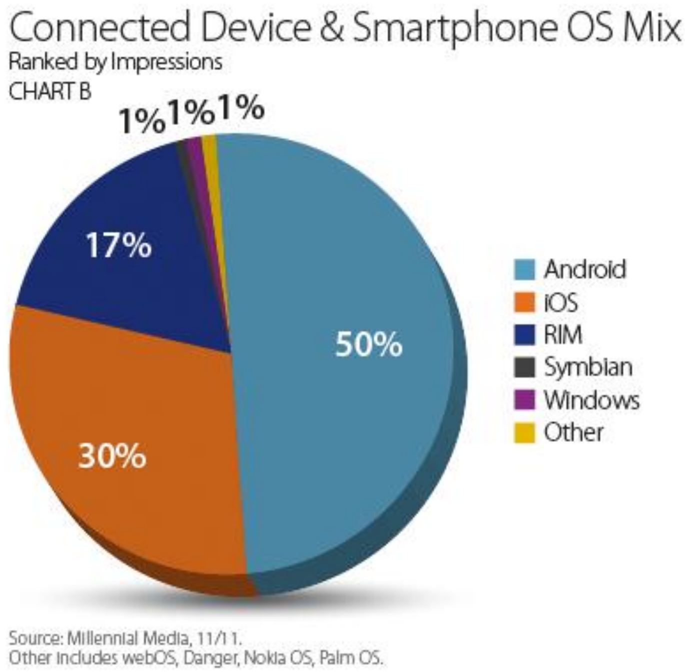


Figure 23 Market share of smartphones operating systems

Because of such rapid mobile phones evolution, augmented reality finds its way into smart phones market, transforming it into interactive personal digital assistants (PDA). Furthermore 14 out of 20 the most popular smartphones run on Android operating system (Figure 24).

Top 20 Mobile Phones
Ranked by Impressions
CHART B

Rank	Devices	November 2011	Type	OS
1	Apple iPhone	13.54%	Smartphone	iOS
2	BlackBerry Curve	5.87%	Smartphone	BlackBerry OS
3	Motorola Droid X	5.27%	Smartphone	Android
4	HTC Evo	3.18%	Smartphone	Android
5	LG Optimus	3.08%	Smartphone	Android
6	BlackBerry Bold	3.03%	Smartphone	BlackBerry OS
7	HTC Desire	2.99%	Smartphone	Android
8	BlackBerry Torch	2.77%	Smartphone	BlackBerry OS
9	Samsung Nexus S	2.48%	Smartphone	Android
10	Samsung Vibrant Galaxy S	2.36%	Smartphone	Android
11	Samsung Galaxy S	1.66%	Smartphone	Android
12	HTC Droid Incredible	1.53%	Smartphone	Android
13	BlackBerry Pearl	1.25%	Smartphone	BlackBerry OS
14	ZTE Score	1.16%	Smartphone	Android
15	Motorola Droid	1.06%	Smartphone	Android
16	Samsung Fascinate	1.03%	Smartphone	Android
17	HUAWEI Ascend	1.02%	Smartphone	Android
18	HUAWEI Ideos	0.87%	Smartphone	Android
19	HTC MyTouch 4G Glacier	0.84%	Smartphone	Android
20	BlackBerry Bold Touch	0.82%	Smartphone	BlackBerry OS

Source: Millennial Media, 11/11.

millennial media's mobilemix™
THE MOBILE DEVICE INDEX

Figure 24 The most popular smartphones

That is why "Nonogram solver" was created for smartphones with Android OS.

3.1.3. SIMILAR PRODUCTS

Layar - first AR system designed for Android OS. It displays real time digital information on top of reality in the camera screen of the mobile phone. While looking through the phone's camera lens, a user can see houses for sale, popular bars and shops, tourist information of the area, play a live game, etc.

Figure 25 Layar demonstration

WikiTude Drive - first augmented reality turn-by-turn navigation application for Android smart phones. Application uses phone’s camera and GPS receiver in tandem, layering selected route over a live view of what’s ahead of the car.

Figure 26 WikiTude Drive demonstration

TagWhat - basically a social networking application that uses augmented reality. It lets users to tag whatever they see in front of them using tag feature. People visiting those tagged places will see the details while pointing android phone to places in front.

Figure 27 TagWhat demonstration

Space InvadAR – a vision based game that uses AR. While pointing camera towards a high resolution image, application loads the game.

Figure 28 Space InvadAR demonstration

Sudoku grab – Sudoku puzzles solver. User just needs to point phone's camera to Sudoku puzzle and it gets solved. Solution is overlaid on real world Sudoku puzzle.

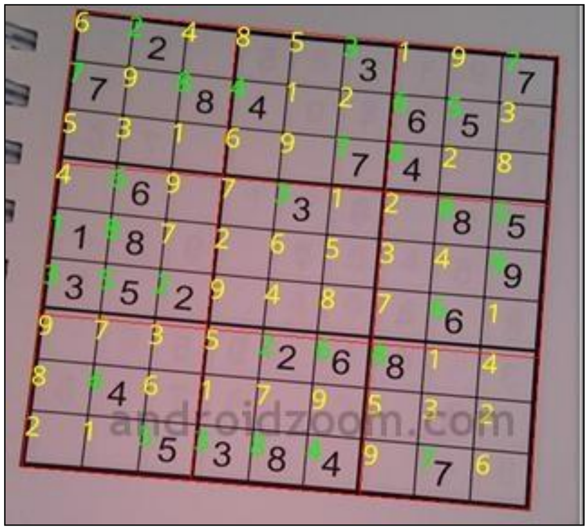


Figure 29 Sudoku Grab example

Comparison of augmented reality application mentioned above.

	Invadar	**TagWhat**	**Wikitude Drive**	**Layar**	**Sudoku Grab**
Supported phones	HTC Desire, Nexus 1	Most of mobile devices with Android OS	HTC Motorola Droid Samsung galaxy, Huawei RBM2, Nexus 1	Most of mobile devices with Android OS	Most of mobile devices with Android OS
Supported OS versions	Éclair, Froyo	Donut or higher	Éclair	Éclair, Froyo	Éclair, Froyo
Price	25 $	Free	Free	Free + paid layers	1 $
Size	3 MB	0.6 MB	2 MB	2 MB	1 MB
Release date	2010 08 08	2010 09 13	2009 10 28	2009 10 29	2010 10 15

3.2. IMPLEMENTATION

3.2.1. USE CASES

This chapter provides Nonogram Solver use cases' description.

System use cases are displayed in use case diagram (Figure 30)

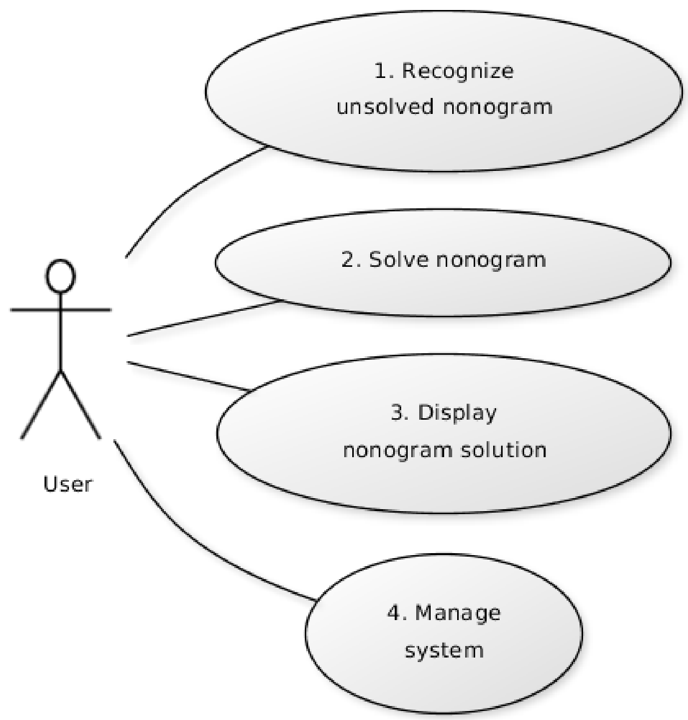


Figure 30. System's use case diagram

1. USE CASE: Recognize unsolved nonogram **User/Actor name:** User **Description:** Recognizes supplied nonogram and converts it to format recognized by the system. In this case matrix will be used. **Conditions before:** No nonogram was supplied as input data. **Invoke conditions:** User starts nonogram solving by pressing 'START' button. System begins nonogram search.

Conditions after: Nonogram solving can be initiated.

2. USE CASE: Solve nonogram

User/Actor name: User

Description: Solves supplied nonogram with one of nonogram solution algorithms.

Conditions before: User starts nonogram solving by pressing 'START' button.

Invoke conditions: Nonogram was recognized in video feed or picture.

Conditions after: Solution can be displayed for user.

3. USE CASE: Display nonogram solution

User/Actor name: User

Description: Augments video feed or picture with nonogram solution.

Conditions before: Nonogram was solved.

Invoke conditions: Nonogram solution was prepared for displaying.

Conditions after: Results can be saved.

4. USE CASE: Manage system

User/Actor name: User

Description: User can view help, solving logs. When nonogram solving is in progress user can terminate it.

Conditions before: None.

Invoke conditions: User selects one of following actions: view help, show logs, terminate solving.

Conditions after: None.

3.2.2. FUNCTIONS

Basic functional requirements list for Nonogram Solver:

1. System should recognize nonogram grids with different dimensions.
2. System should recognize numerals 1, 2, 3, 4, 5, 6, 7, 8 and 9.
3. System should display ■ (Square) symbol for filled squares.
4. System should notify user if there is no objects to recognize.
5. System should recognize objects from pictures.
6. System should allow solving process termination.

In the following tables each requirement is specified.

Table 1 Functional requirement no. 1 detailed description

Requirement number: **1**	Requirement type: **1**	Use case number: **1**
Description: **System should recognize nonograms with different dimensions**		
Rationale: **To be able solve different level nonograms.**		
Source: **Andrej Ušaniov**		
Fit criterion: **Different dimensions of nonograms will be recognized.**		
Customer satisfaction: **5**	Customer dissatisfaction: **2**	
Dependencies: **All requirements using nonograms structure and dimensions**		Conflicts: **None**
Supporting materials: **None**		
History: **Created March 10th, 2011**		

Table 2 Functional requirement no. 2 detailed description

Requirement number: **2**	Requirement type: **1**	Use case number: **1**
Description: **System should recognize numerals 1, 2, 3, 4, 5, 6, 7, 8 and 9**		
Rationale: **Nonogram puzzle consist of 1, 2, 3, 4, 5, 6, 7, 8, 9 numerals. So they should be recognized by the system.**		
Source: **Andrej Ušaniov**		
Fit criterion: **1, 2, 3, 4, 5, 6, 7, 8, 9 numerals will be recognized and used by the system. Numbers will be recognized from video or photo.**		
Customer satisfaction: **3**	Customer dissatisfaction: **2**	
Dependencies: **All requirements where recognized nonogram data is used.**		Conflicts: **None**
Supporting materials: **None**		
History: **Created March 10th, 2011**		

Table 3 Functional requirement no. 3 detailed description

Requirement number: **3**	Requirement type: **1**	Use case number: **3**
Description: **System should display ■ (Black Square) symbol for filled squares**		
Rationale: **Nonogram solution should be visible to the user.**		
Source: **Andrej Ušaniov**		
Fit criterion: **Nonogram solution will be displayed as ■ (Black Square) symbols.**		
Customer satisfaction: **5**	Customer dissatisfaction: **1**	
Dependencies: **None**		Conflicts: **None**
Supporting materials: **None**		
History: **Created March 10th, 2011**		

Table 4 Functional requirement no. 4 detailed description

Requirement number: **4**	Requirement type: **1**	Use case number: **1, 2**
Description: **System should notify user if there is no objects to recognize**		
Rationale: **User should be informed in case there is no data feed or data feed is faulty**		
Source: **Andrej Ušaniov**		
Fit criterion: **System will notify user if there is no objects to recognize by displaying warning window.**		
Customer satisfaction: **4**	Customer dissatisfaction: **2**	
Dependencies: **All requirements using input data (nonograms)**		Conflicts: **None**
Supporting materials: **None**		
History: **Created March 10th, 2011**		

Table 5 Functional requirement no. 5 detailed description

Requirement number: **5**	Requirement type: **1**	Use case number: **1**
Description: **System should recognize objects from pictures**		
Rationale: **There may be a need to use earlier saved picture of nonogram for solving**		
Source: **Andrej Ušaniov**		
Fit criterion: **System will recognize objects from earlier saved pictures of nonograms. Nonogram will be saved in text file.**		
Customer satisfaction: **4**	Customer dissatisfaction: **2**	
Dependencies: **All requirements using input data (nonograms)**		Conflicts: **None**
Supporting materials: **None**		
History: **Created March 10th, 2011**		

Table 6 Functional requirement no. 6 detailed description

Requirement number: **6**	Requirement type: **1**	Use case number: **4**
Description: **System should allow solving process termination**		
Rationale: **If nonogram solving takes too much time (or other reasons), there should be an opportunity to terminate it.**		
Source: **Andrej Ušaniov**		
Fit criterion: **System will allow nonogram solving termination by pressing <Cancel> button.**		
Customer satisfaction: **4**	Customer dissatisfaction: **2**	
Dependencies: **None**		Conflicts: **None**
Supporting materials: **None**		
History: **Created March 10th, 2011**		

3.2.3. ARCHITECTURE

Nonogram solver has 4 basic packages (Figure 31).

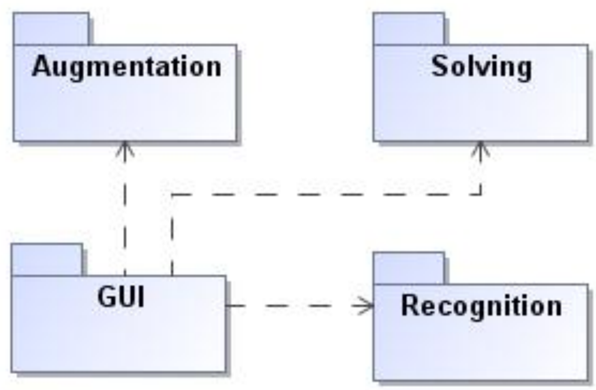


Figure 31. Packages of Nonogram solver

Recognition

Package is used for images recognition. Nonogram is split into different parts then these parts are converted to useful data for nonogram solving and augmentation.

Solving

Package is responsible for nonogram solving

Augmentation

Package is responsible for picture augmentation

GUI

Package is responsible for all graphic windows used in the system, such as Main Window, Help, and Settings. Also it saves results and statistics information of the application

4. RESEARCH

During master‘s study course augmented reality system „Nonogram Solver“ was implemented. Initially the system had 1 object detection algorithm: label blob detection. To increase system‘s performance other object detection algorithms were implemented: simple blob detection, Canny edge detection and Sobel edge detection.

In this section we will describe how 4 selected object detection algorithms were compared and how detected numbers segmentation problem was solved.

4.1. PROBLEMS

Various object detection algorithms exist, in this master’s thesis we have chosen 4 of them for analysis and implementation.

After we detect objects in nonograms image, it is crucial to know the order and logical meaning of these objects. For example, we have to know whether we detected numbers with one numeral or with two numerals. After these numeric objects were detected we have to separate them from original picture and pass to number recognition module. However, after the numbers were detected, we have only coordinates of separate objects without any logical order.

With numbers the main problem exists in determining if 2 or more detected objects that are relatively close to each other belongs to the same number or are separate numerals. For example in Figure 32 we have detected numbers, but we do not know if selected numbers (marked by red circles) are one or separate.

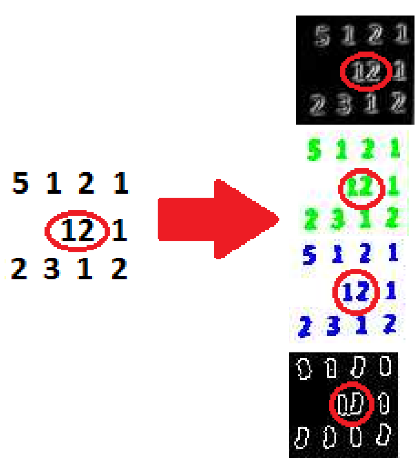


Figure 32 Example of detected numerals

4.2. GOALS

Goals for object detection algorithms analysis and implementation:

- Try implemented object detection algorithms for nonograms with different dimensions;
- Measure object detection time for implemented object detection algorithms;
- Compare object detection time;
- Try suggested method for detected numbers separation for nonograms with different dimensions;
- Measure numbers separation method's convergence and accuracy for implemented object detection algorithms;

4.3. IMPLEMENTATION

We have taken the existing object detection algorithms' implementations and adapted it for Android operating system. These implementations later were used for measuring object detection time and numbers separation method's convergence.

In chapter 4.1 we presented the problem of detected numbers separation. To solve it k-means clustering algorithm (chapter 2.6) was used.

The main disadvantage of the k-means algorithm is that the number of clusters must be supplied. In this section we will describe an approach for calculating that number.

After object detection we have the list of detected objects with coordinates.

For number separation problem the following approach is suggested:

1. Split nonogram into grid area and vertical/horizontal areas.
2. Calculate number of clusters for k-means algorithm.
3. Distribute objects from vertical/horizontal areas among the clusters.
4. Group objects from vertical/horizontal areas.

Each step is described in the following sections.

1. **Split nonogram into grid area and vertical/horizontal areas.**

The goal of this step is to split nonograms as is showed in Figure 33.

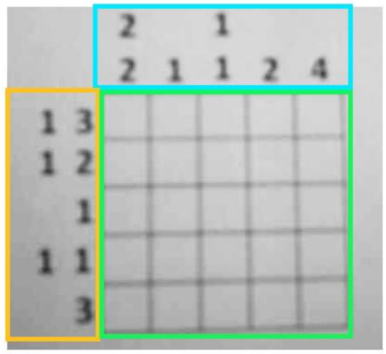


Figure 33 How nonogram should be split after step 1.

To do that coordinates of the nonogram grid must be found. For blob detectors it is an easy task, because the biggest blob always is the grid.

For both Canny and Sobel detectors this task is a bit trickier; to find grid coordinates we used Hough transformation. Hough transformation helps us to find four main lines of the grid (most prominent lines), and intersection coordinates marks 4 corners of the grid ().

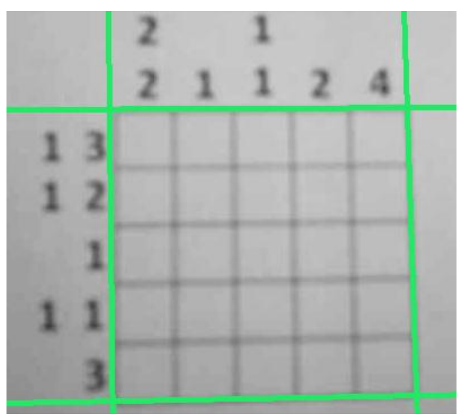


Figure 34 Nonogram with 4 grid lines

2. **Calculate number of clusters for k-means algorithm.**

The goal of this step is to find how many clusters will be used in k-mean cluster algorithm.

After step 1 we have separated nonogram parts and now having each numeral coordinates we can approximately calculate how many rows and columns nonogram have.

Algorithm for column calculation:

```
1. Calculate objects average width;
2. Sort x coordinates in ascending order;
3. ColumnCount := 1;
4. CurrentX := x1;
5. For i := 2; i < N; loop
        If CurrentX - AverageWidth < xi < CurrentX + AverageWidth
        Then
              CurrentX := xi;
        Else
              CurrentX := xi;
              ColumnCount := ColumnCount + 1;
        End if;
   End loop;
```

Rows calculation algorithm is the same: just y coordinates and average height is used.

3. **Distribute objects from vertical/horizontal areas among the clusters.**

The goal of this step is to distribute all objects into clusters. K-means algorithm is used for that. Example can be found in Figure 35.

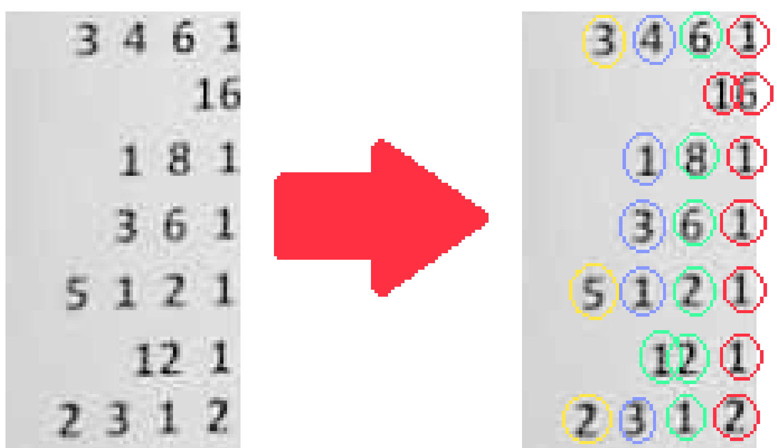


Figure 35 Clustered nonogram fragment

4. **Group objects from vertical/horizontal areas.**

The goal of this step is to group numbers in each cluster after grouping we can separate numbers with one numeral from numbers with multiple numerals. Result is 2 arrays of number objects, coordinates and group number. Object having the same group number belongs to the same number. Later these structures can be passed to number recognition module for further processing.

5. EXPERIMENTATION

First we have compared speed of 4 implemented object detection algorithms.

For data generation nonograms in 9.1 were used. For each algorithm each nonogram was run 10 times.

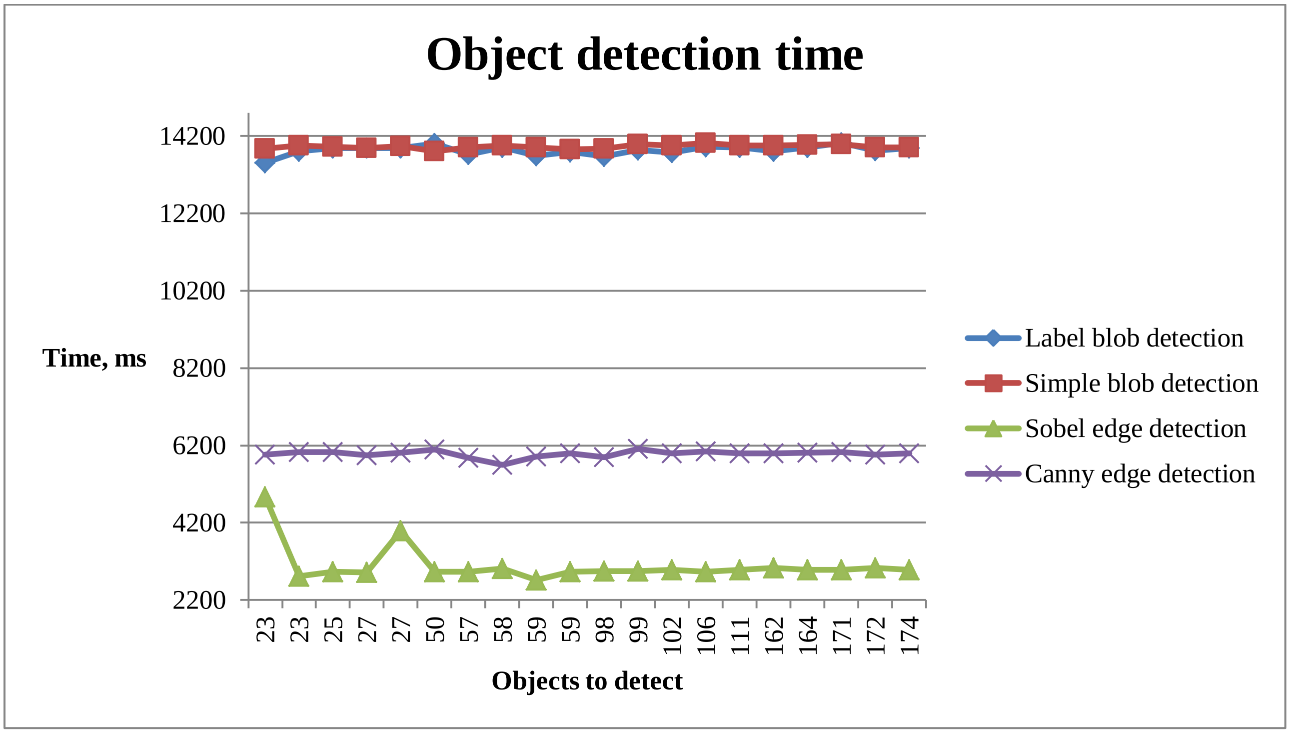


Figure 36 Object detection time graphic representation

In Figure 36 we can see that the best performance was demonstrated by Sobel edge detector followed by Canny edge detector which is about 2 times slower. Both blob detectors had quite a bad time performance comparing to analyzed edge detectors. Graphic results in Figure 36 also show that all algorithms have stable time performance while detectable object number increases.

When talking about number separation factors of speed and accuracy are considered. In this experiment speed is defined as number of iterations required for method's convergence. Accuracy is measured in percent of correctly placed number objects in clusters.

In Figure 37 we can see number of iteration required for convergence. We can clearly see that blob detectors had the worst convergence speed. This is because proposed number separation approach requires accurate grid coordinates detection and blob detectors tend to have poor accuracy while detecting grids (23).

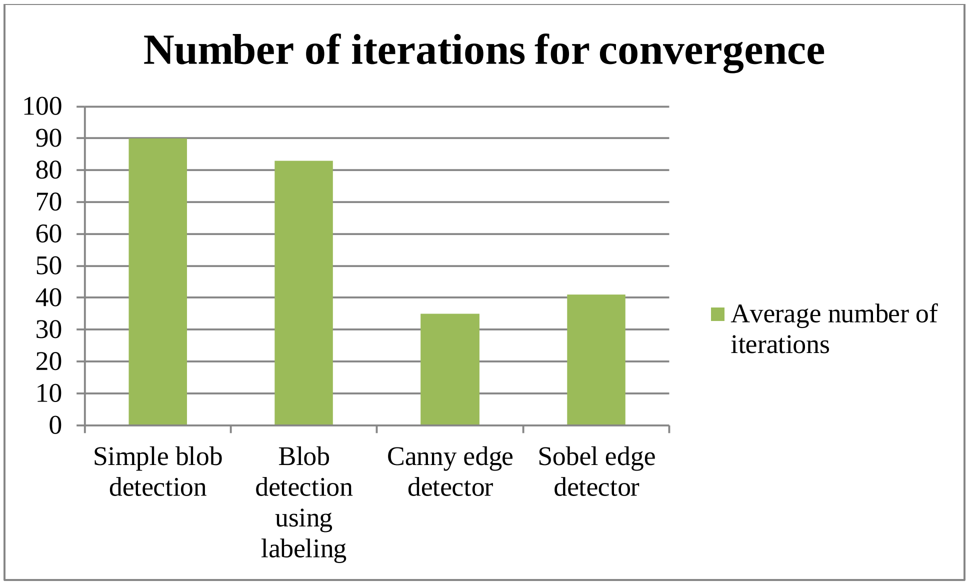


Figure 37 Number of iterations for convergence

When Canny or Sobel edge detector is used grid detection becomes more accurate and in Figure 37 we can that convergence speed increases too.

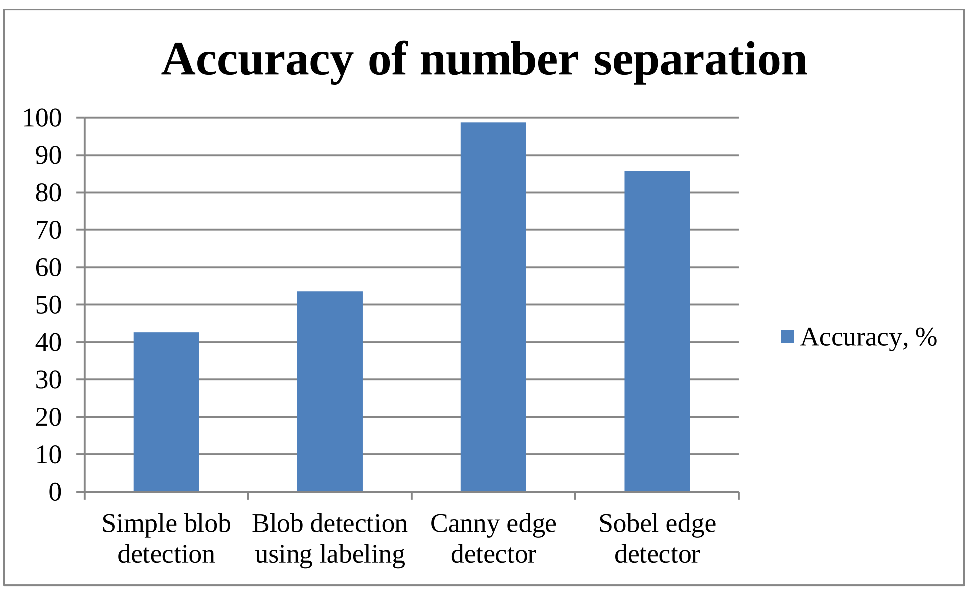


Figure 38 Accuracy of number separation approach

In Figure 38 accuracy of number separation approach is presented. We can see that there is connection between convergence speed and accuracy, because algorithms that had biggest speed has biggest accuracy and vice versa. For example, method used with Canny edge detector had best convergence speed and also had the bets accuracy when separating numbers. This is because accuracy and speed depends on initial clusters calculation (17).

6. CONCLUTIONS

1. Object detection has number separation problem, this problem can be solved with k-means algorithm.
2. Experiment showed that number separation performance depends on object detection accuracy. Suggested number separation approach had best performance when Canny edge detector was used for object detection.
3. Experiment showed that Sobel edge detector had best time while running on mobile device.
4. Experiment proved that the speed of analyzed object detection algorithms has no correlation with number of detectable objects.
5. After analysis and evaluation Canny edge detector is suggested for future augmented reality applications.

7. BIBLIOGRAPHY

1. **Scheinerman M.** *Exploring Augmented Reality.* Haverford : Haverford College, 2009.

2. **Reicher T.** *A Framework for Dynamically Adaptable Augmented Reality Systems.* Munich : Institute of computer science at the Technical University of Munich, 2004.

3. **Sutherland I.** *A head-mounted three dimensional display.* s.l. : AFIPS Fall Joint Computer Conference, 1968.

4. **Raskar R.** *Spatially Augmented Reality, First International Workshop on Augmented Reality.* 1998.

5. **Wagner D., Schmalstieg D.** *Handheld Augmented Reality Displays.* Virtual Reality Conference, 2006.

6. **Wiggers, W.** *A comparison of a genetic algorithm and a depth first search algorithm applied to Japanese nonograms.* Twente : Twente student conference on IT, 2004.

7. **Lindeberg, T.** *Detecting salient blob-like image structures and their scales with a scale-space primal sketch: a method for focus-of-attention.* International Journal of Computer, 1993.

8. **Hinz S.** *Fast and Subpixel Precise Blob Detection and Attribution.* Genua, 2005. p. 457–460.

9. **Rosenfeld A.** *Detecting image primitives using feature pyramids.* Journal of Information Sciences, 1998 m., p. 127-147.

10. **Choi M.B.** *Local Threshold and Boolean Function Based Edge Detection.* 3, s.l. : IEEE Transactions on Consumer Electronics, 1999 m., T. 45.

11. **Anto N.** *Sobel Edge Detection Algorithm.* s.l. : B.Tech Seminar, 2010.

12. **Canny J.** *A computational approach to edge detection.* 1986.

13. **Nadernejad E., Sharifzadeh S., Hassanpour H.** *Edge Detection Techniques: Evaluations and Comparisons.*. s.l. : Applied Mathematical Sciences, 2008 m., T. 2.

14. **Yuancheng L., Ramani D.** *Canny Edge Detection on NVIDIA CUDA.* s.l. : IEEE, 2008.

15. **Azernikov S.** *Sweeping solids on manifolds.* s.l. : Symposium on Solid and Physical Modeling, 2008. p. 249–255.

16. **Mai F., Hung Y., Zhong H. and Sze W.** *A hierarchical approach for fast and robust ellipse extraction.* 8, s.l. : Pattern Recognition, 2008 m., T. 41, p. 2512–2524.

17. **Pal N.R.** *A review on image segmentation techniques.* s.l. : Pattern Recognition, 1993 m., T. 26, p. 1277-1294.

18. **King, M. Amadasun R.A.** *Low level segmentation of multispectural images via agglomerative clustering of uniform neighbours.* s.l. : Pattern Recognition, 1988 m., T. 21.

19. P**avlidis, S.L. Horowitz T.** *Picture segmentation by directed split and merge procedure.* Copenhagen : Proc. 2nd Int. Joint Conf. Pattern Recognition, 1974.

20. **Ohlander R., Price K. and Reddy D.R.** *Picture segmentation using a recursive region splitting method.***.** s.l. : Comput. Graphics Image Process., 1978 m., T. 8, p. 313-333.

21. **Gonzalez J.T.** *Pattern Recognition Principles.* Massachusetts : Addison-Wesley, 1974. T. 56.

22. **Tudor B., Pettey C.** *Gartner Says Worldwide Mobile Phone Sales Grew 35 Percent in Third Quarter 2010.* s.l. : Gartner, 2010.

23. **Greensted A.** Blob detection - The lab book pages. *The lab book pages.* http://www.labbookpages.co.uk/software/imgProc/blobDetection.html.

24. **Cowles J.** Visual Core. *Simple recursive blob detection.* http://visualcore.com/index.php/2008/02/simple-recursive-blob-detection/.

25. **Southampton, Electronics and Computer Science University of.** Computer Vision Demonstration Website. *Sobel edge detection - Computer Vision Demonstration Website.* http://users.ecs.soton.ac.uk/msn/book/new_demo/sobel/.

26. **Gibara, T.** *Canny edge detection implementation.* http://www.tomgibara.com/computer-vision/canny-edge-detector.

8. ACRONYMS

AR – augmented reality

PDA – personal digital assistant

HMD – head-mounted display

SAR – special augmented reality

GPS – global positioning system

OS – operating system

HCI – human-computer interaction

ARS – augmented reality system

HMC – head mounted camera

9. APPENDICES

9.1. NONOGRAMS USED FOR EXPERIMENT

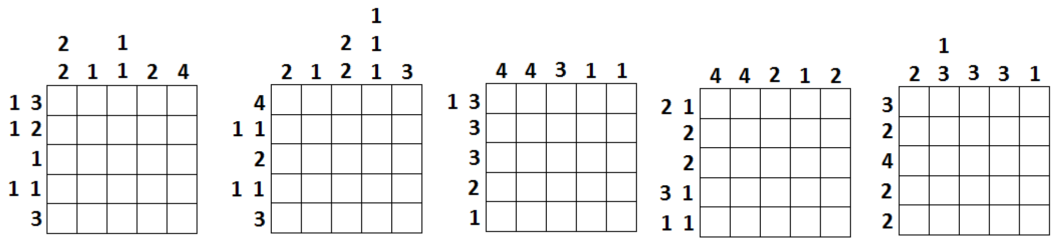


Figure 39 All 5x5 nonograms used for experiment

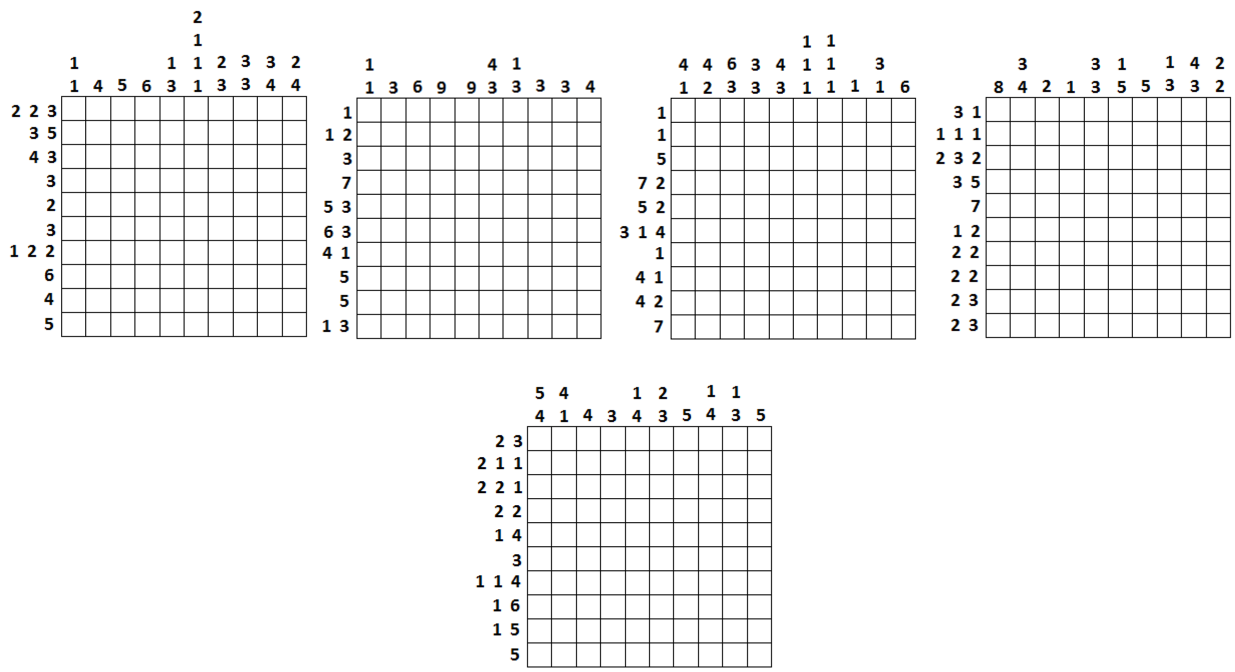


Figure 40 All 10x10 nonograms used for experiment

Column clues: 3 3 2 | 2 2 | 5 2 | 6 4 1 | 8 3 | 12 | 4 5 1 | 2 7 | 2 10 | 2 2 4 1 | 2 1 5 | 2 4 | 2 1 | 1 | 1 1

Row clues:
1 4
1 9
1 5
5 1
4 1
7
2 8
2 5 2
1 4 1 1
1 5
3 4
9
2 5
3 2
4 3 2

Column clues: 1 3 1 | 1 3 | 2 3 | 1 1 1 1 5 | 1 1 1 5 | 6 5 | 8 2 | 10 1 | 1 6 | 1 5 1 | 2 | 3 | 5 3 | 7 5 | 6 5

Row clues:
3 1 2
6 2
3 3
8 3
3 3 3
8 3
2 3 2
1 4
4
3 2
3 3 2
3 2 4
5 4
1 4 6
3 1

Column clues: 4 | 1 5 | 1 4 | 4 4 | 3 1 2 | 6 2 | 1 1 2 1 | 1 3 3 5 | 1 1 1 9 | 3 10 | 1 1 3 1 | 1 1 4 | 3 1 4 | 4 | 4 6

Row clues:
2 2
1 4
7 3
3 1 1 4
6
3 1 1
7
6 1 1
3 1
5 1
2 6 1
6 6 1
10 2 1
4 3
3 4

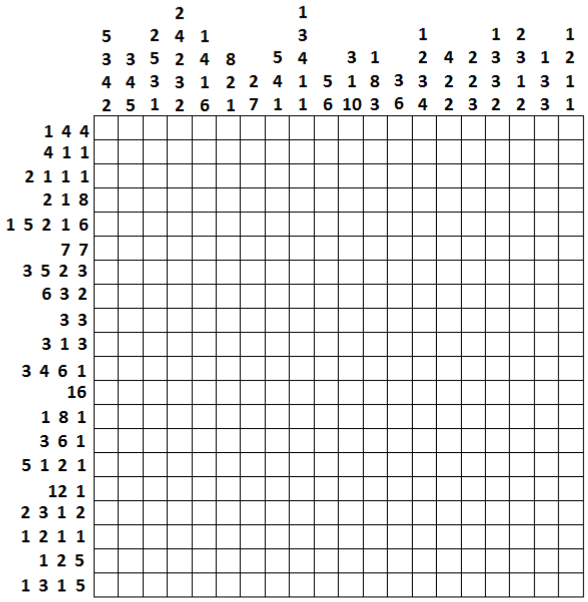
Column clues: 6 | 7 1 | 5 1 1 | 6 1 | 5 2 | 6 3 | 6 2 | 5 3 | 1 2 3 | 1 3 3 | 3 4 | 3 4 2 | 3 5 1 1 | 3 1 2 2 | 2 2 2

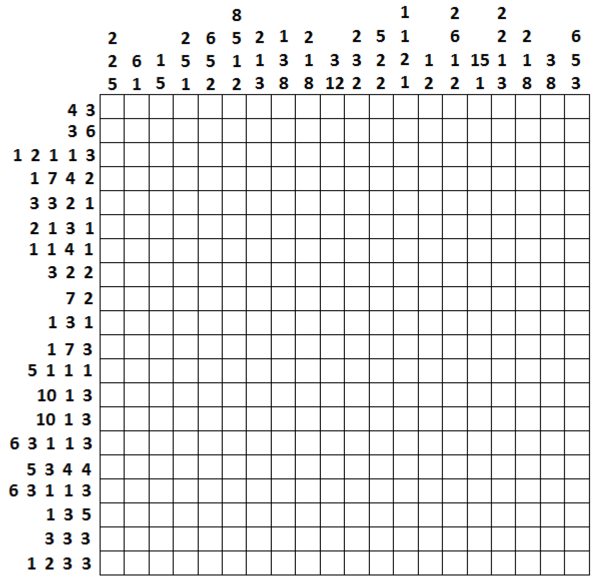
Row clues:
7
5
4 4
7
10 1
13 1
8 5
4 3 3
2 2
2 1
1 1 2
3
1 3
6 3
9 1 1

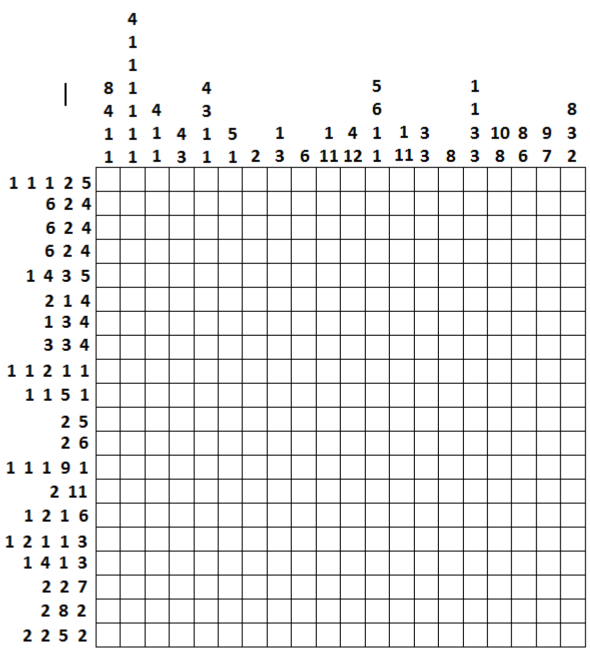
Column clues: 3 2 2 | 3 1 | 1 1 2 | 2 3 3 | 2 6 | 6 3 | 5 2 | 3 2 4 | 3 2 5 | 3 4 3 | 1 5 1 | 3 4 1 | 3 6 | 3 1 1 1 | 1 1 3 1 1

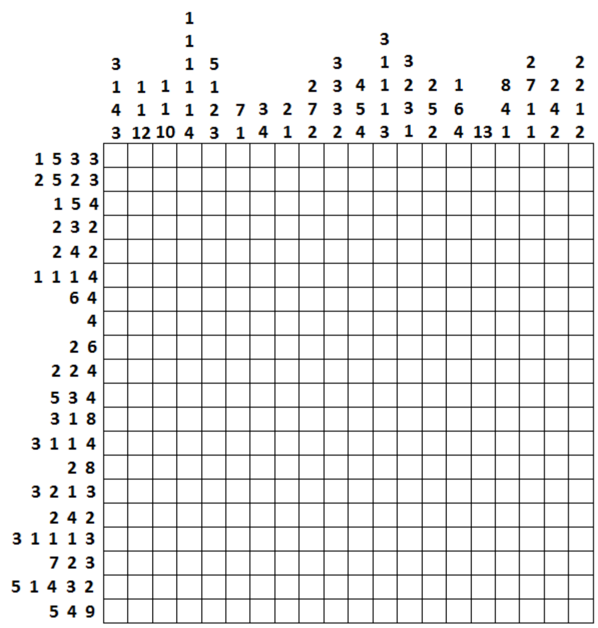
Row clues:
2 8
1 3 3
2 3 4
2 1
6 3
4 1 1 1
2 1 5
1 8
1 3 6
3 4
1 2
1 2
7 1
1 8
6 5

Figure 41 All 15x15 nonograms used for experiment









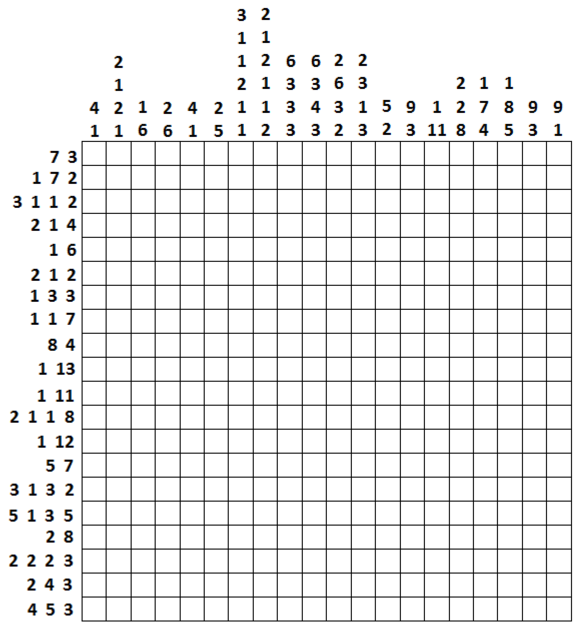


Figure 42 All 20x20 nonograms used for experiment