Anàlisi d'un sistema pel reconeixement d'objectes en imatges en temps real

Albert Vives Bach

Resum—En l'àmbit del reconeixement d'imatges pretenem analitzar i augmentar el rendiment d'un dels millors algorismes coneguts avui en dia per detectar tots els objectes d'una imatge en menys d'un segon. El nostre propòsit serà entendre el seu funcionament, analitzar la complexitat algorítmica que comporta i fer un estudi de possibles millores tan pel que fa a la capacitat de detecció d'objectes com en la velocitat amb què ho fa. Presentarem els resultats de l'analisi complert de la complexitat algorítmica i què implica això en quant a les possibiltats que tenim de millora i també inclourem la importància dels paràmetres d'entrada de l'algorisme Randomized Prim (RP) acompanyat d'un estudi sobre quins valors permeten augmentar-ne el rendiment.

Paraules clau—Detecció, RP, Prim, reconeixement, objectes, IoU, propostes, graf, arbre, segmentació, superpíxel, color, algorisme, GroundTruth.

Abstract—In the field of image recognition, we aim to analyze and improve performance of one of the best algorithms known nowadays to detect all objects in an image in less than a second. Our main purpose is to understand how it works, analyze the algorithmic complexity involved and get some estimations on the possible improvements as regards the ability to detect objects as the speed with which it does. We will show the results from the entire analysis of algorithmic complexity and what this implies in terms of chances for improvement. We have also included the importance of the input parameters of the algorithm randomized Prim (RP) accompanied by a study what values can increase its performance.

Index Terms—Detection, RP, Prim, recognition, objects, IOU, proposals, graph, tree, segmentation, superpixel, color, algorithm GroundTruth.

1 Introducció

UNA de les eines més potents que s'esta desenvolupant actualment gràcies a l'aparició de la visió per computador és el reconeixement automàtic d'objectes en imatges.

El reconeixement d'objectes en imatges consisteix en l'obtenció de finestres (o com les anomenarem al llarg de l'article, propostes) que permeten ubicar tots els objectes dins d'una imatge.

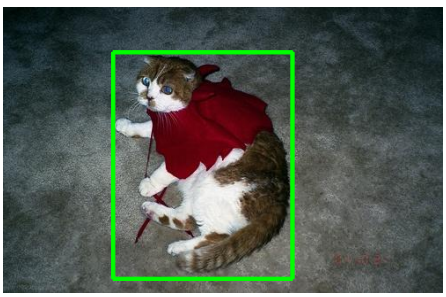

Figura 1. Imatge amb la proposta de l'algorisme RP ressaltada en color verd.

En la Figura 1 podem veure un exemple de quin seria el resultat esperat d'executar aquest algorisme, amb tots els objectes destacables de la imatge resaltats per una proposta el més ajustada possible.

Aquest procés ens permet focalitzar-nos únicament en aquelles parts de la imatge de major interès i pot ser aprofitat per multitutd de camps actualment en fase de desenvolupament per millorar els temps dedicats en l'extracció d'informació a partir d'imatges.

En aquest projecte ens centrarem en l'anàlisi de l'algorisme Prim aleatoritzat (Randomized Prim's algorithm o RP) aplicat a aquest camp.

Gràcies a la eficiència d'aquest algorisme podem obtenir 1000 propostes d'objectes en una imatge en aproximadament 0,7 segons i s'ha pogut demostrar en les competicions de PASCAL VOC 2007 i 2012 i amb el conjunt de dades SUN2012, que millora els seus competidors en multitud d'escenaris diferents on s'ha provat l'algorisme RP.

- *E-mail de contacte: albertvives91@gmail.com*

- *Menció realitzada: Computació*
- *Treball tutoritzat per: Jordi Gonzalez Sabaté (CVC)*
- *Curs 2013/14*

2 OBJECTIUS

L'objectiu principal del meu treball és comprendre com funciona i quins resultats s'obtenen amb el conegut algorisme RP (Randomized Prim) en l'àmbit del reconeixement d'objectes en imatges, així com realitzar un estudi de la seva complexitat algorítmica i analitzar quins punts d'aquest algorisme són susceptibles de millora amb una sèries d'anàlisis sobre els paràmetres d'entrada d'aquest. Podríem resumir els objectius plantejats en aquest article amb els següents 3 punts:

- Comprendre el funcionament de l'algorisme RP.
- Obtenir l'anàlisi de la complexitat de l'algorisme.
- Analitzar els paràmetres d'entrada a l'algorisme per tal d'optimitzar-ne el rendiment en quant a capacitat de detecció d'objectes i reducció del temps d'execució per cada imatge.

3 ESTRUCTURA DEL DOCUMENT

A continuació es prensenta un breu apartat per resumir els passos a seguir i la metologia empleada per assolir cadascun dels objectius proposats. Tot seguit hi ha un resum explicatiu de l'estat de l'art en el reconeixement d'objectes en imatges.

Posteriorment trobem un apartat que entra en detall en el funcionament de l'algorisme RP. Després veurem com s'han dut a terme els estudis dels paràmetres d'entrada de l'algorisme i quins resultats n'hem extret, veurem quina complexitat presenta i quines parts del codi podem millorar.

Per últim es troben unes breus conclusions del projecte i possibles línies futures de treball a partir dels resultats obtinguts.

4 METODOLOGIA

4.1 Estudi previ i inicialització del projecte

Per tal d'assolir amb èxit el propòsit i objectius d'aquest projecte al inciar la primera fase he hagut de familiaritzar-me amb l'algorisme RP seguints els següents passos:

- Analitzar la documentació existent i el codi font de l'algorisme RP.

- Analitzar els conjunts de dades que s'han utilitzat i els resultats que s'han obtingut amb el seu ús recentment (l'any 2013).

- Preparar l'entorn necessari per executar l'algorisme i preparar un GroundTruth per poder determinar les millores que es van obtenint amb l'algorisme.

4.2 Optimització i Complexitat

Un cop tenim els coneixements necessaris sobre el funcionament de l'algorisme RP i disposem de l'entorn per treballar-hi, passem a estudiar vies de millora:

- Optimitzar els paràmetres de configuració que actualment vénen amb un valor per defecte amb l'aplicació de tècniques d'intel·ligència artificial per tal de millorar el resultat de l'algorisme en la majoria dels entorns on s'apliqui.

- Analitzar la complexitat algorítmica de l'RP i extreure conclusions de quines parts del codi s'haurien d'abordar en una fase posterior.

5 ESTAT DE L'ART

En un anàlisi que es feia en el report "Prime Object Proposals with Randomized Prim's Algorithm" [3] l'any 2013 es demostra que avui en dia l'algorisme RP no es queda enrere amb els seus competidors en l'àmbit del reconeixement d'objectes en imatges.

Es compara el 'Randomized Prim' amb algorismes com el 'Selective Search' [10], el 'Objectness' [11] i el 'Rahtu' [12] en termes de lineal/log VUS (Volume Under Surface) i en termes d'IoU i detection rate (sent aquests 2 últims els que focalitzarem en aquert article).

Pel que fa a la comparativa respecte al lineal/log VUS, l'RP es situa sempre al capdamunt de la classificació obtenint uns valors de 0.59 pel cas lineal, i 0.28 pel logarítmic, mentre que els seus rivals només arriben a 0.49 en el lineal (és a dir, 0.1 menys que l'RP) i 0.25 en el logarítmic (per tant, 0.03 menys que l'RP).

D'altra banda els resultats en comparació al detection rate no es queden enrere ja que quan fixem un llindar de IoU per sobre de 0.6, i fem servir més de 5.000 propostes de la resposta de l'algorisme, gairebé sempre ens trobem per sobre dels nostres competidors, al voltant del 0.85 de detection rate.

6 ALGORISME PRIM ALEATORITZAT

6.1 Funcionament

L'algorisme de 'Prim aleatoritzat' es basa en extreure els millors arbres d'expansió (spanning trees) aleatoris a partir d'una segmentació de la imatge en 'superpíxels', grans superfícies de la imatge que tenen quelcom en comú.

A partir d'un Graf de connectivitat associat als superpíxels de la imatge, aquest algorisme és un mètode incremental que va aprofundint en el coneixement que tenim del nostre arbre d'expansió i a cada iteració, analitza els vèrtexs del Graf sense utilitzar i afegeix al nostre arbre el vèrtex que té una major probabilitat de contenir un objecte que forma part del nostre arbre actual.
L'objectiu final és, en el nostre cas, obtenir tots els objectes de la imatge a partir de les finestres (bounding-boxes) que obtenim de cadascuna de les associacions entre superpíxels.

6.2 Cerca binària

La clau de l'eficiència d'aquest algorisme rau en la estructura de dades max-heap que utilitza per mantenir un seguiment dels nodes a analitzar de forma dinàmica. El fet de fer servir una estructura max-heap permet que la inserció i eliminació de nodes fent servir una cerca binària (BST) no superi la complexitat logarítmica.

Aquest algorisme es pot executar per 4 espais de color diferents: HSV, Lab, Opponent i rg.
Les característiques que s'analitzen per al vector de característiques són la similitud de color, la dimensió dels superpíxels i la relació entre distàncies fronteres (contorns)

als superpíxels.

6.3 Terminació de l'algorisme

És interessant mencionar com ho fa aquest algorisme per determinar quan s'ha d'aturar. Per fer-ho ens basem en dos paràmetres:

- La probabilitat de que l'aresta mostrejada no connecti superpíxels del mateix objecte: (1- ρn,m).
- Una proporció extreta del nombre d'objectes obtinguts del conjunt d'entrenament que tenen una area menor que la del arbre d'expansió: α(Tk)).

Aquests dos paràmetres es relacionen entre sí segons la fórmula descrita a continuació (1).

$$\varepsilon(Tk,ek) = (1- \rho n,m + \alpha(Tk))/2 \qquad (1)$$

6.4 Pseudocodi del Randomized Prim

A continuació es motra el pseucodi del 'Randomized Prim' indicant les principals accions que realitza per tal d'obtenir les propostes d'objectes en base les segmentacions que fa de la imatge i a l'arbre d'expansió de cada superpixel.

En el pseudocodi s'han simplificat algunes operacions que impedien la legibilitat del codi i no aportaven informació sobre el funcionament bàsic de l'algorisme.

També s'han extret alguns càlculs en mètodes apart amb noms que permetessin interpretar el contingut d'aquests.

```
//sp = superpixel; nSps = numero de superpixels
funcio RP(rgbI, params, alpha, condicio_de_sortida)
        imatge   = rgbI.convertirAlColorspace(
                   params.colorspace ) );
        segImg   = segmentaImatgeEnSP( imatge,
                   params.spParams() );
        Graph graf = construirGraf(
                     rgbI, segImg, params.fWeights );
        nProposals = params.nombre_de_propostes();
        nSps = segImg.nSps;
        grups_sp = vector( nProposals, nSps );
        arbre = inicialitzarArbreExpansio( nSps );
        per ( k = 0; k < nProposals; k++ )
          seguent_sp=obtenirSpAleatori(nSps);
          suma_sp = segImg.normArea16b(
                    seguent_sp );
          veinatge = graf.veinsSeguentNode(
                    seguent_sp );
          per ( n = 0; n < veinatge.size(); n++ )
            arbre.afegirMillorNode( veinatge[n] );
          fiper
          sortir = 0;
          mentre sortir = 0
            seguent_sp = obtenirSeguentNode(
                         grups_sp );
            si suma_sp > condicio_de_sortida
              sortir = 1;
            sino
              suma_sp += alpha[suma_sp];
              suma_sp += segImg.normArea16b(
                         seguent_sp );
              veinatge = graf.obtenirVeinatge(
                         seguent_sp );
              per n = 0; n < veinatge.size(); n++
                arbre.actualitzaNodes( seguent_sp );
              fiper
            fisi
          fimentre
        xmin, y_min = VALOR_PER_DEFECTE;
        xmax, ymax = 0;
        per s = 0; s < nSps; s++
          BBox = segImg.bbox( s );
          si xmin > BBox.jMin
            xmin = BBox.jMin;
          fisi
          si ymin > BBox.iMin
            ymin = BBox.iMin;
          fisi
          si xmax > BBox.jMax
            xmax = BBox.jMax;
          fisi
          si ymax > BBox.iMax
            ymax = BBox.iMax;
            fisi
          fiper
          bbProposals[k] = [xmin, ymin, xmax, ymax];
          arbre.Reset();
        fiper
fifuncio
```

7 PROVES PRELIMINARS

Per familiaritzar-nos amb l'algorisme RP i el conjunt de dades hem fet unes quantes proves per veure quines eren les millors propostes que era capaç de donar-nos en diferents escenaris. En aquestes proves hem destacat de color verd les finestres que identifiquen correctament als objectes que s'espera que l'RP pugui detectar en la imatge i de color vermell es mostren les millors propostes que s'han obtingut per a cada objecte. Per saber quina era la millor proposta que ens oferia l'RP hem comparat cada proposta amb l'objecte esperat en termes d'IoU (Intersecció sobre la Unió) i ens hem quedat amb el millor en cada cas.

Figura 2. Tren mal detectat a causa de la similitud de color amb objectes laterals.

En una primera aproximació amb la classe dels trens veiem que en general, la gran dimensió d'aquesta classe d'objectes fa que el nostre algorisme els pugui identificar bé (Figures 3 i 4) tot i que veiem que a causa de la similitud de colors a vegades es pot confondre (Figura 2).

Figura 3. Tren correctament detectat per la gran dimensió de l'objecte.

Figura 4. Tren correctament detectat pel color uniforme i la gran dimensió de l'objecte.

Això però, no seria un inconvenient molt greu degut a que sabem que aquest defecte del RP es pot eliminar fent servir més d'un colorspace (recordem que en tenim 4 que són el rg, el opponent, el Lab i el HSV).

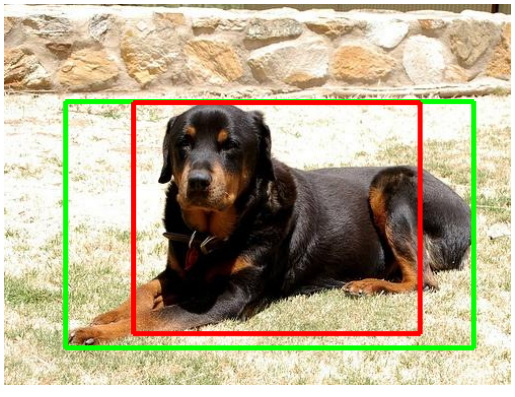

Figura 5. Gos correctament detectat tret de les potes, que tenen un color diferent i sobresurten de la forma esperada del cos.

Amb una altra tipus de classe, com és la del gos, podem veure que generalment és capaç de detectar-los sempre que no hi hagi parts del cos relativament petites (en comparació amb el tamany del gos) que s'allunyen de la forma geomètrica que es pot esperar en què es trobarà l'animal (de peu, veient les 4 potes, o estirat formant un forma de rectangle allargat), com veiem a la Figura 5.

D'altra banda veiem que amb cossos massa petits, l'algorisme és capaç de detectar l'objecte però té tendència a agafar regions llargues de la imatge i no és capaça d'ajustar-se perfectament al gos. Això es pot canviar amb el paràmetre 'c' i 'min_size'.

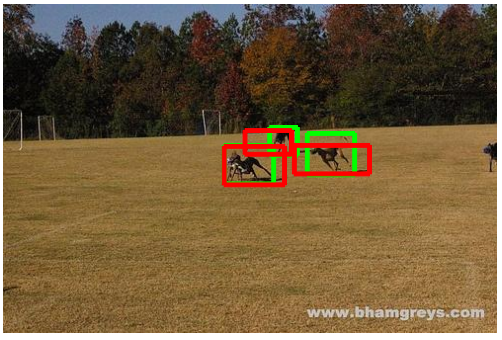

Figura 6. Gossos massa petits per detectar-ne correctament la forma complerta.

En el següent exemple de la classe bicicleta tornem a notar que amb objectes poc rígids l'algorisme tendeix a fallar (Figura 7).

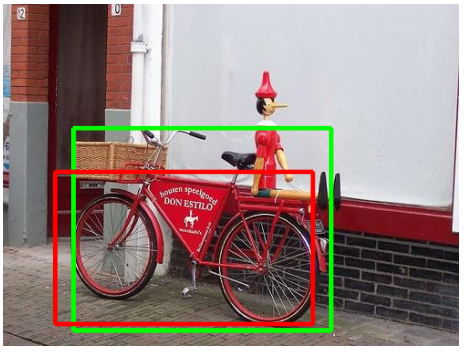

Figura 7. Imatge de la classe Bicicleta amb error de detecció en cossos no rígids.

A continuació veiem un exemple de cos rígid en què efectivament l'algorisme RP ha sigut capaç de detectar l'objecte perfectament (Figura 8).

Figura 8. Classe Cadira amb detecció correcta en un cos rígid.

Per útlim veurem el cas de la classe Cotxe. Aquests tipus d'objectes són més rígids però en alguns casos veiem que l'algorisme torna a fallar si l'objecte és massa petit ja que tendeix a quedar-se amb objectes grans (Figura 9). Per tant, caldria decrementar el valor de 'c' per evitar que descuidi objectes petits però això produiria un increment en el temps d'execució necessari.

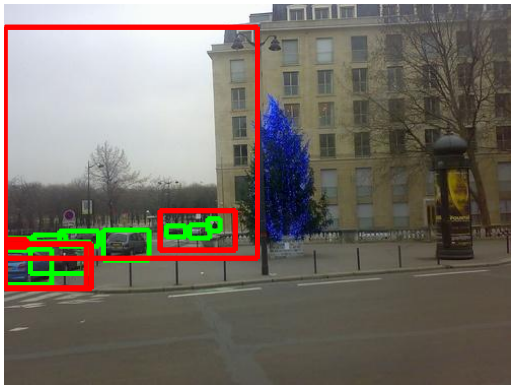

Figura 9. Classe Cotxe amb detecció incorrecta per la tendencia a agafar objectes grans.

En contraposició, veiem un altre en què ha pogut detectar la majoria de cotxes però segueix haven-t'hi l'inconvenient en la confusió pel color (Figura 10).

Figura 10. Classe Cotxe amb detecció incorrecta pel color de fons.

8 CONJUNT DE DADES

En les proves que s'han realitzat en anteriors projectes en el report "Prime Object Proposals with Randomized Prim's Algorithm" [3], es va determinar que s'obtenien els millors resultats amb el dataset (o conjunt de dades) PASCAL VOC2007 [13] i per tant en aquest article ens centrarem en aquest conjunt de dades per poder comparar els resultats que obtinguem amb els que s'han obtingut anteriorment.

Tal com explicàvem en la secció 5 (Estat de l'art) l'algorisme 'Randomized Prim' presenta els millors resultats en la majoria de casos però aquí analitzarem quins paràmetres d'entrada poden millorar-ne la capacitat de detecció d'objectes i alhora, com aconseguir que aquest procés produeixi bons resultats en poc temps, ja que en molts camps de visió per computadors el temps és un factor determinant. Posem per exemple que emprem el reconeixement d'objectes en un vehicle automòbil que circula de forma autònoma i que pretenem detectar quan hem d'aturar-nos a un semàfor. En aquest cas seria un greu defecte que fóssim capaços de detectar el semàfor en 2 segons, i saber de quin color estar, ja que en aquest temps podríem haver-lo passat de llarg.

El conjunt de dades de PASCAL VOC2007 [13] conté 20 classes d'objectes com ara trens, ocells, autobusos, televisors, gats, bicicletes, etc. Tot i així nosaltres ens centrarem en la detecció de tots ells, sense diferenciar de quina classe són perquè ens interessa que l'algorisme sigui el més genèric possibles perquè doni resultats acceptables en tots els àmbits.

9 ESTUDI DELS PARÀMETRES DE CONFIGURACIÓ

9.1 Motivació i procediment

Amb l'objectiu d'obtenir els paràmetres que optimitzen el rendiment de l'algorisme RP en quant a capacitat de detecció d'objectes i velocitat de processament, hem processat imatges del dataset PASCAL VOC2007 [13] i n'hem extret un seguit de gràfiques que ens han permès entreveure la tendència de l'algorisme en funció de cada paràmetre de la configuració.

Per qüestions de temps i recursos disponibles (només disposàvem de dues setmanes en la fase d'optimització i un ordinador amb 1 GHz de memòria RAM) hem testejat sobre 100 imatges un total de 80 configuracions que combinen diferents paràmetres d'entrada a l'algorisme RP i que controlem externament abans de la crida a aquest. En el nostre cas hem optat per usar com a llenguatge de programació el MatLab perquè permet una gran capacitat de càlcul matemàtic i ja inclou la possibilitat d'usar funcions compilades en el llenguatge C/C++ propi del RP).

Els paràmetres de configuració avaluats han estat els següents:

- 'Sigma': Pot prendre valors entre 0 i 1 i s'utilitza en la fase de segmentació de la imatge en superpíxels per aplicar un filtre Gaussià i simplificar la imatge. Tenint en compte que feien servir aquest paràmetre amb valor 0.8 nosaltres hem provate les següents possibilitats: 0.3 – 0.5 – 0.7 – 0.8 -0.9.
- 'C': S'utilitzava anteriorment amb valor 500, tot i que pot prendre qualsevol valor positiu i nosaltres hem provat: 70 – 100 – 250 – 500. Aquest paràmetre genera una 'tendència' cap a l'obtenció de components més llargs.
- 'Min_size': Aquest paràmetre força a que els propostes d'objecte resultants tinguin com a mínim la dimensió especificada per aquest valor. Actualment es fa servir amb valor 100 i nosaltres hem provat les següents combinacions: 30 – 60 – 100 – 150.

D'altra banda era imprescindible obtenir un valor únic

entre 0 i 1 que ens indiqués com de bona era la resposta de l'RP amb cada configuració, i per això hem optat pel càlcul de la Intersecció sobre la Unió (IoU), que té en compte la regió (o àrea) de l'objecte detectada i si la finestra proposada s'ajusta al màxim possible a la dimensió de l'objecte o per contra, selecciona una àrea de la imatge molt més gran i que no ens aporta tant de valor.

A més a més, hem analitzat el percentatge de detecció d'objectes (Detection rate) per al qual hem considerat que un objecte ha estat correctament detectat a partir de superar el 0.5 en IoU (per seguir amb la mateixa consideració que havien pres en el report 'Prime Object Proposals with Randomized Prim's Algorithm' [3]).

Per últim hem inclòs un anàlisi del temps empleat per l'algorisme per duu a terme aquests càlculs ja que l'objectiu final és reduir complexitat i temps de l'algorisme, sense minvar la capacitat de detecció que ja ofereix d'entrada.

Per tant, per cada configuració hem analitzat la resposta de l'algorisme RP en termes de IoU (intersecció sobre la unió), Detection rate (taxa de detecció) i Delay (retard) per tal d'esbrinar quina configuració ens centra millor els objectes en les imatges, quina ens detecta un major nombre d'imatges i quina és la configuració més ràpida alhora de detectar objectes en imatges.

Cal tenir en compte que el temps total dels resultats presentats a continuació són per a 100 imatges/configuració, tenint en compte el càlculs 'getGroundTruth' i 'getMetrics', que són externs al RP i tenen una duració constant en tots els casos, amb una complexitat lineal que depèn del nombre de propostes obtingudes i del nombre d'imatges analitzades.

Per últim destacarem que en totes les proves hem inclòs una última columna destacada en color vermell que representa el resultat que ha tingut l'experiment amb els valors per defecte de l'algorisme RP per tal de poder tenir una estimació de la millora aconseguida amb cada modificació.

9.2 Resultats

9.2.1 Resultats en base a la Intersecció sobre la Unió (IoU)

Els millors valors per 'sigma' són entre 0.7 i 0.9 però no aporta una gran diferència quan cerquem el millor IoU ja que en totes les combinacions sempre trobem almenys un cas en què els resultats superen el 0.8 de IoU (Figures 11, 12 i 13).

L'algorisme millora notòriament amb valors petits de 'c' (entre 70 i 100), com podem veure en les primeres combinacions de la Figura 11.
L'algorisme millora sensiblement amb valors grans de min_size (entre 100 i 150) com veiem en la Figura 13.

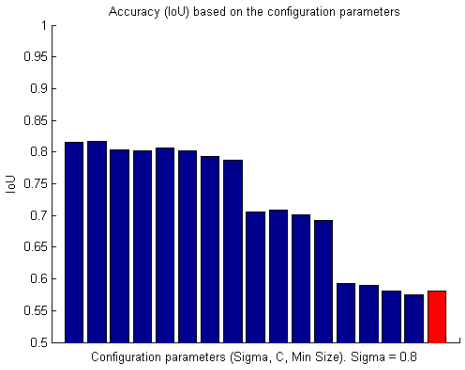

Figura 11. Intersecció sobre la Unió en funció dels valors C = [70, 100, 250, 500] i Min_size = [30, 60, 100, 150] amb Sigma = 0,8. En vermell es destaca el resultat que s'obtenia amb els valors per defecte.

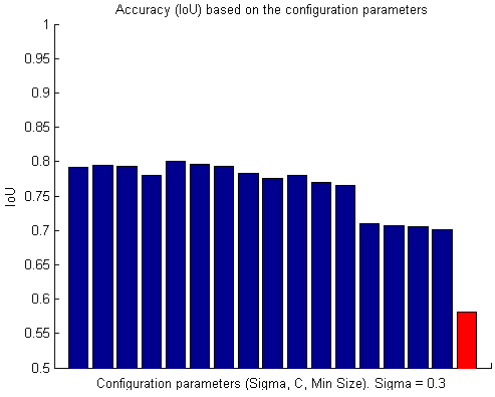

Figura 12. Intersecció sobre la Unió en funció dels valors C = [70, 100, 250, 500] i Min_size = [30, 60, 100, 150] amb Sigma = 0,3. En vermell es destaca el resultat que s'obtenia amb els valors per defecte.

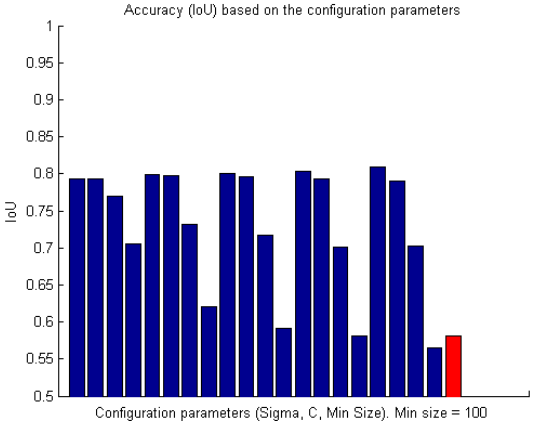

Figura 13. Intersecció sobre la Unió en funció dels valors Sigma = [0.3, 0.5, 0.7, 0.8, 0.9] i C = [70, 100, 250, 500] amb Min_size = 100. En vermell es destaca el resultat que s'obtenia amb els valors per defecte.

Millor configuració => Sigma: 0.8 - C: 70 - Min_size: 60
IoU: 0.8173; Detection_rate: 0.9180; Delay: 725.6767 segons.
Cal destacar també que amb un baix valor de 'c' (com ara 70) i un alt valor de 'min_size' (100 - 150) tenim molt bons resultats en poc temps (Figura 14).

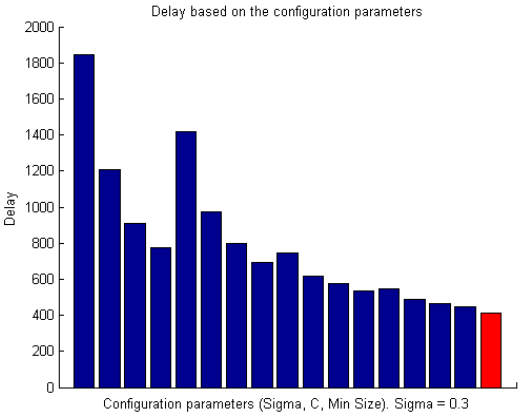

Figura 14. Delay en funció dels valors C = [70, 100, 250, 500] i Min_size = [30, 60, 100, 150] amb Sigma = 0,3. En vermell es destaca el resultat que s'obtenia amb els valors per defecte.

9.2.2 Resultats en base al percentatge de detecció (Detection Rate)
El paràmetre 'sigma' no influeix gaire en valors alts de Detection rate (Figures 15 i 16).

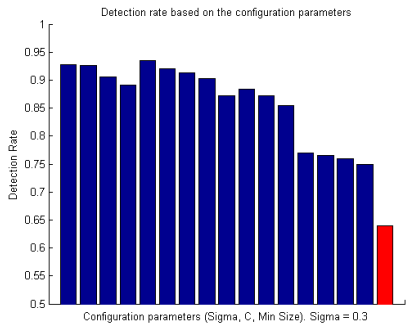

Figura 15. Detection rate en funció dels valors C = [70, 100, 250, 500] i Min_size = [30, 60, 100, 150] amb Sigma = 0,3.

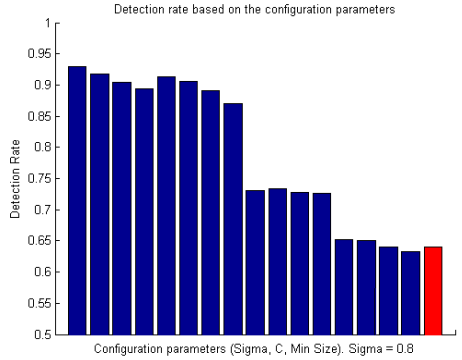

Figura 16. Detection rate en funció dels valors C = [70, 100, 250, 500] i Min_size = [30, 60, 100, 150] amb Sigma = 0,8.

Com més baix és el valor de 'c' millors resultats s'obtenen si comparem la Figura 16, on obtenim un detection rate >= 0.9 amb valors de 'c' = 70 i 'c' = 100 i que si ho comparem amb la Figura 17 veiem que decau en picat amb valors grans de 'c'.

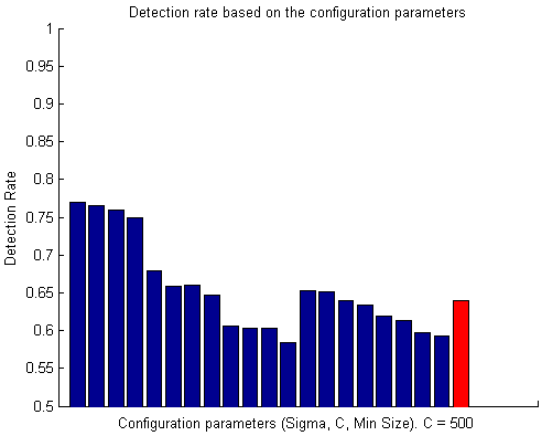

Figura 17. Detection rate en funció dels valors Sigma = [0.3, 0.5, 0.7, 0.8, 0.9] i Min_size = [30, 60, 100, 150] amb C = 500.

El 'min_size' no té un impacte molt gran en el detection rate tot i que és relativament millor per a valors grans (Figura 18).

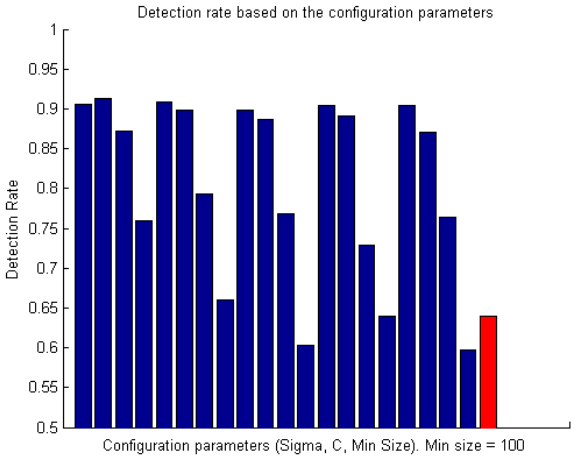

Figura 18. Detection rate en funció dels valors Sigma = [0.3, 0.5, 0.7, 0.8, 0.9] i C = [70, 100, 250, 500] amb Min_size = 100.

Millor configuració => Sigma: 0.5 - C: 70 - Min_size: 30
IoU: 0.8075; Detection_rate: 0.9360; Delay: 1164.9 segons.

9.2.3 Resultats en base al temps d'execució (Delay)

El paràmetre sigma tampoc influeix en el temps d'execució ja que per molt que en variem el valor les oscil·lacions es produeixen per cause de canvi de Min_size (Figura 19).

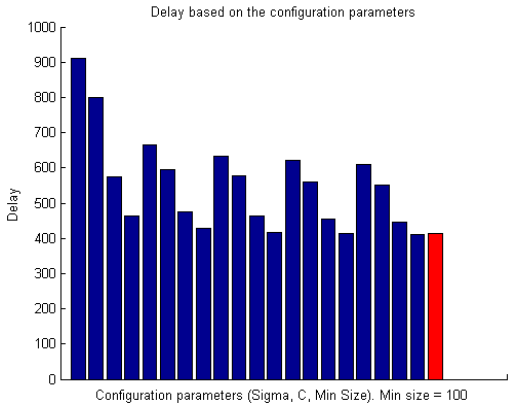

Figura 19. Delay en funció dels valors Sigma = [0.3, 0.5, 0.7, 0.8, 0.9] i C = [70, 100, 250, 500] amb Min_size = 100.

Com més gran és el valor de 'c' més ràpid va l'RP (Figures 20 i 21).
Com més gran és el valor de 'min_size' més ràpid va, tot i que a partir d'un cert valor de 'c' (entre 150-200) s'estabilitza.

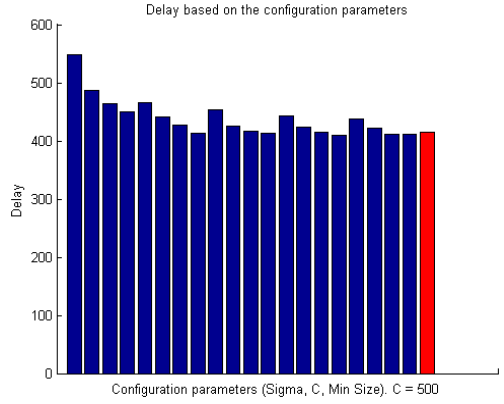

Figura 20. Delay en funció dels valors Sigma = [0.3, 0.5, 0.7, 0.8, 0.9] i Min_size = [30, 60, 100, 150] amb C = 500.

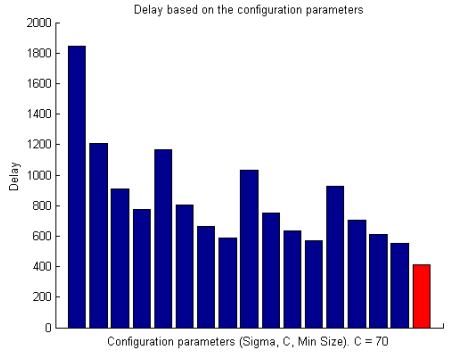

Figura 21. Delay en funció dels valors Sigma = [0.3, 0.5, 0.7, 0.9] i Min_size = [30, 60, 100, 150] amb C = 70.

Millor configuració => Sigma: 0.8 - C: 250 - Min_size: 100
IoU: 0.5742; Detection_rate: 0.6335; Delay: 410.2334 segons

9.3 Conclusions

Com a conclusions principals podem extreure per tant, que el paràmetre 'sigma' no ens aporta una millora prou significativa com per tenir-lo en compte i que el paràmetre 'c' millora amb valors petits, mentre que el 'min_size' millora amb valors grans.

La base de dades que ens ha proporcionat un major encert en la detecció d'objectes ha estat el VOC2007, tot i que la tendència és pràcticament idèntica per a tots els datasets, la qual cosa reafirma les nostres conclusions sobre els millors i pitjors valors de configuració.

10 COMPLEXITAT ALGORÍTMICA

10.1 Metodologia

Per avaluar la complexitat algorítmica de l'RP he hagut d'assumir que les inicialitzacions de les variables amb valors constants tenen una complexitat que es pot obviar, O(0) i totes les operacions bàsiques (suma, resta, multiplicació i divisió), les comparacions i assignacions amb valors no constants tenen una complexitat constant O(1).

El nombre de camins en aquest algorisme era tan elevat que no era viable analitzar-los tots i he enfocat l'anàlisi cap al pitjor dels casos en totes les possibles bifurcacions (decisions condicionals) i bucles, de tal manera que sempre he escollit la branca més complexa de les que podia tenir una estructura if – else, i sempre que un bucle tenia una condició de terminació que depenia de múltiples factors li he assignat la complexitat d'iterar el major nombre de vegades possible.

D'aquesta manera tenim un anàlisi que ens permet saber què pot arribar a passar i amb aquests resultats podem focalitzar-nos en les seccions més complexes o que tenen bucles o condicions innecessàries.

L'anàlisi l'he dut a terme primerament fitxer a fitxer, de forma individualitzada, de forma que en qualsevol moment es pot consultar quina és la complexitat de cadascun d'ells per separat.

Dins de cada fitxer, he analitzat cadascuna de les funcions per separat i n'he extret la complexitat en funció de les crides a externs (altres fitxers i funcions del propi codi o de C++). Per últim he afegit a sota de cada resultat, el mateix anàlisi però substituint la complexitat de cada crida externa pel seu valor real.

De forma addicional he inclòs un fitxer common.h que inclou l'anàlisi de la complexitat de funcions pròpies de C++ de gran utilitat per al resultat final d'aquest anàlisi.

10.2 Resultats i Conclusions

La complexitat total de l'algorisme RP és la següent:

$$O(12 * n^2 + n * log2(n) * (n + 21) + 62 * n^3 + 3 * n^4 + 5 * n^5 + n^7) \quad (2)$$

Per tant, podem concloure segons la complexitat ex-

pressada en l'anterior fórmula (2) que l'algorisme RP té una complexitat n^7. Això implica que reduir el nombre d'iteracions és crucial i reaprofitar les operacions que es realitzen a l'interior dels bucles també.

Al llarg de tot el codi es realitzen un seguit d'assercions que afegeixen complexitat al codi i que no aporten major seguretat, amb la qual cosa caldria eliminar-les.

Les principals funcionalitats que utilitza l'RP són:

- Conversió de Colorspace (funció convertToColorspace del fitxer image.h).

$$O\big(7 + 31 * O(height) * O(width)\big) = O(n^2) \quad (3)$$

- Segmentació en superpíxels (funció SegImage del fitxer seg_image.h.

$$O(4 * n + n * log2(n) + 3 * n^3 + 5 * n^4) = O(n^4) \quad (4)$$

- Generació de l'arbre d'expansió (funcio Graph del fitxer graph.h).

$$O(31 * n^2 + n^6) = O(n^6) \quad (5)$$

- Generació de propostes (rp.h).

$$O\big(n^3 + n * log2(n)\big) = O(n^3) \quad (6)$$

Podem deduir que cal simplificar al mínim la generació de l'arbre d'expansió, que és la que ens aporta una major complexitat.

Propostes de futures millores:

- Image.h: Sempre es fa la comparació de colorspace amb el valor RGB i es recorre 3 vegades 'h' (*height*) i 'w' (*width*) en Opponent.

- mexFelzenSegmentIndex.cpp: Es poden reaprofitar alguns bucles sobre la variable '*num_css*' per adelantar càlculs que actualment estan repartits pel codi però que no tenen dependències entre sí.

- RP.m: Hi ha 3 bucles sobre la variable '*nSegs*' que es podrien agrupar.

- RP_mex.cpp: No cal el bucle sobre la variable '*bbProposalsVector*', podrien realitzar-se aquests càlculs dins del mètode RP i aprofitar el bucle intern.

- Seg_image.h: Hi ha 17 sentències d'operacions d'entrada/sortida repetitives que s'han d'analitzar i simplificar al mètode *ExtractSpInfo*.

11 Conclusió

L'algorisme Randomized Prim és molt ràpid i precís alhora de reconèixer objectes en imatges ja que triga un promig de 0.7 segons per imatge i detecta al voltant del 91% dels objectes correctament.

Tot i així, en base als estudis que hem realitzat durant aquest projecte sabem que se li poden aplicar millores per tal d'incrementar-ne el rendiment.

Depenent de la categoria d'objecte que s'analitzi, si és més gran o més petit, si el color es confon amb el fons o si l'objectes és més o menys rígid l'algorisme tendeix a l'error i això es pot millorar amb els paràmetres avaluats 'c' i 'min_size' i amb la combinació de diferents espais de color (colorspace) com el HSV, el Lab o el opponent però això, sempre va en detriment de la rapidesa del algorisme.

Per una altra banda, és possible aprofundir en els valors precisos que milloren el rendiment del RP i ja partim d'una configuració millor que la incial.

Per últim, amb l'anàlisi de la complexitat del RP hem descobert que hi ha una sèrie de bucles innecessaris o mal aprofitats i que moltes de les operacions es poden compactar per tal d'optimitzar l'algorisme.

12 Futures línies de treball

En base als resultats d'aquest projecte, podem preveure que caldrà una posterior tasca per aprofundir en els valors que fan que els paràmetres d'entrada al RP obtinguin el major percentatge de detecció d'objectes en el menor temps possible.

D'altra banda també queda oberta una línia de treball per optimitzar els bucles que estan desaprofitats i cal un anàlisi més profund de la generació de l'arbre d'expansió que, per ara, és la part del codi amb major complexitat del RP.

Agraïments

Voldria agrair el suport d'en Júnior Fabian i d'en Jordi Gonzalez pel suport constant al llarg d'aquest semestre que ha resultat imprescindible per duu a bon port aquest projecte.

Bibliografia

[1] PDOLLAR. A Seismic Shift in Object Detection [En línia]. Enllaç web: http://pdollar.wordpress.com/2013/12/10/a-seismic-shift-in-object-detection/ - Nova perspectiva del processament d'imatges basat en la Segmentació [última consulta: 28/març/2014]

[2] Kristen Grauman i UT-Austin. Sliding window detection, 29/gener/2009. 42 pàgines [Document PDF en línia]. Enllaç web: http://www.cs.utexas.edu/~grauman/courses/spring2009/slides/sliding_windows.pdf - Què és un Classificador i com s'avalua [última consulta: 11/febrer/2014]

[3] Santiago Manen, Matthieu Guillaumin i Luc Van Gool. Prime Object Proposals with Randomized Prim's Algorithm, 2013. 8 pàgines [Document PDF en línia]. Enllaç web: http://biwinas03.ee.ethz.ch/mguillau/publications/Manen2013iccv.pdf Funcionament de l'algorisme Randomized Prim i la seva aplicació en reconeixement d'objectes en imatges [última consulta: 28/juny/2014]

[4] Smanenfr, Random Prim's Algorithm for Object Proposals, febrer/2014. [En línia] Enllaç web: https://github.com/smanenfr/rp#rp – Codi font de l'algorisme RP i demo amb el setup [última consulta: 15/juny/2014]

[5] Mark Everingham, The PASCAL Visual Object Classes Challenge 2007. [En línia]. Enllaç web: http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2007/#testdata – Competició 2007 de detecció d'objectes en imatges. Conté l'enllaç als conjunts de dades amb els què s'ha experimentat amb l'RP [última consulta: 23/abril/2014]

[6] MathWorks, Inc., XML and MATLAB: Navigating a Tree. [En línia]. Enllaç web: http://blogs.mathworks.com/community/2010/11/01/xml-

and-matlab-navigating-a-tree/ – Permet obtenir la informació d'un XML en forma d'abre [última consulta: 02/juny/2014]

[7] José A. Mañas, Análisis de Algoritmos: Complejidad. [En línia]. Enllaç web: http://www.lab.dit.upm.es/~lprg/material/apuntes/o/ – Detalla com duu a terme un anàlsis de la complexitat algorítmica. [última consulta: 15/juny/2014]

[8] MathWorks, Inc, Access Data. [En línia]. Enllaç web: http://www.mathworks.es/es/help/matlab/access-data.html – Conté tot un seguit de funcions C que es poden utilitzar desde MatLab i que permeten llegir i escruire en arrays [última consulta: 15/juny/2014]

[9] cplusplus.com, Standard C++ Library reference. [En línia]. Enllaç web: http://www.cplusplus.com/reference – Conté la descripció de mètodes bàsics de C++ [última consulta: 23/juny/2014]

[10] K. Van de Sande, J. Uijlings, T. Gevers, and A. Smeulders. Segmentació amb el mètode 'selective search' (cerca selectiva) pel reconeixement d'imtages. In ICCV, 2011.

[11] B. Alexe, T. Deselaers, and V. Ferrari. Mesura de la 'objectualitat' (objectness) de les finestres d'imatge. IEEE Trans. on PAMI, 2012.

[12] E. Rahtu, J. Kannala, and M. Blaschko. Aprenent un sistema de detecció d'objectes independent de la categoria. In ICCV, 2011.

[13] Mark Everingham, Luc van Gool, Chris Williams, John Winn, Andrew Zisserman, The PASCAL Visual Object Classes Challenge 2007. [En línia]. Enllaç web: http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2007/index.html - Conté informació referent al conjunt de dades de PASCAL VOC2007 empleat en les proves d'aquest article [últim consulta: 20/juny/2014]