Universitatea din București
Facultatea de Matematică și Informatică

LUCRARE DE LICENȚĂ

Corectarea automată a testelor grilă

Student:
Ciprian-Mihai CEAUȘESCU

Îndrumător științific:
Conf. univ. dr. Bogdan ALEXE

București
Iunie, 2017

Cuprins

Prefață

De-a lungul timpului, evaluarea cunoștințelor unui elev sau student a constituit un subiect destul de controversat, din punct de vedere al impactului său moral, deoarece rezultatul acesteia conduce la realizarea unor clasamente și selecții, impactând viitorul fiecărei peroane care participă la acea evaluare.

Evaluarea reprezintă un proces destul de amplu, care presupune ca fiecare evaluator să acorde o notă care poate fi sub forma unui calificativ sau punctaj elevului sau studentului, fără a apela la subiectivism, care să reflecte capacitatea sa de efectuare a unor activități care i-au fost explicate anterior.

Formele prin care un evaluator poate deveni subiectiv sunt:
- evaluarea elevilor sau studenților nu se face în funcție de obiectivele vizate de un anumit curs și de conținutul acestuia, ci în funcție de nivelul tuturor elevilor sau stundenților din acea clasă. Acestă modalitate de evaluare conduce la indulgența profesorului de a oferi rezultate bune

mod greșit.
- starea de moment a profesorului, fiind obosit, stresat, generos sau indiferent.

În timpul procesului de evaluare a unui elev sau student intervin anumiți factori rezultați din activitatea și natura personalității evaluatorului respectiv. Cei mai importanți dintre acești factori sunt:
- efectul "Halo": rezultatul evaluării se raportează la impresia pe care și-a făcut-o evaluatorul despre elev sau student. Problema care apare în cazul acestui factor este acela că un elev sau student care are o reputație bună poate primi o notă mai mare, în comparație cu un elev sau student care nu are aceeași reputație, însă cunoștințele lor sunt similare.
- efectul "blând": un evaluator deține o capacitate mai bună de acordare a notelor pentru elevi sau studenți abia după o perioadă de cunoaștere a acestora.elevilor săi, rezultate care nu reflectă însă realitatea.
- subaprecierea unor elevi pentru că răspund după ce răspunde un elev mai bun, iar această diferență conduce la evaluarea sa într-un
- efectul "Pygmalion": elevul sau studentul își schimbă comportamentul în funcție de modul în care este apreciat de către profesor.
- generiozitatea: modul de acordare a unor note mari comparativ cu pregătirea elevului sau studentului respectiv.
- contaminarea: celelalte note ale elevului sau studentului influențează rezultatul primit la evaluarea curentă.

Pentru a evita situațiile în care subiectivismul poate influența în mare parte rezultatul unui elev sau student, se poate apela la o abordare sub forma unui test grila, a cărui rezultat va constitui nota persoanei evaluate.

Lucrarea de față propune un algoritm de corectare a testelor grilă, folosind o interfață grafică prietenoasă. Formatul acestor teste grilă este folosit la concursul de admitere la domeniul de studii Calculatoare și Tehnologia Informației, specializarea Tehnologia Informației, din cadrul Facultății de Matematică și Informatică, Universitatea din București.

Lista figurilor

Capitolul 1

Introducere

De-a lungul timpului, se observă faptul că tendințele din lumea tehnologiei sunt într-o continuă schimbare. Aceste schimbări vor influența modul în care noi, oamenii, vom folosi tehnologia. În ultima vreme accentul cade în special pe: cloud computing, mobilitate, social business, dar și pe conceptul foarte important de big data. Susținerea acestor piloni este realizată de realitatea virtuală și augumentată, robotică, sisteme cognitive, dar și Internet of Things, tehnologii importante care au ca nucleu de interes *inteligența artificială*.

Conceptul de inteligență artificială este reprezentat de un domeniul al informaticii al cărui prim scop este acela de a dezvolta sisteme tehnice capabile să rezolve probleme dificile legate de inteligeța umană. Acest domeniu studiază creierul uman, astfel că se încearcă reproducerea modului în care informația este analizată și procestă în momentul rezolvării problemelor. Se definesc două paradigme: *simbolică și conecționistă*. Paradigma simbolică ilustrează faptul că sistemul cognitiv prezintă cunoștințele și stările lucrurilor înconjurătoare prin simboluri, diferit față de cea conecționistă care prezintă informația prin activarea unor unități simple, neurori, pe baza unor valori de activare.

Aplicația de corectare automată a testelor grilă care este descrisă în prezenta lucrare aplică principiile unui subdomeniu al inteligenței artificiale, și anume, vederea articială.

Conceptul de vedere artificială reprezintă modul în care o mașină de calcul poate extrage informații din imagini digitale pe care ulterior le va procesa. De asemenea, domeniul vederii artificiale include o gamă variată de procese prin care se încearcă simularea sistemului vizual uman.

Printre cele mai importante operații care se pot realiza în vederea ariticială se regăsesc procesarea, analizarea și înțelegerea imaginilor digitale, dar și analiza unei cantități mari de date din lumea reală cu scopul de a produce informație sub formă numerică sau simbolică, în funcție de deciziile mașinii de calcul la un moment de timp. Procesele sistemului vizual uman sunt destul de complexe. În mai puțin de o zecime de secundă, o persoană poate analiza o cantitate destul de mare de informație. Retina sistemului vizual uman este capabilă să proceseze cu ușurință 10^9 operații pe secundă, iar cortexul vizual are o capacitatea de procesare chiar mai mare decât retina.

Imaginile sunt formate pe un anumit senzor, în momentul în care acesta din urmă primește un răspuns reprezentat de lumina reflectată de către o sursă, emisă direct asupra unei scene. Pentru a se produce răspunsul trebuie să existe o lentilă potrivită care produce o corespondență între scenă și imaginea rezultată. De asemenea, trebuie să existe și o sursă de lumină care ajută la crearea imaginii formate cu ajutorul intesității luminoase și opțional al culorii reflectate de fiecare punct al scenei (figura 1.1).

Culorile dintr-o imagine pot fi obținute cu ajutorul combinării diferitelor intesități luminoase a celor trei canale principale de culoare R - roșu, G - verde, respectiv B - albastru. Pentru a fi definită o anumită culoare, trebuie să plecăm de la valorile fiecărui canal de culoare: $r = \frac{R}{R+G+B}$, $g = \frac{G}{R+G+B}$ and $b = \frac{B}{R+G+B}$, astfel încât r+g+b=1.

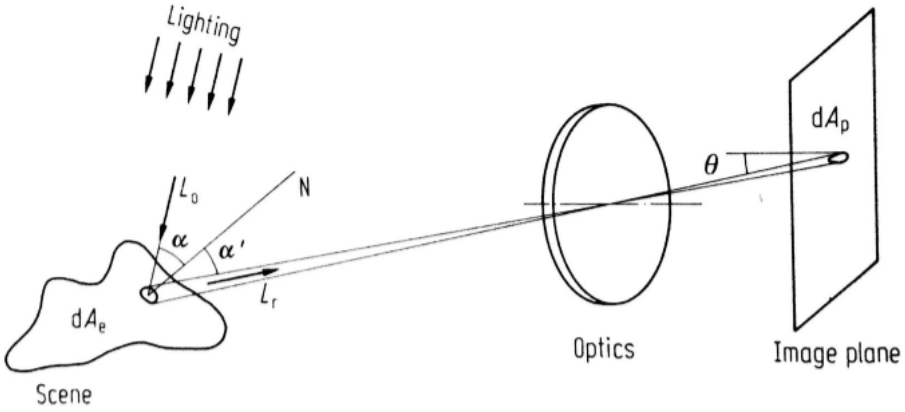


Figura 1.1 Formarea imaginii

Sistemul vizual uman este unul complex din punct de vedere al puterii lui de analiză a imaginilor digitale, având capacitatea de a separa în mod corect obiectele dintr-o imagine de fundalul acesteia. De exemplu, se pot obține informații despre forma și transparența unei patale în diverse faze ale sursei de lumină, dar și comparativ cu fundalul imaginii din care floarea face parte (figura 1.2).

Figura 1.2 Sistemul vizual uman partiționează imaginea în mai două segmente: floare, respectiv fundal

De-a lungul timpului, psihologii au încercat să descopere modul în care funcționează sistemul vizual uman. În urma cercetărilor realizate s-a ajuns la concluzia că domeniul vederii artificiale poate acoperi cu o mai mare exactitate o anumită arie a imaginilor digitale.

Se vor analiza o serie de iluzii optice care demonstrează faptul că în anumite condiții aplicațiile din domeniul vederii artificiale dau un răspuns sigur în urma procesării imaginilor digitale, comparativ cu sistemul vizual uman care întâmpină anumite dificultăți.

În figura 1.3 se prezintă iluzia clasică Muller-Lyer, unde cele două linii orizontale par a avea lungimi diferite, probabil din cauza efectelor de perpectivă diferite.

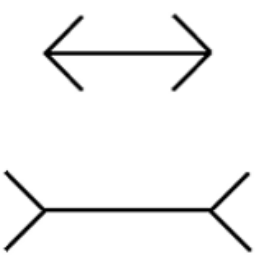


Figura 1.3

În figura 1.4 se prezintă iluzia optică a pătratului "alb" B aflat în umbră și a pătratului "negru" A aflat în lumină, care par a avea intensități diferite, însă acestea au intensități identince.

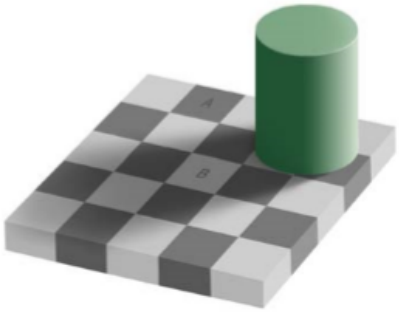


Figura 1.4

În figura 1.5 vom analiza pozițiile pe care se află x-urile roșii în partea din stângă, comparativ cu pozițiile lor în partea din dreapta. Se observă că este mult mai ușor de numărat de x-urile roșii din partea stângă decât din partea dreaptă, fapt cauzat de un răspuns "instantaneu" dat de asocierile între culori pe care le face creierul uman.

Figura 1.5

În concluzie, pentru un anumit tip de aplicații sistemul vizual uman oferă un răspuns mult mai precis, dar și mult mai rapid decât un sistem care integrează vederea artificială. De multe ori însă, un mod mai ușor de procesare și analizare al unui set mare de imagini digitale este oferit de către un sistem de vedere artificială antrenat într-un anumit scop pentru a oferi un răspuns cât se poate de precis.

Capitolul 2

Python

Aplicația de corectare automată a testelor grilă care este descrisă în prezenta lucrare a fost dezvoltată cu ajutorul limbajului de programare Python. Python este utilizat pentru o mare varietate de aplicații, fiind astfel un limbaj de programare uzual, proiectat de Guido van Rossum și dezvoltat de compania Python Software Foundation. Unul dintre avantajele importante ale utilizării acestui limbaj de programare este acela ca Python este un limbaj de scripting, astfel încât codul scris se interpretează, nu se compliează, fapt care generează un timp redus în dezvoltarea și testarea aplicațiilor. Paradigmele de programare (prin paradigmă a se înțelege modul în care putem clasifica limbajele de programare), pe care le adoptă limbajul Python sunt următoarele:

1. orientată obiect - bazată pe conceptul de obiect care conține date, sub formă de câmpuri (atribute) și proceduri (metode)
2. imperativă - bazată pe o execuție secvențială a intrucțiunilor care modifică starea programului
3. funcțională - bazată pe aplicarea de funcții, astfel încât calculele devin o evaluare de funcții matematice
4. procedurală - bazată pe apeluri de proceduri definite anterior, fiind similara cu paradigma funcțională
5. reflectivă - bazată pe capacitatea unui program de a-și schimba stuctura și comportamentul în timpul execuției acestuia.

Aplicația de corectare automată a testelor grilă a fost realizată în mediul de dezvoltare PyCharm, dezvoltat de compania cehă JetBrains, fiind un mediu specializat care oferă funcționalități de analiză a codului, testare unitară, integrare cu sisteme de gestionare a mai multor versiuni ale unei aplicații, dezvoltare de aplicații web folosind Django. Fișierele dezvoltate în Python se numesc scripturi și au extensia ".py". Un aspect important este faptul că un proiect realizat în PyCharm este stucturat în mai multe fișiere cu extensia ".py", definite ca module. Un modul poate importa alte module, astfel încât aceste importuri trebuie realizate la începutul unui script sau modul. Fiecare modul are definit un tabel de simboluri, în care se vor salva informații despre variabilele folosite în script, dar și numele modulelor importate de acesta. În cazul importării fară a se utiliza un anumit modul, se va primit un mesaj de atenționare (s-a efectuat o importare a unui modul care nu se folosește).

Stuctura de date de bază utilizată în aplicație este reprezentată de matrice, motivul fiind faptul că imaginile au o structura matriceala. Pentru o imagine în tonuri de alb și negru, adică o imagine binară, se poate folosi o matrice bidimensională, în care fiecare pixel are o valoare de 0, adica negru sau 1, adica alb. Pentru o imagine în tonuri de gri se poate folosi o matrice bidimensională, în care fiecare pixel are o valoare între 0 și 255, unde 0 înseamnă negru, 255 înseamnă alb, iar 127 cea mai bună nuanță de gri. Pentru imaginile color se poate folosi o matrice tridimensională, pentru că se dorește o reprezentare RGB (Red Green Blue), motiv pentru care fiecare pixel reprezintă o combinație din intensitatea canalului roșu (Red), a celui verde (Green) și a celui albastru (Blue). Pentru a obține o imagine în tonuri de gri dintr-o imagine color, fiecare pixel care aparține imaginii în tonuri de gri va avea o valoare calculată folosind formula următoare:

$x = 0.299 * r + 0.587 * g + 0.114 * b$, unde r, g, b sunt valorile fiecărui canal în parte, care aparțin pixelului color, iar x este valoarea rezultată care va fi atribuită pixelului din imaginea în tonuri de gri.

În dezvoltarea aplicației descrisă în această lucrare s-au utilizat următoarele module pe care limbajul Python le poate importa:

1. NumPy
2. Tkinter
3. UnitTest
4. OpenCV
5. TensorFlow

Rolul acestor module urmează a fi explicat în continuare.

2.1 NumPy

Modulul NumPy (Numerical Python) este o extensie care aparține limbajului Python, având ca scop fundamental manipularea unor stucturi de date de dimensiuni mari, printre care mulțimile de elemente (arrays) și matricile (matrices), dar și o varietate de funcții matematice de nivel înalt care operează peste aceste structuri de date. Obiectul principal folosit de către acest modul este o mulțime multidimensională, definită sub forma unui tabel de elemente (de obicei numere), toate având același tip de date, indexată folosind un tuplu format din numere întregi, pozitive. În NumPy, dimensiunile unui vector sau ale unei matrice poartă denumirea de *axe*, iar numărul total al acestora se numește *rang*.

Exemplu 1: Coordonatele unui punct într-un spațiu 3D, [1, 2, 1] este o mulțime care are rangul 1, deoarece are o singură axă, iar aceasta are dimensiunea 3.

Exemplu 2: Mulțimea [[1., 0., 0.], [0., 1., 2.]] are rangul egal cu 2, fiind astfel o mulțime de elemente bidimensională. Prima dimensiune, adică prima axă are lungimea 2, iar cea de-a doua dimensiune are lungimea 3.

Clasa care aparține modulului NumPy, și care are rolul de a manipula mulțimile se numește *ndarray*. Printre cele mai importante câmpuri ale unui obiect de tipul ndarray se află:
- *ndarray.ndim* - reprezentând numărul de axe (dimensiuni) ale mulțimii (echivalentul rang-ului unei mulțimi)
- *ndarray.shape* - tuplul de numere întregi pozitive care reprezintă dimensiunea fiecărei axe a mulțimii (pentru o matrice de *n rânduri* și *m coloanei* valoarea tupului va fi (n,m)
- *ndarray.size* - numărul total de elemente ale mulțimii, având o valoare egală cu produsul elementelor întregi din tuplul returnat de câmpul shape.

2.2 Tkinter

Modulul Tkinter este o bibliotecă standard, creată pentru limbajul de programare Python, necesară pentru crearea interfețelor grafice dintre utilizator și mașina care găzduiește o anumită aplicație. Python, alături de Tkinter, oferă programatorului un mod ușor și intuitiv pentru crearea unei interfețe grafice care să ofere o anumită funcționalitate utilizatorului (se folosește o paradigmă orientată obiect).

Pașii care trebuie parcurși pentru crearea unei interfețe grafice în Python sunt:
- importarea modulului *tkinter*
- crearea ferestrei principale a aplicației (cea care se va deschide prima dată în momentul rulării aplicației)
- adăugarea unor componente interfeței (butoane, câmpuri în care se poate introduce text)
- adăugarea unui eveniment care poartă denumirea de "mainloop", cu scopul de a acționa în momentul în care utilizatorul folosește o anumită componentă adăugată în interfață.

2.3 UnitTest

Modulul UnitTest este biblioteca standard pentru testare unitară a aplicațiilor scrise în Python, deseori găsindu-se sub denumirea de "PyUnit", dezvoltată de Kent Beck și Erich Gamma. Acest modul oferă o listă de clase care permit efectuarea testelor automate asupra aplicației, partajarea unor setări speciale pentru pornirea sau oprirea testelor, agregarea unor teste într-o anumită colecție (spre exemplu crearea unei suite de teste pe baza claselor de echivalență existente în cadrul unei aplicații).

Pentru o bună funcționare a activității de testare, modulul UnitTest oferă câteva concepte importante, care interconectate, conduc la realizarea cu ușurință a acestei componente importante din dezvoltarea unei aplicații software:
- pregătirea testului: reprezintă activitatea premergătoare testării propriu-zise. În unele cazuri poate conduce la curățarea unor baze de date de test, crearea unor noi baze de date sau a unor directoare noi.
- cazuri de testare: reprezintă cea mai mică unitate de testare, folosită pentru a determina răspunsul unei componente a aplicației pentru un caz particular de parametrii. Modulul UnitTest oferă o clasă de bază, TestCase, care se utilizează în momentul creării unui caz de testare.
- suită de teste: reprezintă o colecție de cazuri de testare sau chiar o colecție de suite de teste. Scopul acesteia este de a agrega testele care trebuie executate împreună.
- componenta care execută testele și oferă răspuns imediat programatorului. Se oferă informații cu privire la testele care s-au rulat, alături de rezultatele acestora.

2.4 OpenCV

Modulul OpenCV(Open Source Computer Vision) este o bibliotecă folosită în dezvoltarea aplicațiilor de vedere artificială. Dezvoltarea acestui modul, de către Gary Bradsky, a început în anul 1999 la Intel, iar prima variantă a fost finalizată în anul 2000. În prezent, OpenCV suportă o varietate de algortmi din domeniul vederii artificiale, dar și al machine learning-ului.

OpenCV-Python este API-ul corespunzător bibliotecii standard de OpenCV, care combină modulul de vedere artificială din limbajul C++ și limbajul Python.

2.5 TensorFlow

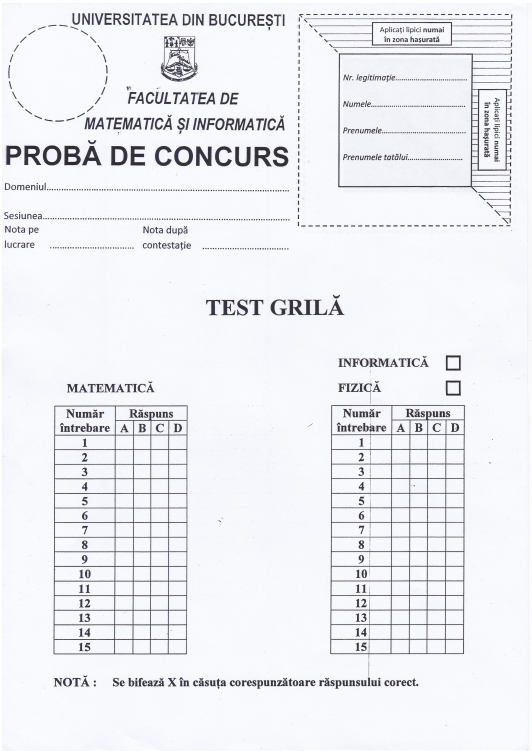
TensorFlow este o bibliotecă open source folosită pentru operații de calcul științific, unde datele sunt reprezentate sub forma unui graf. Această bibliotecă este folosită în dezvoltarea aplicațiilor de *machine learning*, dezvoltată de către cei de la Google pentru crearea, antrenarea rețelelor neurale cu scopul de a detecta și descifra modele similare cu cele ale gândirii umane.

Machine learning este un subdomeniu al informaticii, care după definiția lui Arthur Samuel din 1959 reprezintă "abilitatea mașinilor de calcul de a învăța și reproduce o anumită funcționalitate fară a fi programată în acest sens". Printre aplicațiile acestui subdomeniu se regăsesc filtrarea email-urilor, detectarea răufăcătorilor dintr-o rețea, detectarea caracterelor dintr-o imagine (OCR - Optical Character Recognition).

Graful sub care sunt reprezentate datele are nodurile reprezentate de operațiile matematice, iar muchiile de vectori multidimensionali care se numesc tensori. Se definește *rang-ul* unui tensor dimensiunea acestuia. De exemplu, tensorul [1, 2, 3] este un vector cu rang-ul 3.

Capitolul 3
Implementare

Lucrarea de față propune un algoritm de corectare a testelor grilă care au un format prezentat în figura 2.1. Subiectele primite de către elevi sunt structurate în 4 tipuri de variante (numerotate de la 1 la 4), fiecare având un subiect de matematică, unul de informatică și unul de fizică. Grila de matematică este obligatorie, iar dintre grilele de fizică și informatică elevul trebuie sa rezolve una. În funcție de tipul grilei pe care elevul decide să o rezolve, se va trece numărul corespunzător acesteia (1, 2, 3 sau 4) în căsuța de fizică sau informatică, după cum urmează: dacă un elev primește varianta 2 și alege să rezolve grila de fizică, se va completa numărul 2 în dreptul opțiunii dorite (figura 2.2).


UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

PROBĂ DE CONCURS

Aplicați lipici numai în zona hașurată

Nr. legitimație

Numele

Prenumele

Prenumele tatălui

Domeniul

Sesiunea

Nota pe lucrare ..... Nota după contestație .....

TEST GRILĂ

INFORMATICĂ ☐

FIZICĂ ☐

MATEMATICĂ

Număr întrebare	Răspuns A	B	C	D
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				

Număr întrebare	Răspuns A	B	C	D
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				

NOTĂ : Se bifează X în căsuța corespunzătoare răspunsului corect.

Figura 2.1

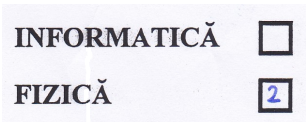


Figura 2.2

Corectarea unui set de teste completate de către elevi se bazează pe următorul scenariu:

1. scanarea tuturor testelor elevilor și salvarea acestora într-un anumit director
2. introducerea răspunsurilor corecte de către comisia de admitere în aplicație
3. pregătirea imaginii pentru a fi procesată - eliminarea zgomotului care apare în urma printării și scanării acesteia
4. găsirea în imagine a celor două grile existente

5. determinarea răspunsurilor bifate de către elev
6. determinarea tipului de grilă la care elevul a răspuns (fizică sau informatică), dar și a numărului acestei grile (1, 2, 3 sau 4)
7. compararea răspunsurilor bifate de către elev cu cele introduse de către comisia de admitere
8. determinarea notei finale care se bazează pe formula: Nota finală = 0.3p * numărul răspunsurilor corecte + 1p (oficiu).

3.1 Pasul 1

Scanarea tuturor lucrărilor completate de către elevi se va face cu ajutorul unui scanner, iar în urma acestui proces imaginile se vor salva într-un director.

3.2 Pasul 2

Înterfața grafică a aplicației descrise în prezenta lucrare conține o funcționalitate de adăugare a răspunsurilor corecte. Comisia de admitere accesează această funcționalitate, introducând valoarea 1 pe poziția pe care răspunsul corect la o anumită întrebare, respectiv 0 pentru cele 3 poziții incorecte (figura 2.3). De asemenea, se va alege tipul grilei pentru care se vor introduce rezultatele la un anumit moment (exemplu: Matematica 1 - reprezintă grila corespunzătoare variantei 1).

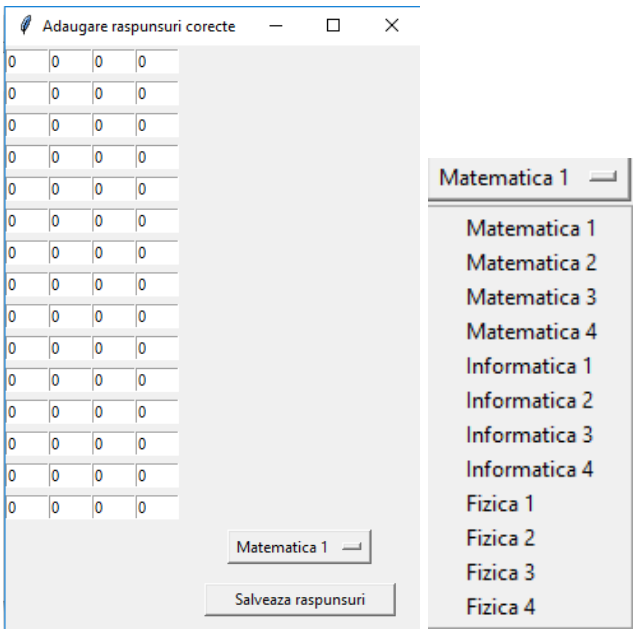


Figura 2.3

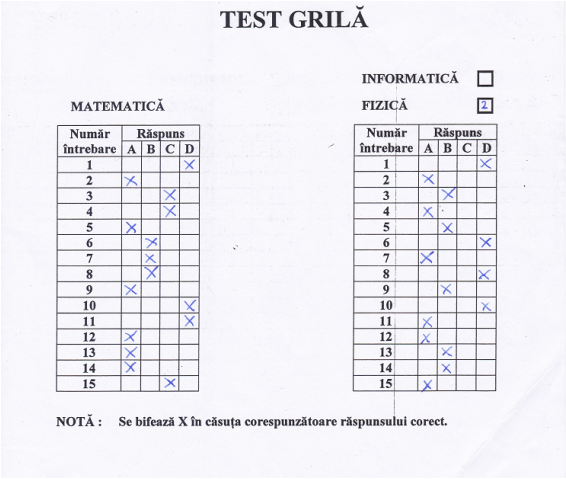
3.3 Pasul 3

Pregătirea imaginii pentru a fi procesată este un pas foarte important, adăugat în prezenta lucrare după primele rezultate ale detectării grilelor existente în pagină. Datorită imprimării grilelor, dar și a scanării acestora este inevitabil ca imaginile rezultate să nu aibă zgomot. Drept urmare acest pas este unul util în vederea detectării corecte a informațiilor dorite.

Eliminarea zgomotului din imaginile rezultate a fost realizată prin definirea unui modul care conține 4 metode:
- prepareImage(img): metodă care primește ca parametru o imagine pe care o transformă din imagine RGB în imagine în tonuri de gri, aplicându-i un filtru de blurare și returnează imaginea “outputSmooth.jpg“
- removeNoiseByPixel(img, column, line, passFactor: care primește ca parametru o imagine, pozițiile unui pixel în imagine (coloană, rând) și o valoare numerică între 0 - 255. Rezultatul acestei metode este o imagine în care pixelii cu o intensitate sub valoarea lui passFactor devind 0 (pixeli negri), iar cei cu o intensitate peste valoarea lui passFactor devin 255 (pixeli albi).
- removeNoise(img, passFactor): metodă care primește ca parametru o imagine și o valoare numerică între 0 - 255, va parcurge fiecare pixel al imaginii, apelând metoda removeNoiseByPixel
- removeNoiseMain(imgName, passFactor): metoda principală a modulului, care citește imaginea, o redimensionează pentru ca timpul procesării să fie mai mic și apelează pentru imaginea citiă metodele prepareImage și removeNoise. Rezultatul imaginii se va numi “noiseRemoved.jpg“.

În urma pregătirii imaginii pentru a fi procesată se observă rezultatul aplicării metodei removeNoiseMain asupra imaginii inițiale:

1. Imaginea inițială



TEST GRILĂ

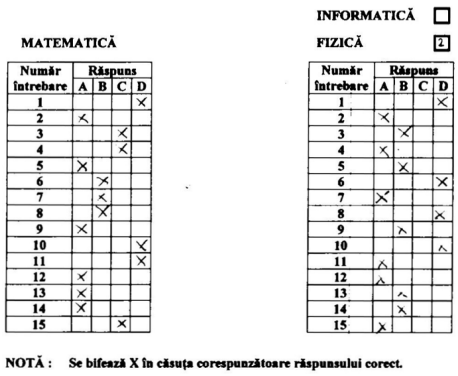
INFORMATICĂ ☐
MATEMATICĂ FIZICĂ ☒

Număr întrebare	Răspuns A	B	C	D
1				X
2	X			
3			X	
4			X	
5	X			
6		X		
7		X		
8		X		
9	X			
10				X
11				X
12	X			
13	X			
14	X			
15			X	

Număr întrebare	Răspuns A	B	C	D
1				X
2	X			
3		X		
4	X			
5		X		
6				X
7	X			
8				X
9		X		
10				X
11	X			
12	X			
13		X		
14		X		
15	X			

NOTĂ : Se bifează X în căsuța corespunzătoare răspunsului corect.

2. Imaginea noiseRemoved



TEST GRILĂ

INFORMATICĂ ☐
MATEMATICĂ FIZICĂ ☒

Număr întrebare	Răspuns A	B	C	D
1				X
2	X			
3			X	
4			X	
5	X			
6		X		
7		X		
8		X		
9	X			
10				X
11				X
12	X			
13	X			
14	X			
15			X	

Număr întrebare	Răspuns A	B	C	D
1				X
2	X			
3		X		
4	X			
5		X		
6				X
7	X			
8				X
9		X		
10				X
11	X			
12	X			
13		X		
14		X		
15	X			

NOTĂ : Se bifează X în căsuța corespunzătoare răspunsului corect.

3.4 Pasul 4

Găsirea în imagine a celor două grile existente implică definirea mai multor module în aplicația descrisă în prezenta lucrare.

Un prim modul este findTest, în care se definește o metodă cu aceeași denumire care primește ca parametru numele unei imagini. Metoda parcurge următorii pași: