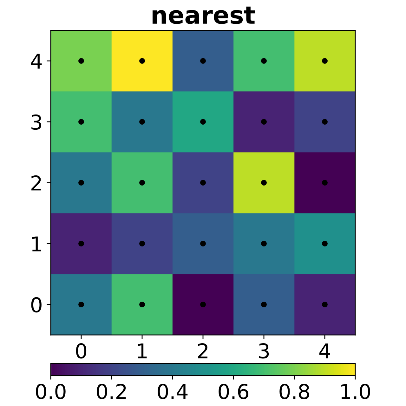
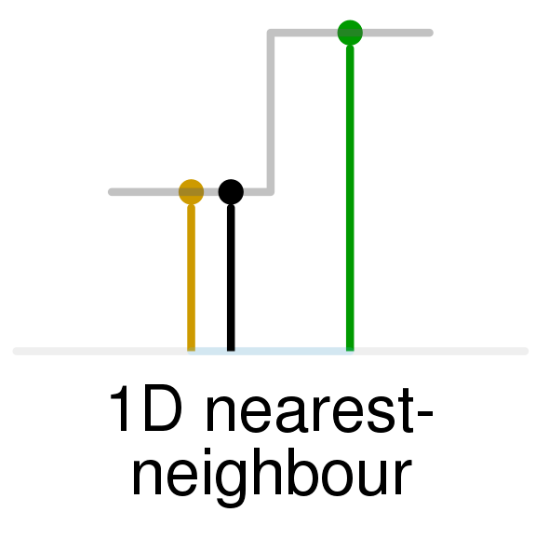
**Povecanje rezolucije**

Svaka slika ima svoju rezoluciju, njenu sirinu i visinu u pixelima. Prilikom promene njene rezolucije, dolazi do manjka ili viska pixela, pa se oni moraju izbaciti ili dodati. Dok izbacivanje pixela ne trazi dodatne podatke, kod povecanja rezolucije dolazi do nedostatka informacija. Zadatak sledeca 3 algoritma je da nadomeste te informacije i time popune rupe na slici.

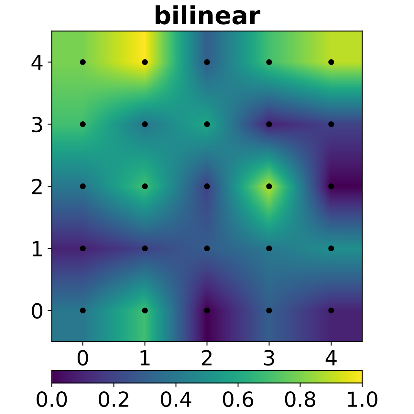
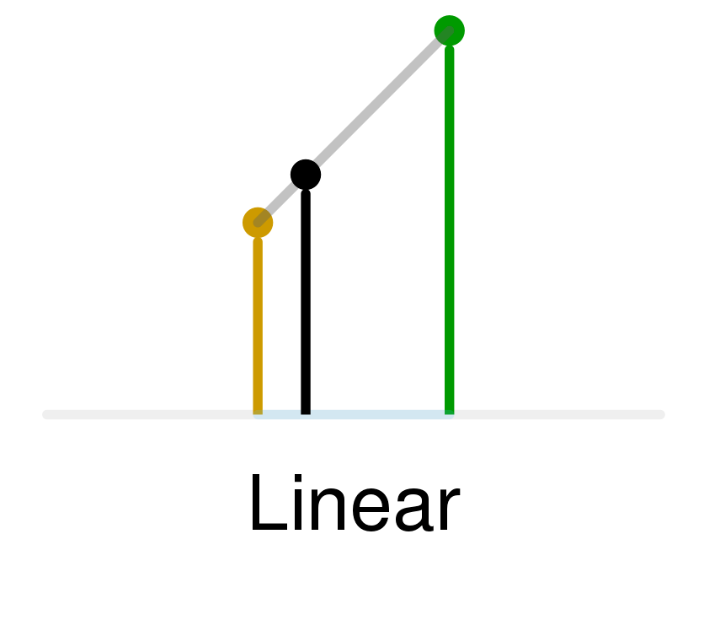
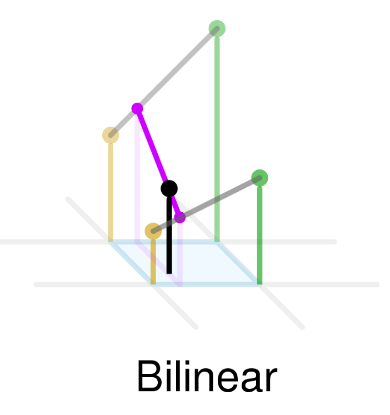
**Nearest Neighbor interpolacija**

Prvi nacin povecavanja rezolucije je najjednostavniji. On ne generise nikakve dodatne informacije, nego za novu sliku koristi samo originalne pixele. Za svaki pixel na novoj slici gleda se gde bi se nalazio na originalnoj slici i za svoju vrednost uzima vrednost najblizeg pixela.

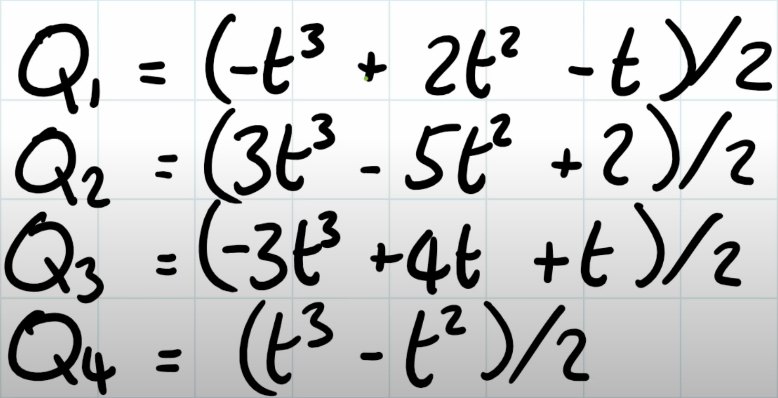
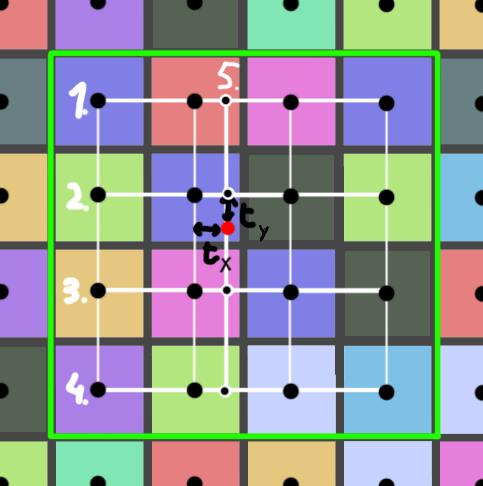
**Bilinearna interpolacija**

Sledeci algoritam je malo slozeniji I za razliku od provg generise nove informacije. Ponovo gleda poziciju novog pixela na staroj slici. Uzimaju se vrednosti 4 temena (pixela) kvadrata u kome se nalazi. Prvo se linearno interpolira izmedju dve gornje, pa izmedju dve donje tacke, pa na kraju izmedju 2 dobijene vrednosti. Time se dobija vrednost novog pixela.

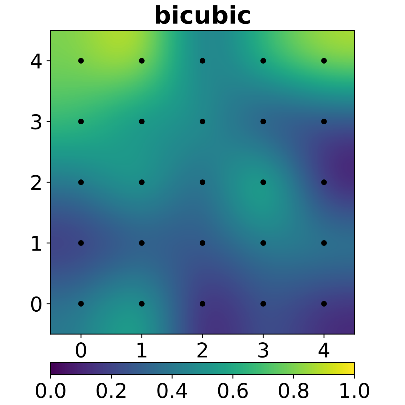
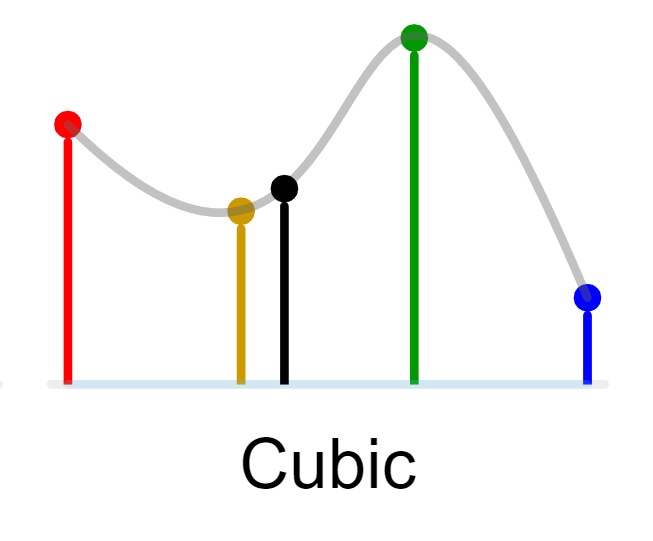
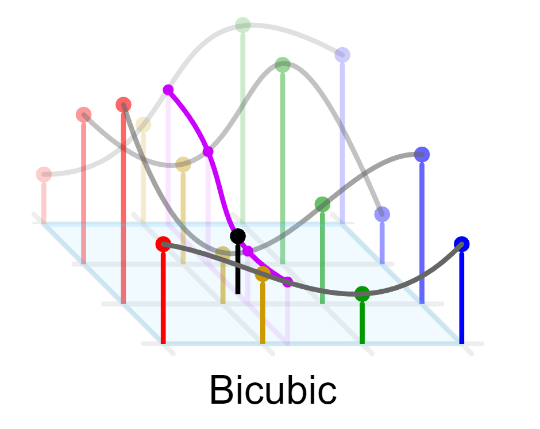
**Bikubna interpolacija**

Poslednji algoritam je najslozeniji. Umesto 4 uzima vrednosti 16 tacaka – 4 puta 4 kvadrat pixela. Nako toga, kubno interpolira prvu cetvorku pixela koju pripadju istoj x osi koristeci catmull-rom krivu. Vrednost tacke koja se dobija ovom interpolacijom je zbir 4 vrednosti, vrednosti svakog od pixela pomnozenog sa svojim koeficijentom, a oni su:

,gde je t rastojanje izmedju koordinate novog pixela na staroj slici i pixela na originalnoj slici koji se nalazi levo od njega.

Zatim ide druga, treca i cetvrta interpolacija. Interpoliraju se 4 dobijene vrednsoti, ali se sada za t uzima rastojanje od gornjeg pixela, i time imamo vrednost novog pixela.

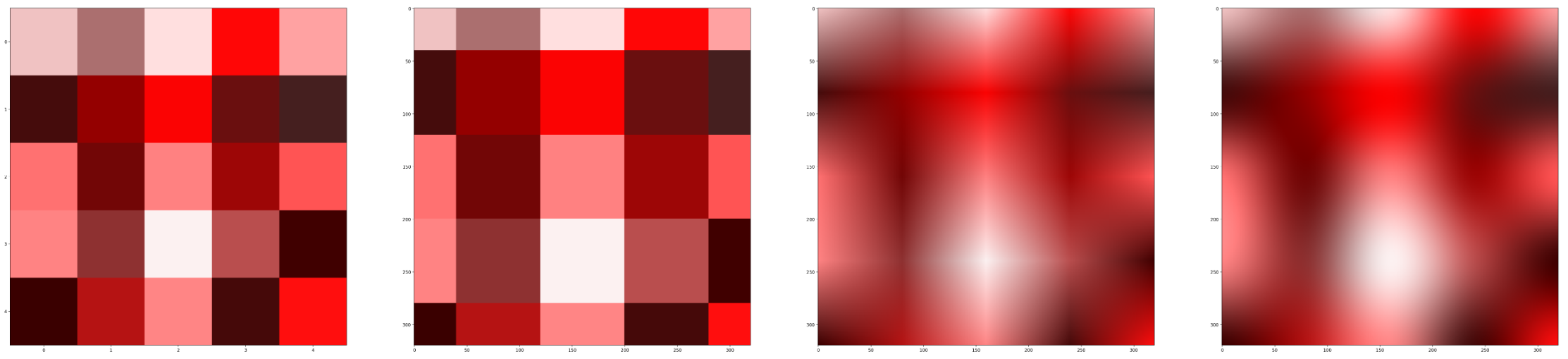
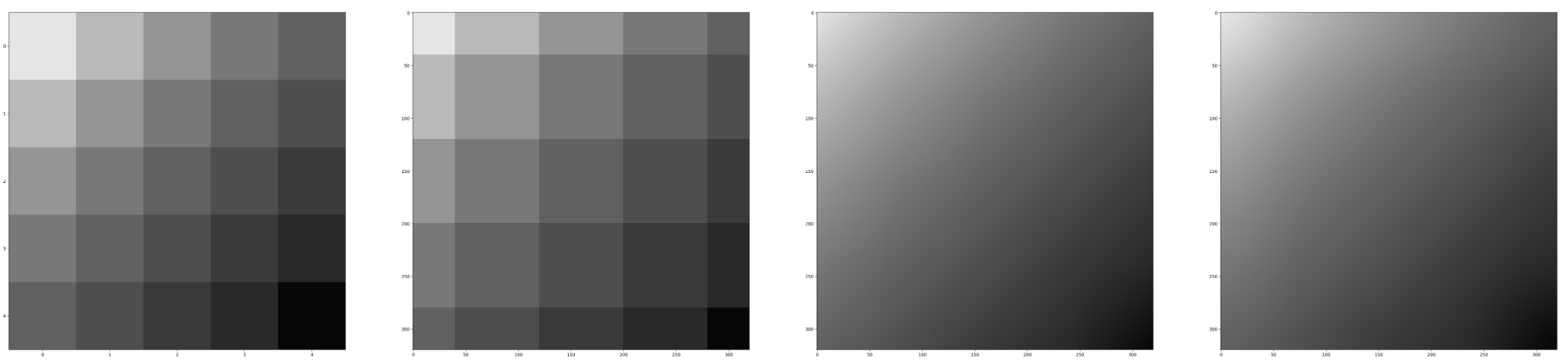
  

**Rezultati**

Sledi prikaz povecavanja reazolucije slike 512x512 pixela 4 puta.

Povecavanje iste slike koja je smanjena na 128x128, na originalnu vrednost 512x512, tj 4 puta.

Povecavanje rezolucije sivog i crveneg gradijenta 64 puta.

Original Nearest Neighbor Bilinear Bicubic 

Na prvom primeru je teze videti razliku zato sto je slika velike rezolucije, medjutim na drugom primeru se lakse vidi razlika izmedju 3 nacina povecavanja. Ako dodatno zumiramo slike iz drugog primera:

Sada se moze primetiti velika razlika izmedju prvog i druga dva algoritma. Nearest neighbor interpolacija samo povecava broj pixela i ne dodaje nista novo slici. Razlika izmedju bilinearne i bikubne interpolacije je manje ocigledna, ali se i dalje mogu videti razlike. Ako pogledamo nos, na bilinearnoj se mogu videti vertikalne linije, dok kod bikubne toga nema.

Primer sivog i crvenog gradijenta sluzi da bi se te razlike u specificnim slucajevima lakse uocile. Kod sivog nema prevelike razlike, ali kod crvenog je stvar mnogo ociglednija. Kod bilinearnog algoritma se formiraju horizontalne i vertikalne linije. Ovo je posledica linearnog interpoliranja izmedju objekata sa istim x ili y korodinatama, a znamo da su pixeli rasporedjeni bas na taj nacin. Bikubna interpolacija zato uzima u obzir 4 puta vise tacaka i time resava taj problem.

Medjutim, bolji algoritam dolazi sa svojom cenom. Dok je prvom(nearest neighbor) za povecavanje iz prvog primera bilo potrebno 3.72 sekunde, bilinearnom je bilo potrebno 70.58, a bikubnom cak 113.78 sekundi. U optimizovanom okruzenju ovaj broj je mnogo manji, ali zato mozemo reci da je bilinearni sporiji 19, a bikubni 30 puta.

**Zakljucak**

Iako druga dva algoritma daju mnogo bolje rezultate, za slike velike rezolucije nekada nisu potrebni. Naravno, u slucaju da zelimo bolji rezultat moramo platiti vremenom. Prvi algoritam je takodje bolji za uvelicavanje slika jako male rezolucije ako ne zelimo da je “zamutimo”. Postoje jos kompleksniji nacini za uvelicavanje, gde neki upotrebljavaju vestacku inteligenciju.

awd