گزارش توضیح مقاله به فارسی

موضوع مقاله:

Unsupervised image segmentation using a simple MRF model with a new implementation scheme

کسری اسکندری

استاد درس: دکتر ابریشمی مقدم

مرداد ۱۴۰۰

چکیده

در این گزارش به بررسی و پیاده سازی مقاله[۱] میپردازیم.هدف اصلی در این مقاله ارائه یک روش برای قطعه بندی معنایی تصویر برپایه میدان تصادفی مارکوفی[۱] می‌باشد. در قطعه بندی معنایی به هرپیکسل یک برچسب نسبت داده می‌شود تا مشخص شود هر ناحیه از تصویر مربوط به کدام کلاس است.در این مقاله این کار بدون استفاده از داده‌های تمرینی و به صورت یادگیری بدون ناظر انجام می‌شود.روش ارایه شده در مقاله از یک روش موسوم به وزن دهی از طریق توابع را انجام میدهد تا بتواند به صورت بدون ناظر پارامترهای مدل را پیش‌بینی کند و قطعه بندی قابل قبولی ارائه دهد.

[۱] Markov random field

فهرست

یک مدل ساده برپایه میدان تصادفی مارکوف برای قطعه بندی

مسئله قطعه بندی را میتوانیم توسط یک چهارچوب بیزی مطرح کنیم. در اینجا میتوانیم هر پیکسل را به عنوان یک متغیرتصادفی درنظر بگیریم که مقدار این متغیر تصادفی یک عدد صحیح بین صفر تا ۲۵۵ میباشد. همچنین $Y = y$ برچسب مورد نظر برای پیکسل مورد نظر میباشد.

$$P(Y = y|F = f) = \frac{p(F = f|Y = y)P(Y = y)}{p(F = f)},$$

به طوری که $P(Y = y|F = f)$ برابر با احتمال پسین برچسب‌ها به شرط پیکسل موردنظر میباشد. همچنین $P(F = f|Y = y)$ برابر با توضیع احتمال مقدار هر پیکسل به شرط برچسب مورد نظر میباشد. همچنین $P(Y = y)$ برابر بااحتمال پیشین مشاهده لیبل مورد نظر میباشد و در اخر $P(F = f)$ برابر با توضیع احتمالاتی تصویر میباشد که نسبت به برچسب‌های متفاوت تغییری نمیکند.

برای اینکه بتوان از قضایای قضیه بیز استفاده کزد باید دو فرض را انجام داد. اولین فرض انجام شده در این مقاله درمورد استقلال مقادیر پیکسل‌های تصویر میباشد. حال اگر فرض کنیم تصویر مورد نظر K قطعه در ویژگی هایش داشته باشد داریم(جلوتر خواهیم دید این ویژگی‌های مورد استفاده در مقاله مقدار سطح خاکستری در عکس‌های سیاه سفید است و مقدار HSV در تصاویر رنگی)

$$P(Y = y|F = f) = \frac{\prod_{k=1}^{K}[p(f^k|Y = y)]P(Y = y)}{p(F = f)},$$

در عموم روش‌های قطعه بندی از یک تابع انرژی استفاده می‌شود و هدف اصلی کاهش دادن این تابع انرژی می‌باشد. این مقاله هم روش مشابهی اختیار کرده و یکی از جملات تابع انرژی به صورت زیر تعریف می‌شود:

$$E_R(y) = \sum_s \left[\beta \sum_{t \in N_s} \delta(y_s, y_t) \right],$$

به طوری که:

$$\delta(y_s, y_t) = \begin{cases} 1 & y_s \neq y_t \\ -1 & y_s = y_t \end{cases}$$

همچنین N_s نمایان‌گر همسایگی‌های پیکسل S میباشد.(در این پیاده سازی از ۸ همسایگی استفاده کرده‌ایم)

مفهوم کاهش دادن این مقدار این است که پیکسل‌های مجاور مقدار برابری داشته باشند و با انجام دادن این کار باعث می‌شود اندازه نواحی قطعه بندی شده بزرگ‌تر بشوند چون عموما در تصاویر نواحی کوچک و تودرتو به عنوان قطعه بندی مناسب شناخته نمیشوند.

مقدار βهم به عنوان یک پارامتر استفاده می‌شود تا اهمیت این جمله را کنترل کنیم، که جلوتر مقاله این مقدار را یک اختیار میکند و با پارامترهای دیگر اهمیت این جمله را کنترل میکند.

با داشتن این مقدار انرژی و استفاده از میدان‌های تصادفی گیبس[۲] می‌توانیم توضیع برچسب هارا به این نحو تعریف کنیم:

[۲] Gibbs random fields

$$P(Y = y) = \frac{1}{Z_R} \exp\left[-\frac{1}{T} E_R(y)\right]$$

به طوری که مقدار Z_R به این صورت استفاده میشود.(توجه شود این ترم فقط نقش نرمالایز مقدار خروجی‌ها را دارد تا بتوانیم از تابع اخیر به عنوان یک توضیع احتمالاتی یادکنیم و جلوتر درمعادلات حذف خواهد شد)

$$Z_R = \sum_{y \in \Omega_y} \exp\left[-\frac{E_R(y)}{T}\right]$$

حال تنها $P(f^k|Y = y)$ برای ما مجهول است. در اینجا برای پیدا کردن این مقدار مقاله فرض دوم خود را درمورد تصویر مطرح میکند و فرض میکند همه لیبل‌های تصویر یک توضیع گوسی استفاده میکنند که هرکدام از این کلاس‌ها میانگین و انحراف معیار خاص خود را دارد.توجه شود احتمال مشاهده توضیع‌های گوسی در تصویر بسیار پایین است اما مقاله از این توضیع برای تخمین توضیع برچسب های موجود در تصویر استفاده می‌کند.

پس میتوان احتمال پسین هرپیکسل را به این صورت نمایش داد:

$$p(f_s^k | Y_S = m) = \frac{1}{\sqrt{2\pi {\sigma_m^k}^2}} \exp\left[-\frac{(f_s^k - \mu_m^k)^2}{2\sigma_m^{k2}}\right]$$

به طوری که μ_m برابر با میانگین برچسب mام و مشابها σ_mبرابر با انحراف معیاربرچسب mام میباشد.

پس میتوان انرژی حاصل از پرچسب دادن به پیکسل‌هارا به این صورت تعریف کرد:

$$E_F = \sum_{s,m=Y_s} \left\{ \sum_{k=1}^{K} \left[\frac{(f_s^k - \mu_m^k)^2}{2(\sigma_m^k)^2} + \log(\sqrt{2\pi}\sigma_m^k) \right] \right\}$$

و در اخر می‌توان انرژی کل تصویر را به صورت جمع وزن‌دار این دو انرژی تعریف کرد:

$$E = E_R + \alpha E_F$$

همچنین می‌توان احتمال گیبس این انرژی را به این صورت نشان داد:

$$P(Y = y|F = f) = {}^{1}/_{Z} \exp\left[-\frac{E}{T}\right]$$

$$Z = \sum_{y \in \Omega_y} \exp\left[-\frac{E}{T}\right]$$

ترفندهای پیاده‌سازی

برآوردگر بیشینه‌گر احتمال پسین[۳] و الگوریتم تبرید شبیه‌سازی‌شده[٤]

یکی از مهم ترین مسائل در پیاده سازی میدان‌های احتمالی مارکوف بحث برآوردگر بیشینه‌گر احتمال پسین می‌باشد [۲].

[۳] Maximum a posteriori (MAP)
[٤] Simulated annealing

$$\hat{y} = \arg \max_{y \in \Omega_Y} P(Y = y | F = f)$$
$$= \arg \max_{y \in \Omega_Y} \frac{1}{Z} \exp\left[-\frac{1}{T} E\right]$$
$$= \arg \min_{y \in \Omega_Y} E.$$

باتوجه به معادله اخیر برای افزایش احتمال یک برچسب گذاری باید انرژی برچسب گذای مذکور را کاهش دهیم.

توجه شود تابع انرژی معرفی شده در این مقاله محدب[۵] نبوده پس پیدا کردن مینیمم سراسری کار ساده‌ای نبوده و ممکن است به مینیمم‌های محلی برسیم. برای رفع این مشکل مقاله موردنظر از الگوریتم متروپلیس-هیستینگز[۶] استفاده کرده است.

در ادامه به برسسی الگوریتم اخیر میپردازیم که در مقالات [٤] ,[٢] ,[٣] ذکر شده است.

الگوریتم متروپلیس-هیستینگز

در ابتدا حالت کلی این الگوریتم را توضیح میدهیم سپس به بررسی حالت خاص آن که در این مقاله مورد استفاده قرار گرفته است میپردازیم.

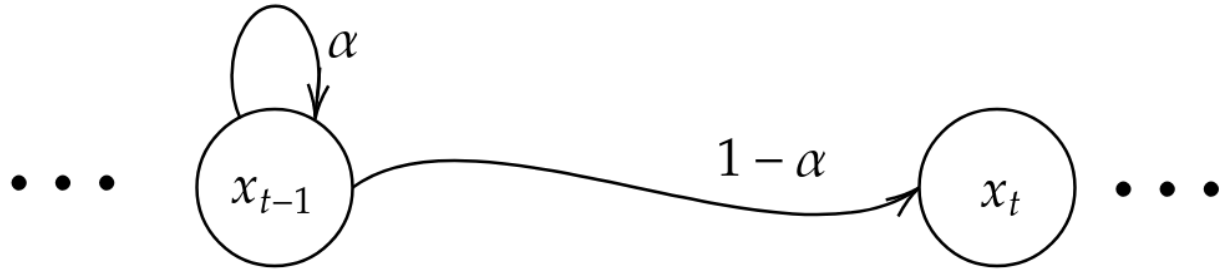
این الگوریتم یک الگوریتم تکراری بوده و درهرگام حالت بعدی سیستم را حدس میزند و بایک ازمون ساده تصمیم میگیرد که وضعیت سیستم را تغییر دهد یا خیر. این الگوریم را میتوان به صورت یک زنجیره احتمالاتی مارکوف به این صورت نمایش داد. به این صورت که در هرگام یک

---

[۵] convex

[۶] Metropolis–Hastings algorithm

تخمین از حالت بعدی پیش‌بینی میکند و با یک احتمال خاص تصمیم میگیرد که به حالت بعدی برود یا در همان حالت بماند.



الگوریتم:

۱. مقدار اولیه برای x در نظر بگیر

۲. برای مقادیر t در بازه یک تا N

a. مقدار بعدی سیستم را باتوجه به حالت فعلی سیستم حدس بزن $x^* \sim q(x^*|x_{t-1})$

b.
$$\alpha = \frac{g(x^*)q(x_{t-1}|x^*)}{g(x_{t-1})q(x^*|x_{t-1})}$$

c.
$$\begin{cases} iff\ \alpha \geqslant 1 :\ x_t = x^* \\ iff\ 0 < \alpha < 1 : \begin{cases} x_t = x^*\ with\ probability\ \alpha \\ x_t = x_{t-1}\ with\ probability\ 1-\alpha \end{cases} \end{cases}$$

به طوری که x نمایان‌گر وضعیت سیستم، $f(x)$ برابر با انرژی سیستم، $g(x)$ برابر با تابع چگالی توضیع انرژی و T انرژی سیستم میباشد.

حال برای استفاده از الگوریتم میتوانیم در این مقاله میتوانیم به جزییات بیشتر این الگوریتم بپردازیم.

فرض کنید تابع $q(x|y)$ یک توضیع نرمال با میانگین y باشد پس میتوان برای تخمین حالت بعدی سیستم چنین عمل کنیم:

$$x^* = x_{t-1} + rand()$$

همچنین چون تابع نرمال نسبت به میانگین خود قرینه می‌باشد، داریم:

$$q\left(x_{t-1}|x^*\right) = q\left(x^*|x_{t-1}\right)$$

طبق محاسبات مقاله داریم:

$$g(x) = \frac{1}{Z}\exp\left(-\frac{E}{T(t)}\right)$$

همچنین برای تایین کردن مقدار دمای سیستم روش‌های مختلفی درمقاله مطرح شده است که در نهایت از عبارت زیر استفاده می‌شود

$$T(t) = \frac{C}{\log(t+1)},\ C = 2$$

همچنین مقدار ثایت مورد استفاده در این عبارت مقدار ثایت ۲ در نظر گرفته شده است. همچنین جلوتر خواهیم دید که این الگوریتم در میانه الگوریتم EM استفاده می‌شود پس نیازی به مقدار دهی اولیه نیست.

می‌توان الگوریتم جدید به این صورت بازنویسی کرد:

۳. مقدار اولیه برای x را مقداردهی کن
٤. برای مقادیر t در بازه یک تا N
   a. مقدار بعدی سیستم را باتوجه به حالت فعلی سیستم حدس بزن

$$x^* = x_{t-1} + rand()$$

$$\alpha = \frac{g(x^*)}{g(x_{t-1})} = \exp\left(\frac{E(x_{t-1})}{T(t-1)} - \frac{E(x^*)}{T(t)}\right) = \exp\left(\frac{\left(E(x_{t-1}) * \log(t) - E(x^*) * \log(t+1)\right)}{C}\right)$$ .b

$$x_t = \begin{cases} x^* & \alpha > Rand() \\ x_{t-1} & otherwise \end{cases}$$ .c

در ادامه لازم به ذکر است که $x = (\mu, \sigma)$ فرض شده است.(این موضوع در هیچ جای مقاله به صورت واضح مطرح نشده ولی درحین پیاده سازی این فرض درنظر گرفته شده است) و هربار با تعییر میانگین و انحراف معیار کلاس‌ها میتوان برچسب گذاری های جدید را با استفاده از این عبارت محاسبه کرد:

$$p(f_s^k | Y_s = m) = \frac{1}{\sqrt{2\pi {\sigma_m^k}^2}} \exp\left[-\frac{(f_s^k - \mu_m^k)^2}{2\sigma_m^{k2}}\right]$$

درواقع برچسبی که بیشترین احتمال برای پیکسل دارد را به پیکسل موردنظر میدهیم.

تخمین پارامترها

در این الگوریتم باید چهار پارامتر مقدار دهی بشود. برای تایین کردن میانگین و انحراف معیار نیاز به داده های اموزشی داریم اما با توجه به بدون ناظر بودن الگوریتم پیشنهادی، دادگان اموزشی در اختیار نیستند پس برای تخمین پارامترهای مذکور از الگوریتم الگوریتم امید ریاضی–بیشینه کردن[۷] استفاده شده است:

این الگوریتم در زیر شرح داده شده است:

۱. یک برچسب گذاری شانسی برای تصویر پیشنهاد میدهیم

[۷] Expectation–maximization algorithm(EM algorithm)

۲. گام-الف (امید ریاضی): مقادیر انحراف از معیار و میانگین هرکلاس را با استفاده از تصویر و برچسب گذاری موجود و فرض گاوسی بودن توضیع برچسب ها حساب کن

$$\mu_m^k = \frac{1}{N} \sum_{s, Y_s = m} f_s^k \,,\; \sigma_m^k = \left[\frac{1}{N-1} \sum_{s, Y_s = m} (f_s^k - \mu_m^k)^2 \right]^{\frac{1}{2}}$$

۳. گام-ب (بیشینه کردن): از میانگین و انحراف معیارهای به دست امده برای کاهش دادن مقدار انژی استفاده کن.

٤. مراحل فوق را تارسیدن به شرط پایان ادامه بده.

تنها دو پارامتر دیگر برای تخمین باقی میماند. همان طور که قبلا هم اشاره شد مقدار βرا برابر یک قرار میدهیم و برای مقدار دهی به αگزینه های مختلفی وجود دارد . میتوان مقدار ثابت را درنظر گرفت اما این مقاله مقدار جدیدی برای این پارامتر پیشنهاد داده که ادعا میکند بهتر از روش های موجود عمل میکند:

$$\alpha(t) = c_1 0.9^t + c_2$$

که در آن مقدار برای مقادیر c_1, c_2 دو پیشنهاد وجود دارد.میتوان هردو را مقدار ثابت $(c_1, c_2) = (80, 1)$ در نظر گرفت یا می‌توان به صورت $(c_1, c_2) = (80, 1/K)$ در نظر گرفت به طوری که K برابر تعداد کلاس‌های تصویر می‌باشد.

پیاده سازی

پیاده سازی این مقاله توسط زبان برنامه نویسی پایتون انجام گرفته و کاملا به صورت شئ‌گرا پیاده شده است.کامنت گذاری ها و راهنمایی برای انواع نوع داده های موجود تا حد امکان انجام شده.

در ادامه به توضیح جزییات پیاده سازی و بهینه سازی های انجام شده که در متن مقاله ذکر نشده بود میپردازیم.

تغییرات میانگین و انحراف از معیار و محسابه برچسب گذاری های جدید

میدانیم با داشتن میانگین و انحراف معیارهای تصویر میتوان یک برچسب گذاری یکتا برای تصویر ارائه داد و مشابها با داشتن برچسب گذاری تصویر می‌توان میانگین و انحراف معیار یکتا برای تصویر محاسبه کرد، همچنین در چند قسمت از الگوریتم مورد بحث مقادیر برچسب گذاری و یا مقادیر میانگین و انحراف از معیار برچسب‌ها تغییر میکرد. محاسبه این مقادیر از روی یک دیگر به صورت ارزیابی کندرو[۸] محاسبه می‌شود. به این صورت که یک متغیر کنترلی وجود دارد که تایین میکند این مقدار میانگین و انحراف از معیار فعلی درست هستند یا خیر و درصورت نیاز این مقادیر محاسبه می‌شوند.

برای جزییات بیشتر میتوانید توابع زیر که با property decorator نوشته شده است مراجعه کنید:

```
@property
  def mu(self):
      if self.__correct_mu_sigma:
          return self.__mu
      self.__mu, self.__sigma = self.__calculate_mu_sigma()
      self.__correct_mu_sigma = True
      return self.__mu

  @property
  def sigma(self):
      if self.__correct_mu_sigma:
          return self.__sigma
      self.__mu, self.__sigma = self.__calculate_mu_sigma()
      self.__correct_mu_sigma = True
```

[۸] Lazy evaluation

```
        return self.__sigma

    @property
    def labels(self):
        return self.__labels

    @labels.setter
    def labels(self, value):
        self.__labels = value
        self.__correct_mu_sigma = False
```

محسابه انرژی بین پیکسل‌ها

همان طور که داخل مقاله مطرح شده بود برای محاسبه انرژی بین کلاس‌ها می‌توانیم از عبارت زیر استفاده کنیم:

$$E_R(y) = \sum_s \left[\beta \sum_{t \in N_s} \delta(y_s, y_t) \right]$$

عبارت اخیر را می‌توان با استفاده دوحلقه تو در تو درپایتون پیاده کرد، اما این کار سرعت اجرای بسیار کمی دارد برای جلوگیری از این مشکل از روش زیر استفاده کرده ایم:

در ابتدا برچسب گذازی های مورد نظر توسط چهار ماتریس زیرهم‌گذاری[۹] می‌شوند.

[۹] convolve

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

سپس از ماتریس های به دست امده مقادیر غیر صفر برابر دو قرار گرفته و همه مقادیر منهای یک می‌شوند و در اخر همه مقادیر موجود در این چهار ماتریس باهم جمع می‌شوند.جزییات بیشتر این پیاده سازی را می‌توانید در تابع زیر مشاهده کنید.

```
  @staticmethod
    def Er(labels, beta=1):
        kernels = [np.array([[0, -1, 0],
                             [0, 1, 0],
                             [0, 0, 0]]),

                   np.array([[0, 0, -1],
                             [0, 1, 0],
                             [0, 0, 0]]),

                   np.array([[0, 0, 0],
                             [0, 1, -1],
                             [0, 0, 0]]),

                   np.array([[0, 0, 0],
                             [0, 1, 0],
                             [0, 0, -1]])]
        Er = 0
        for kernel in kernels:
            tmp = cv2.filter2D(labels, -1, kernel)
            '''
            tmp = 0 iff two corresponding pixels are equal
            tmp!= 0 iff two corresponding pixels are not equal
            as described in paper delta must be -1 if corresponding pixels
are equal otherwise 1
            '''
```

```
            tmp[tmp != 0] = 2
            tmp -= 1
            tmp *= beta
            # if Er is not None:
            #     Er += tmp
            # else:
            #     Er = tmp
            Er += np.sum(tmp)
        return np.sum(Er)
```

پیاده سازی مقادیر α

برای پیاده سازی مقادیر مختلف αورودی تابع باید یک مقدار قابل صدا زدن باشد چون این پرامتر در گام های مختلف می‌تواند مقادیر مختلف داشته باشد. پس مقدار نوع این پارامتر را می‌توان به این صورت نمایش داد.

```
alpha: Callable[[int], float]
```

در متن مقاله سه مدل مختلف برای αپیشنهاد داده شده است که برای شبیه سازی هرحالت یک کلاس جدا معرفی شده و برای ایجاد امکان صدا کردن هریک از نمونه‌های این کلاس از پیاده سازی توابع جادوی[۱۰] استفاده شده است.جزییات بیشتر این پیاده سازی هارا میتوان درزیر مشاهده کرد.

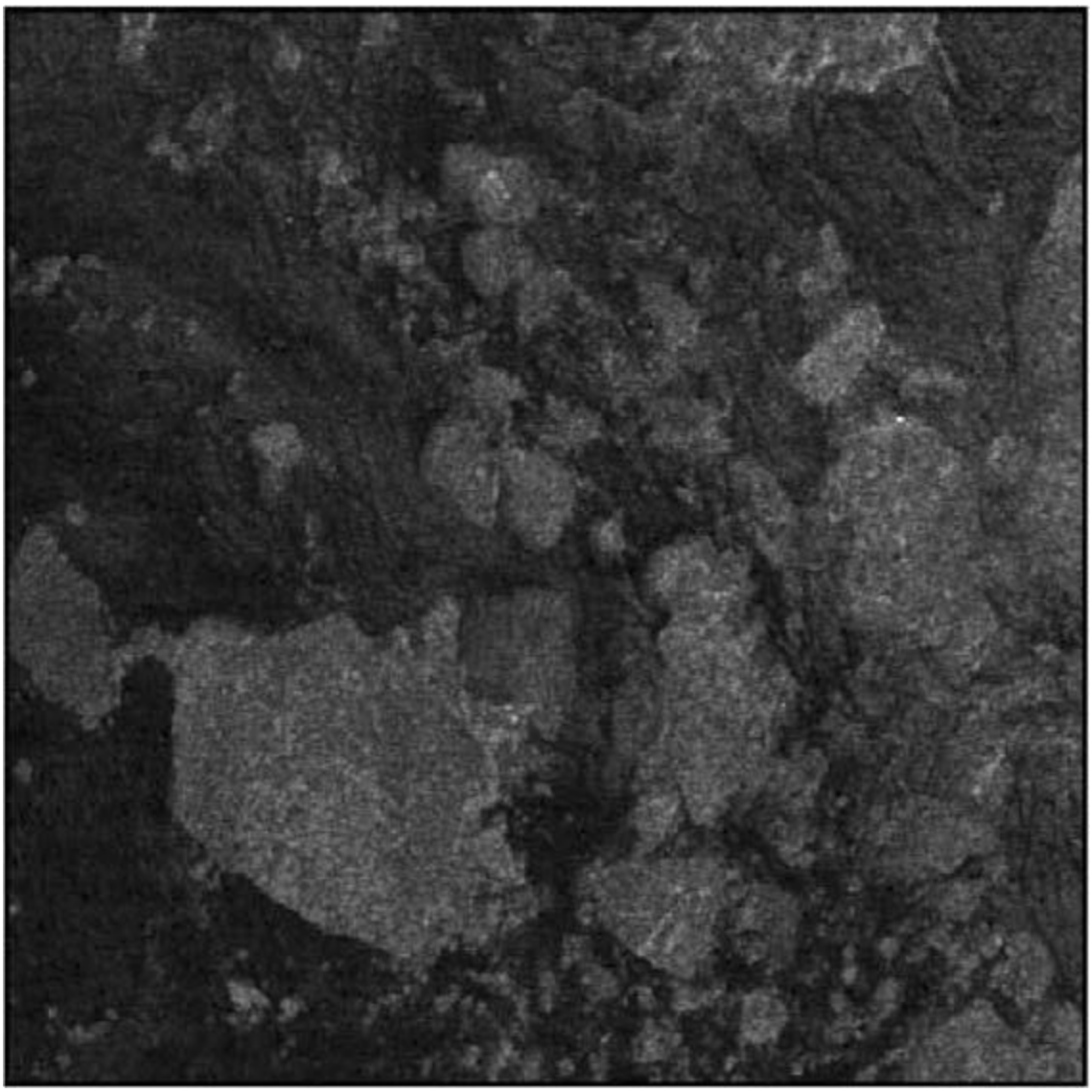
```
class proposed_alpha1:
    def __init__(self, c1=80, c2=1):
        self.c1 = c1
        self.c2 = c2
        self.t = 0

    def __call__(self, t=None):
        if t is not None:
            self.t = t - 1
        self.t += 1
```

[۱۰] Magic functions or dunders

```
        return self.c1 * 0.9 ** self.t + self.c2

    def reset(self):
        self.t = 0


class proposed_alpha2:
    def __init__(self, k, c1=80):
        self.c1 = c1
        self.k = k
        self.t = 0

    def __call__(self, t=None):
        if t is not None:
            self.t = t - 1
        self.t += 1
        return self.c1 * 0.9 ** self.t + 1 / self.k

    def reset(self):
        self.t = 0


class constant_alpha:
    def __init__(self, const):
        self.const = const

    def __call__(self, t=None):
        return self.const

    def reset(self):
        pass
```

آزمایش‌ها و نتایج به دست آمده

در اخر مقاله روش خود را بر روی دیتاست های بسیار متفاوتی ارائه داده که در این بخش به اختصار به بررسی و مقایسه فقط یکی از خروجی‌ها میپردازیم:



شکل ۱- یکی از تصاویر دادگان SAR

شکل ۱ از دادگان SAR است. این دیتاست مجموعه‌ای از تصاویر ماهواره‌ای از یخ‌های قطبی گرفته شده میباشد در این دیتاست دو نوع کلی از تصاویر موجود میباشد، تصاویری که شامل یخ و اب می‌باشد که در این تصاویر مسئله مورد نظر به صورت دوکلاسه مطرح می‌شود، و همچنین دسته دیگری از تصاویر که شامل سه نوع مختلف یخ ها میباشند. پس از اجرای الگوریتم پیاده سازی شده روی شکل ۱ و قطعه بندی آن شکل ۲ حاصل میشود.

شکل ۲ - نتیجه حاصل از قطعه بندی

نتیجه گیری

امروزه قطعه بندی معنایی تصویر به صورت بدون ناظر بسیار پراهمیت و مورد توجه بژوهشگران قرار گرفته است زیرا که تهینه دادگان برچسب گذاری شده کاری بسیار پرهزینه و دشوار می‌باشد. روش‌های بسیار متنوعی برای قطعه بندی تصاویر وجود دارد و استفاده از میدان‌های احتمالی مارکوف یکی از قدیمی‌ترین روش‌های این حوضه میباشد و با توجه به احتمالاتی و اماری بودن

ماهیت روش نمیتوان در تصاویر پیچیده انتظار نتایج بهتری داشت. که البته لازم به ذکر است در سال‌های اخیرپژوهشگران مدل‌های مارکوف را با الگوریتم‌های متفاوتی مانند برش گراف[۱۱] تلفیق نموده‌اند و نتایج قابل توجه و مناسبی به دست اورده‌اند.

[۱۱] Graph cut

فهرست مراجع:

[١] D. A. C. Huawu Deng, "Unsupervised image segmentation using a simple MRF model witha new implementation scheme," ٢٠٠٤.

[٢] D. G. S. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Trans. Pattern Anal,* ١٩٨٤.

[٣] A. R. M. R. A. E. T. N. Metropolis, "Equations of state calculations by fast computing machines,," *J. Chem. Phys,* ١٩٥٣.

[٤] G. Winkler, "Image Analysis, Random Fields and Dynamic Monte Carlo Method," *Springer, New York, USA,* ١٩٩٥.