Recognition

Анализ на задачата от Веселин Георгиев

Задачата в пети кръг от състезанието на PC Magazine & Musala Soft беше от областта "Разпознаване на изображения" и имаше известен елемент на изкуствен интелект. Състезателят трябва да "опознае" скрита за него картинка, като разпитва web сървър, със серия от картинки по негов избор, като единственият отговор от сървъра е "разликата" (в стандартна квадратична метрика) на подадената картинка спрямо търсената. Картинките са черно-бели, в 256 степени на сивото, и с големини до 128x128. Броят питания не беше предварително известен, но се очакваше да е грубо около 500 на най-големите картинки.

Сред подходите за решения на задачата, често срещана е идеята да се разпитва сървъра за някакъв цял регион, като регионът се оцвети плътно с някакъв цвят, и чрез серия от въпроси да се намери оптималният цвят за този регион. Макар и нелоша, тази идея (включително и модификациите, които рекурсивно разцепват интересните региони), страда от недобро оценяване на локалната структура на региона. Например, този алгоритъм въобще няма да различи квадратче 8x8, запълнено със случаен шум (50% бели и 50% черни пиксели), от доста по-приятния случай на шахматно разположение с 4x4 бели/черни квадратчета.

Освен това, по условие, за картинките се знае, че ще са снимки на хора, и едно добро решение трябва да се възползва от този факт, като намира областите с висок контраст (очи, коса, контури и пр.) и концентрира разпитванията си в тези области, докато за неконтрастните части (заден фон, дреха без шарка, небе и т.н.) да използва някаква гладка интерполация.

Ето защо моето решение използва модификация на въпросите, която намира цвета на единичен пиксел чрез едно питане. Всъщност, аз открих и метод, с който се намират цветовете на два пиксела чрез единствено питане. Ето идеята за единични пиксели: съвсем в началото питаме сървъра с изцяло черна картинка, и резултатът запаметяваме като D_0. Когато питаме за единствен пиксел, просто оцветяваме пиксела с цвят 1 (останалите си остават 0) и резултатът от питането, D' вадим от D_0. Така имаме

$$D' - D_0 = 1 - 2x,$$

където x е цветът на неизвестния пиксел.

Модификацията за два пиксела едновременно е следната: оцветяваме първия пиксел с 1, а вторият с 255. Резултатът D'' отново вадим от D_0:

$$D'' - D_0 = 65026 - 2x - 510y,$$

където x, y са неизвестните цветове на, съответно, първия и втория пиксел. Т.е. имаме уравнението:

$$2x + 510y = K \text{ (const)}.$$

Очевидно, ако K не се дели на 510, може да решим цялото уравнение директно. Това е доста вероятно статистически, т.е. "хитрото" решение работи в над 98% от случаите.
Ако K се дели на 510, имаме двузначие (например, ако K = 510, не можем да разграничим случаите (x=255, y=0) и (x=0, y=1).
От статистически съображения трябва да хванем непременно два важни случая - когато и двата пиксела са бели или черни.

1) $K = 0$. Очевидно $x = y = 0$ (няма двузначност)
2) K = 130305 – тогава $x = y = 255$ (пак няма двузначност)

Ако нямаме късмет и случаят е истински двузначен, все пак положението не е толкова лошо: може да пуснем единично питане за единия пиксел, и чрез резултата можем да сметнем цвета на втория пиксел без допълнителен въпрос. Т.е. "хитрото" питане отнема максимум 2 въпроса за 2 пиксела, следователно трябва да го ползваме винаги :)

И така, със 500 питания може да разберем цветовете на почти 1000 пиксела. Оттук нататък в анализа ще приемам, че под "задавам *X* въпроса" се разбира "намирам цветовете на ~2*X* пиксела".

Тривиалният случай, когато картинката е достатъчно малка (< ~850 пиксела), може просто да питаме за всички пиксели и да изведем като резултат точният отговор. В противен случай прилагаме "пълното" решение (визуализация на работата на решението ми е поместена в страницата за пети кръг на сайта на конкурса – konkurs.musala.com – горещо ви препоръчвам да я видите). То е:

Започваме с равномерно питане за пиксели из цялата картинка. Избира се равномерен грид (например, през 8 пиксела по хоризонтала и вертикала) и се намират цветовете на всички пиксели от грида. Размерът му се избира така, че да изразходим около 190-250 питания за тази начална фаза (точната бройка зависи от размерите на картинката - за по-малка картинка ползваме по-малко питания). След като имаме този първоначален грид, можем да интерполираме между известните ни пиксели и така да построим едно грубо начално решение. Това е все едно да имаме една малка (напр. 40x40) картинка, която да up-scale-нем до 128x128 чрез популярен софтуер като *Photoshop*. Тук се натъкнах на неочакваното откритие, че изборът на метод за интерполиране е съществен! Реализирах bilinear, bicubic, и Sinc(Lanczos3) resampling, но последните два се представят по-лошо от bilinear (от гледна точка на оценката за грешка), въпреки, че *изглеждат* по-добре естетически! Оттук нататък ползвах bilinear навсякъде, където мога.

Добре, направили сме първоначалния грид: как да оценим къде са областите с висок контраст? Моето решение ползва числено приближение: дадена точка от грида може да се сравни със съседите си (по грид), и ако разликата е голяма, ще предполагаме, че точката се намира в интересен за решението ни регион. Оценката на тази разлика се прави чрез дължината на градиента – т.е. комбинацията от хоризонталната и вертикалната (средни) крайни разлики. Например, хоризонталната разлика *H* смятаме, като извадим десния съсед от левия и делим на 2 (ако нямаме някой от съседите, просто вадим съществуващия съсед, от цвета на нашата точка, без да делим). Дължината $G = sqrt(H^2 + V^2)$ ползваме за оценка на контраста. Всичките градиенти вкарваме в приоритетна опашка, като в последствие ги вадим една по една, и около всяка от изваждените точки разпитваме за 8-те съседа (при двойно по-голяма разделителна способност, т.е. ако началното разстояние между грид-точките е било *h*, сега питаме за 8-те съседа при разстояние *h*/2). Вадим и разпитваме докато свърши времето.

Човек би си помислил, че е добре тази схема да се прилага рекурсивно, т.е. на ново намерените точки също се оценят градиентите им и се вкарат в приоритетната опашка (разбира се с по-ниска оценка, за да предотвратим решението да се "забие" и да "разпита тотално" за някаква малка област, вместо да проучи по-цялостно снимката). Оказва се, че това е лоша идея и има смисъл само за малки картинки, когато имаме време за много повече от 500 въпроса. Затова моето решение все пак реализира тази схема, но само при извънредни обстоятелства (т.е. допълнителните точки вкарва в отделна опашка, независима от първата, и чак когато първата се изпразни, тогава започва да ползва втората).

Една интересна идея тук е да повдигнем малко приоритета на точките, близо до средата на картинката. Човек интуитивно би очаквал там да е по-интересната част от картинката. Това се оказва добра идея, като в моята реализация, централните точки получават с до 40% бонус в оценката на градиентите си.

Добре, да приемем, че вече сме поразпитали за интересните области на картинката, а времето е накъсяло, таймерът е на 9.5*s* и от нас се очаква да "предадем" решение. Как да интерполираме цветовете между нашия, вече неравномерен, грид? Оказва се, че това е неочаквано труден проблем. Има много идеи, всяка с различни предимства и недостатъци (оптимална така и не намерих). За щастие, може да ги пробваме всичките – "тестването" на конкретна интерполация е *евтино* (само 1 въпрос), така че има смисъл да изредим всичките.

Идея номер едно: намираме 4-те най-близки точки - върхове на правоъгълник, и интерполираме билинейно между тях. За съжаление, неравномерният грид доста пречи на тази идея.

Идея номер две: намираме 4-те най-близки точки въобще. Всяка точка допринася за крайния цвят, като нейната тежест зависи линейно от разстоянието. Проблемът на този поход е, че се получават лесно "прекъснатости" - заради разстоянията се получават области на влияние, подобни на клетките при диаграмите на Вороной. Особено си личи около неравномерните точки.

Идея три: комбинация от двете. Когато можем, ползваме идея 1 (т.е. ако сме заобиколени от 4 "равномерни" точки, и то само ако са много близки - не повече от 4x4 квадратче се допуска), а за останалите - идея 2.

За всяка от трите идеи се оказва благодатно да се приложи леко замазване (blur) на получаваната картинка. Неочаквано, но, макар че губим малко данни така, решението се подобрява! Прилагат се три типа замазване - box blur 3x3 (всеки пиксел се заменя с усредената стойност на него + 8-те съседа), gaussian blur с kernel 3x3, и с 5x5 (за по-мощно замазване). Кой от всичките методи е най-добър зависи от конкретната картинка, което мотивира да пробваме всичките.