Павлюк О.В., Савчинський Б.Д.

Ефективний синтаксичний аналіз та розпізнавання структурованих зображень

1 Вступ

Граматичний підхід до розпізнавання структурованих зображень широко використовується для вирішення різноманітних прикладних задач [1, 2, 3]. Для робіт в цьому напрямку є характерним розгляд зображення, як об'єкта, що складається за певними правилами з великої кількості елементарних частин. Незважаючи на те, що ці частини і правила можуть сильно відрізнятись між собою (наприклад, так, як при розпізнаванні нот [4] та математичних формул [5]), різниця між відповідними їм алгоритмами розпізнавання є значною лише на перший погляд. В роботі [6] описаний формалізм, названий **загальною структурною конструкцією**, що включає в себе граматики, описані в великій кількості робіт, присвячених конкретним прикладним задачам, як окремі випадки. Окрім власне визначення загальної структурної конструкції в [6] наведені загальна постановка та алгоритм розв'язку будь-якої задачі розпізнавання в рамках введеного формалізму. Цей формалізм, безумовно, має значний світоглядний вплив, оскільки описує цілий ряд задач, що до того вважались зовсім різними, з однієї загальної точки зору. Більш за це, він з єдиної точки зору подає навіть на перший погляд цілком різні задачі розпізнавання, – задачі на точний та найкращий збіг.

Задача на точний збіг полягає у відповіді на питання про належність вхідного зображення мові, що генерується певною структурною конструкцією, і, за позитивної відповіді, у відшуканні виводу зображення як послідовності правил, згідно яких воно може бути побудоване. Прикладом роботи, в якій задача формулюється саме таким чином, є [3]. Для формулювання задачі на найкращий збіг кожному правилу приписується деякий штраф за його використання. Розпізнавання вхідного зображення полягає у відшуканні його виводу з мінімальним штрафом. Така постановка використовується, наприклад, у роботі [4].

Незважаючи на всеохоплюючу загальність, а може саме завдяки їй, безпосереднє застосування загальних структурних конструкцій в практичних задачах наштовхується на серйозні практичні труднощі, що пов'язані перш за все із складністю (в значенні теорії складності) загального алгоритму синтаксичного аналізу зображень, які задаються за допомогою загальних структурних конструкцій. Ефективний за часом розв'язок задачі розпізнавання зазвичай досягається в таких двох випадках:

- використовуються сторонні (не пов'язані з структурними конструкціями, часто евристистичні) міркування, на основі яких на першому кроці розпізнавання визначається розташування на зображенні всіх елеметарних фрагментів, частин, з яких воно складається. В подальшому використовуються алгоритми розв'язку задачі на точний збіг для відшукання правил, що пов'язують ці фрагменти [2, 3, 5]. Так в роботі [3] на першому етапі визначається розташування та імена фрагментів, що позначають резистори, діоди, їх маркування і т.і. І лише на другому етапі за допомогою алгоритмів синтаксичного аналізу визначаються елементи електричної схеми як ціле – визначається, яке маркування та який номінал якому графічному позначенню повинні бути присвоєні. Роботи такого характеру мають один спільний недолік, пов'язаний з тим, що, взагалі кажучи, розташування та імена елементарних фрагментів можуть бути визначені лише на основі виводу всього зображення. Це означає, що якість розпізнавання багато в чому залежить від першого етапу – неправильне визначення розташування, розмірів чи імен елементарних фрагментів зображення унеможливлює правильне розпізнавання зображення вцілому;
- використовується окремий, лінійний за часом синтаксичного аналізу підклас загальних структурних конструкцій, а саме той, що відповідає т.з. регулярним граматикам та мовам. У цьому випадку, зазвичай, ставиться та вирішується задача на найкращий збіг [7, 4].

  Алгоритм, що використовується для розв'язку цієї задачі, є окремим випадком алгоритму синтаксичного аналізу в загальних структурних конструкціях. Постановка задачі розпізнавання як задачі на найкращий збіг є характерною поки що лише для зображень, які описуються регулярними мовами, та й то, в основному для тих з них, де простішими евристичними методами досягти прийнятних прикладних результатів неможливо. На даний момент найпопулярнішими задачами такого типу є задачі розпізнавання рукописних текстів (див., напр., [8]).

Для розв'язання задач синтаксичного аналізу у вказаних двох окремих випадках використовуються все ж дещо різні алгоритми. Як вже було сказано, для задач, що описуються регулярними мовами та граматиками, використовується алгоритм, який є окремим випадком алгоритму синтаксичного аналізу в загальних структурних конструкціях. Він полягає в наступному. Всі можливі фрагменти поля зору впорядковуються за їх розмірами. Переглядається кожен фрагмент, починаючи з найменшого, і для нього вирішується, чи може він бути побудований за правилами граматики з менших фрагментів (які на цей момент вже оброблені) і, якщо так, знаходиться його найдешевший вивід. Таким самим чином, насамкінець, знаходиться вивід всього зображення як найбільшого фрагмента. Алгоритми з таким принципом роботи називатимемо **розділяючими**.

Недоліком розділяючих алгоритмів є те, що в процесі їх роботи розглядається значна кількість "зайвих" фрагментів – таких, які взагалі не можуть бути побудовані в даній граматиці, та таких, про які, виходячи з певних евристичних міркувань, можна було б одразу з упевненістю сказати, що вони не увійдуть у вивід всього зображення.

Алгоритми, що застосовуються для розв'язку задачі на точний збіг, позбавлені цього недоліку за рахунок того, що мають наступну структуру. На першому етапі розпізнавання про певні фрагменти поля зору приймається кінцеве рішення щодо їх найменувань. Називатимемо ці фрагменти початковими. Далі алгоритм будується так, щоб уникнути перегляду тих фрагментів, які не можуть бути виведені з початкових: створюється перелік всіх фрагментів, які беруть участь у виводі зображення, і в цей перелік одразу ж записуються всі початкові фрагменти. Далі для всіх пар фрагментів, що задовольняють правилам граматики, генерується фрагмент – результат їх об'єднання, який додається до згаданого переліку. Якщо у процесі генерування створюється фрагмент, що відповідає власне всьому зображенню, то приймається рішення про те, що зображення виводиться в заданій граматиці. Алгоритми цього типу називатимемо **генеративними**. Ускладнення алгоритму в порівнянні з попереднім випадком полягає у пошуку пар фрагментів, які можуть бути об'єднані, та у відповіді на запитання на кожному кроці алгоритму, чи є вже в поточному переліку щойно утворений фрагмент.

У випадку, якщо прикладна задача не може бути описана за допомогою регулярних граматик, формулювання задачі розпізнавання як задачі на точний збіг мотивується перш за все нижчою обчислювальною складністю генеративних алгоритмів. Дійсно, розділяючі алгоритми синтаксичного аналізу в нерегулярних граматиках є обчислювально надто складними. В нашій роботі запропоновано побудувати алгоритм генеративного типу для розв'язку задачі на найкращий збіг. При цьому такий алгоритм має наступні переваги. На відміну від генеративних алгоритмів розв'язку задачі на точний збіг, в нашому алгоритмі на першому етапі щодо імен та положень початкових фрагментів не приймається остаточне рішення, а розглядається певна множина можливих варіантів їх розташування. Причому до цієї множини входять не всі можливі початкові фрагменти, як у випадку розв'язку задачі на найкращий збіг розділяючими алгоритмами, а тільки ті, які залишилися після видалення із множини всіх можливих початкових фрагментів такої її підмножини, фрагменти з якої точно не увійдуть до найкращого виводу вхідного зображення.

Виграш у обчислювальній складності в порівнянні з розділяючими алгоритмами досягається за рахунок значного зменшення кількості фрагментів – як видалені початкові фрагменти, так і всі фрагменти, які можуть бути з них одержані, не розглядаються надалі в процесі виводу.

Робота складається із семи розділів. В наступному розділі визначена двовимірна контекстно-вільна граматика, в третьому розділі наведена формальна постановка задачі на найкращий збіг. Четвертий розділ присвячений вибору локально-адитивної функції відмінності для введеної граматики. В п'ятому та шостому розділах приведений алгоритм розв'язання поставленої задачі та оцінка його складності. У заключному розділі подаються результати застосування запропонованого алгоритму до розпізнавання будівельних планів.

2 Визначення граматики

Будемо називати **полем зору** множину $T(n,m)=\left\{(i,j)\middle|\ i=\overline{1,n}, j=\overline{1,m}\right\}$ – підмножину двовимірної цілочисельної решітки. Величини n і m називатимемо **висотою** та **шириною** поля зору. Елементами поля зору є пікселі.

Множиною сигналів $X = \{0,1\}$ називатимемо множину кольорів, причому 0 відповідає білому кольору, а 1 – чорному.

Зображення x визначимо як функцію $x : T(n,m) \to X$, яка кожному пікселу поля зору ставить у відповідність певний колір. **Висотою** та **шириною** зображення, визначеного на даному полі зору, називатимемо висоту і ширину поля зору. Через X^T позначимо множину всіх можливих зображень.

Надалі розглядатимемо зображення двох типів: **вхідні зображення** – це зображення, які необхідно розпізнавати (наприклад, будівельні плани), та **еталонні зображення** чи **шаблони**, що задають еталонний вигляд елементарних частин, з яких складається зображення в цілому. Прикладами таких елементарних частин є шаблони дверей, вікон, сантехніки тощо. Висота та ширина шаблонів є значно меншою, аніж висота та ширина вхідного зображення. Кожному шаблону поставлена у відповідність певна його точка, що використовується для його зручної прив'язки на полі зору більшого розміру. Цю **точку** називатимемо **вказівною**. Щоб розрізняти ці два типи зображень, для позначення вхідних зображень використовуватимемо літеру x, а еталонні зображення позначатимемо літерою z.

Будемо називати **фрагментом поля зору** його прямокутну підмножину

$$\Pi_{h,w}^{i_1,j_1} = \left\{(i,j) \mid i = \overline{i_1, i_1 + h},\ j = \overline{j_1, j_1 + w},\ i_1, j_1 \geq 1,\ h, w \geq 0,\ i_1 + h \leq n,\ j_1 + w \leq m\right\}.$$

Обмеження зображення x на фрагмент поля зору Π називатимемо **фрагментом зображення** і позначатимемо x_Π. **Висотою** та **шириною** фрагмента зображення вважатимемо висоту h та ширину w фрагмента поля зору, на якому він визначений.

Для заданого еталонного зображення z та фрагмента вхідного зображення x_Π того ж розміру визначимо **локальну функцію відмінності** w, яка приймає значення в множині дійсних чисел. Величина $w(z, x_\Pi)$ показує, наскільки шаблон z відрізняється від фрагмента зображення x_Π.

Нехай задана деяка множина імен K, які слугують для іменування фрагментів зображення. Нехай на фрагменті зображення x_Π вказана певна його точка $FP \in \Pi$, яку зручно використовувати в якості центру локальної, пов'язаної з цим фрагментом системи координат. Фрагмент зображення x_Π з іменем $A \in K$ і виділеною точкою $FP \in \Pi$ називатимемо **сегментом** і позначатимемо $S_A = \langle x_\Pi, A, FP \rangle$. Ім'я A називатимемо **іменем сегмента**, а точку FP будемо називати **вказівною**.

Двовимірною контекстно-вільною граматикою з перекриттям сегментів G називається четвірка $\langle V, K, P, \varepsilon \rangle$, де V – множина терміналів, K – множина нетерміналів, P – множина правил, ε – аксіома.

Терміналами (елементами множини V) є імена шаблонів елементарних частин, з яких складається зображення вцілому. Позначатимемо ці імена маленькими буквами грецького алфавіту: $\alpha, \beta, \gamma \in V$, а самі шаблони через $z_\alpha, z_\beta, z_\gamma$.

Нетермінали (елементи множини K) – це всі можливі імена сегментів зображення, які використовуються в процесі синтаксичного аналізу зображення. Будемо позначати їх великими літерами латинського алфавіту: $A, B, C \in K$.

Аксіома ε – це таке ім'я-нетермінал, яким ми хочемо в результаті розпізнавання назвати все вхідне зображення.

Правила із множини P визначають способи побудови зображення із шаблонів. Правила бувають таких трьох типів: заміни, перейменування та конкатенації.

Правила заміни задаються наступним чином: $A \to \alpha, \quad A \in K, \alpha \in V.$

Нехай z_α – еталонне зображення висоти h і ширини w із вказівною точкою $FP(\alpha)$, $\Pi_{h,w}$ – фрагмент поля зору того ж розміру, що і шаблон z_α, x_Π – заданий на $\Pi_{h,w}$ фрагмент вхідного зображення x. Застосування правила заміни до пари (z_α, x_Π) означає утворення сегмента $S_A = \langle z_\alpha(\Pi), A, FP(\alpha) \rangle$, фрагмент зображення якого $z_\alpha(\Pi)$ є зображенням шаблону z_α на фрагменті поля зору Π. За це сплачується штраф $w(z_\alpha, x_\Pi)$. Сегменти, утворені за правилами заміни, будемо називати **первинними**.

Правила перейменування мають наступний вигляд: $A \to B, \quad A, B \in K.$

Таке правило означає, що будь-який сегмент з іменем B може бути перейменований на сегмент з іменем A.

Правила конкатенації задають способи утворення нового сегмента зображення із існуючих двох сегментів і мають такий узагальнений вигляд:

$$A \to B, C, restriction, FPrule, \qquad A, B, C \in K.$$

Правило конкатенації вказує, як з сегментів S_B та S_C з іменами B та C може бути утворений сегмент S_A з іменем A. Параметр *restriction* – це набір чисел, який задає обмеження на взаємне розташування сегментів S_B і S_C на полі зору для того, щоб до них могло бути застосоване дане правило. Фрагмент зображення $x_{\Pi}(S_A)$ визначається як найменший з фрагментів, що містить фрагменти $x_{\Pi}(S_B)$ та $x_{\Pi}(S_C)$. Параметр *FPrule* задає правило, згідно якого визначається положення вказівної точки $FP(S_A)$ нового сегмента S_A на основі положень вказівних точок $FP(S_B)$ та $FP(S_C)$ сегментів S_B та S_C.

Для того, щоб за правилами граматики можна було будувати зображення, серед них обов'язково має бути хоча б одне правило, в лівій частині якого стоїть ім'я аксіоми ε.

Опишемо проілюстровану на рис. 1 процедуру побудови зображення в граматиці G за цими правилами.

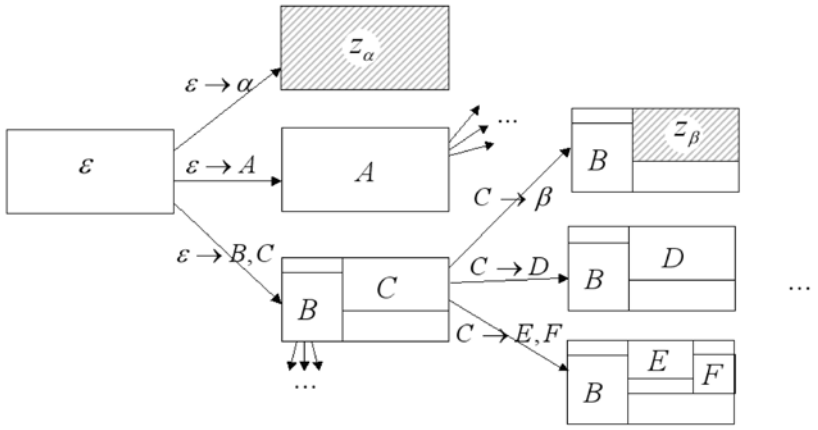


Рис. 1: Приклад побудови зображення в граматиці: прямокутниками позначені сегменти зображення, стрілками – правила, які застосовуються до цих сегментів.

Спочатку на всьому полі зору заданий сегмент S_ε з іменем аксіоми ε. Шукається правило, в якому це ім'я стоїть в лівій частині. Тут можливі такі варіанти:

1. знайдене правило заміни $\varepsilon \to \alpha$. Тоді в фрагменті зображення сегменту S_ε малюється шаблон z_α, ім'я якого стоїть в правій частині правила;
2. знайдене правило перейменування $\varepsilon \to A$. Тоді ім'я сегмента замінюється на нетермінал A, який стоїть в правій частині правила;
3. знайдене правило конкатенації $\varepsilon \to B, C, restriction, FPrule.$ Тоді в сегменті S_ε за даним правилом згідно параметрів *restriction* та *FPrule* виділяються два менших сегменти з іменами B та C.

В результаті виконання одного з описаних сценаріїв утворюється або намальоване зображення (виконалося правило заміни), і на цьому процес побудови закінчується, або один чи два сегменти з іменами-нетерміналами. І поки на зображенні залишаються сегменти з іменами-нетерміналами, для них повторюється описана процедура. Після цього в усі піксели зображення, які не були заповнені еталонними зображеннями, записується 0 – білий колір.

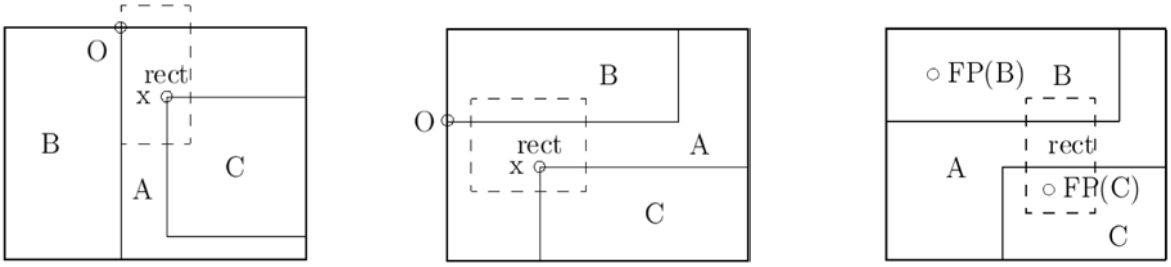
Програмна реалізація містить такі три види задання взаємного розташуванння сегментів в правилах конкатенації:

1. **Правила горизонтальної конкатенації** мають такий вигляд:

   $$A \to B \mid C, rect, FPrule, \qquad A, B, C \in K.$$

   Параметр *rect* є четвіркою чисел, яка задає прямокутник в системі координат із центром в правому верхньому куті сегмента S_B (точка O на рис. 2(а)). В цей прямокутник має потрапити верхній лівий

кут сегмента S_C (точка x на рис. 2(а)) для того, щоб до сегментів S_B і S_C могло бути застосоване дане правило.



(а) (б) (в)

Рис. 2: Приклад застосування правил конкатенації: (а) – горизонтальної, (б) – вертикальної, (в) – відносно вказівної точки.

2. **Правила вертикальної конкатенації** задаються наступним чином:

$$A \to B/C, rect, FPrule, \qquad A, B, C \in K.$$

Параметр $rect$ є четвіркою чисел, яка задає прямокутник в системі координат із центром в лівому нижньому куті сегмента S_B (точка O на рис. 2(б)). В цей прямокутник має потрапити верхній лівий кут сегмента S_C (точка x на рис. 2(б)) для того, щоб до сегментів S_B і S_C могло бути застосоване дане правило.

3. **Правила конкатенації відносно вказівної точки** мають виглад

$$A \to B + C, rect, FPrule, \qquad A, B, C \in K.$$

Тут параметр $rect$ задає прямокутний фрагмент поля зору в системі координат із центром у вказівній точці $FP(S_B)$, в який має потрапити вказівна точка $FP(S_C)$ сегмента S_C (рис. 2(в)).

Ми розглядаємо лише такі правила, які, окрім введених означень, задовольняють ще 2 умови:

1. в результаті застосування правила утворюється сегмент, який не збігається ні з одним із сегментів, що його утворили: він або має більший розмір, або інше ім'я. Це в подальшому забезпечить скінчений час роботи алгоритму синтаксичного аналізу;
2. сегменти не перетинаються своїми чорними пікселами (рис. 3), тобто в області перетину під чорними пікселами одного сегмента знаходяться тільки білі пікселі іншого, а під білими пікселами можуть бути як білі, так і чорні.

Можливість сегментів перетинатися випливає з того, що на координати прямокутника $rect$, який задає їх взаємне розташування, не накладаються жодні обмеження.

Оскільки в правилах конкатенації приймають участь сегменти, які на даний момент вже утворені в граматиці G, то зображення в них на одному й тому самому фрагменті поля зору (області перетину) можуть бути різними. Це випливає з того, що кожен сегмент утворюється за правилами конкатенації та перейменування із первинних сегментів, тож зображення в ньому визначається зображеннями шаблонів, які відповідають цим первинним сегментам, а не фрагментом вхідного зображення.

Рис. 3: Приклад неправильного перекриття сегментів: множини чорних пікселів сегментів перетинаються.

Будемо називати **мовою граматики** G множину всіх зображень e, для яких існує послідовність використання правил граматики, результатом якої є присвоєння імені ε всьому зображенню. Ця

послідовність правил називається **виводом** зображення e в граматиці G. Для мови граматики G введемо позначення $L(G)$.

Кожному правилу із виводу зображення e відповідає сегмент, який утворився за цим правилом. Сегменти, що утворилися за правилами заміни, є первинними.

3 Постановка задачі

Нехай задане вхідне зображення x та деяке зображення e з мови граматики G. Виберемо всі первинні сегменти $s_1, s_2, ..., s_k$, які відповідають виводу зображення e. Кожному первинному сегменту s_i поставимо у відповідність пару "еталонне зображення z_i – фрагмент вхідного зображення x_{Π_i}", до якої було застосоване правило заміни при утворенні цього сегменту. Як вже було сказано, для такої пари визначена локальна функція відмінності $w(z_i, x_{\Pi_i})$, яка показує, наскільки відрізняється шаблон z_i від фрагмента зображення x_{Π_i}. Значення $w(z_i, x_{\Pi_i})$ є штрафом за утворення сегмента s_i.

Назвемо **функцією відмінності** функцію $f_G : L(G) \times X^T \to R$ таку, що величина $f_G(e, x)$ показує, наскільки зображення e відрізняється від зображення x. Функцію відмінності f_G будемо називати **локально-адитивною** по відношенню до локальної функції відмінності w, якщо f_G може бути представлена у вигляді

$$f_G(e, x) = \sum_{i=1}^{k} w(z_i, x_{\Pi_i}). \tag{1}$$

Із врахуванням введених означень задача розпізнавання зображення x може бути сформульована таким чином:

Задача 1 *Задане вхідне зображення x, двовимірна контекстно-вільна граматика з перекриттям сегментів G та локально-адитивна функція відмінності $f_G : L(G) \times X^T \to R$. Необхідно знайти зображення e з мови граматики G, яке найменше відрізняється від вхідного зображення x:*

$$e^* = \operatorname*{argmin}_{e \in L(G)} f_G(e, x). \tag{2}$$

4 Побудова локально-адитивної функції відмінності

Конкретизація задачі полягає у вказанні локально-адитивної функції відмінності f_G. Для цього функцію відмінності f_G оберемо таку, яка дорівнює кількості пікселів у вхідному зображенні x, які не збігаються з пікселами зображення e:

$$f_G(e, x) = \sum_{t \in T} \left| e(t) - x(t) \right|, \tag{3}$$

де $e(t)$ та $x(t)$ – значення вхідного зображення x та зображення e з мови граматики в пікселі t відповідно.

Якби правила граматики G не допускали перекриття сегментів, і зображення e з мови граматики G було побудоване із первинних сегментів, які повністю покривають поле зору без перетинів, то природно було б взяти локальну функцію відмінності w таку, яка дорівнює кількості пікселів у фрагменті x_Π, які не збігаються з пікселами шаблона z:

$$w(z, x_\Pi) = \sum_{t \in T(z)} \left| x_\Pi(t) - z(t) \right|, \tag{4}$$

де $T(z)$ означає множину пікселів шаблона z, $x_\Pi(t)$ та $z(t)$ – значення фрагмента x_Π та шаблона z в пікселі t відповідно.

Очевидно, що в цьому разі функція відмінності f_G, задана формулою (3), була би локально-адитивною по відношенню до локальної функції відмінності w, заданої виразом (4).

Та для описаної нами двовимірної граматики G правила конкатенації введені таким чином, що сегменти в процесі побудови зображення можуть перекриватись, але так, що множини їх чорних пікселів не перетинаються. І в цьому разі зрозуміло, що введена функція відмінності f_G (3) не буде локально-адитивною, адже області перетину сегментів будуть враховані двічі.

Для того, щоб вибрати таку локальну функцію відмінності, щодо якої деяка функція відмінності буде локально-адитивною у випадку перекриття сегментів, виконаємо певні перетворення функції (4), позначивши через $T_b(z)$ множину чорних пікселів еталонного зображення z:

$$\begin{aligned}\sum_{t\in T(z)}\left|x_{\Pi}(t)-z(t)\right| &= \sum_{t\in T(z)}\left(x_{\Pi}(t)-z(t)\right)^2 = \\ &= \sum_{t\in T(z)} x_{\Pi}{}^2(t) - 2\sum_{t\in T(z)} x_{\Pi}(t)\cdot z(t) + \sum_{t\in T(z)} z^2(t) = \\ &= \sum_{t\in T(z)} x_{\Pi}(t) - 2\sum_{t\in T_b(z)} x_{\Pi}(t) + \sum_{t\in T(z)} z(t).\end{aligned} \tag{5}$$

Тепер припустимо, що зображення e з мови граматики G побудоване з первинних сегментів $s_1, s_2, ..., s_k$, що не перетинаються, але не обов'язково мають прямокутну форму. І всі ці сегменти, щільно прилягаючи один до одного, повністю покривають поле зору, на якому визначене зображення e. В такому разі очевидно, що функція відмінності (3) також є локально-адитивною по відношенню до локальної функції (4).

Для розв'язку задачі (1) нам необхідне не саме значення функції f_G в точці мінімуму, а значення її аргумента. Розпишемо f_G як суму двох доданків:

$$f_G(e,x) = \sum_{i=1}^{k} w(z_i, x_{\Pi_i}) = \sum_{i=1}^{k}\sum_{t\in T(z_i)} x_{\Pi_i}(t) + \sum_{i=1}^{k}\left(-2\sum_{t\in T_b(z_i)} x_{\Pi_i}(t) + \sum_{t\in T(z_i)} z_i(t)\right).$$

Перший доданок означає суму значень сигналів по всіх фрагментах вхідного зображення, що відповідають первинним сегментам зображення e, а відтак за умови виконання введених припущень дорівнює сумі значень сигналів по всьому вхідному зображенню x. Ця величина залежить лише від зображення x, не залежить від виводу зображення e, а отже не впливає на розв'язок задачі (1):

$$\begin{aligned}e^* &= \operatorname*{argmin}_{e\in L(G)} f_G(e,x) = \\ &= \operatorname*{argmin}_{e\in L(G)}\left(\sum_{i=1}^{k}\sum_{t\in T(z_i)} x_{\Pi_i}(t) + \sum_{i=1}^{k}\left(-2\sum_{t\in T_b(z_i)} x_{\Pi_i}(t) + \sum_{t\in T(z_i)} z_i(t)\right)\right) = \\ &= \operatorname*{argmin}_{e\in L(G)} \sum_{i=1}^{k}\left(-2\sum_{t\in T_b(z_i)} x_{\Pi_i}(t) + \sum_{t\in T(z_i)} z_i(t)\right).\end{aligned} \tag{6}$$

Отже ми можемо вибрати іншу функцію відмінності $\tilde{f}_G(e,x)$, яка випливає з виразу (6):

$$\tilde{f}_G(e,x) = \sum_{i=1}^{k}\left(-2\sum_{t\in T_b(z_i)} x_{\Pi_i}(t) + \sum_{t\in T(z_i)} z_i(t)\right). \tag{7}$$

При цьому за будь-якого вхідного зображення x та граматики G розв'язок задачі (1) з функцією $\tilde{f}_G(e,x)$ буде тим самим, що і з функцією $f_G(e,x)$. В цьому значенні функції $\tilde{f}_G(e,x)$ та $f_G(e,x)$ є еквівалентними. І з вигляду функції відмінності $\tilde{f}_G(e,x)$ очевидно випливає, що вона буде локально-адитивною по відношенню до такої локальної функції $\tilde{w}$:

$$\tilde{w}(z, x_{\Pi}) = \sum_{t\in T(z)} z(t) - 2\sum_{t\in T_b(z)} x_{\Pi}(t), \tag{8}$$

значення якої залежить тільки від кількості чорних пікселів шаблона z та кількості чорних пікселів, що збіглися в шаблоні z та фрагменті зображення x_{Π}.

Тепер покажемо, що функція відмінності $\tilde{f}_G$ буде також локально-адитивною по відношенню до локальної функції відмінності $\tilde{w}$ навіть тоді, коли граматика допускає перекриття сегментів.

Оберемо деякий первинний сегмент s зображення e та розглянемо пару (z, x_{Π}), з якої цей сегмент був утворений. Шаблон z визначений на фрагменті поля зору того ж розміру і форми (за нашим припущенням не обов'язково прямокутної), як і фрагмент Π, на якому заданий фрагмент зображення x_{Π}. Визначимо

фрагмент поля зору Π' як найменший прямокутний фрагмент, що включає Π, $\Pi \subseteq \Pi'$, і на ньому фрагмент зображення $x_{\Pi'}$, $x_{\Pi} \subseteq x_{\Pi'}$. Відповідно до розмірів Π' побудуємо шаблон z', піксели якого мають такий же колір як і піксели шаблона z там, де z визначений, і білий колір в усіх інших пікселах. Очевидно, що кількість чорних пікселів шаблонів z і z' однакова, так само як і кількість чорних пікселів, що збіглися в шаблоні z і фрагменті зображення x_{Π}, та z' і $x_{\Pi'}$ відповідно. А відтак за формулою (8) значення локальної функції відмінності $\tilde{w}$ не змінилося:

$$\begin{cases} \sum\limits_{t \in T(z)} z(t) = \sum\limits_{t \in T(z')} z'(t) \\ \sum\limits_{t \in T_b(z)} x_{\Pi}(t) = \sum\limits_{t \in T_b(z')} x_{\Pi'}(t) \end{cases} \quad \Rightarrow \quad \tilde{w}(z, x_{\Pi}) = \tilde{w}(z', x_{\Pi'}). \tag{9}$$

Тож штраф за утворення прямокутного сегмента s', який відповідає парі $(z', x_{\Pi'})$, такий самий, як і штраф за утворення сегмента s.

Якщо застосувати описану процедуру до всіх первинних сегментів $s_1, s_2, ..., s_k$ зображення e, то отримаємо множину прямокутних сегментів $s'_1, s'_2, ..., s'_k$, що перекриваються, але в області перекриття вони не будуть перетинатися по чорних пікселах, бо зображення в первинному сегменті задається шаблоном, а до кожного шаблону додалися лише білі піксели. Це повністю відповідає вимозі до сегментів при застосуванні правил конкатенації граматики G. Із виразів (7), (8) та (9) випливає, що

$$\tilde{f}_G(e, x) = \sum_{i=1}^{k} \tilde{w}(z_i, x_{\Pi_i}) = \sum_{i=1}^{k} \tilde{w}(z'_i, x_{\Pi'_i}),$$

тобто функція відмінності $\tilde{f}_G(e, x)$ є локально-адитивною по відношенню до локальної функції відмінності $\tilde{w}$, заданої формулою (8), за умови, що сегменти, які перекриваються, в області перекриття не перетинаються по чорних пікселах.

5 Алгоритм розв'язання задачі

В цьому розділі ми розглянемо генеративний алгоритм розв'язання задачі, сформульованої в третьому розділі. Опишемо цей алгоритм, зображений на рис. 4 у вигляді псевдокоду:

1. Нехай S – вихідна множина первинних сегментів на зображенні, отриманих в результаті застосування правил заміни до шаблонів та фрагментів вхідного зображення. Впорядкуємо сегменти з множини S за їх розміром і послідовно їх оброблятимемо. Цей пункт не відображений в програмі на рис. 4.
2. Нехай s_i – обраний на поточному кроці сегмент. Позначимо через $s_i.name$ ім'я цього сегмента.
   (а) Знаходимо в множині правил P такі правила перейменування, в правій частині яких стоїть ім'я сегмента $s_i.name$. За кожним знайденим правилом p утворюємо новий сегмент s_{new}, який збігається із s_i, але має нове ім'я згідно цього правила (в програмі воно позначено $p.left$). Штраф за утворення нового сегмента ($s_{new}.penalty$) покладаємо рівним штрафу за утворення сегмента s_i.

   Далі виконуємо **процедуру вставки** новоутвореного сегмента s_{new}, яка зображена на рис. 5 і полягає в наступному. Перевіряємо, чи існує вже в множині S сегмент s_{new}. Якщо такий сегмент існує (знайшовся $s_{exists} = s_{new}$), то у випадку, коли новий сегмент має менший штраф, ми замінюємо ним вже існуючий сегмент. Якщо ж сегмент s_{exists} не існує, то вставляємо s_{new} в множину S згідно з його розміром.

   (б) Знаходимо в множині правил P такі правила конкатенації, в яких ім'я сегмента $s_i.name$ входить в праву частину.

   Послідовно оброблюємо ці правила. Знаходимо в множині S всі сегменти s_j, менші за s_i, які можуть бути об'єднані з сегментом s_i за поточним правилом p, тобто мають відповідне ім'я, і їхнє положення задовольняє параметру $p.rect$ цього правила. Позначення $s_j.point$ означає ту

точку, яка має потрапити до прямокутника $p.rect$, тобто лівий верхній кут сегмента s_j для правил горизонтальної та вертикальної конкатенації, або вказівну точку для правил конкатенації відносно вказівної точки.

Послідовно оброблюємо ці сегменти $\{s_j\}$: згідно поточного правила p утворюємо новий сегмент s_{new} із сегментів s_i та s_j. Обчислюємо штраф за вивід цього сегмента як суму штрафів за вивід сегментів s_i та s_j та проводимо процедуру вставки сегмента s_{new}, описану вище.

3. Впорядкованість множини S за розміром забезпечує, що новоутворений сегмент буде вставлений у ще не оброблену частину множини S, адже його розмір не може бути менший за розміри сегментів, які його утворили.

Умови, що накладаються на правила граматики, гарантують нам, що не може бути утворений сегмент, який повністю збігається з одним із сегментів, які його утворили. А це в свою чергу гарантує скінченість множини сегментів S, адже множина K імен сегментів скінчена, так само як і множина фрагментів поля зору.

Перебравши таким чином всі сегменти із множини S, ми утворили всі можливі сегменти в даній граматиці при заданій множині первинних сегментів. Знайдемо в множині S всі сегменти з іменем ε і серед них такий сегмент s^*, який має найменший штраф. Розв'язком задачі (1) є фрагмент зображення, що відповідає сегменту s^*.

```
for(i = 0; i < |S|; i++)
{
    for(p ∈ P, p.type = rename, p.right = s_i.name)
    {
        s_new := s_i;
        s_new.name := p.left;
        s_new.penalty := s_i.penalty;
        insert(s_new);
    }
    for(p ∈ P, p.type = concat, p.right1 = s_i.name)
    {
        знайти всі s_j ∈ S такі, що j < i, s_j.name = p.right2, s_j.point ∈ p.rect;
        for {s_j}
        {
            s_new := new(p, s_i, s_j);
            s_new.penalty := s_i.penalty + s_j.penalty;
            insert(s_new);
        }
    }
}
знайти всі s_r ∈ S такі, що s_r.name = ε;
s*.penalty = ∞;
for {s_r}
    if (s_r.penalty < s*.penalty)  s* := s_r;
```

Рис. 4: Алгоритм розв'язання задачі.

знайти в S сегмент s_{exists} такий, що $s_{exists} = s_{new}$;

if (s_{exists} існує)

if ($s_{new}.penalty < s_{exists}.penalty$)

$s_{exists} := s_{new}$;

else

вставити s_{new} в множину S.

Рис. 5: Процедура вставки сегмента $insert(s_{new})$.

6 Аналіз складності алгоритму

Оцінимо обчислювальну складність наведеного алгоритму. Необхідно обробити $|S|$ сегментів. Витрати на обробку одного сегмента складаються із часу T_1 на пошук сегментів, які можуть бути поєднані з даним, та часу T_2 на вставку новоутвореного сегмента. Позначимо через $N(s)$ кількість сегментів, які можуть бути поєднані з сегментом s. У введених позначеннях складність алгоритму може бути записана як $\sum_{s \in S} \left(T_1 + N(s)T_2\right)$. T_1 є часом пошуку сегментів, вказівна точка чи лівий верхній кут яких потрапляє до фрагмента поля зору, що визначається сегментом s та правилами граматики. Тоді T_1 є часом на регіональний пошук у двовимірному просторі, і, як показано в [9], ця величина може бути оцінена як $T_1 = O\left(\log|S|\right)$. Час T_2 має ту саму оцінку складності. З цього випливає, що складність алгоритму може бути оцінена як $O\left(|S|N(s)\log|S|\right)$. Якщо припустити, що величина $N(s)$ найчастіше є порівняно невеликою (один чи два), тоді складність всього алгоритму можна оцінити як $O\left(|S|\log|S|\right)$.

Обчислимо складність наведеного алгоритму для задачі на найкращу відповідність у випадку граматики, в якій конкатенація розуміється таким чином, що два фрагменти можуть бути поєднані, тільки якщо вони мають однакову висоту (ширину) і впритул прилягають один до одного по горизонталі (вертикалі).

Для зображення $n \times m$ пікселів кількість всіх можливих сегментів зображення $|S| = n^2m^2$, а кількість сегментів $N(s)$, які можуть бути поєднані з даним, асимптотично рівна $(n+m)$. Тож складність нашого алгоритму для цієї задачі рівна $O\left(n^2m^2(n+m)\log(nm)\right)$. За допомогою наведеного в роботі [6] алгоритма Кока-Янгера-Касамі ця задача розв'язується за час $O\left(n^2m^2(n+m)\right)$. Отже, в цьому випадку наш алгоритм повільніший на $\log(nm)$.

Однак у випадку, коли потужність $|S|$ множини сегментів, які оброблюються алгоритмом, є значно меншою за потужність множини всіх можливих сегментів на зображенні, наш алгоритм є ефективнішим за алгоритм Кока-Янгера-Касамі. Значне зменшення множини S може бути досягнуте за рахунок того, що на зображенні попередньо можна виділити області, такі, що сегменти з певними іменами, які знаходяться в цих областях, точно не увійдуть до найкращого зображення e^*, яке є розв'язком задачі (1). Так, на зображеннях будівельних планів нескладно визначити такі дві групи областей:

1. області, на яких зображено стіни кімнат, і в них не може бути сегментів з іменами, які позначають сантехнічне обладнання або меблі;
2. області всередині кімнат, і в них немає сенсу утворювати первинні сегменти, які позначають вікна, двері тощо.

Тож до вихідної множини первинних сегментів доцільно включити лише сегменти з першої області, які позначають фрагменти стін, та сегменти з другої області, які задають внутрішнє обладнання кімнат. А сукупна кількість таких сегментів є значно меншою за кількість всіх можливих сегментів на зображенні: з усіма можливими іменами, положеннями сегментів та їх вказівних точок. За рахунок цього виграш від

застосування саме нашого алгоритму може бути дуже суттєвим, що і буде продемонстровано в наступному розділі.

7 Демонстрація результатів розпізнавання креслень будівельних планів

Експериментальна перевірка розробленого методу мала на меті верифікацію основних ідей, на яких базується розроблений алгоритм. Експерименти виконувались на штучних та реальних планах житлових приміщень, в яких всі кімнати мають прямокутну форму. Перед процедурою розпізнавання виконувалася попередня обробка вхідних зображень, яка полягає у виділенні на плані областей, на яких зображено стіни приміщень, та областей, які задають прямокутні зони всередині кімнат.

Приклади зображень, що розпізнавалися, наведені на рис. 6(а) та 7(а). Граматика, яка описує зображення квартир, у спрощеному вигляді може мати наступний вигляд:

$G = \langle V,K,P, \varepsilon\rangle$

$V = \{wall\_hor, wall\_vert, window\_hor, window\_vert, door\_hor, door\_vert, sink, closet, bath\}$

$K = \{Wall\_hor, Wall\_vert, Room2, Room3, Room, RoomCl, RoomS, Bathroom, Roomset\}$

$\varepsilon = Flat$

P : $Flat \rightarrow Roomset \mid Roomset$

$Roomset \rightarrow Roomset / Roomset$

$Roomset \rightarrow Roomset \mid Roomset$

$Roomset \rightarrow Room$

$Bathroom \rightarrow RoomCl + bath$, (0.1,0.1,0.8,0.8)

$RoomS \rightarrow RoomCl + sink$, (0.1,0.1,0.8,0.8)

$RoomCl \rightarrow Room + closet$, (0.1,0.1,0.8,0.8)

$Room \rightarrow Room3$

$Room \rightarrow Room2$

$Wall\_hor \rightarrow wall\_hor$

$Wall\_hor \rightarrow Wall\_hor \mid wall\_hor$

$Wall\_hor \rightarrow Wall\_hor \mid door\_hor$

$Wall\_hor \rightarrow Wall\_hor \mid window\_hor$

$Wall\_vert \rightarrow wall\_vert$

$Wall\_vert \rightarrow Wall\_vert / wall\_vert$

$Wall\_vert \rightarrow Wall\_vert / door\_vert$

$Wall\_vert \rightarrow Wall\_vert / window\_vert$

Множина терміналів V містить імена еталонних зображень структурних елементів плану квартири: стін (wall_hor, wall_vert), вікон (window_hor, window_vert), дверей (door_hor, door_vert), сантехніки (closet, sink, bath) тощо. Кожному терміналу відповідає еталонне зображення з таким іменем. Множина правил P містить три групи правил: перша група задає структуру плану квартири як множини кімнат та способи побудови кімнат із горизонтальних та вертикальних стін, друга – структуру окремої горизонтальної стіни і, нарешті, третя – структуру вертикальної стіни. Множина K містить всі нетермінали, які були використані при заданні множини правил P.

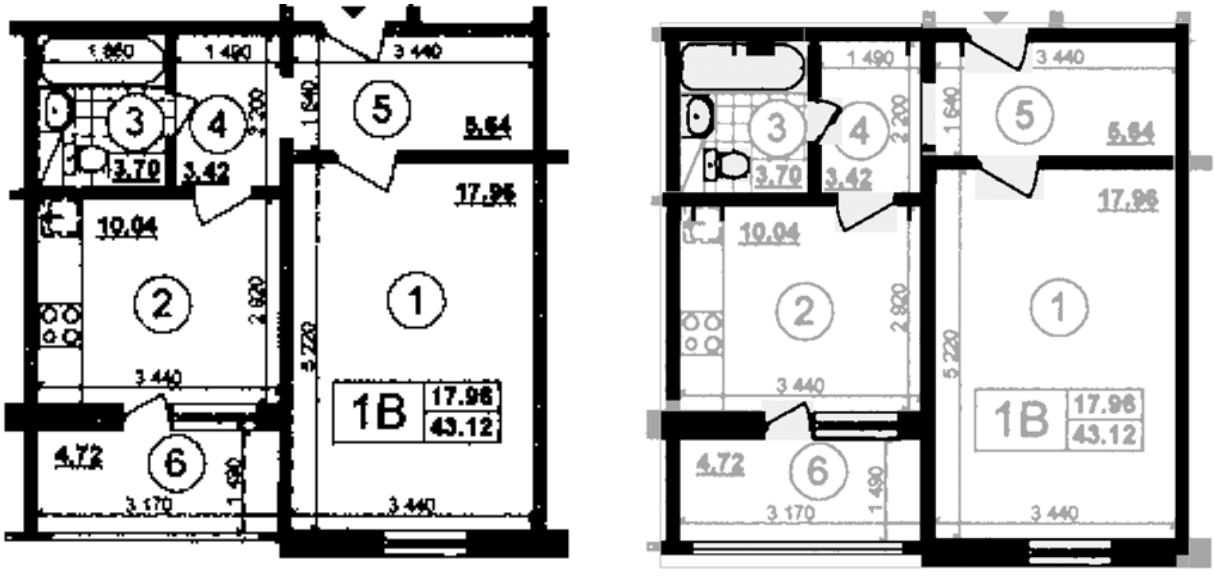
В результаті синтаксичного аналізу вхідних зображень з рис. 6(а) та 7(а) отримані зображення з мови граматики, показані на рис. 6(б), 7(б) чорним кольором (сірим кольором у якості підкладки зображені відповідні їх вхідні зображення). Як видно з цих рисунків, структура зображень була визначена правильно: вірно знайдено всі кімнати, всі стіни, без помилок розпізнано положення всіх дверей, вікон, сантехнічного обладнання тощо. Неточності, допущені при розпізнаванні (невеликі розриви між деякими стінами, неоднакова товщина стін вздовж одної кімнати) є результатом спрощення граматики і можуть бути усунуті застосуванням більш складної граматики чи інших шаблонів.

При розпізнаванні зображення з рис. 6(а) розміром 292×354 пікселі алгоритмом було оброблено протягом 49 секунд всього біля $2,5 \cdot 10^6$ сегментів замість більш ніж 10^{15} всіх можливих на цьому зображенні сегментів, а при розпізнаванні зображення з рис. 7(а) розміром 492×479 пікселів програма обробила відповідно $3 \cdot 10^6$ сегментів за 102 секунди при загальній кількості всіх можливих сегментів 10^{17}.

Отже застосування нашого алгоритму на цих прикладах дозволило отримати надзвичайно великий виграш у обчислювальній складності в порівнянні з розділяючими алгоритмами розв'язку задачі на найкращий збіг, які оброблюють всі можливі сегменти на зображенні, і за рахунок цього розпізнати зображення будівельних планів за прийнятний час.

(а) (б)

Рис. 6: (а) – Перший приклад плану, який розпізнавався. (б) – Зображення, отримане в результаті розпізнавання (чорного кольору), намальоване зверху зображення (а) (сірого кольору).



(а) (б)

Рис. 7: (а) – Другий приклад плану, який розпізнавався. (б) – Зображення, отримане в результаті розпізнавання (чорного кольору), намальоване зверху зображення (а) (сірого кольору).

Література

[1] Шлезінгер М.І., Савчинський Б.Д., Анохіна М.О. Синтаксичний аналіз та розпізнавання друкованих нотних текстів // Управляющие системы и машины, № 4. – 2003. – С. 30–38.
[2] Vladimir M. Kiyko. Recognition of objects in images of paper based line drawings // Third Intern. Conf. on Document Analysis and Recognition. – Monreal: 1995. – Pp. 970–973.
[3] Ivan Aksak, Volodymir Kijko, Viacheslav Matsello, Michail Schlesinger. One generalization of context-free grammars and its application to structural analysis of drawings // Праці. Третя Всеукраїнська міжн. конф. Оброблення сигналів і зображень та розпізнавання образів/ – Київ: 1996. – С. 137–140.
[4] Gary E. Kopec, Philip A. Chou, David A. Maltz. Markov source model for printed music decoding // IS&T/SPIE Int. Symposium on Electronic Imaging. – SPIE Proc. – San Jose: February 1995.
[5] Pascal Garcia, Bertrand Coüasnon. Using a genetic document recognition method for mathematical formulae recognition // IAPR Intern. Workshop on Graphics Recognition / Ed. by Y.-B. Kwon D. Blostein. – LNCS 2390. –Berlin, Heidelberg: Springer-Verlag, 2002. – Pp. 236–244.
[6] Michail I. Schlesinger, Vaclav Hlavač. Ten lectures on statistical and structural pattern recognition. – Dordrecht/Boston/London: Kluwer Academic Publishers, 2002. – P. 519.
[7] Ковалевский В.А. Оптимальный алгоритм распознавания некоторых последовательностей изображений // Кибернетика, № 4. – 1967.
[8] Rejean Plamodon, Sargur N. Srihari. On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey // IEEE Trans. on PAMI. – January 2000. – Vol. 22, no. 1. – Pp. 63–84.
[9] Препарата Ф., Шеймос М. Вычислительная геометрия: Введение. –Москва: Мир, 1989. – С. 478.