Label map BIA

7 messages

Nikolas Moya <nikolasmoya@gmail.com> Wed, Oct 1, 2014 at 5:23 PM
To: Thiago Spina <thiago.spina@gmail.com>

Oi Thiago,

Preciso de um help/opinião.

Em algum lugar está descrito como que é organizado os dados em memória do label map do BIA?

Estou a um tempão quebrando a cabeça. Já descobri algumas coisas, por exemplo.
Em um único pixel (int) de APP->Data.label, é salvo um número. Partindo deste número, é possível chegar em um código do marcador e no label. Eu sei que todas estas operações estão implementas, mas não sei qual chamar quando.

O problema é um pouco mais complexo, pois agora é possível chamar as funções da nova IFT no VISVA. Como eu não iria conseguir trocar toda a estrutura por baixo dos panos do VISVA para a nova IFT, eu fiz uma função iftImage2Scene, dai eu posso chamar as funções da IFT nova e depois só atualizo as estruturas do VISVA.

Por exemplo, a Watershed recebe uma floresta e um labeled set. O label map resultante disto, é 0 para fundo, 1 para objeto 1, etc..
É possível converter de forest->label para Scene e atribuir em APP->Data.label para renderizar. Isto funciona. Porém, APP->Data.label também será 0 para fundo, 1 para objeto 1, etc.., ficando fora do padrão da VISVA/BIA. Eu já tive alguns problemas com a cor dos labels com isto, mas consegui solucionar.

Agora o problema é outro, na hora de selecionar para excluir um marcador, a função que estou usando como referência, feita pelo Paulo Miranda, pega o valor do inteiro da imagem de label, separa o label e o código. Com isto, pelo que entendi, ele consegue inverter a cor da zona de influência apenas do marcador clicado e não de todos os marcadores daquele mesmo label. Como no meu caso a APP->Data.label tem 0, 1, 2... Ao clicar para exlcuir um marcador, alguma operação dos bits dá errado e ele pinta com a cor invertida todos os marcadores daquele label.

Eu coloquei diversos prints espalhados no código do Paulo para tentar entender o armazenamento, por exemplo.
Um pixel do objeto 1 na imagem de label, tem valor 1536. Um pixel de outro objeto tem valor 1792. (256 a mais que 1536).
Mas isto é a imagem de label, os marcadores não deveriam estar lá. Está bem confuso para mim como que ele relaciona marcadores / label. Cada marcador deve ter um código, uma cor, uma posição e um label. A imagem de labels deve ter um código (do marcador?) e o label.

Eu estou com um módulo separado, o primeiro módulo que usa funções da IFT nova nas estruturas do VISVA/BIA. Acho que a única coisa que falta resolver é: na hora de clicar em um marcador para excluir, ele pintar a zona de influencia apenas do marcador clicado e não de todos os que compartilham o mesmo label.

Ah, e vc sabe se o BIA coloca toda a árvore do marcador no conjunto de sementes para remoção? Pois nos meus testes, parecia que ele só colocava aquele pixel clicado.

Muito Obrigado!

--
Nikolas Moya - Campinas / Brasil
Computer Science - Unicamp

Thiago Spina <thiago.spina@gmail.com> Wed, Oct 1, 2014 at 7:23 PM
To: Nikolas Moya <nikolasmoya@gmail.com>

Oi Nikolas,

Vamos lá, eu nunca mexi com o BIA, mas tenho uma ideia de como o Paulo pensa por conta do mflow/caos. No caos o Paulo guarda duas imagens distintas:

1) Label: contém o rótulo de segmentação

2) SeedMkMap (ou coisa do gênero): nesta imagem é guardado um ID crescente a partir de 1 que representa o marcador desenhado pelo usuário. Neste caso, cada brush stroke criado pelo usuário tem um ID novo associado. Isto ocorre dentro das funções OnLeftClick/OnRightClick do mouse handler. Deste modo, quando o usuário quer eliminar um marcador colocado em posição ruim o Paulo primeiro seleciona todos os pixels do marcador como sementes para remoção e guarda em um iftSet* da vida. Enqaunto o usuário não clica em Run (botão do meio do mouse no BIA se não me engano) a árvore que será removida e/ou marcador muda tem a cor de seu rótulo invertida para display da área que será corrigida.

Quando Run é acionado ele pega este conjunto de sementes e passa para a DIFT remover. No BIA deve ser a mesma coisa, mas para economizar espaço ele guarda o ID do marcador e da label em um mesmo inteiro. Lembro que vimos isso juntos no LIV. Se não me engano os primeiros 24 bits guardam o ID do marcador e os últimos 8 guardam até 255 rótulos possíveis (a partir de 0).

Pelo que vejo, as funções mais importantes são estas:

// Arquivo modules/segmentation/interactive/src/addhandler.cpp

```
void AddHandler :: OnLeftDrag(int p, int q){
  AdjRel *A;
  int o,cod;

  o = mod->obj_sel;
  if(o<0) return;
```

// Aqui ele calcula o código como dito acima (24 bits pra ID e 8 pra label).

```
  cod = mod->GetCodeValue(mod->markerID, o+1);
  A = mod->GetBrush();
```

// Aqui ele marca o ID do marcador e a label na imagem seedIdLabel, que só serve pra dizer os pixels que foram selecionados como semenets ou não foram selecionadas

```
  APP->AddSeedsInBrushTrace(p,q,o+1,mod->markerID,
        A,axis);
```

// Aqui ele marca o mesmo ID/label anterior na Data.label. Então na hora de rodar a DIFT a label e ID já foram setados corretamente e serão propagados

```
  APP->DrawBrushTrace(p,q,cod,A,axis);
```

// Aqui ele seta o custo das sementes para INT_MAX, indicando que parece ser uma IFT com Fmin

```
  //Reset the cost to force seed execution:
  APP->DrawBrushTraceCustom(p,q,INT_MAX,A,axis,mod->cost);

  DestroyAdjRel(&A);
  APP->Refresh2DCanvas();
```

```
}
```

// Esta função adiciona um seed e é chamada por AddSeedsInBrushTrace

```
void  BIAClass::AddSeed(int p, int label, int id)
{
    int mk;
    if (id == 0) return;
```

// Aqui ele verifica se o pixel já foi marcado como semente e, se foi, guarda em um seedSet

```
    if (seedIdLabel->data[p] == 0)
        InsertSet(&seedSet, p);
    mk = _GetMarkerValue(id, label);
    seedIdLabel->data[p] = mk;
}
```

O seedSet guarda o ID das sementes selecionadas pelo usuário. A Imagem seedIdLabel guarda tanto o rótulo da semente marcada quanto o ID do marcador como descrito acima. Para tanto, ele utiliza as funções abaixo:

```
int BIAClass :: _GetMarkerID(int mk)
{
    return (mk >> 8); //(int)label/256;
}

int BIAClass :: _GetMarkerLabel(int mk)
{
    return (mk & 0xff);  //(label % 256);
}
```

// Esta função é a mesma coisa que GetCodeValue para a DIFT. Talvez em outros módulos signifique algo diferente

```
int BIAClass :: _GetMarkerValue(int id, int lb)
{
    return ((id << 8) | lb); //(id*256 + (int)lb);
}
```

O Paulo fez isto por dois motivos:

1) Não tínhamos um iftLabeledSet para ao menos guardar a label do marcador.

2) Como ele precisa verificar se um pixel já foi marcado como sementena hora de adicionar mais um pixel como semente **(if (seedIdLabel->data[p] == 0))**, ele precisa de um modo eficiente para fazê-lo. Portanto, ele guarda esta imagem seedIdLabel para fazer consultas rápidas ao seedSet.

Entendeu o processo de adição de marcadores?

Agora o que importa na hora de executar a DIFT se encontra na função:

```
void ModuleInteractive :: Run_DIFT()
{
    Set *S = NULL;
    Set *delSet = NULL, *aux;
    int id, cod;
```

/ Aqui ele copia as sementes marcadas para inserção e os IDs de MARCADORES para remoção **/**

```
    S = APP->CopySeeds();
```

```
    delSet = APP->CopyIdsMarkedForRemoval();


    /** Daí ele percorre o set com IDs de marcadores adicionando a label junto para ter o código apropriado de cada um **/
    aux = delSet;
    while (aux != NULL)
    {
        id = aux->elem;
        cod = GetCodeValue(id, APP->GetSeedLabelById(id));
        aux->elem = cod;
        aux = aux->next;
    }

    /** Neste ponto, como os custos das sementes já foram setados apropriadamente para 0 (ou INT_MAX se a IFT for a partir de domos), dentro de RunDIFT (lib/code_cpp/src/bia_ift.cpp) ele simplesmente não escolhe como fronteira pixels que sejam sementes, analisando os códigos de marcadores selecionados para remoção, escolhendo para remoção todos os outros pixels de árvores enraizadas nos marcadores selecionados. Vide ForestRemoval em bia_ift.cpp
    **/
    RunDIFT(this->Wf, cost,
            pred, APP->Data.label,
            &S, &delSet);

    APP->DelMarkedForRemoval();
    DestroySet(&S);
    DestroySet(&delSet);
}
```

Entendeu? Neste caso, vejo duas possibilidades:

1) Você adapta a sua função para propagar o resultado de cod = mod->GetCodeValue(mod->markerID, o+1), que deve ser guardado na imagem de label.

2) Na hora que você pegar as sementes você separa a label delas e faz a segmentação via libIFT. Em seguida, você analisa todos pixels colocando o código correspondente ID/label da raiz deste pixel em Data.label.

[]'s

2014-10-01 17:23 GMT-03:00 Nikolas Moya <nikolasmoya@gmail.com>:
[Quoted text hidden]

--
Thiago Vallin Spina
PhD candidate
Laboratory of Visual Informatics
Institute of Computing -- Unicamp
http://www.ic.unicamp.br/~tvspina

Thiago Spina <thiago.spina@gmail.com> Wed, Oct 1, 2014 at 7:28 PM
To: Nikolas Moya <nikolasmoya@gmail.com>

O que eu faria:

1) Alteraria a função para supor que as sementes já foram marcadas apropriadamente, alterando apenas o custo

delas dentro da função para 0 e colocando-as na fila. É assim que faço no Cavos, então posso propagar o que eu quiser em label.

[]'s

2014-10-01 19:23 GMT-03:00 Thiago Spina <thiago.spina@gmail.com>:
[Quoted text hidden]
[Quoted text hidden]

Nikolas Moya <nikolasmoya@gmail.com> Thu, Oct 2, 2014 at 2:23 PM
To: Thiago Spina <thiago.spina@gmail.com>

Nossa cara,

vou emoldurar este email na parede.
hahaahha, sério, acho que vou colocar na pasta do visva como documentação para futuros alunos. Me ajudou pacas!

Eu decidi por codificar as sementes pela função mod->GetCodeValue(mod->markerID, o+1), assim elas já terão o código como label e serão copiadas corretas para Data.label depois de rodar a segmentação pela ift nova. Isto resolveu o problema da cor de múltiplos objetos e na hora de selecionar um marcador, ele pintar apenas o marcador clicado e não todos os marcadores daquele label. Era bem os problemas que precisavam ser resolvidos.

O trecho de código da remoção ainda não faz muito sentido para mim. Eu imagino que tenha que fazer algo similar a isto:

```
delSet = APP->CopyIdsMarkedForRemoval(); /** Aqui ele copia [...] os IDs de MARCADORES para remoção **/
tmp = delSet;
while (tmp != NULL)
{
    p = FUNCAO_ID_TO_VOXEL_INDEX(tmp->elem);
    iftInsertLabeledSet(&LS, p, NIL);
    tmp = tmp->next;
}
```

Na IFT nova, as sementes de remoção recebem label -1. Desta forma, eu acho que para a remoção eu não posso codificar o ID e o Label e passar como rótulo da semente, igual eu fiz na inserção. Se eu fizesse isto, esta função de decodificação deveria estar na IFT nova para verificar se é um label -1 ou não.
O problema é que não descobri como que eu converto do ID recebido para o indice do voxel. Mesmo que eu codifique (ID + Label), não tem como converter deste código para o indice na imagem.

Eu poderia mudar a função APP->MarkForRemoval(cod); dentro do handler de remoção DelHandlerDRIFT :: OnLeftClick, para ao invés de armazenar o 'cod' armazenar o indice 'p', que é onde estava o mouse na imagem no momento do clique. Eu fiz isto, mas não deu certo. Ele executou sem erros, mas a região marcada para remoção continuou aparecendo e o mapa de labels não foi atualizado.

Eu acho que resolvendo isto, a DiffWatershed pega a árvore inteira das sementes com rótulo -1 pela função iftTreeRemoval, deixa a região com custo 'disponivel para conquista', e os labels vizinhos vão competir e propagar nesta região. E como os labels vizinhos já estarão codificados, tudo funcionará.

2014-10-01 19:28 GMT-03:00 Thiago Spina <thiago.spina@gmail.com>:
[Quoted text hidden]
[Quoted text hidden]

Thiago Spina <thiago.spina@gmail.com> Thu, Oct 2, 2014 at 2:35 PM
To: Nikolas Moya <nikolasmoya@gmail.com>

2014-10-02 14:23 GMT-03:00 Nikolas Moya <nikolasmoya@gmail.com>:

> Nossa cara,
>
> vou emoldurar este email na parede.
> hahaahha, sério, acho que vou colocar na pasta do visva como documentação para futuros alunos. Me ajudou pacas!

Hehe, glad I could help =).

> Eu decidi por codificar as sementes pela função mod->GetCodeValue(mod->markerID, o+1), assim elas já terão o código como label e serão copiadas corretas para Data.label depois de rodar a segmentação pela ift nova. Isto resolveu o problema da cor de múltiplos objetos e na hora de selecionar um marcador, ele pintar apenas o marcador clicado e não todos os marcadores daquele label. Era bem os problemas que precisavam ser resolvidos.
>
> O trecho de código da remoção ainda não faz muito sentido para mim. Eu imagino que tenha que fazer algo similar a isto:

```
delSet = APP->CopyIdsMarkedForRemoval(); /** Aqui ele copia [...] os IDs de MARCADORES para remoção **/
tmp = delSet;
while (tmp != NULL)
{
    p = FUNCAO_ID_TO_VOXEL_INDEX(tmp->elem);
    iftInsertLabeledSet(&LS, p, NIL);
    tmp = tmp->next;
}
```

Então, seu raciocínio está quase correto, mas a sua resposta está na própria função ForestRemoval do VISVA abaixo.

//Como ele só guarda os IDs de marcadores selecionados para remoção, você vai ter que percorrer o seed set lido com S = APP->CopySeeds(); vendo para cada semente se o ID dela foi selecionados para remoção. Neste caso, você inverteria os loops while(D!=NULL) e while (S!= NULL) abaixo, inserindo em seguida com iftInsertLabeledSet o pixel p como NIL assim como vc fez acima. É feio, mas funciona. A passagem de NIL para indicar sementes a serem removidas é muito mais elegante, mas considerando o código do VISVA atual o modo como descrevi é o que deve causar menos bugs.

```
/*
   Tenho que ler delSet e varrer seedSet removendo todas
   sementes marcadas, setando custo para INT_MAX, etc.
   Tenho tambem que colocar essas raizes numa fila rootSet
   para a etapa seguinte.
*/

D = *delSet;
while(D!=NULL){
  del_mk = D->elem;

  S = seedSet;
```

```
    while(*S!=NULL){
      p = (*S)->elem;
      mk = mark->data[p];
      // Sementes nao podem ser fronteira.(old)
      //_fast_BMapSet1(Fcolor, p);

      if(mk == del_mk){
        InsertSet(&rootSet, p);
        cost->data[p] = USHRT_MAX;
        pred->data[p] = NIL;
        mark->data[p] = 0;
        tmp = *S;
        *S = (*S)->next;
        free(tmp);
      }
      else
        S = &((*S)->next);
    }
    D = D->next;
  }
```

[Quoted text hidden]
[Quoted text hidden]

Nikolas Moya <nikolasmoya@gmail.com> Thu, Oct 2, 2014 at 3:15 PM
To: Thiago Spina <thiago.spina@gmail.com>

Há, saquei a ideia.

Acabei de implementar and it worked like a charm! Cara, muitooo obrigado de novo. Funcionou perfeitamente!

2014-10-02 14:35 GMT-03:00 Thiago Spina <thiago.spina@gmail.com>:
[Quoted text hidden]
[Quoted text hidden]

Thiago Spina <thiago.spina@gmail.com> Thu, Oct 2, 2014 at 3:17 PM
To: Nikolas Moya <nikolasmoya@gmail.com>

No problemo =). Qualquer coisa é só avisar.

2014-10-02 15:15 GMT-03:00 Nikolas Moya <nikolasmoya@gmail.com>:
[Quoted text hidden]
[Quoted text hidden]