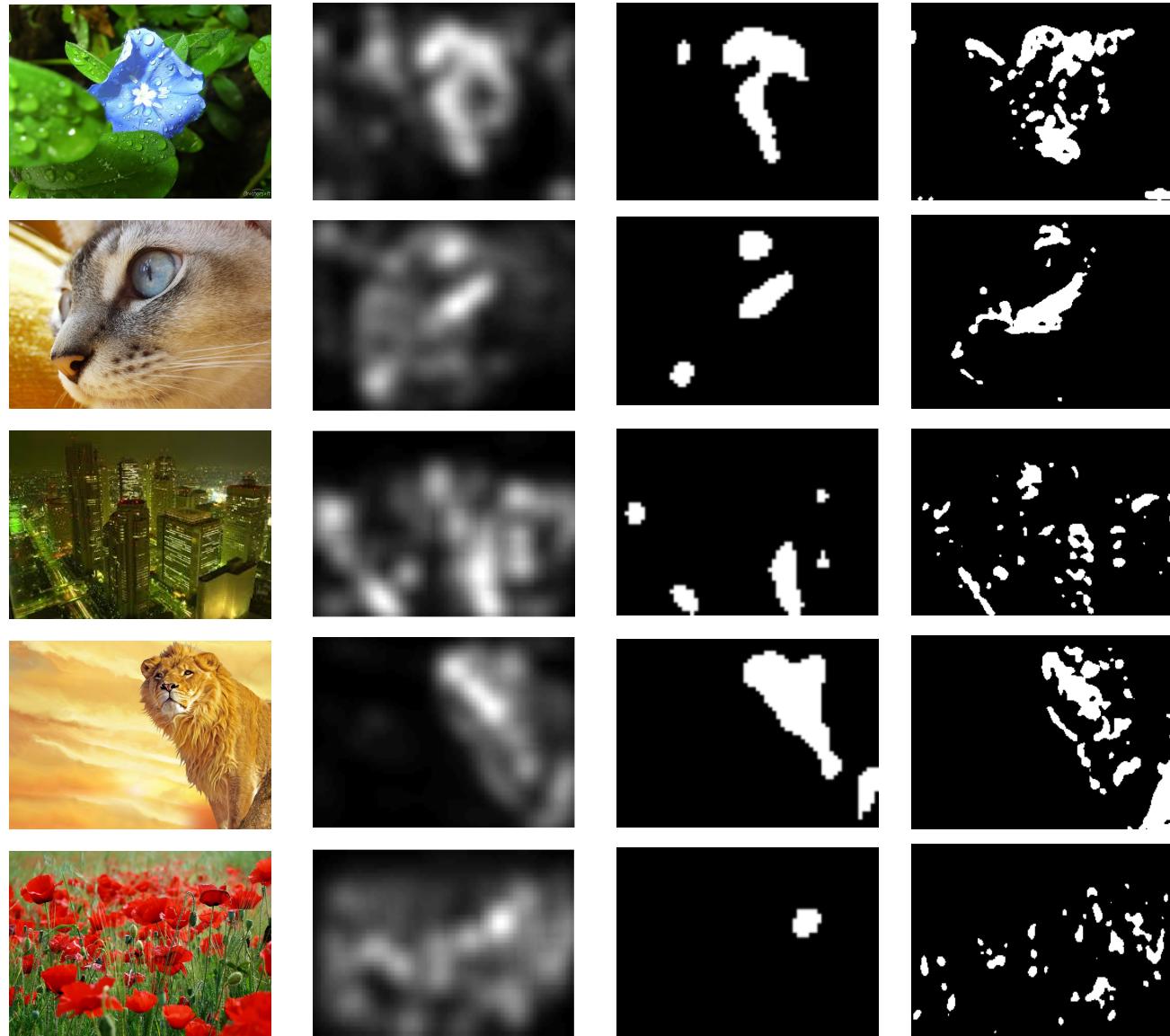


Spectral Residual Approach - Saliency Map

- Se bazeaza pe urmatorul paper: <http://www.klab.caltech.edu/~xhou/papers/cvpr07.pdf>
- Implementare (in Matlab/Octave):
<http://www.klab.caltech.edu/~xhou/projects/spectralResidual/spectralresidual.html>
- Am adaptat si testat algoritmul local si am exemplificat (mai jos) cum functioneaza pe cateva imagini de input. Rezultatele par destul de decente.
- Cum functioneaza (pe scurt):
 - Se calculeaza transformata Fourier 2D a matricei de intensitate a imaginii primite ca input. In Matlab, se foloseste: fft2() (in spate foloseste algoritmul FFT, inseamnand ca functioneaza si eficient) (o alternativa era un algoritm de complexitate mai mare, descris aici: http://www.cs.cf.ac.uk/Dave/Vision_lecture/node20.html). Transformata Fourier 2D se obtine calculand transformata Fourier 1D pentru fiecare coloana (separat) a matricei initiale (fiecare coloana reprezinta un vector), apoi transformata Fourier 1D pentru fiecare linie a matricei rezultante (exact asa functioneaza si fft2()); apeleaza de 2 ori fft()). Observatie: Cum fiecare coloana/linie din matrice este independenta de celelalte => se poate paraleliza.
 - Transformata Fourier descompune un semnal (aproximeaza o functie) intr-o suma de sinusoide. Astfel, pentru o anumita frecventa f, putem afla informatii despre amplitudinea si faza sinusoidei respective. Ne intereseaza informatiile despre amplitudine, mai exact despre "log spectrum"-ul ei. Daca A(f) este matricea de amplitudini (modului fiecarui element obtinut in urma aplicarii transformantei Fourier 2D), matricea "log spectrum" va fi: $L(f) = \log(A(f))$.
 - Daca plotam L(f) (pe o axa frecventa, pe cealalta logaritmul amplitudinii) se observa ca descreste exponential si ca, oarecum, forma grafului tinde sa fie aproximativ aceeasi (plus minus noise). De aceea, se defineste $X(f) = \text{average-}ul$ acestor grafice calculat pentru alte 1.000 (sa zicem) imagini, iar tot ceea ce difera intre X(f) si L(f) (L(f) calculat pentru o imagine concreta) reprezinta particularitatile imaginii de input (pixelii de interes). Cum X(f) nu poate fi calculat concret (ce 1.000 de imagini alegem inainte pentru a predefini X ?!), il aproximam cu: $X(f) = h_n(f) \text{ op } L(f)$, unde op = convolutie si $h_n(f) = 1/n^2 * (\text{matrice nxn plina de 1})$ (putem alege n = 3).
 - Lucrurile specifice imaginii de input (spectral residual) vor fi definite ca: $R(f) = L(f) - X(f)$
 - Saliency map-ul imaginii: $S(x)$, va fi egal cu transformata Fourier 2D inversa aplicata lui R(f), cu un "Gaussian filter" aplicat peste rezultat.
 - Harta obiectelor: $O(x, y)$ se calculeaza astfel:
$$O(x, y) = 1, \text{ daca } S(x, y) > \text{threshold}$$
$$= 0, \text{ altfel}$$
 - Threshold (pentru output-urile de mai jos), a fost ales ca: average(S(x)) * 3. Pot fi experimentate si alte valori.
 - Exact acesti pasi sunt implementati si in codul Matlab din linkul de mai sus. Algoritmul foloseste niste pachete auxiliare, care pot fi instalate insa foarte usor (pkg install -forge image, spre exemplu).

- Cateva exemple (input image/saliency map 64px/object map 64px/object map 256px):



- NOTA: Imaginile au fost scalate la 64px (in cazul primului object map - coloana a 3-a), respectiv la 256px (in cazul celui de-al doilea object map - coloana a 4-a). Rezultatele mai "accurate" se obtin pentru 256, insa timpul de executie este mai mare (aici poate interveni utilitatea paralelizarii). NU faceti mapare pixel-cu-pixel intre imaginea initiala si cele obtinute anterior! (imaginile initiale sunt cele la rezolutie mare: 1600x1200, micsorate aici, nu scalate in Matlab!). De asemenea, imaginile alese sunt destul de "heavy", pentru imagini cu 1 singur obiect (sau mai multe, departate unul de celalalt) functioneaza mai bine.